Aligning and Comparing Vision Representations to Improve Understanding and Performance

Thesis by
Neehar Kondapaneni

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy in Computation and Neural Systems

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2025
Defended May 27, 2025

Neehar Kondapaneni
ORCID: 0000-0002-2782-3977

ACKNOWLEDGEMENTS

To my labmates, Grant Van Horn, Oisin Mac Aodha, Matteo Ronchi, Joseph Marino, Mason McGill, Serim Ryou, Jennifer J. Sun, Tony Zhang, Sara Beery, and Elijah Cole, Kevin Le, Suzanne Stathatos, Laure Delisle, Markus Marks, Manuel Knott, Rogério Guimarães, Raphi Kang, Anna Ding, Madison Van Horn, Sevan Brodjian, Michael Hobley, and Atharva Sehgal: thank you for welcoming me, being great friends and mentors, and all of the fun memories.

To the 355D pod: I couldn't have asked for a more fun group to grind with.

To Caroline Murphy: thank you for making my life so much simpler.

To Markus Marks: you taught me to be optimistic, perhaps the most important skill I have learned as a researcher.

To Oisin Mac Aodha: thank you for your steady mentorship and friendship; you have helped me improve as a researcher each and every year. You always ask the tough questions, forcing me to think more deeply. I hope I'll be able to come up with these questions for myself going forward.

To Pietro Perona: thank you for letting me join your research group. I have met wonderful people and learned interesting things here. Thank you, specifically, for teaching me how to think about the bigger picture both in research and in life.

To Takaki Komiyama, Hiroshi Makino and Pamela Reinagel: thank you for letting me have my first research experiences and giving me space to be creative.

To all of my friends: Andrew Hua, Mazden Mastromauro, Victor Rodrigo, Eric Hytönen, Migal Manickaraj, Brandon Toy, Zachery Iton, Taeho Kim, Prithvi Akella, Tomi Esho, Salvador Gomez, Eric Ocegueda, Alexandre de Faveri, David Garrett and Marina Lecoeuche: thank you all for your friendship. I have so many happy memories with you all. A special thank you to Manu, Eric, Migal, Brandon, and Taeho: thank you all for the memories, thoughtful conversations, and support through my injuries.

To my family – Mom, Dad, Manu, Pedamma, Pedananna, Ammamma, Tata, Amit and Meena: thank you so much for the unwavering support and love. You all have taught me important life lessons and helped me become who I am today.

To Meera Prasad, thank you for the encouragement and your positivity. These last three years would not have been the same without you.

ABSTRACT

Recent advances in large artificial intelligence (AI) models have enabled these models to perform a wide range of real-world tasks with skill levels comparable to or surpassing those of humans. In this thesis, we develop methods to compare, analyze, and align data representations from these powerful models. In Part 1, we develop methods for estimating human knowledge during a learning task and for comparing various data representations. These methods are steps towards a system designed to help us learn from AI. In Part 2, we show how aligning models can be useful in two separate domains. First, we discover and fix a misalignment in the inputs to a powerful foundation model and show how it improves performance. Second, we show that biologically inspired object manipulation tasks can be used as a training signal for learning human-aligned representations of number. Our results demonstrate the potential for alignment and comparison methods to improve the overall performance of AI models, improve our understanding of biological intelligence, and help us discover new patterns in the natural world.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Neehar Kondapaneni et al. "Representational Difference Clustering." In: *ICLR 2025 Workshop on Bidirectional Human-AI Alignment*. URL: https://openreview.net/forum?id=OxhHnD9wuZ.
N.K. participated in developing the idea, writing the manuscript, and creating figures. N.K. also wrote the code and ran the experiments.

[2] Neehar Kondapaneni, Oisin Mac Aodha, and Pietro Perona. "Representational Difference Explanations." In: *arXiv preprint arXiv:2505.23917* (2025). URL: https://arxiv.org/abs/2505.23917.
N.K. participated in developing the idea, writing the manuscript, and creating figures. N.K. also wrote the code and ran the experiments.

[3] Neehar Kondapaneni, Oisin Mac Aodha, and Pietro Perona. "Representational Similarity via Interpretable Visual Concepts." In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=ih3BJmIZbC.
N.K. participated in developing the idea, writing the manuscript, and creating figures. N.K. also wrote the code and ran the experiments.

[4] Markus Marks et al. "A closer look at benchmarking self-supervised pre-training with image classification." In: *International Journal of Computer Vision* (2025), pp. 1–13. URL: https://link.springer.com/article/10.1007/s11263-025-02402-w.
N.K. participated in writing the manuscript and creating figures.

[5] Neehar Kondapaneni and Pietro Perona. "A number sense as an emergent property of the manipulating brain." In: *Scientific Reports* 14.1 (2024), p. 6858. URL: https://www.nature.com/articles/s41598-024-56828-2.
N.K. participated in formulating the idea and experiments, writing the manuscript and conceptualizing figures. N.K. also wrote the code and created the figures.

[6] Neehar Kondapaneni et al. "Less is more: Discovering concise network explanations." In: *ICLR 2024 Workshop on Representational Alignment*. 2024. URL: https://openreview.net/forum?id=JBwpD6Yy8Q.
N.K. participated in developing the idea, writing the manuscript, and creating figures. N.K. also wrote the code and ran the experiments.

[7] Neehar Kondapaneni et al. "Text-image alignment for diffusion-based perception." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 13883–13893. URL: https://openaccess.thecvf.com/content/CVPR2024/html/Kondapaneni_Text-Image_Alignment_for_Diffusion-Based_Perception_CVPR_2024_paper.html.

N.K. participated in developing the idea, writing the manuscript, and creating figures, writing the code, and running the experiments.

[8] Neehar Kondapaneni, Pietro Perona, and Oisin Mac Aodha. "Visual Knowledge Tracing." In: *European Conference on Computer Vision*. Springer. 2022, pp. 415–431. URL: https://link.springer.com/chapter/10.1007/978-3-031-19806-9_24.
N.K. participated in developing the idea, writing the manuscript, and creating figures. N.K. also wrote the code and ran the experiments.

CONTENTS

LIST OF FIGURES

Number *Page*

LIST OF TABLES

Number *Page*

Chapter 1

INTRODUCTION

Intelligence depends on information-rich and organized representations of signals from the real world. In biology, organisms receive sensory input, compress it, and structure it for future use. Similarly, artificial intelligences (AIs) rely on compressed representations of raw data that enable them to perform complex tasks. The past few years have seen remarkable progress in AI, driven by advances in datasets, model architectures, and training methods. Yet, despite this progress, our understanding of a model's internal representations remains surprisingly limited. Richard Feynman once said, "What I cannot create, I do not understand." Amusingly, AI research seems to have embraced a different principle: we regularly create things we do not fully understand. This is not a new phenomenon — humans have long relied on intuition to make progress. For instance, farmers successfully bred livestock for desirable traits centuries before the discovery of genetics. However, it is clear that understanding has benefits: it allows us to be more efficient with our resources and better anticipate possible failures. For example, due to our knowledge of genetics, we have modified plants to be more hardy and cured diseases at their root causes.

Data, like images, videos, text, audio, etc., capture signals that exist in our world. These signals contain redundancy, for example, in a landscape image, many pixels will capture the "sky." Rather than storing all of this redundant information, we compress the raw data into concise formats that (ideally) contain the information we care about. These compressed formats are called *data representations*. This thesis presents research into comparing and aligning data representations to help uncover how both humans and models process visual scenes. The following section provides background on how these approaches can be used to improve our understanding of model behavior, to improve model performance, and build a better understanding of biological vision representations. Finally, I present the motivation and overview for the chapters of this thesis.

1.1 Aligning Humans and Models for Improving Model Understanding

AI has become part of workflows in science [1, 2, 3], software engineering [4, 5], law [6, 7], etc. As it becomes deeply integrated into core systems in our society, we must ask if it is well-aligned with our goals as a community. This concern was famously expressed by Norbert Wiener in 1960:

> "If we use, to achieve our purposes, a mechanical agency with whose operation we cannot efficiently interfere. . . , we had better be quite sure that the purpose put into the machine is the purpose which we really desire."

Broadly speaking, there are two paths towards aligning humans and AI models:

- **Models are trained to mimic human representations and/or behavior.** One of the key drivers behind the adoption of large language models (LLMs) in society are training approaches that have been developed to align model outputs to human preferences [8, 9, 10, 11]. These methods are designed to encourage LLMs to generate content that is safe, factually accurate, and reliable, thereby fostering greater user trust in human–AI interactions.
- **Humans learn to understand model behavior.** A second approach to alignment focuses on helping humans better understand complex models, effectively aligning human mental representations to the model's representation [12, 13, 14, 15]. This line of research aims to generate explanations of model behavior by analyzing how a model transforms different inputs to arrive at an output. The next section explores why learning from AI is valuable, and outlines the methods that enable us to do so effectively.

1.2 Learning from AI

AI models have already surpassed humans in several domains, such as playing chess or go [16], driving cars safely [17], recognizing species in images [18, 19]. In the coming years, they are likely to exceed human performance in many more. In these cases, I argue that it is more valuable for humans to learn from the model's knowledge than for developers to constrain the model to behave like a human. This perspective underlies several chapters of this thesis, which are motivated by the goal of enabling humans to learn from AI, specifically in the context of categorizing visual scenes. In my view, there are three components to effective systems for teaching humans.

1. **Estimation of student knowledge.** In the educational system, this takes the form of conversation, quizzes, and tests.

2. **Explanations that effectively communicate knowledge.** In the educational system, this takes the form of lectures, textbooks, tutorials, assignments, etc. Ideally, explanations are personalized for each student, so that they are targeted to weaknesses in the student's knowledge.

3. **Curricula to lead students towards expert knowledge.** In the educational system, this takes the form of teachers using their experience to inform the sequencing of course material, essentially the design of the syllabus.

Effective teaching systems require a feedback loop from all three components, student knowledge informs the design of curricula, student knowledge requires the adjustment of explanations, and both curricula and explanations improve student knowledge. All of the components to build an effective teaching system have parallels in the machine learning literature, but they have not reached a level of maturity at which they can be combined into a single effective teaching system for visual categorization. I will expand on each in the following subsections.

1.2.1 Knowledge Tracing

Knowledge tracing methods aim to track the knowledge of a student as they learn new material. These methods take a history of questions and student responses as input and tries to predict the student's future responses. Initially, these methods were Bayesian models [20], but they have given way to deep learning based approaches [21]. Recently, pre-trained LLMs have also been shown to be effective at tracing student knowledge [22, 23]. While reasonably large datasets exist for knowledge tracing in text based learning settings, like math, science, statistics, etc. [24, 25], large datasets do not yet exist for visual learning.

1.2.2 Explainable AI (XAI)

Explainable AI (XAI), interpretability, mechanistic interpretabilty, etc. are all common names for methods that aim to break down model behavior in human understandable ways, ie., explanations. XAI methods are grouped into two broad categories:

Intrinsic XAI. Intrinsic XAI methods modify model architectures or behavior to enforce that the model uses an interpretable reasoning process to arrive at a decision.

For example, concept bottleneck models ask models to predict the presence of a concept as an intermediate step to making a decision [26], visual programming methods ask model's to generate executable code that can be directly inspected [27, 28, 29], and B-cos networks enforce that neuron activations to align with specific inputs [30]. However, in general, intrinsic XAI methods tend to reduce model performance by introducing extra constraints [26, 30]. This is not ideal if the goal is to understand model behavior that exceeds human performance.

Post-hoc XAI. The majority of XAI methods are post-hoc XAI methods. These methods try to analyze existing models without introducing any constraints on their training procedure. Broadly, there are two categories of post-hoc XAI methods:

- **Local Methods.** [31, 32, 33] These methods aim to attribute a model's output to a specific part of the input. For example, if a model decided that an image was of a dog, what part of the image did it use to arrive at the decision? It could be the face, tail, body, background, etc.
- **Global Methods.** [14, 34, 35, 36, 37, 38, 39] These methods aim to convey a "global" understanding of the types of features that the model has discovered in the data. Many global methods are a form of dictionary learning. Dictionary learning methods aim to decompose model outputs into a set of interpretable, core "concepts" and measure how much each of these concepts contributed to each model's decision.

Some recent methods combine both local and global approaches to generate more detailed explanations for users [12, 40, 41, 42].

Concept-based XAI. Concept-based XAI methods can be used as a post-hoc XAI method and have had some success in human evaluations [12, 14, 41]. These methods assume models learn human understandable *concepts*, sets of inputs that contain some common characteristic. To extract these concepts, model embeddings are decomposed into a linear combination of k concept embeddings and concept coefficients. The concept embeddings and coefficients are learned using one of many dictionary-learning approaches [13, 36]. Once learned, the concepts are visualized by sampling the inputs that have the largest coefficients for each concept [12, 13, 36].

Evaluation. Hundreds of XAI methods have been developed, but evaluations show that there is significant room for improvement. We have found that XAI methods can

help humans understand models a little better in specific situations, but they can also confuse users [12, 41, 43, 44, 45]. For example, Kim et al. found that users were more likely to trust a model provided any explanation, even if the explanation was for an incorrect prediction, suggesting that the explanations are not precise enough to be trusted. This is related to a critical challenge in explainable AI: the trade-off between generating explanations that are simple enough to be comprehensible to users and those that are faithful to the model's internal computations. In practice, explanation algorithms often favor simplicity, which may obscure the actual mechanisms driving predictions. One possible solution to this issue is to introduce curricula that is designed to slowly close the gap between the user's knowledge and human knowledge. While this has not yet been completely realized, there is some evidence that this may be the case. Recently, Schut et. al showed that explanations for a neural network chess model could help grandmasters (chess experts) improve their play, but these same explanations would be too complex to interpret for a layperson [46].

1.2.3 Teaching Systems

Methods for automating the teaching of categorization tasks have been explored. Work in this space is called *machine teaching* for humans. Generally, machine teaching algorithms select a model for the student, a learning algorithm for the student, and then develop an algorithm to optimize the learning of the student [47]. Many algorithms developed for this task assume that they have knowledge of the learner model [48, 49]. However, the problem quickly becomes difficult under more realistic conditions, such as limited access to the learner's current knowledge state [50]. Despite these challenges, several algorithms have been successful for teaching visual categories, although they often rely on strong assumptions about the learner's behavior [51, 52, 53]. A central obstacle in designing algorithms to teach humans lies in evaluation: assessing teaching effectiveness typically requires costly and complex human-subject experiments. For example, evaluations of these algorithms bring up questions similar to the ones in evaluating explainability methods: How much did the student know initially? How long should we interact with the student? Is the material even learnable? What kind of user interface is appropriate for this experiment? These questions make developing and evaluating machine teaching methods challenging.

1.3 Aligning Representations for Improving Model Performance

Recently, researchers have developed “foundation" models, large vision, language, and multi-modal models that have impressive performance and generalize to many established tasks [2, 18, 19, 54, 55, 56, 57]. The training procedures for these models are often costly and difficult to reproduce, as they make use of very large models, massive datasets, and costly, specialized infrastructure. Rather than re-training new models from scratch, researchers have been adapting and combining foundation models to solve complex tasks. When combining two or more models, it is critical that the outputs of one model are aligned well with the inputs of the next model in order to maximize model performance. These efforts have been valuable and improved model performance on important tasks like image generation [58, 59], image segmentation [1, 60], visual question answering [27, 28, 29], etc.

In a separate line of research, certain types of alignment to human representations has been observed to improve model generalization and performance on classification tasks [61, 62]. Although not an explicit goal, CLIP [18], a powerful foundation model that shows impressive generalizability on an array of classification tasks, also aligns model representations to human knowledge by aligning textual captions for visual data [63].

1.4 Aligning Representations for Modeling Human Visual Psychology

The human brain is effectively a black-box, it’s internal mechanisms are both difficult to understand and hard to observe. Thus, a common approach to studying the human brain is to thoughtfully select an interesting set of stimuli, present it to the brain, and carefully measure the resulting outputs.

There are several “levels" at which one can conduct this kind of experiment. For example, at the neuronal level, patch clamp recordings have been used to understand the behavior of ion channels in a neuron [64]. At a systems level, optogenetics has been used to perturb and record the behavior of biological neural networks [65]. Finally, at a cognitive level, psychophysics has developed careful stimuli to understand the psychology of the human brain [66].

In particular, *visual* psychophysics [67] investigates human visual psychology by measuring how humans respond to specific visual stimuli. Results from this line of research have revealed when humans are able to detect signals from noise [68], the likelihood that humans will notice a minor difference in stimuli [66], the ability of humans to estimate the number of objects in a visual scene with only a quick

glance [69], etc. Of course, the natural follow-up question is: How do these phenomena arise mechanistically? Biological experiments can be costly and invasive, thus, there is great interest in *aligned* models of the human visual system that can be used to guide experimentation and generate hypotheses about how the brain operates [70, 71, 72]. Interestingly, both the process of aligning a model and the aligned model itself are valuable. The methods used to align a model tell us about what data [73], model architectures [74, 75, 76], learning signal [77, 78, 79], etc. are possibly relevant in how humans learn to perceive the world. The model itself can be used in experiments that can generate hypotheses about how the human visual system may react to certain stimuli [80, 81, 82, 83].

1.5 Thesis Motivation and Organization

This thesis is organized into three parts. In Part 1, I present work motivated by the goal of learning from vision models. In Chapter 2, I introduce a method for tracing student knowledge on visual categorization tasks. In this study, we collected a dataset in which students were shown an image and asked to predict its category. As feedback, the students received the correct category name. Upon completing this project, it became clear that (1) collecting data in this manner was too costly and (2) richer forms of feedback would be necessary to teach visual categories effectively. This realization, combined with the emergence of models surpassing human performance in visual categorization, shifted my focus toward the challenge of generating explanations that could help humans learn patterns discovered by the vision model. Concept-based explainable AI (XAI) methods (Sec. 1.2.2) have been widely used to extract interpretable concepts from model representations. However, modern models learn thousands of concepts, making it difficult for users to navigate these explanations and understand how each concept contributes to overall performance. A core principle of effective teaching is identifying what a student does not yet understand. Ideally, we would like to filter model concepts so that we show the human student features they have not yet discovered. One natural approach is to compare human and model representations to identify concepts unique to the model. However, collecting the data needed to estimate human representations is costly. As a proxy for this ultimate goal, I decided to develop methods that could compare two model representations and show us concepts that were unique to each model. Specifically, in Chapter 3, I describe an approach that compares model representations using visual concepts. This method extracts concepts for each model independently and compares the concepts to help users gain insight into conceptual

differences between models. Our results revealed interesting and subtle differences between models that could be linked to differences in model performance. However, we found that the explanations could be challenging to interpret. In Chapter 4, we conduct experiments that identify three issues with using concept-based XAI for comparison:

1. **Independent concept extraction leads to overlapping concepts.** When concept extraction is performed separately for each model, especially if the models are similar, the resulting concepts tend to overlap. This overlap can result in nearly identical explanations that fail to highlight meaningful differences between model representations.

2. **Sampling images with the maximum coefficients can hide model differences.** When representations and concepts are similar, sampling the inputs with the maximum coefficients can hide subtle differences between concept coefficients on the rest of the dataset. These concept coefficient differences are critical for understanding how the two models differ in performance.

3. **Linear combinations of concepts are uninterpretable.** Asking users to mentally perform a weighted linear combination over concept explanations to understand how a model represents an input places an unreasonable cognitive burden on the user. In practice, this leads to vague or inaccurate interpretations, particularly when comparing how different models represent an input.

These insights motivated the design of a new method for comparing model representations. Unlike our previous approach, our method jointly analyzes both representations to generate targeted explanations that focus only on differences. It avoids visualizing samples with the maximum coefficients and avoids using linear combinations, leading to outputs that are more intuitive and easier to interpret. As we show in Chapter 4, this new approach produces clearer comparisons and more actionable insights into how model representations differ. In the future, I believe it will be possible to directly compare human representations to model representations allowing us to efficiently discover gaps in human knowledge. In Chapter 5, I explore several possible paths toward realizing this goal.

In Part 2, I present two applications of aligning representations. In Chapter 6, we discover a misalignment between the text and image inputs that are passed to a powerful multi-modal model. After correcting this misalignment, we observe significant

performance improvements, and achieve state-of-the-art (SOTA) performance in several challenging tasks. In Chapter 7, we develop a biologically inspired learning framework for exploring number perception in vision systems. We hypothesize that a children's interaction with the world, namely, the way it manipulates objects, has a critical role in developing and organizing mental representations of number. We find that training a model with our proposed learning framework results in a human-like representation that can reproduce several known psychophysical properties of human number perception.

Part 3 contains the conclusion and discusses directions for future work.

References

[1] Uriah Israel et al. "A foundation model for cell segmentation." In: *arXiv preprint arXiv:2311.11004* (2023).

[2] Alexander Kirillov et al. "Segment anything." In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 4015–4026.

[3] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold." In: *nature* 596.7873 (2021), pp. 583–589.

[4] Windsurf Team. *Windsurf*. `https://windsurf.com/`. Accessed: 2025-05-07. 2024.

[5] Cursor Team. *Cursor: The AI Code Editor*. `https://www.cursor.com/en`. Accessed: 2025-05-07. 2024.

[6] Lex Machina. *Lex Machina Legal Analytics*. `https://www.lexisnexis.com/en-us/products/lex-machina.page`. Accessed: 2025-05-08. 2025.

[7] Bloomberg Law. *AI in Legal Practice: Explained*. `https://pro.bloomberglaw.com/insights/technology/ai-in-legal-practice-explained/`. Accessed: 2025-05-08. 2025.

[8] Tianhao Shen et al. "Large language model alignment: A survey." In: *arXiv preprint arXiv:2309.15025* (2023).

[9] Shengxiong Ouyang et al. "Pseudo-label generation-evaluation framework for cross domain weakly supervised object detection." In: *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2021, pp. 724–728.

[10] Yang Liu et al. "Trustworthy llms: a survey and guideline for evaluating large language models' alignment." In: *arXiv preprint arXiv:2308.05374* (2023).

[11] Yuntao Bai et al. "Training a helpful and harmless assistant with reinforcement learning from human feedback." In: *arXiv preprint arXiv:2204.05862* (2022).

[12] Thomas Fel et al. “CRAFT: Concept recursive activation factorization for explainability.” In: *CVPR*. 2023.

[13] Hoagy Cunningham et al. “Sparse autoencoders find highly interpretable features in language models.” In: *ICLR*. 2024.

[14] Been Kim et al. “Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TACV).” In: *ICML*. 2018.

[15] Lisa Schut et al. “Bridging the human-ai knowledge gap: Concept discovery and transfer in alphazero.” In: *arXiv preprint arXiv:2310.16410* (2023).

[16] David Silver et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.” In: *Science* 362.6419 (2018), pp. 1140–1144. DOI: 10.1126/science.aar6404.

[17] Kristofer D Kusano et al. “Comparison of Waymo rider-only crash data to human benchmarks at 7.1 million miles.” In: *Traffic Injury Prevention* 25.sup1 (2024), S66–S77.

[18] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision.” In: *International Conference on Machine Learning* (2021), pp. 8748–8763. (Visited on 08/21/2023).

[19] Maxime Oquab et al. “Dinov2: Learning robust visual features without supervision.” In: *arXiv preprint arXiv:2304.07193* (2023).

[20] Albert T Corbett and John R Anderson. “Knowledge tracing: Modeling the acquisition of procedural knowledge.” In: *User modeling and user-adapted interaction* 4 (1994), pp. 253–278.

[21] Chris Piech et al. “Deep knowledge tracing.” In: *Advances in neural information processing systems* 28 (2015).

[22] Unggi Lee et al. “Language model can do knowledge tracing: Simple but effective method to integrate language model and knowledge tracing task.” In: *arXiv preprint arXiv:2406.02893* (2024).

[23] Seyed Parsa Neshaei et al. “Towards modeling learner performance with large language models.” In: *arXiv preprint arXiv:2403.14661* (2024).

[24] Douglas Selent, Thanaporn Patikorn, and Neil Heffernan. “Assistments dataset from multiple randomized controlled experiments.” In: *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*. 2016, pp. 181–184.

[25] Youngduck Choi et al. “Ednet: A large-scale hierarchical dataset in education.” In: *Artificial Intelligence in Education: 21st International Conference, AIED 2020, Ifrane, Morocco, July 6–10, 2020, Proceedings, Part II 21*. Springer. 2020, pp. 69–73.

[26] Pang Wei Koh et al. “Concept bottleneck models.” In: *International conference on machine learning*. PMLR. 2020, pp. 5338–5348.

[27] Damiano Marsili et al. *Visual Agentic AI for Spatial Reasoning with a Dynamic API*. 2025. arXiv: 2502.06787 [cs.CV]. URL: https://arxiv.org/abs/2502.06787.

[28] Dídac Surís, Sachit Menon, and Carl Vondrick. "Vipergpt: Visual inference via python execution for reasoning." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 11888–11898.

[29] Tanmay Gupta and Aniruddha Kembhavi. "Visual programming: Compositional visual reasoning without training." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 14953–14962.

[30] Moritz Böhle, Mario Fritz, and Bernt Schiele. "B-cos networks: Alignment is all we need for interpretability." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10329–10338.

[31] Ramprasaath R Selvaraju et al. "Grad-CAM: visual explanations from deep networks via gradient-based localization." In: *IJCV* (2020).

[32] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should I trust you?" Explaining the predictions of any classifier." In: *KDD*. 2016.

[33] Scott M. Lundberg and Su-In Lee. "A unified approach to interpreting model predictions." In: *NeurIPS*. 2017.

[34] Amirata Ghorbani et al. "Towards automatic concept-based explanations." In: *NeurIPS* (2019).

[35] Ruihan Zhang et al. "Invertible concept-based explanations for cnn models with non-negative concept activation vectors." In: *AAAI*. 2021.

[36] Thomas Fel et al. "A holistic approach to unifying automatic concept extraction and concept importance estimation." In: *NeurIPS* (2023).

[37] Matthew Kowal et al. "Understanding Video Transformers via Universal Concept Discovery." In: *CVPR*. 2024.

[38] Eleonora Poeta et al. "Concept-based explainable artificial intelligence: A survey." In: *arXiv:2312.12936* (2023).

[39] David Bau et al. "Understanding the role of individual units in a deep neural network." In: *PNAS* (2020).

[40] Jessica Schrouff et al. "Best of both worlds: local and global explanations with human-understandable concepts." In: *arXiv:2106.08641* (2021).

[41] Reduan Achtibat et al. "From attribution maps to human-understandable explanations through concept relevance propagation." In: *Nature Machine Intelligence* (2023).

[42] Neehar Kondapaneni et al. "Less is more: Discovering concise network explanations." In: *ICLR 2024 Workshop on Representational Alignment*. 2024. URL: https://openreview.net/forum?id=JBwpD6Yy8Q.

[43] Peter Hase and Mohit Bansal. "Evaluating explainable AI: Which algorithmic explanations help users predict model behavior?" In: *arXiv preprint arXiv:2005.01831* (2020).

[44] Sunnie SY Kim et al. "HIVE: Evaluating the human interpretability of visual explanations." In: *European Conference on Computer Vision*. Springer. 2022, pp. 280–298.

[45] Subhash Kantamneni et al. "Are sparse autoencoders useful? a case study in sparse probing." In: *arXiv preprint arXiv:2502.16681* (2025).

[46] Christoph Schuhmann et al. "LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs." In: *arXiv preprint arXiv:2111.02114* (2021). (Visited on 08/29/2023).

[47] Xiaojin Zhu. "Machine teaching: An inverse problem to machine learning and an approach toward optimal education." In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 29. 1. 2015.

[48] Adish Singla et al. "Near-optimally teaching the crowd to classify." In: *ICML*. 2014, pp. 154–162.

[49] Anette Hunziker et al. "Teaching multiple concepts to a forgetful learner." In: *NeurIPS* (2019).

[50] Weiyang Liu et al. "Towards black-box iterative machine teaching." In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3141–3149.

[51] Edward Johns, Oisin Mac Aodha, and Gabriel J Brostow. "Becoming the expert-interactive multi-class machine teaching." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2616–2624.

[52] Oisin Mac Aodha et al. "Teaching categories to human learners with visual explanations." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3820–3828.

[53] Pei Wang, Kabir Nagrecha, and Nuno Vasconcelos. "Gradient-based algorithms for machine teaching." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1387–1396.

[54] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022), pp. 10674–10685. (Visited on 07/27/2023).

[55] Josh Achiam et al. "Gpt-4 technical report." In: *arXiv:2303.08774* (2023).

[56] Junnan Li et al. "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models." In: *arXiv preprint arXiv:2301.12597* (2023). (Visited on 08/09/2023).

[57] Hugo Touvron et al. "Llama 2: Open foundation and fine-tuned chat models." In: *arXiv preprint arXiv:2307.09288* (2023).

[58] Rinon Gal et al. “An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion.” In: *arXiv preprint arXiv:2208.01618* (2022). (Visited on 08/21/2023).

[59] Nataniel Ruiz et al. “DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation.” In: *arXiv preprint arXiv:2208.12242* (2022). (Visited on 08/21/2023).

[60] Neehar Kondapaneni et al. “Text-image alignment for diffusion-based perception.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 13883–13893. URL: https://openaccess.thecvf.com/content/CVPR2024/html/Kondapaneni_Text-Image_Alignment_for_Diffusion-Based_Perception_CVPR_2024_paper.html.

[61] Ilia Sucholutsky and Tom Griffiths. “Alignment with human representations supports robust few-shot learning.” In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 73464–73479.

[62] Robert Geirhos et al. “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.” In: *International conference on learning representations*. 2018.

[63] Raja Marjieh et al. “Words are all you need? language as an approximation for human similarity judgments.” In: *arXiv preprint arXiv:2206.04105* (2022).

[64] Erwin Neher and Bert Sakmann. “Single-channel currents recorded from membrane of denervated frog muscle fibres.” In: *Nature* 260.5554 (1976), pp. 799–802.

[65] Edward S Boyden et al. “Millisecond-timescale, genetically targeted optical control of neural activity.” In: *Nature neuroscience* 8.9 (2005), pp. 1263–1268.

[66] Gustav Theodor Fechner. *Elemente der psychophysik*. Vol. 2. Breitkopf u. Härtel, 1860.

[67] Fergus W Campbell and John G Robson. “Application of Fourier analysis to the visibility of gratings.” In: *The Journal of physiology* 197.3 (1968), p. 551.

[68] Arthur E Burgess et al. “Efficiency of human visual signal discrimination.” In: *Science* 214.4516 (1981), pp. 93–94.

[69] W Stanley Jevons. “The power of numerical discrimination.” In: *Nature* 3.67 (1871), pp. 281–282.

[70] Martin Schrimpf et al. “Brain-score: Which artificial neural network for object recognition is most brain-like?” In: *bioRxiv* (2018).

[71] Daniel LK Yamins and James J DiCarlo. “Using goal-driven deep learning models to understand sensory cortex.” In: *Nature neuroscience* 19.3 (2016), pp. 356–365.

[72] Grace W Lindsay. "Convolutional neural networks as a model of the visual system: Past, present, and future." In: *Journal of cognitive neuroscience* 33.10 (2021), pp. 2017–2031.

[73] Bria Long et al. "The BabyView dataset: High-resolution egocentric videos of infants' and young children's everyday experiences." In: *arXiv preprint arXiv:2406.10447* (2024).

[74] Samanwoy Ghosh-Dastidar and Hojjat Adeli. "Spiking neural networks." In: *International journal of neural systems* 19.04 (2009), pp. 295–308.

[75] Ziming Liu, Eric Gan, and Max Tegmark. "Seeing is believing: Brain-inspired modular training for mechanistic interpretability." In: *Entropy* 26.1 (2023), p. 41.

[76] Ghislain St-Yves et al. "Brain-optimized deep neural network models of human visual areas learn non-hierarchical representations." In: *Nature communications* 14.1 (2023), p. 3329.

[77] Timothy P Lillicrap et al. "Random synaptic feedback weights support error backpropagation for deep learning." In: *Nature communications* 7.1 (2016), p. 13276.

[78] Benjamin Scellier and Yoshua Bengio. "Equilibrium propagation: Bridging the gap between energy-based models and backpropagation." In: *Frontiers in computational neuroscience* 11 (2017), p. 24.

[79] Neehar Kondapaneni and Pietro Perona. "A number sense as an emergent property of the manipulating brain." In: *Scientific Reports* 14.1 (2024), p. 6858. URL: https://www.nature.com/articles/s41598-024-56828-2.

[80] Ruixing Liang et al. "Unidirectional brain-computer interface: Artificial neural network encoding natural images to fMRI response in the visual cortex." In: *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2024, pp. 1851–1855.

[81] Thomas Langlois et al. "Passive attention in artificial neural networks predicts human visual selectivity." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 27094–27106.

[82] Guohua Shen et al. "Deep image reconstruction from human brain activity." In: *PLoS computational biology* 15.1 (2019), e1006633.

[83] Neehar Kondapaneni, Pietro Perona, and Oisin Mac Aodha. "Visual Knowledge Tracing." In: *European Conference on Computer Vision*. Springer. 2022, pp. 415–431. URL: https://link.springer.com/chapter/10.1007/978-3-031-19806-9_24.

Part I

Learning from AI

Chapter 2

VISUAL KNOWLEDGE TRACING

Neehar Kondapaneni, Pietro Perona, and Oisin Mac Aodha. "Visual Knowledge Tracing." In: *European Conference on Computer Vision*. Springer. 2022, pp. 415–431. URL: https://link.springer.com/chapter/10.1007/978-3-031-19806-9_24.

2.1 Abstract

Each year, thousands of people learn new visual categorization tasks — radiologists learn to recognize tumors, birdwatchers learn to distinguish similar species, and crowd workers learn how to annotate valuable data for applications like autonomous driving. As humans learn, their brain updates the visual features it extracts and attend to, which ultimately informs their final classification decisions. In this work, we propose a novel task of tracing the evolving classification behavior of human learners as they engage in challenging visual classification tasks. We propose models that jointly extract the visual features used by learners as well as predicting the classification functions they utilize. We collect three challenging new datasets from real human learners in order to evaluate the performance of different visual knowledge tracing methods. Our results show that our recurrent models are able to predict the classification behavior of human learners on three challenging medical image and species identification tasks.

2.2 Introduction

Humans excel at learning new concepts even when they have only received limited explicit supervision [2, 3]. Key to our success is our ability to extract informative and generalizable representations from the world around us and our ability to update these representations given relatively sparse feedback. This capacity, in turn, enables us to perform complex tasks such as spatial navigation and visual categorization with apparent ease.

Despite recent progress that has been made in computer vision in learning visual representations through self-supervision alone [4, 5, 6], large amounts of supervision are still required to make best use of the resulting features [7]. In light of this, it is important for us to better understand: (i) what are the properties that make

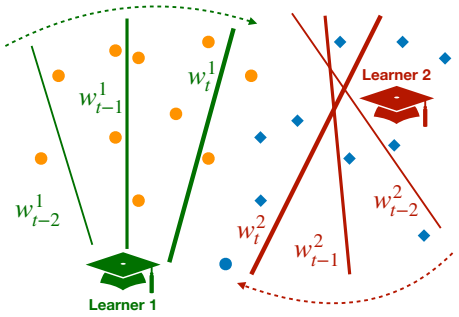


Figure 2.1: Overview. We model a learner as an evolving classifier in a learned feature space. We assume the feature space is static and, as the learner is presented with images and class labels over time, their internal classification function self-updates. Here, we illustrate this for three time-steps, for two learners, learning a binary classification task (orange versus blue).

representations learned by *humans* so effective, (ii) how are these representations learned, and (iii) can we predict the classification behavior of humans during learning? Our ultimate goal is to obtain better insight into how humans are such effective learners, which can then potentially inform new learning mechanisms for future artificial systems.

In order to attempt to address some of these questions, in this work we explore the problem of visual knowledge tracing. In the educational data mining community, knowledge tracing is the problem of monitoring and predicting the evolving knowledge state of a learner engaged in a learning task [8]. Recent work has applied advances in deep learning to knowledge tracing for question and answer-style text datasets and has investigated applications in domains such as mathematics education [9, 10, 11]. However, tracing the behavior of humans that are engaged in learning and performing challenging visual categorization tasks is underexplored. Most closely related is the work on deep metric learning that attempts to learn human aligned visual representations from sparse human annotations [12]. However, these works typically make simplifying assumptions, e.g. assuming the learners are not changing over time (i.e. they are 'static') or that the visual criteria used by the learners is the same across all learners. Instead, we explore a more challenging visual knowledge tracing setting where the learners are assumed to be non-stationary during learning, i.e. the visual features they use to perform the classification task at hand can, and likely do, change over time (see Fig. 2.1).

We present a recurrent neural network-based approach for visual knowledge tracing. Once trained, our models are capable of predicting the classification behavior of human learners that were not observed during training. The proposed models make use of the history of previous learner responses, images, and ground truth class labels in order to predict their future responses. Through experiments on three challenging image classification datasets we show that our models are superior to baseline approaches. Our models are capable of tracing the learning dynamics more

accurately than non-recurrent baselines. We make the following three contributions: (i) A new model for visual knowledge tracing that jointly estimates the visual features and per-time-step classification function used by non-stationary human learners. (ii) A new set of annotations for three benchmark evaluation datasets collected from humans engaged in learning challenging visual classification tasks. (iii) A detailed comparison of several visual knowledge tracing methods on these datasets.

2.3 Related Work

2.3.1 Metric Learning

The goal of metric learning is to learn perceptual embeddings such that distance in the lower dimensional embedding space encodes information related to semantic similarity. Pre-deep learning approaches to metric learning were primarily concerned with learning embeddings directly for each item in an input set. In the case of learning from human supervision, approaches that use relative similarity judgements have been shown to be effective [13, 14, 15, 16]. These methods have also been extended to the adaptive setting where the model can decide which items to request annotations for in order to speed up training [17].

More recently, end-to-end metric learning methods have attempted to parameterize an embedding function (e.g. a convolutional neural network) directly [18, 19]. As a result, they are able to embed any new item into the embedding space, even those not seen at training time. Representative earlier applications of this line of work include image ranking [20] and face recognition [21]. The standard assumption made by the majority of these methods is that only *one* similarity criterion is being used. However, when collecting data from human annotators, different individuals may be using different visual criteria when making classification and similarity judgements, e.g. one individual could be using shape, while the other is using color. Furthermore, the same individual may change the criteria they use conditioned on the specific items they are shown, and could change to another criteria when shown another set of items at a different point in time.

There has been some work that attempts to deal with the fact that *different* similarity criteria may be being used. These range from fully supervised, where the similarity criteria is known at training time [22], through to unsupervised methods that attempt to estimate the criteria [23, 24, 25, 26]. Attempts have also been made to probabilistically estimate item embeddings along with annotator-specific parameters representing the individual criteria they are using [26].

One common assumption made by the above methods is that the learner is stationary, i.e. they are used a predefined and fixed similarity criteria, or small set of criteria, which do not evolve or change over time. This is a reasonable assumption to make when dealing with common everyday object categories where the annotators will likely be familiar with the objects depicted and have an a priori understanding of how the visual features of the objects may vary. However, this assumption is violated in cases where the annotator is in the process of *learning* the visual concepts of interest. In this work, we address this non-stationary setting and show that by doing this, we can more accurately predict the visual classification behavior of real human learners.

2.3.2 Human Category Representation

Existing models of human category representation and learning can be clustered into five major groups: rule-based, prototype-based, exemplar-based, knowledge / theory-based, and decision boundary-based approaches [27, 28]. The current consensus is that humans likely use multiple different category-learning systems depending on the specific nature of the task at hand [29, 30]. For example, in rule-based tasks, the optimal policy may be easy to verbalize and thus efficiently encoded via a set of rules. In practice however, perceptual tasks such as fine-grained visual categorization can be much harder to represent in this way [31]. Several works have attempted to extract perceptual embeddings that align with human similarity judgements from coarse [32, 33] and more fine-grained [34] image collections. [35] showed that with simple linear transformations, pre-trained deep image classifiers can be predictive of human similarity judgements. Relevant to our work, [36] investigated whether human learning dynamics mimic gradient descent in Artificial Neural Networks when learning visual categories. However, they assumed they had access to the feature space used by the human, whereas we instead attempt to learn this. In this work, we also aim to extract human-aligned representations, but in the more challenging setting whereby our learners are not static, but instead are in the process of learning the categorization task.

2.3.3 Knowledge Tracing

The problem of modelling the hidden state of dynamic learners as they interact with a learning task has also been tackled in the knowledge tracing literature. Bayesian Knowledge Tracing-based methods model a learner's knowledge state by assuming that the learner can be represented as a Markov process which updates online during learning [37]. Building on this line of work, Deep Knowledge Tracing (DKT)

instead uses a recurrent neural network as the underlying tracing model [9], and fully self-attention-based methods have also been proposed [10, 11]. It is important to note that conventional knowledge tracing attempts to model knowledge acquisition as a binary variable at the 'skill' level (i.e. the visual class) as opposed to the 'instance' level (i.e. a specific image). In contrast, our approach jointly learns an image embedding function in addition to being able to capture and predict the individual learning trajectories of multiple different learners. A detailed comparison of the original DKT model and our model is provided in the supplementary material (see Sec. 2.7.2).

2.3.4 Machine Teaching

Estimating the representations used by humans is an important component for developing automated teaching algorithms and systems. Machine teaching algorithms address the teaching problem by generating sequences of instructional examples to show to novice learners in order to improve their ability on a given task [38]. Machine teaching has applications in crowdsourcing where the aim is to efficiently train crowdworkers, in addition to education where the goal is to train new experts, e.g. in medical image analysis [39, 40].

There is a growing body of work in computer vision that attempts to teach visual concepts to human learners, e.g. [41, 42, 43, 44, 45]. However, many of these works assume a fixed feature space that is generated before teaching begins [41, 42, 43]. In one experiment, [43] showed that representations that are better aligned with human perception result in improved learner performance on the downstream teaching task. While we do not explicitly investigate teaching algorithms in this work, we instead explore a setting where data is collected from humans engaged in learning a visual categorization task with instructional images selected by a 'random' teacher. Importantly, the representations and learner parameterizations extracted by our model can be used directly with computer assisted teaching methods.

2.4 Method

Our goal is to estimate the image classification function used by a human learner that has been provided with a sequence of images and corresponding ground truth class labels as training data. We begin by outlining the problem, and then present our approach to human visual knowledge tracing.

2.4.1 Problem Setup

Given an image $\mathbf{x}$ as input, we model a human learner as a classification function that returns a response, $r^k = \text{argmax}_c\, P(c|\mathbf{x}, \theta^k)$. Here, r is a discrete class label representing the class response for learner k, i.e. $r \in \{1, ..., C\}$, where C is the number of possible classes, and θ^k are unobserved parameters representing the state of the learner. Our learners are not stationary as their internal 'knowledge state' changes depending on the information they have previously been exposed to that is relevant to the task. As a result, for a given learner k we model their classification function at time t as $r^k_{t+1} = \text{argmax}_c\ P(c|\mathbf{x}, \theta^k_t, \mathbf{x}^k_{1:t}, y^k_{1:t}, r^k_{1:t})$. Here, $(\mathbf{x}^k_{1:t}, y^k_{1:t}, r^k_{1:t})$ is the history of images, ground truth class labels, and responses that a learner k has seen, and provided, up to and including time t.

Specifically, at each training time-step, a learner k is presented with an image $\mathbf{x}$, they provide their response r, and are given feedback in the form of the correct class label y (Fig. 2.2B). However, fitting a model for an individual learner with a single response per time-step is difficult. Alternatively, requesting more responses per time-step would reduce the number of teaching examples presented to the learner in the same amount of time. Instead, to overcome this limited information setting, we train a model ϕ across many learners, allowing the model to discover knowledge states and learning rules shared across all learners. Once trained, our model can make predictions for how a learner, who was not observed during training, will classify an image based on their prior classification behavior.

2.4.2 Tracing Human Learners

Our tracing model ϕ can be decomposed into a feature extractor f, a classification function ψ, and a non-learned and non-linear transformation σ (softmax). We explore how to represent the feature extractor and classification function.

A natural choice for the feature extractor f is a Convolutional Neural Network (CNN). Given an image $\mathbf{x}$ as input, the feature extractor outputs a D dimensional vector $\mathbf{z} = f(\mathbf{x})$. We will assume that all learners use the same underlying feature extractor which remains constant over time i.e. $f = f^k_t$, and that they simply differ in the relative importance they place on different visual features. While these are both big assumptions to make, they are not overly restrictive. For example, a novice and an expert might engaged in the same visual classification task but differ in the set of visual features they select in order to make their decision. Furthermore, while we assume that the feature extractor remains constant over the time interval

of our experiments, we do not assume that the classification function ψ used by a learner remains static. In the next sections we will explore different choices for this classification function, comparing simple static classifiers with more expressive recurrent models.

Static Tracing Model

The first model we explore is the simplest. Here we assume that all learners use the same classifier which does not vary over time. In this setting, ψ is a multi-class linear classifier with a weight matrix $\mathbf{w}$ and per-class biases $\mathbf{b}$,

$$\phi_{static}(\mathbf{x}) = \sigma(\psi(f(\mathbf{x}))) = \sigma(\mathbf{w}^\top f(\mathbf{x}) + \mathbf{b}). \tag{2.1}$$

This model is similar to conventional metric learning approaches which do not attempt to capture any annotator specific differences related to individual biases or temporal changes. At training time we simply estimate one set of parameters for all learners. This model does not take the response history into account.

Time-Sensitive Tracing Model

One obvious limitation of the static tracing model is that it does not take into account the fact that a learner will likely change over time, i.e. they may be much worse at a new classification task early on, but may improve over time as they are shown sequences of example images along with their associated ground truth class labels. A more advanced model, that captures this temporal evolution, is one that has a different classifier for each time-step,

$$\phi_{static\_time}(\mathbf{x}) = \sigma(\mathbf{w}_t^\top f(\mathbf{x}) + \mathbf{b}_t). \tag{2.2}$$

Again, the same classifiers are shared across all learners, but in this case the weights and biases are different at each time-step, i.e. $\mathbf{w}_t \neq \mathbf{w}_{t-1}$.

2.4.3 Recurrent Tracing Models

The previous tracing models do not account for the fact that individual learners may start with different levels of ability and update their internal knowledge state in different ways depending on the information that they are provided with. [9] showed that recurrent networks could be used to track the skill acquisition of human learners engaged in learning math quiz questions. Direct application of their model to our visual categorization setting is not possible as they assume one hot encodings of

the query and learner responses as input. Their approach also uses large training sets – on the order of thousands of learners and tens of thousands of interactions. Furthermore, they model knowledge acquisition at the 'concept' (i.e. visual category) and not 'instance' (i.e. a specific image) level, and thus their approach is not capable of making predictions for items not seen during training. We build on [9] and adapt it to our visual category learning setting by presenting two different recurrent-based models for human visual knowledge tracing.

Direct Response Model

Our first model uses a recurrent network to directly predict the responses of a learner given their previous response history,

$$\phi_{direct}(\mathbf{x}) = \sigma(\psi_{rnn}(\mathbf{z}^k_{1:t}, y^k_{1:t}, r^k_{1:t}, \mathbf{z}, y)). \tag{2.3}$$

Here, ψ_{rnn} is a recurrent network (in practice we represent this using an LSTM [46]) and $\mathbf{z}_t = f(\mathbf{x}_t)$ are visual features extracted from our CNN.

This model assumes that a learner's knowledge state at time t is defined by the images they have previously seen and their past classification responses. Recurrent models can produce unique transformations for individual learners by conditioning on their hidden states. In this case, after the shared feature extractor transforms an image into a feature vector, the model modifies the feature vector with a series of non-linear transformations conditioned on the learner's hidden state. The final linear layer transforms the feature vector into a predicted response. Note that this model is also conditioned on the current query image $\mathbf{z} = f(\mathbf{x})$ and the corresponding ground truth class label y.

Classifier Prediction Model

Our second recurrent model attempts to provide a more interpretable approximation of human classification. In this case, instead of letting the recurrent model directly predict the probability for each response, it instead attempts to approximate the weights of a linear classifier used internally by the learner. Importantly, the values this classifier takes will depend on the response history of a given learner and will differ at each time-step,

$$\mathbf{w}^k_t, \mathbf{b}^k_t = \psi_{rnn}(\mathbf{z}^k_{1:t}, y^k_{1:t}, r^k_{1:t}, y), \tag{2.4}$$

$$\phi_{cls\_pred}(\mathbf{x}) = \sigma(\mathbf{w}_t^\top \mathbf{z} + \mathbf{b}_t). \tag{2.5}$$

Unlike the previous direct prediction recurrent model, we now explicitly represent the classification function used by an individual learner. Also, here the features $\mathbf{z}$ for the query image are not processed by the recurrent network in Eqn. 2.4. Instead they are evaluated using the much simpler predicted classifier weights in Eqn. 2.5. This decoupling is advantageous in applications like machine teaching where we have to query the tracing model multiple times at each time-step in order to determine the next image to show learners. Reducing the computation required to perform these queries will result in faster teacher algorithms.

2.4.4 Training Tracing Models

We jointly estimate the parameters of the feature extractor f and classification functions ϕ for each of the above models using a standard cross entropy loss,

$$\mathcal{L} = -\frac{1}{KT}\sum_{k=1}^{K}\sum_{t=1}^{T}\log(\phi(\mathbf{x}_t^k)_r). \tag{2.6}$$

Here, $\phi(\mathbf{x}_t^k)_r$ indicates the predicted probability from a model ϕ choosing class r, for learner k, at time-step t. The training objective aims to minimize the difference between the learner responses from our training set and the tracing model outputs.

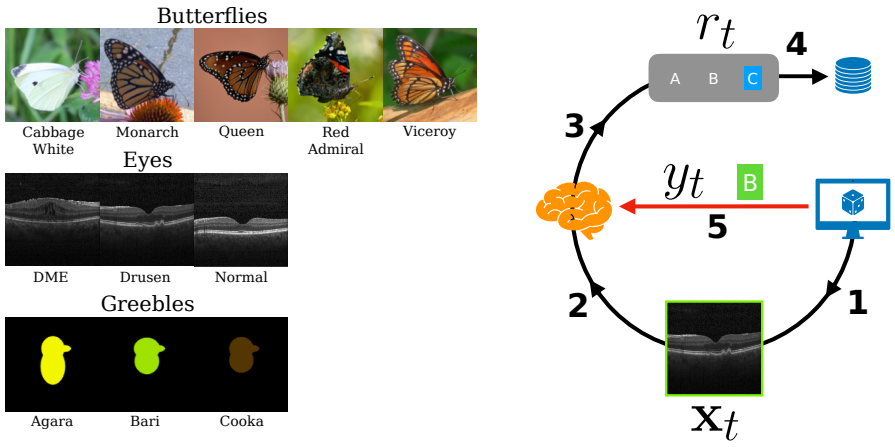
2.5 Experiments

In this section we evaluate the different proposed models for visual knowledge tracing on data we collected from real human participants[1].

2.5.1 Datasets

Traditional image classification datasets mostly contain labels produced by annotators familiar with the subject material, e.g. [48, 49, 50], or they have at least received detailed instructions and examples on how to annotate them, e.g. [51]. As a result, these datasets do not contain annotations from learners engaged in learning a task and are thus not suitable for evaluating visual knowledge tracing. While some work has focused on teaching crowd learners (e.g. [41, 43, 44]), they often use teaching image sequences that are determined offline and fixed. For our tracing experiments, we require unbiased sequences of images that are randomly selected for each learner. Some of these existing works compare their approaches to a random image selection baseline, but the size of these random teaching subsets is insufficient for thorough evaluation of our different tracing approaches, e.g. [43] have random selection data

[1] Code and dataset — https://github.com/nkondapa/VisualKnowledgeTracing



(A) Image Datasets (B) Data Collection

Figure 2.2: Collecting visual knowledge tracing datasets. (A) Example images from the three different datasets from our experiments. 'Butterflies' contains images of five different species and was originally presented in [43]. 'Eyes' contains optical coherence tomography images of the human retina from [47], and features two diseased classes and one normal one. 'Greebles' is a synthetic dataset we created where the three object classes vary in terms of shape and color. **(B)** Data collection pipeline. A random image is selected (1), shown to the learner (2), and the learner provides a response (3). Their response is stored (4) and the correct class label is provided to them (5).

from only $\sim$ 40 participants. Due to these limitations, we collected annotations from human learners for three challenging fine-grained visual classification datasets.

Image Data

We selected three different image datasets that cover three distinct domains: artificial data where we have full knowledge of the underlying distribution, medical image data, and images of different wildlife species. The first two datasets in particular are representative of the types of visual identification tasks that many humans are interested in learning.

Our first dataset, 'Butterflies', contains images from five different common species of North American butterflies. The 'Cabbage White' class is immediately recognizable, 'Red Admiral' can be learned relatively easily, and the remaining three are difficult to discriminate. This dataset was originally used in [43] and contains between 386 and 481 images per class, for a total of 2,224 images. Our second dataset, 'Eyes', is a three-class subset of a large collection of publicly available images of the human retina from [47]. It contains two diseased classes, 'Diabetic Macular Edema' (DME) and 'Drusen', and one 'Normal' class. We manually selected 200 images from each

class. The third dataset is a challenging synthetic one we created called 'Greebles'. It contains three classes, where the underlying feature space used to generate the images is known by design. The relevant features are the body length and color, but the images also includes some irrelevant variation in the form of the head size and body width. The distinctions between the classes can be subtle, making the task challenging. It contains 1,200 images in total, with an even number per class. Visual examples for each of the datasets are presented in Fig. 2.2A. Note that the single examples in Fig. 2.2A do not covey the visual diversity of the datasets.

Human Data Collection

For each of the previously described datasets, we collected data from human participants that were engaged in learning the classification task. Each learner was presented with 30 training images and 15 test images. During the training phase they were provided with ground truth feedback indicating the correct class labels. This feedback was not provided in the test phase. For each image, we asked learners to rank the top three classes in order of most likely to least likely to be correct (see Sec. 2.12.1 for further information). The training and testing examples were randomly selected for each learner. An overview of the data collection process for one iteration, for one learner, is shown in Fig. 2.2B.

The data was collected using a custom built a web application and the participants were recruited through the crowd sourcing app Prolific [52]. In total, we collected data from 150 learners for each dataset, where each individual could only do the task once. The median time spent on the Butterflies task, including training and testing, was ~ 11.7 minutes with a median of 11 correct on the test phase. The corresponding statistics for the Eyes dataset was ~ 12.1 minutes with a median of 13 correct, and for the Greebles dataset is was ~ 8.8 minutes with a median of 9 correct.

Our human learners demonstrated learning across all three datasets. In Fig. 2.3 we plot two histograms for each dataset, the first histogram displays the number of correct responses during the training (i.e. teaching) phase and the second reports the same for the test phase. For all of the datasets, we see a right skew in the histogram of correct responses in the testing phase, indicating that most learners are providing correct answers after training. The skewness for each dataset is -0.45 for Butterflies, -0.81 for Eyes, and -0.37 for Greebles. The skewness is correlated to the amount of improvement learners showed on a given dataset. The numbers imply that the Greebles task is the most difficult and the learners demonstrate the least improvement.

The Eyes dataset, on the other hand, is clearly the easiest. We provide additional analysis in the supplementary material (Sec. 2.9).

2.5.2 Implementation Details

All models we are considering primarily consist of a feature extractor and a classification function. The feature extractor is implemented as a CNN with eight layers (two convolutional, two max-pool, four linear), and is the same across all models (Sec. 2.8). The feature extractor produces a 16 dimensional embedding for an input image which is then processed by the respective classification function. We evaluated one of our models on higher dimensional feature spaces, but found no impact on performance (Sec. 2.7.7). For the two recurrent models, ϕ_{direct} and ϕ_{cls_pred}, we use a three layer LSTM-based [46] fully connected network with a hidden dimension of size 128. The output of the LSTM is passed to a small two-layer network that transforms the output into the desired representation (either a response or a classifier (see Sec. 2.4.3)). We provide a detailed description of the architectures and inputs in the supplementary material (Sec. 2.12.2).

All models are trained using mini-batch stochastic gradient descent with a batch size of 16 using the Adam optimizer [53]. For the feature extractor, we use a learning rate of 1e-5, and we use a learning rate of 1e-3 for each of the different classification functions. We train using a cross entropy loss. Models are trained with early stopping. Training ends when the best validation loss does not improve for 35 epochs. The upper limit on the number of training epochs is 400. Data is split into train (70%), validation (13.3%), and test (16.7%) splits at the learner level, i.e. sequences from the same individual learner cannot be in more than one split. To ensure more robust results, we validate models by re-shuffling the data five times and report the averages across the splits using micro and macro average precision.

2.5.3 Results

Tracing Human Learners

We compute the precision and recall curves for each of the visual knowledge tracing models on the held-out learners and report the average precision for each. We also include one additional model for completeness, the ground truth baseline model (GT Label). This baseline does not fit any parameters and simply predicts the corresponding ground truth label of the image for all learners at all time-steps (instead of predicting the learner's response). Results are summarized in Table 2.1. Standard deviations are between 0.01 and 0.03 for all models, and do not change the interpretation of the results.

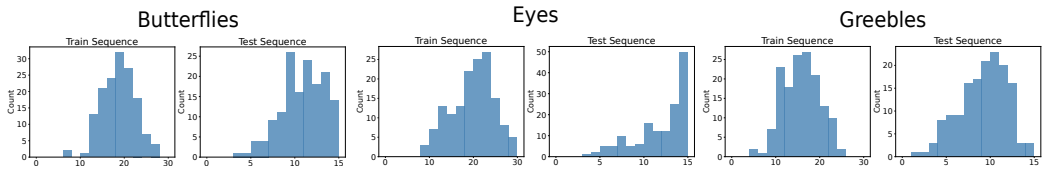


Figure 2.3: Human learner performance on our three datasets. For each dataset we provide histograms of learner performance on the respective training and testing sequences. The training results are always worse as it include responses from all time-steps, including when the learner has just started the task and is unfamiliar with the classes. For 'Greebles', the worse performance on the test set compared to the other datasets indicates that learners find this task more challenging.

We observe that the recurrent models, ϕ_{direct} and ϕ_{cls_pred}, out-perform the baselines in tracing the learner on the Butterflies (∼8%) and Eyes datasets (∼6%). However, on the Greebles dataset, the static tracing model, ϕ_{static}, outperforms the recurrent models by (∼3%). Both recurrent models are comparable in terms of performance, indicating the reduction in computation described in Sec. 2.4.3 does not come with a reduction in performance. The time-sensitive tracing model, ϕ_{static_time}, is clearly the worst at tracing learners, likely owing to the unrealistic assumptions it makes about them.

Finally we explore the differences between how different models trace human learners. Fig. 2.4 shows the average probability of predicting an image correctly for each time-step conditioned on each class for the Butterflies dataset. In the top row, we present the training and test-split average accuracy on each of the fives classes over time for the human learners. As previously noted, the training sequences contains 30 randomly selected images and the test sequence contains 15 images. We can see that on average, some classes are much easier than others. In the bottom row, we average model-predicted probabilities for ∼50 images in each class. To produce the probabilities for the recurrent ϕ_{cls_pred} model, we processes the sequential data of the same set of learners in the top panel. The static tracing model, ϕ_{static}, estimates a hyperplane for each class that roughly tracks the average probability of being correct per class. As expected, this model is not capable of capturing any learning behavior. In contrast, ϕ_{cls_pred} more faithfully traces how the average probability evolves over time. Note, that we process each of the test images independently for ϕ_{cls_pred}.

Table 2.1: Performance of different visual knowledge tracing approaches on data from human learners. We observe that our two recurrent based models, the direct response ϕ_{direct} and the classifier prediction ϕ_{cls_pred}, perform best on the Butterflies and Eyes dataset but are worse on the synthetic Greebles task. Learners found the Greebles task the most challenging, and as a result, there was much less learning occurring compared to the first two datasets. 'GT Label' is an additional baseline that uses the corresponding ground truth class label y as the prediction of the learner's response r.

	Greebles				**Eyes**				**Butterflies**			
	Train		Test		Train		Test		Train		Test	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
GT Label	0.48	0.51	0.58	0.61	0.56	0.56	0.69	0.69	0.45	0.44	0.50	0.49
ϕ_{static}	0.63	0.52	0.67	0.58	0.60	0.59	0.67	0.68	**0.55**	**0.53**	**0.64**	**0.61**
ϕ_{static_time}	0.52	0.44	0.49	0.40	0.34	0.34	0.33	0.34	0.54	0.52	0.61	0.59
ϕ_{direct}	0.70	0.59	**0.77**	0.64	**0.66**	**0.65**	**0.75**	**0.74**	**0.55**	**0.53**	0.60	0.57
ϕ_{cls_pred}	**0.71**	**0.62**	**0.77**	**0.65**	0.65	**0.65**	0.74	**0.74**	0.54	0.52	0.60	0.57

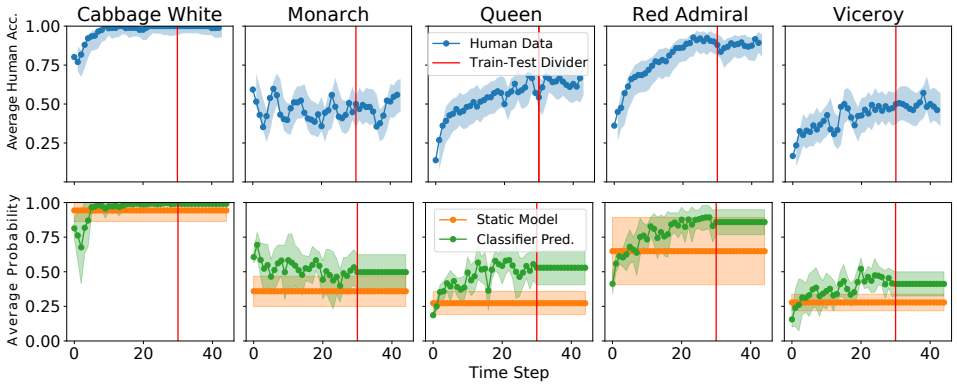


Figure 2.4: Comparing student accuracy and tracing model predicted accuracy. (Top) The smoothed average human learner accuracy for class over time from the Butterflies dataset. The shadowed regions indicate confidence intervals as the number of samples in each time and class bin are not guaranteed to be the same. **(Bottom)** The average probability of having a class correctly predicted by the static ϕ_{static} model (orange) and the recurrent ϕ_{cls_pred} model (green). At each time-step, for each learner in the test set, the models predict class probabilities for $\sim$ 50 images per class. The probabilities are averaged (solid line) and the shadows indicate one standard deviation. In both rows, the red line indicates the point at which the learners switch from training to testing. After that, the models will continue to produce the same probabilities on the test images for the remaining time-steps as the sampled images do not change.

2.5.4 Discussion

Comparing Models

We observe that our recurrent models are quite effective at tracing human learner knowledge on visual categorization tasks for datasets where there is a clear learning signal. On the Greebles dataset, which is the most challenging and displays the

least amount of learning, we see that the simple static tracing model ϕ_{static} is less prone to over-fitting and is thus marginally better. The time-sensitive tracing model ϕ_{static_time} performs the worst overall. Unlike the recurrent methods, it is unable to share information between time-steps, forcing it to fit a classifier at each time-step with only ~90 training points (the number of learners in the training set after we split out the validation and test sets). This makes it extremely prone to over-fitting on the limited training data that is available.

The direct response model ϕ_{direct} and the classifier prediction model ϕ_{cls_pred} differ in their outputs and their inputs. ϕ_{direct} includes $\mathbf{z}$, the representation of the query image $\mathbf{x}$ to be classified by the learner, as input to the LSTM. ϕ_{cls_pred} outputs weights of a multi-class linear classifier that is used to classify the representation $\mathbf{z}$. Unlike ϕ_{direct}, $\mathbf{z}$ is not an input to the LSTM for ϕ_{cls_pred}. However, both models incorporate the ground truth label y for $\mathbf{x}$ as input. These structural differences between the models make little difference to the tracing performance, but the ϕ_{cls_pred} model is more efficient if it needs perform evaluation multiple times for new query items. In supplementary experiments (2.7.1), we explore the impact of removing the ground truth class label y for the query image from the input. We observe that this results in a large decrease in performance for both models, suggesting that this class information is valuable to the model when making future predictions. Also in the supplement, we compare two cognitive models (prototype and exemplar [54]) (2.7.3), a Transformer model [55] (2.7.4), a pre-trained ResNet feature extractor (2.7.5), and input meta-information to the tracing models (2.7.6). We find that the cognitive, Transformer, and ResNet models do not out-perform the recurrent architectures, but are worth further exploration. Additionally, including meta-information in the input vectors results in a performance increase across all models we tested, suggesting this a promising direction for future work. Finally, we explore the representations learned by the recurrent models (Secs. 2.10, 2.11).

Limitations and Future Work

Currently, we train our feature encoder f from scratch for each task on relatively small amounts of image data. One source of improvement would be to pre-train the feature space so that it better reflects human visual similarity judgements. Such an improved feature space would provide a better starting point for task specific finetuning. Replacing the LSTM with an appropriately designed Transformer network [56] is another change that could result in greater flexibility for the model. Transformers are

better able to capture long-term dependencies and have been shown to be useful in knowledge tracing on non-visual educational datasets [10, 55]. However, it remains to be seen if this would be valuable for visual knowledge tracing.

Our approaches do not explicitly model memory decay — the phenomenon of memory 'fading' due to the passage of time [57, 58]. This is likely to be more of a problem when tracing over longer time horizons, e.g. days or weeks, as opposed to the multiple minute long sessions that our learners engage in. Similarly, given the short time durations of our teaching sessions, we assume that no significant 'feature learning' is happening for individual learners. Instead we model learners as attending to a subset of the different visual features that are captured in our joint embedding space. In future work, it would be interesting to further explore if these two assumptions are valid.

Applications

Successful tracing of human learners has implications for crowd-sourcing annotations, metric learning, and machine teaching. Early detection of poor annotators in crowd sourcing would reduce monetary and time costs in labelling datasets. Additionally, identifying annotators with 'specialist' knowledge could allow for targeted crowd sourcing, tailoring to the abilities of each individual annotator. A successful tracing algorithm should be able to predict future performance with increasing confidence as the learners are being trained on the annotation task.

The most impactful domain for a successful tracing algorithm is automated teaching, e.g. teaching medical image interpretation skills [39, 40]. Teaching humans is challenging because their knowledge state is unobserved, it changes over time, and the information they provide using current interfaces can be limited. In this work, we show that with a reasonable amount of training data (i.e. data from $\sim$ 150 learners), and only a single response at each time-step, we are able to capture information about the learner's current knowledge for visual classification tasks. The types of visual knowledge tracing approaches presented could be used in conjunction with machine teaching methods. The more a teaching algorithm knows about the learner, the more effective it can be when selecting examples to present to them.

