

The Effects of Stimulus Statistics on Representational Similarity in a Model of Mouse Visual Cortex

by

Parsa Torabian

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2024

© Parsa Torabian 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Deep convolutional neural networks have emerged as convincing models of the visual cortex, demonstrating remarkable ability to predict neural activity. However, the specific combination of factors that optimally align these models with biological vision remains an open question. Network architecture, training objectives, and the statistics of training data all likely play a role, but their relative contributions and interactions are not fully understood.

In this study, we focus on the role of training data in shaping the representations learned by deep networks. We investigate how the degree of 'realism' in the training data affects the similarity between network activations and neural recordings from mouse visual cortex. We hypothesised that training on more naturalistic stimuli would lead to greater brain-model similarity, as the visual system has evolved to process the statistics of the natural world. We leveraged the Unity video-game engine to generate custom training datasets with the ability to control for three distinct factors: the realism of the virtual environment, the motion statistics of the simulated agent, and the optics of the modelled eye. Deep networks were trained on datasets generated from all eight permutations of these three experiment variables using a self-supervised learning approach. The trained models were subsequently compared to mouse neural data from the Allen Institute using representational similarity analysis.

Our results reveal that the realism of the virtual environment has a substantial and consistent effect on brain-model similarity. Networks trained on the more realistic meadow-environment exhibited significantly higher similarity to mouse visual cortex across multiple areas. In contrast, the effects of motion statistics and visual optics were more subtle and area-specific. Furthermore, all possible interactions between these three factors were statistically significant, suggesting complex nonlinear relationships.

Acknowledgements

I have had the great fortune of exploring the intersection of neuroscience and deep learning in this thesis, and it is owed to a collection of wonderful people I can never thank enough.

I would like to thank my family for supporting me through the highs and lows of this project. To my wonderful wife, Hedyeh Elahinia, thank you for believing in me at every step. To my gracious parents, Alireza Torabian and Elham Moghadam, thank you for being my unwavering support. To my sister, Neekta Torabian, thank you for being a reliable source of optimism and joy.

I owe a particular debt of gratitude to my supervisor, Dr. Bryan Tripp. You have exemplified the insight, wisdom, patience, and kindness that one can only hope for in a mentor. I have learned an incredible amount, the least of which amounts to neuroscience.

Thank you to my unofficial mentors, who I've had the good fortune of crossing paths and ideating with throughout the duration of this project, Dr. Shahab Bakhtiari, Dr. Graham Taylor, Dr. Blake Richards, Dr. John Mielke, Dr. Michael Buice, and Dr. Stefan Mihales. Your guidance and perspectives have been critical to this work.

To my lab mates at the BRAIN lab at the University of Waterloo: Yinghan Chen, Nolan Dey, Guarav Ramashish, Xueyang Yao, Salman Khan, Sahand Shaghaghi, and Kinjal Patel – thank you for being good friends, kind colleagues, and insightful mentors.

I would like to acknowledge the Digital Resource Alliance of Canada for providing the compute resources for this project and to the excellent support staff for helping resolve technical issues.

Dedication

To my parents, Alireza and Elham, to whom I owe everything.

Table of Contents

Author’s Declaration	ii
Abstract	iii
Acknowledgements	iv
Dedication	v
List of Figures	ix
List of Tables	x
1 Introduction	1
1.1 Neuroscience and Computer Vision	1
1.2 Mice as a model organism	2
1.3 Experiment Overview	3
2 Background	5
2.1 Prior Work Modelling Visual Cortex	5
2.1.1 Neural Network Architectures	5
2.1.2 Loss Function	7
2.1.3 Training Data	8

2.2	Approach in this Study	9
2.2.1	CNN Backbone: MouseNet	9
2.2.2	Training Regime: Dense Predictive Coding	10
2.2.3	Training Data: Unity Environment Recordings	12
2.2.4	Quantifying Similarity	13
2.2.5	Allen Institute Calcium Imaging Dataset	14
3	Methods	17
3.1	Unity Environment	17
3.1.1	Realistic Movement	21
3.1.2	Realistic Vision	21
3.1.3	Realistic Environment	22
3.2	Movement Modelling	23
3.2.1	Two-Stage Model	24
3.2.2	Preprocessing	24
3.2.3	Motion Modelling with VARMA	27
3.3	Training and Analysis Details	29
3.3.1	Adapting MouseNet	29
3.3.2	Quantifying Similarity	29
4	Results	32
4.1	Training has a significant effect on similarity	34
4.2	Realistic environment yields most significant gains	36
4.3	Realistic Motion results highly area-dependant	37
4.4	Realistic vision has little effect	38
4.5	Contrastive loss is not a good proxy for mouse brain	38
5	Discussion	40
5.1	Limitations	40
5.2	Future Directions	41

References	43
APPENDICES	50
A Similarity Scores throughout training	51
A.1 Mouse View	51
A.1.1 Mouse Areas	52
A.1.2 Mouse Regions	55
A.1.3 Mouse Layers	58
A.2 CNN View	60
A.2.1 CNN Regions	60
A.2.2 CNN Layers	64

List of Figures

1.1	High-level experiment outline	3
2.1	Schematic diagram of MouseNet	15
2.2	Overview of Dense Predictive Coding	16
3.1	Screenshot from within Unity Editor	18
3.2	Screenshots from the training data	19
3.3	Two motion conditions	21
3.4	Two environment conditions	22
3.5	Head-mounted motion capture data	23
3.6	Running state duration statistics	24
3.7	Discontinuities in Euler angles	26
3.8	Quaternion representation of rotations	27
3.9	Invalid data due to sensor noise	28
3.10	Motion model feature correlations	31
4.1	Similarity scores per mouse area	33
4.2	Change in similarity scores vs untrained models	35
4.3	Average similarity score per condition	36
4.4	Similarity difference between environment conditions per CNN area	37
4.5	Loss and Top-1 Accuracy Curves	39

List of Tables

3.1	Dataset nomenclature definitions	20
4.1	Global regression results	33
4.2	ANOVA analysis	34
4.3	Regressions per mouse area	35

Chapter 1

Introduction

The human brain is a marvel of systems design – a complex network of billions of neurons and trillions of connections that allow us to perceive the world. Scientists have long been fascinated and perplexed by the mechanisms underlying our perception and its computational underpinnings. Yet it remains one of the most poorly understood systems. Of the brain’s many functions, vision is one of the most crucial and impressive. In a fraction of a second, our brains take in visual information from our eyes and process it to recognize objects, detect motion, discern colour, and construct a coherent picture of the world around us. In this work, we contribute to the large body of impressive neuroscience and computer vision research that aims to elucidate vision.

This thesis is structured into four chapters. Chapter 1 provides a high-level introduction, offering context and outlining the structure of the work. Chapter 2 covers the requisite technical background, including related prior research and the foundational work upon which this thesis builds. Chapter 3 presents a detailed methodology useful for replicating this work. Chapter 4 presents the results and outcomes. Chapter 5 is a discussion of the results, limitations, and directions for future work.

1.1 Neuroscience and Computer Vision

In the past few decades, there have been many parallel developments which have bolstered our understanding of visual processing in the brain. Beginning in 2012, the first computational models of visual processing approaching human-level performance were developed [Krizhevsky et al., 2012]. Since then, the field of computer vision and deep learning has

made great strides in developing models that can process visual information – often with human-like or even superhuman ability. In parallel, neuroscientists have made strides in measurement techniques and experiments on biological neural networks. The number of neurons that scientists have managed to record simultaneously has approximately doubled every seven years, reflecting a trend similar to Moore’s Law [Stevenson and Kording, 2011]. This has led to an explosion of data, enabling researchers to directly measure and map the activity of neurons in the visual system.

These two fields – neuroscience and computer vision – have begun to converge in fascinating ways. On one hand, our growing understanding of the brain’s visual system has inspired new approaches in AI, leading to artificial neural networks whose structure and function are loosely modelled on biological vision. On the other hand, computational models have become powerful tools for modelling and predicting the brain’s responses to visual stimuli. By comparing patterns of activity in Artificial Neural Networks (ANNs) to those measured in animal brains, neuroscientists can gain insight into the computational principles and mechanisms underlying vision.

This thesis explores the intersection of neuroscience and AI in the context of mice and their visual processing. It aims to increase our understanding of what makes artificial neural networks more or less ”brain-like” in their function. This work investigates how the choice of training data – the set of images or videos used to ”teach” a neural network – affects its ability to predict neural responses in mouse visual cortex.

1.2 Mice as a model organism

Mice offer several advantages as a model organism for vision research: they are relatively inexpensive to house and care for, they reproduce and mature quickly, and scientists have developed sophisticated tools for measuring and manipulating neural activity [Grienberger et al., 2022]. In short, it is relatively easy to collect data on mice. By studying mice, we can gain insights about visual processing generally, and perhaps these could lead to insights that apply to other mammals including humans.

1.3 Experiment Overview

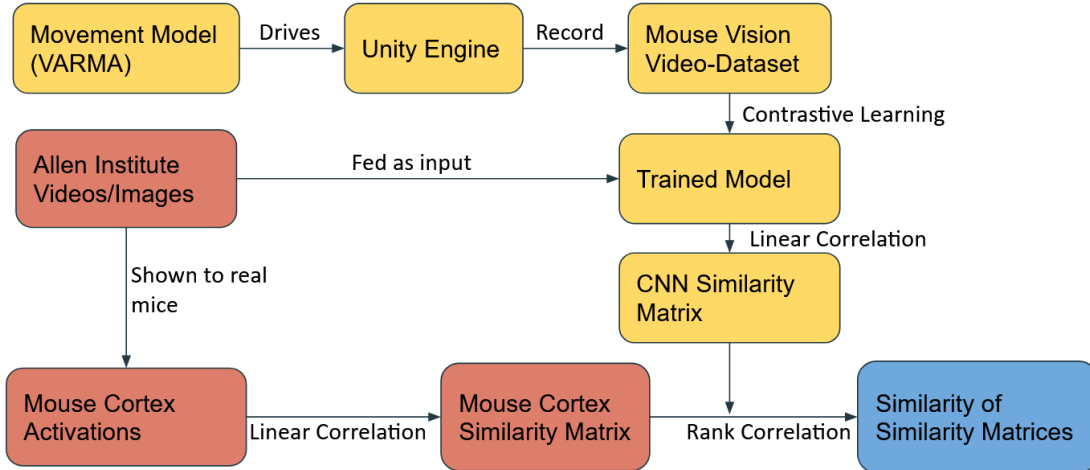


Figure 1.1: A high-level overview of the experimental setup. Training data was recorded from the Unity Video Game Engine in accordance with the desired motion model. The training data was used to train MouseNet models using dense predictive coding, a contrastive learning algorithm. Afterwards, video clips and neural recordings, courtesy of the Allen Institute, were used to compare the representations between trained MouseNet models and mouse visual cortex using representation similarity analysis.

We used the Unity video-game engine [Singh and Kaur, 2022] to generate custom training datasets with varying degrees of naturalism. Some featured a realistic ‘meadow’ environment, while others used a simplistic ‘spaceship’ environment with sharp geometric features. We also varied the motion of the virtual camera used to record the videos, making it more or less similar to the way a mouse moves. Finally, we experimented with two different models of the mouse’s eye optics reflecting our best attempt at replicating mouse visual optics against a simple baseline.

We used artificial neural networks with a structure inspired by the mouse visual system called MouseNet [Shi et al., 2022]. These networks were trained using a learning algorithm called Dense Predictive Coding (DPC) [Han et al., 2019], an algorithm with parallels to known biological learning mechanisms [Price and Gavornik, 2022]. A more extensive overview of the training regime is provided in Section 2.2.2. Essentially, the networks

learned to predict future frames in video sequences, forcing them to learn useful visual representations.

After training MouseNet networks on these different datasets, we compared the patterns of activity in the networks to those previously measured in the mouse visual cortex using two-photon calcium imaging data from the Allen Institute [de Vries et al. \[2019\]](#). We found that using the more realistic meadow environment consistently led to neural networks whose activity was more similar to real mouse brains. The effects of camera motion and eye optic variables were more subtle and depended on the specific area of the visual cortex under question.

Chapter 2

Background

2.1 Prior Work Modelling Visual Cortex

Understanding how the brain processes visual information is a fundamental goal of neuroscience. One approach to this problem is to compare the representations learned by ANNs to those measured in biological neural networks given a common input. By training and identifying ANNs that best explain the brain’s representations, we hope to gain insights into the computational principles that underlie biological vision. Furthermore, these biologically similar ANNs can serve as powerful models for understanding and predicting neural responses in the visual system. This line of study, the modelling of biological neural networks using deep networks, is a large and interdisciplinary field with many analytical lenses. This section contains a non-exhaustive review of past and concurrent work most closely related to this study, with a focus on computer vision and mouse visual cortex. We provide an overview of the key ANN architectures, techniques, and datasets used in the comparative study of ANNs and biological vision.

2.1.1 Neural Network Architectures

The architecture of an ANN plays a crucial role in determining its performance and internal representations. In the context of modelling visual cortex, the choice of architecture is often guided by the performance of classical computer vision tasks (such as classification accuracy), heuristic based (‘visual cortex is wider than it is deep’), or more precise designs based on anatomical and physiological constraints.

Seminal work by [Yamins et al. \[2014\]](#) explored a broad space of Convolutional Neural Network (CNN) architectures, concluding that performance on image classification was correlated with how predictive the CNN’s representations were of primate areas V4 and inferior temporal (IT) cortex. Thus, the classical CNNs designed for image classification remain popular both as models and as baselines. AlexNet [\[Krizhevsky et al., 2012\]](#), was the first of these high performance models and is a popular choice for models of primate visual cortex [\[Khaligh-Razavi and Kriegeskorte, 2014, Cichy et al., 2016, Zhang et al., 2021, Doshi and Konkle, 2022\]](#). In [Margalit et al. \[2023\]](#), the authors used a Residual Model (ResNet) [He et al. \[2015a\]](#) for introducing their Topographic Deep Artificial Neural Network (TDANN) training regime. textcolorblackResNets introduced the skip-connection novelty, where some feedforward connections would skip a set of intermediary layers to allow for better synthesis of upstream and downstream representations. Follow-up work by [Finzi et al. \[2022\]](#) also leverages a ResNet to build a model of ventral, dorsal, and parietal processing streams using human fMRI responses. [Bonnen et al. \[2021\]](#) leverage a network called ‘VGG’ [Simonyan and Zisserman \[2014\]](#) when investigating whether perirhinal cortex, a region in the medial temporal lobe, is involved in perception as well as memory. [Dobs et al. \[2022\]](#) also leverage the VGG architecture in their investigation of specialization, comparing object vs facial detection. VGG models are a simplification of the CNN concept, composed of only simple convolutional layers and normalization layers. However, these models were originally designed and refined to achieve high classification accuracy without much consideration for biological parallels or modelling.

Concurrent work by [Nayebi et al. \[2023\]](#) challenged the notion of using such deep and high-resolution architectures for modelling visual cortex, especially in the context of rodent vision. Rodent visual acuity is low, and its visual cortex is much more ‘shallow’ and ‘broad’ compared to the classical ‘deep’ and ‘narrow’ CNNs. They perform experiments across three classes of networks: “deep”, “normal”, and “shallow”. They note that more shallow models could be key to building more general-purpose models of vision.

Researchers have also begun to experiment with the number of parallel pathways, or streams, in their models. The idea is to mimic the various parallel streams and functional specializations observed in visual cortex. In [Bakhtiari et al. \[2021\]](#), the authors used a two-stream CNN architecture and showed that specialization similiar to the dorsal and ventral pathways of mouse visual cortex can emerge from self-supervised learning. [Nayebi et al. \[2023\]](#) explored three variants, a single-stream pathway, a dual-stream pathway, and a six-stream pathway to model mouse visual cortex, but their results did not show significant variance between these conditions. [\[Margalit et al., 2023\]](#) showed the emergence of functionally different streams with just a single pathway with their TDANN training regime, which explicitly induced local correlations across spatial indices.

Other studies have used architectures that are more directly constrained by biological data. For example, MouseNet [Shi et al., 2022], which we discuss in greater depth below (Section 2.2.1), has an architecture that is based on neuron counts and connectivity patterns measured in the mouse visual system. In Tripp [2019], the author defines a procedure for generating CNN architectures based on neural tract data and electrophysiology studies.

2.1.2 Loss Function

One of the key choices in the design of a deep learning algorithm is the choice of loss function, which is core to defining the task that a model is learning and optimizing against. This has proven a fruitful area of experimentation for researchers looking to model visual cortex.

The most common loss function is the cross-entropy loss typically used for an image classification task. Before self-supervised learning became well-established in the broader deep learning literature, image classification was the dominant training task [Khaligh-Razavi and Kriegeskorte, 2014]. The most common approach is to train models using the ImageNet dataset [Deng et al., 2009] on image classification, such as in Margalit et al. [2023], Bonnen et al. [2021], Zhang et al. [2021], and Doshi and Konkle [2022]. However, supervised objectives with predefined labels are uncommon in nature. Biological neural networks likely developed without the presence of external labels, instead relying on self-supervised or unsupervised learning mechanisms.

Another approach quickly gaining popularity is self-supervised training. Here, the network is trained to minimize a distance metric between representations of semantically similar inputs (positive pairs) and maximize the distance between representations of dissimilar inputs (negative pairs). Positive and negative pairs are not explicitly labelled but are instead derived. For example, an image might retain its semantic content under some distortion function. Thus, an image and its distortion would be automatically set as positive samples, in contrast to another image and its distortions. Such a learning rule, whereby an animal must retain semantic continuity of its percepts, would be more readily available compared to a supervised learning rule, where labelled data explaining a concept would be sparse. This advantage of self-supervised learning was exemplified in Mineault et al. [2021], where the authors showed that self-supervision could be used to model the dorsal stream in mouse cortex, thought to represent visual motion, but not using a supervised approach. In Bakhtiari et al. [2021], the authors used a contrastive loss (Dense Predictive Coding, elaborated further below in Section 2.2.2) to train their two-stream network, where positive pairs were spatial and temporal indices of the next video sequence and

negative pairs were indices from a random other frame. [Nayebi et al. \[2023\]](#) did extensive experiments contrasting self-supervised methods with supervised classification, showing that self-supervised methods are key to building general-purpose models of mouse visual cortex.

There is an array of alternative approaches to defining a loss function, or task, in an attempt to replicate properties of visual cortex. In [Margalit et al. \[2023\]](#), the authors define a loss function which explicitly encourages the representations to be spatially correlated, as it is in cortical sheets. [Keller et al. \[2021\]](#) propose an approach involving variational auto-encoders and a Bayesian Evidence Lower Bound (ELBO) loss as an alternative to TDANN. [Blauch et al. \[2022\]](#) similarly try to induce spatial correlations by explicitly minimizing a 'wiring cost function' which aims to capture the distance of neurons across both breadth and depth of the model.

2.1.3 Training Data

The choice of training data significantly influences the internal representations learned by ANNs. We use the term training data in contrast to analysis data to disambiguate between the data that is used to change model weights with the data used to elicit responses for comparisons to biological neural networks.

One of the most widely used training datasets is ImageNet [[Deng et al., 2009](#)], a large image dataset containing over 14 million images across 1,000 object categories. Many studies have used ImageNet for training models, especially as it relates to comparisons with brain areas associated with object recognition [[Margalit et al., 2023](#), [Bonnen et al., 2021](#), [Zhang et al., 2021](#), [Doshi and Konkle, 2022](#), [Dobs et al., 2022](#)]. Other popular image datasets similarly derive from the object recognition literature, such as CIFAR-10 [[Krizhevsky et al.](#)] and EcoSet [[Mehrer et al., 2021](#)].

