Predictive Self-Supervised Learning in Brains and Machines

Inauguraldissertation

zur
Erlangung der Würde eines Doktors der Philosophie
vorgelegt der
Philosophisch-Naturwissenschaftlichen Fakultät
der Universität Basel

von

Manu Srinath Halvagal

Basel, 2025

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät
auf Antrag von

Prof. Dr. Friedemann Zenke

(Erstbetreuer)

Prof. Dr. Rainer Friedrich

(Zweitbetreuer)

Prof. Dr. Andrew Saxe

(Externer Experte)

Basel, den 15. April 2025

Prof. Dr. Heiko Schuldt
Dekan

“*What I cannot create, I do not understand.*”

Richard P. Feynman

To my parents.

Abstract

The brain possesses a remarkable ability to transform chaotic, ever-changing sensory input into coherent percepts by building structured internal representations. How these representations are learned and updated remains a fundamental question in neuroscience. Existing neuroscience models of representation learning in the brain typically fail to capture rich representations in network simulations, while machine learning models successfully learn such representations but lack biological plausibility. This thesis bridges this gap by developing a computational model of predictive representation learning that integrates insights from both fields, building on longstanding theories of predictive processing in neuroscience and recent advances in self-supervised learning algorithms. Through a series of computational studies, we demonstrate that predictive learning principles can effectively drive representation formation in hierarchical networks without requiring biologically implausible mechanisms like backpropagation. We first develop Latent Predictive Learning (LPL), a normative synaptic plasticity rule that learns invariant object representations using temporal persistence. The plasticity rule not only learns useful representations in a biologically plausible manner but also reproduces key empirical phenomena observed in the brain and makes novel experimental predictions. Next, we develop Recurrent Predictive Learning (RePL), a framework that additionally captures dynamic sequential properties. We first analyse its learning dynamics in its original machine learning context and then demonstrate its effectiveness in more biologically relevant settings. These results establish predictive self-supervised learning as a compelling framework for understanding biological learning while offering insights for developing better, and more brain-like artificial intelligence systems.

Acknowledgements

Many people have helped me in the completion of this thesis, in ways both big and small. It is difficult to convey the depth of my gratitude in words, but I will try.

First and foremost, I want to express my profound gratitude to my advisor, Friedemann Zenke, whose scientific guidance and unwavering support have been instrumental throughout this journey. In every sense of the word, he has been a mentor to me. His intellectual rigor, coupled with his patience and encouragement, created an ideal environment for research and growth. I have learned immensely from him, and his trust and belief in my abilities have been a constant source of motivation. I always walked away from our meetings with a clearer understanding of the problem at hand and fresh enthusiasm to tackle it. I count myself fortunate to have had the opportunity to work with him and his lab, which has been a place of intellectual curiosity, exciting science, and genuine camaraderie.

I would like to extend my sincere appreciation to my thesis committee members, Rainer Friedrich and Andrew Saxe, for their invaluable feedback and insightful suggestions over the years. Their perspectives have enriched this work and challenged me to think more deeply about my research.

This thesis would not have been possible without the contributions of my collaborators. I am deeply grateful to Axel Laborieux and Ashena Mohammadi for their brilliance, dedication, and friendship. Their insights and hard work have been crucial to this research, and our collaborative efforts in coding, writing, and debugging together have been some of the most rewarding experiences of my doctoral studies. Special thanks also to Julian Rossbroich for sharing this PhD journey alongside me. I will always treasure our stimulating discussions, spirited debates, and the many moments of laughter we shared.

I am indebted to the members of the Zenke lab, past and present, for creating such a supportive, intellectually stimulating, and, above all, fun environment: Ashena, Axel, Fabian, Jeremias, Julia, Julian, Manvi, Mattias, Peter, and Tengjun. I will include Baba and Anna in this list as honorary members of the lab, as they have been an integral part of the lab's culture and my own experience. I am particularly grateful for all of your support during the difficult months of my injury. Thank you all for the thought-provoking lab meetings and impromptu discussions and for making the lab feel like a second home. I wish I had more time to spend with each of you and look forward to following your scientific journeys in the years to come. I would also like to acknowledge the broader community at the Friedrich Miescher Institute for providing such a welcoming and collaborative atmosphere that has greatly enhanced my research experience. On that note, a special thanks also to my friend Atul, whose sharp analytical mind has frequently helped me navigate research challenges and whose friendship has been invaluable throughout this journey.

My friends have been an essential pillar of support throughout this process. To my friends from home and from Lausanne who have remained constants in my life despite the distance; to my friends in Basel who have made this city feel like home; to my friends from the scientific community who have broadened my intellectual horizons — your encouragement, understanding, and companionship have been invaluable.

Finally, my deepest gratitude goes to my parents, whose unconditional love and support have sustained me not just through this PhD but throughout my life. I would not be where I am today without their sacrifices and guidance. Thank you for instilling in me a love of learning, a sense of curiosity, and a belief in the power of perseverance. This achievement is as much yours as it is mine.