2.6 Conclusion

More accurate models of human visual classification will lead to improved methods for crowd annotation collection, better techniques for automatically teaching visual

knowledge to human learners, and perhaps provide us with insight into how we can build future artificial systems that are more data efficient. To this end, in this work we explored the problem of visual knowledge tracing — the task of predicting the internal, potentially time varying, image classification function used by human learners. To do this, we presented a series of models that range in complexity from basic static linear classifiers all the way to recurrent models that take a learner's prior response history into account when making predictions about their future behavior. We collected new annotations for three challenging visual classification tasks from humans engaged in a visual learning task in order to benchmark the performance of these different models. Our results show that our recurrent neural network-based models resulted in the most faithful reproductions of unobserved learner predictions on real image datasets. Finally, we outlined limitations of our work and pointed to open questions that require further investigation.

Acknowledgments

Thanks to the anonymous reviews for their valuable feedback. This work was in part supported by the Turing 2.0 'Enabling Advanced Autonomy' project funded by the EPSRC and the Alan Turing Institute and also by the Simons Collaboration on the Global Brain.

Appendix

2.7 Additional Experiments

In this section, we present several additional models and also consider the impact of embedding dimension on performance.

2.7.1 RNN Variants

We can vary the input information to both of the recurrent models in three ways. The notation in parentheses maps to the entries in the later supplementary tables.

(base): The models only receive the history of images, ground truth class labels, and learner responses

$$\psi_{rnn}(\mathbf{z}^k_{1:t}, y^k_{1:t}, r^k_{1:t})). \tag{2.7}$$

(y): In addition to the history, the model receives the ground truth class label of the image shown to the learner at the current time-step

$$\psi_{rnn}(\mathbf{z}^k_{1:t}, y^k_{1:t}, r^k_{1:t}, y)). \tag{2.8}$$

(y, $\mathbf{z}$): Finally, as in the main paper, the model can include both the ground truth class of the image and the representation of the image from the learned CNN

$$\psi_{rnn}(\mathbf{z}^k_{1:t}, y^k_{1:t}, r^k_{1:t}, \mathbf{z}, y)). \tag{2.9}$$

The results of the variants are presented in Tables 2.2, 2.3, and 2.4.

2.7.2 DKT

Next, we adapt Deep Knowledge Tracing (DKT) [9] to our setting. We deviate from the original DKT method in two main ways. First, the types of queries (e.g. math problems) in educational datasets do not allow for instance-level representations. Instead, skills (i.e. question types) were jointly encoded with information about whether the problem was answered correctly by the learner. Second, the output of DKT was the learner's probability of being correct for each skill, not a particular question instance.

We modify the DKT algorithm to make it appropriate for the setting described in our work. We replace skills with the class-level label for an image and convert the output into a probability distribution over the class labels such that it can be trained with the cross-entropy loss

$$\phi_{dkt}(y) = \sigma(\psi_{rnn}(y^k_{1:t}, r^k_{1:t}, y)). \tag{2.10}$$

At a high-level, this model variant encodes no instance-level (i.e. image) information to make its predictions.

We observe that this DKT model (ϕ_{dkt}) performs slightly worse in all cases, indicating that image information is valuable to enable the models to better trace learner performance. The results of the DKT variant are presented in Tables 2.2, 2.3, and 2.4.

2.7.3 Cognitive Models

Cognitive models make stronger assumptions on how humans learn. We modify the prototype and exemplar models described in the cognitive science literature [54] and evaluate them on our datasets.

Prototype

The prototype model proposes that learners store a prototypical image for each class. Each new image is compared to the learner's class prototypes and the highest similarity class is selected. In our formulation, the class prototype is the average feature representation of previously seen images of that class. In the following equations, τ is the current time-step, $P^k_\tau(c)$ is the prototype of class c for learner k at time-step τ, and δ is the dirac-delta function and acts as a selector for images from class c,

$$P^k_\tau(c) = \frac{1}{\tau - 1} \cdot \sum_t^{\tau-1} z^k_t * \delta(y^k_\tau - c), \tag{2.11}$$

$$\hat{r}^k_\tau(c) = \frac{sim(P^k_\tau(c), z^k_\tau)}{\sum_c sim(P^k_\tau(c), z^k_\tau)}. \tag{2.12}$$

Exemplar

The exemplar model proposes that learners store previously seen images in a memory bank of exemplars. Query images are compared to all of the exemplars. The learner chooses the class with the highest total similarity to the query image. In the following equations, $E^k_\tau(c)$ is the sum of the class c similarity scores for learner k at time-step τ with respect to the current image z^k_τ. Following [54], we introduce a learnable parameter γ to scale the similarities (this value is fixed to 1 in the prototype model),

$$E_\tau^k(c) = \sum_t^{\tau-1} sim(z_t^k, z_\tau^k) \cdot \delta(y_\tau^k - c), \tag{2.13}$$

$$\hat{r}_\tau^k(c) = \frac{sim(E_\tau^k(c), z_\tau^k)^\gamma}{\sum_c sim(E_\tau^k(c), z_\tau^k)^\gamma}. \tag{2.14}$$

Both models compute similarity by using an exponential decay function over the Euclidean distance between feature representations of the images,

$$sim(z_i, z_j) = e^{-c*d(z_i, z_j)}. \tag{2.15}$$

Finally, instead of learning the feature space separately with visual similarity experiments, we jointly estimate a CNN along with the model parameters to discover the feature space.

We find that these models perform worse than the models presented in the main paper. However, simple modifications (like weighting the history of exemplars or images in the prototype) may help. Exploring the space of cognitive models is an interesting direction for future work. The results of these variants are presented in Tables 2.2, 2.3, and 2.4.

2.7.4 Transformers

Recently, the knowledge tracing community has found the Transformer architecture to be an effective model for tracing human learners in non-visual tasks. We modify the SAINT model [55] for our visual learning setting. First, we introduce a CNN-based feature extraction stage to embed images. The encoder receives the current image's embedding and its ground truth label. The decoder gets the previous learner response. The decoder predicts the learner's response to the image (also passed to the encoder).

The Transformer model does surprisingly poorly on these datasets. We expect that future work exploring Transformer architectures designed for this task will demonstrate performance on par with the recurrent models. The results of the Transformer model are presented in Tables 2.2, 2.3, and 2.4.

2.7.5 ResNet Backbone Experiments

We swap out our CNN backbone with a ResNet-18 [59] pre-trained on ImageNet. We freeze the weights in layers 1, 2, and 3, but leave layer 4 to be learned. The output of layer 4 is passed to a fully-connected layer that reduces the output of the layer to

Table 2.2: Performance of all model variants on the Butterflies dataset. The model variant is denoted in the subscript corresponding to the same subscripts in 2.7.1. One can see that $\phi_{direct(base)}$ performs poorly for a recurrent model. This model does not have access to any information about the current time-step and is effectively guessing both the image that will be shown and the associated response. We also show the per-class average precision scores on the train sequence in addition to the micro and macro scores from before. These scores show that the benefit of the recurrent models appear primarily in classes that have large changes in average performance (e.g. Red Admiral) over the training period. The models with † are models presented in Table 1 of the main paper. The scores are reported with their standard deviations and the top average performers in each column are in bold.

	Butterflies								
	Train							Test	
	Cabbage White	Monarch	Queen	Red Admiral	Viceroy	Micro	Macro	Micro	Macro
GT Label†	0.94±0.03	0.32±0.01	0.36±0.04	0.65±0.04	0.27±0.01	0.48±0.02	0.51±0.02	0.58±0.02	0.61±0.02
ϕ_{static}†	0.95±0.01	0.37±0.02	0.34±0.06	0.66±0.03	0.27±0.03	0.63±0.02	0.52±0.02	0.67±0.02	0.58±0.03
ϕ_{static_time}†	0.97±0.01	0.34±0.03	0.29±0.03	0.35±0.03	0.24±0.02	0.52±0.02	0.44±0.01	0.49±0.01	0.40±0.01
ϕ_{dkt}	0.95±0.02	0.43±0.02	0.45±0.04	0.72±0.05	0.33±0.06	0.67±0.02	0.57±0.02	0.74±0.02	0.64±0.02
$\phi_{transformer}$	0.96±0.01	0.36±0.03	0.34±0.02	0.69±0.06	0.26±0.01	0.62±0.02	0.52±0.02	0.68±0.05	0.58±0.04
$\phi_{prototype}$	0.95±0.02	0.32±0.03	0.30±0.05	0.59±0.08	0.27±0.03	0.53±0.03	0.48±0.03	0.63±0.01	0.54±0.02
$\phi_{exemplar}$	0.90±0.03	0.30±0.03	0.27±0.04	0.35±0.13	0.26±0.03	0.44±0.05	0.42±0.04	0.53±0.07	0.45±0.07
$\phi_{direct(base)}$	0.34±0.02	0.29±0.03	0.23±0.02	0.26±0.02	0.22±0.03	0.27±0.01	0.27±0.01	0.20±0.01	0.20±0.01
$\phi_{direct(y)}$	0.97±0.02	0.43±0.03	**0.48±0.05**	0.76±0.07	**0.38±0.04**	**0.71±0.02**	0.60±0.02	**0.78±0.02**	**0.66±0.01**
$\phi_{direct(y,\mathbf{z})}$†	0.97±0.01	0.41±0.04	0.44±0.06	0.77±0.07	0.36±0.03	0.70±0.03	0.59±0.02	0.77±0.03	0.64±0.03
$\phi_{cls\_pred(base)}$	0.96±0.01	0.41±0.06	0.32±0.05	0.59±0.14	0.25±0.02	0.59±0.06	0.51±0.05	0.61±0.06	0.52±0.05
$\phi_{cls\_pred(y)}$†	**0.98±0.01**	**0.46±0.02**	**0.48±0.04**	**0.78±0.05**	**0.38±0.02**	**0.71±0.02**	**0.62±0.01**	0.77±0.02	0.65±0.02
$\phi_{cls\_pred(y,\mathbf{z})}$	**0.98±0.01**	0.45±0.01	0.47±0.04	**0.78±0.05**	0.37±0.02	0.70±0.02	0.61±0.01	0.77±0.03	**0.66±0.02**

the desired dimensionality, as opposed to the final classifier used for the ImageNet classification task. The results of these experiments are presented in Table 2.5.

2.7.6 Including Per-Class Accuracy as Input

We find that including some meta-information can help with tracing performance. To do this, we compute a learner's accuracy on each class at each time-step and concatenate this vector to the input of three tracing models (ϕ_{static}, ϕ_{direct} and ϕ_{cls_pred}). We find a boost in performance across all models. The results are presented in Table 2.6, where we observe a boost in performance. It is likely that other sources of meta-information (such as time-taken on an example) will also help [60].

2.7.7 Varying Embedding Dimension

We demonstrate that varying the embedding dimension of the feature extractor has little effect on the performance of the direct response model (Table 2.7).

Table 2.3: Performance of all model variants on the Eyes dataset. Please see the caption of Table 2.2 for more details.

	Eyes						
	Train					Test	
	DME	Drusen	Normal	Micro	Macro	Micro	Macro
GT Label†	0.56±0.02	0.54±0.03	0.58±0.02	0.56±0.02	0.56±0.02	0.69±0.02	0.69±0.01
ϕ_{static}†	0.63±0.03	0.53±0.05	0.62±0.02	0.60±0.03	0.59±0.03	0.67±0.02	0.68±0.02
ϕ_{static_time}†	0.32±0.01	0.35±0.01	0.36±0.01	0.34±0.00	0.34±0.01	0.33±0.01	0.34±0.02
ϕ_{dkt}	0.63±0.02	0.60±0.03	0.65±0.02	0.63±0.02	0.63±0.02	0.74±0.03	0.73±0.03
$\phi_{transformer}$	0.41±0.09	0.41±0.05	0.42±0.09	0.41±0.08	0.41±0.08	0.37±0.02	0.39±0.02
$\phi_{prototype}$	0.61±0.04	0.50±0.04	0.56±0.04	0.56±0.03	0.56±0.03	0.65±0.04	0.65±0.05
$\phi_{exemplar}$	0.57±0.02	0.48±0.02	0.59±0.03	0.54±0.02	0.55±0.02	0.68±0.04	0.67±0.04
$\phi_{direct(base)}$	0.38±0.02	0.40±0.01	0.38±0.02	0.38±0.01	0.39±0.01	0.35±0.01	0.34±0.01
$\phi_{direct(y)}$	**0.65±0.03**	**0.62±0.03**	0.68±0.03	**0.66±0.02**	**0.65±0.02**	**0.75±0.01**	0.73±0.02
$\phi_{direct(y,\mathbf{z})}$†	0.64±0.02	**0.62±0.03**	**0.69±0.02**	**0.66±0.02**	**0.65±0.02**	**0.75±0.01**	**0.74±0.02**
$\phi_{cls\_pred(base)}$	0.48±0.05	0.39±0.02	0.48±0.05	0.45±0.04	0.45±0.03	0.44±0.02	0.44±0.02
$\phi_{cls\_pred(y)}$†	**0.65±0.02**	**0.62±0.03**	**0.69±0.01**	0.65±0.03	**0.65±0.02**	0.74±0.02	**0.74±0.04**
$\phi_{cls\_pred(y,\mathbf{z})}$	0.64±0.01	**0.62±0.03**	**0.69±0.02**	0.65±0.02	**0.65±0.02**	**0.75±0.01**	**0.74±0.02**

Table 2.4: Performance of all model variants on the Greebles dataset. Please see the caption of Table 2.2 for more details.

	Greebles						
	Train					Test	
	Agara	Bari	Cooka	Micro	Macro	Micro	Macro
GT Label†	0.51±0.02	0.37±0.03	0.43±0.03	0.45±0.02	0.44±0.02	0.50±0.01	0.49±0.01
ϕ_{static}†	0.63±0.03	**0.43±0.04**	0.55±0.05	**0.55±0.03**	**0.53±0.03**	**0.64±0.01**	**0.61±0.01**
ϕ_{static_time}†	**0.64±0.04**	0.39±0.02	0.54±0.04	0.54±0.03	0.52±0.03	0.61±0.01	0.59±0.02
ϕ_{dkt}	0.59±0.03	0.41±0.03	0.49±0.05	0.52±0.02	0.50±0.02	0.59±0.02	0.55±0.02
$\phi_{transformer}$	0.52±0.11	0.36±0.03	0.45±0.05	0.46±0.07	0.45±0.06	0.44±0.08	0.43±0.08
$\phi_{prototype}$	0.58±0.03	0.42±0.01	0.54±0.05	0.52±0.02	0.51±0.02	0.58±0.02	0.57±0.02
$\phi_{exemplar}$	0.59±0.02	**0.43±0.03**	0.52±0.05	0.52±0.03	0.51±0.03	0.63±0.01	0.60±0.01
$\phi_{direct(base)}$	0.37±0.02	0.35±0.02	0.36±0.01	0.36±0.00	0.36±0.00	0.34±0.02	0.34±0.02
$\phi_{direct(y)}$	0.59±0.02	0.41±0.03	0.51±0.04	0.52±0.02	0.50±0.03	0.59±0.02	0.55±0.02
$\phi_{direct(y,\mathbf{z})}$†	0.62±0.03	0.42±0.03	0.55±0.05	**0.55±0.03**	**0.53±0.03**	0.60±0.02	0.57±0.03
$\phi_{cls\_pred(base)}$	0.63±0.03	0.40±0.02	**0.56±0.06**	**0.55±0.02**	**0.53±0.03**	0.61±0.02	0.60±0.03
$\phi_{cls\_pred(y)}$†	0.62±0.03	0.41±0.03	0.54±0.03	0.54±0.02	0.52±0.03	0.60±0.02	0.57±0.03
$\phi_{cls\_pred(y,\mathbf{z})}$	0.63±0.03	0.41±0.03	0.55±0.03	**0.55±0.02**	**0.53±0.01**	0.61±0.02	0.59±0.02

Table 2.5: Performance of models trained using a pre-trained ResNet with partially frozen weights. Method details are described in Sec. 2.7.5. We only compare model variants that appear in the main text. Similar to the original experiment results the classifier prediction model (ϕ_{cls_pred}) performs the best. However, the overall performance decreases slightly across the board. We observe a larger decrease for the direct response model (ϕ_{direct}), likely due to the larger dependence that it has on the feature space.

	Butterflies								
	Train							Test	
	Cabbage White	Monarch	Queen	Red Admiral	Viceroy	Micro	Macro	Micro	Macro
GT Label	0.94±0.03	0.32±0.01	0.36±0.04	0.65±0.04	0.27±0.01	0.48±0.02	0.51±0.02	0.58±0.02	0.61±0.02
ϕ_{static}	0.95±0.01	0.36±0.02	0.32±0.04	0.65±0.04	0.27±0.03	0.62±0.02	0.51±0.02	0.67±0.03	0.57±0.03
ϕ_{static_time}	0.44±0.02	0.25±0.02	0.22±0.01	0.33±0.03	0.19±0.02	0.28±0.01	0.29±0.00	0.27±0.02	0.34±0.02
ϕ_{direct}	**0.97±0.01**	**0.41±0.03**	0.38±0.06	0.74±0.07	0.28±0.03	0.66±0.03	0.55±0.02	0.70±0.03	0.58±0.04
ϕ_{cls_pred}	**0.97±0.01**	0.39±0.02	**0.49±0.02**	**0.75±0.06**	**0.32±0.03**	**0.69±0.01**	**0.59±0.01**	**0.75±0.01**	**0.65±0.02**

Table 2.6: Performance of models after concatenating per-class accuracy information to the input vector for the tracing model. Method details are described in (Sec. 2.7.6). We only compare model variants that appear in the main text. We observed a boost for all models.

	Greebles				**Eyes**				**Butterflies**			
	Train		Test		Train		Test		Train		Test	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
ϕ_{static}†	0.63	0.52	0.67	0.58	0.60	0.59	0.67	0.68	0.55	0.53	0.64	0.61
$\phi_{static+perClAcc}$	0.65	0.54	0.69	0.58	0.63	0.62	0.69	0.70	**0.59**	**0.57**	**0.66**	**0.65**
ϕ_{direct}†	0.70	0.59	0.77	0.64	0.66	0.65	0.75	0.74	0.55	0.53	0.60	0.57
$\phi_{direct+perClAcc}$	**0.71**	**0.62**	0.78	**0.67**	**0.69**	**0.69**	0.78	0.78	0.53	0.51	0.61	0.57
ϕ_{cls_pred}†	**0.71**	0.62	0.77	0.65	0.65	0.65	0.74	0.74	0.54	0.52	0.60	0.57
$\phi_{cls\_pred+perClAcc}$	**0.71**	0.61	**0.79**	**0.67**	**0.69**	**0.69**	**0.79**	**0.79**	0.53	0.51	0.60	0.58

Table 2.7: Effect of embedding dimension on model performance. We train the ϕ_{direct} tracing model on the butterflies dataset with different embedding dimensions. We find that embedding dimension has no impact on performance.

	Butterflies								
	Train							Test	
	Cabbage White	Monarch	Queen	Red Admiral	Viceroy	Micro	Macro	Micro	Macro
ϕ_{direct_dim8}	0.98±0.01	0.41±0.03	0.45±0.06	0.77±0.07	0.35±0.03	0.70±0.03	0.59±0.03	0.77±0.03	0.65±0.04
ϕ_{direct_dim16}	0.98±0.01	0.42±0.05	0.46±0.04	0.77±0.07	0.37±0.03	0.70±0.02	0.60±0.02	0.78±0.03	0.66±0.03
ϕ_{direct_dim32}	0.98±0.01	0.43±0.04	0.46±0.03	0.77±0.06	0.35±0.02	0.70±0.02	0.60±0.01	0.78±0.02	0.67±0.02
ϕ_{direct_dim64}	0.97±0.01	0.43±0.03	0.48±0.04	0.78±0.06	0.37±0.03	0.71±0.02	0.61±0.02	0.79±0.01	0.66±0.01

2.8 CNN Architecture Details

In Table 2.8 we describe the architecture of the CNN used to encode images for all of the models.

Table 2.8: Structure of the CNN backbone used to learn the image representation. The bolded and italicized entries are variable and depend on the experiment and dataset. The number of image channels (***img_chns***) is three for the Butterflies and Greebles dataset, but is one for Eyes. The Butterflies and OCT datasets contain larger images (144 x 144), and so ***img_feats*** is set to 1296. For the Greebles dataset, the images are (128 x 128) and ***img_feats*** is set to 1204. Finally, the output of the model is the size of the embedding dimension and is set to 16 for all experiments.

CNN backbone						
layer	**in channels**	**out channels**	**k**	**s**	**p**	**activation**
conv1	***img_chns***	8	5	1	2	PReLU
maxpool1	-	-	4	-	-	PReLU
conv2	8	16	5	1	2	PReLU
maxpool2	-	-	4	-	-	-
flatten	-	-	-	-	-	-
linear	***img_feats***	512	-	-	-	PReLU
linear	512	256	-	-	-	PReLU
linear	256	256	-	-	-	PReLU
linear	256	16	-	-	-	PReLU

2.9 Additional Results

We recreate Fig. 2.4 for all datasets and include results from the Direct Response and Time-Sensitive Model. For the Greebles dataset, we include the histograms of the features of the classes to demonstrate the difficulty of the task.

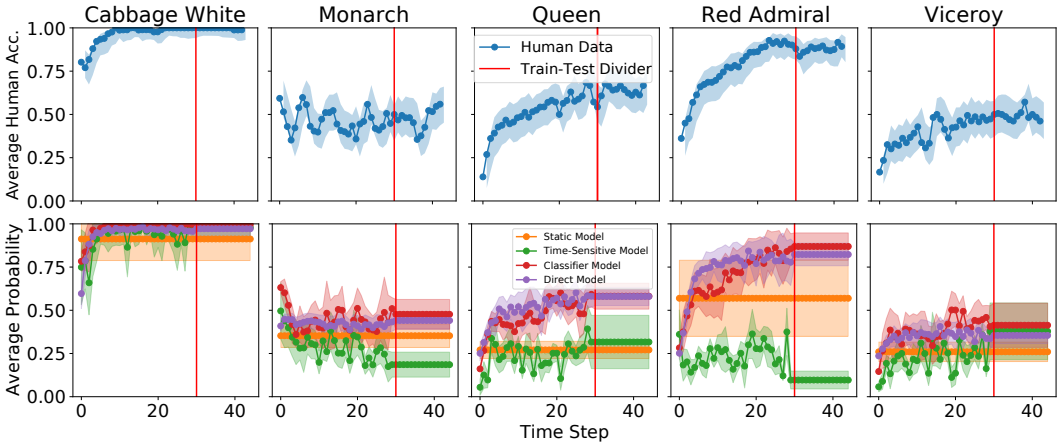


Figure 2.5: Human and model performance on the Butterflies dataset. (Top) The smoothed average human learner accuracy for each class over time on the Butterflies dataset. The shadowed regions indicate confidence intervals as the number of samples in each time and class bin are not guaranteed to be the same. **(Bottom)** The average probability of getting a class correct predicted by the static ϕ_{static} model (orange), ϕ_{static_time} model (green), the direct response ϕ_{direct} model (purple), and the classifier prediction ϕ_{cls_pred} model (red). At each time-step, for each learner in the test set, the models predict class probabilities for $\sim$ 50 images per class. The probabilities are averaged (solid line) and the shadows indicate one standard deviation. While both recurrent models have similar traces, the ϕ_{direct} produces smoother average probabilities.

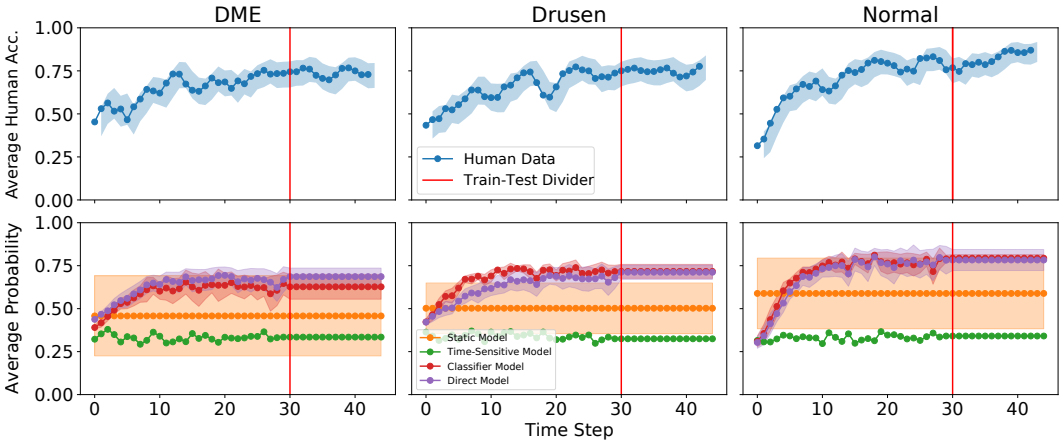


Figure 2.6: Human and model performance on the Eyes dataset. See Fig. 2.5 for a detailed caption.

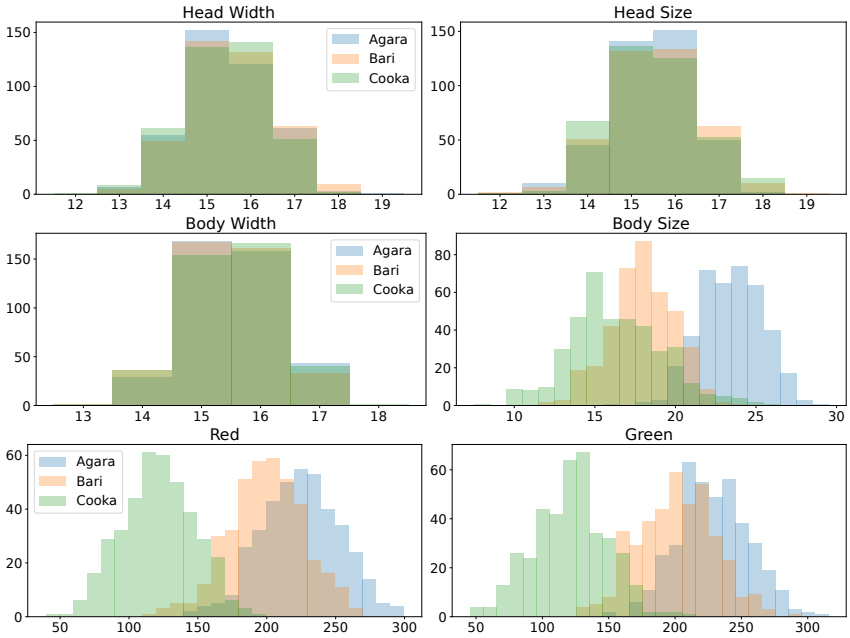


Figure 2.7: Statistics of the Greebles dataset. The Greebles dataset was inspired by the one used in [26]. In our version, the three classes vary in Head Width and Size (top row), Body Width and Size (middle row), and the Red and Green channel for the RGB color (bottom row). The histograms overlap completely for Head Width, Head Size, and Body Width. These variations serve as distractors since they provide no information about which class an image belongs to. The other features, Body Size, the Red channel, and the Green channel have different distributions and can be used to estimate the class. Agara and Bari are most separable by Body Size, Cooka is most separable from both Agara and Bari in the two color channels. However, note that they are not perfectly separated and it is possible, although less likely, for two images from different classes to take on the same properties. This makes the Greebles dataset particularly challenging, since the important features are both subtle and imperfect for distinguishing between classes.

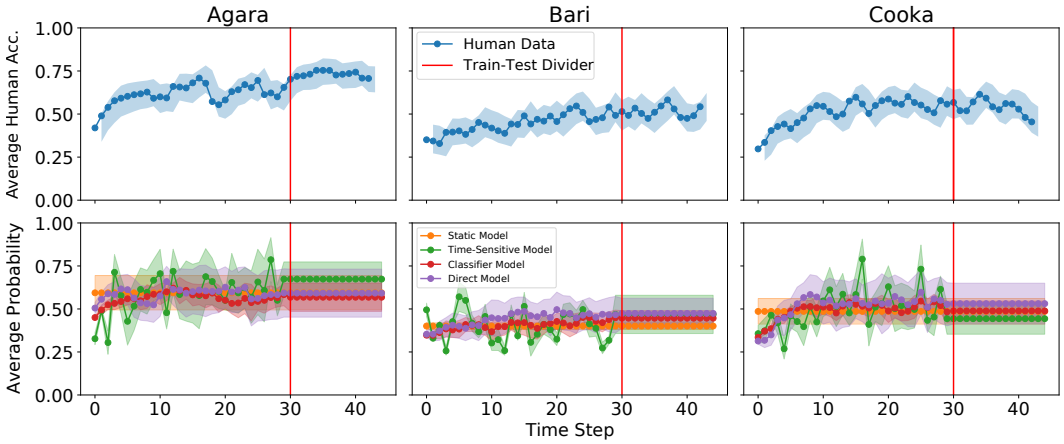


Figure 2.8: Human and model performance on the Greebles dataset. See Fig. 2.5 for a detailed caption.

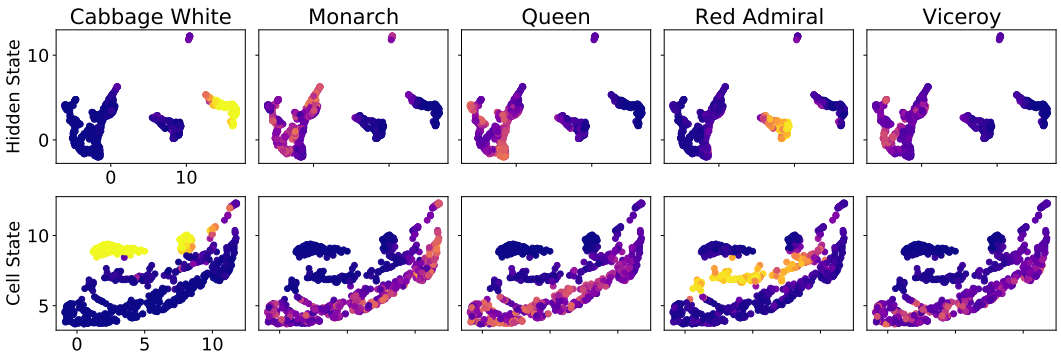


Figure 2.9: Visualizing hidden and cell states of the tracing model. The hidden states and cell states of the LSTM for ϕ_{cls_pred} while tracing 25 test-set learners are plotted in 2D using the UMAP dimensionality reduction algorithm [61]. **(Top)** The hidden state representations are colored according to the probability (purple to yellow) that a response produced with that hidden state would correctly classify an image of the class in the panel title. We can see that the classes that correspond to the best average performance by humans are in well-defined clusters (e.g. Cabbage White), whereas the classes that are commonly mistaken for each other are grouped together and have much weaker probabilities of being correct. **(Bottom)** The cell states are visualized in the same manner. For the cell states, we can see that the clusters seem to be dragged across a single dimension. The Cabbage White and Red Admiral cluster is split in two pieces in the cell state, which we explore in Fig. 2.10.

2.10 Learned Representations

Here we explore the representations learned by the the classifier prediction model (ϕ_{cls_pred}) on the Butterflies dataset. In Fig. 2.9 we visualize the internal state of the model and in Fig. 2.10 we provide an in depth comparison for two different learners.

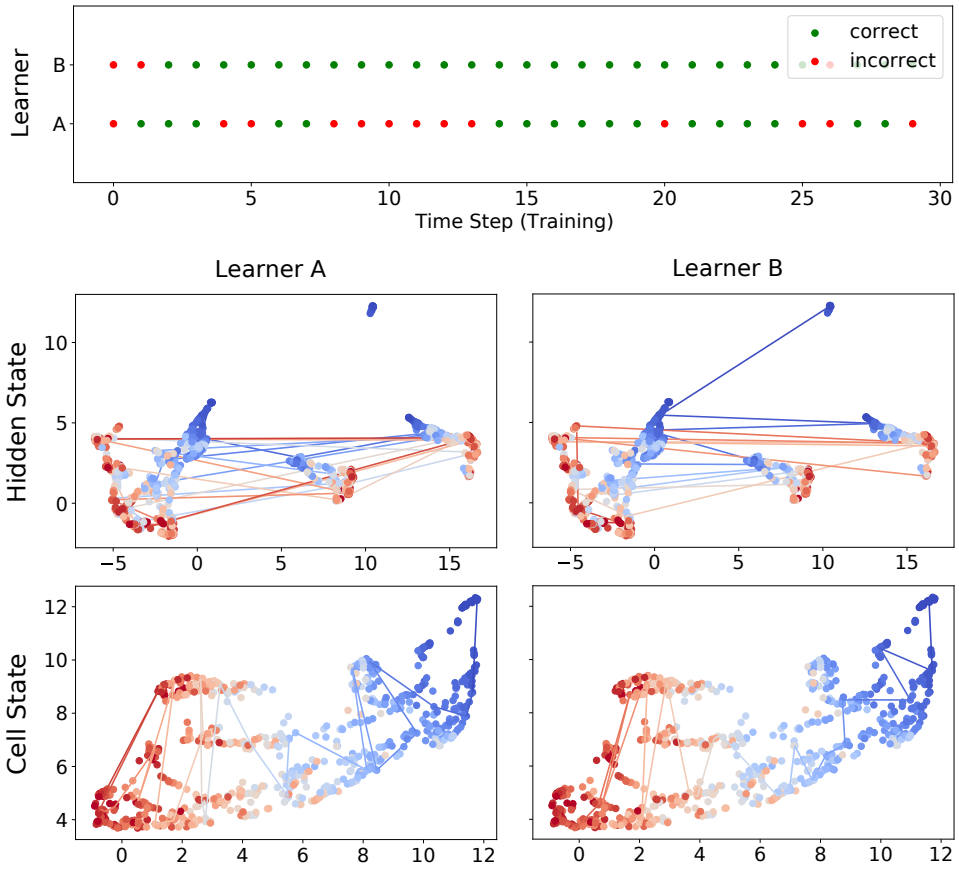


Figure 2.10: Tracing individual student performance. **(Top)** The sequence of correct and incorrect responses made by two human learners during training. We selected these two learners as they demonstrate different learning behaviors. It seems Learner B may already be familiar with butterflies. **(Bottom)** We overlay each learners' trajectory through the hidden and cell states. The colors represent the time-step, where dark blue is the beginning of training, light-grey is the middle, and dark red is the end. We see that Learner B's trajectory quickly skips to the left of the cell state, suggesting the LSTM encodes the learners skill level on all classes in certain dimensions of the cell state and uses the hidden state to translate the skill level into an appropriate response for the image shown to the learner.

2.11 Feature Space

In Fig. 2.11 we visualize the feature space learned by the CNN for the classifier prediction model (ϕ_{cls_pred}) on the Butterflies dataset.

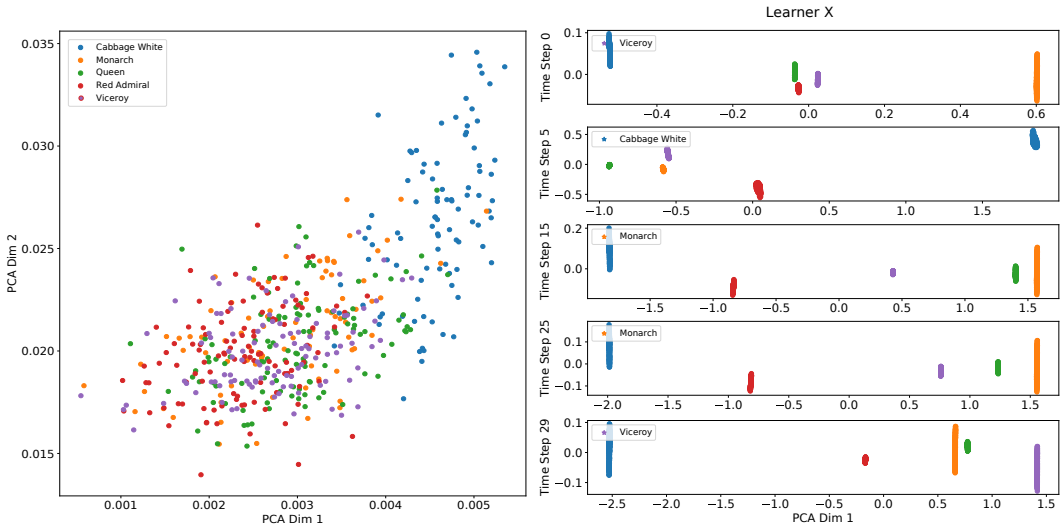


Figure 2.11: Visualizing the feature space. The feature space learned by the CNN must support several types of behaviors, since behavior changes between learners and over time. We use PCA to reduce the learned feature space into two dimensions. **(Left)** We show a subset of images in the Butterflies dataset colored by the ground truth label. Aside from the Cabbage White class, which is the easiest to identify, the representations are difficult to separate. **(Right)** We use the hyperplanes predicted while tracing a single learner X to induce a subspace and visualize the features in that subspace. Within the subspaces, the classes are much better separated. Each row shows a subspace induced by a hyperplane for different time-steps - where the time-step is indicated on the left. The colors represent the class and the labelled color is the target class for the image being evaluated in that time-step. We see that, over time, the target class is pushed further to the right and is better separated from the other classes (see orange cluster in time-step 15 vs. 25). Classes that are confused for each other have less separation, whereas classes like Cabbage White, that are rarely confused, are extremely well-separated from the other classes. Also, note that the subspace orientation (target class moved to the right) matches how the dot product between the hyperplanes and features is translated into probabilities in the model.

2.12 Additional Implementation Details

2.12.1 Types of Learner Responses

There are several ways to request information from the learner: they can provide their best guess, a ranked list of guesses, or confidence scores for each class. In these datasets, we ask for a ranking of each learner's top 3 classes as a balance between time-spent and informativeness. While our models are trained on their top choice (equivalent to their best guess), we hypothesize that the extra information available in the ranked responses can be leveraged to improve response prediction performance. We leave this to future work.

2.12.2 Recurrent Neural Networks

Here we elaborate on the details of the recurrent neural network based models.

Direct Response Model. At each time-step, this model receives the hidden states, cell states, the learner's response to the previous interaction, the embedding of the current image, and the true label of the current image. The model predicts the response of the learner with respect to the input image.

Classifier Prediction Model. At each time-step, this model receives the hidden state, cell state, the embedding of the image from the previous interaction, the learner's response to the previous interaction, and the true label of the current image. The model predicts a classifier that is used to classify the embedding of the input image such that it matches the response of the learner to that input image at that time-step.

Ground truth labels and learner responses are represented as one-hot-encoded vectors. For both models, at the first time-step, some of the values that make up the input vector are not available to the model. For example, the hidden state, cell state, the response to previous interaction, etc. These vectors are initialized to zeros.

References

[1] Neehar Kondapaneni, Pietro Perona, and Oisin Mac Aodha. "Visual Knowledge Tracing." In: *European Conference on Computer Vision*. Springer. 2022, pp. 415–431. URL: https://link.springer.com/chapter/10.1007/978-3-031-19806-9_24.

[2] Ellen M Markman. *Categorization and naming in children: Problems of induction*. mit Press, 1989.

[3] Fei Xu and Joshua B Tenenbaum. "Word learning as Bayesian inference." In: *Psychological review* (2007).

[4] Zhirong Wu et al. "Unsupervised feature learning via non-parametric instance discrimination." In: *CVPR*. 2018.

[5] Ting Chen et al. "A simple framework for contrastive learning of visual representations." In: *ICML*. 2020.

[6] Kaiming He et al. "Momentum contrast for unsupervised visual representation learning." In: *CVPR*. 2020.

[7] Elijah Cole et al. "When does contrastive visual representation learning work?" In: *CVPR*. 2022.

[8] Qi Liu et al. "A survey of knowledge tracing." In: *arXiv:2105.15106* (2021).

[9] Chris Piech et al. "Deep knowledge tracing." In: *Advances in neural information processing systems* 28 (2015).

[10] Shalini Pandey and George Karypis. "A self-attentive model for knowledge tracing." In: *arXiv:1907.06837* (2019).

[11] Shi Pu et al. "Deep knowledge tracing with transformers." In: *International Conference on Artificial Intelligence in Education*. 2020, pp. 252–256.

[12] Mahmut Kaya and Hasan Şakir Bilge. "Deep metric learning: A survey." In: *Symmetry* (2019).

[13] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*. 11. Sage, 1978.

[14] Laurens Van Der Maaten and Kilian Weinberger. "Stochastic triplet embedding." In: *International Workshop on Machine Learning for Signal Processing*. 2012, pp. 1–6.

[15] Michael Wilber, Iljung Kwak, and Serge Belongie. "Cost-effective hits for relative similarity comparisons." In: *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*. Vol. 2. 2014, pp. 227–233.

[16] Brett D Roads and Michael C Mozer. "Predicting the ease of human category learning using radial basis function networks." In: *Neural Computation* (2021).

[17] Omer Tamuz et al. "Adaptively Learning the Crowd Kernel." In: *ICML*. 2011.

[18] Karsten Roth et al. “Revisiting training strategies and generalization performance in deep metric learning.” In: *ICML*. 2020, pp. 8242–8252.

[19] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. “A metric learning reality check.” In: *ECCV*. 2020, pp. 681–699.

[20] Jiang Wang et al. “Learning fine-grained image similarity with deep ranking.” In: *CVPR*. 2014, pp. 1386–1393.

[21] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “Facenet: A unified embedding for face recognition and clustering.” In: *CVPR*. 2015, pp. 815–823.

[22] Andreas Veit, Serge Belongie, and Theofanis Karaletsos. “Conditional similarity networks.” In: *CVPR*. 2017, pp. 830–838.

[23] Kun Ho Kim, Oisin Mac Aodha, and Pietro Perona. “Context Embedding Networks.” In: *CVPR*. 2018, pp. 8679–8687.

[24] Ishan Nigam, Pavel Tokmakov, and Deva Ramanan. “Towards latent attribute discovery from triplet similarities.” In: *ICCV*. 2019, pp. 402–410.

[25] Reuben Tan et al. “Learning similarity conditions without explicit supervision.” In: *ICCV*. 2019, pp. 10373–10382.

[26] Peter Welinder et al. “The multidimensional wisdom of crowds.” In: *NeurIPS* (2010).

[27] Dagmar Zeithamova. *Category learning systems*. The University of Texas at Austin, 2008.

[28] F Gregory Ashby and W Todd Maddox. “Human category learning 2.0.” In: *Annals of the New York Academy of Sciences* (2011).

[29] F Gregory Ashby and Jeffrey B O’Brien. “Category learning and multiple memory systems.” In: *Trends in cognitive sciences* 9.2 (2005), pp. 83–89.

[30] F Gregory Ashby and W Todd Maddox. “Human category learning.” In: *Annu. Rev. Psychol.* 56 (2005), pp. 149–178.

[31] Irving Biederman and Margaret M Shiffrar. “Sexing day-old chicks: a case study and expert systems analysis of a difficult perceptual-learning task.” In: *Journal of Experimental Psychology: Learning, memory, and cognition* 13.4 (1987), p. 640.

[32] Martin N Hebart et al. “Revealing the multidimensional mental representations of natural objects underlying human similarity judgements.” In: *Nature human behaviour* 4.11 (2020), pp. 1173–1185.

[33] Brett D Roads and Bradley C Love. “Enriching imagenet with human similarity judgments and psychological embeddings.” In: *CVPR*. 2021.

[34] Robert M Nosofsky et al. “Toward the development of a feature-space representation for a complex natural category domain.” In: *Behavior Research Methods* 50.2 (2018), pp. 530–556.

[35] Maria Attarian, Brett D Roads, and Michael C Mozer. "Transforming neural network visual representations to predict human judgments of similarity." In: *arXiv:2010.06512* (2020).

[36] Daniel N Barry and Bradley C Love. "Human learning follows the dynamics of gradient descent." In: *PsyArXiv* (2021).

[37] Albert T Corbett and John R Anderson. "Knowledge tracing: Modeling the acquisition of procedural knowledge." In: *User modeling and user-adapted interaction* 4 (1994), pp. 253–278.

[38] Xiaojin Zhu et al. "An overview of machine teaching." In: *arXiv:1801.05927* (2018).

[39] Chi-Tung Cheng et al. "Artificial intelligence-based education assists medical students' interpretation of hip fracture." In: *Insights into Imaging* 11.1 (2020), pp. 1–8.

[40] Elnaz Amiri, Patty Sha, and Evan M Palmer. "Training Novices to Discriminate Retinal Diseases Using Perceptual Learning." In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. 2020, pp. 1456–1460.

[41] Adish Singla et al. "Near-optimally teaching the crowd to classify." In: *ICML*. 2014, pp. 154–162.

[42] Edward Johns, Oisin Mac Aodha, and Gabriel J Brostow. "Becoming the expert-interactive multi-class machine teaching." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2616–2624.

[43] Oisin Mac Aodha et al. "Teaching categories to human learners with visual explanations." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3820–3828.

[44] Pei Wang, Kabir Nagrecha, and Nuno Vasconcelos. "Gradient-based algorithms for machine teaching." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1387–1396.

[45] Pei Wang and Nuno Vasconcelos. "A Machine Teaching Framework for Scalable Recognition." In: *ICCV*. 2021, pp. 4945–4954.

[46] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[47] Daniel S Kermany et al. "Identifying medical diagnoses and treatable diseases by image-based deep learning." In: *Cell* 172.5 (2018), pp. 1122–1131.

[48] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context." In: *ECCV*. 2014.

[49] Grant Van Horn et al. "Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection." In: *CVPR*. 2015.

[50] Alina Kuznetsova et al. “The open images dataset v4.” In: *IJCV* (2020).

[51] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge.” In: *IJCV* (2015).

[52] *Prolific*. `www.prolific.co`, accessed Mar 7 2022.

[53] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization.” In: *arXiv:1412.6980* (2014).

[54] Robert M Nosofsky, Brian Meagher, and Parhesh Kumar. “Contrasting Exemplar and Prototype Models in a Natural-Science Category Domain.” In: *CogSci*. 2020.

[55] Youngduck Choi et al. “Towards an appropriate query, key, and value computation for knowledge tracing.” In: *Proceedings of the Seventh ACM Conference on Learning@ Scale*. 2020.

[56] Ashish Vaswani et al. “Attention is all you need.” In: *NeurIPS* (2017).

[57] Burr Settles and Brendan Meeder. “A trainable spaced repetition model for language learning.” In: *ACL*. 2016.

[58] Anette Hunziker et al. “Teaching multiple concepts to a forgetful learner.” In: *NeurIPS* (2019).

[59] Kaiming He et al. “Deep residual learning for image recognition.” In: *CVPR*. 2016.

[60] Dongmin Shin et al. “Saint+: Integrating temporal features for ednet correctness prediction.” In: *LAK21: 11th International Learning Analytics and Knowledge Conference*. 2021.

[61] Leland McInnes et al. “UMAP: Uniform Manifold Approximation and Projection.” In: *The Journal of Open Source Software* (2018).

Chapter 3

REPRESENTATIONAL SIMILARITY VIA INTERPRETABLE VISUAL CONCEPTS

Neehar Kondapaneni, Oisin Mac Aodha, and Pietro Perona. "Representational Similarity via Interpretable Visual Concepts." In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=ih3BJmIZbC.

3.1 Abstract

How do two deep neural networks differ in how they arrive at a decision? Measuring the similarity of deep networks has been a long-standing open question. Most existing methods provide a single number to measure the similarity of two networks at a given layer, but give no insight into *what* makes them similar or dissimilar. We introduce an interpretable representational similarity method (RSVC) to compare two networks. We use RSVC to discover shared and unique visual concepts between two models. We show that some aspects of model differences can be attributed to unique concepts discovered by one model that are not well represented in the other. Finally, we conduct extensive evaluation across different vision model architectures and training protocols to demonstrate its effectiveness. Code: github.com/nkondapa/RSVC

3.2 Introduction

The accuracy of deep neural networks has steadily increased over the last few years thanks to improvements in model architectures, dataset size, and pretraining strategies. However, much less is understood regarding *how* the representations of different models have changed to make the models more effective. Thus, there is growing interest in developing methods that allow practitioners to compare different networks. Comparing the activation matrices of two neural networks over the same set of inputs underpins current *representational similarity* methods, e.g. CCA [2], CKA [3], RSA [4], and Brain-score [5]. While these approaches provide a score denoting the similarity between two different models, they do not identify the specifics of what makes two models' computations similar or dissimilar, and what aspects of a representation lead to differences in model decisions.

In parallel, methods for concept-based eXplainable AI (XAI) have improved our ability to understand what features individual models use to arrive at decisions. Understanding these features is critical for ensuring model fairness and identifying potential sources of bias [6, 7]. In general, XAI methods sacrifice model fidelity to produce explanations that are simple enough for human interpretation [8, 9]. Thus, there is a tension between model fidelity and human understanding.

We propose that contrasting two models is an effective way to identify and highlight what makes a model unique, so that users can identify critical features that drive differences in model behavior. To investigate this idea we develop a new approach that extends concept-based XAI methods so they can quantitatively measure representational similarity and provide interpretable insights into the differences between models. We name our method *Representational Similarity via interpretable Visual Concepts* (RSVC). Our method builds on interpretability approaches in which the activations of a given layer are decomposed into a coefficient matrix and a vector basis. These approaches group images that produce similar activation patterns together. Thus, we define a visual concept to be the equivalence class of images that produce a similar activation pattern. After computing visual concepts from each model, our method evaluates whether the visual concepts from one model are also used by the other model. We make three contributions:

- RSVC, a new approach for providing human-interpretable insights into model differences (Fig. 6.1).
- A validation strategy to link representational differences to model decisions.
- Experiments to show that RSVC can measure similarity at both coarse- and fine-grained levels.

3.3 Related Work

3.3.1 Representational Similarity

Similarity methods attempt to quantify the similarity/dissimilarity between pairs of different models [2, 3, 10, 11, 12]. Models can be compared based on their *functional* (i.e. how their outputs differ) or their *representational* (i.e. how the features activations of intermediate layers differ) similarity [13]. While functional similarity can tell us about how model outputs vary, two models can achieve the same performance with significantly different representations. These differences matter, e.g. while two different forms of pretraining can achieve similar performance on certain datasets, they may transfer poorly to others [14].

Representational similarity metrics have successfully been used to analyze the differences between architectures [15, 16], explore the effects of different kinds of pretraining [14, 17, 18], develop novel strategies for efficient ensembling [19], or perform ensembling that is robust to distribution shifts [20]. Some methods have leveraged representational similarity to build tools for text-to-image generation [21] or model-to-model translation [22]. In neuro- and cognitive science, representational similarity is used to measure how well models are able to approximate the neural recordings of the brain and/or the behavior of humans [4, 5, 23, 24, 25]. In the disentanglement literature, representational similarity has been used to evaluate how well the learned representation matches known ground truth latent factors. A popular approach is to use regularized linear predictors to map learned factors to ground truth latent factors [26, 27, 28, 29, 30]. Recently, [22] proposed a method that uses correlated activation patterns across networks to mine for "Rosetta Neurons." These neurons provide insights about features that re-occur consistently in many models. In contrast to our proposed method, Rosetta Neurons do not quantify the overall similarity between networks and do not identify neurons that explain model differences. Most closely related to our work is that of [5] and [11] which use a linear regression model as a similarity metric between the activations of two networks. We describe these methods in more detail in Sec. 3.4.2.

The primary limitation of most existing representational similarity methods is that they can quantify how similar the representations of two models are, but can not tell users *what* makes the models similar or dissimilar. We propose a new approach that aims to address this question. Our approach leverages concept-based explainability as an intermediate step to measure representational similarity.

3.3.2 Explainable AI (XAI)

XAI methods aim to answer the questions (1) *what* features did a model use to arrive at a decision and (2) *where* is the relevant information in the input. Local explanation methods focus on pixel-based attribution, in which a heatmap indicates the region of the image that is most relevant to the model's decision [31, 32, 33]. While local explanations are able to answer the "where" question, it can be challenging to interpret "what" is being highlighted as attributions can be noisy.

To better address the "what" question, global concept-based explanations can be used [8, 34, 35, 36, 37, 38, 39]. These approaches discover groups of images (or

image regions) that share some visual feature that is relevant to the model's decision making. In addition, "glocal" methods have been developed to answer both questions simultaneously [40, 41, 42, 43].

[44] take a different approach to explaining model decisions whereby linear surrogate models are trained to reproduce the output of a deep neural network. Then, each training image is given a score that reflects how much it contributed to the final weights of the learned model. In [45], this approach was extended to analyze the differences between two models by identifying the differences in training images that each model relies on to arrive at different decisions. While our work also aims to compare two models in an interpretable manner, our approach uses the activations of the original model itself. This gives RSVC the advantage of being able to link representational changes to functional model behavior.

3.4 Method

We propose a new approach to compare the representations of two models using concepts. Our approach is closely related to methods for computing representational similarity, such as BrainScore [5] and the metric in [11], and builds upon prior work in concept extraction [8, 35, 41]. We bridge approaches from these two fields, resulting in an interpretable method to measure representational similarity for deep neural networks (Fig. 6.1).

3.4.1 Concept Extraction

[8] showed that many concept based explainability approaches can be generalized as dictionary learning methods. In these approaches, a set of n input images $\mathcal{I}$ is used to compute activations from a specific layer l of a neural network resulting in an activation matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, where each row is an activation vector of dimensionality d. Then, a dictionary learning algorithm can be used to approximate the activations as $\mathbf{A} \approx \mathbf{UW}$. The row vectors of the vector basis $\mathbf{W} \in \mathbb{R}^{k \times d}$ can be interpreted as a set of k concepts. Similarly, the rows of the coefficient matrix $\mathbf{U} \in \mathbb{R}^{n \times k}$ represent the importance of a particular concept vector in $\mathbf{W}$ for a given image. By visualizing the images that have the largest concept coefficients for a given concept, we are able to understand *what* visual concepts the network has identified in the data. The effectiveness of such concept based XAI methods has been demonstrated via user studies in previous work [34, 35, 41].

Our goal is to develop a method to compare the representational similarity between two models, not just in terms of a single numerical score, but via interpretable

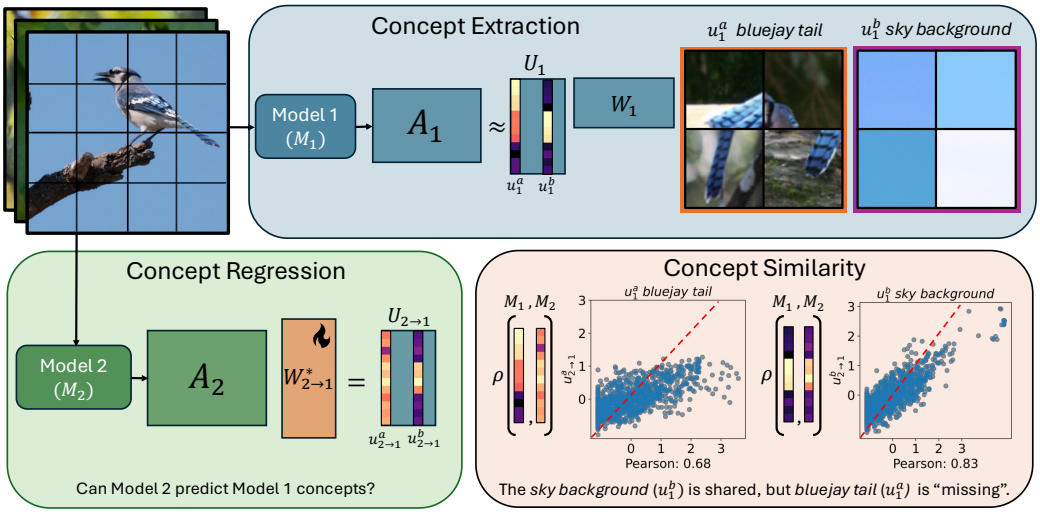


Figure 3.1: Representational Similarity via interpretable Visual Concepts (RSVC). (Concept Extraction): First, activations for a set of image patches, $\mathcal{I}^c$, are computed for each model (M_1 and M_2). Second, the activation matrix for M_1 is factorized into the *concept coefficient matrix* $\mathbf{U}_1$ and the *concept basis* $\mathbf{W}_1$, i.e. $\mathbf{A}_1 \approx \mathbf{U}_1\mathbf{W}_1$. Each entry in a column vector of the coefficient matrix $\mathbf{U}_1$ represents the strength of a concept in an image. Concepts are visualized by the image patches that correspond to the top n coefficients. Here, we highlight only two concepts, u_1^a and u_1^b. The top four images for these concepts indicate that u_1^a represents *bluejay tail* and u_1^b represents *sky background*. (Concept Regression): To measure concept similarity, we learn a weight matrix $\mathbf{W}^*_{2\to1}$ to map $\mathbf{A}_2$ to the concept coefficient matrix $\mathbf{U}_2$. We denote the predicted coefficient matrix as $\mathbf{U}_{2\to1}$. (Concept Similarity): Finally, we compute the correlation between columns of $\mathbf{U}_{2\to1}$ and $\mathbf{U}_1$. If $\mathbf{A}_2$ contains a concept in $\mathbf{U}_1$, then the predicted coefficient vector should be highly correlated to the real coefficient vector. In this example, we see that the *bluejay tail* concept is poorly represented in M_2, but both models share the *sky background* concept.

concepts. For each model, we use the concept extraction approach proposed in CRAFT [41]. We denote the first model as M_1, and the second as M_2. Importantly, we make no assumption that these two models are from the same model family, e.g. one could be a CNN and the other a Vision Transformer. We outline the concept extraction process for M_1, but the same process is applied to M_2.

For a specific object class c, we select the set of images $\mathcal{I}_1^c$ that M_1 predicted to contain class c. By grouping images according to model predictions, as opposed to only the ground truth labels, we are able to identify concepts used in all images that the model believes are part of class c. This allows the method to provide better insight into both correct and incorrect predictions. Images are usually composed of several visual concepts, so we extract evenly spaced patches from the image to form

a set of "concept proposal" images. These patches are more likely to contain a single visual concept and are easier to interpret. We re-use $\mathcal{I}_1^c$ to refer to the set of concept proposal images.

Each concept proposal is resized to the model's input resolution and passed through the network. Note that all networks are trained with cropping augmentation, such that patches are in the domain of training images. We denote the activations from a specific layer l and class c as $\mathbf{A}_1 \in \mathbb{R}^{|\mathcal{I}_1^c| \times d}$, where d is the dimension of the activations of the layer. We then use a dictionary learning algorithm with k components to decompose $\mathbf{A}_1$ into a matrix $\mathbf{U}_1 \in R^{|\mathcal{I}_1^c| \times k}$ and $\mathbf{W}_1 \in \mathbb{R}^{k \times d}$, such that $\mathbf{A}_1 \approx \mathbf{U}_1\mathbf{W}_1$. We refer to $\mathbf{U}_1$ as the *concept coefficient matrix* and $\mathbf{W}_1$ as the *concept basis*. We repeat this process for M_2, resulting in $\mathbf{U}_2$ and $\mathbf{W}_2$. Intuitively, each row of a concept coefficient matrix $\mathbf{U}$ encodes the contribution of each concept vector in $\mathbf{W}$ to the activation vector of a particular image in $\mathcal{I}$.

To measure concept similarity between two models, we need to understand how each concept reacts to the *same* set of images. For example, if both networks have discovered a concept that reacts to the color red, they would both have an activation pattern that spikes when red objects are presented. Following this logic, we propose that if two concept vectors are encoding the same information, their concept coefficients over the same set of images should be correlated. To obtain a shared set of images, we take the union over the image sets $\mathcal{I}_1^c \cup \mathcal{I}_2^c$ to form $\mathcal{I}^c$. The proposals are passed through the model M_1 to produce $\mathbf{A}_1$ for the shared concept set. Given the concept basis $\mathbf{W}_1$ (which is specific to M_1), we re-compute $\mathbf{U}_1$ over the shared set of images. We repeat this process for M_2 to compute $\mathbf{U}_2$. In practice, we use a non-negative least squares solver, since the original coefficient matrices are non-negative (Sec. 3.9).

3.4.2 Concept Similarity

Here we address the following question: in a specific pair of layers, does M_1 encode the same concepts as M_2? In the following sections, we make use of $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{U}_1$ and $\mathbf{U}_2$ to compute the similarity between concepts encoded in M_1 and M_2. We consider two different approaches that trade-off computational cost and error. In Sec. 3.8.1 we describe a correlation based metric that has a low computational cost that we use to coarsely compare many layers of two neural networks.

In order to more accurately measure similarity in a single layer, we propose a regression based metric similar to the strategy in [11] and the BrainScore [5], which

was originally introduced to compare artificial neural networks to biological neural networks. In [11], the outputs of a convolutional layer in one model are mapped to the outputs of a convolutional layer in another network using a sparse weight matrix. The prediction error between the predicted outputs and true outputs are used as a metric for the similarity between the two layers. In BrainScore, for a set of n stimuli (e.g. images), the activations from a layer of the DNN are stored in a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, where d is the dimensionality of the layer activations. For the same set of stimuli, neural recordings are measured from an animal and processed forming a vector $\mathbf{y} \in \mathbb{R}^n$ for each target "neuroid." A linear mapping is introduced to predict the neural responses from the DNN activations, $\mathbf{y} = \mathbf{A}\mathbf{w}^* + b$, where $\mathbf{w}^*$ are the weights of the regressor and b is the bias. The regression model is trained on a set of training images and evaluated on held out test images where the predicted outputs of the regressor are compared to the true neural responses using Pearson correlation, giving a score between -1 and 1.

To compare two neural networks using these methods, each column of $\mathbf{A}_2$ would serve as prediction targets for regression with $\mathbf{A}_1$ resulting in a similarity score for each column of $\mathbf{A}_2$. However, visualizing and interpreting each neuron results in an explanation that is too complex for users. Instead, a more interpretable result can be achieved by setting the coefficient matrix $\mathbf{U}_2$ as regression targets for $\mathbf{A}_1$ (Fig. 6.1). Essentially, RSVC encodes similarity by measuring how well M_1 can predict the concept coefficients of M_2 and vice-versa,

$$\mathbf{A}_1\mathbf{W}^*_{1\to 2} = \mathbf{U}_{1\to 2} \quad \text{and} \quad \mathbf{A}_2\mathbf{W}^*_{2\to 1} = \mathbf{U}_{2\to 1}, \tag{3.1}$$

where we learn $\mathbf{W}^*$ such that the following regularized (l_1) mean-squared error is minimized.

$$\min_{\mathbf{W}^*} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{A}\mathbf{W}^* - \mathbf{U})^2 + \lambda \|\mathbf{W}^*\|_1. \tag{3.2}$$

l_1 regularization guides the regression model to seek a sparse set of neurons in M_1 that can be used to predict U_2 and reduces over-fitting to the regression training data. We also compute baselines for each model from their own activation matrices, learning $\mathbf{W}^*_{1\to 1}$ and $\mathbf{W}^*_{2\to 2}$. Finally, we compute the Pearson and Spearman correlation between columns of the predicted coefficient matrix and columns of the true coefficient matrix to get a similarity score for each concept between -1 and 1. We refer to the score as cross-model concept similarity (CMCS) when computed across two models, and as same-model concept similarity (SMCS) when computed within the same model.

Finally, we investigate whether similar or dissimilar concepts are more important to model decisions by applying concept integrated gradients [8]. Concept integrated gradients measure the contribution of each concept to a model's decision (Sec. 3.9.1).

3.4.3 Replacement Test

To link differences in predicted concept coefficients and real concept coefficients to model behavior, we conduct a "replacement test."

for $i = 1$ to K **do**
  $\bar{\mathbf{U}}^i_{2\to1} \leftarrow \text{Copy}(\mathbf{U}_1)$
  $\bar{\mathbf{U}}^i_{2\to1}[:, i] \leftarrow \mathbf{U}_{2\to1}[:, i]$
  $\bar{\mathbf{A}}^i_{2\to1} \leftarrow \bar{\mathbf{U}}^i_{2\to1} W_1$
  $\bar{\mathbf{z}}^i_{2\to1} \leftarrow h(\bar{\mathbf{A}}^i_{2\to1})$
  $\bar{\mathbf{y}}^i_{2\to1} \leftarrow \text{argmax}(\bar{\mathbf{z}}^i_{2\to1})$
end for

For each model comparison, we conduct a replacement operation over each column of the coefficient matrix and keep track of the resultant reconstructed activation matrix, model logits, and model predictions (as seen in the pseudocode on the left). We perform the replacement operation using the same model predicted coefficients $\mathbf{U}_{1\to1}$ (baseline) and also the cross-model predicted coefficients $\mathbf{U}_{2\to1}$. We measure the impact of replacement at three levels during a classification task. We denote h as the classification head that produces logits ($\mathbf{z}$) for each input image. We compute the mean l_2-distance between each row of $\bar{\mathbf{A}}^i_{2\to1}$ and $\bar{\mathbf{A}}^i_{1\to1}$, the mean KL-divergence over the logits $\bar{\mathbf{z}}^i_{2\to1}$ and $\bar{\mathbf{z}}^i_{1\to1}$, and the match accuracy between $\bar{\mathbf{y}}^i_{2\to1}$ and $\bar{\mathbf{y}}^i_{1\to1}$.

3.4.4 Interpreting Low Similarity Concepts

How should we visually compare predicted and real concepts? Concept based XAI methods like CRAFT [41] and CRP [42] visualize the n images with the largest concept coefficient as representatives of the visual feature encoded by the concept. The same approach can be used to visualize similar concepts, since, by definition, similar concepts have highly correlated activation patterns over the shared set of concept proposal images. However, this approach is misleading when low similarity concepts are discovered. When a concept is dissimilar it may share the same top n images, but have entirely uncorrelated coefficients over the remaining images. This possibility is further amplified due to the mean squared error (MSE) loss used to estimate the regression matrix $\mathbf{W}^*_{1\to2}$. The MSE loss penalizes prediction error on the largest coefficients disproportionately, leading to a higher chance the two models share the same top n images.

To address this, we develop a new approach to specifically visualize how two models are dissimilar with respect to a concept. We start by visualizing the target concept by

using an image collage corresponding to the top n concept coefficients. The target concept is compared to the predicted concept by visualizing the top n over-predicted coefficients and top n under-predicted coefficients. This allows users to reason about visual features that one model entangles or disentangles with the target concept, improving their overall understanding of the compared models. To ensure sample diversity, we enforce that we visualize one patch per image. We also exclude the top-10 real concept images when selecting the over and under predicted points. In Figs. 3.2 and 3.5, we demonstrate our proposed approach for interpreting the dissimilarity of the concepts. We include more visualizations in Sec. 3.8.2.

3.4.5 Implementation Details

We choose a ResNet-18 (RN18), ResNet-50 (RN50) [46], ViT-S, ViT-L [47], DINO ViT-B (DINO) [48], and MAE ViT-B (MAE) [49] from the timm library [50] for our experiments. All models were trained on ImageNet [51]. For our exploration with DINO and MAE, we finetune the models on NABirds [52]. We compare four pairs of models: RN18 vs. RN50, ViT-S vs. ViT-L, RN50 vs. ViT-S, and DINO vs. MAE. Model performance on their respective datasets are reported in Tab. 4.1. The first two pairs have a clear difference in performance, implying that there are significant representational differences between the models. The second two pairs are roughly equal in overall performance, allowing us to explore how representational differences may result in different behavior even when overall performance is the same. For the l_1 penalty, we sweep λ on a subset of data and find $\lambda = 0.1$ to be a reasonable choice (Fig. 3.18). Additionally, we set the number of concepts $k = 10$ to balance reconstruction error and computational cost (Fig. 3.19). We provide further details on the concept extraction, concept comparison and computational cost in Sec. 3.9.