In addition to static image datasets, researchers have also explored the use of video datasets to capture the temporal dynamics of visual processing. This has been especially useful in the context of self-supervised learning [[Bakhtiari et al., 2021](#), [Price and Gavornik, 2022](#), [Nayebi et al., 2023](#)]. For instance, [Bakhtiari et al. \[2021\]](#) used the UCF-101 dataset [[Soomro et al., 2012](#)], which consists of videos of human actions, as a part of DPC training. The temporal information in videos provides richer cues and are more aligned with the dynamic nature of visual perception in biological systems. Video datasets also provide opportunities for diverse training regimes, opening the door to more general visual systems beyond object recognition.

After training, ANN weights are typically frozen and used in inference mode. They are presented with common inputs also shown to their biological counterpart to which they are being compared. These datasets, referred to here as analysis datasets, are core to this line of work and have fortunately grown in quality and quantity in recent years. The Natural Scene Dataset (NSD) [Allen et al., 2021] is a large-scale dataset of human visual cortex using high-resolution functional magnetic resonance imaging (fMRI) data. The dataset includes responses from eight participants who viewed a set of over 70,000 natural scene images. Studies such as Margalit et al. [2023] and Finzi et al. [2022] have utilized the NSD to develop and evaluate models of visual representation. Majaj et al. [2015] introduced a dataset of multielectrode recordings from macaque V4 and IT cortex. For mouse visual cortex, the Allen Institute’s 2-photon calcium imaging dataset [de Vries et al., 2019], used extensively in this work, provides neural recordings from the mouse visual cortex in response to natural movies, natural images, static gratings, and white noise.

2.2 Approach in this Study

2.2.1 CNN Backbone: MouseNet

MouseNet [Shi et al., 2022] is a CNN architecture designed to model the mouse visual cortex. The architecture is based on anatomical and physiological data from the mouse visual system, including neuron counts, layer thickness, and connectivity patterns. Notably, MouseNet is a wider, more shallow, and more densely connected model compared to classical CNNs (Figure 2.1). MouseNet kernel sizes and channel counts are also much larger than typical CNNs, as determined by anatomical data. MouseNet only processes images of size 64x64-pixels which corresponds to the resolution of the training data in this study. The network consists of eight core "areas", corresponding to mouse-visual areas LGN, VISp (V1), VISl, VISrl, VISal, VISli, VISpl, and VISpor. With the exception of LGN, each area is divided into cortical layers corresponding to layers L2/3, L4, and L5 (Figure 2.1).

One of the key distinguishing features of MouseNet is its use of anatomically-derived hyper-parameters. For example, the number of channels in each convolutional layer is determined by the estimated neuron counts measured in the corresponding mouse visual areas. Similarly, the kernel sizes and strides are chosen to match the spatial distribution of inputs to neurons in each corresponding area. Among CNNs, MouseNet is the closest architectural approximation of mouse visual cortex.

To evaluate MouseNet, Shi et al. [2022] trained the network on image classification and found that MouseNet achieved 90% accuracy on CIFAR-10 classification and 37% accuracy on downsampled (64x64-pixel) Imagenet classification, while maintaining its biologically inspired architecture. For reference, CIFAR-10 classification accuracy can be close to 99% but typically with one or two orders of magnitude more parameters than MouseNet’s relatively efficient 2.1 million parameters [Antonio et al., 2023, Shi et al., 2022]. These numbers showed that the explicitly biological design of MouseNet does not preclude the functional performance which motivates the use of other CNNs. Similar to this study, Shi et al. [2022] analysed the learned representations in each layer of MouseNet and compared them to neural recordings from the corresponding mouse visual areas. The results showed that MouseNet is a promising step in building representations similar to those found in the mouse visual cortex. The comparable classification accuracy of MouseNet to other commonly used CNNs, along with it being the only CNN explicitly designed to model mouse visual cortex, led to its selection as the backbone architecture for this study.

2.2.2 Training Regime: Dense Predictive Coding

The MouseNet models are trained through DPC [Han et al., 2019]. DPC is a variant of Contrastive Predictive Coding (CPC) [Oord et al., 2019] for video data. These algorithms are self-supervised, meaning they do not use explicitly provided labels to learn representations. CPC algorithms typically learn representations by contrasting positive samples drawn from one distribution against negative samples from another distribution. With DPC, the key idea is to learn latent representations which can be used to correctly classify the next frame (positive sample) in a video sequence against another random frame (negative sample). The intuition is that if the model can identify the correct next frame amongst a set of other frames, then it must have learned a useful semantic representation, both spatially and temporally. The DPC learning algorithm retains the spatial dimensions throughout and requires classification of the correct spatial index as well as the temporal one.

More formally, in DPC an input video dataset is divided into several non-overlapping "clips" $x_1, x_2, \dots, x_n$, each containing five frames. Then five consecutive sequences of clips are used as input to predict the embedding for the next three consecutive video clips (see Figure 2.2). This process involves a non-linear encoder function $f(\cdot)$ which transforms each video clip x_t into a latent representation z_t . In this study, MouseNet serves as $f(\cdot)$. Then, an aggregation function $g(\cdot)$ – a convolutional gated recurrent unit [Tokmakov et al., 2017] – temporally combines t consecutive latent representations into a context representation c_t :

$$z_t = f(x_t)$$

$$c_t = g(z_1, z_2, \dots, z_t)$$

Here, x_t has dimensions $\mathbb{R}^{F \times H \times W \times C}$ as number of frames per clip, height, width, and channels respectively. Similarly, z_t is a feature map with dimensions $\mathbb{R}^{1 \times H' \times W' \times D}$. In this work, D is the total number of L5 output channels in Mousenet. The underlying idea of the predictive task is that if future representations can be generated from c_t , then c_t and the latent representations $z_1, z_2, \dots, z_t$ must capture relevant semantics of the input video. For instance, if a model can see a tree moving right across its field of view through time and can correctly classify seeing a tree further right in its field of view in the future, then it has likely developed a good representation of its environment and relative motion. Next, a predictive function $\phi(\cdot)$ is introduced to forecast the future. Specifically, $\phi(\cdot)$ – a two-layer fully-connected network – takes the context representation c_t as input and predicts the next clip representation $\hat{z}_{t+1}$:

$$\hat{z}_{t+1} = \phi(c_t) = \phi(g(z_1, z_2, \dots, z_t))$$

$$\hat{z}_{t+2} = \phi(c_{t+1}) = \phi(g(z_1, z_2, \dots, z_t, \hat{z}_{t+1}))$$

Here, c_t denotes the context representation from time step 1 to t , and $\hat{z}_{t+1}$ represents the predicted latent representation for sequence at the time step x_{t+1} . At each time step t , the model uses the previously generated embedding $\hat{z}_{t-1}$ as input for generating the next embedding $\hat{z}_t$, enforcing the prediction to depend on all prior observations and predictions.

In DPC, the model classifies one real sample among many noise or negative samples. Here, the loss used for the predictive task encourages the predicted representation $\hat{z}$ to be close to the ground truth z , analogous to a Noise Contrastive Estimation (NCE) loss [Isola et al., 2015].

In the forward pass, both the ground truth z and the predicted $\hat{z}$ representations are computed. For the t -th time step, the loss looks at representations z_t and $\hat{z}_t$ ($z_t, \hat{z}_t \in \mathbb{R}^{H' \times W' \times D}$), preserving spatial information. The feature vector at each spatial location of the feature map is denoted as $z_{i,k} \in \mathbb{R}^D$ and $\hat{z}_{i,k} \in \mathbb{R}^D$, where i is the temporal index and k is the spatial index $k \in \{(1, 1), (1, 2), \dots, (H, W)\}$. The similarity of the predicted and ground-truth positive pair is computed using the dot product $\hat{z}_{i,k}^\top \cdot z_{j,m}$. The objective is to optimize:

$$\mathcal{L} = - \sum_{i,k} \left[\log \left(\frac{\exp(\hat{z}_{i,k}^\top \cdot z_{i,k})}{\sum_{j,m} \exp(\hat{z}_{i,k}^\top \cdot z_{j,m})} \right) \right]$$

This is a cross-entropy loss that distinguishes the positive pair from all other negative pairs. For a predicted feature vector $\hat{z}_{i,k}$, the only positive pair is $(\hat{z}_{i,k}, z_{i,k})$, i.e., the predicted and ground-truth features at the same time step and spatial location. All other pairs $(\hat{z}_{i,k}, z_{j,m})$ where $(i, k) \neq (j, m)$ are negative pairs. The loss encourages higher similarity for the positive pair compared to any negative pairs. Training the network with a larger batch size increases the number of negative pairs.

The use of DPC for training MouseNet is motivated by several factors. Self-supervised learning techniques have been shown to learn useful and transferable representations without the need for explicit labels, making them appropriate for modelling the unsupervised nature of learning in biological systems [Zhuang et al., 2021, Nayeibi et al., 2023]. Furthermore, there are intriguing parallels between CPC and predictive coding, a phenomenon observed where the brain actively predicts future sensory inputs based on past experiences and current context, in biological cortex [Price and Gavornik, 2022].

2.2.3 Training Data: Unity Environment Recordings

One of the core contributions in this work is the introduction of a custom-built Unity environment Singh and Kaur [2022] to generate varying degrees of “realistic” or ethologically relevant training data. Unity is a popular game development engine that has recently gained traction in the machine learning community as a platform for creating virtual environments used in the training and testing of AI agents Singh and Kaur [2022], Juliani et al. [2018]. The aim is to quantify which factors of training data are the most relevant to training models of mouse visual cortex.

Unity offers several advantages over other virtual environment platforms, such as MuJoCo, Unreal Engine, OpenAI Gym, and Meta’s Habitat. Unity provides high-quality 3D graphics. With the exception of Unreal Engine, most virtual environments used in the machine learning community are focused on reinforcement learning, as opposed to building high-quality visuals. The Unreal Engine is most similar to Unity, but at the commencement of this work lacked the depth of run-time integrations with other machine learning software. Thus, Unity stands out for this work and allows the design of complex and realistic virtual worlds that can interface with the rest of our Python-based programming environment.

To set up our Unity environment, we first define the virtual world’s layout, including the geometry of the ground, the placement of objects, and lighting conditions. Unity provides a convenient “High Definition Render Pipeline” (HDRP) to build and package high-quality elements together. We then specify the physical properties of the objects, such as their mass, friction, and colliders, to ensure realistic interactions within the environment. Next, we implement the desired behaviours and dynamics of the scene using Unity’s API, which is implemented in the C# programming language. This allows for programmatic control over the environment’s elements, such as the swaying of grass and leaves, the generation of visual effects, and driving the ‘mouse’ agent within the virtual environment.

For integrating the Unity environment with our python-based pipeline, we utilise the Unity ML-Agents toolkit. The ML-Agents toolkit provides a flexible API for the two-way flow of data between Python and Unity’s event loop [Juliani et al. \[2018\]](#). Our python-based motion model was used to feed movement commands into the Unity engine to drive the movement of the mouse-agent within the virtual environment.

By leveraging the powerful features of Unity, its HDRP, and the ML-Agents toolkit, we are able to create a highly customized virtual environment that can generate a wide variety of video data for training. This flexibility allows us to control the various factors of visual statistics that are explored in the following sections.

2.2.4 Quantifying Similarity

This work leverages Representation Similarity Analysis [[Kriegeskorte et al., 2008](#)] (RSA) as the primary similarity metric and evaluation criteria. RSA is a useful analysis technique as it allows the comparison of representations in fundamentally different spaces provided a common set of inputs. Here, natural movies from the Allen Brain Observatory Open 2-photon Calcium Imaging Dataset [[de Vries et al., 2019](#)] serve as the common input for comparing representations in biological mouse-cortex with the trained models.

RSA is a two-step process. First, a common set of inputs is passed through two different networks, in this case, one of them biological: visual input shown to mice in the Allen Institute experiments; the other artificial: data passed through MouseNet models trained on custom videos via DPC. For each network, the pairwise Pearson correlation of the inputs is calculated, resulting in a symmetrical Similarity Matrix (SM) with a diagonal of ones. The bottom triangle of these SM matrices, excluding the diagonal, is then compared using a Kendall Tau rank correlation to yield a Similarity of Similarity Matrix (SSM).

2.2.5 Allen Institute Calcium Imaging Dataset

We use the open 2-photon calcium imaging dataset from the Allen Brain Observatory. This dataset contains neural recordings from mice who were presented with various visual stimuli. We use data corresponding to presentations of 30 seconds of natural movies (from the 1958 film 'Touch of Evil'), with a frame rate of 30Hz. For each of these frames, the Allen Institute provides pre-processed 'event data', corresponding to normalized neural activations [de Vries et al., 2019].

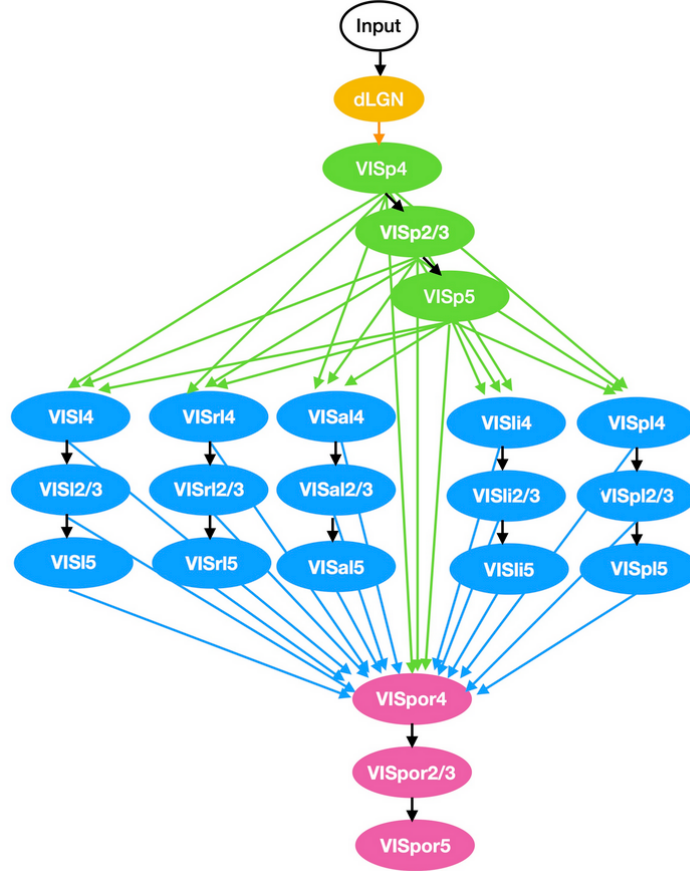


Figure 2.1: Adapted from Shi et al. [2022], showing the schematic diagram of MouseNet layers and connections. MouseNet is wider, has larger kernels, and is more interconnected than typical CNN architectures. This makes MouseNet structurally more similar to mouse visual-cortex. Each area, with the exception of LGN, is divided into Layers L2/3, L4, and L5. MouseNet features are drawn from all L5 regions, rather than from the most downstream layer as is most common with CNNs.

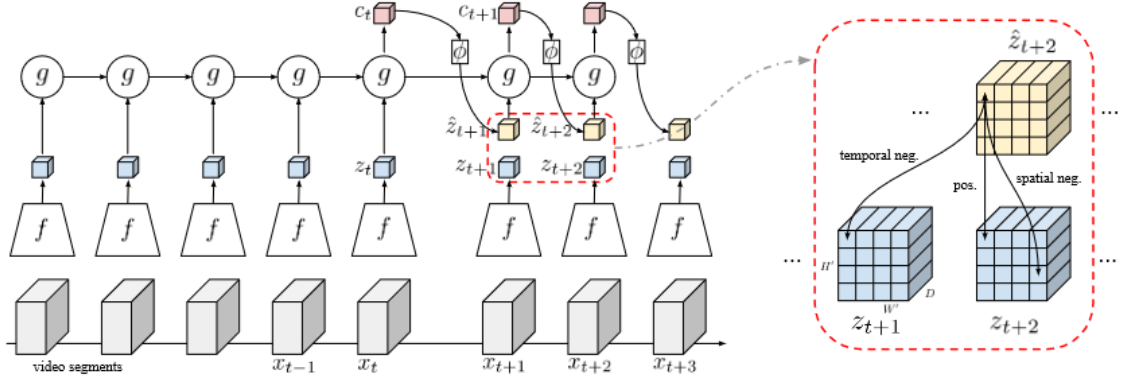


Figure 2.2: Adapted from Han et al. [2019], showing an overview of DPC training. Video clips (x_t) are sequentially passed through a backbone neural network (f), before its outputs (z_t) are fed into an autoregressive model (g). The outputs (c_t) of this autoregressive model are used to generate a prediction for the next feature vector ($\hat{z}_{t+1}$). The model must then classify the correct spatial and temporal indices against all other options in the batch.

Chapter 3

Methods

In this work, we aim to investigate the role of visual statistics in mouse visual processing. Our method follows three primary phases: 1) building a virtual environment within the Unity video game engine, 2) driving the movement of an embodied mouse-agent within the Unity environment, and 3) recording training videos from the perspective of the agent. The output video footage is subsequently used to train an artificial neural network (MouseNet) via a self-supervised contrastive learning algorithm (DPC). The resulting trained neural networks are then analysed using Representational Similarity Analysis (RSA), comparing them against mouse visual cortex. Eight sets of videos are recorded for all the permutations of our three experimental variables, videos with: 'realistic movement', 'realistic environment', and 'realistic vision' (Table 3.1). Here 'realistic' is used as an adjective to denote our best attempt at modelling ethologically relevant features compared to a trivial and simplistic 'non-realistic' alternative condition, outlined below. This work aims to establish which elements of visual statistics are key components of mouse visual processing.

The following sections break down the methodology of the various stages of this work (Figure 1.1), looking at the Unity video game engine itself, the movement modelling that drives the embodied agent, the subsequent self-supervised training of MouseNet models, and lastly the RSA analysis.

3.1 Unity Environment

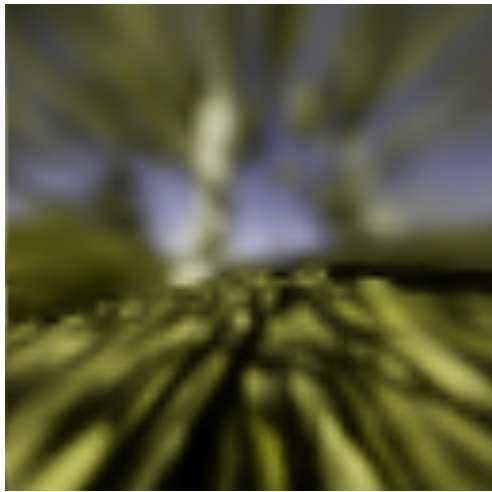
The Unity Engine lies at the centre of the custom datasets we developed. Unity is a video game engine used to render 2D, 3D, Augmented Reality (AR), and Virtual Reality

(VR) environments. In this study, the Unity Engine was used to model a 3D rendering of a grassy meadow, an environment deemed typical and natural to the one in which field mice likely developed. The environment was modelled to contain features such as swaying grass, rocks, dirt paths, trees, apples, cliffs, and clouds. We leveraged Unity Engine’s High Definition Render Pipeline to ensure the best quality of visuals. A “mouse” agent (henceforth referred to as ‘agent’) was placed within this environment and its “vision” was recorded from two cameras. Owing to computation constraints, despite recording both cameras only the output from the right one was used in training for this work.