Contents

Chapter I

Introduction

The human brain's capacity to transform unstructured sensory input into coherent percepts is one of its most remarkable capabilities. Our visual system exemplifies this process. The light that falls on our retinas, for instance, is a jumble of impinging photons, with never quite the same pattern of intensity and color, even from moment to moment. The brain interprets these varying and noisy patterns into stable and coherent internal representations of the external world, furnishing us with a seamless and meaningful visual experience [1]. This selective perception and abstraction forms the foundation of our understanding of the world, allowing us to recognize objects, navigate environments, and interact with others.

The internal representations that the brain constructs are useful only insofar as they reflect the statistical regularities and causal structure of the external world [2]. Therefore, in an ever-changing world, these representations are also not static but continuously updated, learned, and re-learned throughout life [3]. Despite a large body of knowledge on the physiological mechanisms of learning in the brain [4–9] and many theoretical models attempting to explain them [10–17], the precise computational principles governing how the brain establishes and updates its representations remain largely unknown.

Classical phenomenological [14], mechanistic [3, 11, 12], and normative [13, 16] models of learning in the brain have provided valuable insights into many aspects of learning and memory. Challenges remain, however, in bridging the gap between these models and the complex, high-dimensional, and hierarchical representations that the brain constructs. Concurrently, the field of artificial intelligence (AI) has made rapid and significant strides in building artificial deep neural networks (DNNs) for solving perceptual tasks [18]. DNNs routinely learn complex, hierarchical representations from high-dimensional data, proffering them as powerful computational tools for building and testing theories of representation learning [19–21]. However, the typical learning algorithms used in AI are not biologically plausible, precluding straightforward insights into the brain's learning mechanisms.

This thesis aims to develop and examine a computational model of representation learning that bridges these gaps. Specifically, we build on long-standing ideas of predictive learning in the brain [15, 16] and recent advances in self-supervised learning (SSL) algorithms for training DNNs [22–29] to develop a *predictive* representation learning model that is both biologically plausible and demonstrably effective for solving perceptual tasks in artificial systems.

The remainder of this chapter provides a brief review of the relevant background and rationale for this work. Section 1.1 introduces the basic principles of learning in the brain, discussing synaptic plasticity and the normative principles of predictive processing. Section 1.2 provides an overview of learning in machines, focusing on deep learning, DNNs, and SSL. Section 1.3 motivates predictive SSL as a model of learning in the brain and outlines the goals of this work. Finally, Section 1.4 provides an outline of the thesis.

1.1 Learning in the brain

The brain processes information through the coordinated activity of billions of neurons connected by trillions of synapses. Neurons are fascinating objects in their own right. These cells are highly specialized for inter-cellular communication, with cell bodies that can generate electrical activity, branching projections known as dendrites for collecting electrical signals from thousands of other neurons, and long, thin projections called axons that carry signals to yet other neurons. The unit of neuronal communication is the action potential, an electrical impulse or “spike" that travels down a neuron’s axon and triggers the release of chemicals known as neurotransmitters at the synapse, the junction between this neuron and the next. The release of neurotransmitters, in turn, triggers electrical activity in the receiving neuron with a strength that depends on the amount of neurotransmitter released and various properties of the pre- and post-synaptic neurons. This effective communication strength between neurons is known as synaptic efficacy or synaptic strength. Finally, when a neuron receives a signal from another neuron weighted by the synaptic strength of each connection, it integrates these signals, and if the sum exceeds a certain threshold, it fires its own electrical impulse that is then further transmitted to other neurons.

The synaptic strengths between neurons are not fixed but can change over time in response to experience, a process known as synaptic plasticity. Synaptic plasticity is thought to be the primary mechanism underlying learning and memory in the brain [4, 6], although other physiological processes such as neurogenesis [5], homeostatic regulation [8], and epigenetic modifications [9] also play important roles.

Synaptic plasticity is a complex and multifaceted process. It is known to be influenced by a wide range of factors, such as the types of neurons involved, the strength of pre- and post-synaptic activity [3, 4], the timing of pre- and post-synaptic spikes [6, 30], non-local neuromodulatory chemical signals [31], and many other factors, all ultimately related to some underlying behavioural goal. The sheer scale and complexity of these processes have called forth a wide range of theoretical and computational models to help understand them [10–17].

Models of neuronal networks To study the brain's learning mechanisms with theory and computation, it is necessary to work with mathematical abstractions of the brain's neurons (and synapses). These can range from fully biophysically detailed models of single neurons [32] to abstract models of binarized neurons that capture very minimal features of neuronal communication and plasticity [12]. For any given problem, the choice of model depends on the level of detail required for the specific question at hand. In this thesis, we mostly focus on abstract models of rate-based neurons where the activity of a neuron is represented as a continuous variable encoding the average firing rate of the neuron over time. These models are computationally efficient and used widely in theoretical neuroscience [33, 34] and machine learning [18]. Their computational efficiency allows us to study large-scale networks of neurons and explore the emergent properties of these networks, something that is crucial for studying the learning of high-dimensional representations. However, in order to make contact with experimental data, it is often necessary to consider more detailed neuron models. Here, integrate-and-fire models of spiking neurons [34, 35] provide a good middle ground between rate-based models and biophysically detailed models, capturing the temporal dynamics of neuronal activity while remaining computationally tractable. We use these models in Chapter II to study learning in spiking neural networks [36, 37].