3.5 Results

3.5.1 Discovering A "Toy" Concept

While understanding models on real data is closely related to real-world use cases, it results in complex concepts that can be more challenging to interpret. To better understand the properties of RSVC, we design an experiment in which we train a model on images modified with a simple toy visual concept. In this experiment, we are able to control what the model learns and explore how RSVC works in a more controlled setting.

We train two ResNet-18 models from scratch on all classes from a modified NABirds dataset [52]. The first model, M_{ps}, is trained to make use of the toy concept in its

decisions and the second, M_{nc} is trained to become invariant to the toy concept. The toy concept is a $20px \times 20px$ pink square that is stochastically placed on the images at a random location. For M_{ps}, the concept appears on images from the Common Eider class with a 70% probability, giving the concept predictive power. For M_{nc}, the concept appears on images from any class with a 50% probability, giving the concept no predictive value. Thus, M_{ps} should learn to attend to the visual concept while M_{nc} should learn to ignore the concept, since it is simply noise. Finally, both models are tested on a dataset in which the concept appears on images from the Common Eider class with a 100% probability. Provided that M_{nc} successfully learns to ignore the concept, RSVC should have a low similarity score to any concept in M_{ps} that primarily fixates on the pink square. Recall that concepts are visualized using image patches. This allows us to break the 100% correlation between the pink square and the image during testing, such that only some patches contain the added concept. Both models achieve ~34% classification accuracy on the test set of NABirds.

In Fig. 3.2, we visualize the similarity from M_{nc} to Concept 1 in M_{ps}. We find that M_{nc} has a near zero similarity to Concept 1 in M_{ps}, which we visually identify to be a concept that fixates on the pink square. Importantly, the modified training paradigm does not affect the similarity scores between other concepts in the two models. In Fig. 3.10, we show that M_{nc} has high similarity to two other concepts in M_{ps}. Thus, we show that RSVC clearly identifies the primary conceptual difference between the two models in this controlled experiment.

3.5.2 Concept Similarity vs. Concept Importance

To analyze real data, we start by exploring the relationship between concept similarity and importance in the penultimate layers of different models trained on ImageNet [51]. We compute the cross-model concept similarity (CMCS) and the concept importance (CI) for every extracted concept. In Fig. 3.3, we plot the concept similarity from M_1 to M_2 against the concept importance for M_2 across four pairs of models. In Sec. 3.8.7, we compare same-model concept similarity (SMCS) to CI. In both of these figures, a point with low similarity and high importance would indicate that in layer l, M_1 can *not* predict the coefficients of a concept that M_2 finds important in decision making. We use color to indicate the density of points in a region (warmer colors indicate more density). As expected, we find that SMCS values are significantly higher than CMCS values, since the model is predicting its own concepts.

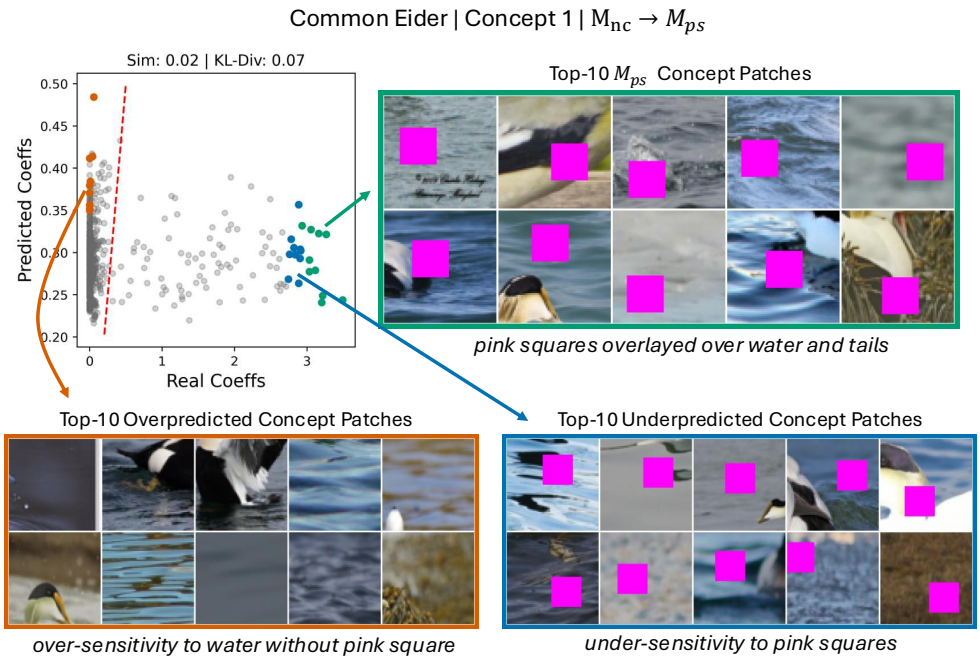


Figure 3.2: Adding and discovering a toy concept. Here we train two ResNet-18 models, M_{ps} and M_{nc}. M_{ps} is trained to associate a pink square (i.e. Concept 1) with the Common Eider class, while M_{nc} is trained to be invariant to the pink square concept. We find that the similarity score from $M_{nc} \rightarrow M_{ps}$ for Concept 1 is ~ 0.0, indicating that M_{nc} is unable to predict Concept 1 from M_{ps}. To understand various aspects of the differences between the two models, RSVC inspects three distinct regions of the predicted vs. real coefficient scatter plot (Sec. 3.4.4). (Green): RSVC visualizes images corresponding to the top-10 M_{ps} target concept coefficients. This allows the user to understand what the target concept is encoding. This concept clearly reacts strongly to the pink square visual feature. (Blue): RSVC visualizes the image patches with the largest M_{nc} under-predicted coefficients. M_{nc} under-reacts to the pink square when compared to M_{ps}. (Orange): RSVC visualizes the image patches corresponding to the top-10 M_{nc} over-predicted coefficients. The over-predicted patches show that M_{nc} cannot distinguish between background and the pink square.

In cross-model comparisons, we observe that models tend to have medium/high similarity for most concepts, since dense regions in the plot tend to be above 0.6 similarity. We also notice that the similarity scores from ViTs and ResNets have a different overall structure, with ResNets having a longer tail of low similarity and low importance concepts. Finally, except for DINO vs. MAE we find that there are several low/medium similarity concepts that also have a medium/high importance. In Sec. 3.8.5, we systematically vary the training protocol (seed and data) for a ResNet-18 model and measure the impact of the changes on model similarity. We

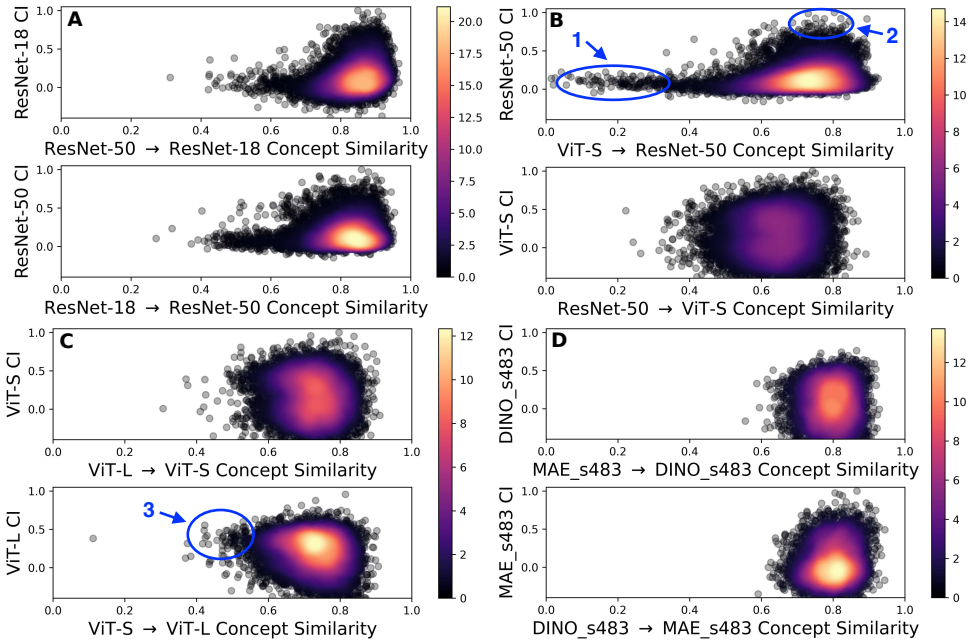


Figure 3.3: Concept similarity vs. concept importance. We compare four pairs of models using CMCS: (A) RN18 vs. RN50, (B) RN50 vs. ViT-S, (C) ViT-S vs. ViT-L, and (D) DINO vs. MAE. The y-axis represents the concept importance (CI) measured using concept integrated gradients. Warmer colors represent the density of points in a region. We highlight several regions in the plots: (1) low similarity and low importance concepts that are unique to a model but contribute little to its decisions, (2) high importance and high similarity concepts that are shared across both models and also contribute greatly to decision making, (3) low similarity, high importance concepts that only one model has discovered, but are very important to that model's decisions.

find that changes in model training lead to intuitive changes in similarity and use RSVC to reveal some concepts that suggest how the two models differ (Fig. 3.12). In summary, we observe two key results: (1) model differences are largely driven by medium similarity, medium importance concepts, jointly contributing to significant changes in model behavior and (2) some models do learn "unique" low similarity, high importance concepts. In the following sections we explore both of these results further.

3.5.3 Replacement Test

In order to better understand how variations in similarity impact model behavior we conduct a replacement test (described in Sec. 3.4.3). This test allows us to

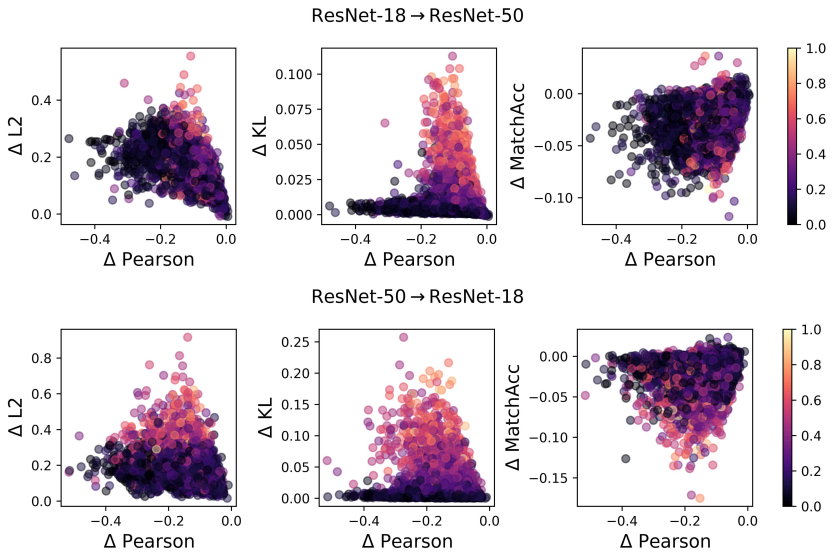


Figure 3.4: Replacement test results. We determine whether poorly predicted coefficients for concepts actually impact model behavior (Sec. 3.5.3). We use color to represent the concept importance (warmer is higher importance). When ignoring low importance concepts, we observe expected trends, i.e. decreases in similarity (ΔPearson) result in increases in the l_2-distance, increases in KL-divergence on the classifier logits, and decreases in model accuracy. The effect also seems to be scaled by importance, for example, changes to low importance concepts (black) has no impact on ΔKL.

measure how changes in concept similarity impacts the l_2-distance of the activations, KL-divergence of the logits, and match accuracy of the predictions. We investigate this question because it is possible that changes in Pearson correlation are due to changes in predicted coefficients on unimportant images for a particular concept, leading to no change in model behavior. In Fig. 3.4, we visualize the change in Pearson correlation (ΔPearson) against the change in the three aforementioned metrics. We use color to indicate concept importance (warmer colors are more important). As expected, we observe a trend showing that l_2-distance increases as similarity decreases. For the KL-divergence, we observe two trends: (1) when the importance is sufficiently high, the KL-divergence increases as similarity decreases and (2) when the importance is low, there is no effect on the model's logits. Finally, we observe a trend that shows that model predictions change as a function of both similarity and importance. We find that these trends roughly hold for all models, although the structure of the plots changes for ViTs. See Sec. 3.8.8 for more results.

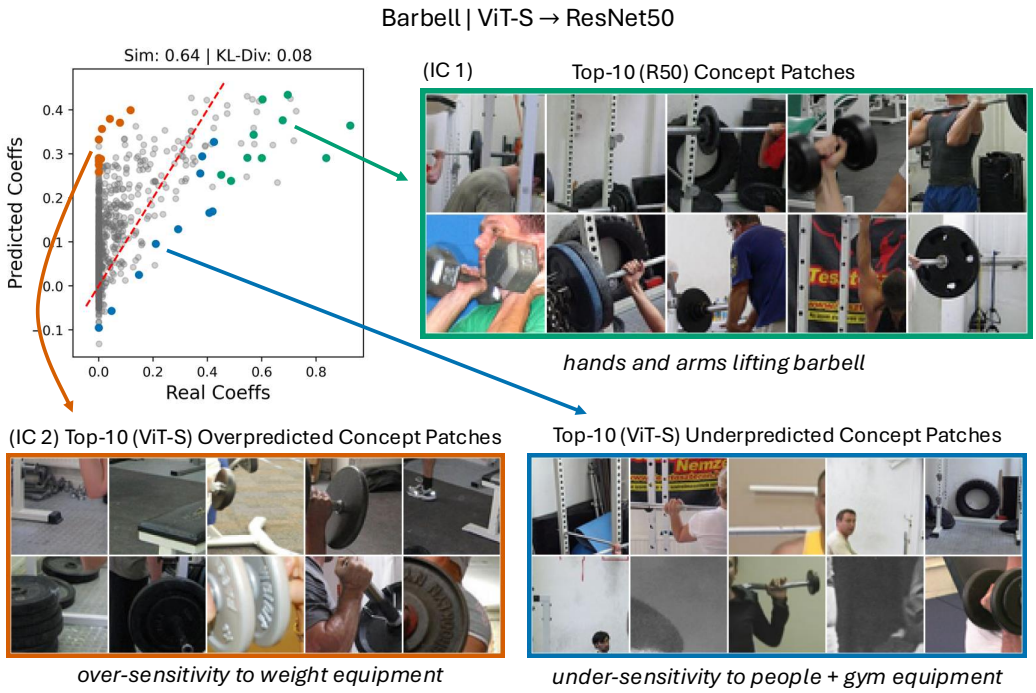


Figure 3.5: Interpreting low similarity concepts. In this example, we find a RN50 concept for the barbell class that the ViT-S is not able to predict. (Green): The RN50 concept reacts to images of hands lifting barbells. Additionally, many images contain vertical supports for a squat rack. We train a regression model on the ViT-S activations to predict the RN50 concept coefficients. (Blue): The ViT-S regression model under-reacts to images containing hands, people, and squat racks. (Orange): It over-reacts to images that have a greater focus on weight plates. These results suggest that the the specific concept of hands lifting barbells is not represented in the ViT-S. In Sec. 3.5.5 we use an LLVM to analyze the image collages (IC1 and IC2) and find that it detects similar differences in the visualizations.

3.5.4 Low Similarity Concepts

In Fig. 3.3 we observe that model comparisons identify low similarity, high importance concepts. These concepts are particularly interesting because they identify visual features that one model has constructed that the other has not. In Fig. 3.5 we apply our proposed approach for understanding the dissimilarity between predicted and real concepts (see Sec. 3.4.4). We analyze a RN50 concept used on the barbell class, which primarily reacts to images containing hands/arms lifting a barbell. When a regression model is trained on the ViT-S activations to predict the coefficients of the RN50 concept, the model becomes over-sensitive to any image with weight plates or gym equipment and remains under-sensitive to images of people around gym

equipment. This result suggests that the ViT-S does not have a feature encoding "hands lifting weights." In Fig. 3.8, we show several examples of important low similarity concepts that are visually interpretable.

3.5.5 LLVM Generated Concept Difference Analysis

Fig. 3.5 contains a manual analysis of the difference between a visual concept for the barbell class from two different models. Here we use an LLVM (ChatGPT-4o [53]) to analyze image collages and describe them. This approach has a similar goal to [54], in which the authors use an LLVM to interpret specific MLP heads in a CLIP ViT model. The input and output structure for this experiment is provided in Fig. 3.9. We do not emphasize differences in the system prompt, instead we ask the model to describe both similarities and differences. We provide the top-k image collage (IC1) and the over-predicted image collage (IC2) as input to the LLVM. The LLVM outputs for the barbell concept are presented below. We observe that the LLVM description is similar to the manual description in the caption:

LLVM Analysis of a Visual Concept from the Barbell class

IC1: A collage of people engaging in strength training exercises, with a focus on lifting dumbbells, barbells, and performing weightlifting movements.
IC2: A collage showcasing gym equipment and weights, with some emphasis on exercises involving barbells and dumbbells but with fewer people depicted.
Similarity: Both collages center on weightlifting and gym-related activities, featuring equipment and exercises.
Difference: IC1 highlights people actively performing weightlifting exercises, while IC2 primarily focuses on the gym equipment and setup with minimal human involvement.
Semantically different: [Yes] IC1 emphasizes the activity of weightlifting, whereas IC2 focuses more on the tools and environment of the gym.

3.6 Limitations

We limit our analysis to computer vision models. In practice, there is nothing restricting the application of the RSVC approach to large language models, but we leave this for future work. Our work builds on concept-based XAI methods, and thus inherits the trade-off between fidelity and interpretability. In particular, we note that the reconstruction error varies for each model, which may explain some properties of concept similarity in our experiments. However, we are agnostic to the precise concept extraction method used and thus our approach will benefit from further

advances in these methods. For example, incorporating a recursive strategy like the one presented in CRAFT [41], may significantly improve the number of interpretable comparisons discovered. Outside of overly artificial settings whereby two models are trained on completely different datasets, we note that it can be challenging to compare the representations of two models trained on the same or similar datasets (e.g. ImageNet) as done in this work. However, we believe that this more challenging setting is of most interest and relevance.

3.7 Conclusion

We introduced a new method for exploring representational similarity via interpretable visual concepts (RSVC). In contrast to existing representational similarity methods that simply provide a single numerical score to denote similarity, our approach fuses ideas from concept-based explainability and shows us *what* visual concepts make two models similar or dissimilar. In particular, we demonstrate that comparing models can be an effective path towards understanding what concepts a model is missing. In future, this may be helpful in identifying sources of model failures. We presented experiments on a range of different vision models and demonstrated that our approach is general and can be applied across a variety of different backbone models, irrespective of the pretraining objective used by the model. Finally, we suggest that explaining the functional differences in two models' behavior could serve as a valuable testbed for future XAI research. We hope that our work opens the door to further investigation into how concepts are represented inside of deep networks.

Acknowledgments

NK and PP were supported by the Simons Foundation and the Resnick Sustainability Institute. OMA was supported by a Royal Society Research Grant. The authors thank Rogério Guimares, Markus Marks, Atharva Sehgal, and the anonymous reviewers for their valuable feedback.

Appendix

3.8 Additional Experiments

3.8.1 Layerwise Concept Similarity

Correlation

We propose a correlation based comparison method that has a lower computational cost than the regression method presented in the main text (Sec. 3.4.2). This approach allows us to efficiently compare many concepts and layers for two models. Recall that each column of $\mathbf{U}$ contains the concept coefficients of a specific concept for each image. If the concepts encode the same information, they should have highly correlated activation patterns since they would react similarly to the same proposal images. Thus, we compute the correlation for each vector $\mathbf{u}_1^i \in \text{Columns}(\mathbf{U}_1)$ and $\mathbf{u}_2^j \in \text{Columns}(\mathbf{U}_2)$. We measure both Pearson and Spearman correlation to form the correlation matrices $\mathbf{R}^\rho \in \mathbb{R}^{k\times k}$ and $\mathbf{R}^S \in \mathbb{R}^{k\times k}$. Since concepts extracted from each network do not have a direct correspondence, MCS measures the concept similarity between a concept from M_1 and all of the concepts from M_2 and keeps the maximum. We compute the maximum concept similarity (MCS^c) over each dimension of the correlation matrix for each concept and class:

$$\text{MCS}_1^c = \max_i \mathbf{R}_{ij} \quad \text{and} \quad \text{MCS}_2^c = \max_j \mathbf{R}_{ij}. \tag{3.3}$$

MCS is fast to compute and can be done with relatively few image samples (Sec. 3.9.2). Thus, we use it to compute layerwise concept similarity which gives us coarse-grained insights into the similarity between each layer of two different models. For layerwise concept similarity, we compute the correlation matrix $\mathbf{R}^c$ between M_1 and M_2 at each pair of layers for every class c. Then, for each correlation matrix we compute the *mean* maximum concept similarity (MMCS) over each dimension and then take the average of the two matrices:

$$\begin{aligned} \text{MMCS}_m &= \frac{1}{k \cdot c} \sum_{c=1}^{c} \sum_{i=1}^{k} \text{MCS}_m^{c,i}, \\ \text{MMCS} &= (\text{MMCS}_1 + \text{MMCS}_2)/2. \end{aligned} \tag{3.4}$$

However, correlation based similarity can be affected by confounds from concept extraction. For example, extracted concepts can entangle or disentangle visual features that are encoded by the respective models. Suppose $\mathbf{U}_1$ encodes features for both ears and snouts of a dog together in a single concept, but these two features are disentangled into two concepts in $\mathbf{U}_2$, the maximum match between these concepts

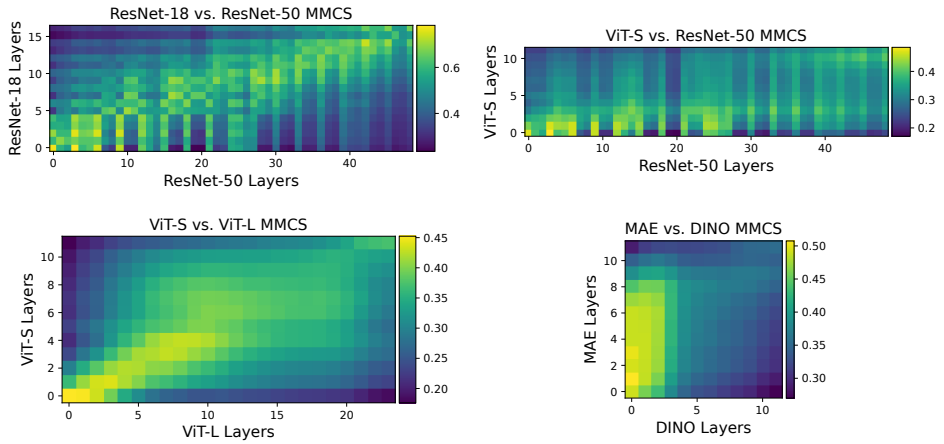


Figure 3.6: Layerwise mean-max concept similarity. We compare four pairs of models across many selected layers using Pearson correlation. Each entry in the matrix is the mean maximum concept similarity (MMCS) between M_1 and M_2 at a particular pair of layers. Brighter colors represent higher MMCS values. We see that, in general, concept similarity is highest in earlier layers and decays as networks get deeper. We also notice that there is a slight increase in similarity towards the final layers (Sec. 3.8.1).

would be lower even though both networks are encoding the same information. Additionally, extracted concepts do not contain all the information in the network since there is some reconstruction error when learning the decomposition. Thus, correlation matrices can tell us when two concepts are highly similar, but do not tell us if a concept is missing in a layer. However, they can serve as a noisy lower bound for measuring concept similarity (Sec. 3.8.1).

Comparing Model Layers

In Fig. 3.6, we explore how concept similarity arises across many layers of each network. We compute the MMCS at each pair of layers and visualize the resulting matrix. For all models, we find that concept similarity is highest at the early layers of each model and decays gradually as network depth increases. In all model comparisons, we see a slight increase in similarity towards the final layers relative to the preceding layers, suggesting that the way networks organize information converges as the network get closer to producing a final decision. Interestingly, [24] also found the last layer to have better properties for concept extraction.

We also notice several properties unique to each comparison. When comparing different sized models with the same architecture, the similarity is related to the

relative depth of the layer. For ResNets, we notice that matrices show a pattern of increased similarity after residual blocks and lower similarity for layers within blocks. For the ViT-S and ViT-L we find that there is a broad band of concept similarity in the middle of each network in which the ViT-S layers 4 through 9 have higher similarity to ViT-L layers 5 through 20. In addition, there is an increase in concept similarity between the last layer of the ViT-S and the last three layers of the ViT-L. When comparing the RN50 to ViT-S we find that concept similarity of layers 0 through 25 are most similar to layers 0 through 3. This finding matches observations found in previous work, in which relatively earlier layers in the ViT match relatively later layers in the ResNet [16]. Finally, when comparing the MAE to DINO, we see high concept similarity between the first 3 layers of DINO and the first 8 layers of the MAE. However, this similarity decays significantly as the layer index of DINO increases. This divergence in concept similarity may be due to differences between supervised and self-supervised training, but further research is needed.

Comparing MCS (Pearson) to Lasso Regression (Pearson)

In Sec. 3.8.1, we claimed that MCS (Pearson) could serve as a noisy lower bound to better measurements of similarity like CMCS (Pearson). This is because MCS is computed over concepts extracted by the decomposition method, which introduces its own entanglements, disentanglements, and reconstruction error. In Fig. 3.7, we compare the penultimate layers of the four pairs of models and compute both MCS (Pearson) and the CMCS (Pearson). We find that MCS is correlated to CMCS, but, as expected, under-predicts the concept similarity. For this experiment only, due to the fact that we computed Pearson correlation over images in the training set of the deep neural network, we compare to the mean similarity score that is computed from the held-out folds of the five lasso regression models. The held-out folds are not part of the training set for the *regression models*, but were part of the training set of the deep neural network. However, this comparison is more fair, since we compare the two methods on the same set of images.

3.8.2 Analyzing Concepts

In Fig. 3.8 we display some more examples of dissimilar concepts found through RSVC. We also provide interpretations of the dissimilarities between the concepts. Note that these dissimilarities do not necessarily indicate worse performance on images from this class because there are many possible correct strategies when trying to make a decision about an image.

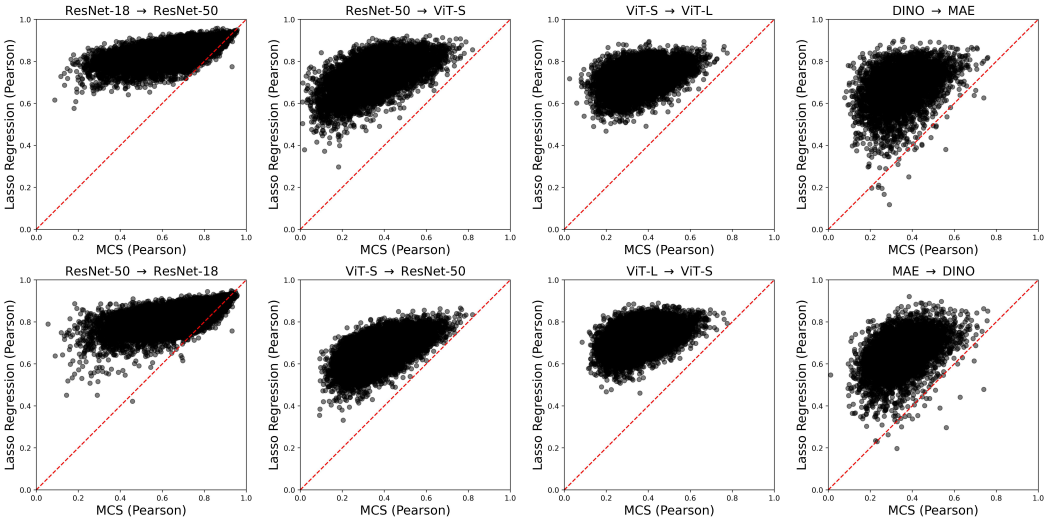


Figure 3.7: MCS (Pearson) vs. lasso regression. We see that the most points lie above the red-line. This means that lasso regression (followed by Pearson correlation on the predicted and real coefficients) usually predicts a higher similarity value than the MCS values directly on the columns of the coefficient matrix. Thus, we experimentally validate that the Pearson correlation acts as a noisy lower bound on concept similarity.

Rugby Ball | RN18 → RN50

The RN50 has learned a visual concept that entangles the arms of a rugby player and the rugby ball. We see in the under-predicted samples that the regression model under-predicts samples with the players' limbs and rugby balls together. When visualizing the over-predicted samples we can see that the regression model increases sensitivity to both close-ups of rugby balls and to the legs of the players. These results suggest that the RN18 encodes for the legs of the rugby players independently of the rugby ball and the regression model is using these independent features to try and reproduce the RN50's concept. It also suggests that the RN18 is not encoding the arms as an independent feature.

Grey Whale | ViT-S → ViT-L

It appears that the ViT-L has learned a visual concept for whales surfacing parallel to the surface of the water. We can see that the regression model under-predicts images of the whales back and also images of its eye in a horizontal orientation. The regression model seems to have increased sensitivity to breaching whales, either raising their tails or their heads. This suggests that the ViT-S has entangled calm whales floating at the surface with active whales breaching the water.

Strawberry | RN50 → ViT-S

This concept is one of the lowest similarity concepts that causes a meaningful change in model behavior (as seen by the KL-Divergence). We can see that the ViT-S has learned a concept for mixed fruits that include strawberries. The under-predicted and over-predicted samples show that the RN50 has no ability to reproduce this pattern, suggesting that it ignores mixed fruits entirely.

Volleyball | RN18 → RN50

The RN50 has learned to encode volleyball players in a variety of active positions. There seems to be an emphasis on hands and arms near or above the net. The RN18 under-predicts close-up images of volleyballs and players at the net and it tends to over-predict images containing balls high in the air and close-ups of nets/grids. This suggests that the regression model is trying to reproduce the behavior of the RN50 concept using a variety of related features learned by the RN18. We explore the top-6 neurons that most contribute to the regression model's prediction of this concept in Fig. 3.11. We compute the neuron contribution using permutation feature importance (Sec. 3.9). We find that the regression model uses RN18 neurons that are highly sensitive to volleyballs in the air, players near nets, and close-ups of nets qualitatively explaining some of the differences between the predicted and real concept.

Bluejay | ViT-B MAE → ViT-B DINO

The DINO model has isolated the tail of the bluejay as an important visual feature. The regression model under-predicts images with bluejay tails and over-predicts random images, suggesting that this concept is not independently encoded in the feature space of the MAE.

3.8.3 LLVM Generated Concept Difference Analysis

In Fig. 3.9, we describe a procedure that uses LLVM's for analyzing the image collages produced by RSVC. We apply this method to the comparisons in Fig. 3.8 and report the results below. We find that the LLVM annotations are fairly similar to the manual annotations.

Rugby Ball | RN18 → RN50

IC1: A collage featuring various close-ups of players holding rugby balls during gameplay, emphasizing hands and ball contact.

IC2: A collage with mixed shots of rugby gameplay, focusing on players' lower bodies, shorts, and action involving the rugby ball from different perspectives.

Similarity: Both collages depict scenes from rugby games and include elements of players and rugby balls in action.
Difference: IC1 focuses on hand interactions with the rugby ball, while IC2 emphasizes lower body dynamics and broader gameplay elements.
Semantically different: [Yes] IC1 highlights ball handling, whereas IC2 portrays physical movement and positioning.

Grey Whale | ViT-S → ViT-L
IC1: A collage featuring close-up views of whales, focusing on surface water interaction and detailed textures of their bodies.
IC2: A collage showing whales from various perspectives, including body parts like tails, heads, and fins, with more diverse settings and details.
Similarity: Both collages depict whales and highlight different parts of their bodies in aquatic environments.
Difference: IC1 emphasizes close-up textures and water interaction, while IC2 provides broader views of whales, showing distinct body parts and their environment.
Semantically different: [Yes] IC1 focuses on textures and surface detail, while IC2 captures a wider perspective of whale features and behaviors.

Strawberry | RN50 → ViT-S
IC1: A collage featuring various fruits, with a focus on strawberries along with other fruits like grapes, blueberries, and kiwi, emphasizing vibrant colors and diversity.
IC2: A collage entirely centered on strawberries, showing detailed close-ups of their textures and seeds in different lighting and angles.
Similarity: Both collages prominently showcase strawberries and their textures.
Difference: IC1 includes a mix of other fruits alongside strawberries, while IC2 exclusively focuses on strawberries and their intricate details.
Semantically different: [Yes] IC1 presents a broader context of mixed fruits, whereas IC2 isolates strawberries as the sole subject.

Volleyball | RN18 → RN50
IC1: A collage depicting volleyball gameplay, focusing on players' hands, the volleyball, and interactions around the net during action-packed moments.
IC2: A collage showing volleyball gameplay as well but with a wider variety of scenes, including players near the net, different environments, and more casual setups.
Similarity: Both collages revolve around volleyball and the central theme of gameplay near the net.

Difference: IC1 emphasizes action shots with players and the ball in dynamic moments, while IC2 includes a mix of gameplay contexts and less action-oriented scenes.
Semantically different: [Yes] IC1 focuses on the intensity and dynamics of volleyball action, whereas IC2 portrays a broader variety of volleyball scenarios and contexts.

Bluejay | ViT-B MAE → ViT-B DINO
IC1: A collage of close-up views of blue jay feathers, primarily showcasing the distinctive blue and black patterns on their wings and tails.
IC2: A collage of blue jays in various settings, including full-body shots, close-ups of their heads, and natural habitats like branches and foliage.
Similarity: Both collages feature blue jays and focus on their unique physical features and natural environment.
Difference: IC1 highlights detailed patterns on blue jay feathers, while IC2 captures a broader context of the bird's appearance and behavior in its habitat.
Semantically different: [Yes] IC1 emphasizes feather patterns, whereas IC2 provides a holistic view of blue jays in their environment.

3.8.4 Specificity of RSVC on the Toy Concept Experiment

In Sec. 3.5.1 we showed how RSVC can be used to distinguish between a model trained to use the pink square concept and a model trained to ignore the pink square concept. However, training with the pink square concept could have undesirable effects on other model concepts for the Common Eider class. In Fig. 3.10, we show two shared concepts that are unaffected by the pink square. This experiment suggests that our toy concept training procedure does not impact other concepts learned by the networks.

3.8.5 Varying ResNet-18 Training

Next, we conduct controlled variations of model training and measure how it effects model concept similarity in the last layer. We train a ResNet-18 model on variations of the NABirds dataset.

We start by comparing two ResNet-18 models trained on NABirds with different seeds, 4834586 (R18 s483) and 87363356 (R18 s873). We find that, despite varying the seed during training, both models discover highly similar concepts (Fig. 3.13**A**). Then, we train a ResNet-18 model on a modified version of the NABirds dataset in which waterbirds (169 classes) have been excluded during training (R18 NAB-WB). After training, the backbone is frozen and just the classification head is trained on the

full dataset. We compare this model to R18 s483 trained on the full dataset (34%). We find that training without waterbirds results in a significant decrease in performance (25%) and, surprisingly, only a slight increase in dissimilar concepts (Fig. 3.13**B**).

We then explore if introducing novel features from an out-of-domain dataset would result in more dissimilar concepts. In this experiment, we train one model (R18 NAB+SC) on a combined dataset of NABirds and Stanford Cars [55] achieving 37% accuracy on the combined classification task. To compare to R18 s483, we freeze the backbone and re-train the model head on both NABirds and Stanford Cars, achieving 26% accuracy. We find training the backbone on Stanford Cars significantly increases concept dissimilarity (Fig. 3.13**C**). Interestingly, we find that the increase in dissimilarity is bi-directional, both models are less able to predict the concepts of their contrasted pair. In order to better understand the bi-directional nature of this dissimilarity, we visualize a few concepts from each model in Figure 3.12. These concepts were selected by (1) filtering concepts above the 75th percentile in delta KL-divergance, (2) visualizing the 15 lowest delta Pearson concepts and (3) manually selecting concepts that were easiest to interpret. We find dissimilar concepts from the R18 NAB+SC model that seem to be semantic concepts specific to car models. In contrast, we found no dissimilar car related concepts that met the criteria from R18 s483. Instead, we find that concepts that meet this criteria are from NABirds classes and tend to be challenging to interpret. We then visualize R18 s483 concepts that result in the largest kl-divergence when replaced by the predictions from R18 NAB+SC. When visualizing these concepts we find two car related concepts that seem to be primarily driven by color, but not car model. These results match the intuition that R18 NAB+SC should contain more complex, car model aligned concepts that can be used to better classify images from Stanford Cars. However, it is not a complete explanation of model behavior and further analysis is needed to make concrete statements about concept dissimilarity.

Finally, we compare a ResNet-18 trained on ImageNet and fine-tuned on NABirds to a NABirds model (R18 ImgNet PT). Unsurprisingly, this results in the most dissimilar concepts (Fig. 3.13**D**). In sum, we find that larger changes during training results in more dissimilar concepts.

3.8.6 DINO and MAE Seed Variation Experiments

In Section 3.5, we compared a DINO pretrained model and a MAE pretrained model that were finetuned on the NABirds dataset. In those experiments, models were finetuned with the seed set to 4834586. In this section, we explore comparisons to models finetuned with a different seed, 87363356. The models finetuned with the new seed are denoted as DINO_s873 and MAE_s873. We compare two pairs of models, DINO_s873 to MAE_s873 and DINO_s483 to DINO_s873. In Figure 3.14, we show that changing the seed does change the concepts learned, but that the general relationship between the different pretraining strategies is preserved. In particular, we find that comparing DINO models finetuned with different seeds results in a higher average similarity (∼0.89) than models with different pretraining strategies (∼0.80), indicating that the seed has a smaller impact on finetuning than the initialization.

3.8.7 Same Model Concept Similarity vs. Concept Importance

In this section, we validate the feasibility of the regression task. Due to the reconstruction error inherent in decomposition methods, it is not possible to perfectly predict the concept coefficients from the activation matrix. However, in Fig. 3.15, we show that the regression models do well when trying to do same-model concept regression and significantly better than cross-model concept regression.

3.8.8 Additional Replacement Tests

In Figs. 3.16 and 3.17 we visualize the replacement tests for the three pairs of models not presented in the main text. We see the same effects as before, with a decrease in similarity corresponding to increasing changes in model behavior. Notably, although DINO and MAE models (finetuned on NABirds) have high similarity relative to the other models, they show stronger changes in model behavior for smaller changes in similarity. However, it is not clear whether these differences are due to changes in dataset or changes in pretraining.

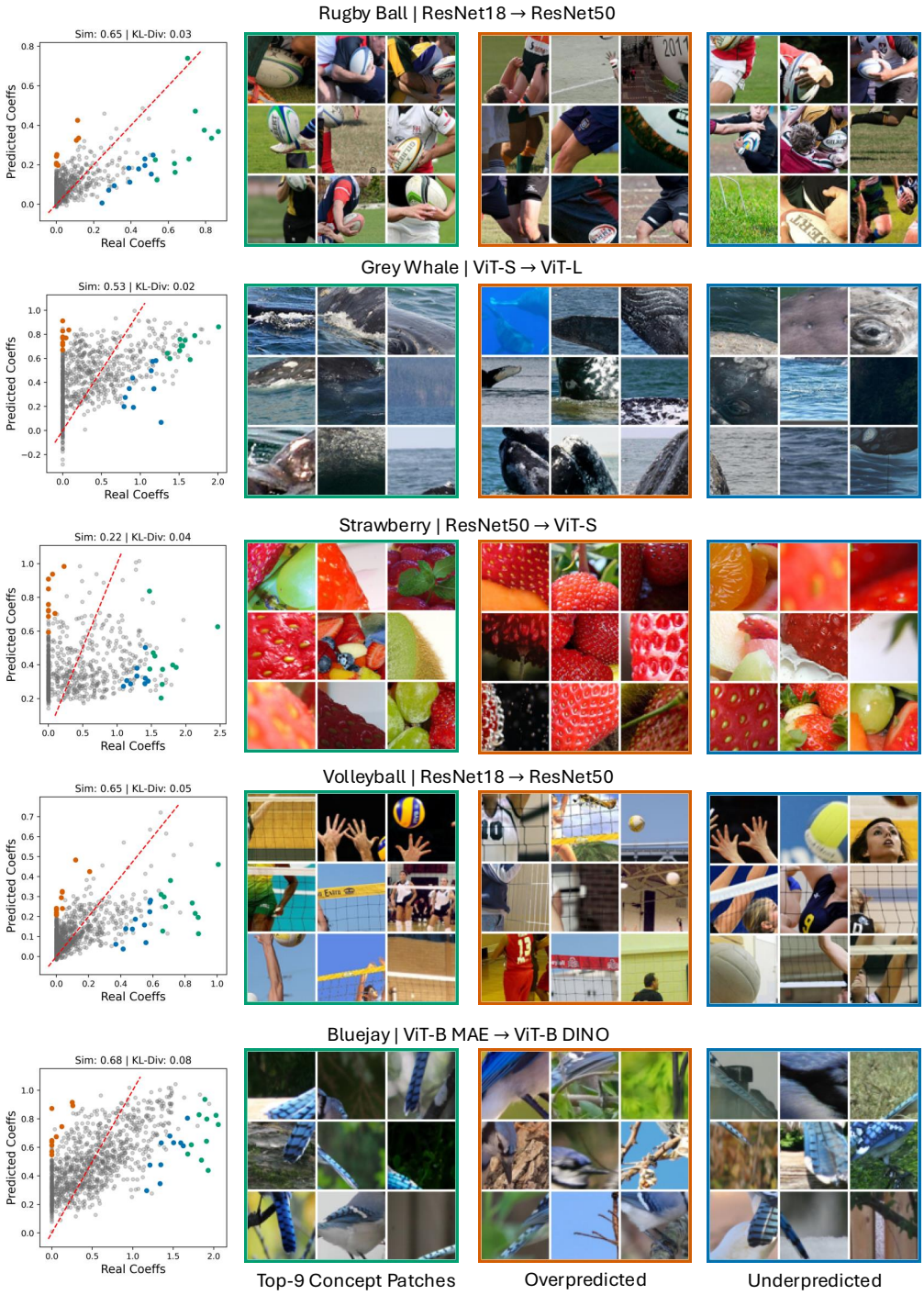


Figure 3.8: Qualitative samples. In each row, we show visualizations for selected concepts from different model comparisons. In the first column of each row, we show scatter plots between real and predicted concept coefficients. Colored points mark the top-9 images in different subregions of the scatterplot. Each subregion indicates a different aspect of dissimilarity. (Green): Top-9 images for the real concept. These images are used to help the user understand what the target concept pays attention to. (Blue): Top-9 images that are underpredicted by the contrasted model. (Orange): Top-9 images that are over-predicted by the contrasted model. You may need to zoom in to best analyze the image grids. We discuss possible interpretations of the concepts in Sec. 3.8.2. See Sec. 3.4.4 for a detailed breakdown of how to interpret these plots.

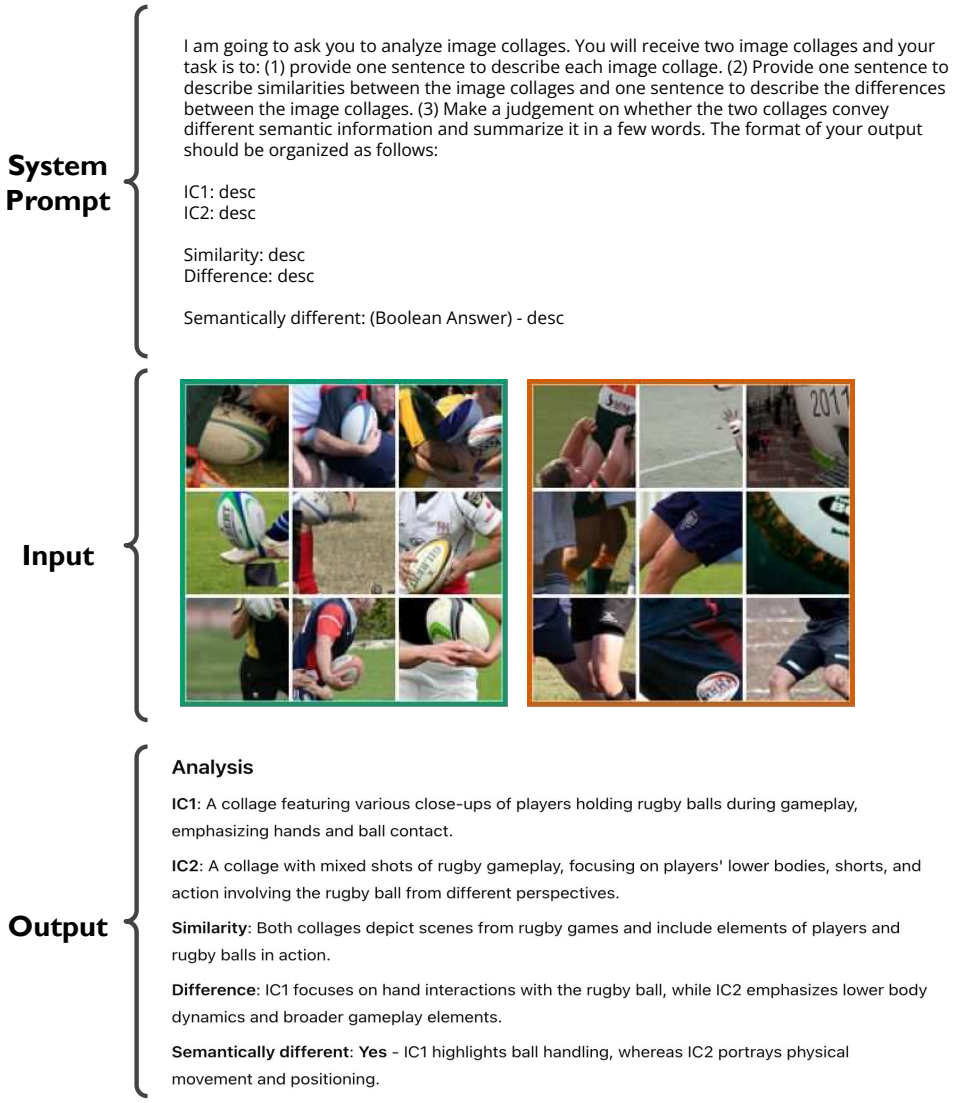


Figure 3.9: LLVM concept difference analysis. We use ChatGPT-4o [53] to analyze concept differences. We provide a general system prompt asking the LLVM to describe the collages, provide a description of the similarities and differences between the collages, and provide a final judgement on whether there is a semantic difference between the collages. We provide an example for the rugby ball class. The LLVM receives the image collage (IC) corresponding the top-k concept images (IC1) and the over-predicted concept images (IC2), see Fig. 3.5 for more details on image collages. Here, the LLVM notices that IC1 focuses on hands and ball contact, whereas the IC2 focuses more on lower bodies. The described difference matches the manual annotation provided in Sec. 3.8.2.

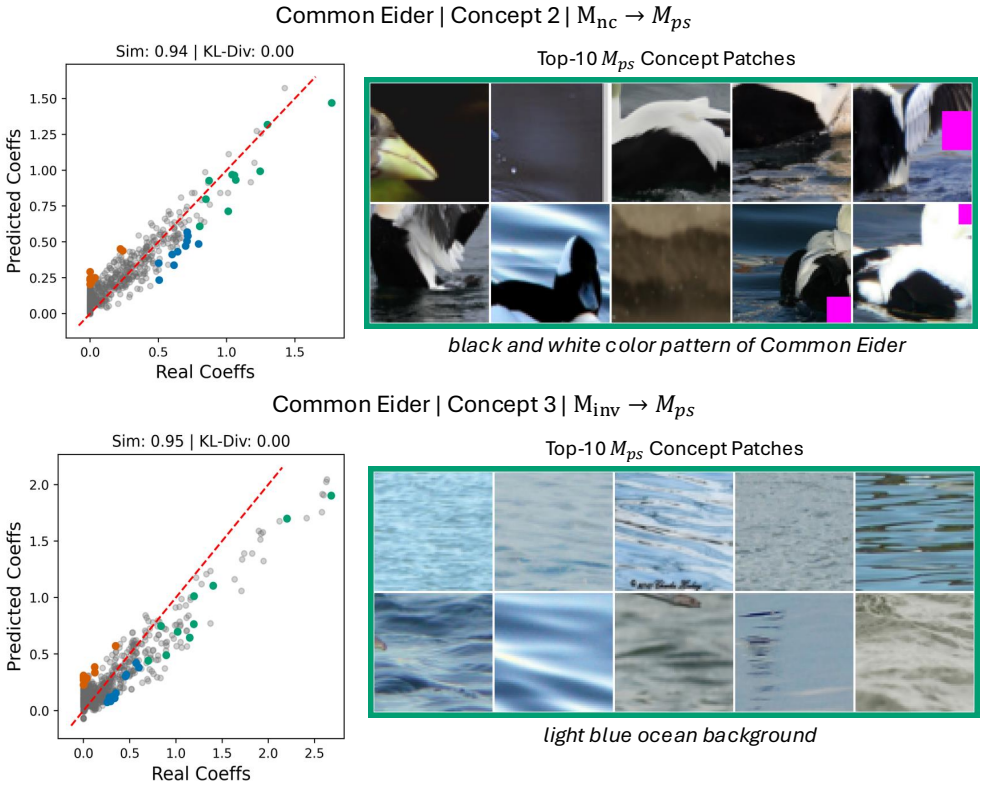


Figure 3.10: Specificity of toy concept. In Fig. 3.2, we showed that M_{nc} is not able to predict the pink square concept from M_{ps}. In this figure, we show that the toy concept does not impact the similarity between other concepts learned by the networks. We visualize the top-10 patches from Concept 2 and Concept 3 of M_{ps} in the same class (Common Eider). These concepts correspond to the white and black color pattern of the bird and a water background. Note that these models have been trained from scratch on NABirds resulting in a relatively low 34% accuracy. This leads to noisier concepts that are more challenging to interpret. Importantly, we can see that M_{nc} still has a very high similarity score for these two concepts, highlighting the specificity of RSVC.

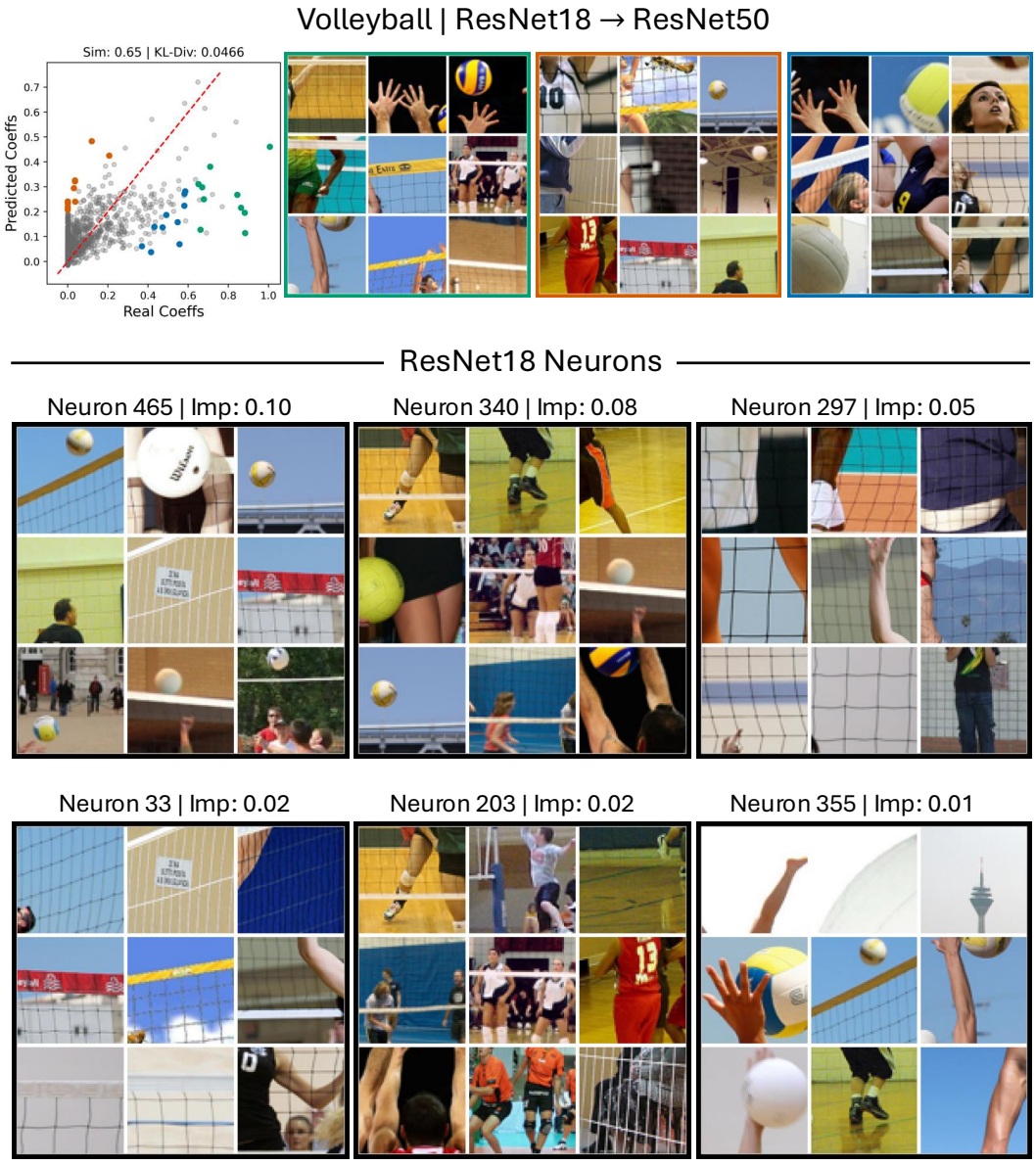


Figure 3.11: Neuron analysis for volleyball concept difference. In Fig. 3.8 we visualized a RN50 concept for the volleyball class that the RN18 did not contain. In this figure, we explore the top-6 neurons used by the regression model to predict the RN50 concept. We find that Neuron 465 is sensitive to edges between a volleyball net and the background. It seems to mistake some grid-like textures for nets as well (image [1, 0], [1, 1], and [2, 0]). In addition, it seems to be sensitive to volleyballs high in the air. Neuron 340 seems to activate for athletes in indoor gyms and seems partial to lower bodies. Neuron 297 is sensitive to close-ups of nets with hands or arms in the frame. In summary, these neuron visualizations help to explain some of the images over-predicted by the regression model.

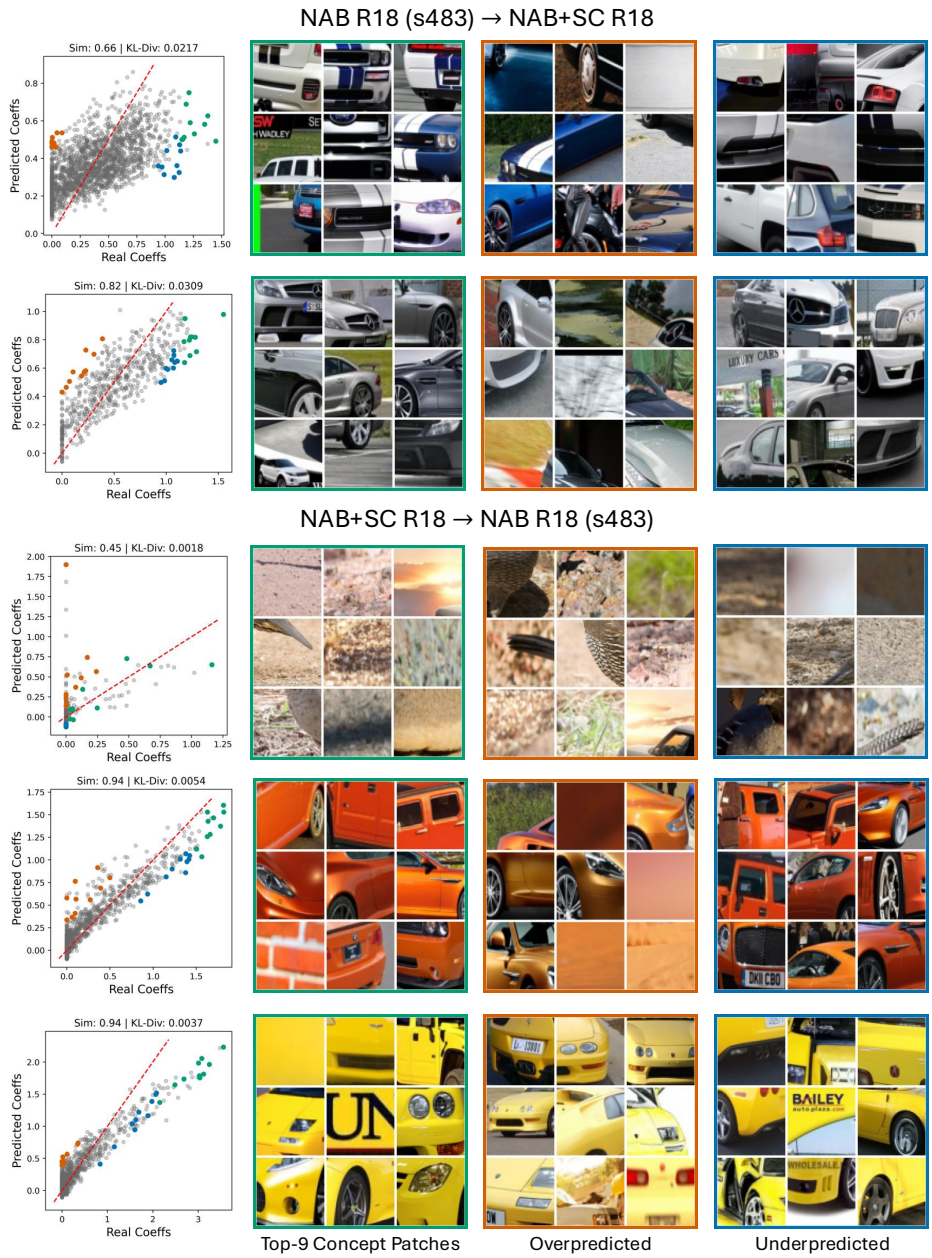


Figure 3.12: Impact of training on stanford cars. In each row, we show visualizations for selected concepts from comparing R18 NAB+SC to R18 s483. In the first two rows, we visualize two R18 NAB+SC concepts that R18 s483 cannot reproduce. The first concept is a racing stripe that is associated with the Shelby Mustang. The R18 s483 model appears to sometimes entangle this concept with a blue color, irrespective of the car model. The second concept appears to be common features associated with Mercedes cars. For this concept, the difference between the two models is more abstract and challenging to interpret. We visualize NAB R18 s483 concepts in the next three rows. First, we show a R18 s483 concept that R18 NAB+SC is unable to predict. We see that this concept is very abstract without a clear pattern, but is generally related to sandy textures. Next, we visualize two car-related concepts from R18 s483. We find that these concepts are sensitive to the combination of the presence of a car and a specific color. For the orange car concept, the R18 NAB+SC makes small over-predictions with different shades of orange. For the yellow car concept, the over-predicted group shows a different shade of yellow and a specific style of car. A discussion of these results is available in Sec. 3.8.5.

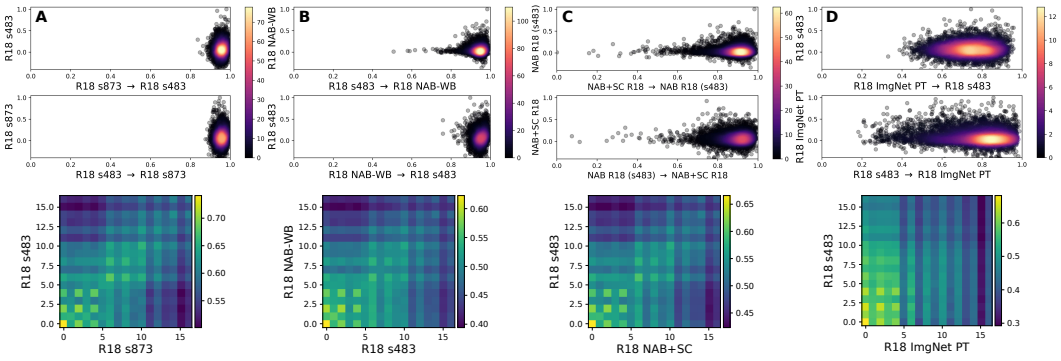


Figure 3.13: Effect of seed and dataset on ResNet-18 similarity. We compare several pairs of ResNet-18 models while varying their training protocols. We use the same base model in all comparisons, a ResNet-18 model trained with the seed set to 4834586 (R18 s483). (A) We compare the base model to a model trained with seed 87363356 (R18 s873) and find that the two models are highly similar despite the change in seed. (B) We train a ResNet-18 on a modified dataset where we exclude 169 classes that belong to the coarse category of waterbirds (R18 NAB-WB). When comparing to the seed variation experiment, we see a slight increase in the number of dissimilar concepts. (C) We train a ResNet-18 on a combined dataset of NABirds and Stanford Cars (R18 NAB+SC). To compare to the base model, we freeze the base model's backbone and re-train the linear classifier on this combined dataset. We find that introducing Stanford Cars results in a significant increase in dissimilar concepts. (Right) Finally, we compare to a model pre-trained on ImageNet and fine-tuned on NABirds (R18 ImgNet PT). We find that training on ImageNet introduces many novel concepts that are dissimilar to the features of the base model.

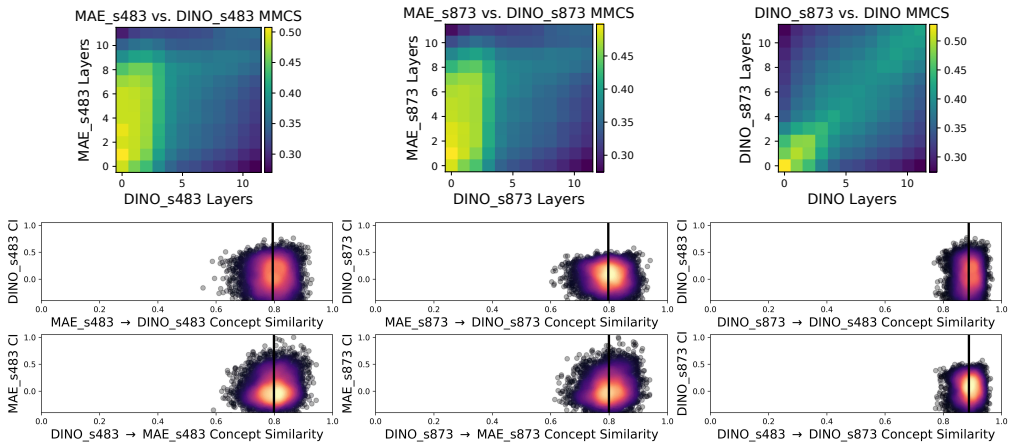


Figure 3.14: DINO and MAE seed variation. We explore the effects of varying seed on finetuning a DINO and MAE model on the NABirds dataset. **(Left):** We show layerwise and last layer comparisons of MAE_s483 vs. DINO_s483. These plots are reproductions from the main text. The black line denotes the average concept similarity. For this comparison, the average similarity in both directions is 0.80. **(Center):** We compare DINO_s873 vs. MAE_s873. We see a similar layerwise matrix and last layer similarity to DINO_s483 vs. MAE_s483. The average similarity for both models is, once again, 0.80. **(Right):** We compare DINO_s483 vs. DINO_s873 and find that there is a better layer-to-layer mapping in the layerwise comparison matrix. In addition, the average similarity in both directions is 0.89, higher than comparisons across the different pretraining strategies. Taken together, these results indicate that individual concepts change due to different seeds, but the global structure of the relationship between these models is not affected by seed.

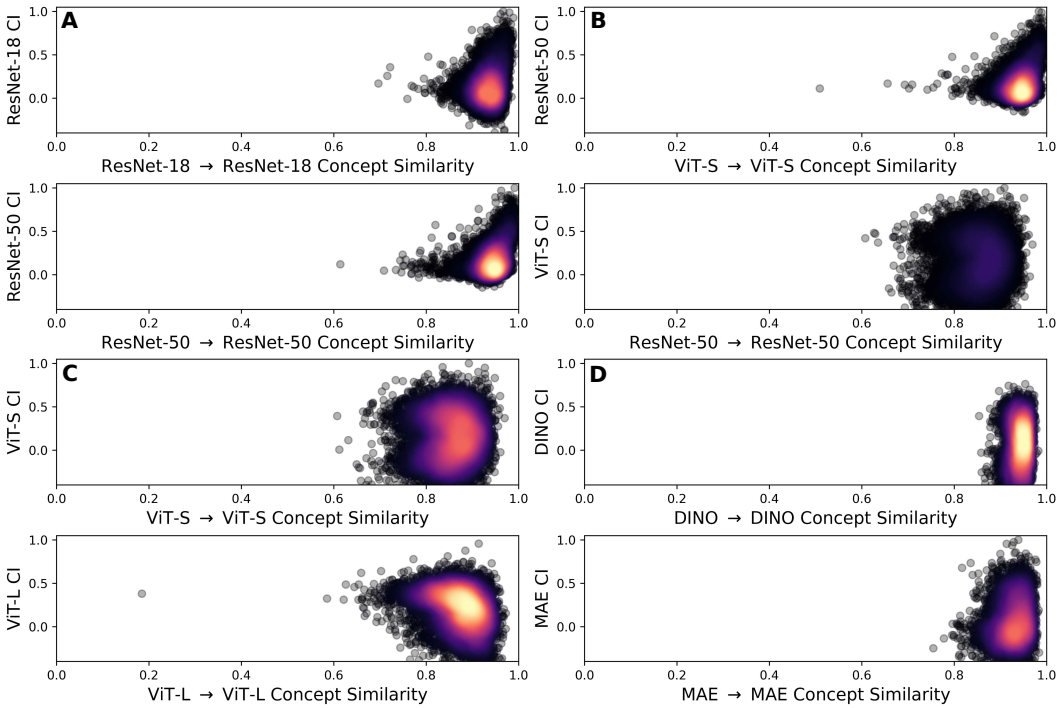


Figure 3.15: SMCS vs CI. We visualize same-model concept similarity (SMCS) against the concept importance. We find that reconstructing more important concepts tends to be easier for ResNets. However, for some ViT models, there can be important learned concepts that are hard to predict. Importantly, SMCS is significantly higher than CMCS indicating that the regression task is feasible.