Figure 3.1: A screenshot of the meadow environment from within the Unity Editor. Note, this is not a sample of any of the recordings/training data.

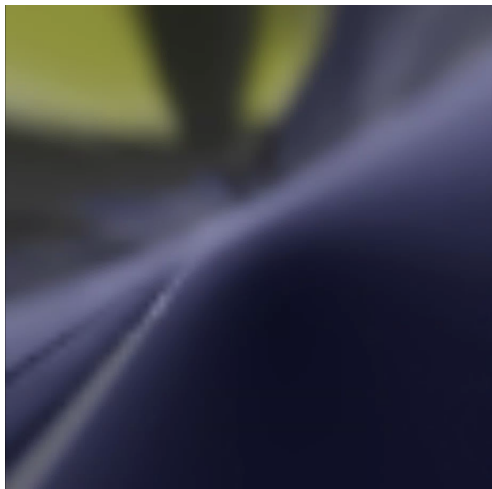
In this work, we toggle three experimental variables which alter the final video outputs resulting in eight different video datasets. These are ‘realistic vision’, ‘realistic movement’, and ‘realistic environment’. For each of the eight conditions, two hours of video were recorded at a 50Hz frame rate. Figure 3.2 show samples from the training data.



(a) Meadow Environment, Mouse Optics



(b) Meadow Environment, pinhole camera



(c) Spaceship environment, mouse optics



(d) Spaceship environment, pinhole camera

Figure 3.2: Screenshots from the training data. These show the meadow environment vs and spaceship environment through both the modelled mouse optics and default pinhole camera.

Dataset	Realistic Environment	Realistic Vision	Realistic Motion
<i>///</i>	No	No	No
<i>//M</i>	No	No	Yes
<i>/M/</i>	No	Yes	No
<i>E//</i>	Yes	No	No
<i>E/M</i>	Yes	No	Yes
<i>EV/</i>	No	Yes	Yes
<i>EV/</i>	Yes	Yes	No
<i>EVM</i>	Yes	Yes	Yes

Table 3.1: Dataset nomenclature definition. There are three experiment variables leading to eight permutations of training datasets. A / represents an absense of the equivalent ‘realistic’ condition. Thus, *///* represents training data recorded from the spaceship environment, using the pinhole camera, and driven by non-realistic motion. Conversely, *EVM* represents recordings from the meadow environment, using a camera parameterized by mouse-optics, and driven by the VARMA motion model

3.1.1 Realistic Movement

It is reasonable to expect that the specific motion a mouse typically undertakes is an important factor in the development and specialisation of mouse visual processing. To validate this hypothesis, we drove the agent with two distinct types of inputs. First, with data generated from a time series model of mouse movement, representing 'realistic' mouse motion. Second, by directing the agent to drive in a perfectly straight line and occasionally turning left or right, representing 'non-realistic' mouse motion. The details of the time series model and its training are outlined in Section 3.2.

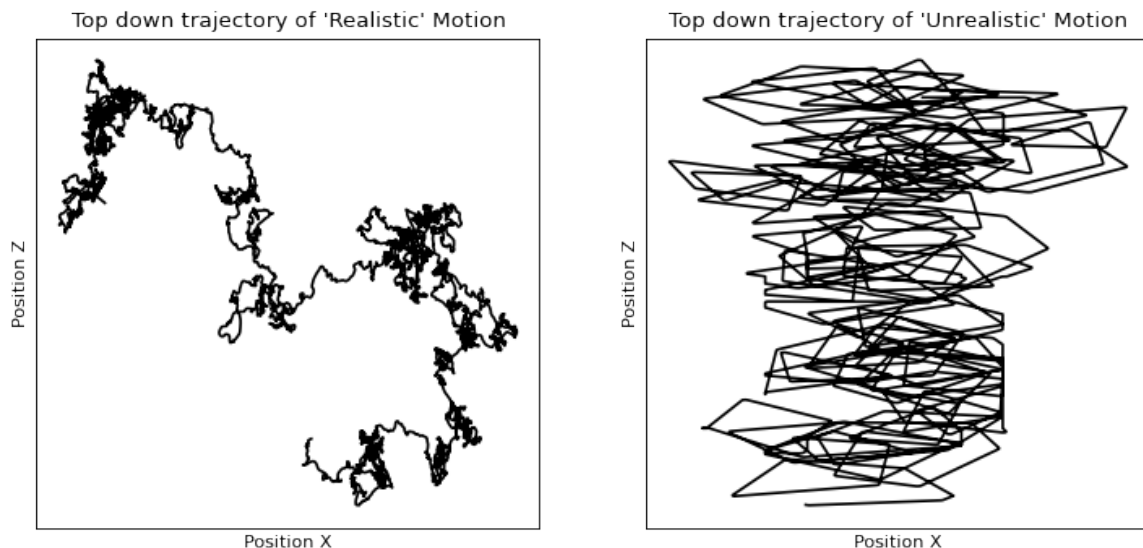


Figure 3.3: Comparison of realistic motion (left) with non-realistic motion (right) from a bird's eyed view.

3.1.2 Realistic Vision

The Unity Engine provides the ability to customise the optical properties of each camera that is capturing video output. As mentioned, two cameras are virtually mounted on the agent. The cameras have horizontal viewing angles of 135 degrees, are separated 1cm apart and oriented 100 degrees away from one another, as in the eyes of a field mouse [Prusky et al., 2006]. To toggle realism in the vision of the agent, we used two sets of optical settings. 'Non-realistic' vision was set to be the default pinhole camera provided by

Unity Engine. 'Realistic vision' was determined by setting the focal length at 2.6mm and a numerical aperture (NA) of 0.49 [Geng et al., 2011]. This means objects 10-15cm away from the agent are in focus, consistent with the near-sightedness of mice. Additionally, the red and green colour channels are averaged together, while the blue channel remains unchanged [Ali and Klyne, 1985].

3.1.3 Realistic Environment

It was also hypothesised that the specific environment mouse vision developed within likely has a large effect on its processing. To test this, a 'non-realistic' environment was generated within the Unity Engine. This second, non-realistic environment was modelled after a spaceship: with grey metallic walls, roofs, and ceilings. It contains hallways punctuated by large open rooms. It also has bright neon light strips across the walls and floor. We developed this environment in contrast with the naturalistic and relatively feature rich meadow environment. We refer to the non-realistic environment as the Spaceship Environment and the realistic version as the Meadow Environment. We leveraged third party asset packages from the Unity Asset Store, namely the NatureManufacture package [NatureManufacture] for meadow and foliage assets, Survival Kit Lite [Rowntec-Assets] for fruit assets, and Sci-Fi Styled Modular Pack [Ebal-Studios] for the spaceship environment.

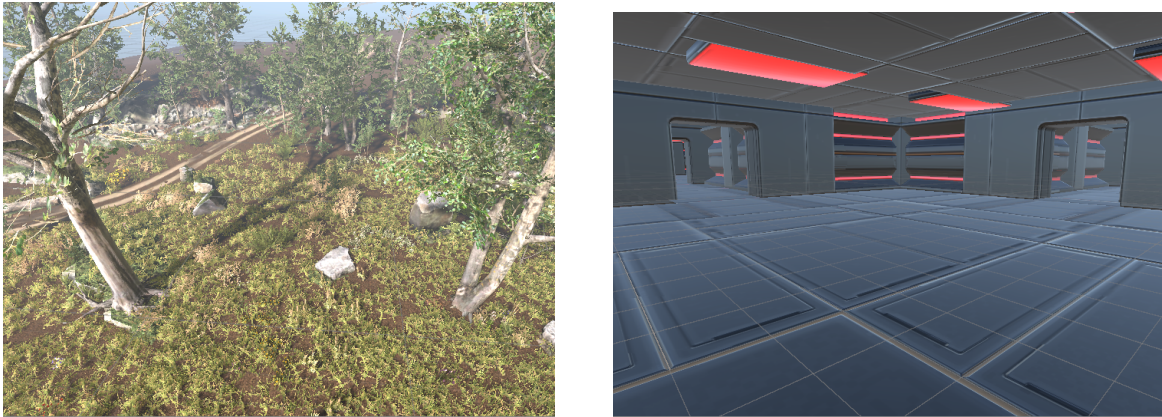


Figure 3.4: Realistic Meadow Environment (left) and the non-realistic Spaceship Environment (right), captured from within the Unity Engine editor. These are not sampled from the training data.

3.2 Movement Modelling

In order to model mouse movements, we used data on mouse movements courtesy of Adrien Peyrache. A mouse was placed in a 50cm x 50cm box while its motion was recorded by a head-mounted motion capture device at 100 Hz. The data captured the position of the mouse within a 50cm by 50cm range and the orientation of the mouse’s head in global Euler xyz coordinates. We received 30 minutes of this recording. Given that our virtual unity environment is much larger than 50cm x 50cm, and the fact that we wished to have more than 30 minutes of data, we needed to generate new and unbounded data which captured the basic statistical properties of mouse motion. We elected to use a Vector Autoregressive Moving Average (VARMA) model, which acts like an infinite impulse response filter with white noise as input. The VARMA model was chosen due to its simplicity and ability to model vectors of motion through time. However, the data was not appropriate in its raw form for modelling. Below we outline the steps involved in creating a time-series model of mouse motion.

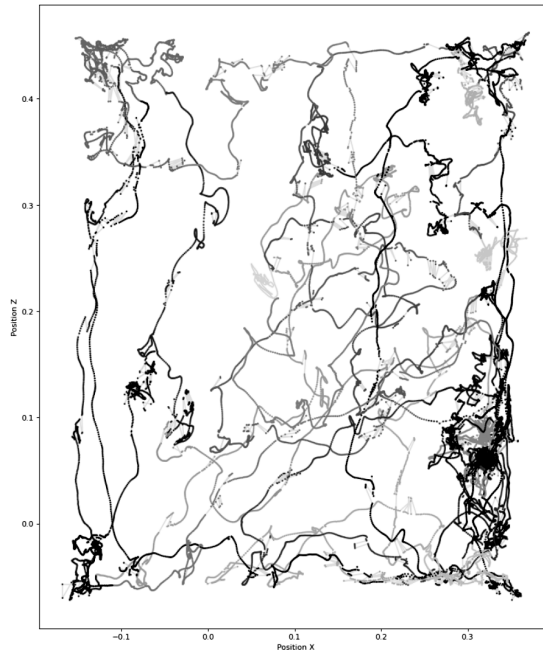


Figure 3.5: Bird’s eyed view of the head-mounted motion capture data, courtesy of Adrien Peyrache.

3.2.1 Two-Stage Model

Qualitative analysis of the motion data revealed two distinct and alternating states: running and stopping. The mouse would alternate between periods of running within the box and periods of standing stationary in place. We determined that such distinct patterns of motion would be difficult to capture in a single model, so we elected to model these two states separately.

To do this, we hand labelled the motion data into periods of running and stopping. To capture the transition between running and stopping, we fit an exponential distribution using maximum likelihood estimation to the state durations for the two states separately. The running model had parameters (offset, decay) = (400, 723) and the stopping model had parameters (offset, decay) = (400, 907) in units of 1/50th a second. An exponential distribution was chosen due to the qualitative similarity to the distribution of state durations and its intuitive interpretation. To switch smoothly between models, we introduced a seven-frame blank interval corresponding to 350ms and linearly interpolated.

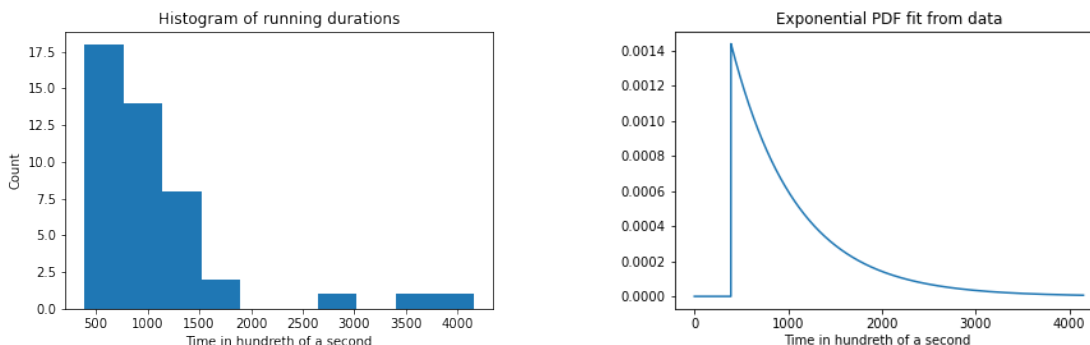


Figure 3.6: Distribution of the duration of running states (left) and the exponential function fit from the data (right). A similar distribution was fit and used to sample the duration of stopped states as well.

3.2.2 Preprocessing

The raw features of the data included three dimensions of position data and three dimensions of orientation data. The position data was naturally limited to the 50cm by 50cm box, and in order to extend the agent's range we decided to model the mouse's velocity rather than its position. This was done on the surface-plane (xz), and the vertical plane

(y) was kept as an absolute position to avoid unrealistic drift off the ground. We also converted the absolute velocities to relative, egocentric velocities. We adopted the convention of forward/backward as the z-axis, left/right as the x-axis, and up/down as the y-axis. This was done in order to better fit the data to the purpose of driving an agent within a virtual environment.

The raw data, however, was not suitable for modelling directly. The orientation data was given in global Euler angles which contain discontinuities, and the data frequently passed through these discontinuities. Discontinuities naturally present a challenge to most models, especially the autoregressive and linear VARMA model. Beyond discontinuities, there were also readily discernible invalid entries in the data, as seen in Figure 3.9. These invalid entries created a distinct noisy pattern, likely due to either sensor imperfections or the fit of the sensor on the mouse’s head.

To account for these data challenges, we designed a two-stage pre-processing pipeline. The rotation coordinates were transformed into quaternions, the data pre-processed and cleaned, and subsequently converted back into the dimensions used for VARMA modelling. The final dimensions used for VARMA modelling were egocentric velocities in the (xz)-plane, position in the vertical y-axis, angular velocity in the y-axis (panning), and rotation in the (xz)-axes (roll and yaw). This setup proves useful on a few accounts. The egocentric velocities yielded unbounded spatial coordinates in the (xz)-plane, while modelling y-position directly retained reasonable vertical bounds. The choice of rotation features neutralized the discontinuities, as natural mouse motion has a limited roll and yaw range, but can undergo the full range of panning. The specifics of the pre-processing are elaborated in the remainder of this section.

In order to preprocess and clean the raw motion data, the rotation coordinates were transformed from global Euler angles into quaternions. Quaternions are a four-dimensional extension to the complex number system and are often used to describe 3-dimensional rotations [Shoemake, 1985]. Quaternions are a useful alternative representation of orientation given they are continuous. Any 3-dimensional orientation can be represented by a location on the four-dimension unit hypersphere. Transforming the rotations to quaternions helped distinguish discontinuities from sensor noise, allow for downsampling, and a smoother change in coordinate space. However, any given orientation which is represented by quaternion-vector 'Q' is also represented by '-Q'. Thus, orientations are mapped twice onto the unit hypersphere. To account for this in our model, we “flip normalise” consecutive quaternions which are more than $\pi/2$ radians apart in the motion data. This resulted in a smooth and continuous representation of orientation throughout time in four dimensions.

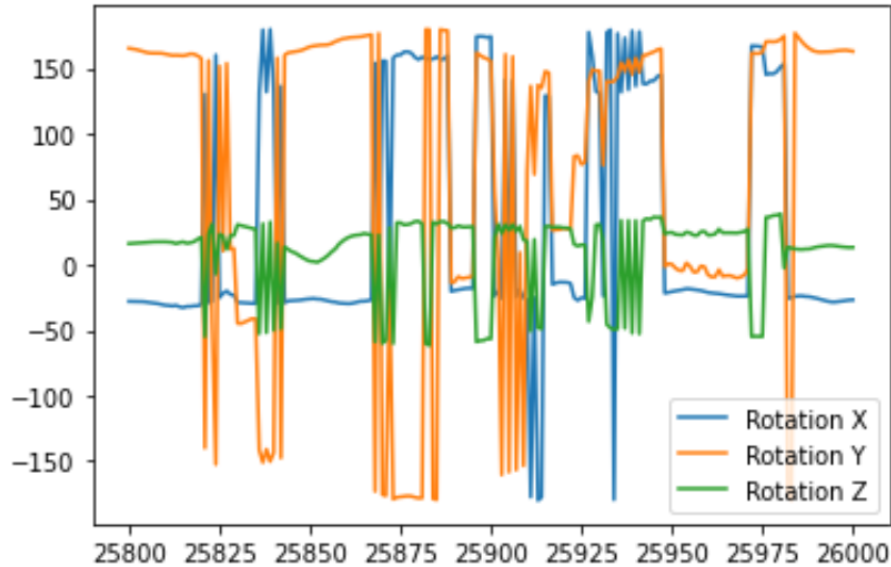


Figure 3.7: Discontinuities owing to Euler representation of angles. Difficulties with modelling these angles motivated the switch to quaternion representations for preprocessing.

We classified “invalid indices” by the hypersphere angle between two consecutive quaternions. If two consecutive angles were more than 0.125 radians apart, it was classified as an “invalid index”. The threshold was chosen as corresponding to a limit of 15 degrees per hundredth of a second, a reasonable upper bound on the rotation speed of a mouse’s head as well as corresponding to a recognizable tail in the distribution of consecutive angles. Invalid indices were nullified and in their place interpolated using a variant of linear interpolation known as spherical linear interpolation (slerp) [Shoemake, 1985], as we were interpolating along the quaternion hypersphere.

After interpolation, the data was low-pass filtered backwards and forwards with a 25Hz 5th-order Butterworth filter, before being downsampled by a factor of two. This brought the data down from a 100 Hz sampling rate down to 50Hz, matching the frame rate of the Unity models and the framerate of the movies being captured. The quaternions were then normalised back onto the unit hypersphere, as the linear filtering introduced slight deviations, which ended up being minimal given the sampling rate. After this final step, units were converted into their final form as egocentric velocities in the (xz)-plane, position in the vertical y-axis, angular velocity in the y-axis (panning), and rotation in the (xz)-axes (roll and yaw).

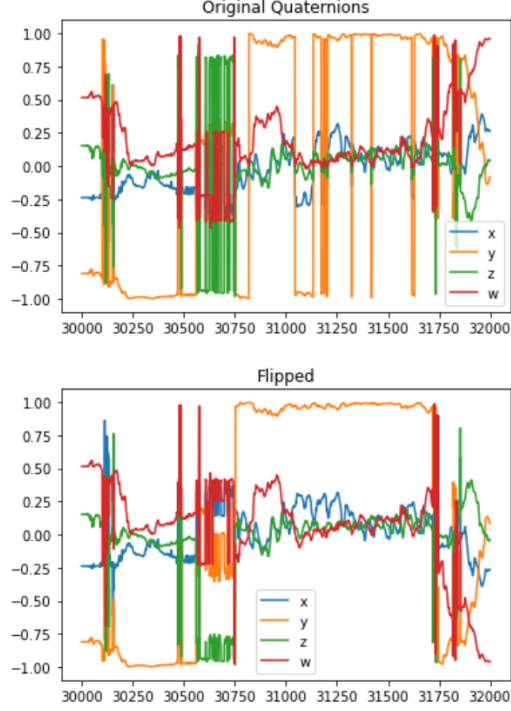


Figure 3.8: Quaternion representation of angles before and after ‘flip normalization’. The ‘flip-normalized’ quaternions also conveniently highlight areas of easily discernable sensor noise, which are subsequently filtered out.