1.1.1 Models of synaptic plasticity

Models of synaptic plasticity postulate learning rules that are typically, but not always, expressed as a mathematical equation that describes how the synaptic strength w is updated as a general function g of pre-synaptic activity x, post-synaptic activity z, and a range of other factors Φ:

$$\Delta w = g(x, z, \Phi) \quad . \tag{1.1}$$

The exact form of the learning rule and the factors Φ can vary widely depending on the specific model and learning scenario [38]. Donald Hebb's famous principle: "neurons that fire together, wire together" [3], is perhaps the most well-known learning rule in neuroscience. Hebbian learning posits that synapses between neurons that fire together are strengthened, whereas those that do not are weakened. In its simplest form, this rule can be expressed as:

$$\Delta w = \eta x z \quad , \tag{1.2}$$

where η is the learning rate. In the Hopfield network model with binary neurons [12], the learning rule leads to associative memory storage. It is important to note that the rule in its basic form is generally unstable and requires careful modifications to prevent runaway growth of synaptic strengths [39, 40] in all but the simplest network models. With modifications that include a learning threshold, the rule has also been shown to be capable of explaining the emergence of neuronal selectivity with the Bienenstock-Cooper-Munro (BCM) rule [11], learning principal components with Oja's rule [41], and learning independent components with ICA [13]. However, Hebbian learning is only one of many learning rules that have been proposed over the years to capture the diverse range of observed synaptic plasticity phenomena [38]. The development of plasticity models that can explain the full range of observations remains an open challenge in neuroscience.

Normative models of synaptic plasticity Models of synaptic plasticity can be broadly divided into three categories depending on their level of abstraction: mechanistic, phenomenological, and normative [42]. Mechanistic models attempt to describe the detailed biophysical mechanisms that give rise to synaptic changes. Phenomenological models, on the other hand, focus on describing experimental observations as concisely as possible. Lastly, normative models attempt to explain synaptic changes in terms of some underlying computational goal, such as maximizing information transfer or minimizing prediction errors. The boundaries between these categories are not always sharp, and many models can be classified under more than one category. For example, the BCM theory [11] is a phenomenological model that can be derived from a normative principle of maximizing information content [13]. While valuable for understanding specific phenomena, mechanistic and phenomenological models are often difficult to relate to broader behavioural goals. Normative models, on the other hand, provide a clear link between synaptic plasticity and the brain's computational goals but are typically abstract and require additional constraints to make them biologically plausible [42]. However, their promise of a more principled and general understanding of biological learning motivates the top-down approach to modeling that is employed in this thesis. The particular normative principle that we will focus on falls under the general theoretical framework of predictive processing.

1.1.2 Predictive processing

The brain has been variously described as a prediction machine [15], a Bayesian inference engine [43], and a free-energy minimizer [44]. A common thread running through these descriptions is the idea that the brain maintains an internal model of the world, uses it to make predictions of future sensory inputs, and compares them with actual sensory inputs to update the brain's belief about the current state of the world. This process is thought to encompass perception, action, and learning in the brain [45], explaining perception as a process of probabilistic inference over the causes of sensory inputs [43], action as a process of active inference to minimize discrepancy between predicted and desired outcomes [46], and learning as a process of updating the internal model to minimize prediction errors [15, 44]. Prediction errors are thought to arise when the actual sensory input deviates from the predicted input, driving the brain to update its internal model so as to make better predictions in the future.

Prediction errors and learning Prediction errors consistent with this predictive processing framework have been observed in the visual cortex [15, 47, 48], auditory cortex [49], the motor system [50], and across brain regions [51, 52]. Computational models of predictive processing that learn based on prediction error minimization have been elaborated to varying levels of detail [15, 53, 54] but have thus far failed to show a clear capability to learn complex representations of high-dimensional data streams in hierarchical networks [55]; but see [56]. While a fully elaborated circuit-level model of predictive processing that explains the plethora of observed phenomena [48–52, 57–59] is still elusive, prediction nonetheless remains a compelling normative principle for explaining learning and plasticity.

1.2 Learning in machines

Machine learning, and in particular deep learning, has revolutionized AI and, increasingly, the fundamental ways in which we interact with technology [60]. Originating from attempts to implement basic logical functions and pattern recognition in networks of simple artificial neurons [61, 62], deep learning has evolved into a field that has successfully replicated at least some narrow capabilities of the brain, such as visual object recognition [63], speech recognition [64], and natural language processing [60]. Deep learning relies on deep neural networks (DNNs) to process information through networks of interconnected units, similar to the brain. However, DNNs use computational units that are vastly simpler than biological neurons and rely on gradient-based optimization of objective functions to learn the strengths of inter-unit connections that define the network's behavior.