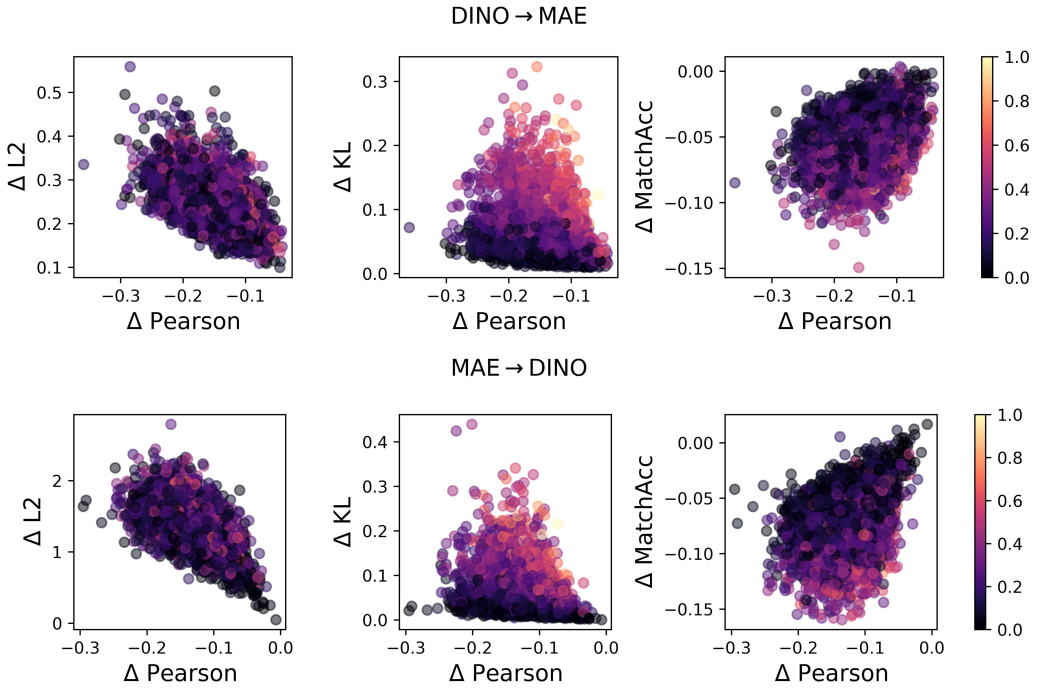


Figure 3.16: Replacement test for DINO vs. MAE (NABirds). We find that for the DINO vs. MAE comparison. As Pearson correlation decreases the l_2-distance increases, KL-divergence increases, and the match accuracy decreases. Notably, the Pearson correlation decreases a smaller amount than for the other three pairs of models, but the change in the three metrics is on the same order as the other comparisons. This suggests that these two models are more sensitive to changes in a concept.

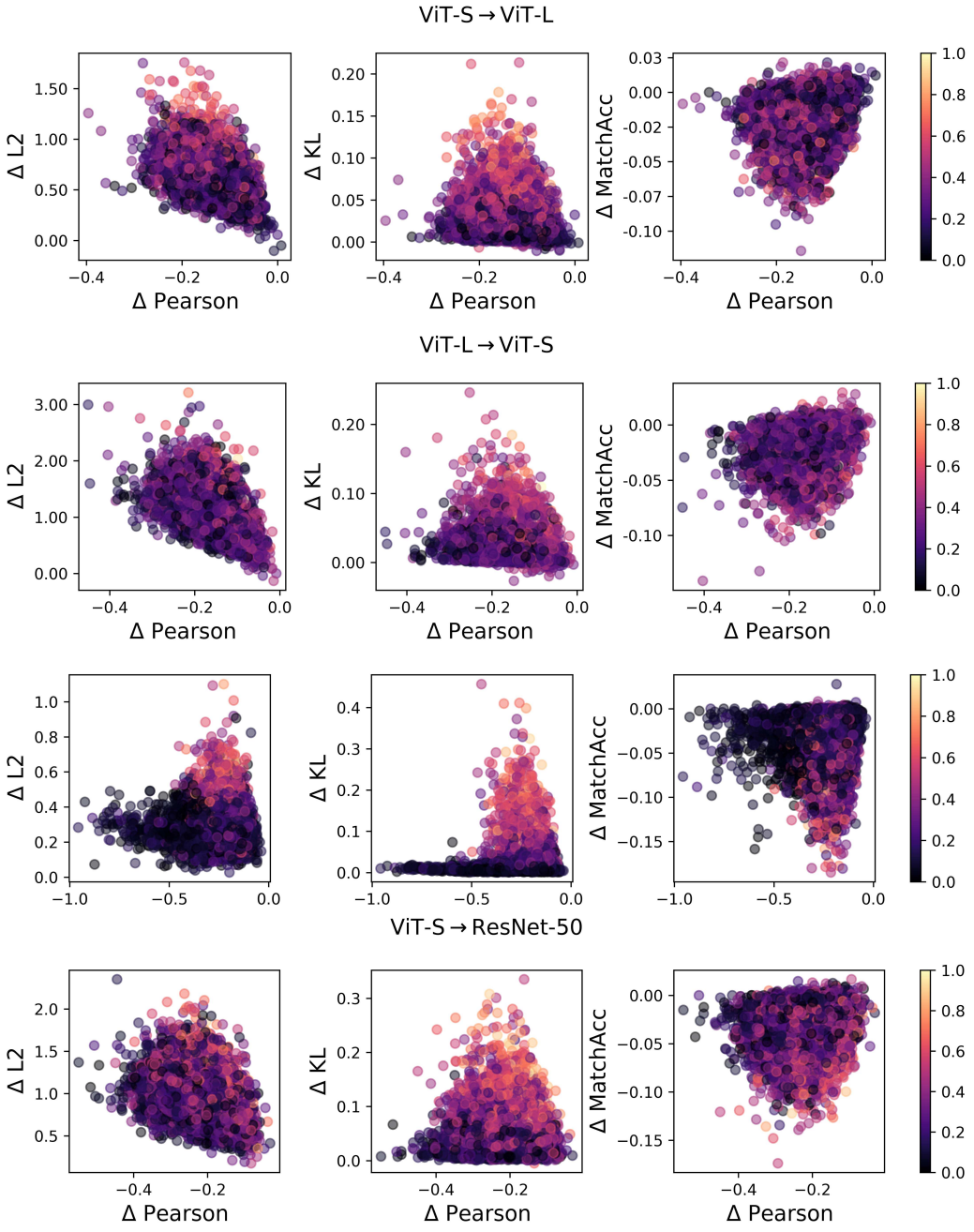


Figure 3.17: Replacement test for ViT-S vs. ViT-L and RN50 vs. ViT-S. We find that for these model comparisons, as Pearson correlation decreases the l_2-distance increases, KL-divergence increases, and the match accuracy decreases.

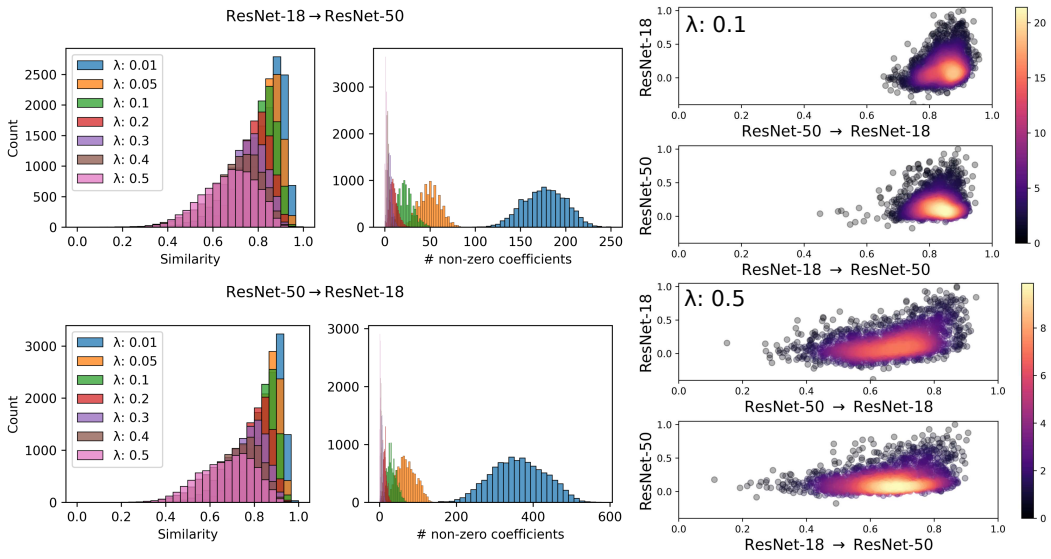


Figure 3.18: Impact of regularization on regression. Here we vary the λ for the l_1 penalty on the regression model. We use the first 200 classes of ImageNet for these visualizations. In the left column, we visualize the distribution of similarity values for each value of λ. In the center, we visualize the number of non-zero coefficients. In the right column, we visualize the similarity vs. importance plots for $\lambda = 0.1$ and $\lambda = 0.5$. We find that, as expected, increasing the l_1 penalty reduces similarity by increasing the number of zeroed coefficients. In all experiments in the paper, we use an l_1 penalty of 0.1.

3.9 Additional Implementation Details

3.9.1 Concept Extraction

For each model considered in this study, we provide information about concept extraction in Tab. 3.2. First, images are resized to 224x224 and then processed into 16 evenly spaced 64x64 pixel patches. Patches are then resized back to the image resolution of the network. We sample 100 images per model for concept extraction. All models taken from the timm library were trained with Inception style random cropping. Custom trained models were trained using random resized cropping with horizontal flipping. This ensures that the resized patches are in-domain for the network. To produce an activation matrix $\mathbf{A}$ that can be decomposed, the outputs of the network are processed. The ResNets [46] produce outputs with a batch b, channel c, height h and width w dimension. To create a matrix that can be decomposed, we use global average pooling over the h and w dimensions. The ViTs [47] produce outputs with a batch b, sequence s, and feature dimension d. We select the class token from the sequence dimension resulting in a two-dimensional matrix. We use NNMF for ResNets since they contain ReLU layers and can produce positive only

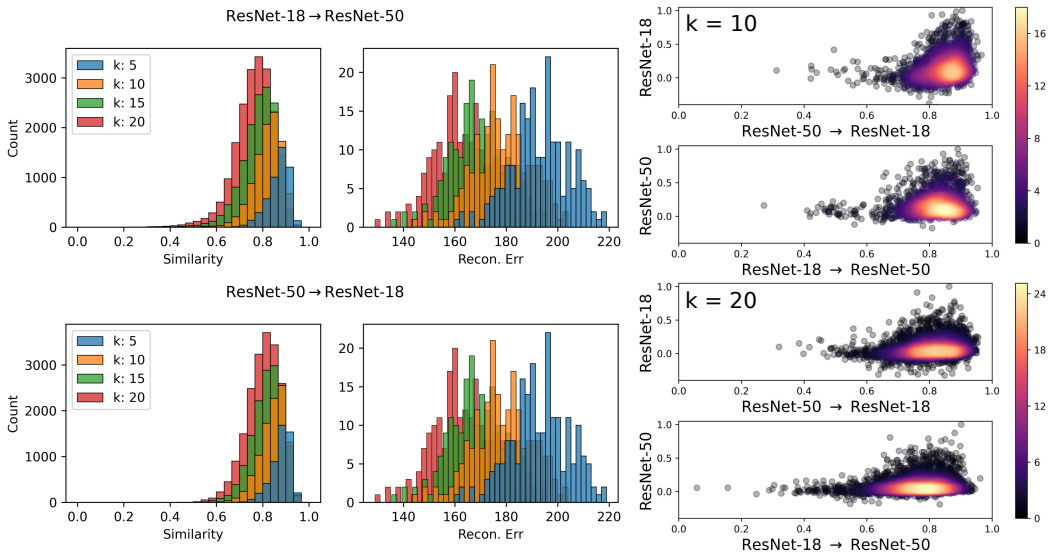


Figure 3.19: Impact of number of concepts on similarity. Here we vary k, the number of concepts in the dictionary and explore the impact on the similarity distribution. We use the first 200 classes of ImageNet for these visualizations. In the left column, we plot the distribution of similarity scores for 5, 10, 15, and 20 concepts. In the center column, we visualize the distribution of reconstruction errors for different number of concepts. As expected, increasing the number of concepts results in lower reconstruction errors. In the right column, we visualize similarity vs. importance for 10 and 20 concepts. We observe that increasing the number of concepts disproportionately increases the number of dissimilar concepts. For all results in the paper we use 10 concepts.

activations. NNMF restricts the $\mathbf{U}$ and $\mathbf{W}$ matrix to be positive. For, ViT models, we use Semi-NMF [56] which allows for both positive and negative values in the $\mathbf{W}$ matrix, but requires positive values in the $\mathbf{U}$ matrix. We use a non-negative least squares solver to fit coefficients to a new set of data points:

$$\begin{aligned} \min_{\mathbf{U}_1} \quad & \|\mathbf{A}_1 - \mathbf{U}_1\mathbf{W}_1\|_2^2, \\ \text{subject to} \quad & \mathbf{U}_1 \geq 0. \end{aligned} \tag{3.5}$$

Concept Integrated Gradients

Integrated gradients measures the importance of each pixel by averaging the gradients of the input image, as the input image is varied from a baseline value to its true value [57]. To compute concept integrated gradients the formulation is modified. Let h_1 represent the head of M_1, i.e. the final layer(s), and $\mathbf{A}_1$ be the output activations from the layer preceding h_1. As described earlier, we factorize $\mathbf{A}_1 \approx \mathbf{U}_1\mathbf{W}_1$.

We denote row vectors of $\mathbf{U}_1$ as $\mathbf{r}_1^i \in \text{Rows}(\mathbf{U}_1)$, such that $\mathbf{r}_1^i \in \mathbb{R}^{1\times d}$. To link model predictions to learned concepts, we compute model predictions as

$$\hat{\mathbf{z}}_1^i = h_1(\mathbf{r}_1^i \mathbf{W}_1), \tag{3.6}$$

where $\hat{\mathbf{z}}_1^i \in \mathbb{R}^{1\times d}$ is a row vector of prediction probabilities. Then, to compute concept integrated gradients, we average over the gradients as we linearly step from a baseline vector $\mathbf{r}^b = 0$ to $\mathbf{r}_1^i$

$$\varphi(\mathbf{r}_1^i) = (\mathbf{r}_1^i - \mathbf{r}^b) \times \int_0^1 \nabla_{\mathbf{r}_1^i} h_1 \left(\left(\alpha \mathbf{r}^b + (1-\alpha)(\mathbf{r}_1^i - \mathbf{r}^b \right) \right) \mathbf{W}) d\alpha. \tag{3.7}$$

Thus, for each class and concept we have a single value that represents the importance of that concept to model decisions. We implement concept integrated gradients based on the implementation in the xplique library [58]. For all experiments we integrate over 30 steps.

3.9.2 Concept Similarity

Correlation. Pearson and Spearman are computed using scikit-learn [59]. We use 50 images for each class from the training set of the model. Images are resized to 224×224. We use a patch size of 64×64 resulting in 16 patches per image. Thus, Pearson and Spearman correlation is computed using 800 total patches per class. The patches are resized and passed through the model to generate activations at a given layer.

Regression. We use lasso-regression [60] with a 0.1 weight on the l_1 penalty. We visualize the effect of this parameter on similarity in Fig. 3.18. Regression models are trained on the activations from at least 5 images (80 patches) and at most 200 images (3200 patches) sourced from the original training split of the dataset. For each concept and class, we train five lasso-regression models on different equally sized folds. The regression model weights are averaged and then the model is evaluated on images from the validation/test split of the original dataset. The inputs and targets are standardized to have a mean of zero and a standard deviation of one. The regression is trained using the Celer library [61]. Finally, the predicted coefficients are unnormalized before the Pearson and Spearman correlation are computed. To compute feature importance scores for regression models, we use the permutation feature importance implementation from scikit-learn [59]. We use the default parameters with 5 repeats and the random state set to 0.

Layerwise Comparisons. We list all of the layers used in the layerwise comparisons in Tab. 3.4.

Spearman Correlation. In all experiments, we found Spearman correlation to be very similar to Pearson correlation, thus we have excluded these results.

Visualizing Dis-similar Concepts. We select one patch per image in order of maximum concept coefficient. The top n patches for the real images are excluded from the pool of images used to visualize the under-predicted and over-predicted coefficients.

3.9.3 Computational Cost

All experiments were conducted using on a machine with an AMD Ryzen 7 3700X 8-Core Processor and a single GeForce RTX 4090 GPU. In Table 3.3, we detail the computational cost of each step of our proposed method. For a comparison between a ResNet-18 and a ResNet-50 on all 1000 classes of ImageNet, RSVC, and MCS take approximately 20 hours.

Table 3.1: Model performance.

Model	timm Model	ImageNet Accuracy	NaBirds Accuracy
ResNet-18	resnet18.a2_in1k	70.6%	–
ResNet-50	resnet50.a2_in1k	79.8%	–
ViT-S	vit_small_patch16_224.augreg_in21k_ft_in1k	81.3%	–
ViT-L	vit_large_patch16_224.augreg_in21k_ft_in1k	85.8%	–
DINO ViT-B	vit_base_patch16_224.dino	–	71.2%
MAE ViT-B	vit_base_patch16_224.mae	–	71.2%

Table 3.2: Concept extraction.

Model	Layer Post-processing	Method	Number of Concepts	Patch Size	Recon. Error (Last Layer)
ResNet-18	GAP	NNMF	10	64	176.2
ResNet-50	GAP	NNMF	10	64	205.5
ViT-S	Class Token	Semi-NMF	10	64	926.9
ViT-L	Class Token	Semi-NMF	10	64	1650.8
DINO ViT-B	Class Token	Semi-NMF	10	64	191.0
MAE ViT-B	Class Token	Semi-NMF	10	64	656.5

Table 3.3: Computational cost for ResNet18 vs. ResNet-50 on ImageNet.

Step	**sec/it**	**Total Time**
Activation Extraction (RN50)	1.50	25m
Concept Extraction (RN50)	2.00	33m
Concept Comparison (RSVC) Last Layer	9.00	2h30m
Concept Comparison (MCS) All Layers	14.56	4h
Concept Int. Grad	41.56	11h 30m
Regression Evaluation	2.30	38m
Total Time	-	19h36m

Table 3.4: Selected layers.

Model	Layers
ResNet-18	act1, layer1.0.act1, layer1.0.act2, layer1.1.act1, layer1.1.act2, layer2.0.act1, layer2.0.act2, layer2.1.act1, layer2.1.act2, layer3.0.act1, layer3.0.act2, layer3.1.act1, layer3.1.act2, layer4.0.act1, layer4.0.act2, layer4.1.act1, layer4.1.act2
ResNet-50	act1, layer1.0.act1, layer1.0.act2, layer1.0.act3, layer1.1.act1, layer1.1.act2, layer1.1.act3, layer1.2.act1, layer1.2.act2, layer1.2.act3, layer2.0.act1, layer2.0.act2, layer2.0.act3, layer2.1.act1, layer2.1.act2, layer2.1.act3, layer2.2.act1, layer2.2.act2, layer2.2.act3, layer2.3.act1, layer2.3.act2, layer2.3.act3, layer3.0.act1, layer3.0.act2, layer3.0.act3, layer3.1.act1, layer3.1.act2, layer3.1.act3, layer3.2.act1, layer3.2.act2, layer3.2.act3, layer3.3.act1, layer3.3.act2, layer3.3.act3, layer3.4.act1, layer3.4.act2, layer3.4.act3, layer3.5.act1, layer3.5.act2, layer3.5.act3, layer4.0.act1, layer4.0.act2, layer4.0.act3, layer4.1.act1, layer4.1.act2, layer4.1.act3, layer4.2.act1, layer4.2.act2, layer4.2.act3
ViT-S	block0, block1, block2, block3, block4, block5, block6, block7, block8, block9, block10, block11
ViT-L	block0, block1, block2, block3, block4, block5, block6, block7, block8, block9, block10, block11, block12, block13, block14, block15, block16, block17, block18, block19, block20, block21, block22, block23
DINO ViT-B	block0, block1, block2, block3, block4, block5, block6, block7, block8, block9, block10, block11
MAE ViT-B	block0, block1, block2, block3, block4, block5, block6, block7, block8, block9, block10, block11

References

[1] Neehar Kondapaneni, Oisin Mac Aodha, and Pietro Perona. "Representational Similarity via Interpretable Visual Concepts." In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=ih3BJmIZbC.

[2] Harold Hotelling. "Relations between two sets of variates." In: *Biometrika*. 1936.

[3] Simon Kornblith et al. "Similarity of neural network representations revisited." In: *ICML*. 2019.

[4] Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. "Representational similarity analysis-connecting the branches of systems neuroscience." In: *Frontiers in systems neuroscience* (2008).

[5] Martin Schrimpf et al. "Brain-score: Which artificial neural network for object recognition is most brain-like?" In: *bioRxiv* (2018).

[6] Margot E Kaminski and Jennifer M Urban. "The right to contest AI." In: *Columbia Law Review* (2021).

[7] Mauritz Kop. "EU artificial intelligence act: the European approach to AI." In: Stanford-Vienna Transatlantic Technology Law Forum, Transatlantic Antitrus and IPR Developments. 2021.

[8] Thomas Fel et al. "A holistic approach to unifying automatic concept extraction and concept importance estimation." In: *NeurIPS* (2023).

[9] Hoagy Cunningham et al. "Sparse autoencoders find highly interpretable features in language models." In: *ICLR*. 2024.

[10] Maithra Raghu et al. "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability." In: *NeurIPS* (2017).

[11] Yixuan Li et al. "Convergent learning: Do different neural networks learn the same representations?" In: *International Workshop on Feature Extraction: Modern Questions and Challenges at NeurIPS*. 2015.

[12] Minyoung Huh et al. "The platonic representation hypothesis." In: *ICML*. 2024.

[13] Max Klabunde et al. "Similarity of neural network models: A survey of functional and representational measures." In: *arXiv:2305.06329* (2023).

[14] Zhenda Xie et al. "Revealing the dark secrets of masked image modeling." In: *CVPR*. 2023.

[15] Thao Nguyen, Maithra Raghu, and Simon Kornblith. "Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth." In: *ICLR*. 2021.

[16] Maithra Raghu et al. “Do vision transformers see like convolutional neural networks?” In: *NeurIPS* (2021).

[17] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. “What is being transferred in transfer learning?” In: *NeurIPS* (2020).

[18] Young-Jin Park et al. “Quantifying Representation Reliability in Self-Supervised Learning Models.” In: *UAI*. 2024.

[19] Wentao Zhang et al. “Efficient diversity-driven ensemble for deep neural networks.” In: *ICDE*. 2020.

[20] Yoonho Lee, Huaxiu Yao, and Chelsea Finn. “Diversify and disambiguate: Out-of-distribution robustness via disagreement.” In: *ICLR*. 2023.

[21] Robin Rombach, Patrick Esser, and Bjorn Ommer. “Network-to-network translation with conditional invertible neural networks.” In: *NeurIPS* (2020).

[22] Amil Dravid et al. “Rosetta neurons: Mining the common units in a model zoo.” In: *ICCV*. 2023.

[23] Lukas Muttenthaler et al. “Human alignment of neural network representations.” In: *ICLR*. 2023.

[24] Thomas Fel et al. “Harmonizing the object recognition strategies of deep neural networks with humans.” In: *NeurIPS* (2022).

[25] Jannis Ahlert et al. “How aligned are different alignment metrics?” In: *ICLR Workshop on Representational Alignment (Re-Align)*. 2024.

[26] Cian Eastwood and Christopher KI Williams. “A framework for the quantitative evaluation of disentangled representations.” In: *ICLR*. 2018.

[27] Cian Eastwood et al. “Dci-es: An extended disentanglement framework with connections to identifiability.” In: *ICLR*. 2023.

[28] Francesco Locatello et al. “Challenging common assumptions in the unsupervised learning of disentangled representations.” In: *ICML*. 2019.

[29] Karsten Roth et al. “Disentanglement of correlated factors via hausdorff factorized support.” In: *ICLR*. 2023.

[30] Sunny Duan et al. “Unsupervised model selection for variational disentangled representation learning.” In: *ICLR*. 2020.

[31] Ramprasaath R Selvaraju et al. “Grad-CAM: visual explanations from deep networks via gradient-based localization.” In: *IJCV* (2020).

[32] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ““Why should I trust you?" Explaining the predictions of any classifier.” In: *KDD*. 2016.

[33] Scott M. Lundberg and Su-In Lee. “A unified approach to interpreting model predictions.” In: *NeurIPS*. 2017.

[34] Been Kim et al. “Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TACV).” In: *ICML*. 2018.

[35] Amirata Ghorbani et al. “Towards automatic concept-based explanations.” In: *NeurIPS* (2019).

[36] Ruihan Zhang et al. “Invertible concept-based explanations for cnn models with non-negative concept activation vectors.” In: *AAAI*. 2021.

[37] Matthew Kowal et al. “Understanding Video Transformers via Universal Concept Discovery.” In: *CVPR*. 2024.

[38] Eleonora Poeta et al. “Concept-based explainable artificial intelligence: A survey.” In: *arXiv:2312.12936* (2023).

[39] David Bau et al. “Understanding the role of individual units in a deep neural network.” In: *PNAS* (2020).

[40] Jessica Schrouff et al. “Best of both worlds: local and global explanations with human-understandable concepts.” In: *arXiv:2106.08641* (2021).

[41] Thomas Fel et al. “CRAFT: Concept recursive activation factorization for explainability.” In: *CVPR*. 2023.

[42] Reduan Achtibat et al. “From attribution maps to human-understandable explanations through concept relevance propagation.” In: *Nature Machine Intelligence* (2023).

[43] Neehar Kondapaneni et al. “Less is more: Discovering concise network explanations.” In: *ICLR 2024 Workshop on Representational Alignment*. 2024. URL: https://openreview.net/forum?id=JBwpD6Yy8Q.

[44] Andrew Ilyas et al. “Datamodels: Predicting predictions from training data.” In: *ICML*. 2022.

[45] Harshay Shah et al. “Modeldiff: A framework for comparing learning algorithms.” In: *ICML*. 2023.

[46] Kaiming He et al. “Deep residual learning for image recognition.” In: *CVPR*. 2016.

[47] Alexey Dosovitskiy. “An image is worth 16x16 words: Transformers for image recognition at scale.” In: *ICLR*. 2021.

[48] Mathilde Caron et al. “Emerging properties in self-supervised vision transformers.” In: *ICCV*. 2021.

[49] Kaiming He et al. “Masked autoencoders are scalable vision learners.” In: *CVPR*. 2022.

[50] Ross Wightman. *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. 2019. DOI: 10.5281/zenodo.4414861.

[51] Jia Deng et al. “Imagenet: A large-scale hierarchical image database.” In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[52] Grant Van Horn et al. “Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection.” In: *CVPR*. 2015.

[53] Josh Achiam et al. “Gpt-4 technical report.” In: *arXiv:2303.08774* (2023).

[54] Yossi Gandelsman, Alexei A. Efros, and Jacob Steinhardt. “Interpreting CLIP’s Image Representation via Text-Based Decomposition.” In: *ICLR*. 2024.

[55] Jonathan Krause et al. “3D Object Representations for Fine-Grained Categorization.” In: *ICCV Workshops*. 2013.

[56] Chris HQ Ding, Tao Li, and Michael I Jordan. “Convex and semi-nonnegative matrix factorizations.” In: *PAMI* (2008).

[57] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic attribution for deep networks.” In: *ICML*. 2017.

[58] Thomas Fel et al. “Xplique: A Deep Learning Explainability Toolbox.” In: *Workshop on Explainable Artificial Intelligence for Computer Vision at CVPR* (2022).

[59] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python.” In: *JMLR* (2011).

[60] Robert Tibshirani. “Regression shrinkage and selection via the lasso.” In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* (1996).

[61] Mathurin Massias, Alexandre Gramfort, and Joseph Salmon. “Celer: a fast solver for the lasso with dual extrapolation.” In: *ICML*. 2018.

Chapter 4

REPRESENTATIONAL DIFFERENCE EXPLANATIONS

Neehar Kondapaneni, Oisin Mac Aodha, and Pietro Perona. "Representational Difference Explanations." In: *arXiv preprint arXiv:2505.23917* (2025). URL: https://arxiv.org/abs/2505.23917.

4.1 Abstract

We propose a method for discovering and visualizing the differences between two learned representations, enabling more direct and interpretable model comparisons. We validate our method, which we call *Representational Differences Explanations* (RDX), by using it to compare models with known conceptual differences and demonstrate that it recovers meaningful distinctions where existing explainable AI (XAI) techniques fail. Applied to state-of-the-art models on challenging subsets of the ImageNet and iNaturalist datasets, RDX reveals both insightful representational differences and subtle patterns in the data. Although comparison is a cornerstone of scientific analysis, current tools in machine learning, namely post hoc XAI methods, struggle to support model comparison effectively. Our work addresses this gap by introducing an effective and explainable tool for contrasting model representations.

4.2 Introduction

In recent years, deep learning researchers and engineers have explored the costs and benefits of using larger datasets and more complex architectures. These changes have often led to distinct models with different representations of the same data. An intuitive approach to understanding the effects of different architectures and training choices is to analyze the *representational differences* between models. Dictionary learning (DL)-based explainable AI (XAI) methods, like sparse autoencoders (SAEs) and non-negative matrix factorization (NNMF), are powerful tools for analyzing model representations that surface model *concepts*, i.e. semantically meaningful sub-components of the input data [2, 3, 4, 5, 6, 7, 8]. These approaches are formulated as a dictionary learning problem [9] such that model representations are decomposed into a linear combination of learned concept vectors. Concept vectors are then converted into human-friendly explanations by selecting a subset of input items (e.g. a set of images) that maximally align with the vector. These explanations have been

shown to help users better understand models [2, 4, 8, 10]. However, when adapting existing DL-based XAI methods for comparing models with known differences, we find that they often generate explanations that are unrelated to the known difference between models (Sec. 4.5.2 and Sec. 4.5.3).

We identify three issues with existing DL-based XAI methods that limit their power of analysis, especially when *comparing* representations. First, when representational differences are relatively small, concepts from different models tend to overlap and it is thus difficult to spot differences. Second, we observe that existing methods visualize concepts by sampling items with the largest activations [2, 4, 6, 7, 11], which tend to be the ones that are easiest to interpret, and miss more nuanced differences, thus offering *incomplete* explanations. Finally, to understand nuanced differences between models, users need to consider the weighted sum of several concepts *via their incomplete explanations* which can be imprecise and difficult.

We propose a new XAI method, named *Representational Difference Explanations* (`RDX`) for explaining model differences. Rather than focusing on one representation at a time, `RDX` compares two representations against each other to isolate the *differences* between them (Fig. 6.1). Additionally, unlike DL-methods which generate *incomplete* explanations for concept vectors, we take the perspective that a concept and its explanation are *one and the same thing*. We enforce that a concept is equivalent to a small set of similar inputs that can be interpreted by a human.

We make the following three contributions:

- A new method, `RDX`, that identifies explainable differences between model representations.
- A metric to measure the effectiveness of such representation difference explanations.
- Experiments comparing `RDX` against baseline methods for explaining model differences.

4.3 Related Work

Explainability. Explainable AI methods for computer vision attempt to generate explanations to help users understand model behavior. There are two broad classes of methods: *local methods* [12, 13, 14, 15] attribute regions of an image to a model's decision and often take the form of heatmaps, while *global methods* [2, 3, 16] generate a global explanation (e.g. a grid of images or image regions) that represent a visual concept that is learned by a model. Visual concepts are usually defined as images that the model considers to be similar. These visual concepts help users achieve a

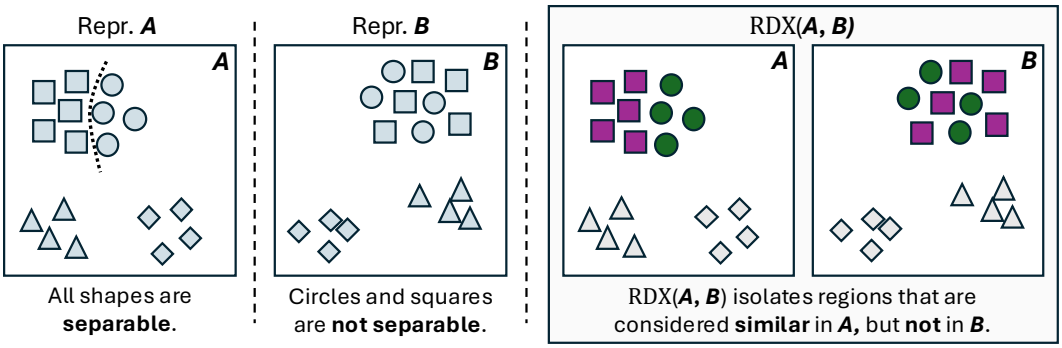


Figure 4.1: Intuition behind our method. *Representational Difference Explanations* (RDX) aim to highlight the substantive differences between two representations (e.g. $\boldsymbol{A}$ and $\boldsymbol{B}$, which are the embedding matrices produced by two different models for the same set of data). Here $\boldsymbol{A}$ supports discrimination between circles and squares, whereas $\boldsymbol{B}$ does not. Clustering the two representations independently would not reveal the square/circle sub-structure in $\boldsymbol{A}$. By "subtracting" $\boldsymbol{B}$ from $\boldsymbol{A}$, RDX reveals which items are considered similar in $\boldsymbol{A}$, but not in $\boldsymbol{B}$. RDX isolates *differences*, and ignores data that may be equally well grouped in both representations, such as the triangles and diamonds.

more general understanding of model behavior [2, 4, 8, 10, 17]. For example, they can reveal that a model has learned to use water as a cue for detecting a certain species of waterbird. Local and global methods can also be combined to provide detailed explanations that describe both the concepts and the image regions used by a model when making a decision [4, 5, 6, 18, 19].

Representational Similarity. Representational similarity methods [20, 21, 22, 23, 24] aim to quantify the similarity between network representations. These methods operate by passing the same set of items through two models to generate two embedding matrices. These embedding matrices are then compared, resulting in a single value that quantifies their degree of similarity. While these approaches can provide useful, coarse-grained insights [25, 26, 27, 28, 29, 30, 31], they do not help with understanding fine-grained model differences. Recently, several methods have been proposed which compare networks through interpretable concepts [5, 6, 17]. Both RSVC and USAEs [5, 17] extract concepts independently for each model and match them in a subsequent step, resulting in partially overlapping concepts that can make interpretation challenging. USAEs[6] use a "universal" sparse autoencoder to learn a common representational space across several models, making it challenging to analyze new models or datasets. Additionally, none of these methods are designed to specifically seek out differences, although differences may be detected as a

byproduct of their approaches. In contrast, our approach uses information from both representations simultaneously to discover differences between them. It requires no training, making it easy to apply to new models and bespoke datasets.

Comparing Graphs. Our approach is related to graph comparison methods. Many methods have been developed for comparing graphs, including methods for matching the largest common subgraphs [32], detecting anomalies [33], grouping network types [34], and measuring the similarity between graphs via kernels [35]. The majority of existing methods are concerned with developing specialized strategies for handling challenges associated with comparing very large web-scale graphs with mismatched nodes. In addition, these approaches aim to quantify network similarity with a score rather than to visualize and understand qualitative differences. In contrast, our approach is designed to provide fine-grained, qualitative understanding of the differences between two "graphs" that have the same nodes, but different edge weights. While some approaches [36, 37] have been developed to visualize differences between graphs, these methods focus on highlighting the addition and removal of nodes. Most relevant to our work is DiSC [38], a modification of the spectral clustering algorithm. DiSC addresses a setting in which there are two experimental conditions, where the same types of measurements are taken in both conditions. Given this shared feature space, it seeks out *features* that cluster together in one condition, but not in the other. This paradigm is relevant for biological experiments, in which genes may co-activate in certain experimental conditions. Our approach differs from DiSC in two key ways: (1) Neural networks do not have a shared feature space, therefore we focus on discovering differential clusters of *inputs*, not features. (2) We construct an affinity matrix emphasizing the *difference* between representations. This makes our approach more flexible than DiSC, since it can be used with any clustering algorithm.

4.4 Method

We propose a method, `RDX`, to explain differences between models via concepts by identifying inputs that *only* one of the two models considers to be semantically related. To do so, we construct an affinity matrix that assigns high affinity to pairs of inputs that are similar, according to representation $\boldsymbol{A}$ and dissimilar according to representation $\boldsymbol{B}$, and perform clustering to reveal distinctive similarity structures in $\boldsymbol{A}$. At a high-level, `RDX` performs the following steps: (1) compute the pairwise distances between inputs in $\boldsymbol{A}$ and $\boldsymbol{B}$ to build distance matrices, $\boldsymbol{D}_A$ and $\boldsymbol{D}_B$, (2) compute the *normalized* difference between the matrices to form *difference* matrices

$\boldsymbol{G}_{A,B}$ and $\boldsymbol{G}_{B,A}$, and (3) use the difference matrix to sample difference explanations, i.e. explanations that reveal where the two representations disagree. Intuitively, negative edges in $\boldsymbol{G}_{A,B}$ indicate that the corresponding pair of inputs were closer together in $\boldsymbol{A}$ than they were in $\boldsymbol{B}$.

As input, we have n data items from which we compute two embedding matrices obtained from two different models, $\boldsymbol{A} \in \mathbb{R}^{n \times d_A}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times d_B}$, where d_A and d_B are the embedding dimensions for models A and B respectively. $\boldsymbol{A}$ and $\boldsymbol{B}$ contain embeddings over the same set of inputs, where each row corresponds to the same input item, i.e. each row is an embedding vector. We refer to the ith embedding vector in $\boldsymbol{A}$ as $\boldsymbol{a}_i$. We consider several options for each step of `RDX` and provide details for the best choices in the following sections. Additional model variants are described in Sec. 4.9.

4.4.1 Computing Normalized Distances

To contrast representations using their distance matrices, the distances must be comparable.

Neighborhood Distances. We compute the pairwise Euclidean distance matrices, $\boldsymbol{D}_A \in R^{n \times n}$ and $\boldsymbol{D}_B \in R^{n \times n}$, for each embedding matrix separately. For each entry $\boldsymbol{a}_i$ in a given embedding matrix, we rank all other entries by their distance to $\boldsymbol{a}_i$. This rank is used as the scale-invariant nearest neighbor distance between $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$. Thus, distances are integers between 0 and n. We refer to the outputs as the normalized distance matrices $\bar{\boldsymbol{D}}_A$ and $\bar{\boldsymbol{D}}_B$.

4.4.2 Constructing Difference Matrices

Given $\bar{\boldsymbol{D}}_A$ and $\bar{\boldsymbol{D}}_B$, we develop a method for comparing the normalized distances that emphasizes differences of where either model considers two inputs similar. The method is asymmetric. Here we present it in one direction.

Locally Biased Difference Function

Consider two pairs of embeddings with indices i, j and i, k. Suppose $\bar{\boldsymbol{D}}_A^{ij} = 500$ and $\bar{\boldsymbol{D}}_B^{ij} = 600$ in the first pair, and $\bar{\boldsymbol{D}}_A^{ik} = 1$ and $\bar{\boldsymbol{D}}_B^{ik} = 101$ in the second. Comparing the distances across the representations $(500 - 600,\ 1 - 101)$ results in the same amount of change $(-100,\ -100)$, but a change from a distance of 1 to a distance of 101 suggests a more important conceptual difference. To address this issue, we propose a locally-biased difference function:

$$\boldsymbol{G}_{A,B}^{ij} = \tanh(\gamma \cdot (\bar{\boldsymbol{D}}_A^{ij} - \bar{\boldsymbol{D}}_B^{ij})/(\min(\bar{\boldsymbol{D}}_A^{ij}, \bar{\boldsymbol{D}}_B^{ij})). \tag{4.1}$$

By dividing by the minimum distance across both representations, this function prioritizes differences in embedding distances in which either representation considers the embeddings to be similar. This ensures that large differences in distant embeddings are ignored, but large differences in nearby embeddings are emphasized. To avoid exponential growth in our difference function when distances are small, we apply a tanh function to normalize the outputs, with γ controlling how quickly the function saturates. Given an item indexed by i, this function will output negative values when the distance between i and j is smaller in $\boldsymbol{A}$ than in $\boldsymbol{B}$. Thus, negative matrix entries denote items that are closer in $\boldsymbol{A}$ than in $\boldsymbol{B}$.

4.4.3 Sampling Difference Explanation Grids

The next step is to communicate representational differences to the user. For visual data, sets of images, presented as an image grid of 9-25 images, have been used to communicate visual concepts [2, 4]. We aim to sample m sets of images (i.e. image grids) from the difference matrix. Each set of images should contain images that are considered similar by only $\boldsymbol{A}$, i.e. indices that have pairwise negative matrix entries in $\boldsymbol{G}^{ij}_{A,B}$. We refer to this set of images as a *difference explanation* (E) that defines a concept unique to one model and we refer to the collection of difference explanations as $\mathcal{E}^{\boldsymbol{A}}$. If we consider the difference matrix as the adjacency matrix of a graph, we are essentially looking for a subgraph of size $|E|$ with large negative values on all edges. There are many options for sampling subgraphs, we consider a direct sampling (see Sec. 4.9.4) and a spectral clustering based approach.

Spectral Clustering. We convert $\boldsymbol{G}_{A,B}$ into an affinity matrix: $\boldsymbol{F}_{A,B} = \exp(-\beta \cdot \boldsymbol{G}_{A,B})$. To ensure the affinity matrix is symmetric, we average it with its transpose. From this affinity matrix, we seek to sample a set of m difference explanation grids $\mathcal{E}^{\boldsymbol{A}}$. Given an affinity matrix, spectral clustering solves a relaxed version of the normalized cut problem [39]. Normalized cuts (N-Cut) seek out a partition of a graph that minimizes the sum of the cut edges, while balancing the size of the partition [40]. Since, edges in $\boldsymbol{F}_{A,B}$ are large when inputs are closer in $\boldsymbol{A}$ than they are in $\boldsymbol{B}$, spectral clustering is biased to finding partitions in which inputs are close together in $\boldsymbol{A}$, but far apart in $\boldsymbol{B}$. In practice, when both representations have a similar structure, edges in that structure will have an affinity close to 1, since the difference is near 0. To discard these regions, we generate $m + 1$ clusters and discard the cluster with lowest mean affinity as it represents regions we are uninterested in. Spectral clusters can contain too many inputs to be visualized all at once. To convert each cluster into an explanation grid (E), we define the k-neighborhood affinity (KNA).

For each image in the spectral cluster, the KNA is the sum of the edges between that image and its k-nearest neighbors (also from within the cluster). Recall that larger affinity edges indicate more disagreement about the similarity between two images, thus we select the image and neighbors corresponding to the max KNA (see pseudo-code Sec. 4.9.1).

4.4.4 Representational Alignment

When models have significant representational differences, it is possible that these differences could be mitigated by aligning the representations. For example, both models may be the same up to some (e.g. linear) transformation. In these settings, it can be useful to first maximize the alignment between models before exploring the representational differences, since this can reveal fundamental differences between the models. To align the representation $\boldsymbol{A} \in \mathbb{R}^{n \times d_A}$ to $\boldsymbol{B} \in \mathbb{R}^{n \times d_B}$, we learn a transformation matrix $\boldsymbol{M}_{AB} \in \mathbb{R}^{d_A \times d_A}$ that minimizes the centered kernel alignment (CKA) loss [41] between the transformed $\boldsymbol{A}$ and $\boldsymbol{B}$:

$$\boldsymbol{M}^*_{AB} = \arg\min_{\boldsymbol{M}} 1 - \text{CKA}(\boldsymbol{AM}, \boldsymbol{B}), \tag{4.2}$$

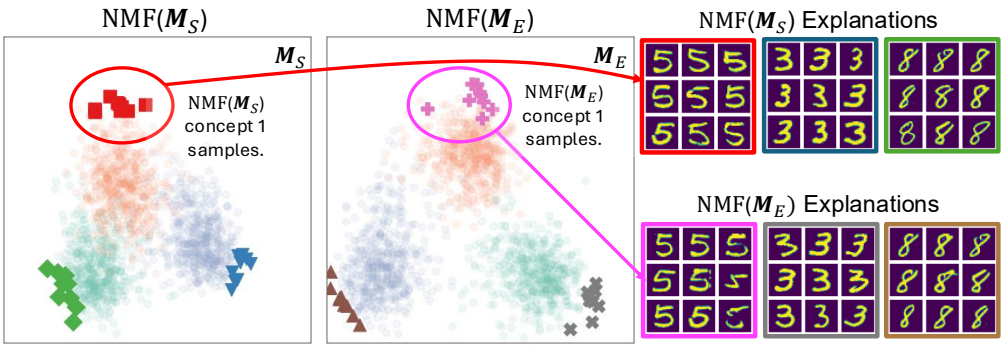
where $\text{CKA}(\cdot, \cdot)$ represents the linear CKA similarity. We denote aligned representations as $\boldsymbol{A}'$. See Sec. 4.10.2 for training details.

4.4.5 Evaluating Explanations

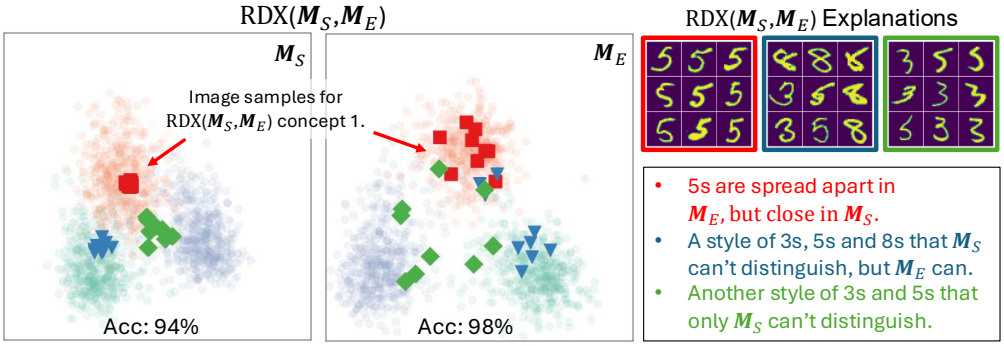
All explanation methods in this work produce a set of explanation grids for both representations. Recall that $\mathcal{E}^{\boldsymbol{A}}$ is the set of explanation grids for $\boldsymbol{A}$. The goal of a difference explanation grid is to identify sets of items that are closer together in one representation than they are in the other. We develop a metric, the binary success rate (BSR), to evaluate whether a method has succeeded at this task. $\bar{\boldsymbol{D}}^{ij}_A$ represents the distance between two items in representation $\boldsymbol{A}$ and $\bar{\boldsymbol{D}}^{ij}_B$ represents the same for representation $\boldsymbol{B}$. We measure how frequently the distance for a pair of items from an explanation grid is smaller in $\boldsymbol{A}$ than in $\boldsymbol{B}$. In the pseudo-code above, we provide the algorithm for computing $\text{BSR}(\mathcal{E}^{\boldsymbol{A}})$.

Algorithm 1: Evaluation of Explanations on Representation A.

Input: Grids $\mathcal{E}^{\boldsymbol{A}} = \{E_1, \ldots, E_k\}$, and distances $\bar{\boldsymbol{D}}_A, \bar{\boldsymbol{D}}_B$
`BSR = 0`
for each grid $E \in \mathcal{E}^{\boldsymbol{A}}$ **do**
    for each pair $(i, j) \in e, i \neq j$ **do**
        if $\bar{\boldsymbol{D}}^{ij}_A < \bar{\boldsymbol{D}}^{ij}_B$ **then**
            `BSR += 1`
        end if
    end for
end for



(A) NMF explanations are indistinguishable despite significant differences in representation.



(B) RDX highlights differences between M_S and M_E, by identifying concepts specific to M_S.

Figure 4.2: Comparing RDX to NMF. We train a small CNN on a modified MNIST dataset that only contains images of the digits 3, 5, and 8. We compare a strong model checkpoint representation (M_S, 94% accuracy) with a final 'expert' model representation (M_E, 98% accuracy). The left and middle columns show PCA projections of the M_S and M_E representations, respectively. The transparent colors indicate classes in the dataset: 3 (light-blue), 5 (light-orange), and 8 (light-green). The right most columns visualize the images selected by the explanation methods. We extract three concepts for each method. **(A)** We generate explanations using NMF [4] with maximum sampling [4, 9, 17]. The bold colored points on the PCA plots indicate the location of the sampled images. The images can be seen in the right-most column. We show that this approach for concept visualization results in indistinguishable explanations despite both models containing significant representational differences. **(B)** We use RDX to generate three explanations for images that are considered highly similar by M_S, but dissimilar by M_E. We overlay the points sampled by this approach on both models' representations. In the right column, we visualize the explanations.

4.5 Results

We conduct three types of experiments to evaluate the effectiveness of RDX. (Section 4.5.2) We compare two simple representations with subtle differences to show that existing XAI methods fail to explain these differences. (Section 4.5.3) We show that these trends continue to hold in more realistic settings. Through modifications

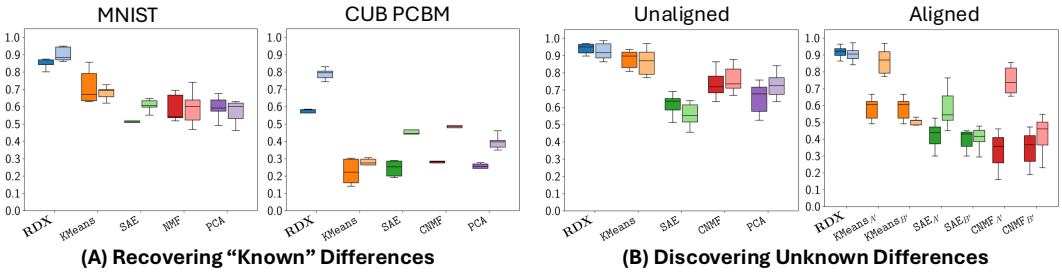


Figure 4.3: Binary success rate evaluation of XAI methods. For each XAI method, we compute the binary success rate (BSR) (Sec. 4.4.5, where higher is better) on all difference experiments. We use neighborhood distances to measure BSR (Sec. 4.4.1). Each method (x-axis) is assigned a different color, we show BSR($\mathcal{E}^{\boldsymbol{A}}$) (darker box) and BSR($\mathcal{E}^{\boldsymbol{B}}$) (lighter box). **(Panel A)** We show results on the MNIST and CUB PCBM experiments (Sec. 4.5.3), in which we modify the representation and test if RDX can help identify the modification. **(Panel B)** We show results when comparing large vision models with unknown differences (Sec. 4.5.4). We compare recovering differences with and without an initial alignment step (Sec. 4.4.4). In all cases, our RDX approach consistently outperforms the dictionary learning baselines.

of various models, we manipulate representations to have "known" differences. We then stage comparisons and assess whether existing XAI methods are able to recover these differences. (Section 4.5.4) We use RDX to compare models with unknown differences and find that it can reveal novel insights about models and datasets.

4.5.1 Implementation Details

We use several models in our evaluation. Unless specified otherwise, for our modified MNIST experiments, we use a 2-layer convolutional network with an output dimension of 64. We also train a post-hoc concept bottleneck model (PCBM) [42] with a ResNet-18 [43] backbone on the CUB dataset [44] using the standard training procedure [42]. Finally, we conduct experiments using several models that are available from the timm library [45]: DINO [46] vs. DINOv2 [47] and CLIP [48] vs. CLIP-iNat (i.e. a CLIP model fine-tuned on data from the iNaturalist platform [49]). In these experiments, models are compared on subsets of images from 2-4 commonly confused classes in ImageNet [50] or iNaturalist [49]. More training details are provided in Sec. 4.10.1. We compare our approach to several DL for XAI baselines: sparse auto-encoders (SAE) [51], non-negative matrix factorization (NMF) [52], principal component analysis (PCA) [53], and KMeans [54]. We use convex non-negative matrix factorization (CNMF) [55] if the activations of the last layer contains negative values. We provide details on baseline methods in Sec. 4.10.3 and details for RDX are provided in Sec. 4.10.4. We conduct ablations in Sec. 4.9.

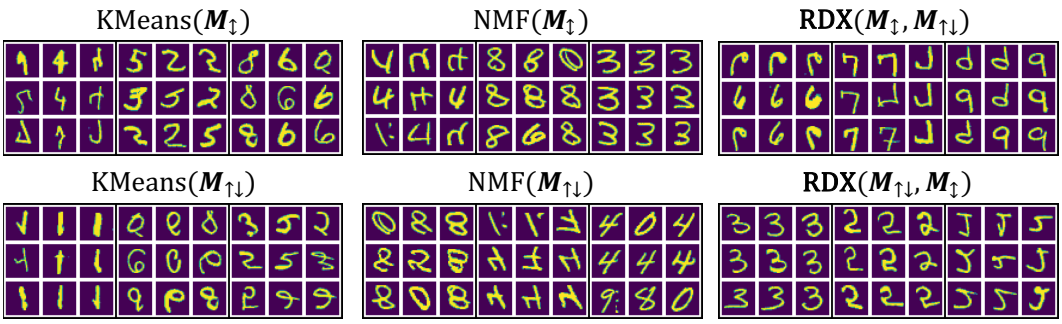


Figure 4.4: Recovering vertical flip modifications on MNIST. Explanations produced by three XAI methods, `RDX`, `KMeans`, and `NMF`, to compare models $M_{\updownarrow}$ and $M_{\uparrow\downarrow}$. Both models are trained on a dataset with vertically flipped and normal images. $M_{\updownarrow}$ is trained to associate the original label to flipped digits and $M_{\uparrow\downarrow}$ is trained to predict a new set of labels for flipped digits. We expect $M_{\updownarrow}$ to mix flipped and unflipped digits while $M_{\uparrow\downarrow}$ should separate them. We generate 3 explanations for each method. **(Col 1, 2)** `KMeans` and `NMF` generate explanations that are difficult to understand. **(Col 3)** `RDX` captures the expected difference. `RDX`($M_{\updownarrow}M_{\uparrow\downarrow}$) reveals that $M_{\updownarrow}$ represents flipped and normal 6s, 7s, and 9s closer together than in $M_{\uparrow\downarrow}$. `RDX`($M_{\uparrow\downarrow}$, $M_{\updownarrow}$) shows that $M_{\uparrow\downarrow}$ has clean clusters of 3s, flipped 5s, and flipped 2s without any mixing.

4.5.2 Dictionary Learning Fails to Reveal Differences in Similar Representations

Dictionary learning (DL) approaches for XAI are commonly used to discover and explain vision models [4, 5, 6, 16]. We hypothesize that explanation grids sampled from DL concepts are not helpful for describing *differences* between similar representations, even if the representations contain behaviorally significant differences. To test this, we train a 2-layer convolutional network with an output dimension of 8 on a modified MNIST dataset that contains only images for the digits 3, 5, and 8. We compare a checkpoint from epoch 1 with strong performance (94% accuracy) to the final, 'expert' checkpoint at epoch 5 (98%). We refer to these checkpoints as M_S (strong model) and M_E (expert model). We conduct this experiment to assess if an XAI method can reveal subtle differences between two models.

A good difference explanation should reveal the concepts that explain why M_S under-performs M_E by 4%. In Fig. 4.2A we show that `NMF` with maximum sampling generates effectively the same explanation grid for both representations. This is because `NMF` has learned highly similar concepts for both representations, and the representational differences are captured in smaller and noisier concept coefficients

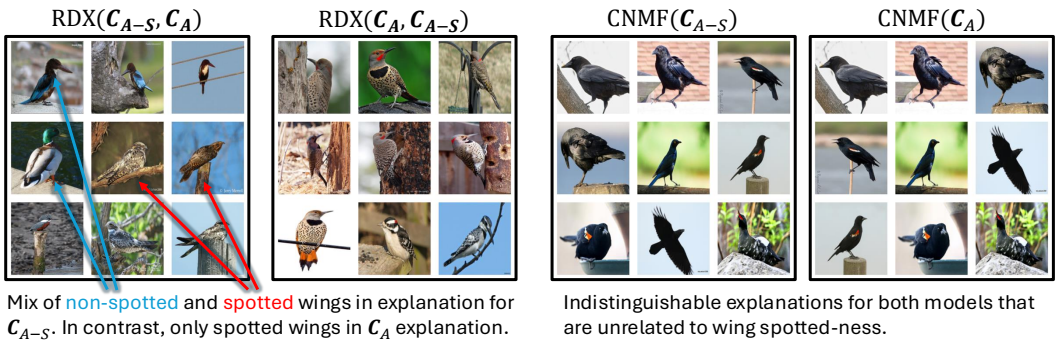


Figure 4.5: Recovering the "Spotted Wing" concept in CUB. We train a post-hoc concept bottleneck model on the CUB dataset. For each image, we use the predicted concept logits as the image's embedding vector (i.e. representation). Here we compare a model using the complete concept representation ($\boldsymbol{C}_A$) with a model representation *without* the spotted wing concept ($\boldsymbol{C}_{A-S}$). We visualize one of five generated explanations for each model using `RDX` and `CNMF`. We observe that `RDX`'s explanation focuses on the spotted wing concept and that $\boldsymbol{C}_{A-S}$ considers images with and without spotted wings to be more similar than $\boldsymbol{C}_A$ does. In contrast, the `CNMF` explanations for each model are both unrelated to the spotted wing concept and indistinguishable from each other, since the representations are highly similar and `CNMF` discovers nearly the same concepts in both.

for images that $\boldsymbol{M}_S$ is less certain about. Maximum sampling selects the images with the largest coefficients, meaning these images are not sampled when visualizing concepts (Fig. 4.8A). In Fig. 4.7, we show that `SAE` and `KMeans` also fail to explain representational differences. An alternative approach to understanding differences could be to inspect individual images of interest and try to understand them through their concepts. In Fig. 4.8B, we show the difficulty of reasoning about an image via a weighted combination of concept explanations. In contrast, `RDX` concepts are defined by their explanation grid and are sampled from difference regions. This ensures that `RDX` explanations select images considered similar by $\boldsymbol{M}_S$ that $\boldsymbol{M}_E$ does not consider similar. In Fig. 4.2B, we can see that $\boldsymbol{M}_S$ is confused by certain styles of 3s, 5s, and 8s that look similar when compared to $\boldsymbol{M}_E$. In Fig. 4.7C we visualize the reverse direction for `RDX` and find that $\boldsymbol{M}_E$ contains clusters of challenging 3s and 5s, that are confused by $\boldsymbol{M}_S$. Finally, in Sec. 4.8.1 we discuss if perfectly monosemantic DL concepts would solve these issues. We argue that monosemanticity is likely infeasible when trying to compare representational differences and that, even if achieved, it cannot solve the issue of weighted combinations of explanations.

4.5.3 RDX Recovers "Known" Differences

Here we evaluate different XAI approaches by comparing MNIST trained models that have a modified training procedure. We select modifications such that we can have strong expectations on the differences between the learned representations. Specifically, we trained on a MNIST dataset with vertically flipped digits, where $\boldsymbol{M}_{\updownarrow}$ was trained to use the same labels for both normal and flipped digits and $\boldsymbol{M}_{\uparrow\downarrow}$ was given new labels for flipped digits. The training modifications and expected representational changes are provided in Tab. 4.4. In Fig. 4.4, we visualize the outputs of three XAI methods for comparing $\boldsymbol{M}_{\updownarrow}$ and $\boldsymbol{M}_{\uparrow\downarrow}$. We clearly see that RDX's explanations focus on the actual expected difference. It shows that $\boldsymbol{M}_{\updownarrow}$ considers flipped and normal digits as being more similar than $\boldsymbol{M}_{\uparrow\downarrow}$. In contrast, KMeans and NMF result in unfocused and seemingly random explanations. In Fig. 4.3 we can see that this trend is consistent as all baseline methods have a lower BSR than RDX.

To explore differences between models trained on more complex images, we use a post-hoc concept bottleneck model (PCBM) trained on the CUB bird species dataset (Sec. 4.10.1). The CUB PCBM ($\boldsymbol{C}_A$) predicts a score for 112 human-defined concepts, these concepts are then used to make species classification decisions, where we treat the concept predictions as a feature vector for an image. In each comparison, we remove a single concept from the feature vectors and compare the representations. The list of eliminated concepts used in this experiment can be found in Tab. 4.4. In Fig. 4.3A (right) we report the BSR score for each method for this experiment. We find that RDX variants perform better than the baselines, especially for difference explanations on the complete representation $\boldsymbol{C}_A$. In Fig. 4.5, we visualize the outputs of RDX and CNMF when comparing a model without the spotted wing concept ($\boldsymbol{C}_{A-S}$) against $\boldsymbol{C}_A$. As expected, we find that difference explanations show that $\boldsymbol{C}_{A-S}$ mixes images with and without spots, whereas, $\boldsymbol{C}_A$ is much better at grouping images with spotted wings. In contrast, we show that CNMF can result in both unrelated and indistinguishable explanations. We show more examples in Sec. 4.8.2. Taken together, these results indicate that RDX is capable of revealing how changes in both training and fine-grained concepts can affect a model's representation.

4.5.4 RDX Discovers "Unknown" Differences

In our final experiment, we test the effectiveness of RDX for knowledge discovery by applying it to two models with unknown differences. We compare DINO with DINOv2 on four groups of ImageNet classes. We also compare CLIP against an iNaturalist fine-tuned CLIP (CLIP-iNat) model on three groups of different species. We conduct

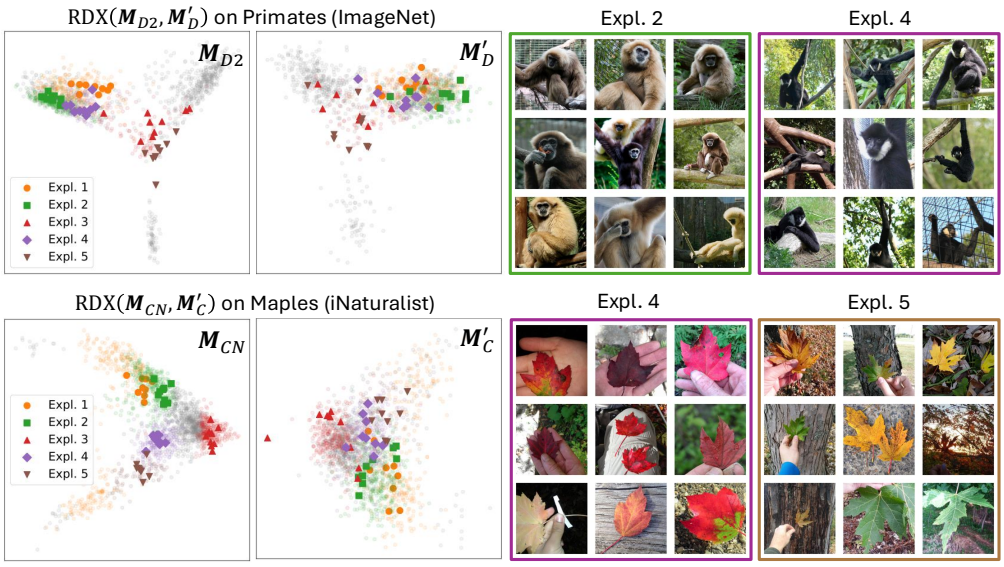


Figure 4.6: Discovering unknown differences. We use RDX to generate difference explanations for representations with unknown differences. We visualize two comparisons with alignment. In both comparisons, we visualize the shared structure (gray), cluster membership (light colors), and selected samples for explanations (dark colors) on PCA projections of the representations. We can see that the selected indices are grouped compactly in the left PCA plot, but are spread apart in the right one. **(RDX(M_{D2}, M'_D)) on Primates**. We discover unique concepts in DINOv2 for commonly confused primates in ImageNet. In the PCA plot, we see that the green (Expl. 2) and purple (Expl. 4) explanations are cleanly separated in M_{D2}, but mixed in M'_D. The explanations show that only DINOv2 has unique concepts for tan-colored gibbons and for gibbons with white chin fur. **(RDX(M_{CN}, M'_C)) on Maples**. We compare the representations of CLIP-iNat and CLIP on four types of maple trees from iNaturalist. We see that CLIP-iNat contains a unique concept for fall-colored Red Maple leaves (Expl. 4) and a second concept that mixes fall-colored and green Silver Maple leaves (Expl. 5). Further analysis is provided in Sec. 4.5.4.

all of the knowledge discovery experiments with and without alignment. We align one model at a time, resulting in twice the number of comparisons for baseline methods. We can see in Fig. 4.3 (Panel B) that RDX outperforms all baseline methods in discovering representational differences. Additionally, we see that alignment makes it more challenging to discover differences for the baseline methods, but RDX maintains good performance in both settings. In Fig. 4.6 (RDX(M_{D2}, M'_D)), we visualize difference explanations by comparing DINOv2 to DINO using images from three primate classes from ImageNet. We find that DINOv2 does a better job at organizing two types of gibbons with different visual characteristics, suggesting that it would be more capable than DINO at fine-grained classification. In Fig. 4.6 (RDX(M_{CN}, M'_C)),

we visualize fine-grained difference explanations on species of maple trees. We find that *only* CLIP-iNat contains well-separated concepts for two different species of maple, despite both clusters sharing a secondary characteristic of leaves with fall-colors. While CLIP does not mix the images from these concepts, we see that it does not group them as tightly, suggesting it may be organizing images using a different characteristic like color. We apply `RDX` to several more examples in Sec. 4.8.3 and use a vision language model to assist in the analysis. Finally, we also discuss general limitations in Sec. 4.7.

4.6 Conclusions

As models become larger and more powerful they encode more and more different concepts, making it critical to focus our attention on describing unique concepts the models have discovered. In this work, we posit that comparing representations allows us to filter away common structure and reveal concepts that may be more interesting the user. We propose `RDX`, a new approach for isolating the differences between two representations. `RDX` requires no training, can be applied to any model that generates an embedding for an input, and is a general framework that can easily be modified with different choices for its intermediate steps. In our experiments we show that `RDX` is able to recover known model differences, and is also able to surface interesting unknown differences. These differences can teach us both about model differences and also about the training data used. The next step is testing `RDX` in real-life applications to see if it can be used to help experts, such as radiologists or ecologists, discover new structures in their models and datasets.

Acknowledgments

We thank Atharva Sehgal, Rogério Guimarães, and Michael Hobley for providing feedback on the work. OMA was supported by a Royal Society Research Grant. NK and PP were supported by the Resnick Sustainability Institute.

Appendix

4.7 Limitations

Here we discuss some of the limitations of RDX and our analysis.

Compute. Computing and storing the full pairwise distance matrix requires $\mathcal{O}(n^2)$ memory, which may become impractical for large n. In this work, we are able to apply our method to at least 5000 data points and we have not explored larger values of n.

Concept Definition. While our decision to define concepts by an explanation of k images is helpful for users, it does not allow us to communicate concepts like "roundness" that may react linearly over a range of image types. Instead, concepts like "'roundness" would be discretized into sub-concepts that can be communicated by an explanation grid.

BSR agreement with human-interpretability. In Fig. 4.19, we find that two methods can have the same BSR, but have significant differences in what they focus on in their explanations. Thus, we propose that BSR should not be directly optimized for, but should instead be a proxy metric, and that qualitative results should always be used to support BSR scores.

Breadth. Our approach works on any representation, but we focus on vision models in our experiments. Future work should explore if this approach can be useful when comparing text and multi-modal representations.