3.2.3 Motion Modelling with VARMA

We chose to model the agent’s motion using a Vector Autoregressive Moving Average (VARMA) model. A VARMA model takes a set of vector inputs (X_{t-i}) and models a vector output (X_t). It does so by linearly regressing past outputs (the autoregressive components κ_i) and a linear combination of moving average inputs (θ_i) in the form of white noise (ϵ). In this way, a VARMA model is an infinite impulse response filter being applied to white noise. VARMA models contain two hyperparameters which determine the size of the model. They are typically denoted as the hyperparameters p and q , representing how far back in time the autoregressive and moving average components extend.

$$X_t = c + \epsilon_t + \sum_{i=1}^p \kappa_i X_{t-i} + \sum_{i=1}^q \theta_i \epsilon_{t-i}$$

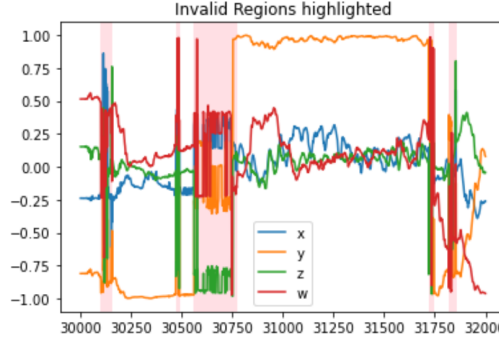


Figure 3.9: Invalid data entries within the raw real-mouse motion data. This is likely due to sensor imperfections or the fit of the sensor on the mouse’s head.

The VARMA models were trained using the Python-based statsmodels package [Seabold and Perktold, 2010]. A VARMAX model was fitted with default settings (50 iterations by maximum likelihood via Kalman filter).

To determine the hyperparameters of our model, we first did a sparse search of the parameters and qualitatively analysed which yielded realistic looking outputs. This was a simple process as models which failed often did so drastically, generating very implausible mouse motion. We determined that the model orders (p, q) yielding plausible motion were typically in the single digit range. At this point, we elected to determine our model quantitatively using the Akaike Information Criteria (AIC). The AIC is a commonly used criterion for comparing time series models with one another [Stoica and Selen, 2004]. It is defined as $AIC = 2k - \ln(L)$, where K represents the number of parameters in the model and L is the maximum value of the likelihood function. The AIC balances minimising the number of parameters with maximising the likelihood function. Thus, a lower score is better. We subsequently did a grid search of hyperparameters (p, q) and found that $(p, q) = (7, 2)$ is the best for the running state and $(p, q) = (5, 5)$ is best for the stopping state.

To validate the data generated by our model, we performed four checks. First, we looked at the cross-correlation of features between the reference training data and the generated data (Figure 3.10). Second, we looked at the individual and pairwise distribution of features. Finally, we ran the generated data through the Unity Engine and qualitatively assessed whether the model-generated inputs were both A) plausible and B) similar to the training data (with the exception of being unbounded).

3.3 Training and Analysis Details

Once we had the video datasets recorded from the Unity Engine, the next step was to use those datasets to train the MouseNet models. Five models each with a different random seed were trained using gradient descent for each of the eight dataset conditions for 100 epochs. Batch sizes of 32 were used for a total of 281 mini-batches per epoch. An Adam optimizer [Kingma and Ba, 2014] was used with a learning rate of 0.001 and a weight decay of 10^{-5} .

For DPC training we divided training videos into sequences of eight five-frame clips (see Section 2.2.2 for more details). Sequences were organized into random mini-batches of size 32. The batch size was limited by 32GB memory on Nvidia V100 graphical processing units (GPUs). The model used the first five clips (25 frames) in each sequence as context, and was trained to classify each of the three subsequent five-frame clips from among negative examples in the same mini-batch. Note that the negative samples were drawn from other clips in the batch, including the two remaining within-sequence clips.

We retained the same recurrent head as in [Han et al., 2019]. We use a convolutional GRU unit with an input size of 144, with the same hidden size, a kernel size of one, one hidden layer, and a dropout ratio of 0.1.

3.3.1 Adapting MouseNet

Given that MouseNet was designed to handle single images, it has been adapted in this work to be used as the backbone for a video-based DPC training regime. For every video block of single image frames, the frames are fed individually through MouseNet and their features concatenated before being used in downstream processing. Unlike the common approach to feature extraction in CNNs, whereby the last and final layer is used as the feature extractor, MouseNet features are extracted from layer 5 (L5) of each area. The extracted features at each area are of differing spatial dimensions and are thus downsampled into 4x4 grids using average-pooling prior to concatenation. All Mousenet models were initialized using kaiming initialization [He et al., 2015b] prior to training.

3.3.2 Quantifying Similarity

For calculating neural SM, we first downsample event data by a factor of five, which is also the clip size used in DPC training. Thus, both neural recordings and CNN activations

have aligned similarity matrices with dimensions 180x180 (900 total 'Touch of Evil' frames divided by five).

Noise ceilings were calculated by taking the similarity scores between two biological-mice samples. The idea is that the highest measure of similarity between an artificial neural network and mouse-cortex should be capped by the similarity scores between two biological mice. The noise ceiling was determined by taking the median similarity score of 100 pairs of mice. We selected for the CEX2-CRE2 mice, which was the transgenic line with the highest noise ceilings. This selection resulted in recordings from five distinct areas of the mouse visual cortex: VISp, VISl, VISal, VISpm, and VISam. Notably, we did not include VISrl in this analysis due to its very low noise ceiling. These recordings come from cortical depths between 175-250 μm , corresponding to cortical layer 2/3, and 265-250 μm , corresponding to cortical layer 4. All similarity scores presented in this work are normalized by their respective noise ceilings, and all noise ceilings have a value of at least 0.3.

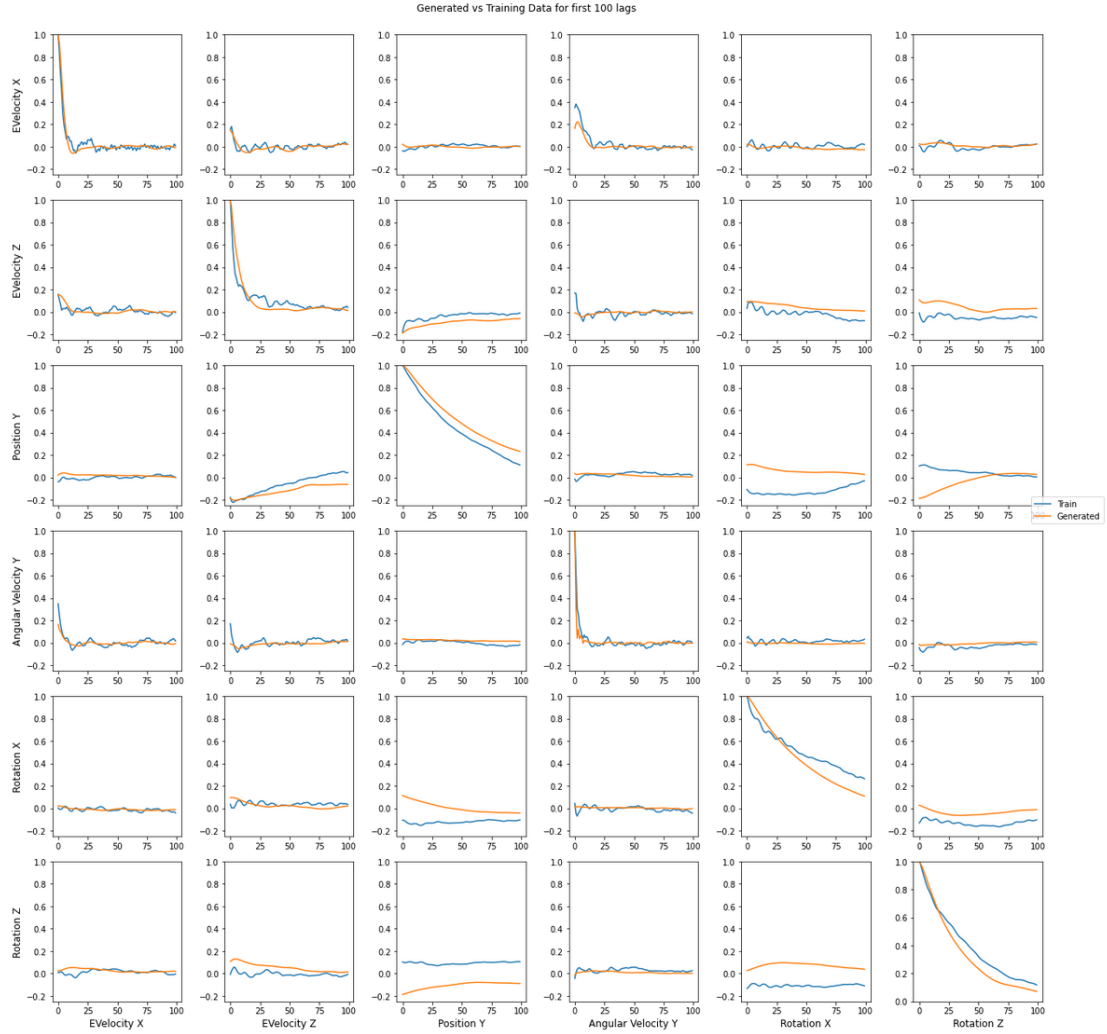


Figure 3.10: Auto and cross-correlations for the running model for the first 100 lags across both the training data (blue) and the generated data (orange).

Chapter 4

Results

In all analyses the default comparison will be to models trained for 100 epochs on the /// dataset condition – also known as the "base condition". The untrained condition was not used as the base condition for regressions as it consistently resulted in highly skewed residuals. However, Figure 4.1 includes similarity scores for untrained models and Section 4.1 explores additional comparisons against a set of untrained models. While the analysis explores the end-state after 100 epochs, Appendix A shows the similarity scores throughout training.

For each of the eight dataset conditions, five MouseNet models were trained, each with different random seeds. For each trained model, similarity scores were calculated for each combination of mouse-area to CNN-area. A regression over all similarity scores, taking mouse-areas as features, is shown in Table 4.1. In this regression, there are two sets of one-hot encoded variables: one specifying the mouse area (columns of the similarity matrix) and the other specifying the experimental condition. The coefficients of the former represent the similarity scores of the model trained with base condition. While this provides a good high level summary of the training results, the impact of training is not uniform across mouse areas, as can be seen in Figure 4.1.

All conditions with a realistic environment enabled have a significant effect with respect to the base condition. The dataset with only realistic motion enabled (/M) also shows a significant, although small, effect above the base condition.

An ANOVA analysis is shown in Table 4.2. This confirms the regression results showing that there is a significant difference between the various training conditions compared to the baseline, with the exception of the /V/ and the /M/ conditions.

Variable	Coefficient	P > t
VISal Layer 2-3	0.4187	0.000
VISal Layer 4	0.6363	0.000
VISam Layer 2-3	0.5883	0.000
VISl Layer 2-3	0.5097	0.000
VISp Layer 2-3	0.5988	0.000
VISp Layer 4	0.6741	0.000
VISpm Layer 2-3	0.5511	0.000
$//M$	0.0154	0.000
$/V/$	-0.0045	0.141
$/VM$	0.0041	0.179
$E//$	0.0590	0.000
E/M	0.0303	0.000
$EV/$	0.0533	0.000
EVM	0.0489	0.000

Table 4.1: Regression analysis of various dataset training conditions. The coefficients in the shaded area represents the similarity of the baseline condition $///$ at that mouse-area. Coefficients for the dataset conditions (unshaded) represent change in similarity compared to baseline.

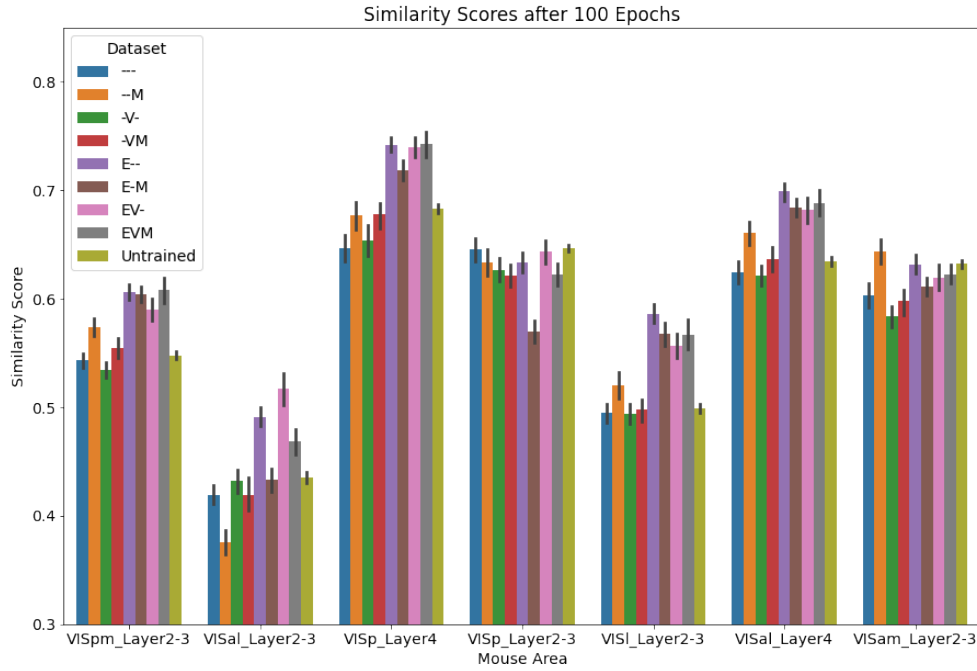


Figure 4.1: The mean similarity scores by mouse-area and dataset condition. Error bars reflect 95% confidence intervals over random seeds and CNN-areas.

Effect	F Value	Pr > F
<i>E//</i>	95.0731	0.0000
<i>/V/</i>	0.1321	0.7201
<i>//M</i>	0.7398	0.3999
<i>EV/</i>	25.1064	0.0001
<i>E/M</i>	50.9053	0.0000
<i>/VM</i>	13.0832	0.0017
<i>EV M</i>	41.2990	0.0000

Table 4.2: ANOVA results: A 2x2x2, within-subject ANOVA, controlling for CNN-Areas as the subject.

In the following sections, we outline some of the key takeaways from the experiments.

4.1 Training has a significant effect on similarity

We want to look at the effects of training compared to the untrained, randomly initialized models in order to elucidate the effects of the training regime as a whole. Figure 4.2 shows the average change in similarity score, across CNN areas and random seeds, of the trained models and 30 randomly initialized MouseNet models. The max and mean are shown in Figure 4.2, showing that this training regime does have a significant effect on similarity scores, with the maximum change in similarity starting at 0.05 for all mouse areas.

Figure 4.3 shows the average similarity score across all areas at the end of training. While this view masks the non-uniform effects owing to different mouse-areas and CNN-areas, it does show a key result: all conditions with the realistic environment boost similarity and outperform the other conditions.

	VISal Layer 2-3	VISal Layer 4	VISam Layer 2-3	VISl Layer 2-3	VISp Layer 2-3	VISp Layer 4	VISpm Layer 2-3
<i>///</i>	0.42	0.62	0.60	0.49	0.65	0.65	0.54
<i>///M</i>	-0.04	0.04	0.04	0.03		0.03	0.03
<i>/V/</i>		-0.02			-0.02		
<i>/VM</i>					-0.02	0.03	
<i>E//</i>	0.07	0.07	0.03	0.09		0.09	0.06
<i>E/M</i>		0.06		0.07	-0.08	0.07	0.06
<i>EV/</i>	0.10	0.06	0.02	0.06		0.09	0.05
<i>EVM</i>	0.05	0.06	0.02	0.07	-0.02	0.10	0.07

Table 4.3: Regressions per mouse area. The constant coefficient is the baseline condition (dataset *///*), and the first row (shaded) captures the similarity score at that baseline condition. The remaining coefficients show the effect of switching to a given dataset condition, per area. Scores with p-values above 0.05 are omitted as blanks.

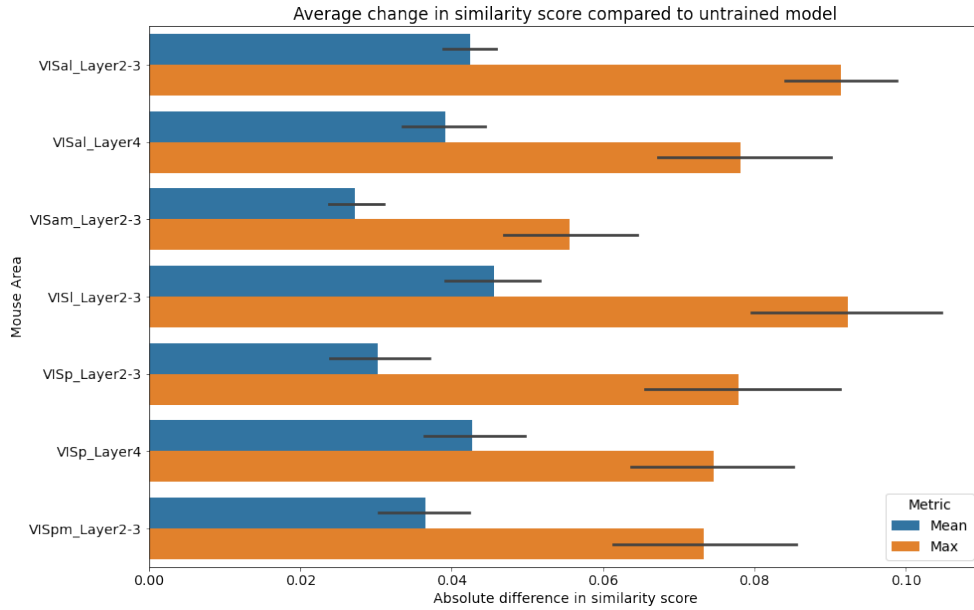


Figure 4.2: Change in similarity score, across CNN-area and random seed, compared to the untrained models. Error bars show standard deviation.