Artificial deep neural networks More specifically, DNNs are composed of layers of artificial neurons that transform input data through a series of nonlinear transformations [18]. Each neuron receives inputs $\boldsymbol{x} \in \mathbb{R}^N$ from a previous layer of neurons or raw sensory inputs, computes a weighted sum of these inputs, and applies a nonlinear activation function to produce an output z_i for each neuron i in the layer. Concretely, the output of neuron i can be expressed as:

$$z_i = f\left(\sum_{j=1}^{N} w_{ij} x_j + b_i\right) \quad , \tag{1.3}$$

where w_{ij} is the weight of the connection strength from neuron j in the previous layer to neuron i, b_i is a bias term for neuron i, and f is a nonlinear activation function, typically a rectified linear unit (ReLU) [65] or a sigmoid function [18]. The vector of outputs $\boldsymbol{z} \in \mathbb{R}^N$ from all neurons in the layer is, in turn, passed as input to the next layer of neurons or used as the final output of the network. By stacking multiple layers of neurons, DNNs can potentially represent increasingly complex and abstract features of the input data. This hierarchical organization into many layers of neurons, ranging from tens to hundreds or even thousands, gives rise to the moniker "deep" learning.

Gradient-based optimization Very generally, the weights of the connections between neurons in a DNN are learned by minimizing a task-specific objective function that quantifies the discrepancy between the network's output $\boldsymbol{y}$ and the desired output $\boldsymbol{y}^*$ for a given input $\boldsymbol{x}$. This discrepancy is also known as the loss function $\mathcal{L}$. For example, a simple loss function for a regression task could be the mean squared error:

$$\mathcal{L}(\boldsymbol{y}, \boldsymbol{y}^*) = \frac{1}{2}\sum_{i=1}^{N}(y_i - y_i^*)^2 \quad . \tag{1.4}$$

The weights (and biases) are updated in the direction that most reduces the loss function on a set of training examples by computing the gradient of the loss with respect to the weights and adjusting the weights in the opposite direction of the gradient. Concretely, the learning updates can be written as:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L} \quad , \tag{1.5}$$

where $\boldsymbol{\theta}$ is the parameter vector consisting of all weights and biases in the network concatenated together, η is the learning rate that determines the size of the update, and $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ is the gradient of the loss with respect to the parameters averaged over the training examples. Over many iterations of this optimization process on a large dataset, the network's parameters reach a configuration that extracts useful representations of the input data and maximizes the network's performance on the task [66]. The gradient for the parameters in multilayer networks is typically computed using the backpropagation algorithm [67], essentially an efficient application of the chain rule of calculus to neural networks. Gradient-descent optimization using backpropagation has been the workhorse of deep learning for many years and has been instrumental in its success [18].

1.2.1 Deep learning and the brain

Despite its successes in solving complex perceptual and learning tasks, deep learning is considered to be a poor model of the brain [19] for several reasons. First, the neurons in DNNs are much simpler than biological neurons and lack many of the complex biophysical properties that make biological neurons so interesting [68]. Second, many aspects of the backpropagation algorithm used in DNNs are thought to be biologically implausible, but the search for biologically plausible approximations of backpropagation is an active area of research [69–72]. Third, the representations learned by DNNs are often difficult to interpret and do not always match the representations found in the brain [20]. Other criticisms include the challenges DNNs still face in continual learning [73], generalizing to new tasks [74], and learning from small amounts of data [75].

Despite these differences, DNNs do share some key computational principles with neuronal networks in the brain. Both brains and DNNs process information through networks of interconnected units, represent information in a distributed manner across these units, and learn by adjusting the strengths of inter-unit connections in response to experience. These conceptual similarities, combined with their unmatched ability to learn complex perceptual tasks, proffer DNNs as powerful computational tools for building and testing theories of brain function [19–21]. In particular, generalized computational principles [1] of representational geometry [76], representation learning [66], and gradient-based optimization of objective functions [77] that putatively apply to populations of both artificial and biological neurons can more easily be studied in artificial systems and later tested in biological systems. We take this general perspective in this thesis and use DNNs as testing grounds for developing and examining biologically plausible models of representation learning.

1.2.2 Self-supervised learning

DNNs are typically trained with learning objectives obtained in one of three ways: supervised learning [18], reinforcement learning [78], and unsupervised learning [79]. Since most natural experience arrives without associated rewards (reinforcement learning) or explicit supervisory signals (supervised learning), it is likely that learning in the brain is driven predominantly by unsupervised learning [80]. SSL is a class of unsupervised learning algorithms for training DNNs that leverage the structure of the data itself to define learning objectives. These objectives can involve reconstruction of missing parts of the input [29, 81–84], instance discrimination between similar vs dissimilar pairs of inputs [23, 26, 85], predicting the future from the past [86, 87], predicting the next word in a sentence [60, 88], and many other pretext tasks [24, 25, 89]. The key idea is that the structure of the data itself provides a supervisory signal that can be employed to learn useful representations.