Utility. We find that RDX explanations are useful for identifying representational differences, but more work needs to be done to link these representational differences to performance differences on specific tasks such as classification. In Sec. 4.8.2 (Maples), we see only some RDX explanations align with differences in classification. RDX is unsupervised in that it only requires two sets of representations as input, but in future work it would be interesting to explore incorporating classifier information into RDX explanations.

4.7.1 Societal Impact

We do not anticipate any specific ethical or usage concerns with the method proposed in this work. We propose a method for model comparison which we hope will improve our understanding of model representations. Deeper insight can lead to better detection of bias, better understanding of methods, and discovery of new knowledge about our datasets that may be beneficial for society. However, better understanding can also amplify negative usages of AI.

4.8 Additional Results

4.8.1 Additional Results on MNIST-[3,5,8]

We train a small convolutional network on a modified MNIST dataset containing only images for digits 3, 5 and 8. We compare two checkpoints from training at epoch 1 ($\boldsymbol{M}_S$) and epoch 5 ($\boldsymbol{M}_E$). These checkpoints differ in representation and overall performance. In Fig. 4.2, we showed the results from applying `NMF` to this setting. Here, we also evaluate `SAE` and `KMeans`.

`SAE` and `KMeans` Fail to Explain Representational Differences

In Fig. 4.7 we visualize the explanations generated by `SAE` and `KMeans`. We find that both methods fail to generate explanations that can help us understand the difference between the two representations. The `SAE` generates confusing explanations that may even be misleading. Surprisingly, the `SAE` explanations for $\boldsymbol{M}_S$ are less mixed than $\boldsymbol{M}_E$, suggesting $\boldsymbol{M}_S$ has a better separated representational space, which we know to be incorrect. This is likely a result of random variations in the concepts discovered by the SAE, a phenomena also observed in [56]. `KMeans`, like `NMF`, generates indistinguishable explanations for both representations. This is due to the images near the centroids of similar representations being effectively the same, since these are regions in which model confidence is higher.

General Issues with Interpreting Dictionary Learning Methods.

There are two critical issues with interpreting explanations from dictionary learning methods. We visualize these issues in Fig. 4.8. We show that visualizing the maximal samples of a concept is an *incomplete* explanation of the behavior of that concept. This is because concepts can activate for multiple groups of images at varying strengths and visualizing the top-k images does not tell users about other types of images a concept may react to. These "other" images, with smaller concept coefficients, are critical for understanding the task of comparing two representations, since they can be the source of representational differences (Fig. 4.8A).

Importantly, this issue raises questions about the feasibility of decomposing models into monosemantic concepts. Monosemantic concepts are defined as concepts that have a single, unambiguous, meaning and extracting them are the goal of sparse autoencoder methods for XAI [7, 11, 57]. Consider a concept vector that encodes the concept of "roundness." This concept is neither monosemantic nor polysemantic, as it is too ambiguous for monosemanticity, but not disparate enough to be polysemantic.

On a dataset of objects that are interpolations from squares to circles, this concept would react to all round objects, but most strongly to circles. A maximally sampled explanation for this concept could easily mislead the user into believing that the concept reacts *only* for circles. Trying to convert the "roundness" concept into discrete monosemantic concepts that only react to well-defined shapes leads to questions about the boundaries of the discretization and the number of concepts that can be meaningfully analyzed by a human. When comparing two models that share the "roundness" concept, differences in discretization could lead to partially overlapping concepts, such as those seen in [17].

When comparing two representations it is sometimes necessary to analyze specific images that the two representations disagree upon. When applying dictionary-learning based methods to understand what concepts make up an image, users are required to mentally perform a weighted combination over incomplete concept explanations (Fig. 4.8B). This task is un-intuitive and imprecise in the context of a single model. If the concepts for the two models being compared are even slightly different, this task becomes essentially impossible. Notably, this issue persists even if concepts are monosemantic since most images are likely to contain several concepts.

Using `RDX` to Discover Concepts Specific to

`RDX` is not a symmetric method. In Fig. 4.7C we visualize the second direction `RDX`($\boldsymbol{M}_E$, $\boldsymbol{M}_S$). These explanations reveal images considered similar in $\boldsymbol{M}_E$, but not in $\boldsymbol{M}_S$. These explanations show that the expert model $\boldsymbol{M}_E$ is able to group challenging images of the same digit that $\boldsymbol{M}_S$ is unable to. Additionally, we note that the explanations exclude 8, suggesting that the difference in similarities between images of 8 in $\boldsymbol{M}_E$ and $\boldsymbol{M}_S$ is smaller than the difference in similarities for images of 3 and 5. Indeed, the prediction agreement for linear classifiers trained on these two representations is 95% on 3, 95% on 5, and 98% on 8 matching our expectations.

4.8.2 Additional Results for "Known" Differences

We describe the modifications to models in known difference comparisons in Tab. 4.4. Comparison details are found in Tab. 4.7.

MNIST. In Fig. 4.10, we visualize the explanations from `RDX`, `KMeans`, and `NMF` for $\boldsymbol{M}_{49}$ vs. $\boldsymbol{M}_b$, $\boldsymbol{M}_{35}$ vs. $\boldsymbol{M}_{49}$, and $\boldsymbol{M}_{hfm}$ vs. $\boldsymbol{M}_{hfs}$.

In all comparisons, `RDX` explanations clearly show the expected difference between the two representations. In contrast, `KMeans` and `NMF` generate unfocused explanations

that are often indistinguishable from each other. At best, we find that the baseline approaches may contain 2/6 explanations focused on the known difference between models.

CUB PCBM. In Fig. 4.11 and Fig. 4.12 we visualize five explanations for comparing $\boldsymbol{C}_{A-S}$ vs. $\boldsymbol{C}_A$ and $\boldsymbol{C}_{A-YB}$ vs. $\boldsymbol{C}_A$ using `RDX` and a baseline method. $\boldsymbol{C}_A$ is the CUB PCBM concept vector with all concepts retained. $\boldsymbol{C}_{A-S}$ removes the spotted wing concept from the concept vector and $\boldsymbol{C}_{A-YB}$ removes the yellow back concept from the concept vector. We expect that explanations are composed of images that contain these concepts. In both comparisons, the baseline method (`CNMF` or `SAE`) generates indistinguishable and unfocused explanations that provide no insight about the known differences. On the other hand, `RDX` explanations focus on the known differences and can reveal interesting insights about the impact of removing a concept. In Fig. 4.12, the `RDX` explanations help teach us about how the PCBM uses the "yellow-back" concept. On first glance, the model without the yellow-back concept ($\boldsymbol{C}_{A-YB}$) appears to do a better job of grouping colorful yellow/red birds. When inspected more closely, it becomes clear that there is a mixture of birds with black faces and colored backs and birds with red/yellow faces and black backs. This indicates that the yellow-back concept is used as a fine-grained discriminator between these two color patterns. It also indicates that the PCBM model may be suffering from leakage [58] and does not discriminate between bright red and yellow colors. In the other direction, we see that the yellow-back concept helps organize birds with colorful backs into organized groups (explanations 3-5), but also helps organize "regular" birds into well-separated groups (explanations 1-2).

4.8.3 Additional Results for Discovering "Unknown" Differences

We visualize complete `RDX` results for four comparisons using the representational alignment step from Sec. 4.4.4:

1. $\boldsymbol{M}_D$ vs. $\boldsymbol{M}_{D2}$ on Mittens (Fig. 4.13)
2. $\boldsymbol{M}_D$ vs. $\boldsymbol{M}_{D2}$ on Primates (Fig. 4.14)
3. $\boldsymbol{M}_C$ vs. $\boldsymbol{M}_{CN}$ on Maples (Fig. 4.15)
4. $\boldsymbol{M}_C$ vs. $\boldsymbol{M}_{CN}$ on Corvids (Fig. 4.16)

See Tab. 4.7, for a description of the datasets and Tab. 4.5 for details about the models. Although explaining classifier predictions is not the goal of `RDX`, we train a

Table 4.1: Linear probe accuracy across datasets described in Tab. 4.7.

	Mittens	**Primates**	**Maples**	**Corvids**
DINO (M_D)	0.933	0.918	–	–
DINOv2 (M_{D2})	**0.965**	**0.921**	–	–
CLIP (M_C)	–	–	0.752	**0.790**
CLIP-iNat (M_{CN})	–	–	**0.796**	0.787

linear classifier on these representations to gain an insight into the quality of their organization and to assist in interpretation (Tab. 4.1). Training details are provided in Sec. 4.10.1. We use the classifier accuracies and predictions as supplemental information to understand representational differences.

In all representational difference comparisons, RDX reveals interesting insights about differences in model representations. In the first two comparisons, models perform well and we are able to interpret the discovered concepts fairly easily without using additional information from the classifiers.

Mittens. We find that DINOv2 does a better job of organizing mittens by their orientation (see Fig. 4.13). It also has a unique concept for children's mittens that is not present in DINO. On the other hand, DINO has unique concepts for children around Christmas related items and Christmas items on their own. These two concepts appear to be entangled in DINOVv2.

Primates. We notice that DINO contains four unique concepts that appear to fixate on secondary characteristics (see Fig. 4.14). Explanation 1 seems to react to siamangs on grass, explanation 3 picks up on lower quality images, like screenshots from a video or images taken through enclosure glass, explanation 4 contains scenes of branches and greenery where primates are distant, and explanation 5 contains scenes of a variety of primates behind fencing. In contrast, DINOv2 explanations tend to be focused on the primate type. For example, explanation 1 also picks up on siamangs, but in a variety of environments. Explanation 3 shows gibbons swinging in trees and explanation 5 shows spider monkeys in trees, these two concepts are entangled in DINO, suggesting DINO is more sensitive to the background/activity of the monkeys than the species.

In the next two comparisons (Maples and Corvids), we evaluate how RDX performs when classifier performance is much lower. We expect these representations to be more poorly organized and subsequently more challenging to interpret. In the first comparison (Maples), we select a comparison where there is a large performance

Table 4.2: Ground truth label counts for explanations on Maples.

Maple Type	RDX(M_C, M'_{CN})					RDX(M_{CN}, M'_C)				
	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5
Norway Maple	1	1	2	0	0	9	0	1	0	0
Silver Maple	3	3	1	2	1	0	0	7	0	9
Sugar Maple	3	3	4	0	0	0	9	0	0	0
Red Maple	2	2	2	7	8	0	0	1	9	0

difference (4%). In the second (Corvids), we explore a setting in which fine-tuning CLIP on iNaturalist images did not improve the quality of the representation, although it may have changed it.

Maples. Fine-grained maple tree classification is a challenging task which is beyond the skill of most people. The clear performance gap between model's indicates that CLIP-iNat has learned important features, we explore if RDX is able to help us generate hypotheses on what those might be (see Fig. 4.15). In Fig. 4.6 we analyzed explanations 4 and 5 from RDX(M_{CN}, M'_C). This allowed us to hypothesize that CLIP was biased towards encoding Maple leaves by color/season rather than species. Despite this bias, we find that classifiers trained on both representations perform reasonably well on these two image grids. This suggests that this representational difference may not be important for classification. However, we notice that there are differences in the classifier predictions for explanations 2 and 3. We use this information to propose some hypotheses. The dataset labels for explanation 2 indicate that all of the images are sugar maples. CLIP-iNat gets 7/9 correct, while CLIP gets 5/9. With this information, we hypothesize that CLIP-iNat is able to detect sugar maples leaves in images with a variety of seasons, backgrounds, and lighting. In explanation 3, we see young, bushy maple trees around rocks and waterways. The classification labels indicate the majority of these images contain silver maples. CLIP-iNat gets 8/9 correct, while CLIP gets only 4/9 correct. This suggests that CLIP-iNat has learned to associate this visual concept with silver maples and that this is an effective strategy for classification on this dataset. We also observe some higher-level characteristics of the explanations when analyzed with their ground truth labels. Explanations for CLIP-iNat tend to have labels that correspond with one of the ground truth labels, while CLIP does not (Tab. 4.2. This make sense, as CLIP-iNat was fine-tuned on a classification dataset.

Corvids. Neither representation supports good classification on the challenging Corvids dataset (see Fig. 4.16). However, RDX reveals some interesting concepts unique to each model. For example, RDX($\boldsymbol{M}_C, \boldsymbol{M}'_{CN}$) explanation 4, shows a CLIP specific concept for Corvid footprints. In explanation 5, we see a concept for flocks of crows. In the other direction, CLIP-iNat has a learned a concept for large ravens in natural settings like hillsides or beaches (explanation 1). It has also learned a concept for crows in urban settings like schools, fields and pavement (explanation 2). Additionally, CLIP-iNat makes a stronger distinction between perched crows in urban settings (explanation 4) and flying crows (explanation 5).

Effects of Alignment. In Fig. 4.9, we visualize the effect of alignment when comparing CLIP to CLIP-iNat on the Maples dataset. We see that alignment can result in a significant change in the spectral clusters detected from the affinity matrix. In particular, one discovered concept is only present in the unaligned comparison. This indicates that both representations contain the information to represent the concept, but their initial configurations differ. After alignment, the concepts discovered are more likely to be fundamental differences between the two representations. Although there are some limitations to this interpretation (see Sec. 4.7), we focus on aligned comparisons in our qualitative plots, as it makes interpretation similar.

ChatGPT-4o Analysis. Analyzing and annotating several explanations for each model is time consuming and cognitively demanding. We explore if ChatGPT-4o [59] is capable of annotating the images for us in the Maples and Corvids comparisons. We use the prompt:

Prompt

I am going to ask you to analyze image grids. You will receive a strip of five image grids. The images in the grid will be from the categories: <category list>. Your task is to concisely describe the consistent features that appear in each image grid. You do not need to use complete sentences. The format of your output should be a dictionary like this. E1: desc1, E2: desc2, E3: desc3, E4: desc4, E5: desc5.

The outputs are provided in the figure captions of Fig. 4.15 and Fig. 4.16. We find that the annotations are clear, reasonable, and helpful.

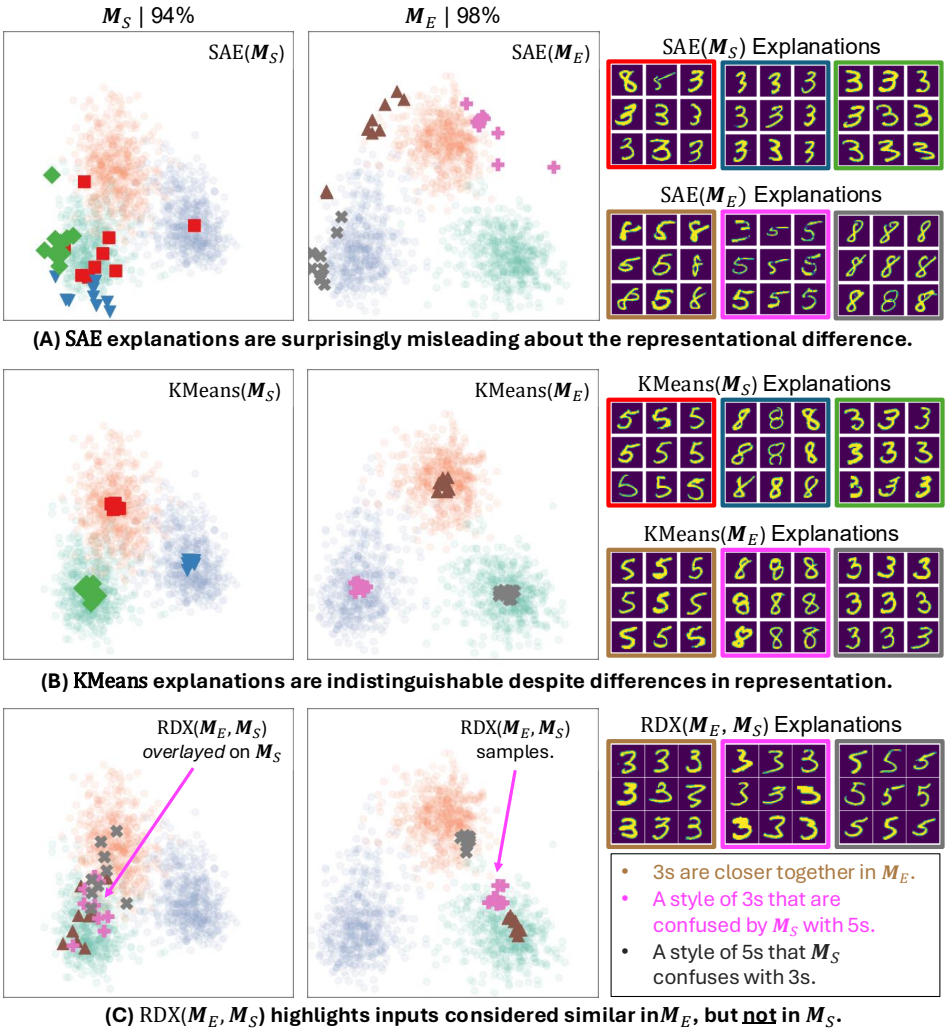
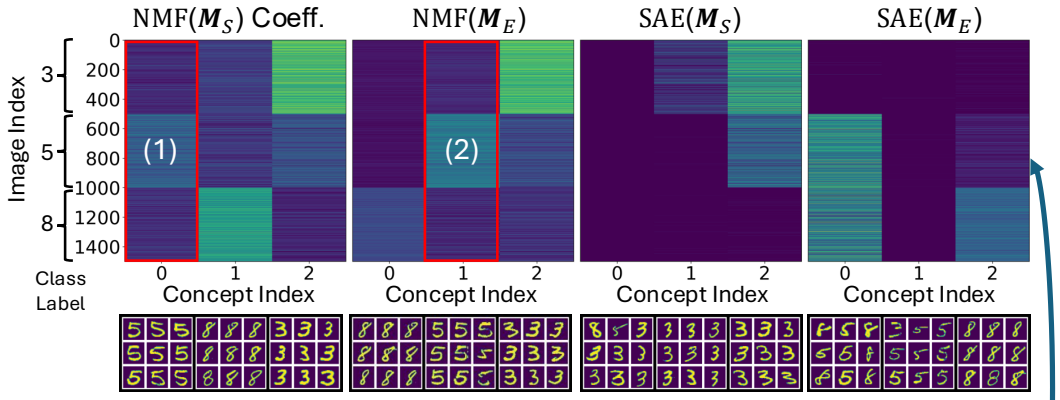
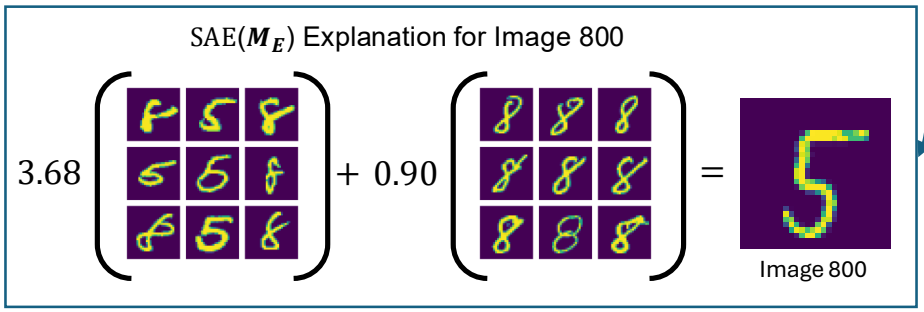


Figure 4.7: Comparing RDX to SAE and KMeans. In Fig. 4.2, we visualized NMF explanations for two model representations, from a 'strong' M_S and an 'expert' model M_E, trained on MNIST-[3,5,8]. Here we show explanations generated using SAE with maximum sampling and KMeans with centroid sampling. **(A)** SAE explanations are confusing and potentially misleading. SAE(M_S) shows mostly 3s in all explanations, whereas SAE(M_E) shows one explanation with mixed 5s and 8s, and two explanations with 5s and 8s respectively. These explanations do not convey which of the two representations is weaker and may even suggest that the M_S is the expert representation. **(B)** KMeans explanations are indistinguishable. Given that these two representations are highly similar, the centroids for the clusters in both representations are nearly the same. **(C)** RDX(M_E, M_S) shows explanations that helps us understand that M_E does a better job of grouping 3s and 5s than M_S, matching the known difference between the two models. The lack of an explanation for 8s suggest that M_S is relatively better at distinguishing 8s. We confirm this in Sec. 4.8.1.



(A) Maximum sampling for NMF and SAE hides complexities of concept vectors.



(B) Weighted combinations of maximally sampled explanations are unintuitive.

Figure 4.8: Interpreting Dictionary Learning Concepts. We identify two issues with dictionary learning methods for XAI that make them challenging to understand. These results are from the same experiment as in Sec. 4.5.2 and Sec. 4.8.1. Maximum sampling to explain concept vectors hides important nuances in model behavior. In (1) and (2), highlighted in red, we can see that $\boldsymbol{M}_S$ and $\boldsymbol{M}_E$ both encode roughly the same concept, with maximum activations for images of 5s (indices 500-1000) and weaker activations for images of 3s and 8s (indices 0-500 and 1000-1500). The generated explanations for both concepts show 5s. However, we can see that the activations for 3s and 8s are relatively lower in (2) than the activations for 3s and 8s in (1). These subtle nuances are critical for understanding how the two models behave differently, but are completely lost in the existing approaches for generating explanations. Thus, existing explanations for dictionary learning concepts are *incomplete*. **(B)** Analyzing a single image through the lens of these concepts is extremely challenging. Users are tasked with using incomplete explanations of a concept in a weighted sum with un-intuitive coefficients. For example, image 800 is an image of a "standard" 5, but is encoded by a weighted combination over a concept of "strange" 5s and 8s and normal 8s. See Sec. 4.8.1 for a discussion on the impact of monosemanticity and polysemanticity in this context.

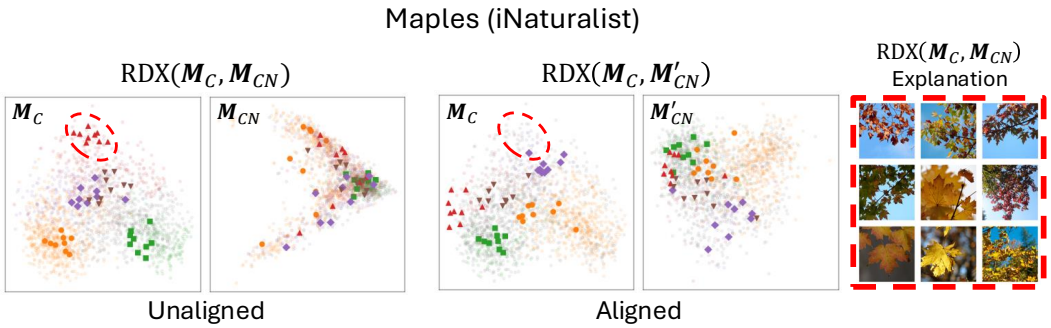


Figure 4.9: Effect of Alignment. (Top) We compare CLIP and CLIP-iNat with and without aligning CLIP-iNat (denoted by M' notation). We visualize PCA plots of the representations with cluster membership (light colors) and samples (dark colors). In each direction, the left plot contains the concept "source" representation while the right plot has the selected clusters overlayed on its representation. **(Unaligned, Left)** We generate five spectral clusters, we can see that they group nicely in the left plot and are spread apart in the right plot. We highlight the region of the red cluster for comparison after alignment. **(Aligned, Right)** After alignment, we can see that it is more difficult to get clusters that have large differences in their distances in the two representations. We find that the region that the red cluster came from in the unaligned comparison is no longer selected after alignment. **Explanation** The red cluster (unaligned) contains maples in fall foliage. Both networks can represent this concept, although the unaligned CLIP-iNat representation does not prioritize it.

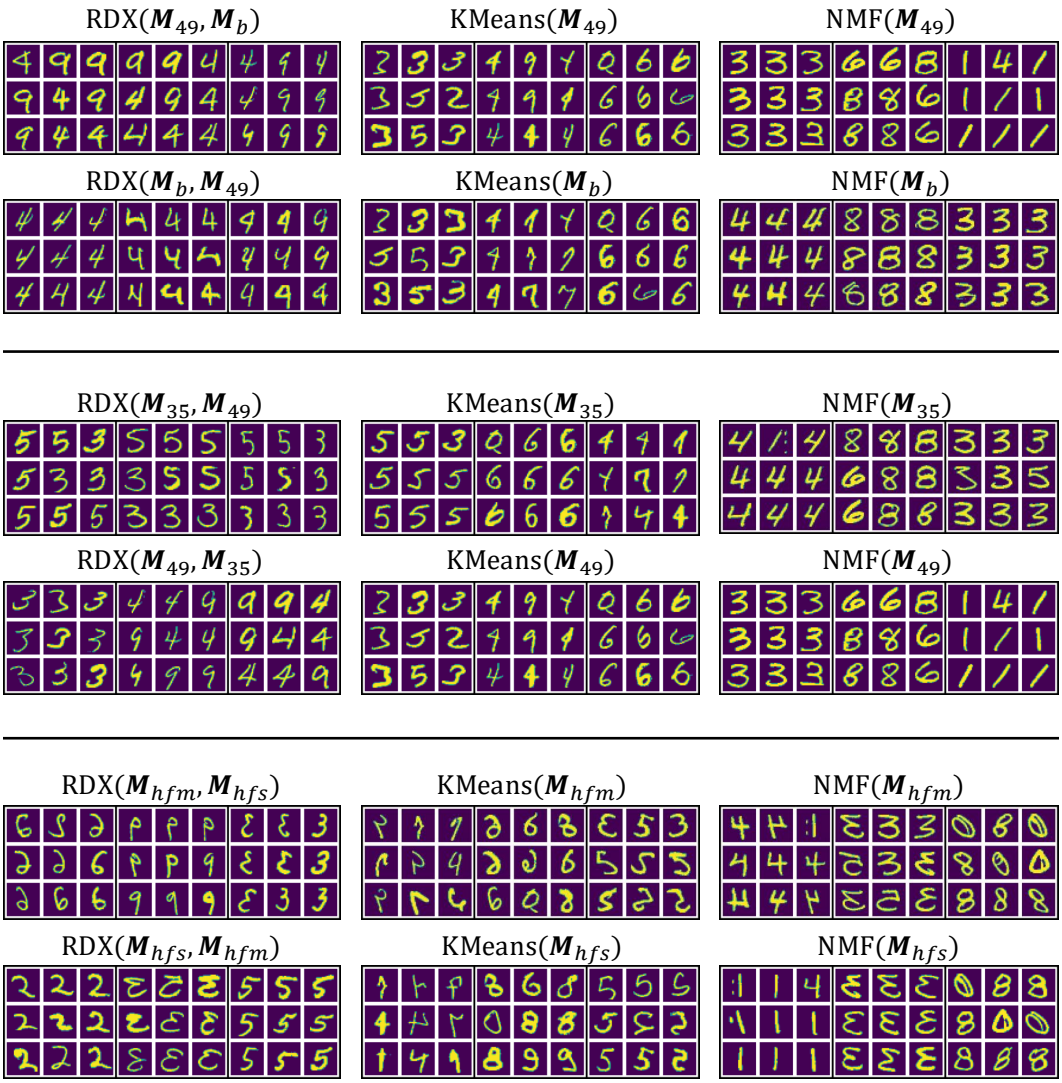


Figure 4.10: Additional explanations for MNIST comparisons. Modifications are described in detail in Tab. 4.4. We compare M_{49} (mixes 4s and 9s) to M_b (no modifications). We observe that `RDX` generates explanations focused on the known difference. `KMeans` and `NMF` have 1/6 explanations related the known difference. **(Middle)** We compare M_{35} (mixes 3s and 5s) to M_{49} (mixes 4s and 9s). `RDX` conveys the modifications made to both models, specifically M_{35} mixes 3s and 5s and that M_{49} mixes 4s and 9s. Also, it shows that M_{49} organizes 3s better than M_{35}. `KMeans` has one explanation related to the known differences, while `NMF` has none. **(Bottom)** We compare M_{hfm} (mixed flipped and unflipped digits) to M_{hfs} (separated flipped and unflipped digits). `RDX` reveals mixing between flipped and unflipped 6s, 9s and 3s in M_{hfm} and no mixing for 2s, flipped 3s and 5s in M_{hfs}. `KMeans` explanations are confusing. `NMF` has 2/6 explanations that align with the known difference.

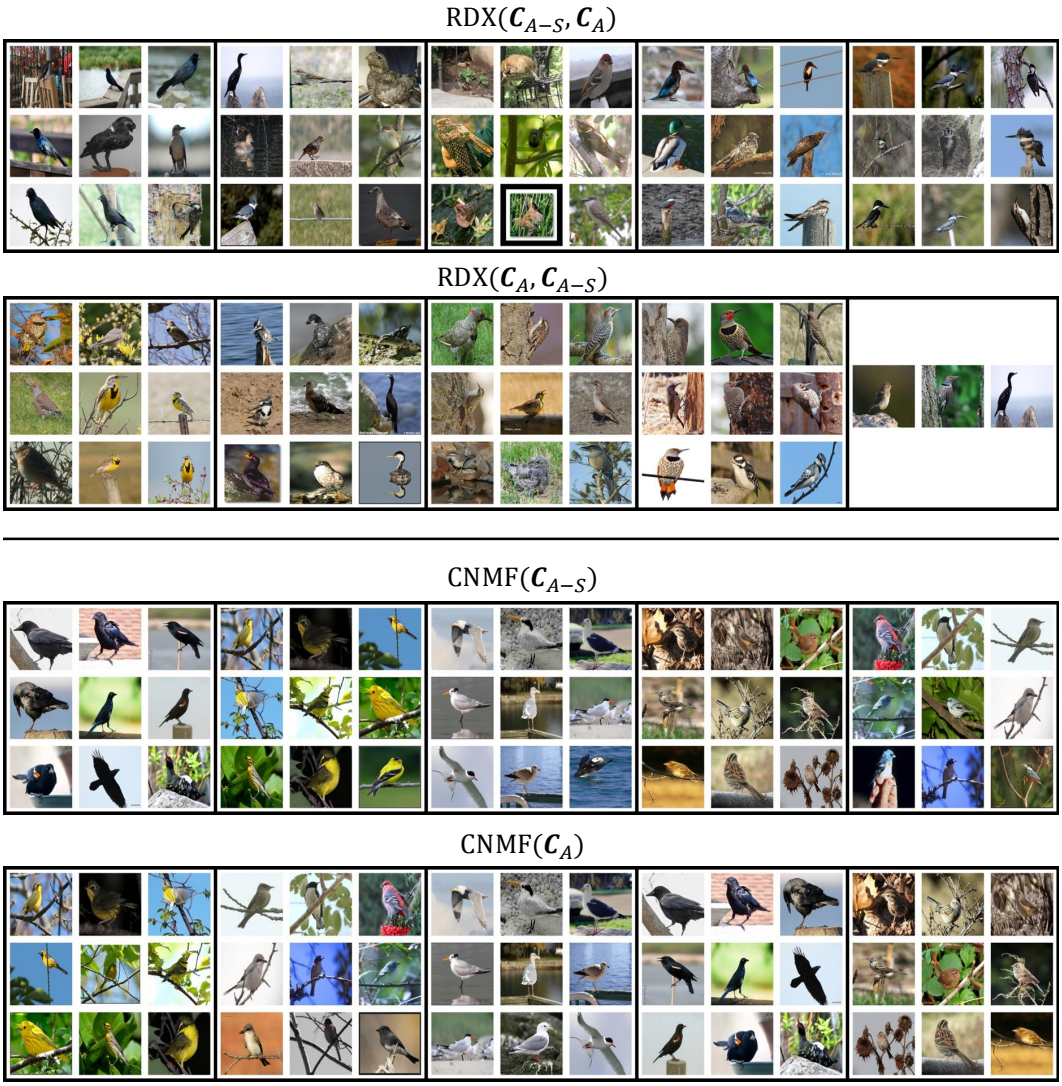


Figure 4.11: Explanations for the "Spotted Wing" concept. We selected a few explanation grids to show in Fig. 4.5. Here we visualize all five explanations generated by two XAI methods, best viewed zoomed in. **(Top)** In row 1, the RDX explanation grids show birds with and without spotted wings mixed together. In row 2, the explanation grids are predominantly made up of birds with spotted wings. In explanation five, we see that RDX can generate clusters with too few images to generate a full grid. **(Bottom)** In both rows, each CNMF explanation grids shows a different kind of bird, unrelated to the known difference. For example, we can see concepts for yellow birds, seabirds, and black birds. The explanations for both models are indistinguishable.

Figure 4.12: Explanation for the "Yellow Back" concept. We visualize explanations from comparing C_{A-YB} vs. C_A. **(Top)** In row 1, the RDX explanation grids show red and yellow birds. Upon closer inspection, one can observe birds that have yellow/red backs mixed with birds with black backs. This is particularly easy to notice in the 4th explanation grid. There are birds with yellow heads and black backs mixed with birds with black heads and yellow backs. In row 2 (explanations 1 - 2) we see that C_A groups birds *without* yellow backs closer together than C_{A-YB}. Explanations 1 - 3 indicate that only C_A groups some types of birds with tan, yellow, and red backs together. **(Bottom)** In both rows, each SAE explanation grids shows concepts that correspond to different bird types, unrelated to the known difference. The explanations for both models are also indistinguishable.

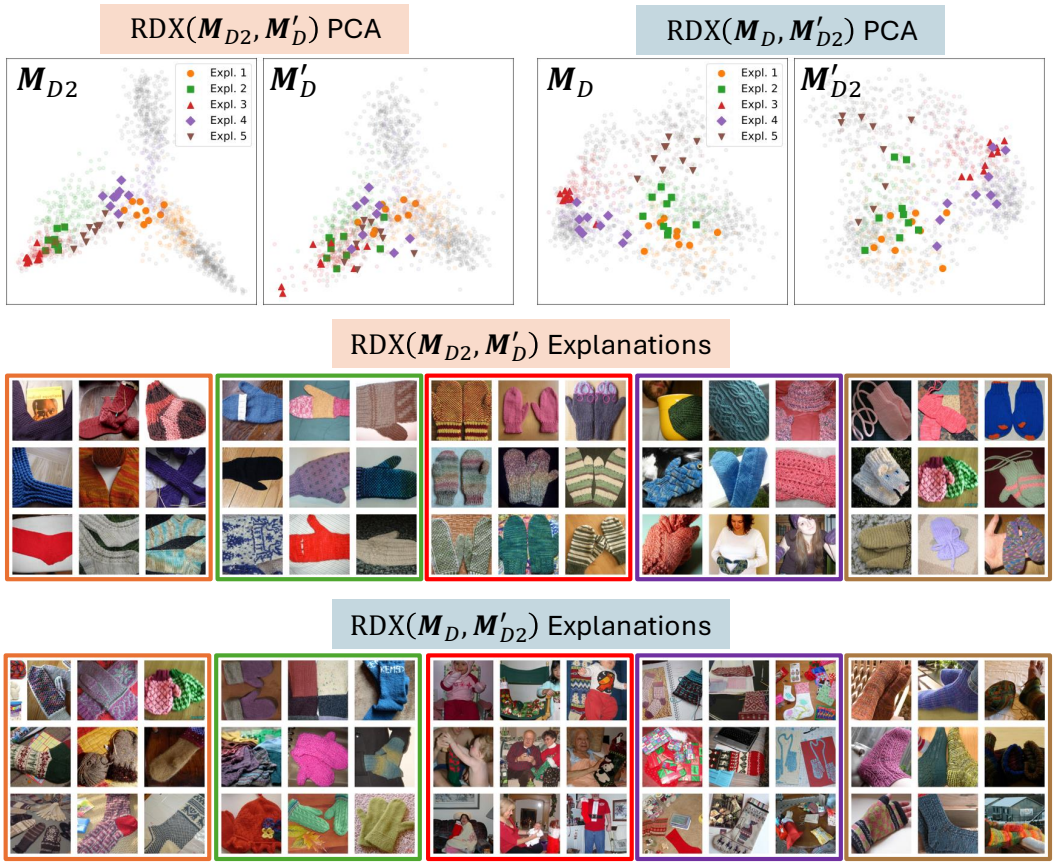


Figure 4.13: Investigating DINOv2 vs. DINO on Mittens (aligned). We visualize RDX difference explanations in both directions on the Mittens dataset (Tab. 4.7). This dataset contains images of mittens, socks and Christmas stocking from ImageNet [50]. For example, RDX(M_{D2}, M'_D) generates explanations for concepts that are in M_{D2} (DINOv2), but not M'_D (aligned DINO). We refer to the explanations as E1 to E5 (left to right). **(Top)** PCA plots of the representations with cluster membership (light colors) and samples (dark colors). In each direction, the left plot contains the concept "source" representation while the right plot has the selected clusters overlayed on its representation. Clusters on the left plot generally appear better organized than in the right plot. (RDX(M_{D2}, M'_D)) E1: crocheted socks, E2: horizontal mittens, E3: vertical pairs of mittens, E4: crocheted mittens, E5: children's mittens. (RDX(M_D, M'_{D2})) E1: multi-colored wool socks, E2: assorted pairs of mittens, E3: children with Christmas decorations, E4: Christmas paraphernalia mittens, and E5: woolen clothes being worn.

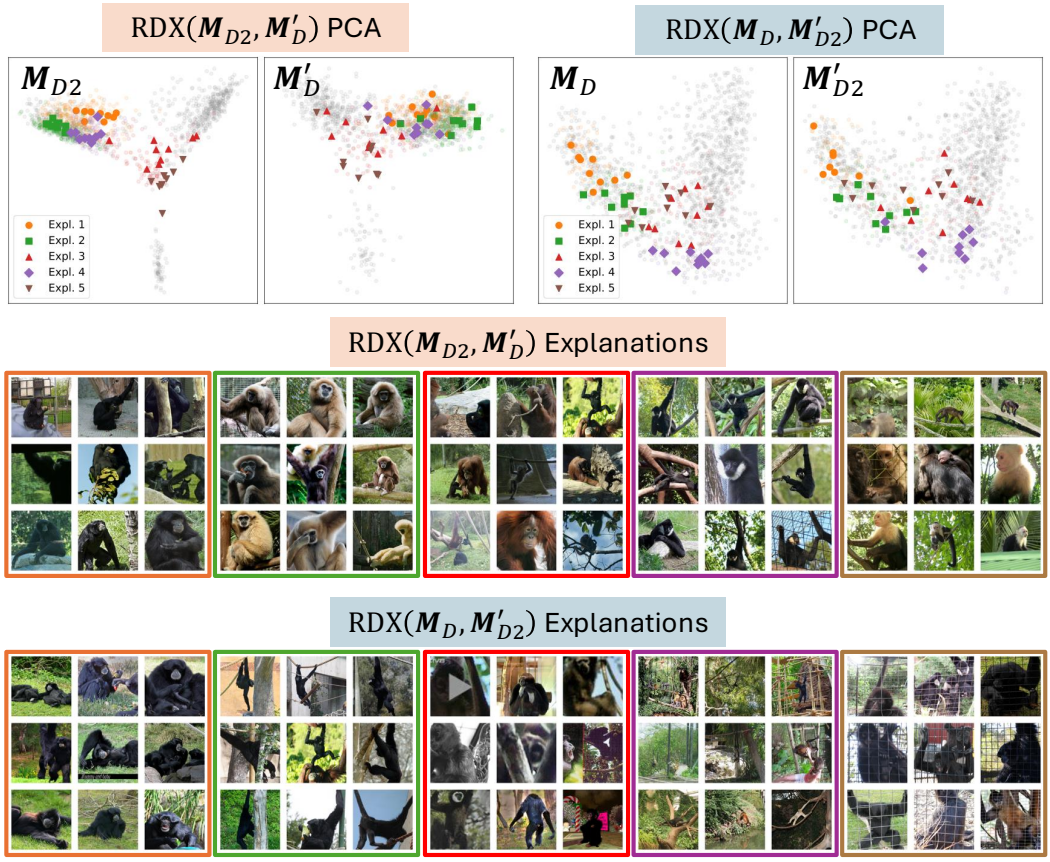


Figure 4.14: Investigating DINOv2 vs. DINO on Primates (aligned). We visualize RDX difference explanations in both directions on the Primates dataset (Tab. 4.7). This dataset contains images of gibbons, siamangs and spider monkeys from ImageNet [50]. For example, RDX(M_{D2}, M'_D) generates explanations for concepts that are in M_{D2} (DINOv2), but not M'_D (aligned DINO). We refer to the explanations as E1 to E5 (left to right). **(Top)** PCA plots of the representations with cluster membership (light colors) and samples (dark colors). In each direction, the left plot contains the concept "source" representation while the right plot has the selected clusters overlayed on its representation. Clusters on the left plot generally appear better organized than in the right plot. (RDX(M_{D2}, M'_D)) E1: black siamangs in diverse environments, E2: tan gibbons, E3: orangutans and gibbons playing, E4: white-cheeked gibbons, and E5: spider monkeys. (RDX(M_D, M'_{D2})) E1: siamangs laying in grass, E2: gibbons swinging, E3: lower resolution images of primates, viewed through glass or from videos, E4: tree environment with distant primates, and E5: assorted primates behind fencing.

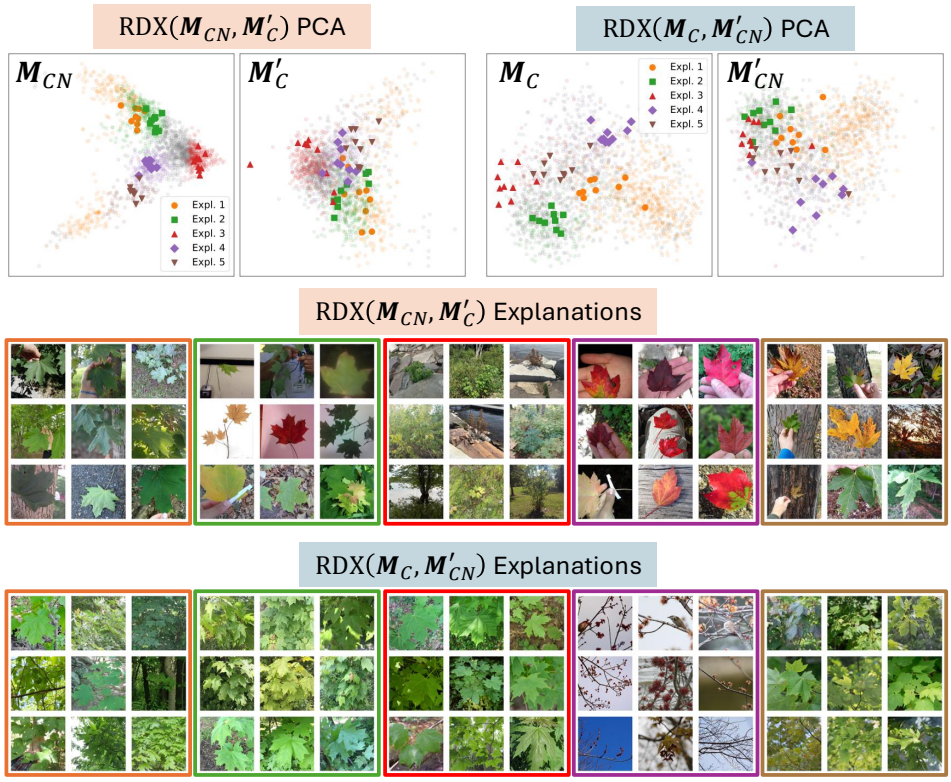


Figure 4.15: Investigating CLIP-iNat vs. CLIP on Maples (aligned). We visualize RDX difference explanations in both directions on the Maples dataset (Tab. 4.7). This dataset contains images of red maples, sugar maples, Norway maples, and silver maples from iNaturalist [49]. RDX(M_{CN}, M'_{CN}) generates explanations for concepts that are in M_{CN} (CLIP-iNat), but not M'_C (aligned CLIP). We refer to the explanations as E1 to E5 (left to right). **(Top)** PCA plots of the representations with cluster membership (light colors) and samples (dark colors). In each direction, the left plot contains the concept "source" representation while the right plot has the selected clusters overlayed on its representation. Clusters on the left plot generally appear better organized than in the right plot. These types of maples have subtle differences beyond the expertise of most people so we use ChatGPT-4o to generate descriptions. (RDX(M_{CN}, M'_C)) E1: "Large, dark green, sharply lobed leaves; smooth surface; some handheld, often against tree bark or forest background," E2: "Varied color (green, red, yellow), symmetric lobes with central point, often single leaves photographed on flat surfaces," E3: "Small clusters of light green to reddish leaves, forest floor or rocky environment, less prominent lobes," E4: "Bright red leaves, often handheld, five lobes with narrow points, smooth margins, clear vein structure," and E5: "Yellow mottled leaves, some black spotting, thick lobes, visible veins, photos taken in autumn light or against tree bark." (RDX(M_C, M'_{CN})) E1: "Leaves with deep sinuses, bright green, flat edges, consistent lighting, often low to ground or with visible bark," E2: "Yellow-green foliage, broad flat leaves with few teeth, tree clusters with hanging leaves, slight curl," E3: "Five-lobed leaves, medium green, fine-toothed edges, spread flat, some variation in lighting and angle," E4: "Red spring buds and samaras, no full leaves visible, bare branches, sky background, some birds," and E5: "Light green leaves with coarsely toothed edges, translucent lighting, some purplish tinge in parts, lobed leaves."

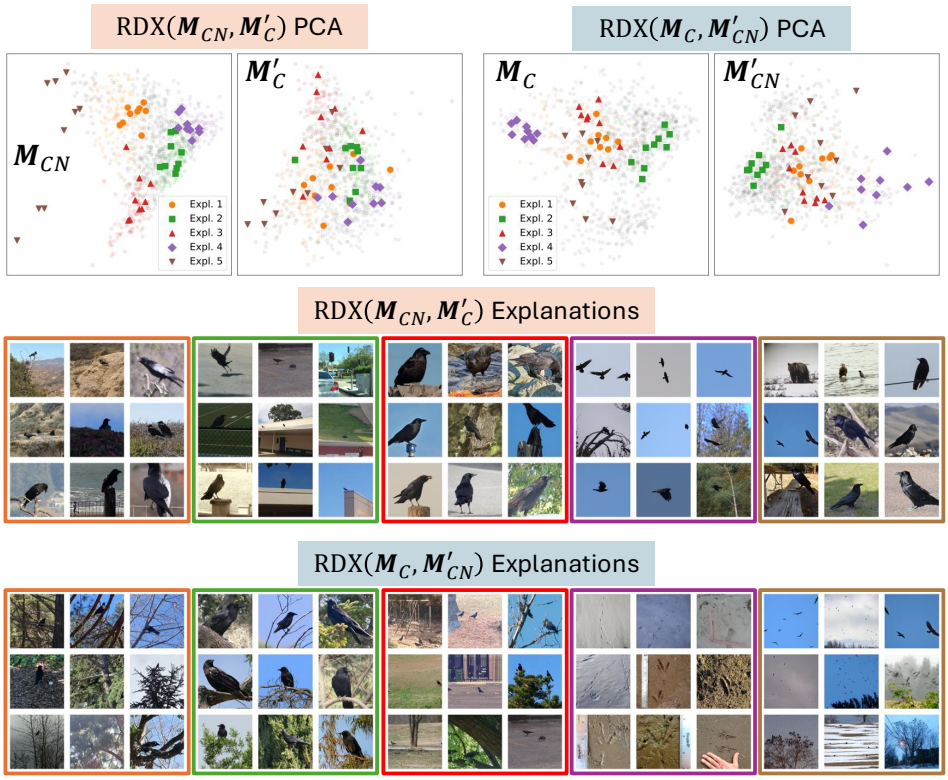


Figure 4.16: Investigating CLIP-iNat vs. CLIP on Corvids (aligned). We visualize RDX difference explanations in both directions on the Corvids dataset (Tab. 4.7). This dataset contains images of crows and ravens from iNaturalist [49]. For example, RDX(M_{CN}, M'_{CN}) generates explanations for concepts that are in M_{CN} (CLIP-iNat), but not M'_C (aligned CLIP). We refer to the explanations as E1 to E5 (left to right). **(Top)** PCA plots of the representations with cluster membership (light colors) and samples (dark colors). In each direction, the left plot contains the concept "source" representation while the right plot has the selected clusters overlayed on its representation. Clusters on the left plot generally appear better organized than in the right plot. These types of corvids have subtle differences beyond the expertise of most people so we use ChatGPT-4o to generate descriptions. (RDX(M_{CN}, M'_C)) E1: 'Birds in arid or rocky environments; perched or flying; often alone or in small groups; slimmer builds; medium size; matte black feathers," E2: 'Urban and suburban settings; birds near buildings, fences, and pavement; typically foraging; in pairs or groups; more compact build," E3: 'Close-up or detailed views of large, shaggy birds; prominent beaks and throat hackles; perched or interacting with environment," and E4: 'Birds flying in sky; high contrast silhouettes; open sky backgrounds; wing shapes and flight patterns emphasized," E5: 'Birds with other wildlife (e.g. bear, eagle); perched alone or with others; prominent size; thick bills and throat feathers." (RDX(M_C, M'_{CN})) E1: "Birds in wooded or forested environments; perched on branches; medium size; matte black feathers; mostly solitary or in pairs," E2: 'Birds on open branches or tall perches; slightly larger size; thick beaks; prominent neck feathers (hackles); more upright posture," E3: 'Birds on ground in urban/park environments; sparse trees; usually in small groups; foraging or walking," E4: 'Footprints in mud, sand, or snow; distinct three-toed tracks; measurement tools in several images; variable substrate," and E5: 'Flocks of birds flying or perched in large groups; sky or treetops visible; misty or open-air environments."

4.9 Additional Methods

4.9.1 K-neighborhood Affinity (KNA) Pseudocode

Let $\boldsymbol{F}_{A,B} \in \mathbb{R}^{n \times n}$ denote the full affinity matrix computed between all image pairs using representations $\boldsymbol{A}$ and $\boldsymbol{B}$. For a given spectral cluster $C \subseteq \{1, \ldots, n\}$, we extract the submatrix $\boldsymbol{F}^{C}_{A,B} \in \mathbb{R}^{r \times r}$, where $r = |C|$, by selecting only the rows and columns of $\boldsymbol{F}_{A,B}$ corresponding to the indices in C. This subset of the affinity matrix $\boldsymbol{F}^{C}_{A,B}$ captures pairwise affinities within the cluster and serves as input to the KNA-based selection procedure.

Algorithm 1 Selecting Image and Neighbors with Maximum KNA

Require: Cluster $C = \{x_1, x_2, \ldots, x_r\}$, Subset of affinity matrix $\boldsymbol{F}_{A,B} \in \mathbb{R}^{r \times r}$, Number of neighbors k

Ensure: Image $x_{\max}$ and its k-nearest neighbors $N_{\max}$

for each image x_i in C **do**
    $N_i \leftarrow$ indices of the k largest values in row $A[i, :]$
    $\mathrm{KNA}(x_i) \leftarrow \sum_{j \in N_i} A[i, j]$
end for
$x_{\max} \leftarrow \arg\max_{x_i} \mathrm{KNA}(x_i)$
$N_{\max} \leftarrow N_{x_{\max}}$
return $x_{\max}, N_{\max}$

4.9.2 Normalized Distance Variants

Both variants compute the pairwise Euclidean distance for each embedding matrix, resulting in $\boldsymbol{D}_A \in R^{n \times n}$ and $\boldsymbol{D}_B \in R^{n \times n}$.

Max-normalized Euclidean Distances. Each distance matrix is divided by the maximum distance in the matrix, such that both $\boldsymbol{D}_A$ and $\boldsymbol{D}_B$ are normalized between 0 and 1. Referred to as $\texttt{RDX}_{MN}$.

Locally Scaled Euclidean Distances. We compute a locally-scaled Euclidean distance that has been shown to have desirable properties for clustering [60]. For each embedding vector $\boldsymbol{a}_i$, this function scales the latent distances between $\boldsymbol{a}_i$ and all other inputs $\boldsymbol{a}_j$ by the distance $\boldsymbol{D}^{ik}_{A}$, where $\boldsymbol{a}_k$ is the 7th neighbor of $\boldsymbol{a}_i$. Referred to as $\texttt{RDX}_{LS}$.

`BSR` Variants. We also use these variants to compute the `BSR` metric. We refer to the variants as $\texttt{BSR}_{MN}$ and $\texttt{BSR}_{LS}$. `BSR` with no subscript uses neighborhood distances.

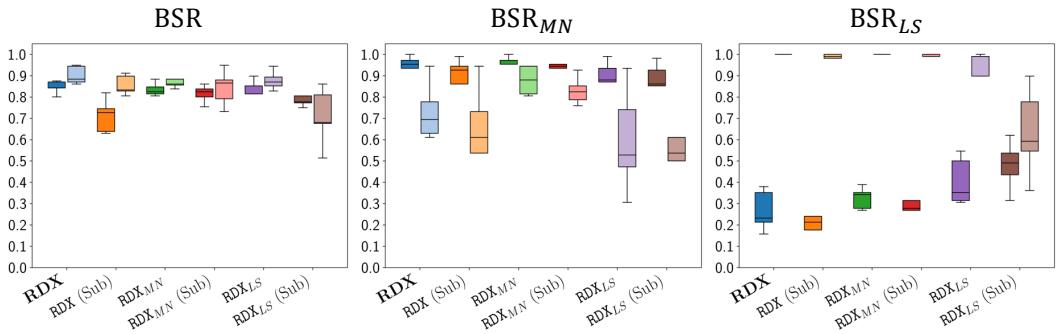


Figure 4.17: `BSR` variants for `RDX` difference function variants. We compute `BSR` variants for all `RDX` distance variants using each difference function. By default we use the locally biased difference function, we denote experiments with the subtraction difference function as `RDX` (Sub). We compute the `BSR` metric on the MNIST experiments. We see, that BSR$_{LS}$ is a poor metric, thus we focus on `BSR` (neighborhood) and BSR$_{MN}$ to assess the different variants. See Sec. 4.9.5 for a longer discussion. Under all distance variants, we can see that `RDX` with the locally biased difference function outperforms subtraction consistently.

4.9.3 Difference Matrix Function Variants

Subtraction. The simplest approach to comparing the normalized distance matrices is subtraction:

$$\boldsymbol{G}_{A,B} = \bar{\boldsymbol{D}}_A - \bar{\boldsymbol{D}}_B. \tag{4.3}$$

If the distance between two inputs is small in $\bar{\boldsymbol{D}}_A$ and large in $\bar{\boldsymbol{D}}_B$, it would result in a large negative value in the difference matrix. If the distances are approximately equal in both matrices, then it would result in a value near zero in the difference matrix. Therefore, images considered similar in only one of the two representations would be identified by large negative values in $\boldsymbol{G}^{ij}_{A,B}$. Unfortunately, subtraction can be sensitive to imperfect normalization and/or large changes in already distant embeddings.

4.9.4 Difference Explanation Sampling Variants

PageRank. We rank nodes in the graph by their PageRank [61]. Let the node with the largest PageRank be i. We select the $|E| - 1$ nodes corresponding to the $|E| - 1$ largest edges with one endpoint at i. We remove these nodes from the pool and iterate until all m sets of explanation grids ($\mathcal{E}$) are sampled.

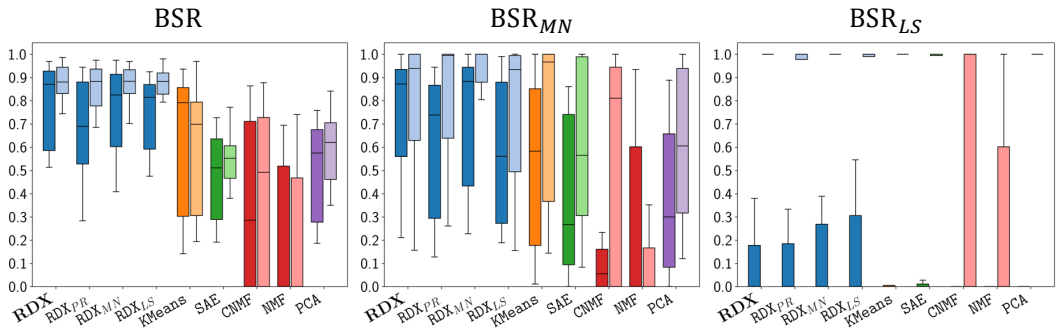


Figure 4.18: BSR variants for RDX variants and baselines. We compute BSR variants for several methods. We evaluate RDX variants with neighborhood distances (RDX), neighborhood distances and PageRank [61] sampling (RDX$_{PR}$), max-normalized distances (RDX$_{MN}$), and locally-scaled distances [60] (RDX$_{LS}$). We compute the BSR metric with all three distance function variants on the MNIST, CUB, and ImageNet/iNaturalist experiments (without alignment). We see that BSR$_{LS}$ is a poor metric as all methods perform perfectly in one of the two comparison directions, suggesting that the scaling technique does not make distances across representations comparable. We focus BSR (neighborhood) and BSR$_{MN}$ to assess the different variants. First, we see broadly that RDX variants outperform all baseline methods. Among them, RDX and RDX$_{MN}$, although RDX$_{MN}$ shows slightly greater variance.

4.9.5 Results

In Fig. 4.17 and Fig. 4.18 we evaluate the different variants for RDX. We find that all RDX variants perform better than baseline methods indicating that using both representations to isolate differences is an effective strategy.

First, when comparing difference functions on known MNIST comparisons (Fig. 4.17), we see a consistent advantage for the locally biased difference function. In all other experiments, we use the locally biased difference function. Second, PageRank [61] sampling is slightly worse than our cluster and sample approach (Fig. 4.18). Third, we notice that BSR$_{LS}$ is a flawed metric. One comparison direction consistently scores near perfectly while the other is quite poor, indicating that distances are not comparable across the two representations. This indicates that local scaling [60], is not appropriate when comparing across representations, although future work may be able to modify it appropriately. Finally, when comparing RDX to RDX$_{MN}$ we notice that they perform reasonably similarly in the metrics. In Tab. 4.3, we show results for a comparison between $\boldsymbol{M}_{35}$ and $\boldsymbol{M}_b$ under BSR and BSR$_{MN}$ We can see that RDX with neighborhood distances performs well under BSR, but worse on BSR$_{MN}$. In contrast, RDX$_{MN}$ performs well on both metrics. We visualize the explanations for

these methods in Fig. 4.19. While both methods have good explanations for $\boldsymbol{M}_{35}$ vs. $\boldsymbol{M}_b$, we can see that in the other direction the two methods differ significantly. RDX with neighborhood distances is much more focused on the known difference than $\texttt{RDX}_{MN}$. This is likely due to the issue described in Sec. 4.4.2. Thus, we use RDX and BSR with neighborhood distances for the main experiments.

Table 4.3: Comparing RDX variants on $\boldsymbol{M}_{35}$ vs. $\boldsymbol{M}_b$ under different BSR variants.

	$\texttt{RDX}(\boldsymbol{M}_{35}, \boldsymbol{M}_b)$	$\texttt{RDX}_{MN}(\boldsymbol{M}_{35}, \boldsymbol{M}_b)$	$\texttt{RDX}(\boldsymbol{M}_b, \boldsymbol{M}_{35})$	$\texttt{RDX}_{MN}(\boldsymbol{M}_b, \boldsymbol{M}_{35})$
BSR	0.80	0.86	0.82	0.88
$\texttt{BSR}_{MN}$	0.95	0.63	0.97	0.81

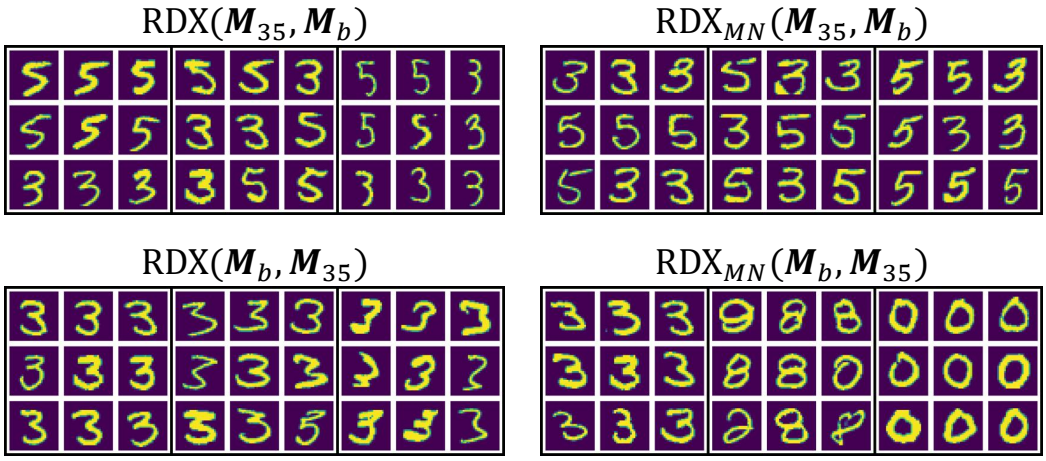


Figure 4.19: Comparing explanations using max-normalized distances vs. neighborhood distances. (Row 1) Both RDX variants generate good difference explanations that capture mixing that is unique to $\boldsymbol{M}_{35}$. (Row 2) RDX with neighborhood distances focuses much more on the known difference with all three explanations showing cleanly grouped 3s. In contrast, $\texttt{RDX}_{MN}$ shows one group of 3s and two other groups unrelated to the known difference.

4.10 Implementation Details

4.10.1 Model Training

Failures of Existing Methods on MNIST-[3,5,8] (Sec. 4.5.2). We train a 2-layer convolutional network with an output dimension of eight on a modified MNIST [62] dataset that only contains the digits 3, 5, and 8 ((MNIST-[3,5,8]). The network is trained for five epochs with a batch size of 128. We use the Adam [63] optimizer with the learning rate set to 1e-2 and a one-cycle learning rate schedule. The global seed is set to 4834586. For the comparison experiment, we select a checkpoint at epoch 1, step 184 with strong performance ($\boldsymbol{M}_S$, 94%) and the final checkpoint at epoch 5 with expert performance ($\boldsymbol{M}_E$, 98%).

Recovering "Known" Differences (Sec. 4.5.3). First, we train a 2-layer convolutional network with an output dimension of 64 on several modified MNIST datasets. See Tab. 4.4 for modification details. The network is trained for five epochs with a batch size of 128. We use the Adam [63] optimizer with the learning rate set to 1e-2 and a one-cycle learning rate schedule. The global seed is set to 4834586 for all models. Models are evaluated on the modified dataset that they were trained on. Second, we train a post-hoc concept bottleneck model (PCBM) [42] on the CUB dataset [44] using the original procedure [42]. The model backbone is a ResNet-18 [43] pre-trained on CUB from pytorchcv [64]. The concept classifier is from scikit-learn [65] and is trained with stochastic gradient descent with the elastic-net penalty. The learning rate is set to 1e-3 and the model is trained for a maximum of 10000 iterations with a batch size of 64. For the comparison experiments, we eliminate a concept by deleting the corresponding concept index from the predicted concept vector. The eliminated concepts are provided in Tab. 4.4. Models are compared on all images in the CUB train set.

Discovering "Unknown" Differences (Sec. 4.5.4). All models in this experiment were downloaded from the timm library [45]. Details are available in Tab. 4.5.

4.10.2 Alignment Training

To align representation $\boldsymbol{A}$ to representation $\boldsymbol{B}$, we learn a transformation matrix $\boldsymbol{M}_{AB} \in \mathbb{R}^{d_A \times d_A}$. We randomly sample 70% of the embeddings in our dataset to train the transformation matrix. The other 30% are used as a validation set. The matrix is trained for 100 steps, with the Adam optimizer [63] with a learning rate of 0.001. We measure the CKA on the validation set and keep the best transformation matrix.

4.10.3 Baselines

For the baseline methods we use the scikit-learn [65] implementations for `PCA`, `NMF`, and `KMeans`. For `CNMF`, we use the pymf [66] implementation. The code for the `SAE` is adapted from [5]. The `SAE` has a linear encoder, a relu activation and a linear decoder. Inputs are z-score normalized. It is trained for 500 epochs with a batch size of 2000 or the maximum number of images. The dimension is set to the number of desired explanations (3 or 5 depending on the experiment). We use a linear learning rate warmup over the first 10 epochs, after which the learning rate is fixed at 0.001. The model is trained with the Adam [63] optimizer. The sparsity coefficient is set to 0.0004. For `PCA`, `NMF`, `CNMF`, and `SAE` we generate explanations by sampling the $|E|$ images with the largest coefficients for each concept vector. For `KMeans`, we sample images closest to the centroid of the cluster.

4.10.4 `RDX` details

We sweep γ on one comparison from each experiment group (see breaks in Tab. 4.7) and select the value that results in the highest performance on `BSR`. We find that a γ of 0.05 or 0.1 works well. We set β to 5 in all experiments.

4.10.5 Comparison Summary

A complete list of comparisons, the data used in the comparison, and the number of images is available in Tab. 4.7. We choose to generate 3 explanations for all MNIST comparisons and 5 explanations for all other comparisons. We choose 3 or 5 because we prefer a small set of explanations for users to analyze. For all experiments, we use 3×3 image grids. In all comparisons where images are from an existing dataset, we use images from the train split because the train split is usually larger. Note that our method is training free and is not impacted by the dataset splits. For the iNaturalist comparisons, 600 research grade images are downloaded from the iNaturalist website [49] with licenses (cc-by,cc-by-nc,cc0). Images are restricted to be a maximum of 500 pixels on the longest side.

4.10.6 Computational Cost

All experiments were conducted using on a machine with an AMD Ryzen 7 3700X 8-Core Processor and a single GeForce RTX 4090 GPU with 128GB of RAM. In Tab. 4.6, we show the time taken for each method on the CUB dataset (5000 images). $\texttt{RDX}_{PR}$ uses PageRank [61] to rank nodes for sampling and is slower. The time for `SAE` varies with the model's output dimension and the number of images. In the table, the ResNet18 has an output dimension of 512.

Table 4.4: Expected effects of "known" differences.

Repr. ID	Modification	Expectation
$\boldsymbol{M}_S$	MNIST dataset only contains 3, 5, and 8. Model checkpoint is from epoch 1, step 184 with 94%	Mistakes on more challenging images. Clusters have slight overlaps.
$\boldsymbol{M}_E$	MNIST dataset only contains 3, 5, and 8. Final model checkpoint with 98%	Clusters have little to no overlap.
$\boldsymbol{M}_b$	None	Baseline model with well-separated clusters.
$\boldsymbol{M}_{35}$	Labels for 5 are replaced with 3.	3s and 5s are mixed together in the representation.
$\boldsymbol{M}_{49}$	Labels for 9 are replaced with 4.	4s and 9s are mixed together in the representation.
$\boldsymbol{M}_{\leftrightarrow}$	Dataset includes horizontally flipped images and uses the original label for flipped images.	Will mix flipped and unflipped digits together.
$\boldsymbol{M}_{\rightleftarrows}$	Dataset includes horizontally flipped images and uses new labels for flipped images.	Horizontally flipped digits are separated into new clusters.
$\boldsymbol{M}_{\updownarrow}$	Dataset includes vertically flipped images and uses the original label for flipped images.	Will mix flipped and unflipped digits together.
$\boldsymbol{M}_{\uparrow\downarrow}$	Dataset includes vertically flipped images and uses new labels for flipped images.	Vertically flipped digits are **separated** into new clusters.
$\boldsymbol{C}_A$	None	Baseline model with organized representational geometry.
$\boldsymbol{C}_{A-S}$	Remove the spotted wing concept.	Representational changes for images of birds with spotted wings.
$\boldsymbol{C}_{A-YB}$	Remove the yellow back concept.	Representational changes for images of birds with yellow backs.
$\boldsymbol{C}_{A-YC}$	Remove the yellow crown concept.	Representational changes for images of birds with yellow crowns.
$\boldsymbol{C}_{A-E}$	Remove the eyebrow on head concept.	Representational changes for images of birds with head eyebrows.
$\boldsymbol{C}_{A-D}$	Remove the duck-like shape concept.	Representational changes for images of birds with duck-like shapes.