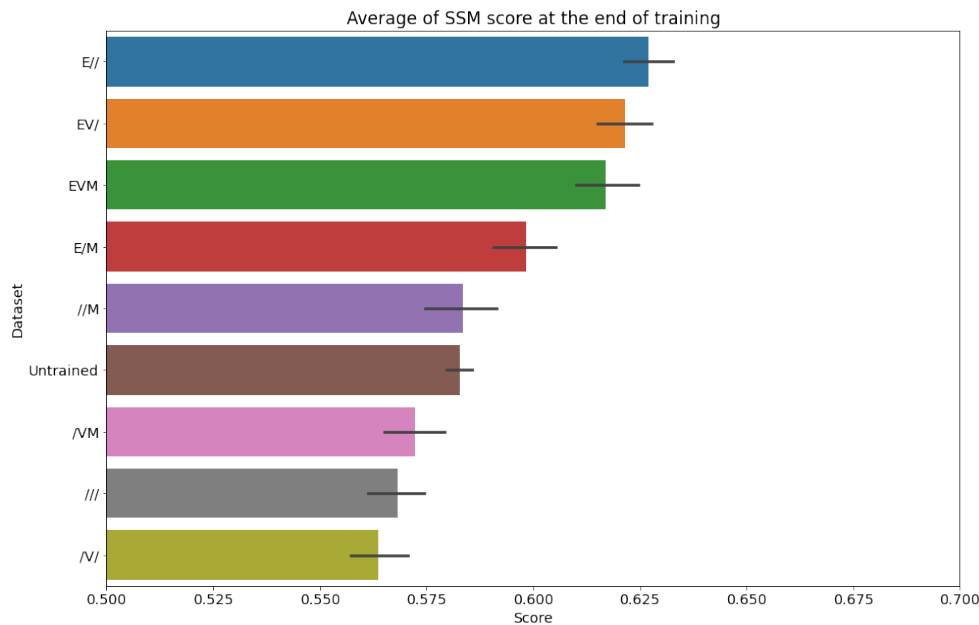


Figure 4.3: Average similarity scores by end of training. Error bars are standard deviation across mouse-areas, CNN-areas, and random seeds.

4.2 Realistic environment yields most significant gains

The Realistic Environment condition has the largest effect on similarity scores. In the aggregate (Table 4.1), the largest increase in similarity is observed in the *E//* dataset condition, which increases similarity by a substantial 5.9%. The next three largest effects are shown by the *EV/*, *EVM*, and *E/M* dataset conditions which increase the score by 5.3%, 4.9%, and 3.0% compared to the baseline *///* condition. Compared to untrained-models, the models trained with the realistic environment condition are the only ones which average higher similarity scores (Figure 4.3).

The only time the realistic environment condition decreases similarity across any mouse area is at mouse area VISp Layer 2-3 (Table 4.3). For *E/M* specifically, the similarity score is uniquely low at VISp Layer 2-3 compared to all other experiment conditions. This condition drops the similarity score by a significant -7.7% below even the untrained-models. Without this outlier at VISp Layer 2-3, training on the *E/M* dataset increases similarity scores by 4.2% from the baseline *///* dataset condition.

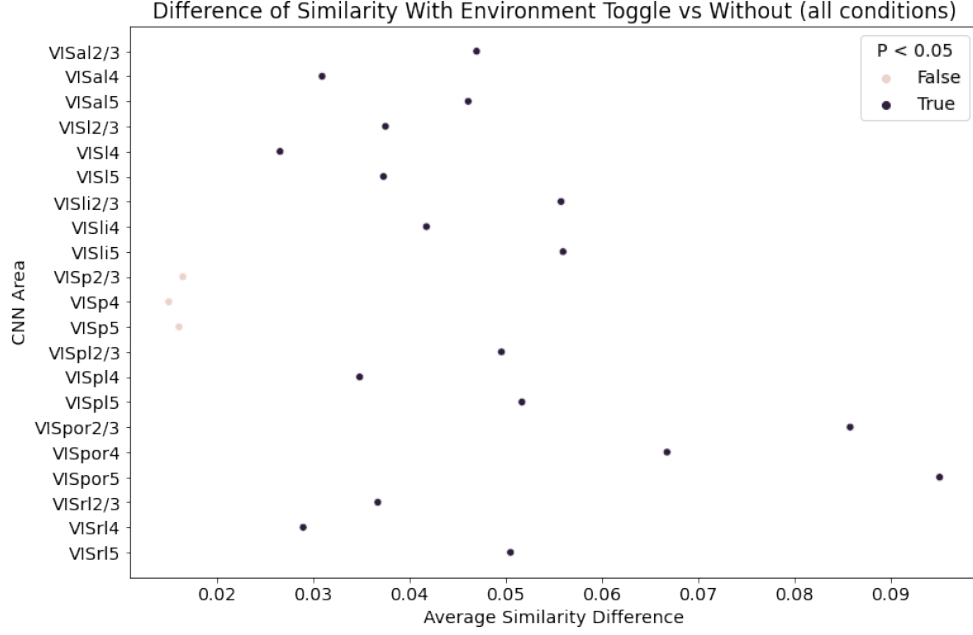


Figure 4.4: Average similarity difference between environment conditions per CNN area. Dark points represent a difference with p-value < 0.05 . Notably, the early CNN areas at VISp show insignificant change, the subsequent parallel CNN areas show a moderate change, and the VISpor – the most downstream CNN area – showed the largest change.

The realistic environment conditions have differing effects across CNN areas as well. Notably, the further downstream in the CNN, the greater the boost to similarity score from enabling a realistic environment. Figure 4.4 shows the average change in similarity between dataset conditions with a realistic environment enabled vs ones with it disabled. The earliest layer in MouseNet is VISp, followed by five parallel streams that ultimately feed into VISpor. The boost owing to the realistic environment condition is smallest at VISp, bounded between 0.02 and 0.06 for the parallel streams, and largest at the VISpor channels.

4.3 Realistic Motion results highly area-dependant

While the realistic motion condition shows modest differences in the aggregate (Table 4.1), there are large area-dependant effects (Table 4.3). In the aggregate, adding the realistic

motion condition slightly decreases the similarity scores compared to the same dataset conditions without realistic motion. The one exception is $//M$ compared to $///$, which increases similarity by 1.5%.

In particular, the high performance of the $//M$ condition corresponds to the only condition in VISam Layer 2-3 which increases similarity compared to the untrained models. This is notable as VISam is typically thought to be the most dorsal region of mouse visual processing, responsible for representing and processing motion [Bakhtiari et al., 2021].

The realistic motion condition sometimes causes large performance degradation. The E/M dataset condition causes a unique and severe decrease in similarity at VISp Layer 2/3. Similarly, VISal Layer 2/3 shows a large -4% at the $//M$ dataset condition. These effects are area-specific and occurred significantly and reliably across five different random seeds (see Appendix A for area-specific similarity scores throughout training).

4.4 Realistic vision has little effect

It is difficult to find a consistent and significant effect owing to realistic vision in this experimental setup. The realistic vision condition decreases visual acuity, making the videos more blurry and removes red colour details. While adding realistic vision in EVM significantly increases similarity compared to E/M , it slightly decreases similarity in all other conditions.

Breaking it down by mouse areas, we see the addition of the realistic vision parameter has a uniformly positive effect on mouse area VISal Layer 2-3, neutral or negative effect on VISal Layer 4, negative effect on VISp Layer 2-3, neutral or positive effect at VISp Layer 4, and mixed results everywhere else.

4.5 Contrastive loss is not a good proxy for mouse brain

While the contrastive training regime has been shown to be effective at training models with increased similarity scores, we show that the contrastive loss and associated accuracy metrics are not positively correlated with similarity. $///$ and $/V/$ are the two dataset conditions which clearly stand out in terms of lowest loss and highest training accuracy. Both of these models are the lowest performing in terms of similarity to mice visual cortex.

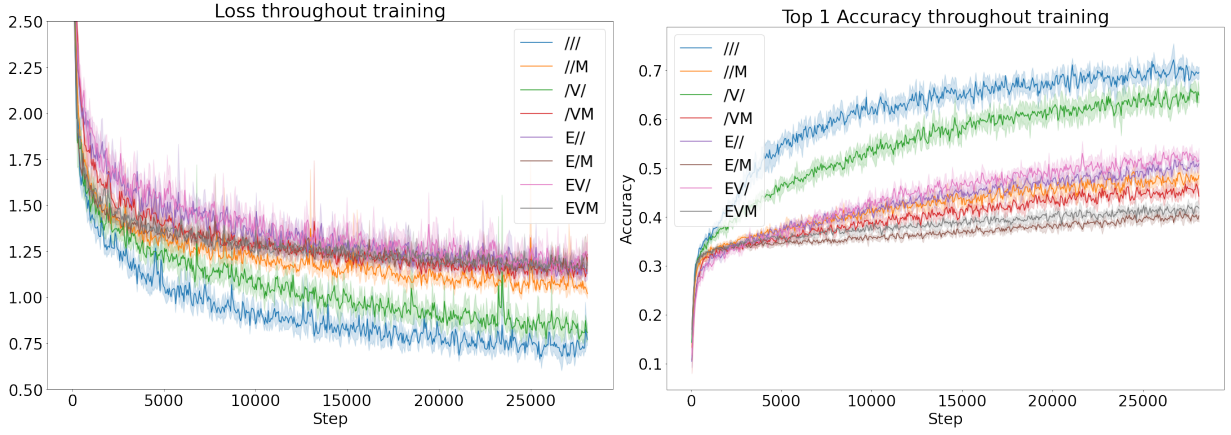


Figure 4.5: Loss and Top-1 Accuracy Curves per mini-batch for all random seeds.

Low loss and high accuracy are likely indicative of a task being "too easy" and thus bypassing the learning of representations more similar to mice cortex.

There is also a noticeably faster drop in the loss, as well as a rise in top-1 accuracy, of models with the motion toggle enabled at the beginning of training (Figure 4.5). This is surprising and counter-intuitive, given that the realistic motion makes the video movements considerably more complex and presumably harder to learn early in training. However, by the end of training the dataset conditions with the realistic motion condition showed similar losses and significantly lower top-1 accuracy compared to their counterparts without the realistic motion conditions. The consistent under-performance of the motion-enabled datasets in terms of top-1 accuracy is not reflected in terms of similarity scores.

Chapter 5

Discussion

This thesis highlights the importance of visual statistics in shaping the representations learned by neural networks. By systematically toggling the realism of the environment, motion, and optical properties in the training data, we have shown that these factors can have a significant influence on the degree to which the learned representations align with those found in biological neural networks.

One of the key findings is that training on videos from a realistic environment leads to the most significant gains in similarity to mouse visual cortex. This suggests that the statistical properties of natural scenes are critical for the development of biologically plausible visual representations. The fact that the boost in similarity is most pronounced in downstream areas of the CNN (e.g., VISpor) aligns with the notion that downstream areas encode more complex and semantically rich features, which are more likely dependent on the complexity and realism of the visual input [Yamins et al., 2014, Margalit et al., 2023, Nayeibi et al., 2023].

5.1 Limitations

There are several limitations to this study that are important to consider. First, the current analysis involving neural activations did not consider any time delays between visual areas. However, biological neural networks do not instantaneously transmit information. This simplification is motivated by toy examples that were run showing the final similarity measures are not sensitive to temporal offsets up to 130ms. The structure of the similarity matrices are dominated by the temporal autocorrelation in the movie stimuli. Extensions to this work aiming to capture temporal dynamics should also investigate this simplification.

Another limitation is the use of a 2D CNN as the backbone model. While the DPC architecture does capture temporal dynamics through a gated recurrent unit, MouseNet itself does not in the same way that 3D-CNN or recurrent networks could. Incorporating the temporal information more explicitly into the model architecture would probably lead to more biologically plausible representations and dynamics.

There is also a potential mismatch between the training environment and the natural habitat of mice. While a daylight meadow scene is used to represent a naturalistic environment, mice are primarily nocturnal and burrowing animals. The meadow environment does not toggle between night-time and day-time and does not contain burrows or otherwise allow for the mouse-agent to do a borrowing-like action.

Another important limitation to consider is the potential conflation of "naturalism" with complexity in this study. None of the datasets were controlled for spatial frequency components or other measures of complexity. Further work would benefit from controlling for these factors to determine which specific elements of naturalism lead to greater similarity.

Furthermore, the mouse motion model was trained on data collected in a 50cm x 50cm cage, which may not fully capture the range and complexity of natural mouse movements in the wild. The data likely oversimplifies the diverse movement patterns that mice exhibit in their natural habitats. For example, mice seem to express an anxiety driven tendency to stay close to walls in their motion [Simon et al., 1994], a phenomenon that is not addressed in the current motion model as it is independent of the environment. Future studies would likely benefit from more complex motion models which are trained on more sophisticated data, incorporate task-based objectives, and are aware of the environment.

Finally, our experiments were cut down due to computational constraints. The contrastive training regime we employed likely would have benefited from batch sizes larger than 32, but that was the largest amount that would fit within the 32GB memory limit on the V100 GPUs. Additionally, despite recording videos from both eyes of the virtual mouse agent, only outputs from the right eye were used for training. This was once again due to memory limits. Future work would benefit from exploring optimization techniques in the deep learning literature in addition to larger memory GPUs.

5.2 Future Directions

This work opens up several exciting avenues for future research. It provides a virtual environment, code resources, and datasets which will hopefully set the stage for further

insights into the complexities of neural computation. One promising direction is to extend the Unity environment with a richer feature set, including more diverse objects, textures, scenes, and other animated agents. This could help to better capture the complexity and variability of natural environments experienced by mice.

References

- M. A. Ali and M. A. Klyne. *Vision in Vertebrates*. Springer US, 1985. ISBN 9781468491319. doi: 10.1007/978-1-4684-9129-6. URL <http://dx.doi.org/10.1007/978-1-4684-9129-6>.
- Emily J. Allen, Ghislain St-Yves, Yihan Wu, Jesse L. Breedlove, Jacob S. Prince, Logan T. Dowdle, Matthias Nau, Brad Caron, Franco Pestilli, Ian Charest, J. Benjamin Hutchinson, Thomas Naselaris, and Kendrick Kay. A massive 7t fmri dataset to bridge cognitive neuroscience and artificial intelligence. *Nature Neuroscience*, 25 (1):116–126, 12 2021. ISSN 1097-6256. doi: 10.1038/s41593-021-00962-x. URL <http://dx.doi.org/10.1038/s41593-021-00962-x>.
- Bruno Antonio, Davide Moroni, and Massimo Martinelli. Efficient adaptive ensembling for image classification. *Expert Systems*, 8 2023. ISSN 0266-4720. doi: 10.1111/exsy.13424. URL <http://dx.doi.org/10.1111/exsy.13424>.
- Shahab Bakhtiari, Patrick Mineault, Timothy Lillicrap, Christopher Pack, and Blake Richards. The functional specialization of visual cortex emerges from training parallel pathways with self-supervised predictive learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25164–25178. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/d384dec9f5f7a64a36b5c8f03b8a6d92-Paper.pdf.
- Nicholas M. Blauch, Marlene Behrmann, and David C. Plaut. A connectivity-constrained computational account of topographic organization in primate high-level visual cortex. *Proceedings of the National Academy of Sciences*, 119(3), 1 2022. ISSN 0027-8424. doi: 10.1073/pnas.2112566119. URL <http://dx.doi.org/10.1073/pnas.2112566119>.
- Tyler Bonnen, Daniel L.K. Yamins, and Anthony D. Wagner. When the ventral visual stream is not enough: A deep learning account of medial temporal lobe involvement in

- perception. *Neuron*, 109(17):2755–2766.e6, 9 2021. ISSN 0896-6273. doi: 10.1016/j.neuron.2021.06.018. URL <http://dx.doi.org/10.1016/j.neuron.2021.06.018>.
- Radoslaw Martin Cichy, Aditya Khosla, Dimitrios Pantazis, Antonio Torralba, and Aude Oliva. Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence. *Scientific Reports*, 6(1), 6 2016. ISSN 2045-2322. doi: 10.1038/srep27755. URL <http://dx.doi.org/10.1038/srep27755>.
- Saskia E. J. de Vries, Jerome A. Lecoq, Michael A. Buice, Peter A. Groblewski, Gabriel K. Ocker, Michael Oliver, David Feng, Nicholas Cain, Peter Ledochowitsch, Daniel Millman, Kate Roll, Marina Garrett, Tom Keenan, Leonard Kuan, Stefan Mihalas, Shawn Olsen, Carol Thompson, Wayne Wakeman, Jack Waters, Derric Williams, Chris Barber, Nathan Berbesque, Brandon Blanchard, Nicholas Bowles, Shiella D. Caldejon, Linzy Casal, Andrew Cho, Sissy Cross, Chinh Dang, Tim Dolbeare, Melise Edwards, John Galbraith, Nathalie Gaudreault, Terri L. Gilbert, Fiona Griffin, Perry Hargrave, Robert Howard, Lawrence Huang, Sean Jewell, Nika Keller, Ulf Knoblich, Josh D. Larkin, Rachael Larsen, Chris Lau, Eric Lee, Felix Lee, Arielle Leon, Lu Li, Fuhui Long, Jennifer Luviano, Kyla Mace, Thuyanh Nguyen, Jed Perkins, Miranda Robertson, Sam Seid, Eric Shea-Brown, Jianghong Shi, Nathan Sjoquist, Cliff Slaughterbeck, David Sullivan, Ryan Valenza, Casey White, Ali Williford, Daniela M. Witten, Jun Zhuang, Hongkui Zeng, Colin Farrell, Lydia Ng, Amy Bernard, John W. Phillips, R. Clay Reid, and Christof Koch. A large-scale standardized physiological survey reveals functional organization of the mouse visual cortex. *Nature Neuroscience*, 23(1):138–151, 12 2019. ISSN 1097-6256. doi: 10.1038/s41593-019-0550-9. URL <http://dx.doi.org/10.1038/s41593-019-0550-9>.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- Katharina Dobs, Julio Martinez, Alexander J. E. Kell, and Nancy Kanwisher. Brain-like functional specialization emerges spontaneously in deep neural networks. *Science Advances*, 8(11):eabl8913, 2022. doi: 10.1126/sciadv.abl8913. URL <https://www.science.org/doi/abs/10.1126/sciadv.abl8913>.
- Fenil R. Doshi and Talia Konkle. Visual object topographic motifs emerge from self-organization of a unified representational space. 9 2022. URL <http://dx.doi.org/10.1101/2022.09.06.506403>.

- Ebal-Studios. Sci-fi styled modular pack - unity asset store. URL <https://assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-styled-modular-pack-82913>. Accessed: 2024-07-11.
- Dawn Finzi, Eshed Margalit, Kendrick Kay, Daniel LK Yamins, and Kalanit Grill-Spector. Topographic dcnnns trained on a single self-supervised task capture the functional organization of cortex into visual processing streams. In *SVRHM 2022 Workshop NeurIPS*, 2022. URL <https://openreview.net/forum?id=E1iY-d13smd>.
- Ying Geng, Lee Anne Schery, Robin Sharma, Alfredo Dubra, Kamran Ahmad, Richard T. Libby, and David R. Williams. Optical properties of the mouse eye. *Biomedical Optics Express*, 2(4):717–738, April 2011. ISSN 2156-7085. doi: 10.1364/BOE.2.000717. URL <https://opg.optica.org/boe/abstract.cfm?uri=boe-2-4-717>. Publisher: Optica Publishing Group.
- Christine Grienberger, Andrea Giovannucci, William Zeiger, and Carlos Portera-Cailliau. Two-photon calcium imaging of neuronal activity. *Nature Reviews Methods Primers*, 2(1), 9 2022. ISSN 2662-8449. doi: 10.1038/s43586-022-00147-1. URL <http://dx.doi.org/10.1038/s43586-022-00147-1>.
- Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. 2019. doi: 10.48550/ARXIV.1909.04656. URL <https://arxiv.org/abs/1909.04656>.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015a. doi: 10.48550/ARXIV.1512.03385. URL <https://arxiv.org/abs/1512.03385>.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. 2015b. doi: 10.48550/ARXIV.1502.01852. URL <https://arxiv.org/abs/1502.01852>.
- Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson. Learning visual groups from co-occurrences in space and time. 2015. doi: 10.48550/ARXIV.1511.06811. URL <https://arxiv.org/abs/1511.06811>.
- Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. 2018. doi: 10.48550/ARXIV.1809.02627. URL <https://arxiv.org/abs/1809.02627>.