This general framework can be divided into two broad categories: input-space and representation-space SSL. Input-space SSL focuses on reconstructing or predicting parts of the raw sensory input [81, 82], whereas representation-space SSL operates in the internal representation space. Instead of reconstructing inputs, representation-space methods aim to directly optimize some measure of representation quality. For example, contrastive learning [22, 23] aims to make representations of related inputs similar while explicitly pushing unrelated ones apart. Other non-contrastive representation-space methods achieve a similar effect by optimizing other measures of representation quality based on entropy maximization [28, 90, 91], prediction [24, 25, 29], and manifold capacity [92]. Such methods have demonstrated impressive performance, sometimes rivaling supervised approaches, despite not using any labeled data [93]. Representation-space SSL methods have been recently consolidated into the broader umbrella of Joint-Embedding Predictive Architectures (JEPAs) [94], a blueprint for building autonomous intelligent agents that predictively learn a world model for guiding behaviour [95, 96].

Representational collapse A core challenge in representation-space SSL is the phenomenon of representational collapse. This occurs when the model learns to ignore the input data and instead find trivial solutions that satisfy the learning objective. For example, a model trained to output similar representations for similar inputs can learn to output the same representation for all inputs, a solution that satisfies the learning objective but is not useful for downstream tasks. Most representation-space SSL methods are susceptible to this problem, and each successful algorithm implements specific strategies to avoid it [89]. On the one hand, contrastive learning methods [22, 23] use objective functions that explicitly push unrelated inputs apart in representation space, thereby preventing the model from collapsing to trivial solutions. A key step in this process is the sampling of negative examples, which are inputs that are unrelated to a given input. On the other hand, non-contrastive methods [24, 25, 28, 29, 90, 91] sidestep the requirement for negative samples by using other strategies. One strategy is to maximize the entropy of the representations with regularization objectives [28, 90]. Another is to leverage architectural and objective function asymmetries [24, 25, 29]. Specifically, these methods use an explicit predictor network to predict representations across some transformation of the input, as well as a stop-gradient operation to prevent adapting the target representations during training.

1.2.3 Predictive self-supervised learning

Temporal persistence and regularity across contiguous moments in time are ubiquitous features of our sensory experience of the world and natural sources of self-supervisory learning signals. Indeed, several SSL models have been developed on precisely this idea, employing both input-space [86] and representation-space [22, 87, 97] frameworks using both contrastive [22] and non-contrastive [87, 97] objectives. These constitute an exciting set of models that successfully implement the predictive learning principle to learn useful representations, unlike previous models that failed to do so [55].

Although most SSL models are built on pretext tasks that do not involve predicting the future, they can be readily adapted to do so. For example, the most common pretext task in image-based SSL is augmentation-based instance discrimination [23, 26]. Here, the DNN is trained to output similar representations for augmented versions of the same image (positive pairs) and dissimilar representations for augmented versions of different images (negative pairs). The models constructed for this task can be easily adapted to predictive tasks by simply defining temporally nearby images in a video as positive pairs and temporally distant images as negative pairs. Therefore, the theoretical and empirical insights into SSL can, in many cases, be adopted into predictive learning models regardless of the original pretext task.

Predictive self-supervised learning and the brain From a biological perspective, non-contrastive representation-space methods are more attractive candidates for modeling predictive learning in the brain for several reasons. First, input-space models would entail that the brain computes prediction error signals at the sensory periphery, for which there is little evidence. On the contrary, prediction errors have been found throughout the cortical hierarchy [7, 48, 57]. Moreover, representation-space methods can focus on learning high-level abstract features as opposed to low-level sensory details that are often irrelevant for higher-level cognitive tasks [98]. Finally, contrastive methods require negative samples, which, in a temporal prediction setup, requires comparing representations that are far apart in time, making the overall learning signal temporally non-local. For these reasons, we adopt non-contrastive representation-space frameworks for the development of predictive learning models in this thesis.

1.3 Motivation

Taking a general perspective, this thesis aims to develop and examine a computational model of predictive representation learning that is both biologically plausible and functionally effective for training DNNs. At a high level, the notion of predictive learning posits that the brain learns by constantly trying to predict the future, thereby developing progressively more useful internal representations that capture the statistical redundancies and predictive factors in sensory inputs [15, 16, 99]. This idea is conceptually appealing because it exploits temporal contiguity, a ubiquitous feature of natural sensory experience, as an intrinsic learning signal. The focus on prediction as a learning signal is timely for two reasons. First, several lines of recent experimental work have shown that temporal contiguities in sensory inputs *do* drive unsupervised learning of cortical [47, 100, 101] and hippocampal [102] representations in mammalian brains. This provides a strong empirical basis for the core idea. Second, recently developed SSL algorithms for training DNNs share deep links with the predictive learning principle and have achieved competitive results on challenging perception tasks [22–29]. As a result, we now know how to train DNNs on various predictive tasks with a growing understanding of the theoretical underpinnings and engineering principles that make them work [89, 94]. Predictive learning therefore carries, apart from its conceptual appeal, growing empirical support and a large body of computational work that can be leveraged to build and test potential models. Taken together, these motivate a serious investigation of predictive SSL as a description of biological learning. This thesis aims to initiate such an investigation and, in the process, explore the applications of this idea in designing artificial learning systems.