Table 4.5: Models and their identifiers from the TIMM library.

Repr. ID	Description	Timm Library ID
$\boldsymbol{M}_D$	DINO	vit_base_patch16_224.dino
$\boldsymbol{M}_{D2}$	DINOv2	vit_base_patch14_reg4_dinov2.lvd142m
$\boldsymbol{M}_C$	CLIP	hf_hub:timm/vit_large_patch14_clip_336.openai
$\boldsymbol{M}_{CN}$	CLIP ft. iNat	hf_hub:timm/vit_large_patch14_clip_336. laion2b_ft_in12k_in1k_inat21

Table 4.6: Runtime (in seconds) for each XAI method.

	RDX	RDX_{PR}	KMeans	CNMF	SAE	PCA	Classifiers
Time (s)	32.71	187.78	9.65	14.95	629.95	8.49	97.3

Table 4.7: Comparison summary table. Experimental settings where we report comparison name, dataset, number of images, concepts, and gamma values. We name the comparisons using one direction, but compare in both directions in all experiments.

Comparison	Comparison Dataset	Num. Ims.	$\lvert\mathcal{E}\rvert$	RDX-γ
$\boldsymbol{M}_S$ vs. $\boldsymbol{M}_E$	MNIST-[3,5,8]	500 x 3	3	0.05
$\boldsymbol{M}_{35}$ vs. $\boldsymbol{M}_b$	MNIST	500 x 10	3	0.1
$\boldsymbol{M}_{49}$ vs. $\boldsymbol{M}_b$	MNIST	500 x 10	3	0.1
$\boldsymbol{M}_{35}$ vs. $\boldsymbol{M}_{49}$	MNIST	500 x 10	3	0.1
$\boldsymbol{M}_{hfm}$ vs. $\boldsymbol{M}_{hfn}$	MNIST w/ hflip	250 x 20	3	0.1
$\boldsymbol{M}_{\text{vfm}}$ vs. $\boldsymbol{M}_{\text{vfs}}$	MNIST w/ vflip	250 x 20	3	0.1
$\boldsymbol{M}_{\uparrow\downarrow}$ vs. $\boldsymbol{M}_{\updownarrow}$	MNIST w/ vflip	250 x 20	3	0.1
$\boldsymbol{C}_{A-S}$ vs. $\boldsymbol{C}_A$	CUB	5000	5	0.1
$\boldsymbol{C}_{A-YB}$ vs. $\boldsymbol{C}_A$	CUB	5000	5	0.1
$\boldsymbol{C}_{A-YC}$ vs. $\boldsymbol{C}_A$	CUB	5000	5	0.1
$\boldsymbol{C}_{A-E}$ vs. $\boldsymbol{C}_A$	CUB	5000	5	0.1
$\boldsymbol{C}_{A-D}$ vs. $\boldsymbol{C}_A$	CUB	5000	5	0.1
$\boldsymbol{M}_D$ vs. $\boldsymbol{M}_{D2}$	Primates-[gibbon, siamang, spider monkey] (ImageNet)	500x3	5	0.05
$\boldsymbol{M}_D$ vs. $\boldsymbol{M}_{D2}$	Clothes-[mitten, Christmas stocking, sock] (ImageNet)	500x3	5	0.05
$\boldsymbol{M}_D$ vs. $\boldsymbol{M}_{D2}$	Buses-[trolley bus, school bus, passenger car] (ImageNet)	500x3	5	0.05
$\boldsymbol{M}_D$ vs. $\boldsymbol{M}_{D2}$	Dogs-[whippet, Saluki, Italian greyhound] (ImageNet)	500x3	5	0.05
$\boldsymbol{M}_C$ vs. $\boldsymbol{M}_{CN}$	Corvids-[Crows, Ravens] (iNaturalist)	500x2	5	0.05
$\boldsymbol{M}_C$ vs. $\boldsymbol{M}_{CN}$	Gators-[American Alligator, American Crocodile] (iNaturalist)	500x2	5	0.05
$\boldsymbol{M}_C$ vs. $\boldsymbol{M}_{CN}$	Maples-[Sugar Maple, Red Maple, Norway Maple, Silver Maple] (iNaturalist)	500x4	5	0.05

References

[1] Neehar Kondapaneni, Oisin Mac Aodha, and Pietro Perona. "Representational Difference Explanations." In: *arXiv preprint arXiv:2505.23917* (2025). URL: https://arxiv.org/abs/2505.23917.

[2] Been Kim et al. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TACV)." In: *ICML*. 2018.

[3] Amirata Ghorbani et al. "Towards automatic concept-based explanations." In: *NeurIPS* (2019).

[4] Thomas Fel et al. "CRAFT: Concept recursive activation factorization for explainability." In: *CVPR*. 2023.

[5] Samuel Stevens et al. "Sparse Autoencoders for Scientifically Rigorous Interpretation of Vision Models." In: *arXiv:2502.06755* (2025).

[6] Harrish Thasarathan et al. "Universal Sparse Autoencoders: Interpretable Cross-Model Concept Alignment." In: *arXiv:2502.03714* (2025).

[7] Hoagy Cunningham et al. "Sparse autoencoders find highly interpretable features in language models." In: *ICLR*. 2024.

[8] Lisa Schut et al. "Bridging the human–AI knowledge gap through concept discovery and transfer in AlphaZero." In: *PNAS* (2025).

[9] Thomas Fel et al. "A holistic approach to unifying automatic concept extraction and concept importance estimation." In: *NeurIPS* (2023).

[10] Julien Colin et al. "What i cannot predict, i do not understand: A human-centered evaluation framework for explainability methods." In: *NeurIPS* (2022).

[11] Leo Gao et al. "Scaling and evaluating sparse autoencoders." In: *ICLR*. 2025.

[12] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks." In: *ICML*. 2017.

[13] Scott M. Lundberg and Su-In Lee. "A unified approach to interpreting model predictions." In: *NeurIPS*. 2017.

[14] Sebastian Bach et al. "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation." In: *PLOS ONE* (2015).

[15] Ramprasaath R Selvaraju et al. "Grad-CAM: visual explanations from deep networks via gradient-based localization." In: *IJCV* (2020).

[16] Ruihan Zhang et al. "Invertible concept-based explanations for cnn models with non-negative concept activation vectors." In: *AAAI*. 2021.

[17] Neehar Kondapaneni, Oisin Mac Aodha, and Pietro Perona. "Representational Similarity via Interpretable Visual Concepts." In: *ICLR*. 2025.

[18] Reduan Achtibat et al. “From attribution maps to human-understandable explanations through concept relevance propagation.” In: *Nature Machine Intelligence* (2023).

[19] Neehar Kondapaneni et al. “Less is more: Discovering concise network explanations.” In: *ICLR 2024 Workshop on Representational Alignment*. 2024. URL: https://openreview.net/forum?id=JBwpD6Yy8Q.

[20] Harold Hotelling. “Relations between two sets of variates.” In: *Biometrika*. 1936.

[21] Simon Kornblith et al. “Similarity of neural network representations revisited.” In: *ICML*. 2019.

[22] Maithra Raghu et al. “Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability.” In: *NeurIPS* (2017).

[23] Yixuan Li et al. “Convergent learning: Do different neural networks learn the same representations?” In: *International Workshop on Feature Extraction: Modern Questions and Challenges at NeurIPS*. 2015.

[24] Minyoung Huh et al. “The platonic representation hypothesis.” In: *ICML*. 2024.

[25] Thao Nguyen, Maithra Raghu, and Simon Kornblith. “Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth.” In: *ICLR*. 2021.

[26] Maithra Raghu et al. “Do vision transformers see like convolutional neural networks?” In: *NeurIPS* (2021).

[27] Zhenda Xie et al. “Revealing the dark secrets of masked image modeling.” In: *CVPR*. 2023.

[28] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. “What is being transferred in transfer learning?” In: *NeurIPS* (2020).

[29] Young-Jin Park et al. “Quantifying Representation Reliability in Self-Supervised Learning Models.” In: *UAI*. 2024.

[30] Wentao Zhang et al. “Efficient diversity-driven ensemble for deep neural networks.” In: *ICDE*. 2020.

[31] Yoonho Lee, Huaxiu Yao, and Chelsea Finn. “Diversify and disambiguate: Out-of-distribution robustness via disagreement.” In: *ICLR*. 2023.

[32] Horst Bunke. “On a relation between graph edit distance and maximum common subgraph.” In: *Pattern recognition letters* (1997).

[33] Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. “Web graph similarity for anomaly detection.” In: *Journal of Internet Services and Applications* (2010).

[34] Edoardo M Airoldi, Xue Bai, and Kathleen M Carley. “Network sampling and classification: An investigation of network model representations.” In: *Decision support systems* (2011).

[35] S Vichy N Vishwanathan et al. “Graph kernels.” In: *JMLR* (2010).

[36] Daniel Archambault. “Structural differences between two graphs through hierarchies.” In: *Graphics Interface*. 2009.

[37] Helen C Purchase, Eve Hoggan, and Carsten Görg. “How important is the “mental map”’?–an empirical investigation of a dynamic graph layout algorithm.” In: *Graph Drawing: 14th International Symposium*. 2007.

[38] Ram Dyuthi Sristi, Gal Mishne, and Ariel Jaffe. “Disc: Differential spectral clustering of features.” In: *NeurIPS* (2022).

[39] Ulrike Von Luxburg. “A tutorial on spectral clustering.” In: *Statistics and computing* (2007).

[40] Jianbo Shi and Jitendra Malik. “Normalized cuts and image segmentation.” In: *TPAMI* (2000).

[41] Aninda Saha, Alina Bialkowski, and Sara Khalifa. “Distilling representational similarity using centered kernel alignment (cka).” In: *BMVC*. 2022.

[42] Mert Yuksekgonul, Maggie Wang, and James Zou. “Post-hoc concept bottleneck models.” In: *ICLR*. 2023.

[43] Kaiming He et al. “Deep residual learning for image recognition.” In: *CVPR*. 2016.

[44] Catherine Wah et al. “The caltech-ucsd birds-200-2011 dataset.” In: (2011).

[45] PyTorch Image Models (timm). https://timm.fast.ai. 2025. (Visited on 05/01/2025).

[46] Mathilde Caron et al. “Emerging properties in self-supervised vision transformers.” In: *ICCV*. 2021.

[47] Maxime Oquab et al. “Dinov2: Learning robust visual features without supervision.” In: *arXiv preprint arXiv:2304.07193* (2023).

[48] Alec Radford et al. “Learning transferable visual models from natural language supervision.” In: *ICML*. 2021.

[49] iNaturalist. https://www.inaturalist.org. 2025. (Visited on 05/01/2025).

[50] Jia Deng et al. “Imagenet: A large-scale hierarchical image database.” In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[51] Andrew Ng et al. “Sparse autoencoder.” In: *CS294A Lecture notes* 2011 (2011).

[52] Daniel Lee and H Sebastian Seung. “Algorithms for non-negative matrix factorization.” In: *NeurIPS* (2000).

[53] Karl Pearson F.R.S. “LIII. On lines and planes of closest fit to systems of points in space.” In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* (1901).

[54] Stuart Lloyd. “Least squares quantization in PCM.” In: *IEEE transactions on information theory* (1982).

[55] Chris HQ Ding, Tao Li, and Michael I Jordan. “Convex and semi-nonnegative matrix factorizations.” In: *PAMI* (2008).

[56] Thomas Fel et al. “Archetypal sae: Adaptive and stable dictionary learning for concept extraction in large vision models.” In: *arXiv:2502.12892* (2025).

[57] Senthooran Rajamanoharan et al. “Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders.” In: *arXiv:2407.14435* (2024).

[58] Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. “Addressing leakage in concept bottleneck models.” In: *NeurIPS* (2022).

[59] ChatGPT-4o. https://openai.com/index/gpt-4o-system-card/. 2025. (Visited on 05/01/2025).

[60] Lihi Zelnik-Manor and Pietro Perona. “Self-tuning spectral clustering.” In: *NeurIPS* (2004).

[61] Lawrence Page et al. *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford infolab, 1999.

[62] Yann LeCun. “The MNIST database of handwritten digits.” In: *http://yann.lecun.com/exdb/mnist/* (1998).

[63] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization.” In: *arXiv:1412.6980* (2014).

[64] Oleg Sémery. *PyTorchCV: Computer Vision Models for PyTorch*. https://pypi.org/project/pytorchcv. 2018.

[65] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python.” In: *JMLR* (2011).

[66] Christian Thurau. *PyMF: Python Matrix Factorization Module*. https://pypi.org/project/PyMF/. Version 0.2. 2011.

Chapter 5

OUTLOOK

Chapters 2, 3, and 4 describe methods that could be used in systems that help us learn from AI. In this section, I outline how estimating student knowledge during learning tasks (knowledge tracing) and comparing representations of data (representational comparison) could be used to actually achieve this goal in practice.

5.1 Expert Data is Abundant

In some domains, expert-level human data is readily available, making it straightforward to train models that approximate human knowledge. For instance, in chess and e-sports, we have access to extensive game logs from high-level players. In these contexts, comparing the representations of superhuman models to those learned from human data offers a way to uncover patterns that human experts may not yet have discovered. Notably, recent work has demonstrated that chess grandmasters can improve their performance by studying strategies learned by AlphaZero [1], revealing insights previously unknown to humans [2].

5.2 Multiple Sources of Supervision

In medicine and biology, multiple diagnostic approaches are often available, each differing in accuracy and cost. For example, skin conditions can be assessed visually or through biopsies, with biopsies being more accurate but significantly more expensive [3, 4]. Recent studies show that vision models can sometimes outperform doctors in diagnosing skin conditions from medical images [5, 6]. This opens up a unique opportunity: by comparing models trained to replicate doctor-level diagnoses against models that achieve superhuman performance, we can identify what additional knowledge or cues the superhuman models have learned, potentially revealing gaps in expert understanding.

5.3 Humans at Multiple Skill Levels

Some platforms feature users with varying levels of expertise, making them ideal testbeds for personalized tutoring systems. Platforms like iNaturalist [7], GeoGuessr [8], and FathomVerse [9] engage users in tasks that rely on visual categorization. In such settings, visual knowledge tracing algorithms can estimate a user's level

of understanding based on their past interactions. By comparing the representations of high-performing users with those of beginners, we can identify missing concepts in beginners and use this insight to guide novice users toward faster improvement.

5.4 Estimating Human Knowledge is Infeasible

In some cases, it may be impractical to explicitly estimate human knowledge. Even then, model comparison remains a powerful tool. By analyzing checkpoints throughout a model's training, we can observe how its internal representations evolve and what concepts emerge over time. Studying this trajectory can reveal the order in which patterns are learned, offering insights into how the model learns and the types of patterns that exist in the dataset.

5.5 General Thoughts about XAI

In Chapter 3, I explored whether existing XAI methods could be adapted for a specific task such as model comparison. For example, I was interested to see if there was a method that could precisely explain why two models disagreed on a particular prediction. In Chapter 4, we discussed some issues with the method proposed in Chapter 3 and developed an improvement. In this process, I realized that many XAI approaches (though not all), had some common issues:

1. **XAI methods are often designed without clear goals in mind.** Many research efforts do not have specific use cases in mind when developing an XAI method, leading to general solutions that struggle when evaluated on specific tasks like comparison.

2. **XAI methods often simplify models too much.** Explanations generated by XAI methods must necessarily be a simplification of model behavior because humans cannot process all aspects of the model's behavior at once. When methods are designed without clear evaluations in mind, they result in general solutions that simplify and discard important aspects of the model's behavior. This is partially a consequence of our evaluation process, since simple, easily understandable explanations are preferred by users. Unfortunately, models are complex and there is no reason to expect that we can understand them via simple explanations. This is especially true, if we are trying to understand patterns discovered by superhuman AI models.

3. **XAI methods are facing a filtering problem.** As models and datasets have become larger, it is possible to generate thousands of different explanations for various aspects of a model's behavior. Of course, this is impossible for human observers to actually process, so we need methods that *filter* explanations to only show us "interesting" material.

5.6 Conclusion

The paths I have described in this section illustrate how knowledge tracing and model comparison can help us identify what AI models know, what humans might be missing, and how to bridge that gap. By leveraging expert data, diverse supervision sources, and user variability, we can begin designing systems that guide human learning in targeted, data-driven ways. Ultimately, this brings us closer to the goal of using AI to teach humans and potentially help us discover new knowledge.

References

[1] David Silver et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.” In: *Science* 362.6419 (2018), pp. 1140–1144. DOI: 10.1126/science.aar6404.

[2] Lisa Schut et al. “Bridging the human–AI knowledge gap through concept discovery and transfer in AlphaZero.” In: *PNAS* (2025).

[3] Elena Losina et al. “Visual screening for malignant melanoma: a cost-effectiveness analysis.” In: *Archives of dermatology* 143.1 (2007), pp. 21–28.

[4] Angela Devine et al. “Costs and cost-effectiveness of cervical cancer screening strategies in women living with HIV in Burkina Faso: The HPV in Africa Research Partnership (HARP) study.” In: *PLoS One* 16.3 (2021), e0248832.

[5] Titus J Brinker et al. “Deep neural networks are superior to dermatologists in melanoma image classification.” In: *European Journal of Cancer* 119 (2019), pp. 11–17.

[6] Holger A Haenssle et al. “Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists.” In: *Annals of oncology* 29.8 (2018), pp. 1836–1842.

[7] iNaturalist community. *iNaturalist: A Community for Naturalists*. https://www.inaturalist.org. Accessed: 2025-06-01. 2023.

[8] GeoGuessr AB. *GeoGuessr: A Geography Game*. https://www.geoguessr.com. Accessed: 2025-06-01. 2023.

[9] Genevieve Patterson et al. “FathomVerse: A community science dataset for ocean animal discovery.” In: *arXiv preprint arXiv:2412.01701* (2024).

Part II

Applications of Aligning Representations

Chapter 6

TEXT-IMAGE ALIGNMENT FOR DIFFUSION-BASED PERCEPTION

Neehar Kondapaneni et al. "Text-image alignment for diffusion-based perception." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 13883–13893. URL: https://openaccess.thecvf.com/content/CVPR2024/html/Kondapaneni_Text-Image_Alignment_for_Diffusion-Based_Perception_CVPR_2024_paper.html.

6.1 Abstract

Diffusion models are generative models with impressive text-to-image synthesis capabilities and have spurred a new wave of creative methods for classical machine learning tasks. However, the best way to harness the perceptual knowledge of these generative models for visual tasks is still an open question. Specifically, it is unclear how to use the prompting interface when applying diffusion backbones to vision tasks. We find that automatically generated captions can improve text-image alignment and significantly enhance a model's cross-attention maps, leading to better perceptual performance. Our approach improves upon the current state-of-the-art (SOTA) in diffusion-based semantic segmentation on ADE20K and the current overall SOTA for depth estimation on NYUv2. Furthermore, our method generalizes to the cross-domain setting. We use model personalization and caption modifications to align our model to the target domain and find improvements over unaligned baselines. Our cross-domain object detection model, trained on Pascal VOC, achieves SOTA results on Watercolor2K. Our cross-domain segmentation method, trained on Cityscapes, achieves SOTA results on Dark Zurich-val and Nighttime Driving. Project page: vision.caltech.edu/TADP/ Code page: github.com/damaggu/TADP

6.2 Introduction

Diffusion models have set the state-of-the-art (SOTA) for image generation [2, 3, 4, 5]. Recently, a few works have shown diffusion pre-trained backbones have a strong prior for scene understanding that allows them to perform well in advanced discriminative vision tasks, such as semantic segmentation [6, 7], monocular depth estimation [7], and keypoint estimation [8, 9]. We refer to these works as diffusion-based perception methods. Unlike contrastive vision language models (e.g., CLIP) [10, 11, 12],

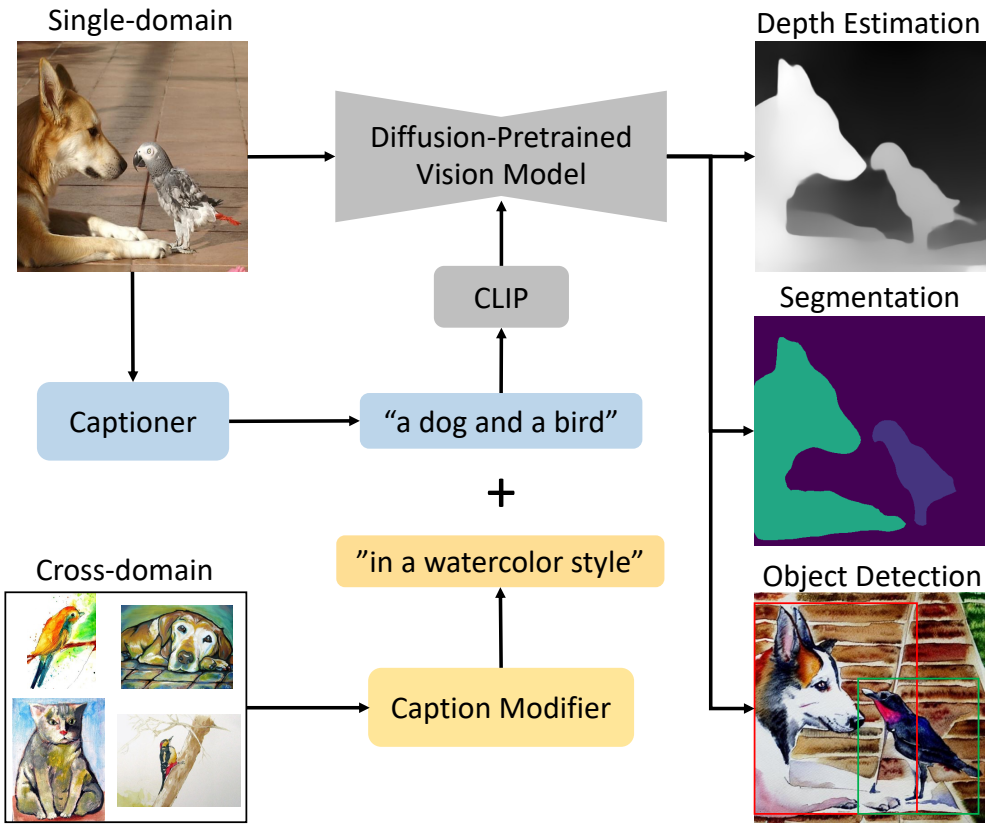


Figure 6.1: Text-Aligned Diffusion Perception (TADP). In TADP, image captions align the text prompts and images passed to diffusion-based vision models. In cross-domain tasks, target domain information is incorporated into the prompt to boost performance.

generative models have a causal relationship with text, in which text guides image generation. In latent diffusion models, text prompts control the denoising U-Net [13], moving the image latent in a semantically meaningful direction [14].

We explore this relationship and find that text-image alignment significantly improves the performance of diffusion-based perception. We then investigate text-target domain alignment in cross-domain vision tasks, finding that aligning to the target domain while training on the source domain can improve a model's target domain performance (Fig. 6.1).

We first study prompting for diffusion-based perceptual models and find that increasing text-image alignment improves semantic segmentation and depth estimation performance. We find that unaligned text prompts can introduce semantic shifts to the feature maps of the diffusion model [14] and that these shifts can make it more

difficult for the task-specific head to solve the target task. Specifically, we ask whether unaligned text prompts, such as averaging class-specific sentence embeddings [7, 10], hinder performance by interfering with feature maps through the cross-attention mechanism. Through ablation experiments on Pascal VOC2012 segmentation [15] and ADE20K [16], we find that off-target and missing class names degrade image segmentation quality. We show automated image captioning [17] achieves sufficient text-image alignment for perception. Our approach (along with latent representation scaling, see Sec. 6.5.1) improves performance for semantic segmentation on Pascal and ADE20k by 4.0 mIoU and 1.7 mIoU, respectively, and depth estimation on NYUv2 [18] by 0.2 RMSE (+8% relative) setting the new SOTA.

Next, we focus on cross-domain adaptation: can appropriate image captioning help visual perception when the model is trained in one domain and tested on a different domain? Training models on the source domain with the appropriate prompting strategy leads to excellent unsupervised cross-domain performance on several benchmarks. We evaluate our cross-domain method on Pascal VOC [15, 19] to Watercolor2k (W2K) and Comic2k (C2K) [20] for object detection and Cityscapes (CS) [21] to Dark Zurich (DZ) [22] and Nighttime (ND) Driving [23] for semantic segmentation. We explore varying degrees of text-target domain alignment and find that improved alignment results in better performance. We also demonstrate using two diffusion personalization methods, Textual Inversion [24] and DreamBooth [25], for better target domain alignment and performance. We find that diffusion pre-training is sufficient to achieve SOTA (+5.8 mIoU on CS→DZ, +4.0 mIoU on CS→ND, +0.7 mIoU on VOC→W2k) or near SOTA results on all cross-domain datasets with no text-target domain alignment, and including our best text-target domain alignment method further improves +1.4 AP on Watercolor2k, +2.1 AP on Comic2k, and +3.3 mIoU on Nighttime Driving. Overall, our contributions are as follows:

- We propose a new method using automated caption generation that significantly improves performance on several diffusion-based vision tasks through increased text-image alignment.
- We systematically study how prompting affects diffusion-based vision performance, elucidating the impact of class presence, grammar in the prompt, and previously used average embeddings.
- We demonstrate that diffusion-based perception effectively generalizes across domains, with text-target domain alignment improving performance, which can be further boosted by model personalization.

6.3 Related Work

6.3.1 Diffusion models for single-domain vision tasks

Diffusion models are trained to reverse a step-wise forward noising process. Once trained, they can generate highly realistic images from pure noise [2, 3, 4, 5]. To control image generation, diffusion models are trained with text prompts/captions that guide the diffusion process. These prompts are passed through a text encoder to generate text embeddings that are incorporated into the reverse diffusion process via cross-attention layers.

Recently, some works have explored using diffusion models for discriminative vision tasks. This can be done by either utilizing the diffusion model as a backbone for the task [6, 7, 8, 9] or through fine-tuning the diffusion model for a specific task and then using it to generate synthetic data for a downstream model [26, 27]. We use the diffusion model as a backbone for downstream vision tasks.

VPD [7] encodes images into latent representations and passes them through one step of the Stable Diffusion model. The cross-attention maps, multi-scale features, and output latent code are concatenated and passed to a task-specific head. Text prompts influence all these maps through the cross-attention mechanism, which guides the reverse diffusion process. The cross-attention maps are incorporated into the multi-scale feature maps and the output latent representation. The text guides the diffusion process and can accordingly shift the latent representation in semantic directions [14, 24, 28, 29]. The details of how VPD uses the prompting interface are described in Sec. 6.4. In short, VPD uses *unaligned* text prompts. In our work, we show how aligning the text to the image by using a captioner can significantly improve semantic segmentation and depth estimation performance.

6.3.2 Image captioning

CLIP [10] introduced a novel learning paradigm to align images with their captions. Shortly after, the LAION-5B dataset [30] was released with 5B image-text pairs; this dataset was used to train Stable Diffusion. We hypothesize that text-image alignment is important for diffusion-pretrained vision models. However, images used in advanced vision tasks (like segmentation and depth estimation) are not naturally paired with text captions. To obtain image-aligned captions, we use BLIP-2 [17], a model that inverts the CLIP latent space to generate captions for novel images.

6.3.3 Diffusion models for cross-domain vision tasks

A few works explore the cross-domain setting with diffusion models [6, 27]. Benigmim et al. [27] use a diffusion model to generate data for a downstream unsupervised domain adaptation (UDA) architecture. In [6], the diffusion backbone is frozen, and the segmentation head is trained with a consistency loss with category and scene prompts guiding the latent code towards target cross-domains. Similar to VPD, the category prompts consist of token embeddings for all classes present in the dataset, irrespective of their presence in any specific image. The consistency loss forces the model to predict the same output mask for all the different scene prompts, helping the segmentation head become invariant to the scene type. Instead of using a consistency loss, we train the diffusion model backbone and task head on the source domain data with and without incorporating the style of the target domain in the caption. We find that better alignment with the target domain (i.e., target domain information included in the prompt) results in better cross-domain performance.

6.3.4 Cross-domain object detection

Cross-domain object detection can be divided into multiple subcategories, depending on what data / labels are at train / test time available. Unsupervised domain adaptation objection detection (UDAOD) tries to improve detection performance by training on unlabeled target domain data with approaches such as self-training [31, 32], adversarial distribution alignment [33] or generating pseudo labels for self-training [34]. Cross-domain weakly supervised object detection (CDWSOD) assumes the availability of image-level annotations at training time and utilizes pseudo labeling [20, 35], alignment [36] or correspondence mining [37]. Recently, [38] used CLIP [10] for Single Domain Generalization, which aims to generalize from a single domain to multiple unseen target domains. Our text-based method defines a new category of cross-domain object detection that tries to adapt from a single source to an unseen target domain by only having the broad semantic context of the target domain (e.g., foggy/night/comic/watercolor) as text input to our method. When we incorporate model personalization, our method can be considered a UDAOD method since we train a token based on unlabeled images from the target domain.

6.4 Methods

Stable Diffusion [5]. The text-to-image Stable Diffusion model is composed of four networks: an encoder $\mathcal{E}$, a conditional denoising autoencoder (a U-Net in Stable Diffusion) ϵ_θ, a language encoder τ_θ (the CLIP text encoder in Stable Diffusion), and

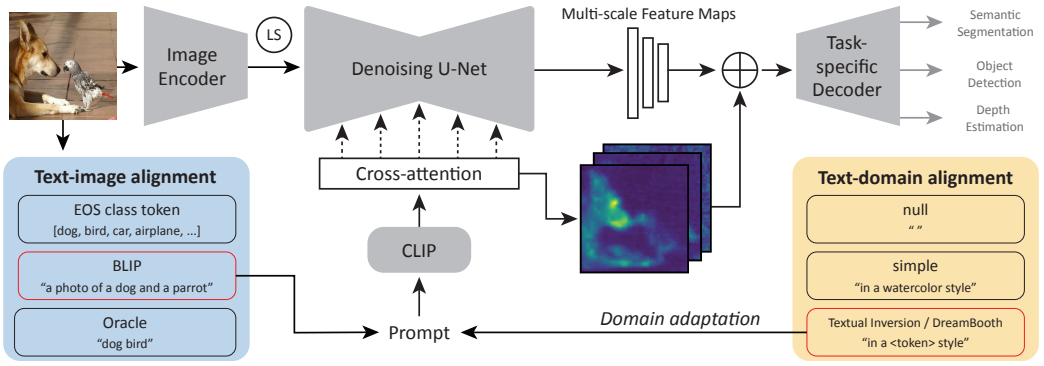


Figure 6.2: Overview of TADP. We test several prompting strategies and evaluate their impact on downstream vision task performance. Our method concatenates the cross-attention and multi-scale feature maps before passing them to the vision-specific decoder. In the blue box, we show three single-domain captioning strategies with differing levels of text-image alignment. We propose using BLIP [17] captioning to improve image-text alignment. We extend our analysis to the cross-domain setting (yellow box), exploring whether aligning the source domain text captions to the target domain may impact model performance by appending caption modifiers to image captions generated in the source domain and find model personalization modifiers (Textual Inversion/Dreambooth) work best.

a decoder $\mathcal{D}$. $\mathcal{E}$ and $\mathcal{D}$ are trained before ϵ_θ, such that $\mathcal{D}(\mathcal{E}(x)) = \tilde{x} \approx x$. Training ϵ_θ is composed of a pre-defined forward process and a learned reverse process. The reverse process is learned using LAION-400M [39], a dataset of 400 million images ($x \in X$) and captions ($y \in Y$). In the forward process, an image x is encoded into a latent $z_0 = \mathcal{E}(x)$, and t steps of a forward noise process are executed to generate a noised latent z_t. Then, to learn the reverse process, the latent z_t is passed to the denoising autoencoder ϵ_θ, along with the time-step t and the image caption's representation $\mathcal{C} = \tau_\theta(y)$. τ_θ adds information about y to ϵ_θ using a cross-attention mechanism, in which the query is derived from the image, and the key and value are transformations of the caption representation. The model ϵ_θ is trained to predict the noise added to the latent in step t of the forward process:

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2\right], \tag{6.1}$$

where $t \in \{0, ..., T\}$. During generation, a pure noise latent z_T and a user-specified prompt are passed through the denoising autoencoder ϵ_θ for T steps and decoded $\mathcal{D}(z_0)$ to generate an image guided by the text prompt.

Diffusion for Feature Extraction. Diffusion backbones have been used for downstream vision tasks in several recent works [6, 7, 8, 9]. Due to its public availability and performance in perception tasks, we use a modified version (see Sec. 6.5.1) of

the feature extraction method in VPD. An image latent $z_0 = \mathcal{E}(x)$ and a conditioning $\mathcal{C}$ are passed through the last step of the denoising process $\epsilon_\theta(z_0, 0, \mathcal{C})$. The cross-attention maps A and the multi-scale feature maps F of the U-Net are concatenated $V = A \oplus F$ and passed to a task-specific head H to generate a prediction $\hat{p} = H(V)$. The backbone ϵ_θ and head H are trained with a task-specific loss $\mathcal{L}_H(\hat{p}, p)$.

Average EOS Tokens. To generate $\mathcal{C}$, previous methods [6, 7] rely on a method from CLIP [10] to use averaged text embeddings as representations for the classes in a dataset. A list of 80 sentence templates for each class of interest (such as "a <adjective> photo of a <class name>") are passed through the CLIP text encoder. We use $\mathcal{B}$ to denote the set of class names in a dataset. For a specific class ($b \in \mathcal{B}$), the CLIP text encoder returns an $80 \times N \times D$ tensor, where N is the maximum number of tokens over all the templates, and D is 768 (the dimension of each token embedding). Shorter sentences are padded with EOS tokens to fill out the maximum number of tokens. The first EOS token from each sentence template is averaged and used as the representative embedding for the class such that $\mathcal{C} \in \mathcal{R}^{|\mathcal{B}| \times 768}$. This method is used in [6, 7], we denote it as $\mathcal{C}_{avg}$ and use it as a baseline. For semantic segmentation, all of the class embeddings, irrespective of presence in the image, are passed to the cross-attention layers. Only the class embedding of the room type is passed to the cross-attention layers for depth estimation.

6.4.1 Text-Aligned Diffusion Perception (TADP)

Our work proposes a novel method for prompting diffusion-pretrained perception models. Specifically, we explore different prompting methods $\mathcal{G}$ to generate $\mathcal{C}$. In the single-domain setting, we show the effectiveness of a method that uses BLIP-2 [17], an image captioning algorithm, to generate a caption as the conditioning for the model: $\mathcal{G}(x) = \tilde{y} \rightarrow \mathcal{C}$. We then extend our method to the cross-domain setting by incorporating target domain information to $\mathcal{C} = \mathcal{C} + \mathcal{M}(\mathcal{P})_s$, where $\mathcal{M}$ is a caption modifier that takes target domain information $\mathcal{P}$ as input and outputs a caption modification $\mathcal{M}(\mathcal{P})_s$ and a model modification $\mathcal{M}(\mathcal{P})_{\epsilon_\theta}$. In Sec. 6.5, we analyze the text-image interface of the diffusion model by varying the captioner $\mathcal{G}$ and caption modifier $\mathcal{M}$ in a systematic manner for three different vision tasks: semantic segmentation, object detection, and monocular depth estimation. Our method and experiments are presented in Fig. 6.2. Following [7], we train our ADE20k segmentation and NYUv2 depth estimation models with fast and regular schedules. On ADE20k, we train using 4k steps (fast), 8k steps (fast), and 80k steps (normal). For NYUv2 depth, we train on a 1-epoch (fast) schedule and a 25-epoch (normal) schedule. For implementation details, refer to Appendix 6.10.

Method	Avg	TA	LS	G	OT	mIoU^{ss}
VPD(R) [7]	✓	✓			✓	82.34
VPD(LS)	✓	✓	✓		✓	83.06
Class Embs			✓		✓	82.72
Class Names			✓		✓	84.08
TADP-0			✓	✓		86.36
TADP-20			✓	✓		86.19
TADP-40			✓	✓		**87.11**
TADP(NO)-20			✓			86.35
TADP-Oracle			✓			89.85

Table 6.1: Prompting for Pascal VOC2012 Segmentation. We report the single-scale validation mIoU for Pascal experiments. (R): Reproduction of VPD, Avg: EOS token averaging, LS: Latent Scaling, G: Grammar, OT: Off-target information. For our method, we indicate the minimum length of the BLIP caption with TADP-*X* and nouns only with (NO).

6.5 Results

6.5.1 Latent scaling

Before exploring image-text alignment, we apply latent scaling to encoded images (Appendix G of Rombach et al. [5]). This normalizes the image latents to have a standard normal distribution. The scaling factor is fixed at 0.18215. We find that latent scaling improves performance using $\mathcal{C}_{avg}$ for segmentation and depth estimation (Fig. 6.3). Specifically, latent scaling improves ~0.8% mIoU on Pascal, ~0.3% mIoU on ADE20K, and a relative ~5.5% RMSE on NYUv2 Depth (Fig. 6.3).

6.5.2 Single-domain alignment

Average EOS Tokens. We scrutinize the use of average EOS tokens for $\mathcal{C}$ (see Sec. 6.4). While average EOS tokens are sensible when measuring cosine similarities in the CLIP latent space, it is unsuitable in diffusion models, where the text guides the diffusion process through cross-attention. In our qualitative analysis, we find that average EOS tokens degrade the cross-attention maps (Fig. 6.4). Instead, we explore using CLIP to embed each class name independently and use the tokens corresponding to the actual word (not the EOS token) and pass this as input to the cross-attention layer:

$$\mathcal{G}_{\text{ClassEmbs}}(\mathcal{B}) = concat(CLIP(b)|b \in \mathcal{B}) \rightarrow \mathcal{C}_{\text{ClassEmbs}}. \tag{6.2}$$

Second, we explore a generic prompt, a string of class names separated by spaces:

$$\mathcal{G}_{\text{ClassNames}}(\mathcal{B}) = \{\text{`` '' } + b|b \in \mathcal{B}\} \rightarrow \mathcal{C}_{\text{ClassNames}}. \tag{6.3}$$

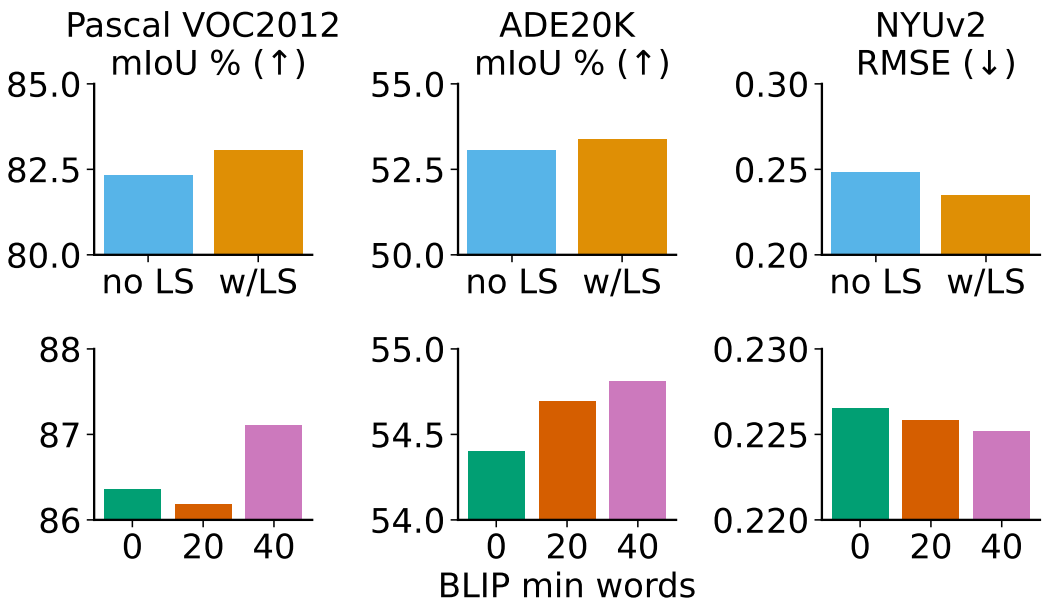


Figure 6.3: Effects of latent scaling (LS) and BLIP caption minimum length. We report mIoU for Pascal, mIoU for ADE20K, and RMSE for NYUv2 depth (right). (Top) Latent scaling improves performance on Pascal ~0.8 mIoU (higher is better), ~0.3 mIoU, and ~5.5% relative RMSE (lower is better). (Bottom) We see a similar effect for BLIP minimum token length, with longer captions performing better, improving ~0.8 mIoU on Pascal, ~0.9 mIoU on ADE20K, and ~0.6% relative RMSE.

These prompts are similar to the ones used for averaged EOS tokens $\mathcal{C}_{avg}$ w.r.t. overall text-image alignment but instead use the token corresponding to the word representing the class name. We evaluate these variations on Pascal VOC2012 segmentation. We find that $\mathcal{C}_{\text{ClassNames}}$ improves performance by 1.0 mIoU, but $\mathcal{C}_{\text{ClassEmbs}}$ reduces performance by 0.3 mIoU (see Tab. 6.1). We perform more in-depth analyses of the effect of text-image alignment on the diffusion model's cross-attention maps and image generation properties in Appendix 6.7.

TADP. To align the diffusion model text input to the image, we use BLIP-2 [17] to generate captions for every image in our single-domain datasets (Pascal, ADE20K, and NYUv2).

$$\mathcal{G}_{\text{TADP}}(x) = \textit{BLIP-2}(x) \rightarrow \mathcal{C}_{\text{TADP}}(x) \tag{6.4}$$

BLIP-2 is trained to produce image-aligned text captions and is designed around the CLIP latent space. However, other vision-language algorithms that produce captions could also be used. We find that these text captions improve performance in all datasets and tasks (Tabs. 6.1, 6.2, 6.3). Performance improves on Pascal

Method	#Params	FLOPs	Crop	mIoU^{ss}	mIoU^{ms}
self-supervised pre-training					
EVA [40]	1.01B	-	896^2	61.2	61.5
InternImage-L [41]	256M	2526G	640^2	53.9	54.1
InternImage-H [41]	1.31B	4635G	896^2	**62.5**	**62.9**
multi-modal pre-training					
CLIP-ViT-B [42]	105M	1043G	640^2	50.6	51.3
ViT-Adapter [43]	571M	-	896^2	61.2	61.5
BEiT-3 [44]	1.01B	-	896^2	**62.0**	62.8
ONE-PEACE [45]	1.52B	-	896^2	**62.0**	**63.0**
diffusion-based pre-training					
VPD_{A32} [7]	862M	891G	512^2	53.7	54.6
VPD(R)	862M	891G	512^2	53.1	54.2
VPD(LS)	862M	891G	512^2	53.7	54.4
TADP-40 (Ours)	862M	2168G	512^2	**54.8**	**55.9**
TADP-Oracle	862M	-	512^2	72.0	-

Table 6.2: Semantic segmentation with different methods for ADE20k. Our method (green) achieves SOTA within the diffusion-pretrained models category. The results of our oracle indicate the potential of diffusion-based models for future research as it is significantly higher than the overall SOTA (highlighted in yellow). See Tab. 6.1 for a notation key and Tab. 6.6 for fast schedule results.

segmentation by ~4% mIoU, ADE20K by ~1.4% mIoU, and NYUv2 Depth by a relative RMSE improvement of 4%. We see stronger effects on the fast schedules for ADE20K with an improvement of ~5 mIoU at (4k), ~2.4 mIoU (8K). On NYUv2 Depth, we see a smaller gain on the fast schedule ~2.4%. All numbers are reported relative to VPD with latent scaling.

We perform some ablations to analyze what aspects of the captions are important. We explore the minimum token number hyperparameter for BLIP-2 to explore if longer captions can produce more useful feature maps for the downstream task. We try a minimum token number of 0, 20, and 40 tokens (denoted as $\mathcal{C}_{\text{TADP-N}}$) and find small but consistent gains with longer captions, resulting on average 0.75% relative gain for 40 tokens vs. 0 tokens (Fig. 6.3). Next, we ablate the Pascal $\mathcal{C}_{\text{TADP-20}}$ captions to understand what in the caption is necessary for the performance gains we observe. We use NLTK [49] to filter for the nouns in the captions. In the $\mathcal{C}_{\text{TADP(NO)-20}}$

Method	RMSE↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL ↓	log10 ↓
default schedule						
SwinV2-L [46]	0.287	0.949	0.994	0.999	0.083	0.035
AiT [47]	0.275	0.954	0.994	0.999	0.076	0.033
ZoeDepth [48]	0.270	0.955	0.995	0.999	0.075	0.032
VPD [7]	0.254	0.964	0.995	0.999	0.069	0.030
VPD(R)	0.248	0.965	0.995	0.999	0.068	0.029
VPD(LS)	0.235	0.971	0.996	0.999	0.064	0.028
TADP-40	**0.225**	**0.976**	**0.997**	**0.999**	**0.062**	**0.027**
fast schedule, 1 epoch						
VPD	0.349	0.909	0.989	0.998	0.098	0.043
VPD(R)	0.340	0.910	0.987	0.997	0.100	0.042
VPD(LS)	0.332	0.926	0.992	0.998	0.097	0.041
TADP-0	**0.328**	**0.935**	**0.993**	**0.999**	**0.082**	**0.038**

Table 6.3: Depth estimation in NYUv2. We find latent scaling accounts for a relative gain of ∼ 5.5% on the RMSE metric. Additionally, image-text alignment improves ∼ 4% relative on the RMSE metric. A minimum caption length of 40 tokens performs the best.We also explore adding a text-adapter (TA) to TADP, but find no significant gain. See Table 6.1 for a notation key.

nouns-only caption setting, we achieve 86.4% mIoU, similar to 86.2% mIoU with $\mathcal{C}_{\text{TADP-20}}$ (Tab. 6.1), suggesting nouns are sufficient.

Oracle. This insight about nouns leads us to ask if an oracle caption, in which all the object class names in an image are provided as a caption, can improve performance further. We define $\mathcal{B}(x)$ as the set of class names present in image x.

$$\mathcal{G}_{\text{Oracle}}(x) = \{\text{`` '' } + b | b \in \mathcal{B}(x)\} \rightarrow \mathcal{C}_{\text{Oracle}}(x) \tag{6.5}$$

While this is not a realistic setting, it serves as an approximate upper bound on performance for our method on the segmentation task. We find a large improvement in performance in segmentation, achieving 89% mIoU on Pascal and 72.2% mIoU on ADE20K. For depth estimation, multi-class segmentation masks are only provided for a smaller subset of the images, so we cannot generate a comparable oracle. We perform ablations on the oracle captions to evaluate the model's sensitivity to alignment. For ADE20K, on the 4k iteration schedule, we modify the oracle captions by randomly adding and removing classes such that the recall and precision are

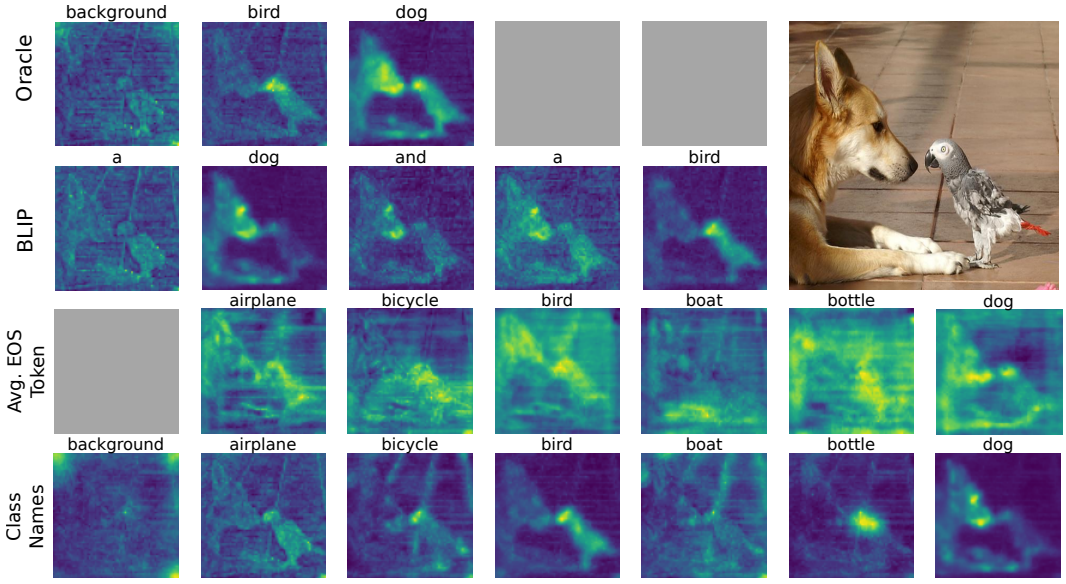


Figure 6.4: Cross-attention maps for different types of prompting (before training). We compare the cross-attention maps for four types of prompting: oracle, BLIP, Average EOS tokens, and class names as space-separated strings. The cross-attention maps for different heads at all different scales are upsampled to 64x64 and averaged. When comparing Average Template EOS and Class Names, we see (qualitatively) averaging degrades the quality of the cross-attention maps. Furthermore, we find that class names that are not present in the image can have highly localized attention maps (e.g., 'bottle'). Further analysis of the cross-attention maps is available in Sec. 6.7, where we explore image-to-image generation, copy-paste image modifications, and more.

at 0.5, 0.75, and 1.0 (independently) (Tab. 6.7). We find that both precision and recall have an effect, but recall is significantly more important. When recall is lower (0.50), improving precision has minimal impact (<1% mIoU). However, precision has progressively larger impacts as recall increases to 0.75 and 1.00 (~3% mIoU and ~7% mIoU). In contrast, recall has large impacts at every precision level: 0.5 - (~6% mIoU), 0.75 - (~9% mIoU), and 1.00 - (~13% mIoU). BLIP-2 captioning performs similarly to a precision of 1.00 and a recall of 0.5 (Tab. 6.2). Additional analyses w.r.t. precision, recall, and object sizes can be found in Appendix 6.8.

6.5.3 Cross-domain alignment

Next, we ask if text-image alignment can benefit cross-domain tasks. In cross-domain, we train a model on a source domain and test it on a different target domain. There are two aspects of alignment in the cross-domain setting: the first is also present in single-domain, which is image-text alignment; the second is unique to the cross-

domain setting, which is text-target domain alignment. The second is challenging because there is a large domain shift between the source and target domain. Our intuition is that while the model has no information on the target domain from the training images, an appropriate text prompt may carry some general information about the target domain. Our cross-domain experiments focus on the text-target domain alignment and use $\mathcal{G}_{\text{TADP}}$ for image-text alignment (following our insights from the single-domain setting).

Training. Our experiments in this setting are designed in the following manner: we train a diffusion model on the source domain captions $\mathcal{C}_{\text{TADP}}(x)$. With these source domain captions, we experiment with four different caption modifications (each increasing in alignment to the target domain), a null $\mathcal{M}_{null}(\mathcal{P})$ caption modification where $\mathcal{M}_{null}(\mathcal{P})_s = \varnothing = \mathcal{M}_{null}(\mathcal{P})_{\epsilon_\theta} = \varnothing$, a simple $\mathcal{M}_{simple}(\mathcal{P})$ caption modifier where $\mathcal{M}_{simple}(\mathcal{P})_s$ is a hand-crafted string describing the style of the target domain appended to the end and $\mathcal{M}_{simple}(\mathcal{P})_{\epsilon_\theta} = \varnothing$, a Textual Inversion [24] $\mathcal{M}_{TI}(\mathcal{P})$ caption modifier where the output $\mathcal{M}_{TI}(\mathcal{P})_s$ is a learned Textual Inversion token <*> and $\mathcal{M}_{TI}(\mathcal{P})_{\epsilon_\theta} = \varnothing$, and a DreamBooth [25] $\mathcal{M}_{DB}(\mathcal{P})$ caption modifier where $\mathcal{M}_{DB}(\mathcal{P})_s$ is a learned DreamBooth token <SKS> and $\mathcal{M}_{DB}(\mathcal{P})_{\epsilon_\theta}$ is a DreamBoothed diffusion backbone. We also include two additional control experiments. In the first, $\mathcal{M}_{ud}(\mathcal{P})$ an *unrelated* target domain style is appended to the end of the string. In the second, $\mathcal{M}_{nd}(\mathcal{P})$ a *nearby* but a different target domain style is appended to the caption. $\mathcal{M}_{TI}(\mathcal{P})$ and $\mathcal{M}_{DB}(\mathcal{P})$ require more information than the other methods, such that $\mathcal{P}$ represents a subset of unlabelled images from the target domain.

Testing. When testing the trained models on the target domain images, we want to use the same captioning modification for the test images as in the training setup. However, $\mathcal{G}_{\text{TADP}}$ introduces a confound since it naturally incorporates target domain information. For example, $\mathcal{G}_{\text{TADP}}(x)$ might produce the caption "a watercolor painting of a dog and a bird" for an image from the Watercolor2K dataset. Using the $\mathcal{M}_{simple}(\mathcal{P})$ captioning modification on this prompt would introduce redundant information and would not match the caption format used during training. In order to remove target domain information and get a plain caption that can be modified in the same manner as in the training data, we use GPT-3.5 [53] to remove all mentions of the target domain shift. For example, after using GPT-3.5 to remove mentions of the watercolor style in the above sentence, we are left with "an image of a bird and a dog." With these *GPT-3.5 cleaned captions*, we can match the caption modifications used during training when evaluating test images. This caption-cleaning strategy

Method	Dark Zurich-val mIoU	ND mIoU
DAFormer [50]	–	54.1
Refign-DAFormer [51]	–	56.8
PTDiffSeg [6]	37.0	–
$\text{TADP}_{\text{null}}$	**42.8**	57.5
$\text{TADP}_{\text{simple}}$	39.1	56.9
$\text{TADP}_{\text{TextualInversion}}$	41.4	**60.8**
$\text{TADP}_{\text{DreamBooth}}$	38.9	60.4
$\text{TADP}_{\text{NearbyDomain}}$	41.9	56.9
$\text{TADP}_{\text{UnrelatedDomain}}$	42.3	55.1

Table 6.4: Cross-domain semantic segmentation. Cityscapes (CD) to Dark Zurich (DZ) val and Nighttime Driving (ND). We report the mIoU. Our method sets a new SOTA for DarkZurich and Nighttime Driving.

lets us control how target domain information is included in the test image captions, ensuring that test captions are in the same domain as train captions.

Evaluation. We evaluate cross-domain transfer on several datasets. We train our model on Pascal VOC [15, 19] object detection and evaluate on Watercolor2K (W2K) [20] and Comic2K (C2K) [20]. We also train our model on the Cityscapes [21] dataset and evaluate on the Nighttime Driving (ND) [23] and Dark Zurich-val (DZ-val) [22] datasets. We show results in Tabs. 6.4, 6.5. In the following sections, we also report the average performance of each method on the cross-domain segmentation datasets (average mIoU) and the cross-domain object detection datasets (average AP).

Null caption modifier. The null captions have no target domain information. In this setting, the model is trained with captions with no target domain information and tested with GPT-3.5 cleaned target domain captions. We find diffusion pre-training to be extraordinarily powerful on its own, with just plain captions (no target domain information); the model already achieves SOTA on VOC→W2K with 72.1 AP_{50}, SOTA on CD→DZ-val with 42.8 mIoU and SOTA on CD→ND with 60.8 mIoU. Our model performs better than the current SOTA [32] on VOC→W2K and worse on VOC→C2K (highlighted in yellow in Tab. 6.5). However, [32] uses a large extra training dataset from the target (comic) domain, so we highlight in bold our results in Tab. 6.5 to show they outperform all other methods that use only images in C2K as examples from the target domain. Furthermore, these results are with a lightweight

Method	Watercolor2k		Comic2k	
	AP	AP_{50}	AP	AP_{50}
Single Domain Generalization (SGD)				
CLIP the gap [38]	–	33.5	–	43.4
Cross domain weakly supervised object detection				
PLGE [35]	–	56.5	–	41.7
ICCM [37]	–	57.4	–	37.1
H2FA R-CNN [36]	–	59.9	–	46.4
Unsupervised domain adaptation object detection				
ADDA [52]	–	49.8	–	23.8
MCAR [33]	–	56.0	–	33.5
UMT [31]	–	58.1	–	–
DASS-Detector (extra data) [32]	–	71.5	–	**64.2**
$TADP_{null}$	42.1	72.1	31.1	**57.4**
$TADP_{simple}$	**43.5**	**72.2**	31.9	56.6
$TADP_{TextualInversion}$	43.2	**72.2**	**33.2**	**57.4**
$TADP_{DreamBooth}$	43.2	**72.2**	32.9	56.9
$TADP_{NearbyDomain}$	42.0	71.5	31.8	56.4
$TADP_{UnrelatedDomain}$	42.2	71.9	32.0	55.9

Table 6.5: Cross-domain object detection. Pascal VOC to Watercolor2k and Comic2k. We report the AP and AP_{50}. Our method sets a new SOTA for Watercolor2K.

FPN [54] head, in contrast to other competitive methods like Refign [51], which uses a heavier decoder. These captions achieve 50.5 average mIoU and 36.6 average AP.

Simple caption modifier. We then add target domain information to our captions by prepending the target domain's semantic shift to the generic captions. These caption modifiers are hand-crafted. For example, "a dog and a bird" becomes "a X style painting of a dog and a bird" (where X is watercolor for W2K and comic for C2K) and "a dark night photo of a dog and a bird" for DZ. These captions achieve 48.0 average mIoU and 37.7 average AP.

Textual Inversion caption modifier. Textual inversion [24] is a method that learns a target concept (an object or style) from a set of images and encodes it into a new token. We learn a novel token from target domain image samples to further increase image-text alignment (for details, see Sec. 6.10.1). In this setting, the sentence template becomes "a <token> style painting of a dog and a bird." We find that, on average, Textual Inversion captions perform the best, achieving 51.1 average mIoU and 38.2 average AP.

DreamBooth caption modifier. DreamBooth-ing [25] aims to achieve the same goal as textual inversion. Along with learning a new token, the stable-diffusion backbone itself is fine-tuned with a set of target domain images (for details, see Sec. 6.10.1). We swap the stable diffusion backbone with the DreamBooth-ed backbone before training. We use the same template as in textual inversion. These captions achieve 49.7 average mIoU and 38.1 average AP.

Ablations. We ablate our target domain alignment strategy by introducing unrelated and nearby target-domain style modifications. For example, this would be "a **dashcam** photo of a dog and a bird" (unrelated) and "a **constructivism** painting of a dog and a bird" (nearby) for the W2K and C2K datasets. "A **watercolor** painting of a car on the street" (unrelated) and "a **foggy** photo of a car on the street" for the ND and DZ-val datasets. We find these off-target domains reduce performance on all datasets.

6.6 Discussion

We present a method for image-text alignment that is general, fully automated, and can be applied to any diffusion-based perception model. To achieve this, we systematically explore the impact of text-image alignment on semantic segmentation, depth estimation, and object detection. We investigate whether similar principles apply in the cross-domain setting and find that alignment towards the target domain during training improves downstream cross-domain performance.

We find that EOS token averaging for prompting does not work as effectively as strings for the objects in the image. Our oracle ablation experiments show that our diffusion pre-trained segmentation model is particularly sensitive to missing classes (reduced recall) and less sensitive to off-target classes (reduced precision), and both have a negative impact. Our results show that aligning text prompts to the image is important in identifying/generating good multi-scale feature maps for the downstream segmentation head. This implies that the multi-scale features and latent representations do not naturally identify semantic concepts without the

guidance of the text in diffusion models. Moreover, proper latent scaling is crucial for downstream vision tasks. Lastly, we show how using a captioner, which has the benefit of being open vocabulary, high precision, and downstream task agnostic, to prompt the diffusion pre-trained segmentation model automatically improves performance significantly over providing all possible class names.

We also find that diffusion models can be used effectively for cross-domain tasks. Our model, without any captions, already surpasses several SOTA results in cross-domain tasks due to the diffusion backbone's generalizability. We find that good target domain alignment can help with cross-domain performance for some domains, and misalignment leads to worse performance. Capturing information about target domain styles in words alone can be difficult. For these cases, we show that model personalization through Textual Inversion or Dreambooth can bridge the gap without requiring labeled data. Future work could explore how to expand our framework to generalize to multiple unseen domains. Future work may also explore closed vocabulary captioners that are more task-specific to get closer to oracle-level performance.

6.6.1 Acknowledgements

Pietro Perona and Markus Marks were supported by the National Institutes of Health (NIH R01 MH123612A) and the Caltech Chen Institute (Neuroscience Research Grant Award). Pietro Perona, Neehar Kondapaneni, Rogerio Guimaraes, and Markus Marks were supported by the Simons Foundation (NC-GB-CULM-00002953-02). Manuel Knott was supported by an ETH Zurich Doc.Mobility Fellowship. We thank Oisin Mac Aodha, Yisong Yue, and Mathieu Salzmann for their valuable inputs that helped improve this work.

Appendix

6.7 Cross-attention Analysis

Qualitative image-to-image variation analysis. We present a qualitative and quantitative analysis of the effect of off-target class names added to the prompt. In Fig. 6.5, we use the stable diffusion image to image (img2img) variation pipeline (with the original Stable Diffusion 1.5 weights) to qualitatively analyze the effects of prompts with off-target classes. The img2img variation pipeline encodes a real image into a latent representation, adds a user-specified amount of noise to the latent representation, and de-noises it (according to a user-specified prompt) to generate a variation on the original image. The amount of noise added is dictated by a strength ratio indicating how much variation should occur. A higher ratio results in more added noise and more denoising steps, allowing a relatively higher impact of the new text prompt on the image. We find that $\mathcal{C}_{\text{ClassNames}}$ (see caption for details) results in variations that incorporate the off-target classes. This effect is most clear looking across the panels left to right in which objects belonging to off-target classes (an airplane and a train) become more prominent. These qualitative results imply that this prompt modifies the latent representation to incorporate information about off-target classes, potentially making the downstream task more difficult. In contrast, using the BLIP prompt changes the image, but the semantics (position of objects, classes present) of the image variation are significantly closer to the original. These results suggest a mechanism for how off-target classes may impact our vision models. We quantitatively measure this effect using a fully trained Oracle model in the following section.

Copy-Paste Experiment. An interesting property in Fig. 6.4 is that the word bottle has strong cross-attention over the neck of the bird. We hypothesize that diffusion models seek to find the nearest match for each token since they are trained to generate images that correspond to the prompt. We test this hypothesis on a base image of a dog and a bird. We first visualize the cross-attention maps for a set of object labels. We find that the words bottle, cat, and horse have a strong cross-attention to the bird, dog, and dog, respectively. We paste a bottle, cat, and horse into the base image to see if the diffusion model will localize the "correct" objects if they are present. In Fig. 6.6, we show that the cross-attention maps prefer to localize the "correct" object, suggesting our hypothesis is correct.

Averaged EOS Tokens: Averaging vs. EOS? Averaged EOS Tokens create diffuse attention maps that empirically harm performance. What is the actual cause of the

decrease in performance? Is it averaging, or is it the usage of many EOS tokens? We replace the averaged EOS tokens with single prompt EOS tokens and find that the attention maps are still diffuse. This indicates that the usage of EOS tokens is the primary cause of the diffuse attention maps and not the averaging.

Quantitative effect of $\mathcal{C}_{\text{ClassNames}}$ on Oracle model. To quantify the impact of the off-target classes on the downstream vision task, we measure the averaged pixel-wise scores (normalized via Softmax) per class when passing the $\mathcal{C}_{\text{ClassNames}}$ to the Oracle segmentation model for Pascal VOC 2012 (Fig. 6.8). We compare this to the original oracle prompt. We find that including the off-target prompts significantly increases the probability of a pixel being misclassified as one of the semantically nearby off-target classes. For example, if the original image contains a cow, including the words dog and sheep, it significantly raises the probability of misclassifying the pixels belonging to the cow as pixels belonging to a dog or a sheep. These results indicate that the task-specific head picks up the effect of off-target classes and is incorporated into the output.

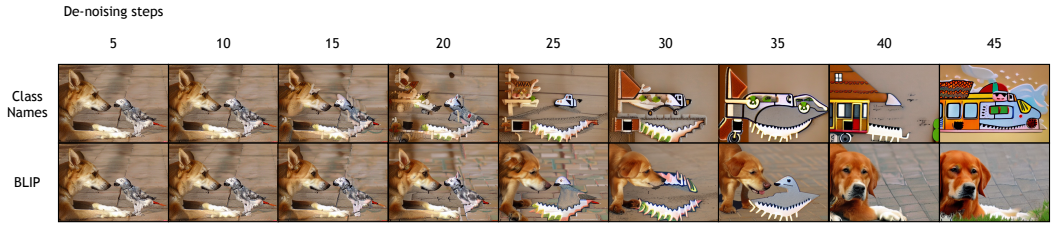


Figure 6.5: Qualitative image-to-image variation. An untrained stable diffusion model is passed an image to perform image-to-image variation. The number of denoising steps conducted increases from left to right (5 to 45 out of a total of 50). On the top row, we pass all the class names in Pascal VOC 2012: *"background airplane bicycle bird boat bottle bus car cat chair cow dining table dog horse motorcycle person potted plant sheep sofa train television"*. In the bottom row we pass the BLIP caption *"a bird and a dog"*.

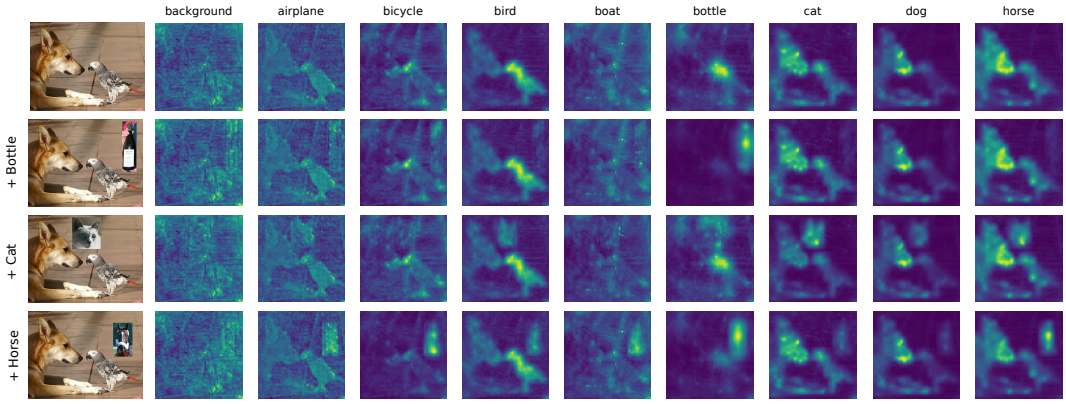


Figure 6.6: Copy-paste experiment. A bottle, a cat, and a horse from different images are copied and pasted into our base image to see how the cross-attention maps change. The label on the left describes the category of the item that has been pasted into the image. The labels above each map describe the cross-attention map corresponding to the token for that label.