- T. Anderson Keller, Qinghe Gao, and Max Welling. Modeling category-selective cortical regions with topographic variational autoencoders. 2021. doi: 10.48550/ARXIV.2110.13911. URL <https://arxiv.org/abs/2110.13911>.
- Seyed-Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS Computational Biology*, 10(11):e1003915, November 2014. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1003915. URL <https://dx.plos.org/10.1371/journal.pcbi.1003915>.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014. doi: 10.48550/ARXIV.1412.6980. URL <https://arxiv.org/abs/1412.6980>.
- Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2:1–28, 2008. URL <https://www.frontiersin.org/articles/10.3389/neuro.06.004.2008/full>.
- Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL <http://www.cs.toronto.edu/~kriz/cifar.html>.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL <https://proceedings.neurips.cc/paper%5Ffiles/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.
- Najib J. Majaj, Ha Hong, Ethan A. Solomon, and James J. DiCarlo. Simple learned weighted sums of inferior temporal neuronal firing rates accurately predict human core object recognition performance. *The Journal of Neuroscience*, 35(39):13402–13418, 9 2015. ISSN 0270-6474. doi: 10.1523/jneurosci.5181-14.2015. URL <http://dx.doi.org/10.1523/JNEUROSCI.5181-14.2015>.
- Eshed Margalit, Hyodong Lee, Dawn Finzi, James J. DiCarlo, Kalanit Grill-Spector, and Daniel L. K. Yamins. A unifying principle for the functional organization of visual cortex. 5 2023. URL <http://dx.doi.org/10.1101/2023.05.18.541361>.
- Johannes Mehrer, Courtney J. Spoerer, Emer C. Jones, Nikolaus Kriegeskorte, and Tim C. Kietzmann. An ecologically motivated image dataset for deep learning yields better models of human vision. *Proceedings of the National Academy of Sciences*, 118(8), 2 2021. ISSN 0027-8424. doi: 10.1073/pnas.2011417118. URL <http://dx.doi.org/10.1073/pnas.2011417118>.

- Patrick J Mineault, Shahab Bakhtiari, Blake A Richards, and Christopher C Pack. Your head is there to move you around: Goal-driven models of the primate dorsal pathway. 7 2021. URL <http://dx.doi.org/10.1101/2021.07.09.451701>.
- NatureManufacture. Naturemanufacture - unity asset store. URL <https://assetstore.unity.com/publishers/6887>. Accessed: 2024-07-11.
- Aran Nayebi, Nathan C. L. Kong, Chengxu Zhuang, Justin L. Gardner, Anthony M. Norcia, and Daniel L. K. Yamins. Mouse visual cortex as a limited resource system that self-learns an ecologically-general representation. *PLOS Computational Biology*, 19(10):e1011506, October 2023. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1011506. URL <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1011506>. Publisher: Public Library of Science.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, January 2019. URL <http://arxiv.org/abs/1807.03748>. arXiv:1807.03748 [cs, stat].
- Byron H. Price and Jeffrey P. Gavnornik. Efficient temporal coding in the early visual system: Existing evidence and future directions. *Frontiers in Computational Neuroscience*, 16, 7 2022. ISSN 1662-5188. doi: 10.3389/fncom.2022.929348. URL <http://dx.doi.org/10.3389/fncom.2022.929348>.
- Glen T. Prusky, Nazia M. Alam, and Robert M. Douglas. Enhancement of vision by monocular deprivation in adult mice. *Journal of Neuroscience*, 26(45):11554–11561, November 2006. ISSN 0270-6474, 1529-2401. doi: 10.1523/JNEUROSCI.3396-06.2006. URL <https://www.jneurosci.org/content/26/45/11554>. Publisher: Society for Neuroscience Section: Articles.
- Rowntec-Assets. Survival kit lite - unity asset store. URL <https://assetstore.unity.com/packages/3d/props/tools/survival-kit-lite-92549>. Accessed: 2024-07-11.
- Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.
- Jianghong Shi, Bryan Tripp, Eric Shea-Brown, Stefan Mihalas, and Michael A. Buice. Mousenet: A biologically constrained convolutional neural network model for the mouse visual cortex. *PLOS Computational Biology*, 18(9):e1010427, September 2022. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1010427. URL <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1010427>. Publisher: Public Library of Science.

- Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '85, pages 245–254, New York, NY, USA, July 1985. Association for Computing Machinery. ISBN 978-0-89791-166-5. doi: 10.1145/325334.325242. URL <https://dl.acm.org/doi/10.1145/325334.325242>.
- P. Simon, R. Dupuis, and J. Costentin. Thigmotaxis as an index of anxiety in mice. influence of dopaminergic transmissions. *Behavioural Brain Research*, 61(1):59–64, 3 1994. ISSN 0166-4328. doi: 10.1016/0166-4328(94)90008-6. URL [http://dx.doi.org/10.1016/0166-4328\(94\)90008-6](http://dx.doi.org/10.1016/0166-4328(94)90008-6).
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2014. doi: 10.48550/ARXIV.1409.1556. URL <https://arxiv.org/abs/1409.1556>.
- Swati Singh and Amanpreet Kaur. Game development using unity game engine. 2022 3rd International Conference on Computing, Analytics and Networks (ICAN), IEEE, 11 2022. doi: 10.1109/ican56228.2022.10007155. URL <http://dx.doi.org/10.1109/ICAN56228.2022.10007155>.
- Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. 2012. doi: 10.48550/ARXIV.1212.0402. URL <https://arxiv.org/abs/1212.0402>.
- Ian H Stevenson and Konrad P Kording. How advances in neural recording affect data analysis. *Nature Neuroscience*, 14(2):139–142, 1 2011. ISSN 1097-6256. doi: 10.1038/nn.2731. URL <http://dx.doi.org/10.1038/nn.2731>.
- P. Stoica and Y. Selen. Model-order selection. *IEEE Signal Processing Magazine*, 21(4): 36–47, 7 2004. ISSN 1053-5888. doi: 10.1109/msp.2004.1311138. URL <http://dx.doi.org/10.1109/MSP.2004.1311138>.
- Pavel Tokmakov, Karteeek Alahari, and Cordelia Schmid. Learning video object segmentation with visual memory. 2017. doi: 10.48550/ARXIV.1704.05737. URL <https://arxiv.org/abs/1704.05737>.
- Bryan Tripp. Approximating the architecture of visual cortex in a convolutional network. *Neural Computation*, 31(8):1551–1591, 8 2019. ISSN 0899-7667. doi: 10.1162/neco_a_01211. URL <http://dx.doi.org/10.1162/neco%5Fa%5F01211>.

- Daniel L. K. Yamins, Ha Hong, Charles F. Cadieu, Ethan A. Solomon, Darren Seibert, and James J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 5 2014. ISSN 0027-8424. doi: 10.1073/pnas.1403112111. URL <http://dx.doi.org/10.1073/pnas.1403112111>.
- Yiyuan Zhang, Ke Zhou, Pinglei Bao, and Jia Liu. Principles governing the topological organization of object selectivities in ventral temporal cortex. 9 2021. URL <http://dx.doi.org/10.1101/2021.09.15.460220>.
- Chengxu Zhuang, Siming Yan, Aran Nayebi, Martin Schrimpf, Michael C. Frank, James J. DiCarlo, and Daniel L. K. Yamins. Unsupervised neural network models of the ventral visual stream. *Proceedings of the National Academy of Sciences*, 118(3):e2014196118, January 2021. doi: 10.1073/pnas.2014196118. URL <https://www.pnas.org/doi/abs/10.1073/pnas.2014196118>. Publisher: Proceedings of the National Academy of Sciences.

APPENDICES

Appendix A

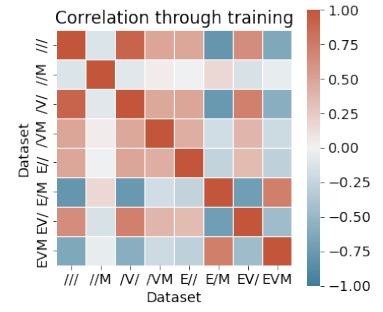
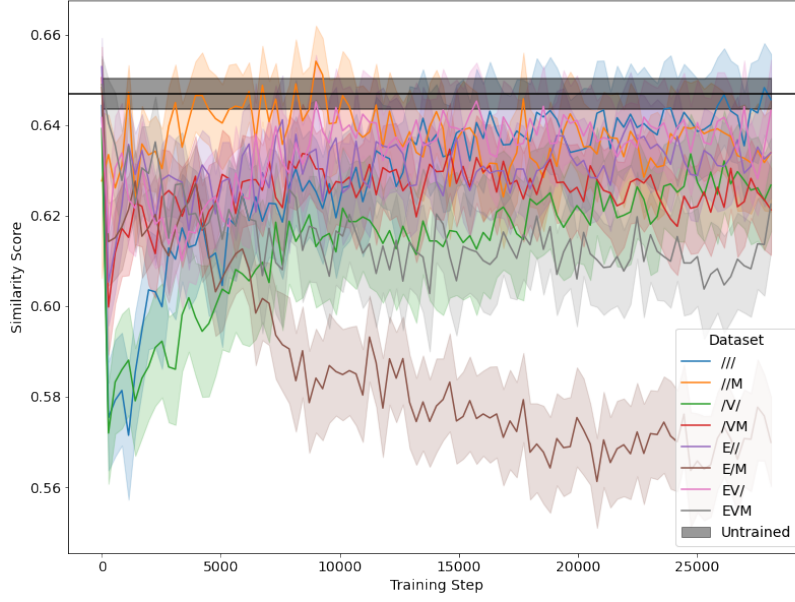
Similarity Scores throughout training

A.1 Mouse View

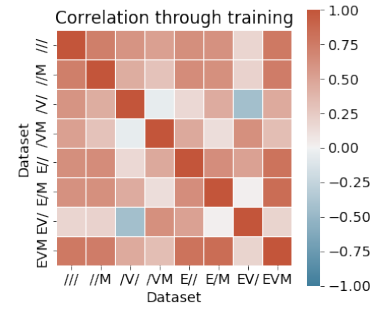
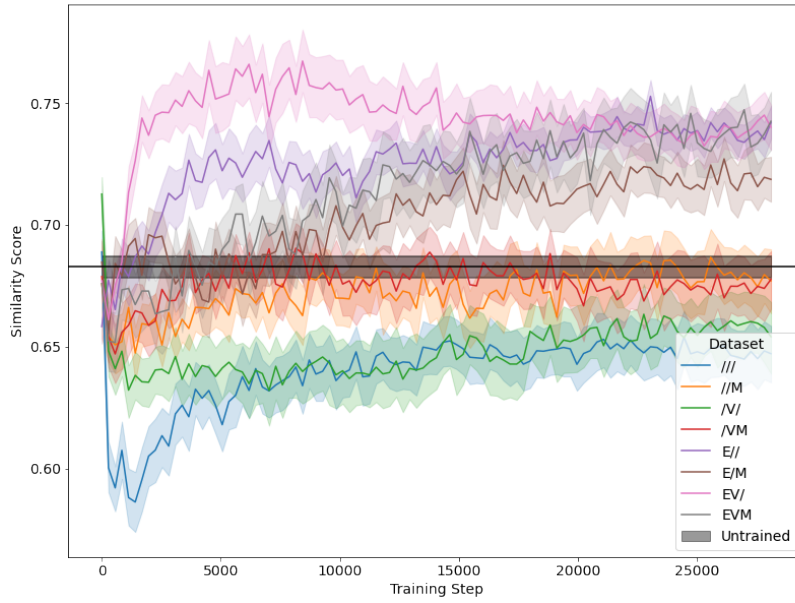
Similarity scores by mouse-area (columns in the SSM matrix).

A.1.1 Mouse Areas

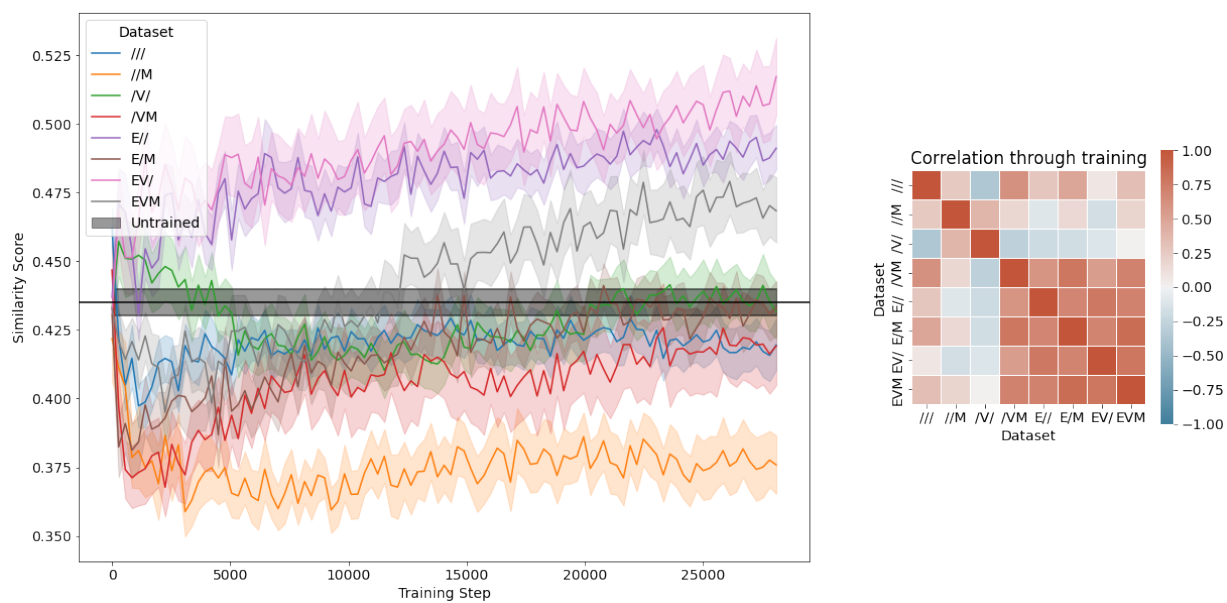
Similarity Score through training for Mouse Area: VIsP_Layer2-3



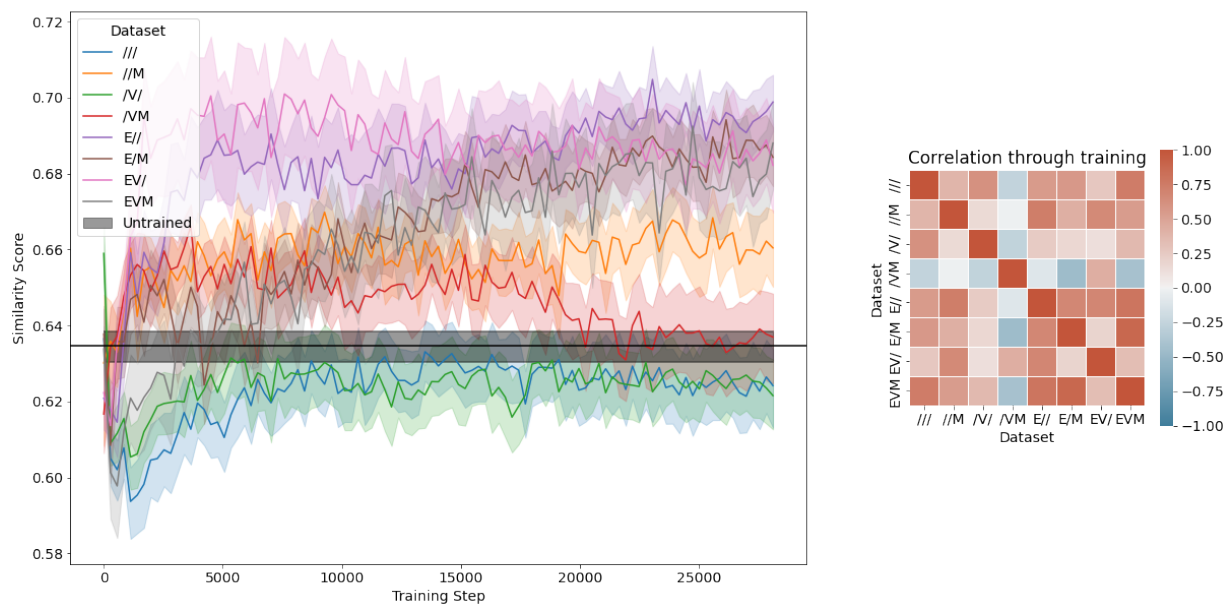
Similarity Score through training for Mouse Area: VIsP_Layer4



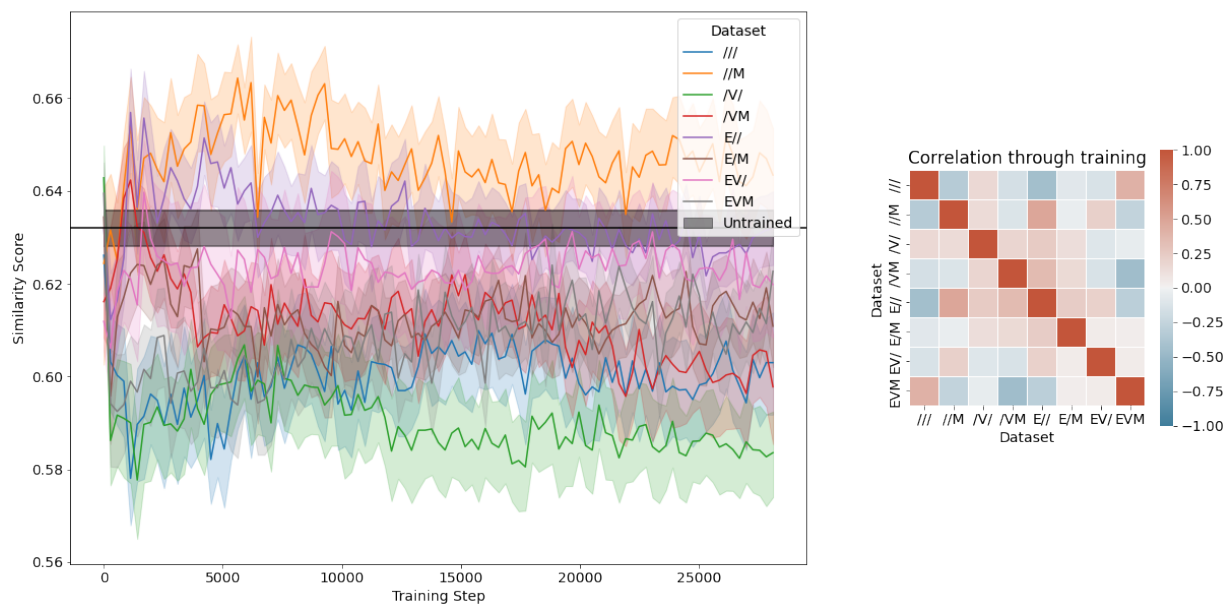
Similarity Score through training for Mouse Area: VISal_Layer2-3