1.4 Outline of the thesis

The remainder of this thesis is organized as follows.

Chapter II presents the Latent Predictive Learning (LPL) rule, a novel biologically plausible normative plasticity rule that leverages temporal persistence to learn invariant object representations in hierarchical sensory networks. Building on variance-regularized representation-space SSL methods, LPL combines classical Hebbian plasticity models like the BCM rule [11] with a predictive learning mechanism. LPL can learn invariant representations in hierarchical network models without the need for backpropagation, is consistent with diverse plasticity phenomena, and also makes new experimental predictions. Overall, this work constitutes a significant step towards building a biologically plausible model of predictive learning that leverages the principles of SSL. However, the LPL rule is geared towards learning invariant features and therefore limited in its ability to learn dynamic properties of sequential data. The next two chapters incrementally address this limitation.

In Chapters III and IV, we consider a different, more versatile class of non-contrastive representation-space SSL models that use an asymmetric predictor-based architecture and a stop-gradient in the loss function in order to avoid representational collapse. We first investigate the learning dynamics of these models in Chapter III in order to understand how exactly they avoid representational collapse. Then, in Chapter IV, we propose a novel predictive learning framework based on these models and investigate its representation-learning capabilities on sequential data.

Chapter III presents our analysis of the learning dynamics in asymmetric predictor-based SSL models. We develop a powerful eigenspace analysis that clarifies their learning dynamics and reveals an implicit variance-regularization mechanism that prevents representational collapse, building a conceptual bridge between the different collapse-avoidance strategies. We further identify a detrimental anisotropy in the learning dynamics of these models and propose a possible solution.

Chapter IV presents Recurrent Predictive Learning (RePL), a novel predictive learning framework based on the asymmetric predictor-based SSL models. We show that RePL learns dynamic object representations that include but go beyond temporal persistence, unlike LPL, which is limited to learning invariant features.

Finally, chapter V concludes the thesis by summarizing the main contributions and discussing future directions for this work.

Chapter II

The combination of Hebbian and predictive plasticity learns invariant object representations in deep sensory networks

Authors: Manu Srinath Halvagal, Friedemann Zenke

A version of this chapter has been published as:

M. S. Halvagal and F. Zenke. "The combination of Hebbian and predictive plasticity learns invariant object representations in deep sensory networks". In: *Nature neuroscience* 26.11 (2023), pp. 1906–1915.

Author contributions: FZ conceived the study. MSH and FZ developed the theory. MSH wrote DNN code, performed simulations, and analysis. FZ developed spiking neural network (SNN) code and performed simulations. MSH and FZ wrote the manuscript.

Abstract

Recognizing objects from sensory stimuli is essential for survival. To that end, sensory networks in the brain must form object representations invariant to stimulus changes, such as size, orientation, and context. While Hebbian plasticity is known to shape sensory networks, it fails to create invariant object representations in computational models, raising the question of how the brain achieves such processing. Here we show that combining Hebbian plasticity with a predictive form of plasticity leads to invariant representations in deep neural network models. We derive a local learning rule that generalizes to spiking neural networks and naturally accounts for several experimentally observed properties of synaptic plasticity, including metaplasticity and spike-timing-dependent plasticity (STDP). Finally, our model accurately captures neuronal selectivity changes observed in the primate inferotemporal cortex in response to altered visual experience. Thus, we provide

a plausible normative theory emphasizing the importance of predictive plasticity mechanisms for successful representation learning.

2.1 Introduction

Recognizing invariant objects and concepts from diverse sensory inputs is crucial for perception. Watching a dog run evokes a series of distinct retinal activity patterns that differ substantially depending on the animal's posture, lighting conditions, or visual context (Fig. 2.1a). If we looked at a cat instead, the resulting activity patterns would be different still. That we can effortlessly distinguish dogs from cats is remarkable. It requires mapping *entangled* input patterns, which lie on manifolds that "hug" each other like crumpled-up sheets of paper, to *disentangled* neuronal activity patterns, which encode the underlying factors so downstream neurons can easily read them out [104]. Such transformations require deep sensory networks with specific network connectivity shaped through experience-dependent plasticity (Fig. 2.1b). However, current data-driven plasticity models fail to establish the necessary connectivity in simulated deep sensory networks. At the same time, supervised machine learning algorithms *do* yield suitable connectivity [18] in DNNs that further reproduce essential aspects of the representational geometry of biological neural responses [105, 106]. This resemblance proffers DNNs as potential tools to elucidate neural information processing in the brain [20, 76].