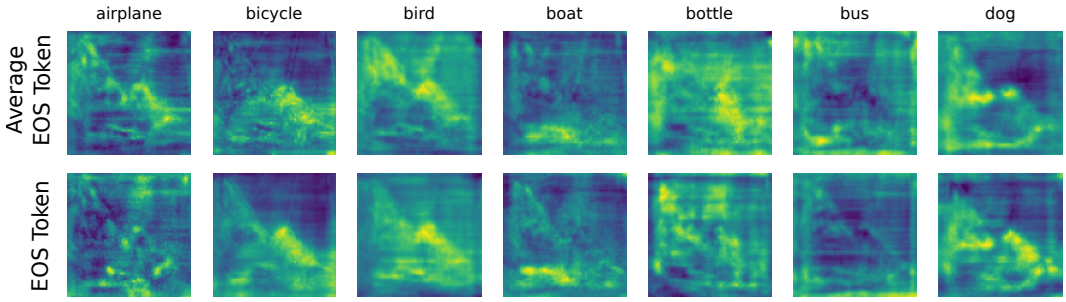


Figure 6.7: Averaging vs. EOS. In [7], for each class name, the EOS token from 80 prompts (containing the class name) was averaged together. The averaged EOS tokens for each class were concatenated together and passed to the diffusion model as text input. We explore if averaging drives the diffuse nature of the cross-attention maps. We replace the 80 prompt templates with a single prompt template: "a photo of a {class name}" and visualize the cross-attention maps. In the top row, we show the averaged template EOS tokens. In the bottom row, we show the single template EOS tokens.

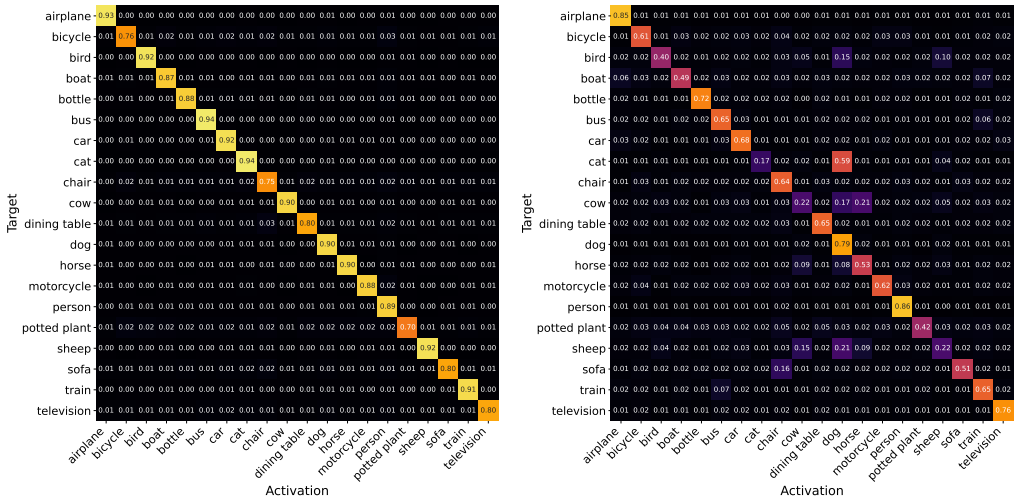


Figure 6.8: Impact of off-target classes on semantic segmentation performance. The matrices show normalized scores averaged over pixels on Pascal VOC 2012 for an oracle-trained model when receiving either present class names (left) or all class names (right).

6.8 Additional ADE20K Results

Method	4K Iters		8K Iters	
	mIoU^{ss}	mIoU^{ms}	mIoU^{ss}	mIoU^{ms}
VPD (null text)	41.5	-	46.9	-
VPD_{A32} [7]	43.1	44.2	48.7	49.5
VPD(R)	42.6	43.6	49.2	50.4
VPD(LS)	45.0	45.8	50.5	51.1
TADP-20 (Ours)	**50.2**	**50.9**	**52.8**	**54.1**
TADP(TA)-20 (Ours)	49.9	50.7	52.7	53.4

Table 6.6: Semantic segmentation fast schedule on ADE20K. Our method has a large advantage over prior work on the fast schedule with significantly better performance in both the single-scale and multi-scale evaluations for 4k and 8k iterations.

		Recall		
		0.50	**0.75**	**1.00**
Precision	**0.50**	49.53	52.00	55.22
	0.75	49.17	51.46	58.62
	1.00	50.20	54.82	63.29

Table 6.7: ADE20K — Oracle Precision-Recall Ablations We modify the oracle captions by randomly adding or removing classes such that the precision and recall are 0.50, 0.75, or 1.00. We train models on ADE20K on a fast schedule (4K) using these captions. The 4k iteration oracle equivalent is highlighted in blue.

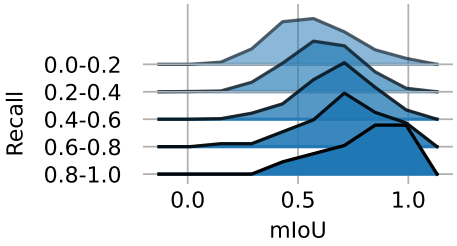


Figure 6.9: Recall analysis. ADE20k mIOU per image with respect to the recall of classes present in the caption. We embedded each word in our caption with CLIP's text encoder. We considered a cosine similarity of ≥ 0.9 with the embedded class name as a match. Linear regression analysis shows positive correlations between recall and mIoU ($r = 0.28$).

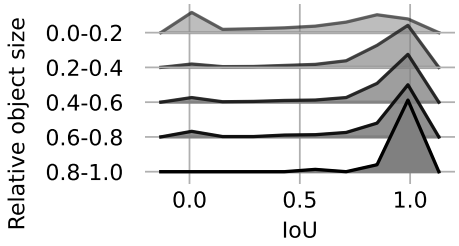


Figure 6.10: Object size analysis. ADE20k IOU per object image with respect to the relative object size (pixels divided by total pixels). Linear regression analysis shows positive correlations between relative object size and the IoU-score of a class ($r = 0.40$).

6.9 Qualitative Examples

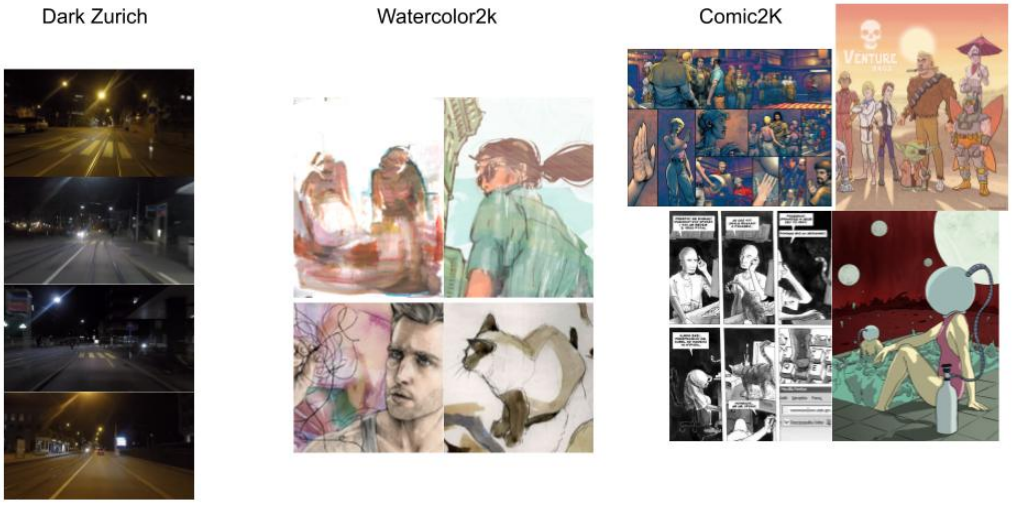


Figure 6.11: Ground truth examples of the tokenized datasets.

Figure 6.12: Textual inversion and Dreambooth tokens of Cityscapes to Dark Zurich.

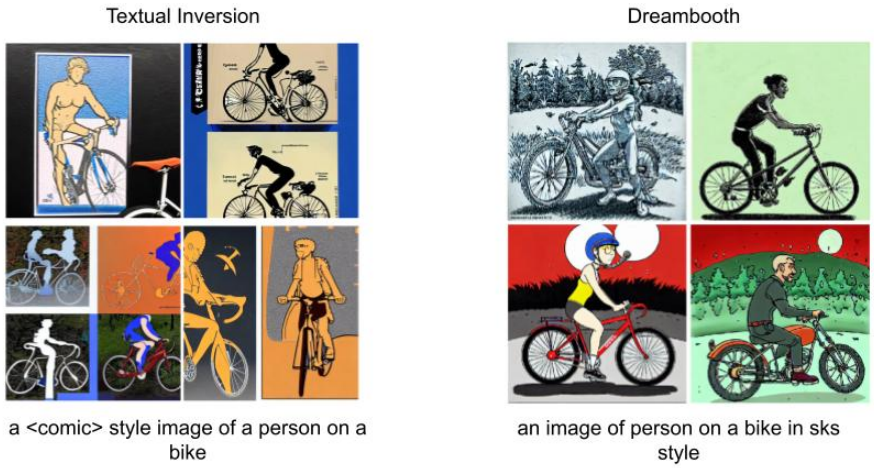


Figure 6.13: Textual inversion and Dreambooth tokens of VOC to Comic.

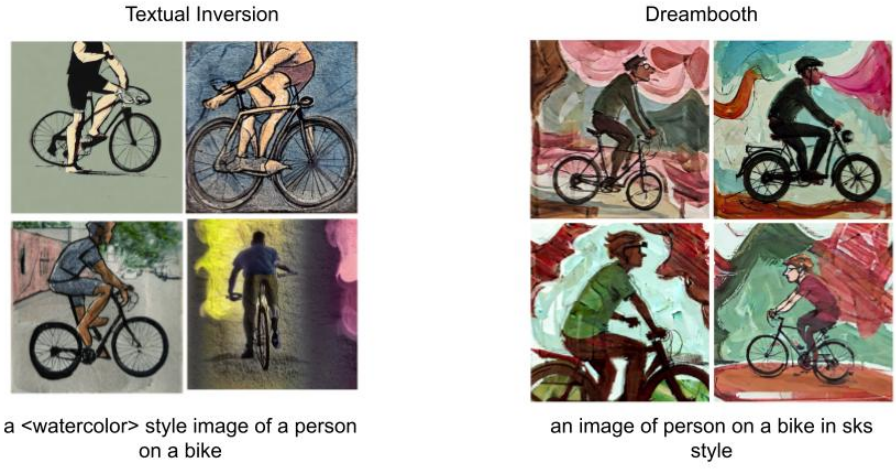


Figure 6.14: Textual inversion and Dreambooth tokens of VOC to Watercolor.

Figure 6.15: Predictions (top) and Ground Truth (bottom) visualizations for Pascal VOC2012.

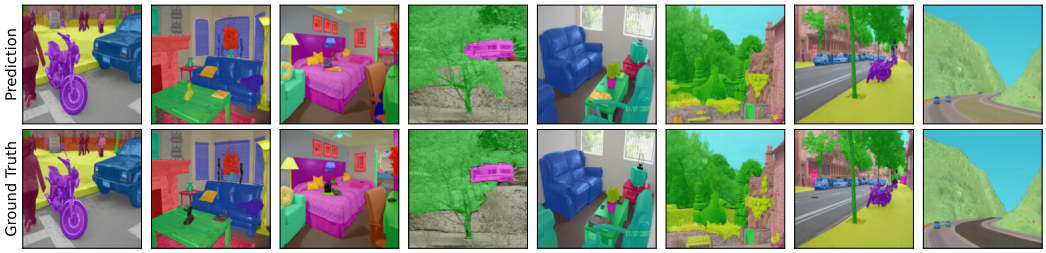


Figure 6.16: Predictions (top) and Ground Truth (bottom) visualizations for ADE20K.

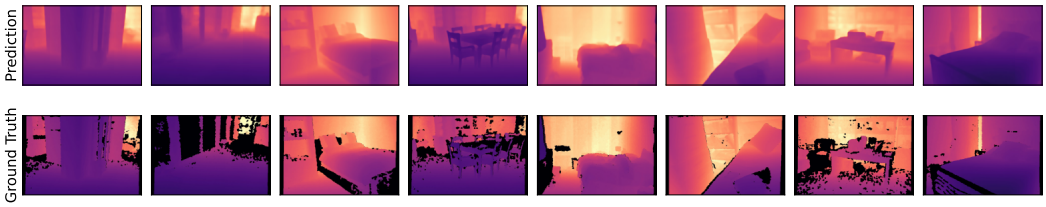


Figure 6.17: Predictions (top) and Ground Truth (bottom) visualizations for NYUv2 Depth.

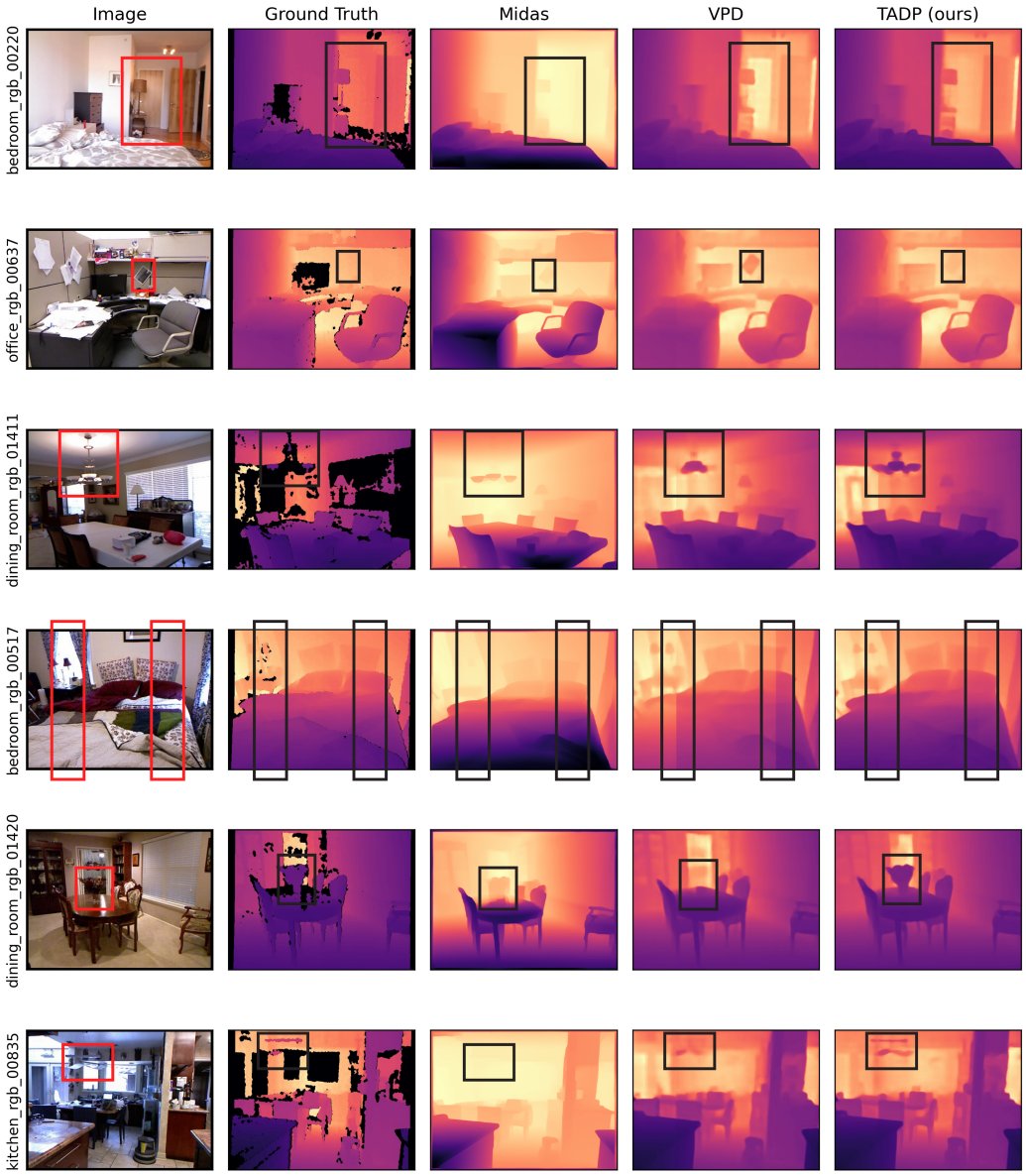


Figure 6.18: Depth estimation comparison: Image, Ground Truth, and prediction visualizations for Midas, VPD, and TADP (ours) in NYUv2 Depth. Black boxes (red on original image) show where TADP is better than Midas and/or VPD.

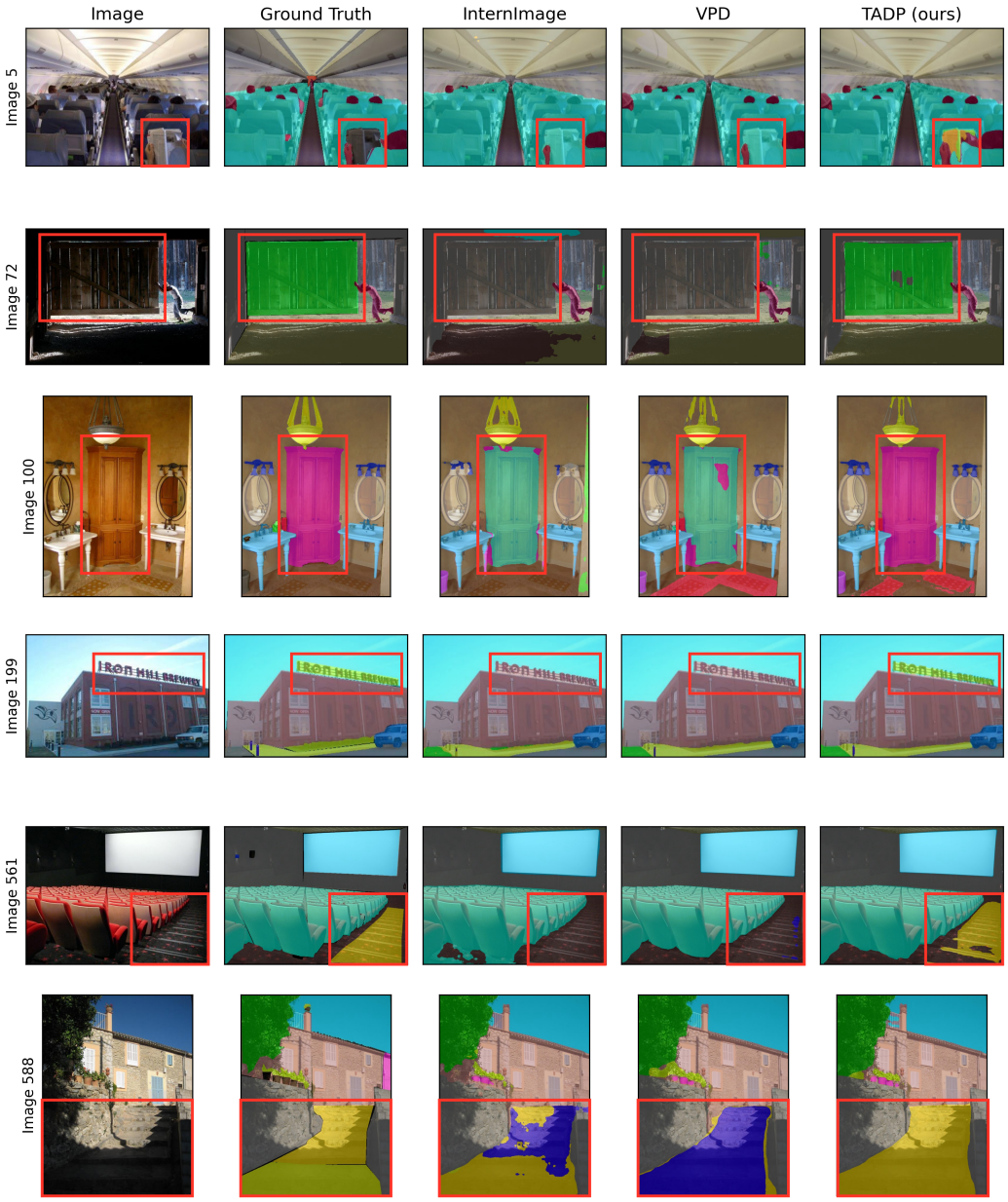


Figure 6.19: Image segmentation comparison: Image, Ground Truth, and prediction visualizations for InternImage, VPD, and TADP (ours) in ADE20K. Red boxes show where TADP is better than InternImage and/or VPD.

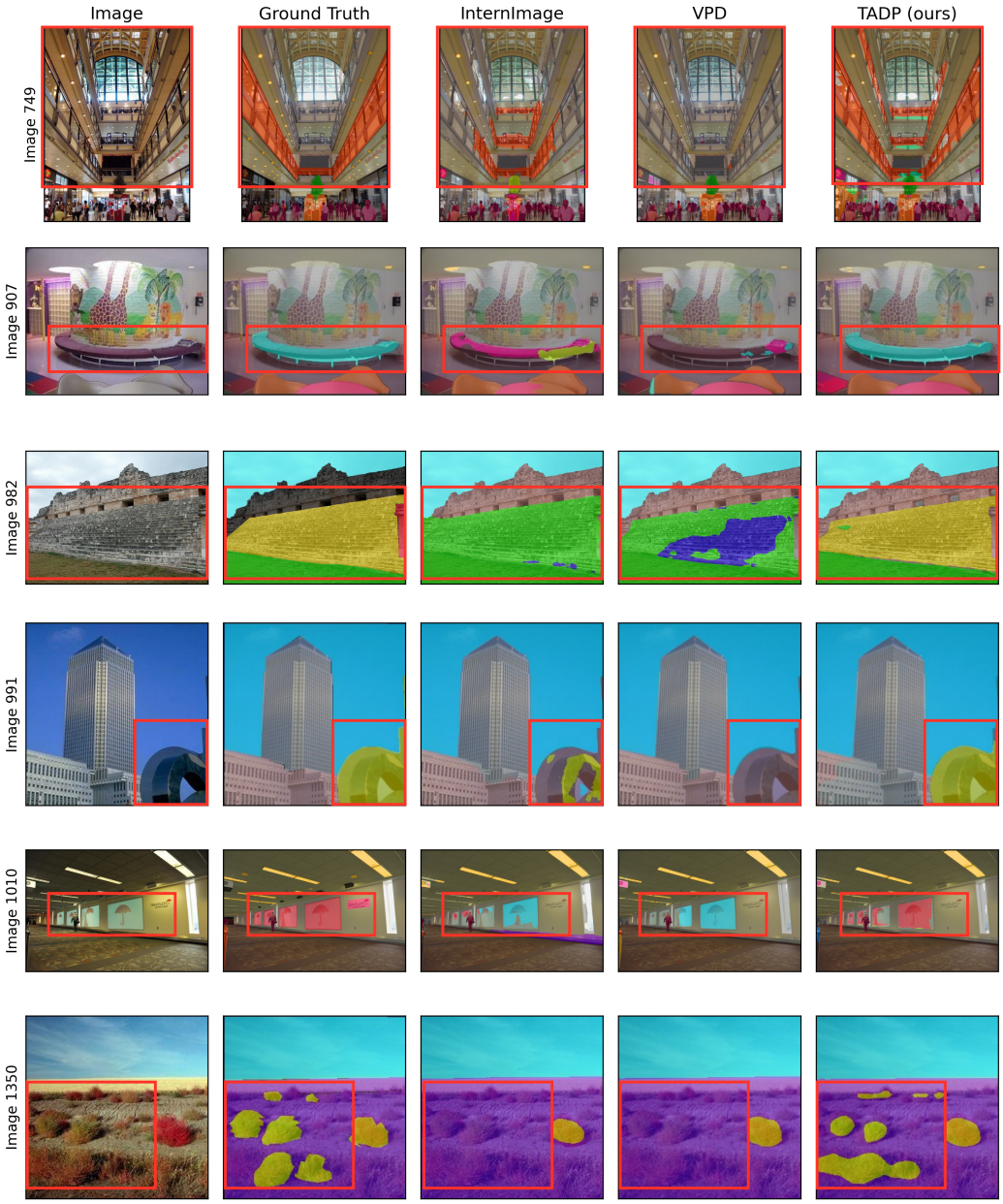


Figure 6.20: Image segmentation comparison: Image, Ground Truth, and prediction visualizations for InternImage, VPD, and TADP (ours) in ADE20K. Red boxes show where TADP is better than InternImage and/or VPD.

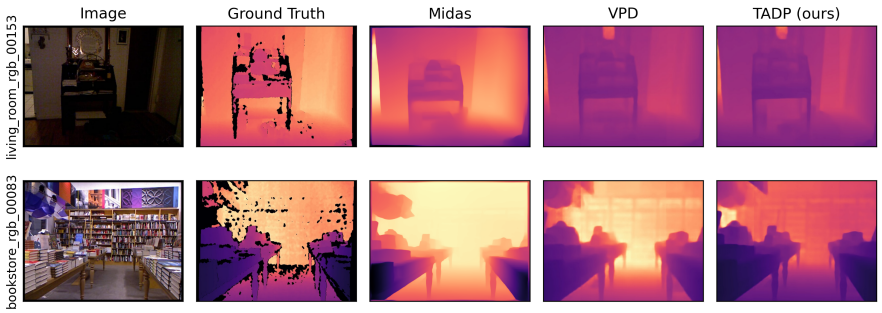


Figure 6.21: Depth estimation comparison: Image, Ground Truth, and prediction visualizations for Midas, VPD, and TADP (ours) in NYUv2 Depth. TADP is worse than Midas and/or VPD in these images in terms of the general scale

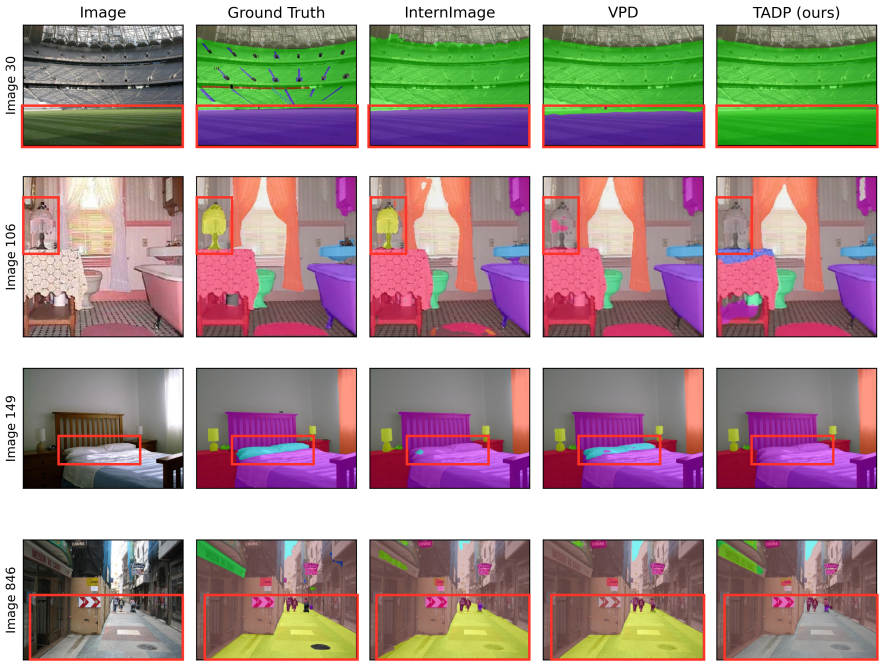


Figure 6.22: Image segmentation comparison: Image, Ground Truth, and prediction visualizations for InternImage, VPD, and TADP (ours) in ADE20K. Red boxes show where TADP is worse than InternImage and/or VPD.

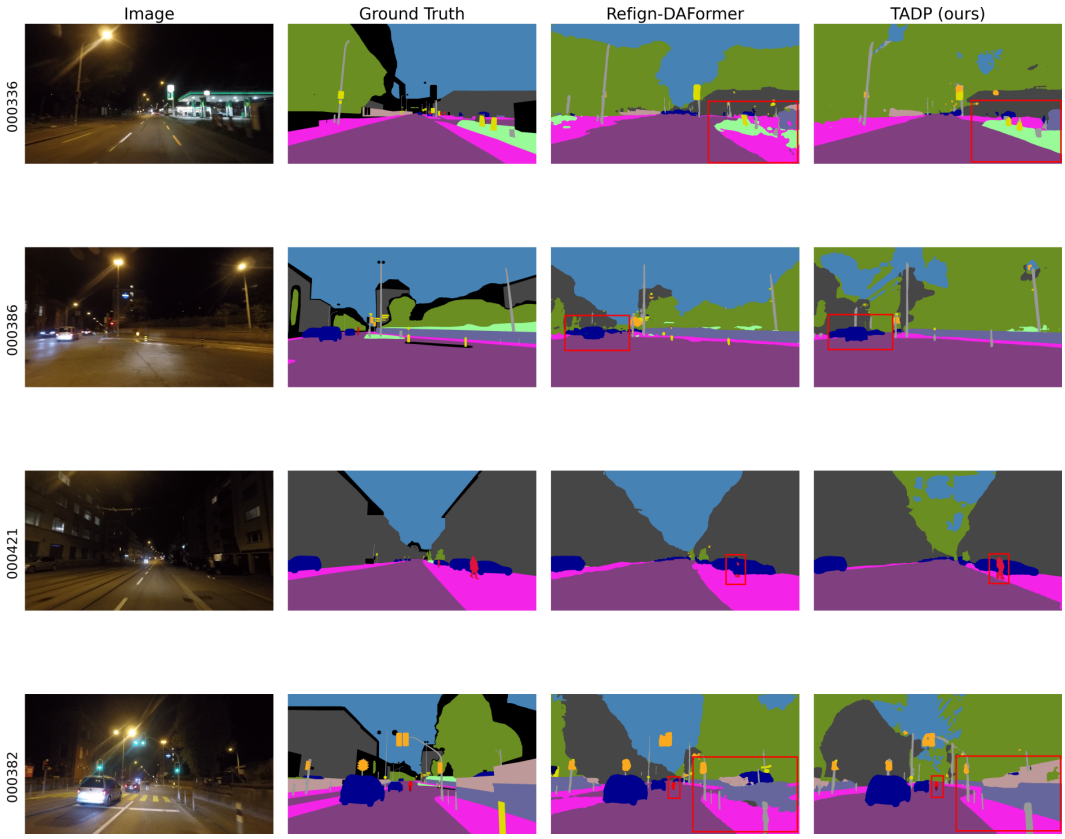


Figure 6.23: Cross-domain image segmentation comparison: Image, Ground Truth, and prediction visualizations for Refign-DAFormer, and TADP (ours) for Cityscapes to Dark Zurich Val. Red boxes show where TADP is better than Refign-DAFormer.

Figure 6.24: Cross-domain object detection comparison: Image, Ground Truth, and prediction visualizations for DASS, and TADP (ours) for Pascal VOC to Watercolor2k. Red boxes show the detections of each model. Notice that TADP not only beats DASS mostly, but also finds more objects than the ones annotated in the ground truth.

6.10 Implementation Details

To isolate the effects of our text-image alignment method, we ensure our model setup precisely follows prior work. Following VPD [7], we jointly train the task-specific head and the diffusion backbone. The learning rate of the backbone is set to 1/10 the learning rate of the head to preserve the benefits of pre-training better. We describe the different tasks by describing H and $\mathcal{L}_H$. We use an FPN [54] head with a cross-entropy loss for segmentation. We use the same convolutional head used in VPD for monocular depth estimation with a Scale-Invariant loss [55]. For object detection, we use a Faster-RCNN head with the standard Faster-RCNN loss [56][1]. Further details of the training setup can be found in Tab. 6.8 and Tab. 6.9. In our single-domain tables, we include our reproduction of VPD, denoted with a (R). We compute our relative gains with our reproduced numbers, with the same seed for all experiments.

[1] Object detection was not explored in VPD.

Hyperparameter	Value
Learning Rate	0.00008
Batch Size	2
Optimizer	AdamW
Weight Decay	0.005
Warmup Iters	1500
Warmup Ratio	$1e-6$
U-Net Learning Rate Scale	0.01
Training Steps	80000

(a) ADE20k - full schedule

Hyperparameter	Value
Learning Rate	0.00016
Batch Size	2
Optimizer	AdamW
Weight Decay	0.005
Warmup Iters	150
Warmup Ratio	$1e-6$
Unet Learning Rate Scale	0.01
Training Steps	8000

(b) ADE20k - fast schedule 8k

Hyperparameter	Value
Learning Rate	0.00016
Batch Size	2
Optimizer	AdamW
Weight Decay	0.005
Warmup Iters	75
Warmup Ratio	$1e-6$
Unet Learning Rate Scale	0.01
Training Steps	4000

(c) ADE20k - fast schedule 4k

Hyperparameter	Value
Learning Rate	$5e-4$
Batch Size	3
Optimizer	AdamW
Weight Decay	0.1
Layer Decay	0.9
Epochs	25
Drop Path Rate	0.9

(d) NYUv2

Hyperparameter	Value
Learning Rate	$5e-4$
Batch Size	3
Optimizer	AdamW
Weight Decay	0.1
Layer Decay	0.9
Epochs	1
Drop Path Rate	0.9

(e) NYUv2 - fast schedule

Hyperparameter	Value
Learning Rate	0.00001
Batch Size	2
Gradient Accumulation	4
Epochs	15
Optimizer	AdamW
Weight Decay	0.01

(f) Pascal VOC

Table 6.8: Single-Domain Hyperparameters.

Hyperparameter	Value
Learning Rate	0.00008
Batch Size	2
Optimizer	AdamW
Weight Decay	0.005
Warmup Iters	1500
Warmup Ratio	$1e-6$
Unet Learning Rate Scale	0.01
Training Steps	40000

(a) Cityscapes → Dark Zurich & NightTime Driving

Hyperparameter	Value
Learning Rate	0.00001
Batch Size	2
Epochs	100
Optimizer	AdamW
Weight Decay	0.01
Learning Rate Schedule	Lambda

(b) Pascal VOC → Watercolor & Comic

Hyperparameter	Value
Prior Preservation Cls Images	200
Learning Rate	$5e-6$
Training Steps	1000

(c) Dreambooth Hyperparameters

Hyperparameter	Value
Steps	3000
Learning Rate	$5.0e-04$
Batch Size	1
Gradient Accumulation	4

(d) Textual Inversion Hyperparameters

Table 6.9: Cross-Domain Hyperparameters.

6.10.1 Model personalization

For textual inversion, we use 500 images from DZ-train and five images for W2K and C2K and train all tokens for 1000 steps. We use a constant learning rate scheduler with a learning rate of $5e-4$ and no warmup. For Dreambooth, we use the same images as in textual inversion but train the model for 500 steps (DZ) steps or 1000 steps (W2K and C2K). We use a learning rate of $2e-6$ with a constant learning rate scheduler and no warmup. We use no prior preservation loss.

References

[1] Neehar Kondapaneni et al. "Text-image alignment for diffusion-based perception." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 13883–13893. URL: https://openaccess.thecvf.com/content/CVPR2024/html/Kondapaneni_Text-Image_Alignment_for_Diffusion-Based_Perception_CVPR_2024_paper.html.

[2] Jiahui Yu et al. "Scaling Autoregressive Models for Content-Rich Text-to-Image Generation." In: *arXiv preprint arXiv:2206.10789* (2022).

[3] Aditya Ramesh et al. "Hierarchical Text-Conditional Image Generation with CLIP Latents." In: *arXiv preprint arXiv:2204.06125* (2022). (Visited on 08/24/2023).

[4] Chitwan Saharia et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding." In: *arXiv preprint arXiv:2205.11487* (2022).

[5] Robin Rombach et al. "High-Resolution Image Synthesis with Latent Diffusion Models." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022), pp. 10674–10685. (Visited on 07/27/2023).

[6] Rui Gong et al. "Prompting Diffusion Representations for Cross-Domain Semantic Segmentation." In: *arXiv preprint arXiv:2307.02138* (2023). (Visited on 07/27/2023).

[7] Wenliang Zhao et al. "Unleashing Text-to-Image Diffusion Models for Visual Perception." In: *arXiv preprint arXiv:2303.02153* (2023). (Visited on 07/27/2023).

[8] Grace Luo et al. "Diffusion Hyperfeatures: Searching Through Time and Space for Semantic Correspondence." In: *arXiv preprint arXiv:2305.14334* (2023).

[9] Luming Tang et al. "Emergent Correspondence from Image Diffusion." In: *arXiv preprint arXiv:2306.03881* (2023).

[10] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision." In: *International Conference on Machine Learning* (2021), pp. 8748–8763. (Visited on 08/21/2023).

[11] Chao Jia et al. "Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision." In: *arXiv preprint arXiv:2102.05918* (June 2021). (Visited on 08/30/2023).

[12] Yangguang Li et al. "Supervision Exists Everywhere: A Data Efficient Contrastive Language-Image Pre-training Paradigm." In: *arXiv preprint arXiv:2110.05208* (Mar. 2022). (Visited on 08/30/2023).

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Vol. 9351. Cham: Springer International Publishing, 2015, pp. 234–241. (Visited on 08/21/2023).

[14] Manuel Brack et al. "SEGA: Instructing Diffusion using Semantic Dimensions." In: *arXiv preprint arXiv:2301.12247* (2023). (Visited on 08/21/2023).

[15] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012)*. 2012. URL: http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[16] Bolei Zhou et al. "Scene Parsing through ADE20K Dataset." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5122–5130.

[17] Junnan Li et al. "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models." In: *arXiv preprint arXiv:2301.12597* (2023). (Visited on 08/09/2023).

[18] Nathan Silberman et al. "Indoor Segmentation and Support Inference from RGBD Images." In: *European Conference on Computer Vision (ECCV)* (2012). Ed. by Andrew Fitzgibbon et al. (Visited on 08/21/2023).

[19] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[20] Naoto Inoue et al. "Cross-Domain Weakly-Supervised Object Detection Through Progressive Domain Adaptation." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 5001–5009.

[21] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 3213–3223.

[22] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. "Guided Curriculum Model Adaptation and Uncertainty-Aware Evaluation for Semantic Nighttime Image Segmentation." In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 7373–7382.

[23] Dengxin Dai and Luc Van Gool. "Dark Model Adaptation: Semantic Image Segmentation from Daytime to Nighttime." In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (2018), pp. 3819–3824.

[24] Rinon Gal et al. "An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion." In: *arXiv preprint arXiv:2208.01618* (2022). (Visited on 08/21/2023).

[25] Nataniel Ruiz et al. “DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation.” In: *arXiv preprint arXiv:2208.12242* (2022). (Visited on 08/21/2023).

[26] Weijia Wu et al. “DiffuMask: Synthesizing Images with Pixel-level Annotations for Semantic Segmentation Using Diffusion Models.” In: *arXiv preprint arXiv:2303.11681* (2023). (Visited on 07/27/2023).

[27] Yasser Benigmim et al. “One-shot Unsupervised Domain Adaptation with Personalized Diffusion Models.” In: *arXiv preprint arXiv:2303.18080* (2023). (Visited on 08/26/2023).

[28] Amir Hertz et al. “Prompt-to-Prompt Image Editing with Cross Attention Control.” In: *arXiv preprint arXiv:2208.01626* (2022). (Visited on 08/29/2023).

[29] Yogesh Balaji et al. “eDiff-I: Text-to-Image Diffusion Models with an Ensemble of Expert Denoisers.” In: *arXiv preprint arXiv:2211.01324* (2022). (Visited on 08/29/2023).

[30] Christoph Schuhmann et al. “LAION-5B: An open large-scale dataset for training next generation image-text models.” In: *arXiv preprint arXiv:2210.08402* (2022). (Visited on 09/25/2023).

[31] Jinhong Deng et al. “Unbiased mean teacher for cross-domain object detection.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4091–4101.

[32] Barış Batuhan Topal, Deniz Yuret, and Tevfik Metin Sezgin. “Domain-Adaptive Self-Supervised Pre-Training for Face & Body Detection in Drawings.” In: *arXiv preprint arXiv:2211.10641* (2022).

[33] Zhen Zhao et al. “Adaptive object detection with dual multi-label prediction.” In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*. Springer. 2020, pp. 54–69.

[34] Junguang Jiang et al. “Decoupled adaptation for cross-domain object detection.” In: *arXiv preprint arXiv:2110.02578* (2021).

[35] Shengxiong Ouyang et al. “Pseudo-label generation-evaluation framework for cross domain weakly supervised object detection.” In: *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2021, pp. 724–728.

[36] Yunqiu Xu et al. “H2fa r-cnn: Holistic and hierarchical feature alignment for cross-domain weakly supervised object detection.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 14329–14339.

[37] Luwei Hou et al. “Informative and consistent correspondence mining for cross-domain weakly supervised object detection.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9929–9938.

[38] Vidit Vidit, Martin Engilberge, and Mathieu Salzmann. “CLIP the Gap: A Single Domain Generalization Approach for Object Detection.” In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023), pp. 3219–3229.

[39] Christoph Schuhmann et al. “LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs.” In: *arXiv preprint arXiv:2111.02114* (2021). (Visited on 08/29/2023).

[40] Yuxin Fang et al. “EVA: Exploring the Limits of Masked Visual Representation Learning at Scale.” In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 19358–19369.

[41] Wenhai Wang et al. “InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions.” In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 14408–14419.

[42] Yongming Rao et al. “DenseCLIP: Language-Guided Dense Prediction with Context-Aware Prompting.” In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022), pp. 18061–18070. (Visited on 08/28/2023).

[43] Zhe Chen et al. “Vision Transformer Adapter for Dense Predictions.” In: *arXiv preprint arXiv:2205.08534* (2022).

[44] Wen Wang et al. “Image as a Foreign Language: BEIT Pretraining for Vision and Vision-Language Tasks.” In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023), pp. 19175–19186.

[45] Peng Wang et al. “ONE-PEACE: Exploring One General Representation Model Toward Unlimited Modalities.” In: *arXiv preprint arXiv:2305.11172* (2023). (Visited on 08/29/2023).

[46] Ze Liu et al. “Swin Transformer V2: Scaling Up Capacity and Resolution.” In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 11999–12009.

[47] Jia Ning et al. “All in Tokens: Unifying Output Space of Visual Tasks via Soft Token.” In: *arXiv preprint arXiv:2301.02229* (2023). (Visited on 08/29/2023).

[48] Shariq Farooq Bhat et al. “ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth.” In: *arXiv preprint arXiv:2302.12288* (2023). (Visited on 08/29/2023).

[49] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O’Reilly Media, Inc., 2009.

[50] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. "DAFormer: Improving Network Architectures and Training Strategies for Domain-Adaptive Semantic Segmentation." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Mar. 2022), pp. 9914–9925. (Visited on 08/30/2023).

[51] David Brüggemann et al. "Refign: Align and Refine for Adaptation of Semantic Segmentation to Adverse Conditions." In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2022). (Visited on 08/30/2023).

[52] Eric Tzeng et al. "Adversarial discriminative domain adaptation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176.

[53] Tom Brown et al. "Language models are few-shot learners." In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[54] Alexander Kirillov et al. "Panoptic Feature Pyramid Networks." In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 6392–6401.

[55] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network." In: *Advances in Neural Information Processing Systems 27 (NIPS 2014)*. 2014. (Visited on 08/29/2023).

[56] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (June 2017), pp. 1137–1149. (Visited on 08/30/2023).

Chapter 7

A NUMBER SENSE AS AN EMERGENT PROPERTY OF THE MANIPULATING BRAIN

Neehar Kondapaneni and Pietro Perona. "A number sense as an emergent property of the manipulating brain." In: *Scientific Reports* 14.1 (2024), p. 6858. URL: https://www.nature.com/articles/s41598-024-56828-2.

7.1 Abstract

The ability to understand and manipulate numbers and quantities emerges during childhood, but the mechanism through which humans acquire and develop this ability is still poorly understood. We explore this question through a model, assuming that the learner is able to pick up and place small objects from, and to, locations of its choosing, and will spontaneously engage in such undirected manipulation. We further assume that the learner's visual system will monitor the changing arrangements of objects in the scene and will learn to predict the effects of each action by comparing perception with a supervisory signal from the motor system. We model perception using standard deep networks for feature extraction and classification, and gradient descent learning. Our main finding is that, from learning the task of action prediction, an unexpected image representation emerges exhibiting regularities that foreshadow the perception and representation of numbers and quantity. These include distinct categories for zero and the first few natural numbers, a strict ordering of the numbers, and a one-dimensional signal that correlates with numerical quantity. As a result, our model acquires the ability to estimate *numerosity*, i.e. the number of objects in the scene, as well as *subitization*, i.e. the ability to recognize at a glance the exact number of objects in small scenes. Remarkably, subitization and numerosity estimation extrapolate to scenes containing many objects, far beyond the three objects used during training. We conclude that important aspects of a facility with numbers and quantities may be learned with supervision from a simple pre-training task. Our observations suggest that cross-modal learning is a powerful learning mechanism that may be harnessed in artificial intelligence.

7.2 Introduction

7.2.1 Background

Mathematics, one of the most distinctive expressions of human intelligence, is founded on the ability to reason about abstract entities. We are interested in the question of how humans develop an intuitive facility with numbers and quantities, and how they come to recognize numbers as an abstract property of sets of objects. There is wide agreement that innate mechanisms play a strong role in developing a *number sense* [2, 3, 4], that development and learning also play an important role [3], that naming numbers is not necessary for the perception of quantities [5, 6], and a number of brain areas are involved in processing numbers [7, 8]. Quantity-tuned units have been described in physiology experiments [4, 9, 10, 11] as well as in computational studies [12, 13, 14, 15].

7.2.2 Related Work

The role of learning in developing abilities that relate to the natural numbers and estimation has been recently explored using computational models. Fang et al. [16] trained a recurrent neural network to count sequentially and Sabathiel et al. [17] showed that a neural network can be trained to anticipate the actions of a teacher on three counting-related tasks – they find that specific patterns of activity in the network's units correlate with quantities. The ability to perceive *numerosity*, i.e. a rough estimate of the number of objects in a set, was explored by Stoianov, Zorzi and Testolin [12, 13], who trained a deep network encoder to efficiently reconstruct patterns composed of dots, and found that the network developed units or "neurons" that were coarsely tuned to quantity, and by Nasr et al. [14], who found the same effect in a deep neural network that was trained on visual object classification, an unrelated task. In these models quantity-sensitive units are an emergent property. In a recent study, Kim et al. [15] observed that a random network with no training will exhibit quantity-sensitive units. After identifying these units, [12, 13, 14, 15] train a supervised classifier on a two-set comparison task to assess numerosity properties encoded by the deep networks. These works showed that training a classifier with supervision, in which the classifier is trained and evaluated on the same task and data distribution, is sufficient for recruiting quantity-tuned units for relative numerosity comparison. Our work focuses on this supervised second stage. Can more be learned with less supervision? We show that a representation for numerosity, that generalizes to several tasks and extrapolates to large quntities, may arise through a simple, supervised pre-training task. In contrast to prior work, our pre-training task

only contains scenes with up to 3 objects, and our model generalizes to scenes with up to 30 objects.

7.2.3 Approach

We focus on the interplay of action and perception as a possible avenue for this to happen. More specifically, we explore whether perception, as it is naturally trained during object manipulation, may develop representations that support a number sense. In order to test this hypothesis we propose a model where perception learns how specific actions modify the world. The model shows that perception develops a representation of the scene which, as an emergent property, can enable the ability to perceive numbers and estimate quantities at a glance [18, 19].

In order to ground intuition, consider a child who has learned to pick up objects, one at a time, and let them go at a chosen location. Imagine the child sitting comfortably and playing with small toys (acorns, Legos, sea shells) which may be dropped into a bowl. We will assume that the child has already learned to perform at will, and tell apart, three distinct operations (Fig. 7.1A). The *put* (P) operation consists of picking up an object from the surrounding space and dropping it into the bowl. The *take* (T) operation consists in doing the opposite: picking up an object from the bowl and discarding it. The *shake* (S) operation consists of agitating the bowl so that the objects inside change their position randomly without falling out. Objects in the bowl may be randomly moved during put and take as well.

We hypothesize that the visual system of the learner is engaged in observing the scene, and its goal is predicting the action that has taken place [20]as a result of manipulation. By comparing its prediction with a copy of the action signal from the motor system it may correct its perception, and improve the accuracy of its predictions over time. Thus, by performing P, T, and S actions in a random sequence, manipulation generates a sequence of labeled two-set comparisons to learn from.

We assume two trainable modules in the visual system: a “perception” module that produces a representation of the scene, and a “classification” module that compares representations and guesses the action (Fig. 7.1).

During development, perceptual maps emerge, capable of processing various scene properties. These range from basic elements like orientation [21] and boundaries [22] to more complex features such as faces [23] and objects [24, 25]. We propose that, while the child is playing, the visual system is being trained to use one or more such maps to build a representation that facilitates the comparison of the pair of images

that are seen before and after a manipulation. These representations are often called embeddings in machine learning.

A classifier network is simultaneously trained to predict the action (P, T, S) from the representation of the pair of images (see Fig. 7.1). As a result, the visual system is progressively trained through spontaneous play to predict (or, more accurately, post-dict) which operation took place that changed the appearance of the bowl.

We postulate that signals from the motor system are available to the visual system and are used as a supervisory signal (Fig. 7.1B). Such signals provide information regarding the three actions of put, take and shake and, accordingly, perception may be trained to predict these three actions. Importantly, no explicit signal indicating the number of objects in the scene is available to the visual system at any time.

Using a simple model of this putative mechanism, we find that the image representation that is being learned for classifying actions, simultaneously learns to represent and perceive the first few natural numbers, to place them in the correct order, from zero to one and beyond, as well as estimate the number of objects in the scene.

We use a standard deep learning model of perception [27, 28, 29]: a *feature extraction* stage is followed by a *classifier* (Fig. 7.1). The feature extraction stage maps the image x to an internal representation z, often called an *embedding*. It is implemented by a deep network [28] composed of convolutional layers (CNN) followed by fully connected layers (FCN 1). The classifier, implemented with a simple fully connected network (FCN 2), compares the representations z_t and z_{t+1} of the *before* and *after* images to predict which action took place. Feature extraction and classification are trained jointly by minimizing the prediction error. We find that the embedding dimension makes little difference to the performance of the network (Fig. 7.3). Thus, for ease of visualization, we settled on two dimensions.

We carried out train-test experiments using sequences of synthetic images containing a small number of randomly arranged objects (Fig. 7.2). When training we limited the top number of objects to three (an arbitrary choice), and each pair of subsequent images was consistent with one of the manipulations (put, take, shake). We ran our experiments twice with different object statistics. In the first dataset the objects were identical squares, in the second they had variable size and contrast. In the following we refer to the model trained on the first dataset as *Model A* and the model trained on the second dataset as *Model B*.

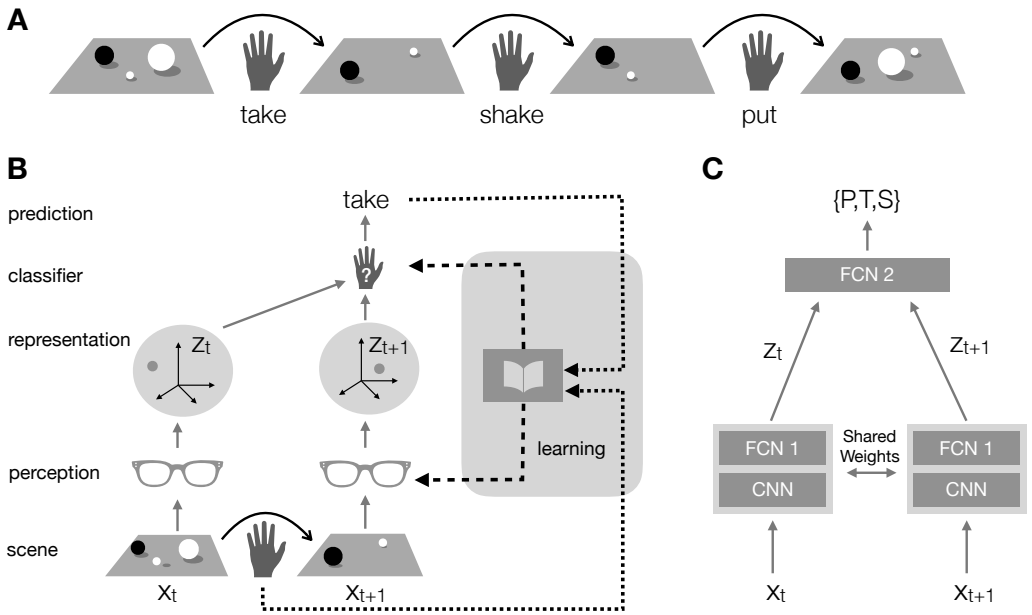


Figure 7.1: Schematics of our model. **(A)** (Left-to-right) A sequence of actions modifies the visual scene over time. **(B)** (Bottom-to-top) The scene changes as a result of manipulation. The images x_t and x_{t+1} of the scene before and after manipulation are mapped by perception into representations z_t and z_{t+1}. These are compared by a classifier to predict which action took place. Learning monitors the error between predicted action and a signal from the motor system representing the actual action, and updates simultaneously the weights of both perception and the classifier to increase prediction accuracy. **(C)** (Bottom-to-top) Our model of perception is a hybrid neural network composed of the concatenation of a convolutional neural network (CNN) with a fully-connected network (FCN 1). The classifier is implemented by a fully connected network (FCN 2) which compares the two representations z_t and z_{t+1}. The two perception networks are actually the same network operating on distinct images and therefore their parameters are identical and learned simultaneously in a *Siamese network* configuration [26]. Details of the models are given in Fig. 7.15.

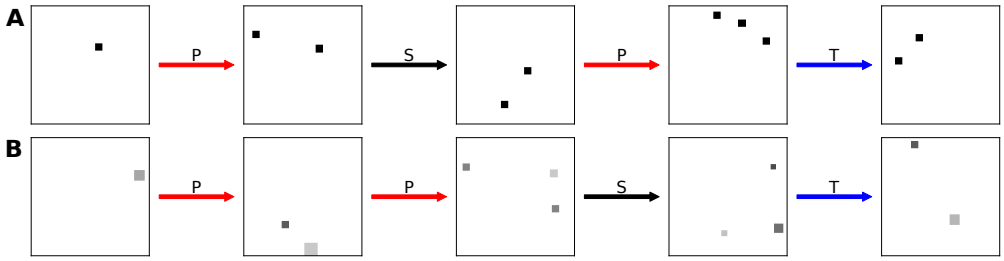


Figure 7.2: Training image sequence samples. We trained our model using sequences of images that were generated by randomly concatenating take (T), put (P) and shake (S) manipulations, while limiting the number of objects to the $\{0 \ldots 3\}$ set (see Methods - Training Sets). We experimented with two different environment/scene statistics: **(A)** Identical objects (15x15 pixel squares) with random position. **(B)** Objects (squares) of variable position, size and contrast. The overall image intensity is a poor predictor of cardinality in this dataset (statistics in Fig. 7.14). Images have been inverted to better highlight objects with low contrast.

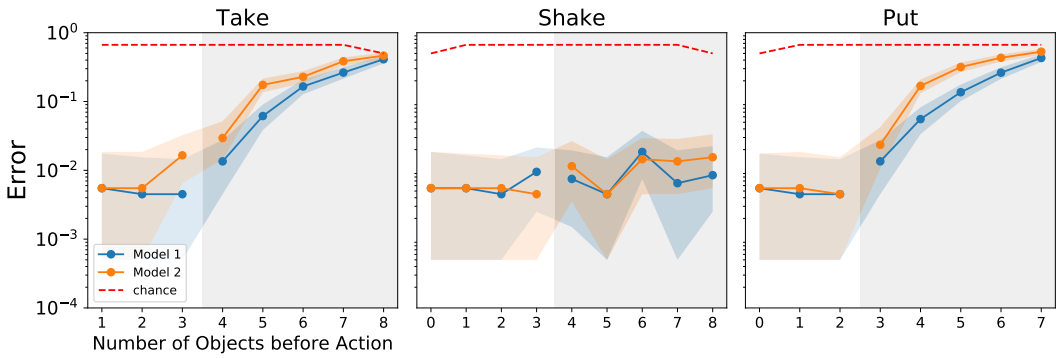


Figure 7.3: Action classification performance. The network accurately classifies actions up to the training limit of three objects, regardless of the statistics of the data (the x axis indicates the number of objects in the scene before the action takes place). Error increases when the number of objects in the test images exceeds the number of objects in the training set. 95% Bayesian confidence intervals are shown by the shaded areas ($272 \leq N \leq 386$). The gray region highlights test cases where the number of objects exceeds the number in the training set. The dashed red line indicates chance level.

7.3 Results

We found that models learn to predict the three actions on a test set of novel image sequences (Fig. 7.3) with an error below 1% on scenes up to three objects (the highest number during training). Performance degrades progressively for higher numbers beyond the training range. Model B's error rate is higher, consistently with the task being harder. Thus, we find that our model learns to predict actions accurately as one would expect from supervised learning. However, there is little ability to generalize the task to scenes containing previously unseen numbers of objects. Inability to generalize is a well-known shortcoming of supervised machine learning and will become relevant later.

When we examined the structure of the embedding we were intrigued to find a number of interesting regularities (Fig. 7.4). First, the images' representations do not spread across the embedding, filling the available dimensions, as is usually the case. Rather, they are arranged along a one-dimensional structure. This trait is very robust to extrapolation: after training (with up to three objects), we computed the embedding of novel images that contained up to thirty objects and found that the line-like structure persisted (Fig. 7.4A). This *embedding line* is also robust with respect to the dimensions of the embedding — we tested from two to 256 and observed it each time (Fig. 7.3).

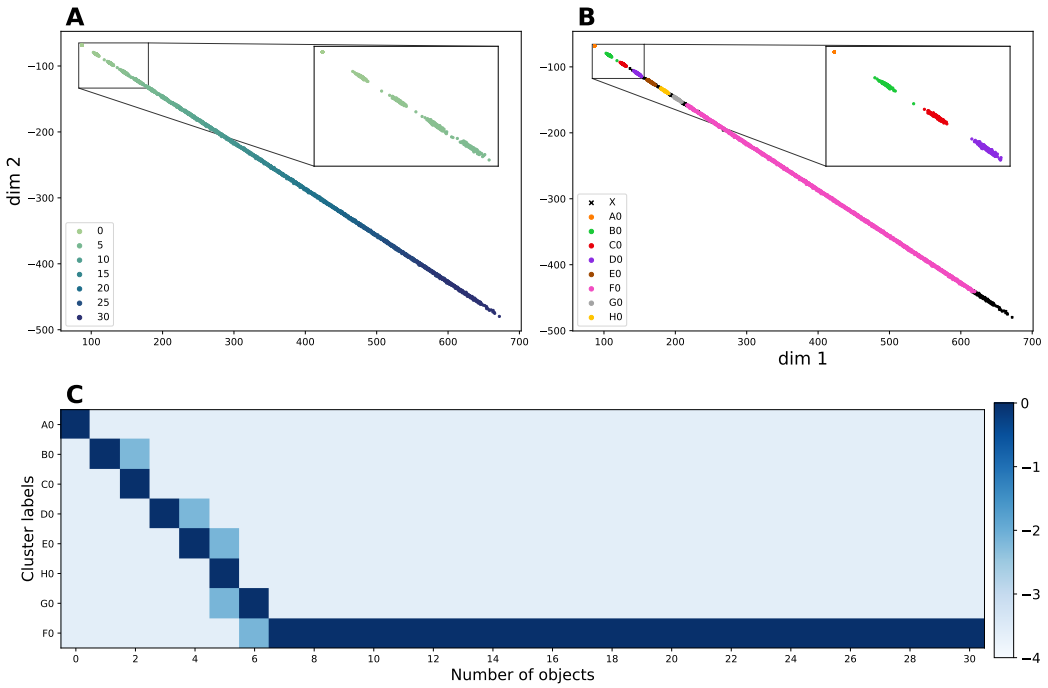


Figure 7.4: The embedding space for Model B. To explore the structure of the embedding space, we generated a dataset with $\{0 \ldots 30\}$ objects, extending the number of objects far beyond the limit of 3 objects in the training task. Each image in the dataset was passed through Model B and the output (the internal representation/embedding) of the image is shown. See Fig. 7.4 for Model A. **(A)** Each dot indicates an image embedding and the embeddings happen to be arranged along a line. The number of objects in each image is color coded. The smooth gradation of the color suggests that the embeddings are arranged monotonically with respect to the number of objects in the corresponding image. The inset shows that the embeddings of the images that contain only a few objects are arranged along the line into "islands." **(B)** We apply an unsupervised clustering algorithm to the embeddings. Each cluster that is discovered is denoted by a specific color. The cluster X, denoted by black crosses, indicates points that the clustering algorithm excluded as outliers. **(C)** The confusion matrix shows that the clusters that are found by the clustering algorithm correspond to numbers. Images containing 0 — 6 objects are neatly separated into individual clusters; after that images are collected into a large group that is not in one-to-one correspondence with the number of objects in the image. The color scale is logarithmic (base 10).

Second, images are arranged almost monotonically along the embedding line according to the number of objects that are present (Fig. 7.4A). Thus, the representation that is developed by the model contains an order. We were curious as to whether the *embedding coordinate*, i.e. the position of an image along the embedding line, may be used to estimate the number of objects in the image. Any one of the features that make up the coordinates of the embedding provides a handy measure for this position, measured as the distance from the beginning of the line – the value of these coordinates may be thought of as the firing rate of specific neurons [30]. We tested this hypothesis both in a relative and in an absolute quantity estimation task. First, we used the embedding coordinate to compare the number of objects in two different images and assess which is larger, and found very good accuracy (Fig. 7.5A). Second, assuming that the system may self-calibrate, e.g. by using the "put" action to estimate a unit of increment, then an absolute measure of quantity may be computed from the embedding coordinate. We tested this idea by computing such a *perceived number* against the actual count of objects in images (Fig. 7.5B). The estimates turn out to be quite accurate, with a slight underestimate that increases as the numbers become larger. Both relative and absolute estimates of quantity were accurate for as many as thirty objects (we did not test beyond this number), which far exceeds the training limit of three. We looked for image properties, other than "number of objects", that might drive the estimate of quantity and we could not find any convincing candidate (see Methods and Fig. 7.2).

Third, image embeddings separate out into distinct "islands" at one end of the embedding line (Fig. 7.4A inset). The brain is known to spontaneously cluster perceptual information [8, 31], and therefore we tested empirically whether this form of unsupervised learning may be sufficient to discover distinct categories of images/scenes from their embedding. We found that unsupervised learning successfully discovers the clusters with very few outliers in both Model A and the more challenging Model B (Fig. 7.4B).

Fourth, the first few clusters discovered by unsupervised learning along the embedding line are in almost perfect one-to-one correspondence with groups of images that share the same number of objects (Figs. 7.4C). Once such distinct *number categories* are discovered, they may be used to classify images. This is because the model maps the images to the embedding, and the unsupervised clustering algorithm can classify points in the embedding into number categories.

Thus, our model learns the ability to carry out instant association of images with a small set of objects with the corresponding number category.

A fifth property of the embedding is that there is a limit to how many distinct number categories are learned. Beyond a certain number of objects one finds large clusters which are no longer number-specific (Fig. 7.4). I.e. our model learns distinct categories for the numbers between zero and eight, and additional larger categories for, say, "more than a few" and for "many."

There is nothing magical in the fact that during training we limited the number of objects to three, our findings did not change significantly when we changed the number of objects that are used in training the action classifier (Fig. 7.6, 7.7), when we restricted the variability of the objects actions (7.7.5), and when "put" and "take" could affect multiple objects at once (7.7.6), i.e. when actions were imprecise. In the last two experiments, we find a small decrease in the separability of clusters in the subitization range (Figs. 7.9, 7.12), such that unsupervised clustering is more sensitive to its free parameter (minimum cluster size).

7.4 Discussion

Our model and experiments demonstrate that a representation of the first few natural numbers, absolute numerosity perception, and subitization may be learned by an agent who is able to carry out simple object manipulations. The training task, action prediction, provides supervision for two-set comparisons. This supervision is limited to scenes with up to 3 objects, and yet the model can successfully carry out relative numerosity estimation on scenes with up to 30 objects. Furthermore, action prediction acts as a pretraining task that gives rise to a representation that can support subitization and absolute numerosity estimation without requiring further supervision.

The two mechanisms of the model, deep learning and unsupervised clustering, are computational abstractions of mechanisms that have been documented in the brain.

A number of predictions are suggested by the regularities in the image representation that emerge from our model.

First, the model discovers the structure underlying the integers. The first few numbers, from zero to six, say, emerge as categories from spontaneous clustering of the embeddings of the corresponding images. Clustered topographic numerosity maps observed in human cortex may be viewed as confirming this prediction [8].