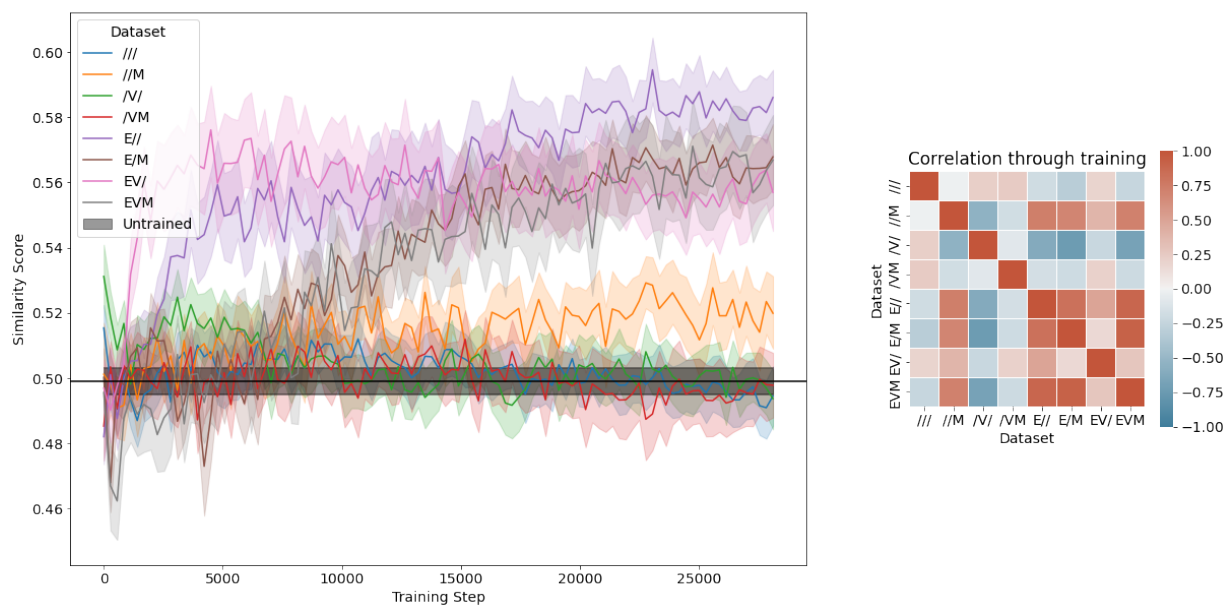
Similarity Score through training for Mouse Area: VISal_Layer4



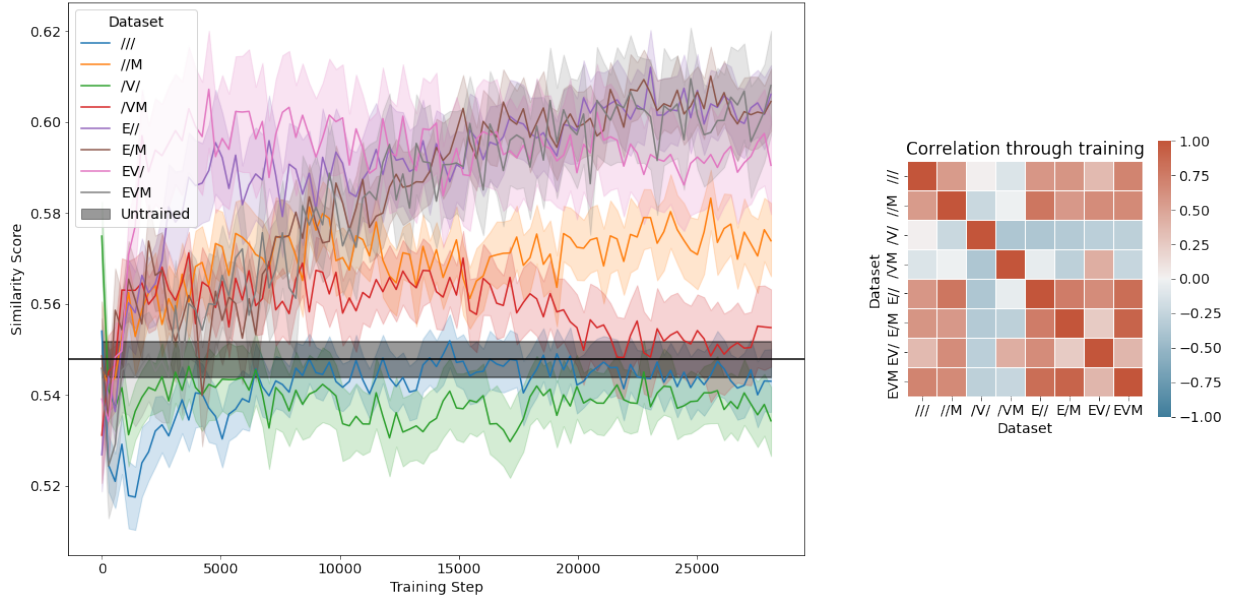
Similarity Score through training for Mouse Area: VISam_Layer2-3



Similarity Score through training for Mouse Area: VISI_Layer2-3



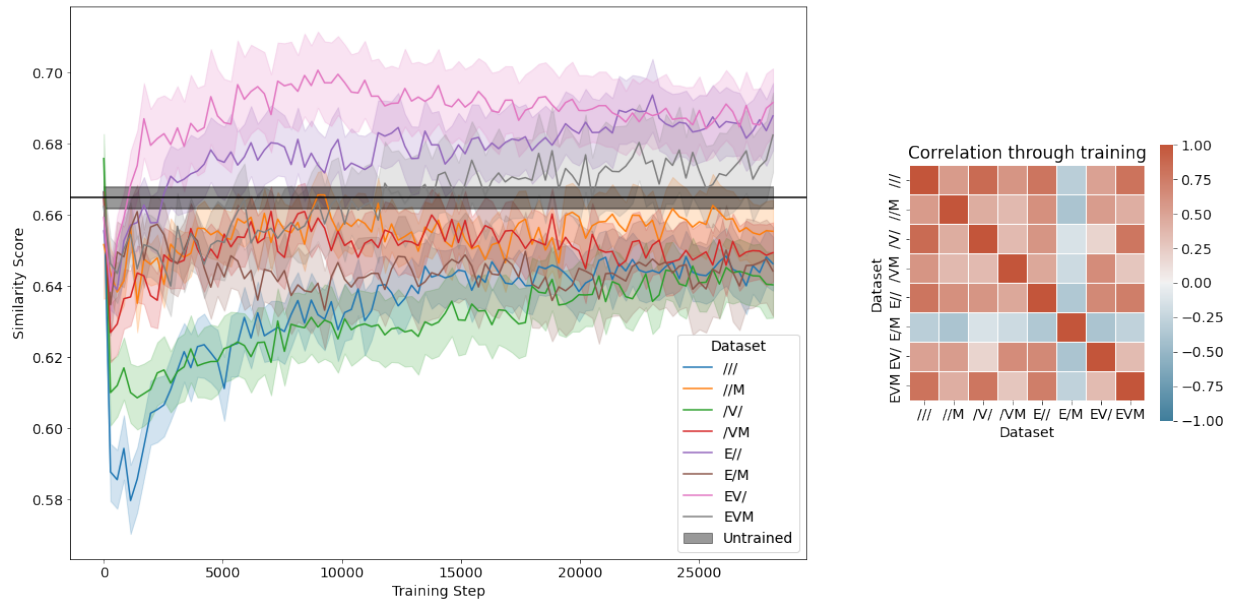
Similarity Score through training for Mouse Area: VISpm_Layer2-3



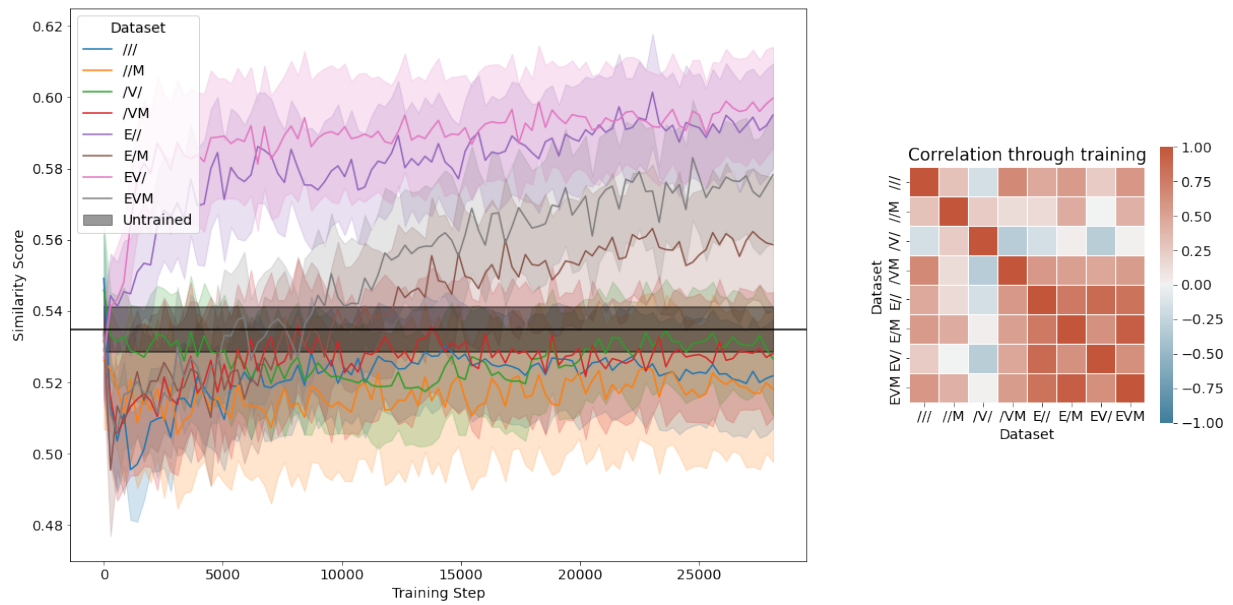
A.1.2 Mouse Regions

Similarity scores of mouse areas grouped by region and plotted through training. For example "VISp_Layer2-3" and "VISp_layer4" would be grouped into "VISp".

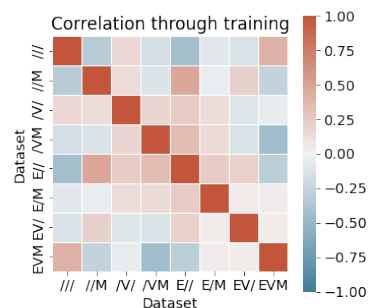
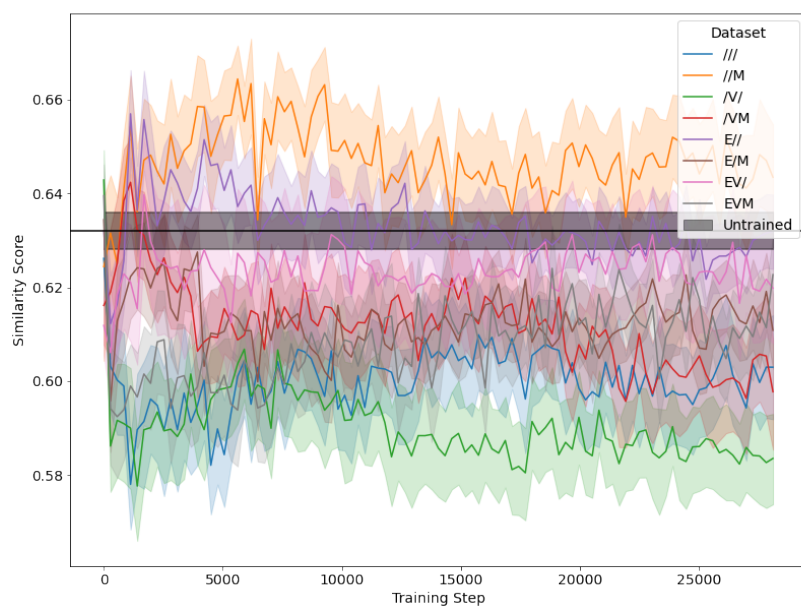
Similarity Score through training for Mouse Region: VISp



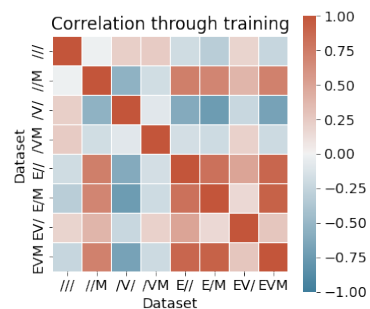
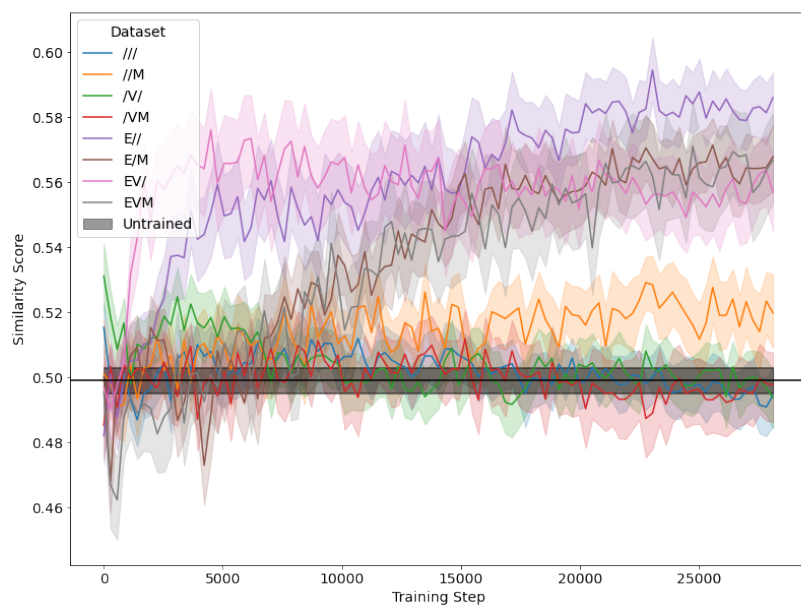
Similarity Score through training for Mouse Region: VISal



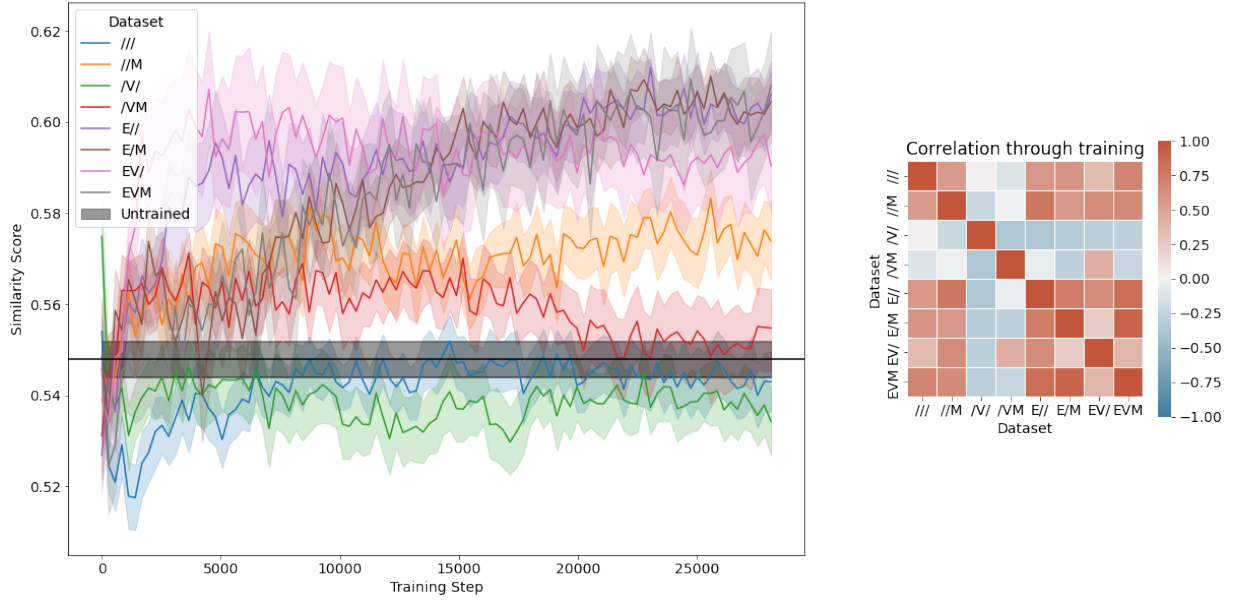
Similarity Score through training for Mouse Region: VISam



Similarity Score through training for Mouse Region: VISI



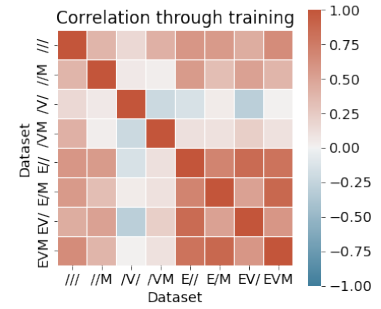
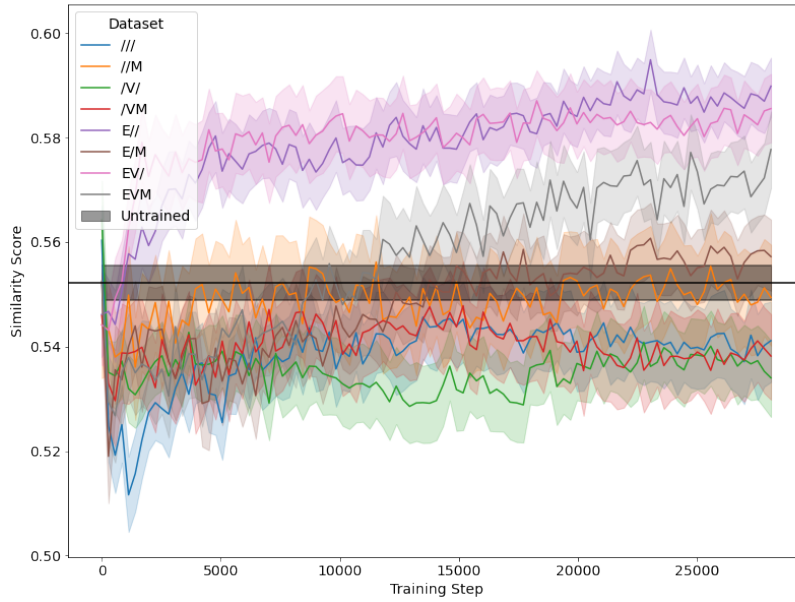
Similarity Score through training for Mouse Region: VISpm



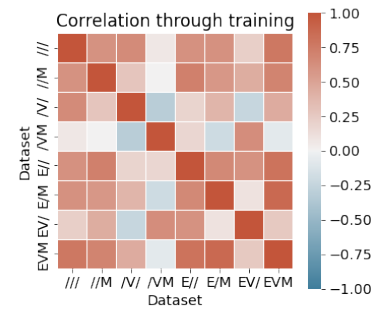
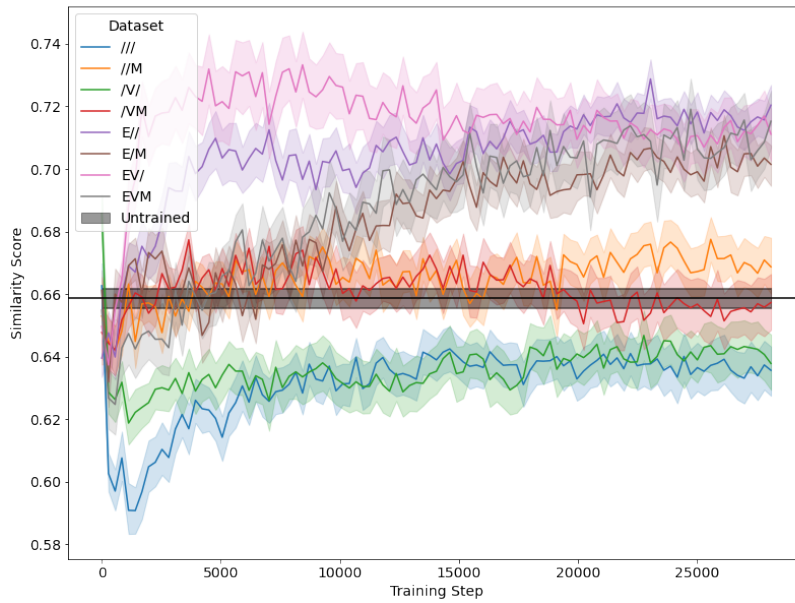
A.1.3 Mouse Layers

The similarity scores across the two measured mouse layers, "layer 2-3" and "layer 4", plotted through training.