Unfortunately, standard deep learning methods are difficult to reconcile with biology. On the one hand, they rely on backpropagation, an algorithm considered biologically implausible, although neurobiology may implement effective alternatives [20, 71, 107–109]. On the other hand, humans and animals cannot learn through strong label-based supervision, as this would require knowledge of a label for *every* input pattern.

Here, self-supervised learning (SSL), a family of unsupervised machine learning algorithms, may offer a remedy. SSL does not need labeled data but instead relies on prediction, a notion also supported by neurobiology [15, 57, 100, 101, 111, 112]. Prediction can happen in the input space by, for instance, reconstructing one part of an image from another, as for autoencoders [82], or by predicting the next word in a sentence, as done in language models. Alternatively, prediction can occur in latent space by requiring internal representations of related inputs to predict each other [22, 23]. Latent space prediction is more compelling from a neuroscience perspective since it does not require an explicit decoder network that computes prediction errors at the input, i.e., the sensory periphery, for which there is little experimental support. Instead, latent prediction errors are computed locally or at network outputs (cf. Fig. 2.1) and drive learning by "pulling" together related internal representations for stimuli that frequently occur close in time (Fig. 2.1c), similar to slow feature analysis (SFA) [16, 113].

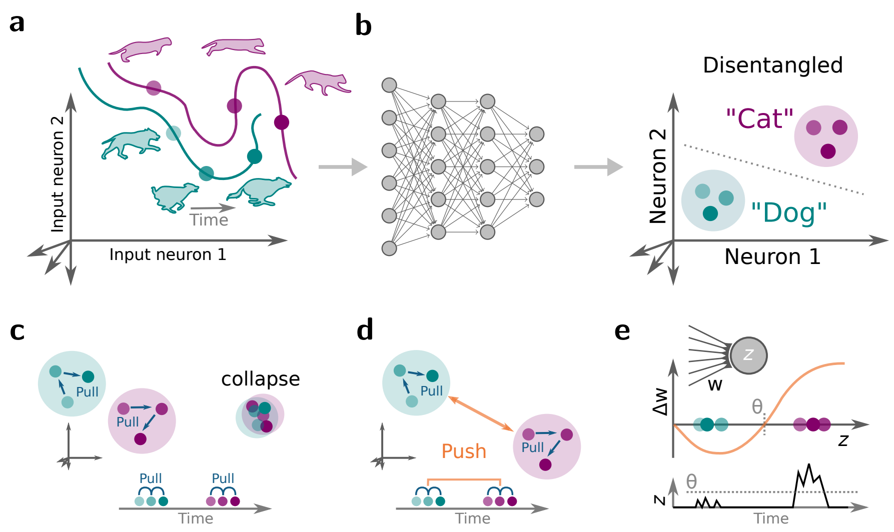


FIGURE 2.1: **Disentangling sensory stimuli with plastic neural networks.** **(a)** Schematic of an evoked response in sensory input neurons. The neuronal response patterns for distinct stimuli correspond to points in a high dimensional space spanned by the neuronal activity levels. The response patterns from different stimulus classes, e.g., cats and dogs, form a low-dimensional manifold in the space of all possible response patterns. Generally, different class manifolds are entangled, which means that the stimulus identity cannot be readily decoded from a linear combination of the neuronal activities. **(b)** Sketch of a deep neural network (left) that transforms inputs into disentangled internal representations that are linearly separable (right). **(c)** Schematic of how predictive learning influences latent representations (left). Learning tries to "pull" together representations that frequently co-occur close in time (bottom). However, without opposing forces, such learning dynamics lead to representational "collapse" whereby all inputs are mapped to the same output and thereby become indistinguishable (right). **(d)** SSL avoids collapse by adding a repelling force that acts on temporally distant representations that are often semantically unrelated. **(e)** Plot of postsynaptic neuronal activity z over time (bottom) and the BCM learning rule (top; [11, 110]) which characterizes the sign and magnitude of synaptic weight change Δw as a function of postsynaptic activity z. Notably, the sign of plasticity depends on whether the evoked responses are above or below the plasticity threshold θ. Using the example of Neuron 1 in panel (b), the BCM rule potentiates synapses that are active when a "Cat" stimulus is shown, whereas "Dog" stimuli induce long-term depression (LTD). This effectively pushes the evoked neuronal activity levels corresponding to both stimuli away from one another, and thereby prevents representational collapse.