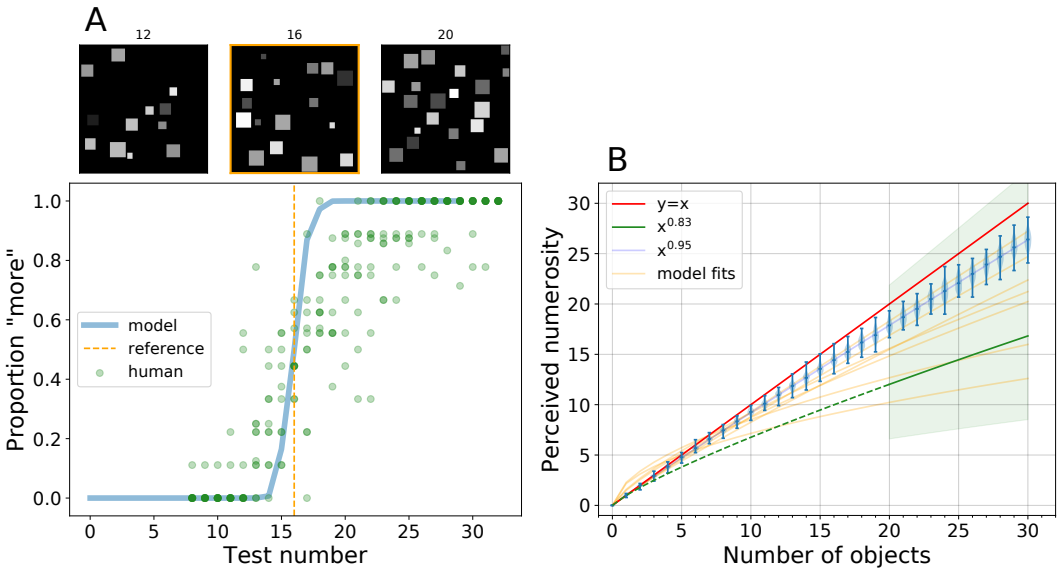


Figure 7.5: Relative and absolute estimation of quantity. (A) Two images may be compared for quantity [32] by computing their embedding and observing their position along our model's embedding line: the image that is furthest along the line is predicted to contain more objects. Here images containing a *test number* of objects (see three examples above containing N=12, 16 and 20 objects) are compared with images containing the *reference* number of objects (vertical orange dashed line, N=16). The number of objects in the test image is plotted along the x axis and the proportion of comparisons that result in a "more" response are plotted on the y-axis (blue line). Human data from 10 subjects [33] is plotted in green. **(B)** The position of images in the embedding space fall along a straight line that starts with 0, and continues monotonically with an increasing number of objects. Thus, the position of an image in the embedding line is an estimate for the number of objects in the scene. Here we demonstrate the outputs of such a model, where we rescale the embedding coordinate (an arbitrary unit) so that one unit of distance matches the distance between the "zero" and the "one" clusters. The y-axis represents such perceived numerosity, which is not necessarily an integer value. The red line indicates perfect prediction. Each violin plot (light blue) indicates the distribution of perceived numerosities for a given ground-truth number of objects. The width of the distributions for the higher counts indicates that perception is subject to errors. There is a slight underestimation bias for higher numbers, consistent with that seen in humans [34, 35]. In fact, Krueger shows that human numerosity judgements (on images with 20 to 400 objects) follow a power function with an exponent of 0.83 ± 0.2. The green line and its shadow depict the range of human numerosity predictions on the same task. The orange lines are power function fits for seven models trained in the same fashion as Model B with different random initializations.

These number categories are naturally ordered by their position on the embedding line, a fundamental property of numbers. The ability to think about numbers may be thought of as a necessary, although not sufficient, step towards counting, addition and subtraction [36, 37]. The dissociation between familiarity with the first few numbers and the ability to count has been observed in hunter-gatherer societies [6] suggesting that these are distinct steps in cognition. In addition, we find that these properties emerge even when the number of objects involved in the action is random, further relaxing the assumptions needed for our model (Sec. 7.7.6).

Second, instant classification of the number of objects in the scene is enabled by the emergence of number categories in the embedding, but it is restricted to the first few integers. This predicts a well-known capability of humans, commonly called *subitization* [18, 38].

Third, a linear structure, which we call *embedding line*, where images are ordered according to quantity, is an emergent representation. This prediction is strongly reminiscent of the *mental number line* which has been postulated in the psychology literature [39, 40, 41, 42]. The embedding line confers to the model the ability to estimate quantities both in relative comparisons and in absolute judgments. The model predicts the ability to carry out relative estimation, absolute estimation, as well as the tendency to slight underestimation in absolute judgments. These predictions are confirmed in the psychophysics literature [32, 34].

Fourth, subitization and numerosity estimation extend far beyond the number of objects used in training. While the model trains itself to classify actions using up to three objects, subitization extends to 5-8 objects and numerosity estimation extends to at least thirty, which is as far as we tested. Extrapolating from the training set is a hallmark of abstraction, which eludes most supervised models [43], yet has been shown in rhesus monkeys [44]. Consensus in the deep networks literature is that models *interpolate* their training set, while here we have a striking example of generalization *beyond* the training set.

Fifth, since in our model manipulation teaches perception, one would predict that children who lack the ability or the drive to manipulate would show retardation in the development of a number sense. A study of children with Developmental Coordination Disorder [45] is consistent with this prediction.

Sixth, our model predicts that adaptation affects estimation, but not subitzation. This is because subitization solely relies on classifiers, which allows for a direct estimate of quantity. Estimation, however, relies on an analog variable, the coordinate along the embedding line, which requires calibration. These predictions are confirmed in the psychophysics literature [32, 34].

Seventh, our model predicts the existence of summation units, which have been documented in the physiology literature [30] and have been postulated in previous models [46]. It does not rule out the simultaneous presence of other codes, such as population codes or labeled-line codes [10].

The model is simple and our clustering method is essentially *parameter-free*. Our observations are robust with respect to large variations in the dimension of the embedding, the number of objects in the training set and the tuning parameters of the clustering algorithm. Yet, the model accounts qualitatively and, to some extent, quantitatively for a disparate set of observations by psychologists, psychophysicists and cognitive scientists.

There is a debate in the literature on whether estimation and subitization are supported by the same mechanisms or separate ones [32, 47]. Our model suggests a solution that supports both arguments: both perceptions rely on a common representation, the embedding. However, the two depend on different mechanisms that take input from this common representation.

It is important to recognize the limitations of our model: it is designed to explore the minimal conditions that are required to learn several cognitive number tasks, and abstracts over the details of a specific implementation in the brain. For instance, we limit the model to vision, while it is known that multiple sensory systems may contribute, including hearing, touch and self-produced actions [48, 49, 50]. Furthermore, the visual system serves multiple tasks, such as face processing, object recognition, and navigation. Thus, it is likely that multiple visual maps are simultaneously learned, and it is possible that our "latent representation" is shared with other visual modalities [14]. Additionally, we postulate that visually-guided manipulation, and hence the ability to detect and locate objects, is learned before numbers. Thus, it would perhaps be more realistic to consider input from an intermediate map where objects have been already detected and located, and are thus represented as "tokens", in visual space, and this would likely make the model's task easier, perhaps closer to Model A than to Model B. However, making this additional assumption is not necessary for our observations.

An interesting question is whether object manipulation, which in our model acts as the supervisory signal during play, may be learned without supervision and before the learner is able to recognize numbers. Our work sheds no light on this question, and simply postulates that this signal is available and, importantly, that the agent is able to discriminate between the three put, take and shake actions. Our model shows that this simple signal on scenes containing a few objects may be bootstrapped to learn about integers, and to perform subitization and numerosity estimation in scenes containing many objects.

Our investigation adds a concrete case study to the discussion on how abstraction may be learned without explicit supervision. While images containing, say, five objects will look very different from each other, our model discovers a common property, i.e. the number of items, which is not immediately available from the brightness distribution or other scene properties. The mechanism driving such abstraction may be interpreted as an implicit *contrastive learning* signal [51], where the *shake* action identifies pairs of images that ought to be considered as similar, while the *put* and *take* actions signal pairs of images that ought to be considered dissimilar, hence the clustering. However, there is a crucial difference between our model and traditional contrastive learning. In contrastive learning, the similarity and dissimilarity training signals are pre-defined for each image pair and the loss is designed to achieve an intended learning goal – to bring the embeddings of similar images together and push the embeddings of dissimilar images apart. In our model, image pairs are associated by an action and the network is free to organize the embeddings in any manner that would be efficient for solving the action prediction task. The learned representation is surprisingly robust – while the primary supervised task, action classification, does not generalize well beyond the three objects used in training, the abstractions of number and quantity extend far beyond it.

7.5 Methods

7.5.1 Network Details

The network we train is a standard deep network [29] composed of two stages. First, a feature extraction network maps the original image of the scene into an embedding space (Fig. 7.1A). Second, a classification network takes the embedding of two sequential images and predicts the action that modified the first into the second (Fig. 7.1B). Given the fact that the classification network takes the embedding of

two distinct images as its input, each computed by identical copies of the feature extraction network, the latter is trained in a Siamese configuration [26].

The feature extraction network is a 9-layer CNN followed by two fully connected layers (details in Fig. 7.15A). The first 3 layers of the feature extraction network are from AlexNet [28] pre-trained on ImageNet [52] and are not updated during training. The remaining four convolutional layers and two fully connected layers are trained in our action prediction task.

The dimension of the output of the final layer is a free parameter (it corresponds to the number of features and to the dimension of the embedding space). In a control experiment we varied this dimension from one to 256, and found little difference in the action classification error rates (Fig. 7.3). We settled for a two-dimensional output for the experiments that are reported here.

The classification network is a two-layer fully connected network that outputs a three-dimensional one-hot-encoding vector indicating a put, take or shake action (details in Fig. 7.15B).

Training procedure

The network was trained with a negative log-likelihood loss (NLL loss) function with a learning rate of 1e-4. The NLL loss calculates error as the -log of the probability of the correct class. Thus, if the probability of the correct class is low (near 0), the error is higher. The network was trained for 30 epochs with 30 mini-batches in each epoch. Each mini-batch was created from a sequence of 180 actions, resulting in 180 image pairs. Thus, the network saw a total of 162,000 unique pairs of images over the course of training.

We tested for reproducibility by training Model B thirty times with different random initializations of the network and different random seeds in our dataset generation algorithm. The embeddings for these reproduced models are shown in Figure 7.7.

Compute

All models were trained on a GeForce GTX TITAN X using PyTorch. Each model takes at most 20 minutes to train. We train a total of 106 models (including supplemental experiments).

7.5.2 Synthetic Dataset Details

Training sets

We carried out experiments using synthetic image sequences where objects were represented by randomly positioned squares. The images were 244x244 pixels (px) in size. Objects were positioned with uniform probability in the image, with the exception that they were not allowed to overlap and a margin of at least 3px clearance between them was imposed. We used two different statistics of object appearance: identical size (15px) and contrast (100%) in the first, and variable size (10px - 30px) and contrast (9.8% - 100%) in the second (Fig. 7.2). Mean image intensity statistics for the two training sets are shown in Figure 7.14. The mean image intensity is highly correlated with the number of objects in the first dataset, while it is ambiguous and thus not very informative in the second. We elaborate on covariates like mean image intensity in the following section.

Each training sequence was generated starting from zero objects, and then selecting a random action (put, take, shake) to generate the next image. The take action is meaningless when the scene contains zero objects and was thus not used there. We also discarded put actions when the objects reached a maximum number. This limit was three for most experiments, but limits of five and eight objects were also explored (Fig. 7.6).

Test sets

In different experiments we allowed up to eight objects per image (Figs. 7.3, 7.6) and thirty objects per image (Figs. 7.4, 7.5A, 7.5B) in order to assess whether the network can generalize to tasks on scenes containing previously unseen numbers of objects. The first test set (up to 8 objects) was generated following the same recipe as the training set. The second test (up to 30 objects) set was generated to have random images with the specified number of objects (without using actions), this test set is guaranteed to be balanced. In section 7.7.1, we use the 30 object test set to estimate covariates for numerosity and analyze their impact on task performance. We were unable to find an image property that would "explain away" the abstraction of number (Fig. 7.2). We note that a principled analysis of the information that is carried out by individual object images is still missing from the literature [53] and this point deserves more attention.

7.5.3 Action classification performance

To visualize how well the model was able to perform the action classification task, we predict actions between pairs of images in our first test set. The error, calculated by comparing the ground truth actions to the predicted actions, is plotted with respect to the number of objects in the visual scene at x_t. 95% Bayesian confidence intervals with a uniform prior were computed for each data point, and a lower bound on the number of samples is provided in the figure captions (Figs. 7.3, 7.3, 7.6).

7.5.4 Interpreting the embedding space

We first explored the structure of the embedding space by visualizing the image embeddings in two dimensions. The points, each one of which corresponds to one image, are not scattered across the embedding. Rather, they are organized into a structure that exhibits five salient features: (a) the images are arranged along a one-dimensional structure, (b) the ordering of the points along the line is (almost) monotonic with respect to the number of objects in the corresponding images, (c) images are separated into groups at one end of the embedding, and these groups are discovered by unsupervised learning, (d) these first few clusters are in one-to-one correspondence with the first few natural numbers, (e) there is a limit to how many number-specific clusters are discovered (Fig. 7.4).

To verify that the clusters can be recovered by unsupervised learning we applied a standard clustering algorithm, and found almost perfect correspondence between the clusters and the first few natural numbers (Fig. 7.4). The clustering algorithm used was the default Python implementation of HDBSCAN [54]. HDBSCAN is a hierarchical, density based clustering algorithm, and we used the euclidean distance as an underlying metric [55]. HDBSCAN has one main free parameter, the minimum cluster size, which was set to 90 in Figure 7.4. All other free parameters were left at their default values. Varying the minimum cluster size between 5 and 95 does not have an effect on the first few clusters, although it does create variation in the number and size of the later clusters. Beyond 95, the algorithm finds only three clusters corresponding to 0, 1 and greater than 1.

One additional structure is not evident from the the embedding and may be recovered from the action classifier: the connections between pairs of clusters. For any pair of images that are related by a manipulation, two computations will be simultaneously carried out; first, the supervised action classifier in the model will classify the action as either P, T, or S (Fig. 7.3) and, at the same time, the unsupervised subitization

classifier (Fig. 7.5A) will assign each image in the pair to the corresponding number-specific cluster. As a result, each pair of images that is related by a P action provides a directed link between a pair of clusters (Fig. 7.5A, red arrows), and following such links one may traverse the sequence of numbers in an ascending order. The T actions provide the same ordering in reverse (blue arrows). Thus, the clusters corresponding to the first few natural numbers are strung together like the beads in a necklace, providing an unambiguous ordering that starts from zero and proceeds through one, two, etc. (Fig. 7.5 A, B). The numbers may be visited both in ascending and descending order. As we pointed out earlier, the same organization may be be obtained more simply by recognizing that the clusters are spontaneously arranged along a line, which also supports the natural ordering of the numbers [41, 56, 57]. However, the connection between the order of the number concepts, and the actions of put and take, will support counting, sum and subtraction.

To estimate whether the embedding structure is approximately one dimensional and linear in higher dimensions we computed the one-dimensional linear approximation to the embedding line, and measured the average distortion of using such approximation for representing the points. More in detail, we first defined a mean-centered embedding matrix with M points and N dimensions, each point corresponding to the embedding of an image. We then computed the best rank 1 approximation to the data matrix by computing its singular value decomposition (SVD) and zeroing all the singular values beyond the first one. If the embedding is near linear, this rank 1 approximation should be quite similar to the original matrix. To quantify the difference between the original matrix and the approximation, we calculated the element-wise residual (the Frobenius norm of the difference between the original matrix and the approximation), then computed the ratio of the Frobenius norm of the residual matrix and the Frobenius norm of the original matrix. The nearer the ratio is to 0, the smaller the residual, and the better the rank 1 approximation. We call this ratio the *linear approximation error*, we show this error compared to some embeddings in Figure 7.7. We computed the embedding for dimensions 8, 16, 64, and 256, (one experiment each) and found ratios of 0.702%, 2.23%, 2.77%, and 2.24%, suggesting that they are close to linear.

7.5.5 Estimating relative quantity

We can use the perceived numerosity to reproduce a common task performed in human psychophysics. Subjects are asked to compare a reference image to a test image and respond in a two-alternative forced choice paradigm with “more” or “less.”

We perform the same task using the magnitude of the embedding as the fiducial signal. The model responds with more if the embedding of the test image has a larger perceived numerosity than the reference image. The psychometric curves generated by our model are presented in Figure 7.5A and match qualitatively the available psychophysics [32, 35].

7.5.6 Estimating absolute quantity

As described above, the clusters are spaced regularly along a line and the points in the embedding are ordered by the number of objects in the corresponding images (Fig. 7.5). We postulate that the number of objects in an image is proportional to the distance of that image's embedding from the embedding of the empty image. Given the linear structure, any one of the embedding features, or their sum, may be used to estimate the position along the embedding line. In order to produce an estimate we use the embedding of the "zero" cluster as the origin. The zero cluster is special, and may be detected as such without supervision, because all its images are identical and thus it collapses to a point. The distance between "zero" and "one", computed as the pairwise distance between points belonging to the corresponding clusters, provides a natural yardstick. This value, also learned without further supervision, can be used as a unit distance to interpret the signal between 0 and n. This estimate of numerosity is shown in Figure 7.5B against the actual number of objects in the image. We draw two conclusions from this plot. First, our unsupervised model allows an estimate of numerosity that is quite accurate, within 10-15% of the actual number of objects. Second, the model produces a systematic underestimate, similar to what is observed psychophysically in human subjects [34].

7.6 Dataset & Code Availability

All data generated or analysed during this study can be found here. The data can also be generated with the code. Code is available here.

7.6.1 Acknowledgements

The California Institute of Technology and the Simons Foundation (Global Brain grant 543025 to PP) generously supported this work. Daniel Israel wrote the code for the jitter and action size supplemental experiments. We are very grateful to a number of colleagues who provided references to the literature and insightful suggestions: Alessandro Achille, Katie Bouman, David Burr, Eli Cole, Jay MacClelland, Markus Meister, Mario Perona, Giovanni Paolini, Stefano Soatto, Alberto Testolin, Kate Stevenson, Doris Tsao, Yisong Yue and two anonymous referees.

Appendix

7.7 Additional Experiments

7.7.1 Controlling for spurious correlates of "number"

Do image properties, other than the abstraction of "object number", drive the quantity estimate of our model? Many potential *confound variables*, such as the count of pixels that are not black, are correlated with object number and might play a role in the model's ability to estimate the number of objects in the scene. If that were the case, one might argue that our model is not learning the abstraction of "number", but rather learning to measure image properties that are correlated with number.

We controlled for this hypothesis by exploiting the natural variability of our test set images. We explored three image properties that correlate with the number of objects and might thus be exploited to estimate the number of objects: (a) overall image brightness, (b) the area of the envelope of the objects in the image, and (c) the total number of pixels that differ from the background. Since objects in training set B vary both in size and in contrast, these three variables are not deterministically related to object number and thus, we reason, confound variable fluctuations ought to affect error rates independently of the number of objects.

We focused on close-call relative estimate tasks (e.g. 16 vs 18 objects), where errors are frequent both for our model and for human subjects, and, while holding the number of objects constant in each of the two scenes being compared, we studied the behavior of error rates as a function of fluctuations in the confound variables. One would expect more errors when comparing image pairs where quantities that typically correlate with the number of objects are anticorrelated in the specific example (Fig. 7.1). Conversely, one would expect lower error rates when the confound variables are positively correlated with number.

In Fig. 7.2 error rates are plotted vs each one of the confound variables when the n. of objects is held constant. We could not find large systematic biases even for extreme variations in the confound variables. In conclusion, we do not find support for the argument that any of the confound variables we studied is implicated significantly in the estimate of quantity.

7.7.2 Interpreting the Embedding Space

Does the dimension of the embedding space influence the action classification error? We wondered what is the effect of this free parameter on the model's performance. We explored this question by training our model repeatedly with the same training

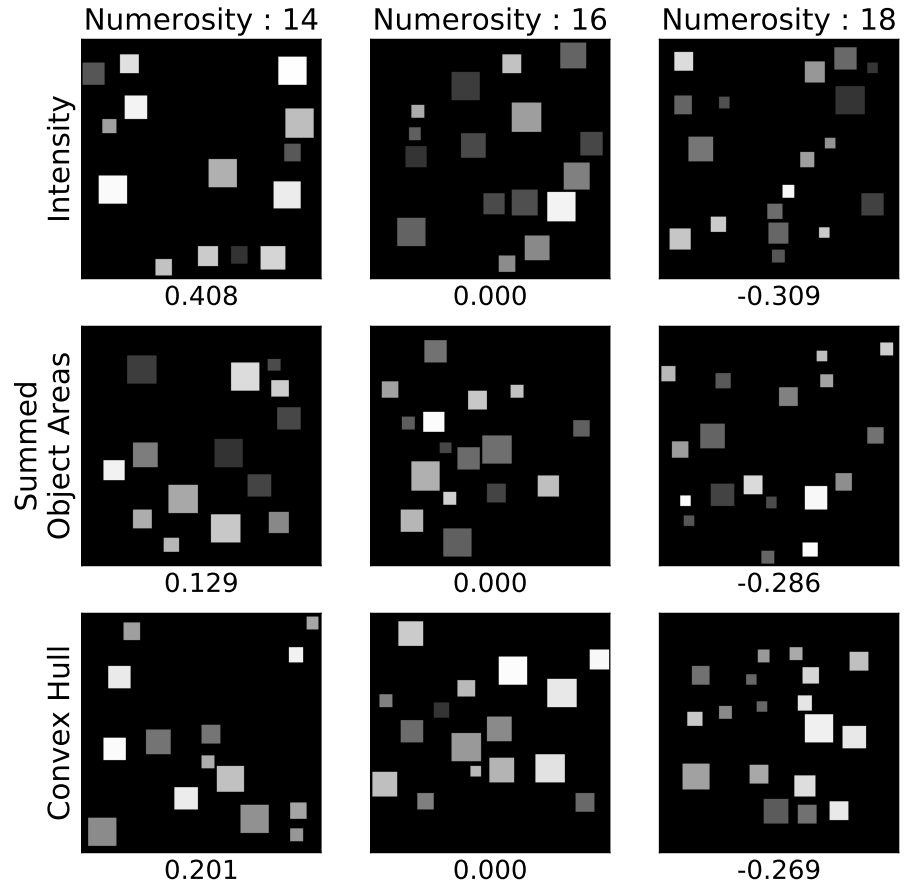


Figure 7.1: Sample images where covariates are anticorrelated with number. We sample images where the three covariates we study (one covariate per row) are anticorrelated with the number of objects. The number below each plot shows the fractional difference from the value of the covariate in the reference image (center column). For example, in the top right, there is a 30.9% decrease in average image intensity when compared to the intensity in the reference image (center column). Another example: in the last row, the scene with 18 objects has a 26.9% smaller convex hull than the corresponding scenes with 14 and 16 objects. For each row, from the lowest numerosity to the highest, the model predicts a perceived numerosity of 12.82, 14.01, and 16.60 (Intensity); 13.21, 14.43, 15.55 (Summed Object Area); 13.22, 15.28, 16.44 (Convex Hull). Thus, our model correctly classifies the relative numerosity for each one of the image pairs that may be formed from each row (our model slightly underestimates numerosity, see Figure 7.5B.) Image pairs formed this way are used in the experiments shown in Figure 7.2, where this manipulation was repeated multiple times and confidence intervals were computed.

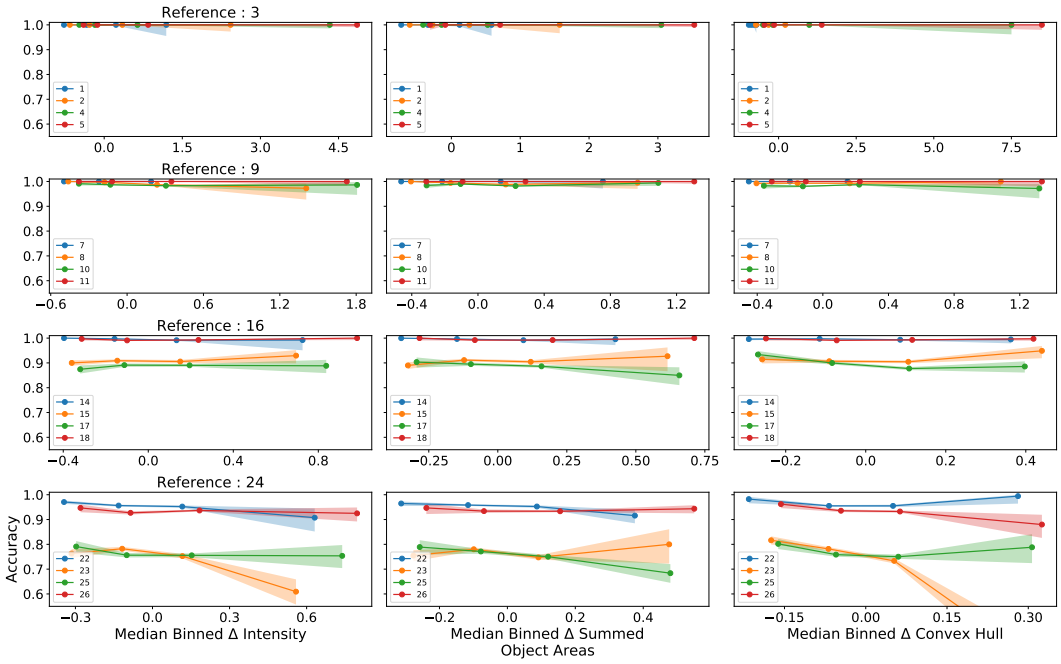


Figure 7.2: Effects of covariates of numerosity. Three covariates of the number of objects in the scene are explored for possible influence on our model's estimate of numerosity. These are average image intensity **(left column)**, the sum of the areas of the objects **(middle column)**, and the area of the objects' convex hull **(right column)**. Each plot shows the error rates in a relative quantity discrimination task like the one in Figure 7.5A. We generate a test set of 4650 test images, 150 images per number of objects. For each plot we chose reference images containing respectively 3, 9, 16 and 24 objects (rows of the figure) and had our model judge relative numerosity w.r. to test images containing a different but similar number of objects (indicated in the legend and associated with colors). Given the stochastic nature of the images, the covariates vary over a wide range for each number of objects (see examples in Fig. 7.1). For each number of objects, we plot the model's error rates (y axis) as a function of the value of the covariate quantity (x axis) which is expressed as fractional difference from the reference image (the values are binned). Shadows display 95% Bayesian confidence intervals($N > 100$, where N is bin size). Horizontal error lines indicate no correlation of numerosity estimation with the covariate quantity. A few lines have slopes that differ slightly from zero indicating a possible correlation. However, some of the slopes indicate a negative correlation (i.e. the better the signal, the higher the error rate). From this evidence it is difficult to conclude that that the model is exploiting anything but "number" to estimate numerosity.

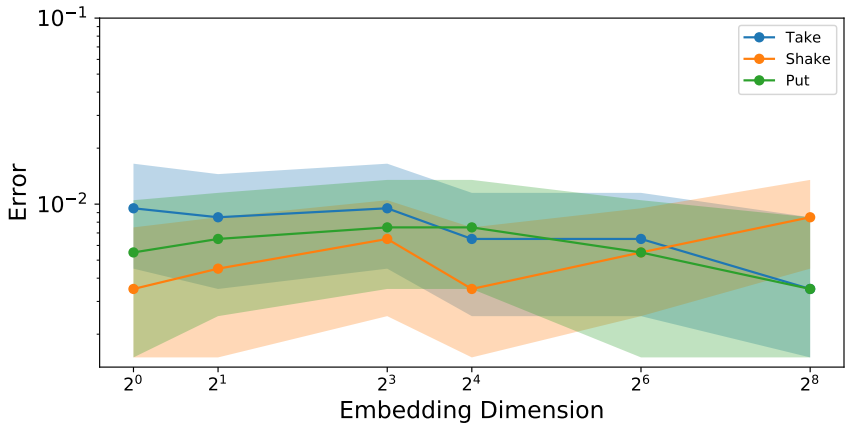


Figure 7.3: Action classification error as a function of embedding dimension. Classification errors for Model B, averaged over the number of items in the scene (0 - 3) are plotted as a function of the dimension of the embedding (a free parameter in our model). Since the effect is minimal we arbitrarily picked a dimension of two for ease of visualization (Figs. 7.4, 7.5). The shadows show 95% Bayesian confidence intervals ($287 \leq N \leq 355$).

images, and varying the dimension of the embedding (Fig. 7.1). Figure 7.3 shows that the effect of the embedding dimension is negligible. This was initially surprising to us. An explanation may be found in the fact that learning produces an embedding that is organized as a line (see Fig 7.4 and Sec. 7.7.4).

Next, we explored the structure of the embedding space in the region where images containing 0-3 objects (the training range) are represented. As discussed in the main text we find that the embedding is organized into clusters (Fig. 7.5 (A,B)). Each cluster contains embeddings of images with the same number of objects. For each pair of images that were generated by a *put* action we drew a red arrow connecting the corresponding embeddings. We used blue arrows for *take* pairs. It is clear from the figure that by following the red arrows one may visit numbers in increasing order: 0-1-2-3 and vice-versa for blue arrows, i.e. the embedding that is produced by our model supports counting up and down.

7.7.3 Varying Training Limit

In our main experiment we trained our model to classify actions with scenes containing from zero to three objects. Does this choice influence qualitatively or quantitatively our observations?

To explore this question we re-trained our model using images that were generated with a total number of three, five and eight objects. As expected, we find that adding

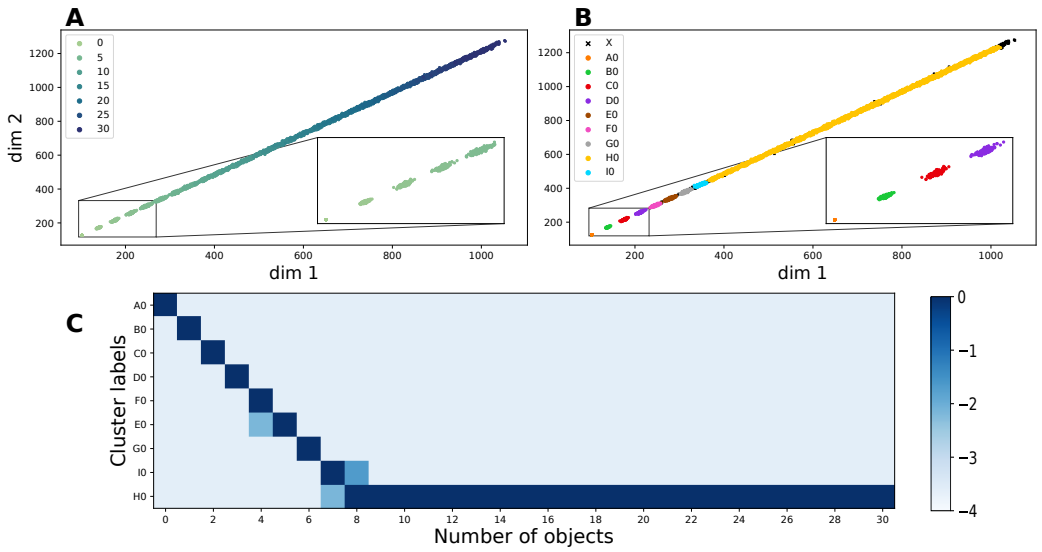


Figure 7.4: The embedding space for Model A. We reproduce Fig. 7.4 for model A. **(A)** Similar to Model B, we observe a monotonically increasing line with well seperated groups at lower quantities. **(B)** We apply an unsupervised clustering algorithm to the embeddings. Each cluster that is discovered is denoted by a specific color. The cluster X, denoted by black crosses, indicates points that the clustering algorithm excluded as outliers. **(C)** The confusion matrix shows that the clusters that are found by the clustering algorithm correspond to numbers. Images containing 0 - 7 objects are neatly separated into individual clusters; after that images are collected into a large group that is not in one-to-one correspondence with the number of objects in the image. The color scale is logarithmic (base 10).

more objects to the training images reduces the action classification error for image pairs with corresponding number of objects (Fig. 7.6). We find no change in the linearity of the embeddings, however, the number of clusters seems to increase with the training limit (Figs. 7.7A,B). This increase in clusters that corresponds with training limit likely explains the improvement in action classification performance.

7.7.4 Reproducibility of the 1D structure of the embedding

The line-like organization of our embedding space is a striking feature. Is this the result of chance, or is this a robust feature that may be reproduced reliably?

We explored this question by repeating our experiments, varying each the random seed used to generate the training images, as well as the random seed used to initialize the model perception network's weights. We show all the embeddings we obtained in Fig. 7.7. Each time we measured how line-like are the embeddings and we report the deviation from an exact line as a percent error below each embedding. We found

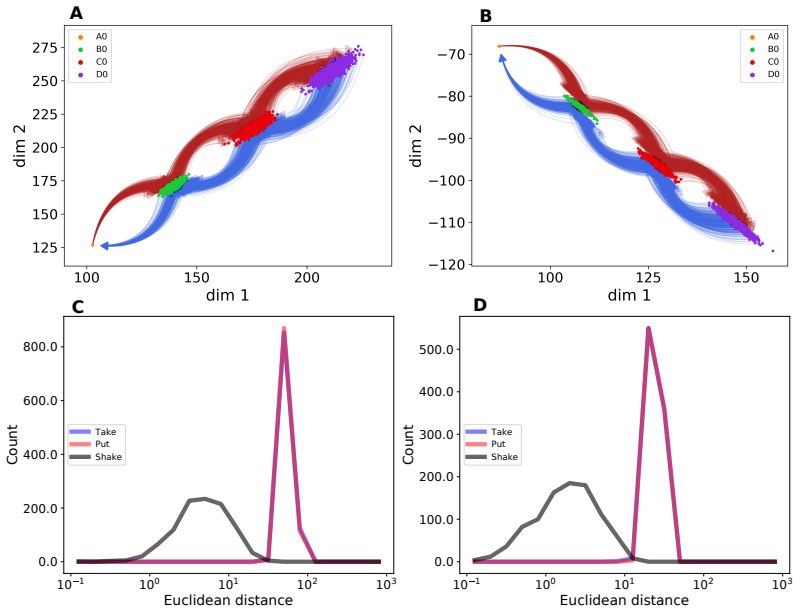


Figure 7.5: Embeddings with topology for Model A and Model B. A close-up look at the embedding space within the training limit. The left side are plots from Model A and the right side from Model B. **(A), (B)** Unsupervised clustering is performed on the embedding space. Each embedding is colored by it's cluster. Each cluster A0 - D0 correspond to images with numerosities 0 - 3. The clusters are well-separated. The "zero" clusters, for both Model A and Model B, are immediately recognizable as they have no variance (orange dot). As numerosity increases, Model A clusters remain well-separated, whereas Model B clusters begin to come closer to each other. We also overlay a topology from the training actions (P), (T), (S). Blue arrows joining a pair of points represent take actions, red arrows represent put actions. Arrows representing shake actions are under the point clouds and are mostly not visible. **(C), (D)** Distances between pairs of points in the embedding space are histogrammed by action. The histograms show the clearly different distribution for shake actions in comparison to take and put actions. Furthermore, the overlap between shake and non-shake actions is smaller for Model A than Model B, explaining the higher performance in action classification for Model A.

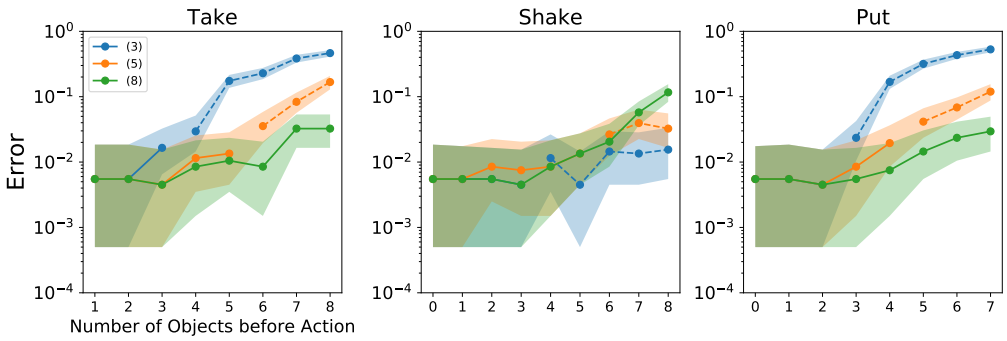


Figure 7.6: Effect of modifying the training limit. (see also Fig. 7.3) In order to explore the effect of the number of objects during training, we trained the network to predict actions using a maximum of 3, 5, or 8 objects with images like those in dataset B (Fig. 7.2B). We tested the network on 8 objects. Each panels shows errors on the training task and are in the same style as Figure 7.3. The line-breaks and dashed lines mark where the training limit ends and the testing region begins, and the legend shows the training limit in parentheses. The shadows provide 95% confidence intervals ($287 \leq N \leq 355$). As expected, the error is lower when the training limit is higher.

that the deviations from a perfect line are very small, and most look perfectly linear with a few exceptions where we see slight kinks in the line.

7.7.5 Restricting Dataset Variability

In our main experiment the arrangement of the objects in the scene varied randomly between *put*, *take* and *shake* actions. The size and contrast were varied as well. This was because we did not wish to presume that the agent (a child) playing with the objects would have to be careful with their motions. Furthermore we did not wish to presume that lighting conditions, and thus image contrast, and object pose, and thus their apparent size, would be preserved during the play session. However, one may suspect that scene randomness could help the model abstract the concept of "number" without being distracted by other factors such as object placement, contrast and size.

We explored the effect of randomness by modifying the process that generates data for Model B. In dataset B, object properties (area, intensity) are completely randomized before and after an action (Fig. 7.2B). We thus constructed a new dataset (Fig. 7.8), where we restricted the randomness before and after an action by reducing the amount of change in an object's area and intensity to a small amount of jitter. However, we still randomize object position, which we find is fundamental to learning a generalizable model of numerosity. We find that even after reducing object variation,

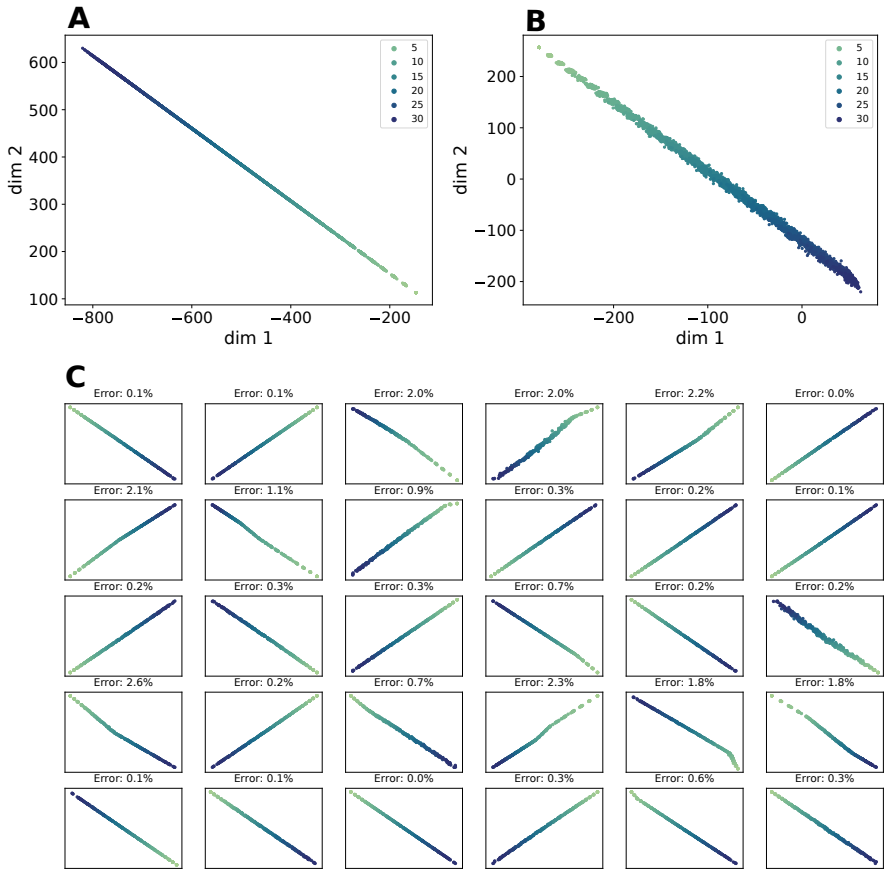


Figure 7.7: Miscellaneous embedding spaces. (see also Fig. 7.4) **(A)** Embedding space for the network trained on dataset B, with up to five objects. **(B)** Embedding space for the network trained on dataset B, with up to eight objects. **(C)** Embedding spaces for 30 different random initializations. We repeated the training procedure 30 times on different random initializations of dataset B, with a training limit of 3 objects. Qualitatively, 21 embedding spaces look like a straight line, six initializations present a slight kink in the line, and three instances either present a large kink or two kinks. The linear approximation error (Methods - Interpreting the Embedding Space) is provided above each subplot and measures the approximate deviation from a purely linear model. An error below 4% predicts an approximately linear embedding line.

the model has learned has the same properties as Model B (Fig. 7.9). However, learning is more sensitive to the initial seed (Fig. 7.10). We refer to this dataset as the *jitter dataset* and model's trained by this dataset as *Jitter Models*.

7.7.6 Imprecise Action Sizes

Will our model learn the abstraction of "number" even when the *put* and *take* actions will place or remove an unpredictable random number of objects?

Figure 7.8: Jitter Datasets. In Jitter Datasets, we restrict the change in size and contrast an object may undergo due to an action. After each action, the size (diagonal) of an object will be allowed to jitter by up to 3 pixels and the contrast by ±0.02% of the maximum contrast. We find that these small perturbations in object representations are sufficient to recreate similar results to those seen with Model B.

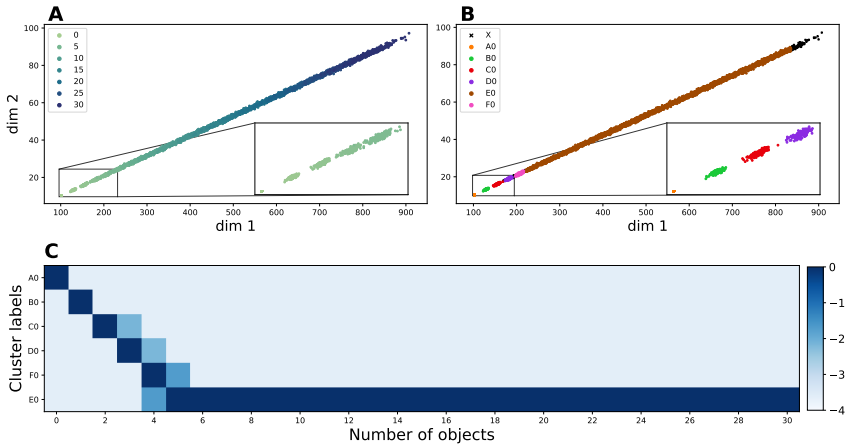


Figure 7.9: Properties of Jitter Models. We find that the important properties of the Model B representation arise with Jitter Models. The model representations are linear, monotonic, with the early numbers easily separable. We set the minimum cluster size to 30 (HDBSCAN), in order to produce the most concise plots. Note the Jitter Model representations are more sensitive to minimum cluster size.

We explored this question by randomizing the number of objects that each action affects in the range 0-3, as opposed to exactly 1 as in the main experiment. We capped the maximum number of objects to 3, like previous experiments. We find that while precise actions help in building distinct clusters in the subitization range, it is not necessary to retain the important properties of the generalizable number line. We refer to this dataset as the *imprecise actions dataset* (Fig. 7.11) and model's trained by this dataset as *Imprecise Action Models*. We find that all the properties of the original model retained (Fig. 7.12) and that the model is reproducible (Fig. 7.13).

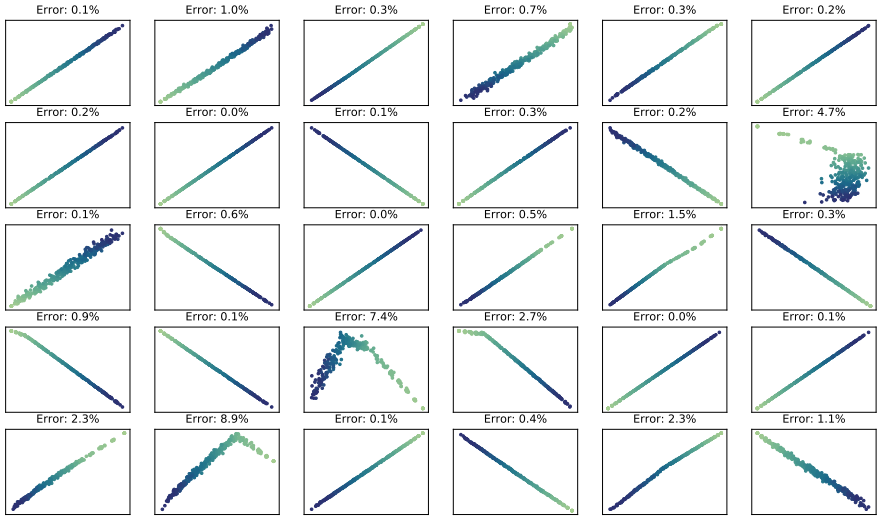


Figure 7.10: Reproducibility of Jitter Models. We vary the initial seed to determine how reproducible the results are. We find model's trained with the jitter dataset learn mostly linear representations, however, certain seeds do result in large kinks. This indicates that visual variability between scenes will help the model learn the abstraction of number.

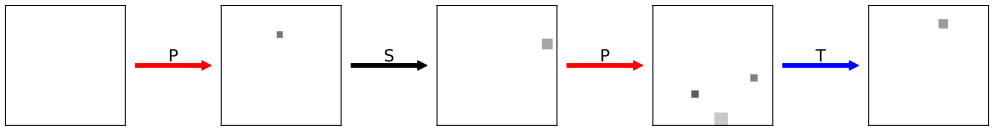


Figure 7.11: Imprecise Action Datasets. In this dataset, we allow the number of objects taken or placed during an action to be 0-3 (limited by the number of objects in the visual scene). The maximum number of objects is still set to 3. This dataset mimics a situation in which the agent is imprecise with their actions and does not always select one object. The object's size and contrast are randomized between actions (like in dataset B).

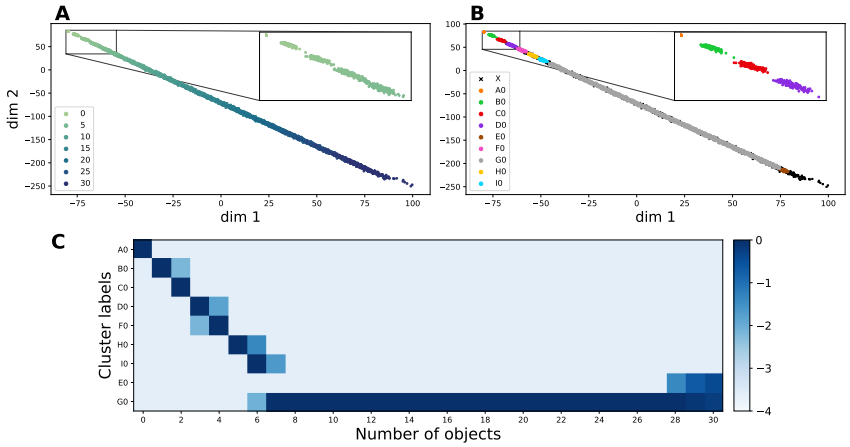


Figure 7.12: Properties of Imprecise Action Models. We find that the important properties of the Model B representation arise with Imprecise Action Models. The model representations are linear, monotonic, with the early numbers easily separable. However, the separability of the early clusters is rougher than with precise action sizes. We set the minimum cluster size to 50 (HDBSCAN), in order to produce the most concise plots. Note the Imprecise Action Model representations are more sensitive to minimum cluster size.

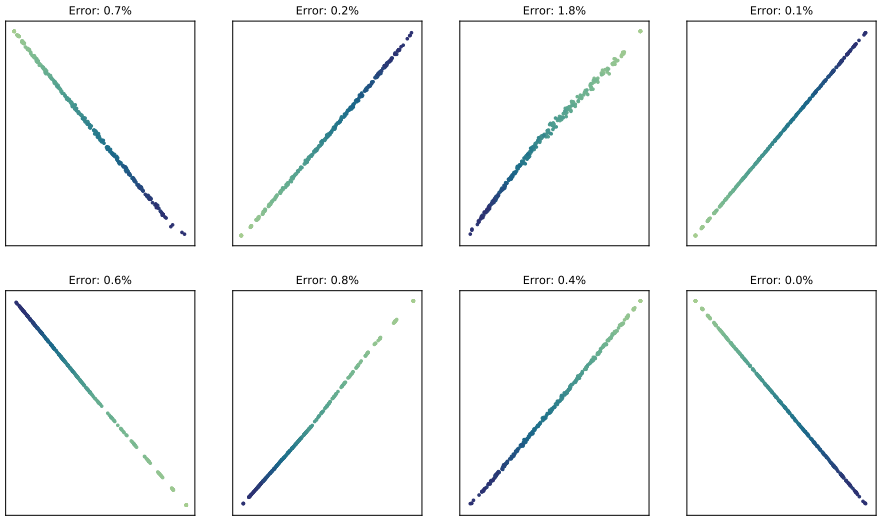


Figure 7.13: Reproducibility of Imprecise Action Models. We vary the initial seed to determine how reproducible the results are. We find model's trained with the imprecise action sizes learn mostly linear representations.

7.8 Dataset Statistics

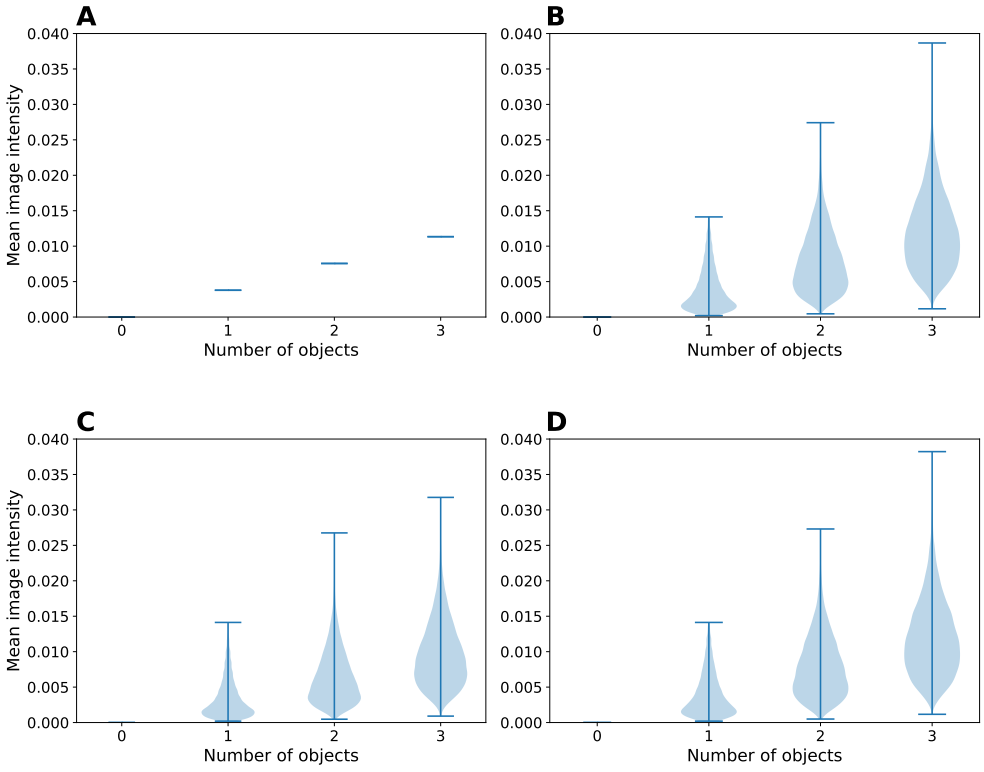


Figure 7.14: Training set statistics. **(A)** In dataset A (Fig. 7.2A) objects have the same size and contrast. Thus, the number of objects predicts the mean image intensity and vice-versa. **(B)** Objects in dataset B (Fig. 7.2B) have variable sizes and variable contrast, thus mean image intensity is not sufficient to predict the number of objects. **(C)** Objects in the jitter datasets (Fig. 7.8) have a restricted, but variable size and contrast. We see the image statistics are similar to that of dataset B, but have a smaller amount of variability. **(D)** Objects in the imprecise actions datasets (Fig. 7.8) have random numbers of objects manipulated in an action. We see the image statistics are effectively the same as that of dataset B.

7.9 Network Details

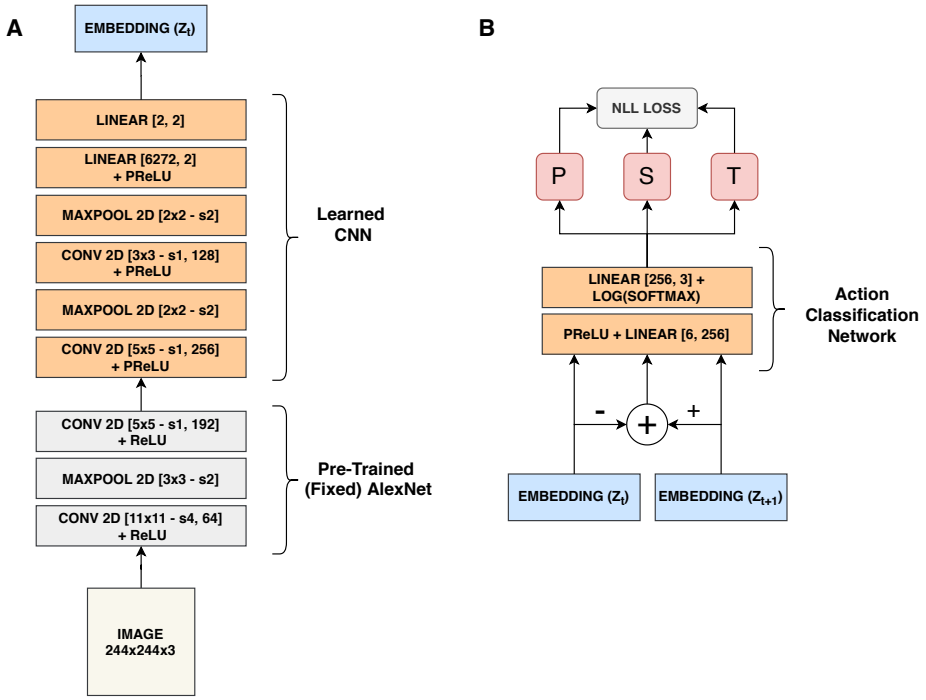


Figure 7.15: Detailed diagram of the network structure.
(A) The feature extraction / embedding network. The gray layers are pre-trained on ImageNet [28, 52] and remain fixed throughout the course of training. The orange layers are randomly seeded and trained simultaneously with the classifier in (B). The details of the layer are described within the brackets. For example, [11x11 - s4, 64] is an 11x11 kernel with a stride of 4 and 64 filters. During a training step, the embedding network accepts an image (x_t) of the visual scene and generates a lower-dimensional feature embedding (z_t) of the visual scene. An action: (P), (T), or (S) modifies the visual scene and the "after" image (x_{t+1}) is passed through the embedding network as well. The outputs of the embedding network, (z_t) and (z_{t+1}) are treated as inputs to the action classification network. The shared embedding network is trained together with the classifier (B), in a Siamese configuration. **(B)** The action classification network is a 2-layer classifier network and is composed of two fully connected layers with a log-softmax transformation on the output. The input is the representation of the visual scene before and after an action is performed. The negative log-likelihood (NLL) loss function is used to train both the action classification network and the embedding network simultaneously. An overview of the entire training paradigm is shown in Figure 7.1.

References

[1] Neehar Kondapaneni and Pietro Perona. "A number sense as an emergent property of the manipulating brain." In: *Scientific Reports* 14.1 (2024), p. 6858. URL: https://www.nature.com/articles/s41598-024-56828-2.

[2] Fei Xu, Elizabeth S Spelke, and Sydney Goddard. "Number sense in human infants." In: *Developmental science* 8.1 (2005), pp. 88–101.

[3] Stanislas Dehaene. *The number sense: How the mind creates mathematics*. OUP USA, 2011.

[4] Pooja Viswanathan and Andreas Nieder. "Neuronal correlates of a visual "sense of number" in primate parietal and prefrontal cortices." In: *Proceedings of the National Academy of Sciences* 110.27 (2013), pp. 11187–11192.

[5] Peter Gordon. "Numerical cognition without words: Evidence from Amazonia." In: *Science* 306.5695 (2004), pp. 496–499.

[6] Pierre Pica et al. "Exact and approximate arithmetic in an Amazonian indigene group." In: *Science* 306.5695 (2004), pp. 499–503.

[7] Stanislas Dehaene et al. "Sources of mathematical thinking: Behavioral and brain-imaging evidence." In: *Science* 284.5416 (1999), pp. 970–974.

[8] Ben M Harvey et al. "Topographic representation of numerosity in the human parietal cortex." In: *Science* 341.6150 (2013), pp. 1123–1126.

[9] Andreas Nieder and Stanislas Dehaene. "Representation of number in the brain." In: *Annual review of neuroscience* 32 (2009), pp. 185–208.

[10] Andreas Nieder. "The neuronal code for number." In: *Nature Reviews Neuroscience* 17.6 (2016), p. 366.

[11] Dmitry Kobylkov et al. "Number neurons in the nidopallium of young domestic chicks." In: *Proceedings of the National Academy of Sciences* 119.32 (2022), e2201039119. DOI: 10.1073/pnas.2201039119. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2201039119. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2201039119.

[12] Ivilin Stoianov and Marco Zorzi. "Emergence of a 'visual number sense' in hierarchical generative models." In: *Nature neuroscience* 15.2 (2012), pp. 194–196.

[13] Marco Zorzi and Alberto Testolin. "An emergentist perspective on the origin of number sense." In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 373.1740 (2018), p. 20170043.

[14] Khaled Nasr, Pooja Viswanathan, and Andreas Nieder. "Number detectors spontaneously emerge in a deep neural network designed for visual object recognition." In: *Science advances* 5.5 (2019), eaav7903.

[15] Gwangsu Kim et al. “Visual number sense in untrained deep neural networks.” In: *Science Advances* 7.1 (2021), eabd6127.

[16] Mengting Fang et al. “Can a recurrent neural network learn to count things?” In: *CogSci*. 2018.

[17] Silvester Sabathiel, James L McClelland, and Trygve Solstad. “Emerging Representations for Counting in a Neural Network Agent Interacting with a Multimodal Environment.” In: *Artificial Life Conference Proceedings*. MIT Press. 2020, pp. 736–743.

[18] W Stanley Jevons. “The power of numerical discrimination.” In: *Nature* 3.67 (1871), pp. 281–282.

[19] Manuela Piazza et al. “Are subitizing and counting implemented as separate or functionally overlapping processes?” In: *Neuroimage* 15.2 (2002), pp. 435–446.

[20] Yosef Singer et al. “Sensory cortex is optimized for prediction of future input.” In: *Elife* 7 (2018), e31557.

[21] David H Hubel and Torsten N Wiesel. “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex.” In: *The Journal of physiology* 160.1 (1962), p. 106.

[22] Rüdiger Von der Heydt, Esther Peterhans, and Gunter Baumgartner. “Illusory contours and cortical neuron responses.” In: *Science* 224.4654 (1984), pp. 1260–1262.

[23] Doris Y Tsao et al. “A cortical region consisting entirely of face-selective cells.” In: *Science* 311.5761 (2006), pp. 670–674.

[24] Doris Y Tsao et al. “Faces and objects in macaque cerebral cortex.” In: *Nature neuroscience* 6.9 (2003), pp. 989–995.

[25] Chou P Hung et al. “Fast readout of object identity from macaque inferior temporal cortex.” In: *Science* 310.5749 (2005), pp. 863–866.

[26] Jane Bromley et al. “Signature verification using a “siamese” time delay neural network.” In: *Advances in neural information processing systems*. 1994, pp. 737–744.

[27] Yann LeCun et al. “Gradient-based learning applied to document recognition.” In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks.” In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[29] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning.” In: *Nature* 521.7553 (2015), pp. 436–444.

[30] Jamie D Roitman, Elizabeth M Brannon, and Michael L Platt. "Monotonic coding of numerosity in macaque lateral intraparietal area." In: *PLoS Biol* 5.8 (2007), e208.

[31] Max Wertheimer. *Laws of organization in perceptual forms.* Kegan Paul, Trench, Trubner & Company, 1938.

[32] David Burr and John Ross. "A visual sense of number." In: *Current biology* 18.6 (2008), pp. 425–428.

[33] Paula A Maldonado Moscoso et al. "Adaptation to hand-tapping affects sensory processing of numerosity directly: evidence from reaction times and confidence." In: *Proceedings of the Royal Society B* 287.1927 (2020), p. 20200801.

[34] Véronique Izard and Stanislas Dehaene. "Calibrating the mental number line." In: *Cognition* 106.3 (2008), pp. 1221–1247.

[35] Lester E Krueger. "Single judgments of numerosity." In: *Perception & Psychophysics* 31.2 (1982), pp. 175–182.

[36] Lisa Feigenson, Stanislas Dehaene, and Elizabeth Spelke. "Core systems of number." In: *Trends in cognitive sciences* 8.7 (2004), pp. 307–314.

[37] Stanislas Dehaene. "Origins of mathematical intuitions: The case of arithmetic." In: *Annals of the New York Academy of Sciences* 1156.1 (2009), pp. 232–259.

[38] David Burr, Giovanni Anobile, and Marco Turi. "Adaptation affects both high and low (subitized) numbers under conditions of high attentional load." In: *Seeing and Perceiving* 24.2 (2011), pp. 141–150.

[39] Frank Restle. "Speed of adding and comparing numbers." In: *Journal of Experimental Psychology* 83.2p1 (1970), p. 274.

[40] Stanislas Dehaene, Serge Bossini, and Pascal Giraux. "The mental representation of parity and number magnitude." In: *Journal of experimental psychology: General* 122.3 (1993), p. 371.

[41] Stanislas Dehaene et al. "Arithmetic and the brain." In: *Current opinion in neurobiology* 14.2 (2004), pp. 218–224.

[42] Rosa Rugani et al. "Number-space mapping in the newborn chick resembles humans' mental number line." In: *Science* 347.6221 (2015), pp. 534–536.

[43] Andrew Trask et al. "Neural arithmetic logic units." In: *Advances in neural information processing systems* 31 (2018).

[44] Jessica F Cantlon and Elizabeth M Brannon. "Shared system for ordering small and large numbers in monkeys and humans." In: *Psychological science* 17.5 (2006), pp. 401–406.

[45] Alice Gomez et al. “Mathematical difficulties in developmental coordination disorder: Symbolic and nonsymbolic number processing.” In: *Research in Developmental Disabilities* 43 (2015), pp. 167–178.

[46] Tom Verguts and Wim Fias. “Representation of number in animals and humans: A neural model.” In: *Journal of cognitive neuroscience* 16.9 (2004), pp. 1493–1504.

[47] Samuel J Cheyette and Steven T Piantadosi. “A unified account of numerosity perception.” In: *Nature Human Behaviour* (2020), pp. 1–8.

[48] Marie Amalric, Isabelle Denghien, and Stanislas Dehaene. “On the role of visual experience in mathematical development: Evidence from blind mathematicians.” In: *Developmental cognitive neuroscience* 30 (2018), pp. 314–323.

[49] Virginie Crollen and Olivier Collignon. “How visual is the number sense? Insights from the blind.” In: *Neuroscience & Biobehavioral Reviews* (2020).

[50] Giovanni Anobile et al. “A Sensorimotor Numerosity System.” In: *Trends in Cognitive Sciences* (2020).

[51] Raia Hadsell, Sumit Chopra, and Yann LeCun. “Dimensionality reduction by learning an invariant mapping.” In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 2. IEEE. 2006, pp. 1735–1742.

[52] Jia Deng et al. “Imagenet: A large-scale hierarchical image database.” In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[53] Alberto Testolin et al. “Visual sense of number vs. sense of magnitude in humans and machines.” In: *Scientific reports* 10.1 (2020), pp. 1–13.

[54] Leland McInnes, John Healy, and Steve Astels. “hdbscan: Hierarchical density based clustering.” In: *Journal of Open Source Software* 2.11 (2017), p. 205. DOI: 10.21105/joss.00205. URL: https://doi.org/10.21105/joss.00205.

[55] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. “Density-based clustering based on hierarchical density estimates.” In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2013, pp. 160–172.

[56] Stanislas Dehaene and Laurent Cohen. “Towards an anatomical and functional model of number processing.” In: *Mathematical cognition* 1.1 (1995), pp. 83–120.

[57] Marco Zorzi, Konstantinos Priftis, and Carlo Umiltà. “Neglect disrupts the mental number line.” In: *Nature* 417.6885 (2002), pp. 138–139.

Part III

Conclusion

Chapter 8

CONCLUSION

The work in this thesis has explored various aspects of aligning and comparing data representations with a few different goals in mind.

8.1 A Step Towards Learning from AI

Learning from AI, aligning human mental representations with those of the model, is driven by the expectation that AI will surpass human capabilities, prompting a need to understand and learn from it. The introduction outlined core challenges in building automated teaching systems to facilitate this process.

In Chapter 2, we introduced an initial approach to tracing visual knowledge as human learners engaged in a visual categorization task. However, there is still much to be done before such systems can reliably estimate student knowledge in automated teaching contexts. A major obstacle is the surprising scarcity of datasets capturing long-term human learning of visual categories. Despite visual categorization being an extremely common task, for instance, naturalists identifying species, medical trainees diagnosing conditions, or geo-guessers locating countries, large-scale datasets tracking visual learning over time are virtually nonexistent. Building such a dataset would benefit not only automated teaching and AI explanation systems, but also the field of visual cognition.

Chapters 3 and 4 investigated methods for comparing two representations of the same input data, with the goal of understanding how differences in representation might explain variations in model behavior. In the context of explainable AI, this introduces a new challenge: can we develop techniques that enable users to anticipate when two models will behave differently on a downstream task? While current methods show promise, further research is needed to build tools that demonstrably achieve this goal. While comparing model representations will be a useful application of such methods, I am most excited about applying these methods to compare human and model representations. In Chapter 5, I explore possible paths towards this goal. For example, as knowledge tracing advances, it may soon be possible to directly contrast human representations with model representations, revealing concepts that models have discovered but students have missed.

These insights could help surface unknowns — echoing how educators diagnose and address conceptual misunderstandings.

8.2 Aligning the Interface Between Models

In Chapter 6, we aligned the text and visual inputs for a powerful visual foundation model. We found that this relatively small change made a dramatic difference in overall model performance. The critical finding came from a qualitative inspection of the interface between the text model that processed the textual inputs and the vision model that generated the outputs. As our methods trend towards complex combinations of pre-trained foundation models, it is critical that we carefully assess if we have successfully aligned the different components of our system to achieve the best performance possible.

8.3 Using Aligned Models to Make Hypotheses about Biological Intelligence

Chapter 7 explores a biologically inspired mechanism for learning numerosity in synthetic visual scenes. While previous work had shown the presence of numerosity sensitive neurons in artificial neural networks, we noticed that these methods implicitly assumed a supervised mechanism for recruiting these neurons to support counting related behaviors. Inspired by the way children interact randomly with objects, we set up a learning game for an artificial neural network in which it was asked to predict changes to the visual scene. This game resulted in a highly organized and human-aligned representation that could reproduce number-related behaviors observed in humans. It is clear that interaction is a key component in learning. As collaborations between AI and robotics increase, AIs will be able to directly interact with the world (embodied AI) and learn from experience. It will be interesting to see if embodied AI representations will naturally align with human representations, or if they will discover new ways to reason about the world.

8.4 Final Thoughts

Going forward, we have an exciting opportunity to learn about the nature of intelligence. AIs have become sufficiently large, complex, and useful that they are objects worthy of study themselves, just as we study biological intelligence. Thankfully, studying artificial intelligence is much more accessible than biological intelligence. We have access to every weight and activation, we can perturb and modify models freely, and we receive feedback on our interventions quickly. By engaging in a loop of analyzing and modifying AI systems, I hope we are able

to develop safer and more powerful models, develop insights about biological intelligence, and discover new patterns in the natural world through the knowledge of super-human AI.