Similarity Score through training for Mouse Layer: Layer2-3



Similarity Score through training for Mouse Layer: Layer4

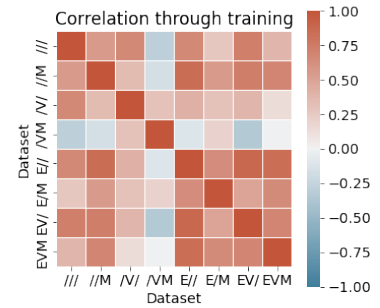
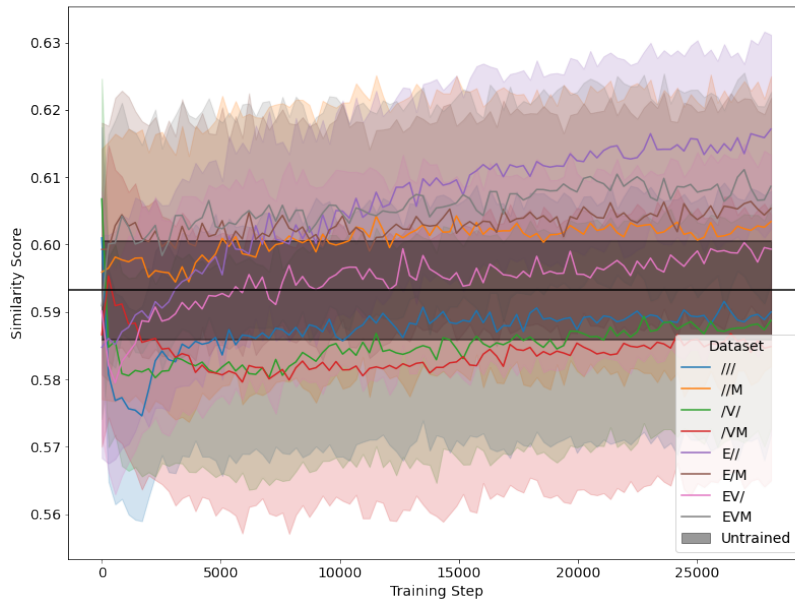


A.2 CNN View

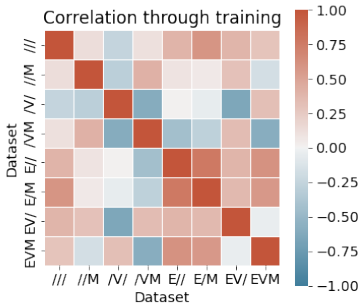
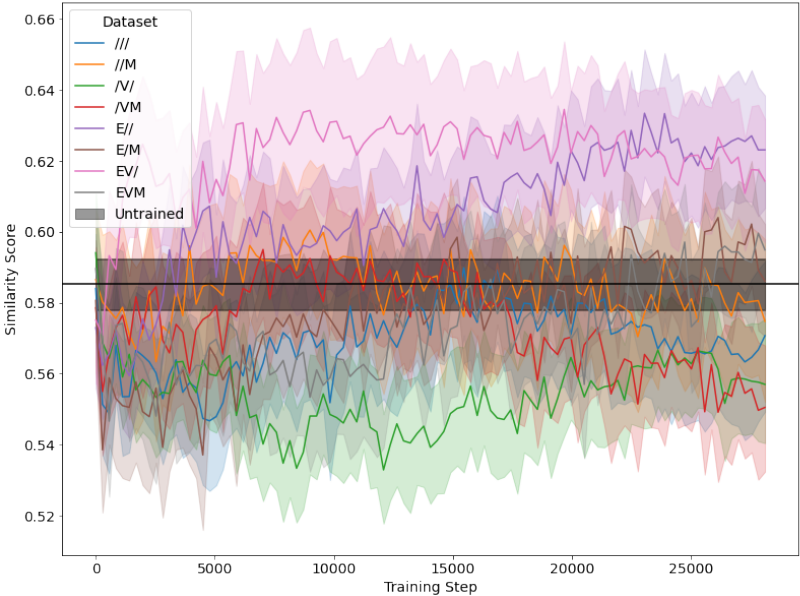
Similarity scores by CNN layer/region (rows in the SSM matrix) are shown below. For brevity, not all 21 CNN areas are shown. Instead, aggregation of similarity score by layer and region are plotted through training.

A.2.1 CNN Regions

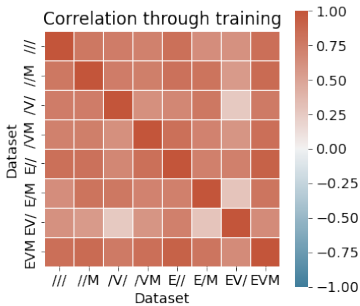
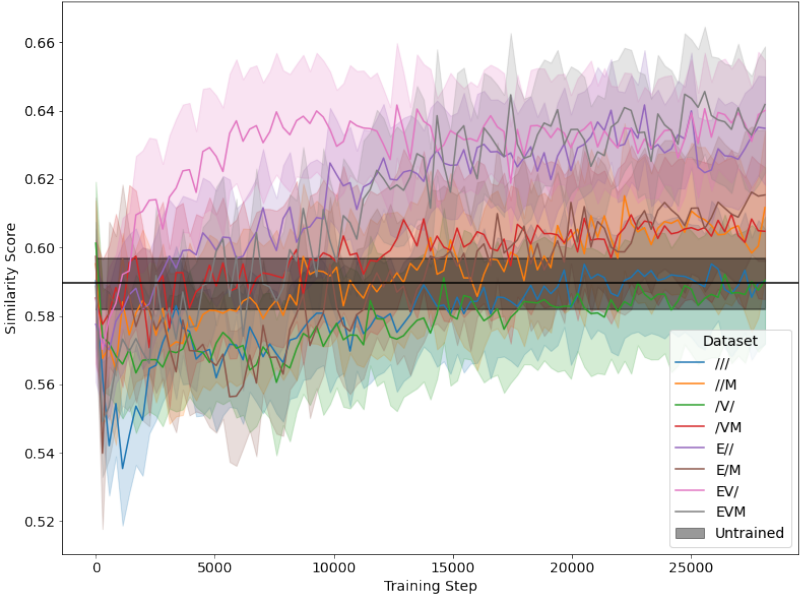
Similarity Score through training for CNN Region: VISp



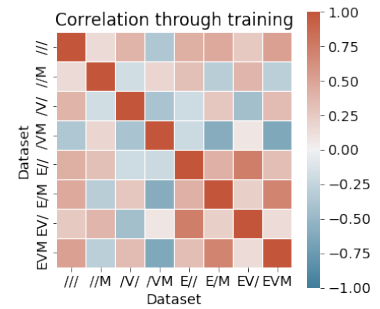
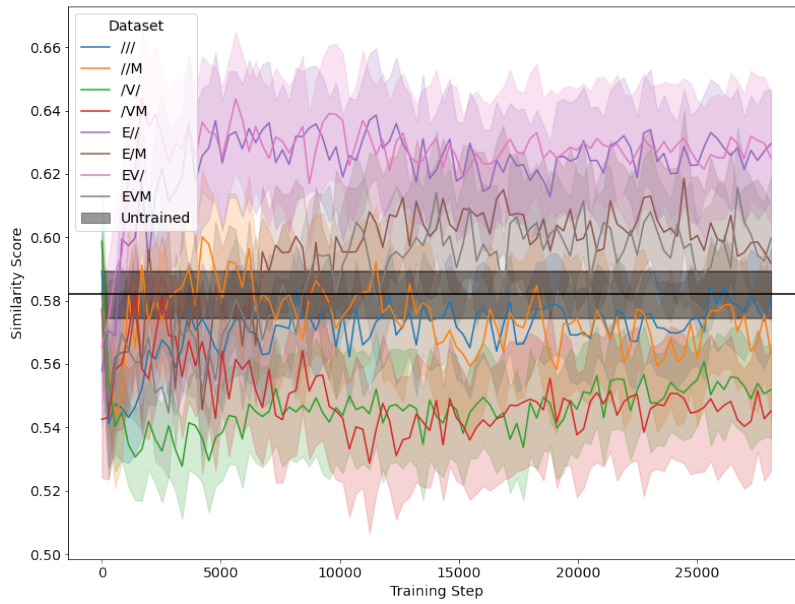
Similarity Score through training for CNN Region: VISal



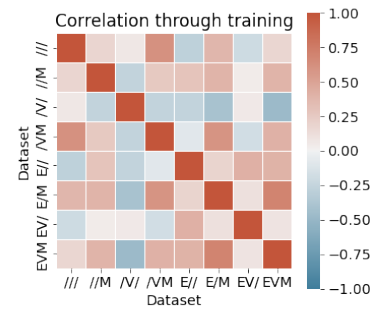
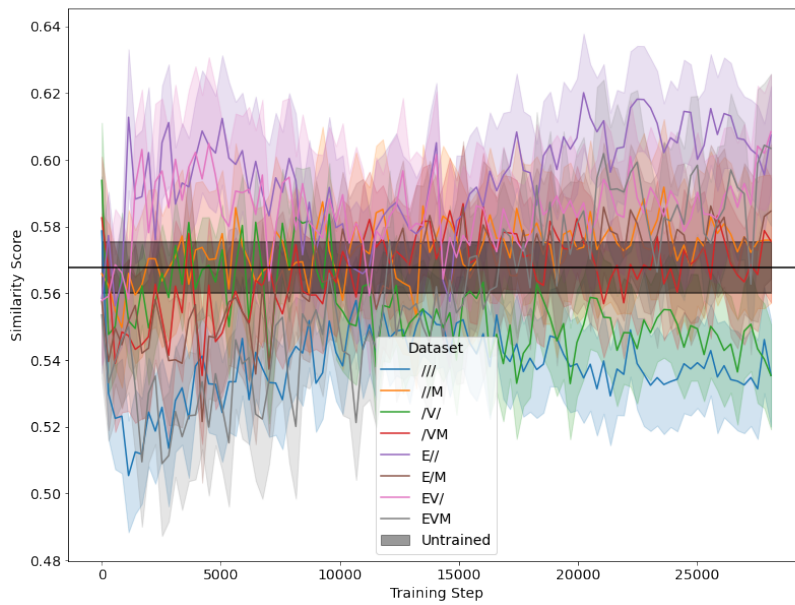
Similarity Score through training for CNN Region: VISI



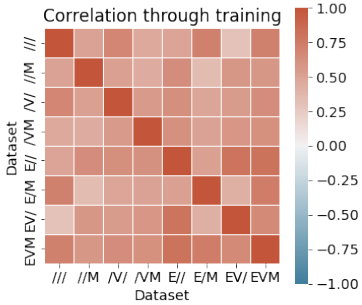
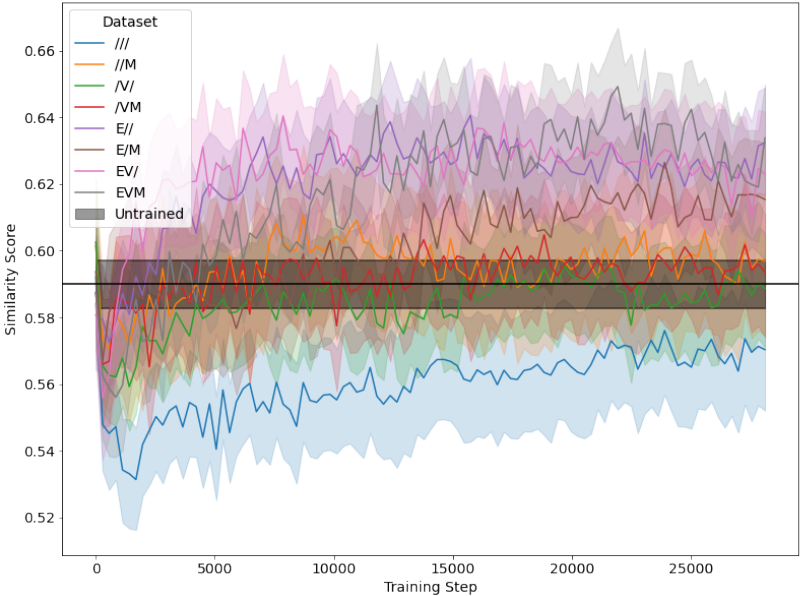
Similarity Score through training for CNN Region: VISli



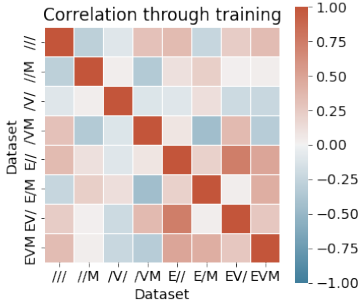
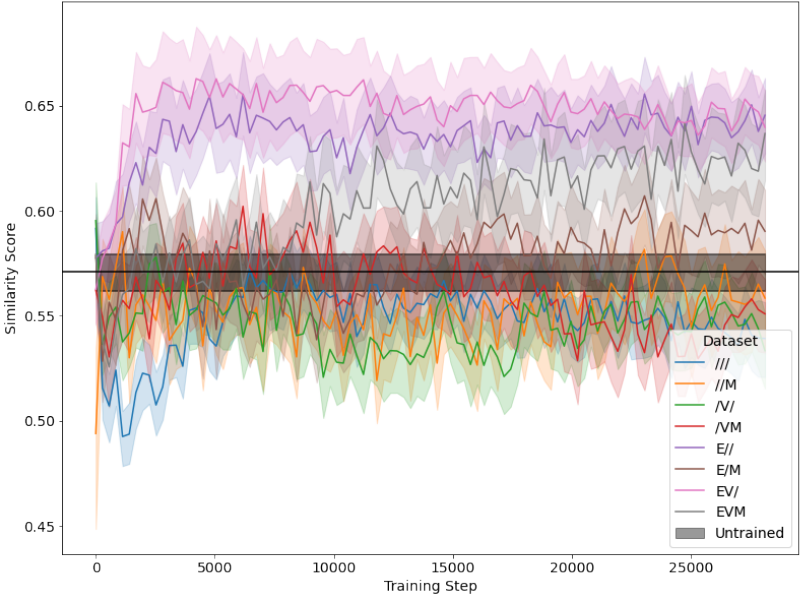
Similarity Score through training for CNN Region: VISpl



Similarity Score through training for CNN Region: VISrl

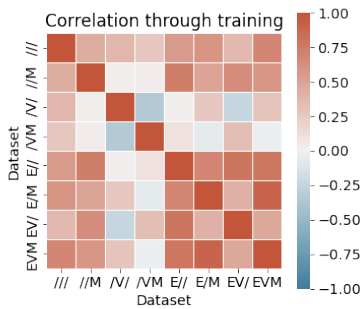
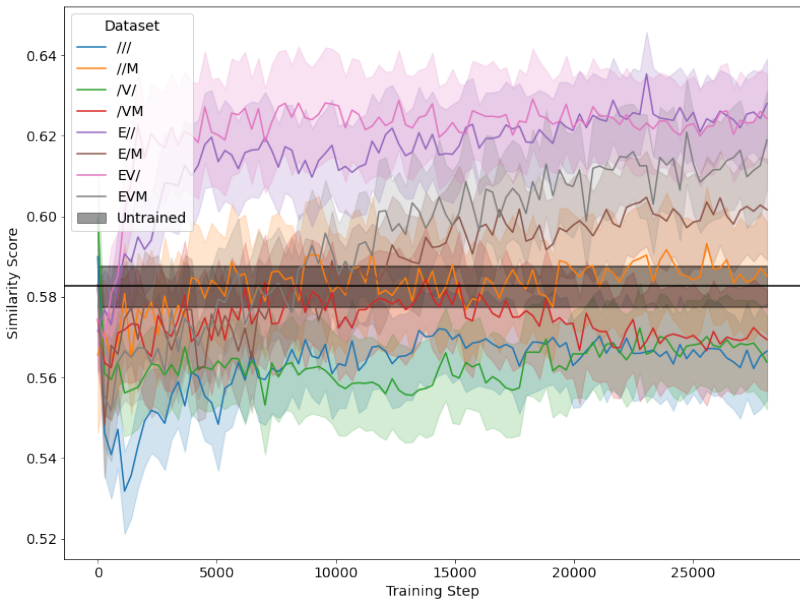


Similarity Score through training for CNN Region: VISpor

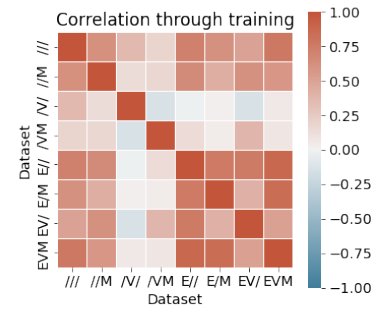
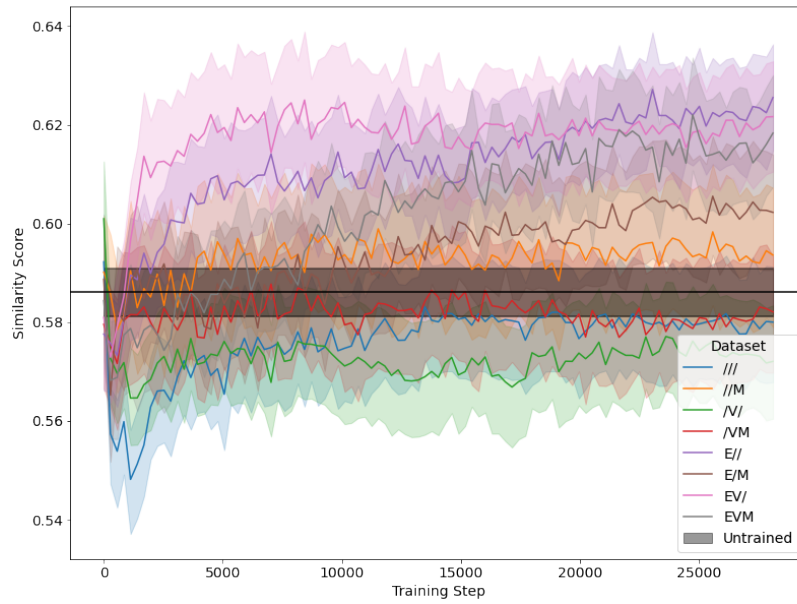


A.2.2 CNN Layers

Similarity Score through training for CNN Layer: 2/3



Similarity Score through training for CNN Layer: 4



Similarity Score through training for CNN Layer: 5