However, a major issue with this strategy is that without any forces opposing this representational pull, such learning inevitably leads to "representational collapse," whereby all inputs are mapped to the same internal activity pattern which precludes linear separability (Fig. 2.1c). One typical solution to this issue is to add forces that "push" representations corresponding to *different* unrelated stimuli away from one another (Fig. 2.1d). This is usually done by invoking so-called "negative samples," which are inputs that do not frequently occur together in time. This approach has been linked to biologically plausible three-factor learning rules [114, 115], but it requires constantly switching the sign of plasticity depending on whether two successive inputs are related to each other or not. Yet, it is unknown whether and how such a rapid sign switch is implemented in the brain.

Another possible solution for avoiding representational collapse without negative samples is to prevent neuronal activity from becoming constant over time, for instance, by maximizing the variance of the activity [28]. Interestingly, variance maximization is a known signature of Hebbian plasticity [35, 41], which has been found ubiquitously in the brain [116, 117]. While Hebbian learning is usually thought of as the primary plasticity mechanism rather than playing a supporting role, Hebbian plasticity alone has had limited success at disentangling representations in DNNs [20, 118, 119].

This article introduces Latent Predictive Learning (LPL), a conceptual learning framework that overcomes this limitation and reconciles SSL with Hebbian plasticity. Specifically, the local learning rules derived within our framework combine a plasticity threshold, as observed in experiments (Fig. 2.1e; [110, 116, 120–122]), with a predictive component, inspired by SSL and SFA, that renders neurons selective to temporally contiguous features in their inputs. When applied to the layers of deep hierarchical networks, LPL yields disentangled representations of objects present in natural images without requiring labels or negative samples. Crucially, LPL effectively disentangles representations as a local learning rule without requiring explicit spatial credit assignment mechanisms. Still, credit assignment capabilities can further improve its effectiveness. We demonstrate that LPL captures central findings of unsupervised visual learning experiments in monkeys and in SNNs and naturally yields a classic STDP window, including its experimentally observed firing-rate-dependence [116]. These findings suggest that LPL constitutes a plausible normative plasticity mechanism that may underlie representation learning in biological brains.

2.2 Results

To study the interplay of Hebbian and predictive plasticity in sensory representation learning, we derived a plasticity model from an SSL objective function that is reminiscent of and extends the classic BCM learning rule [11, 110] (Methods; Supplementary Note A.2).

According to our learning rule, the temporal dynamics of a synaptic weight W_j are given by

$$\frac{\mathrm{d}W_j}{\mathrm{d}t}(t) = \eta x_j(t) f'(a(t)) \left(\underbrace{-\frac{\mathrm{d}z(t)}{\mathrm{d}t}}_{\text{predictive}} + \underbrace{\frac{\lambda}{\sigma_z(t)^2}\left(z(t) - \bar{z}(t)\right)}_{\text{Hebbian}} \right) \tag{2.1}$$

where η is a small positive learning rate, $x_j(t)$ denotes the activity of the presynaptic neuron j, $z(t) = f(a(t))$ is the neuronal activity with the activation function f, and the net input current $a(t) = \sum_k W_k x_k(t)$. We call the first term in parentheses the predictive term because it promotes learning of slow features [16, 113] by effectively "pulling together" postsynaptic responses to temporally consecutive input stimuli. Importantly, it cancels when the neural activity does not change and, therefore, accurately *predicts* future activity. In the absence of any additional constraints, the predictive term leads to collapsing neuronal activity levels [16]. In our model, collapse is prevented by the Hebbian term in which $\bar{z}(t)$, the running average of the neuronal activity appears, which is reminiscent of BCM-theory [11, 110]. Its strength further depends on an online estimate of the postsynaptic variance of neuronal activity $\sigma_z^2(t)$. This modification posits an additional metaplasticity mechanism controlling the balance between predictive and Hebbian plasticity depending on the postsynaptic neuron's past activity.

To make the link to BCM explicit, we rearrange the terms in Eq. (2.1) to give:

$$\frac{\mathrm{d}W_j}{\mathrm{d}t}(t) = \eta\lambda \frac{x_j(t) f'(a(t))}{\sigma_z(t)^2} \left(z(t) - \underbrace{\left(\bar{z}(t) + \frac{\sigma_z(t)^2}{\lambda} \frac{\mathrm{d}z(t)}{\mathrm{d}t} \right)}_{\text{Sliding threshold } \Theta(t)} \right) \tag{2.2}$$

where $\Theta(t)$ corresponds to a time-dependent sliding plasticity threshold (cf. Fig. 2.1e). While the precise shape of the learning rule depends on the choice of neuronal activation function, its qualitative behavior remains unchanged as long as the function is monotonic (Supplementary Fig. A.1). Despite the commonalities, however, there are three essential differences to the BCM model. First, in our model, the threshold depends only linearly on $\bar{z}(t)$ (Supplementary Fig. A.1b), whereas in BCM, the threshold is typically a supralinear function of the moving average $\bar{z}(t)$. Second, the added dependence on the predictive term $-\frac{\mathrm{d}z}{\mathrm{d}t}$ constitutes a separate mechanism that modulates the plasticity threshold depending on the rate-of-change of the postsynaptic activity (Supplementary Fig. A.1c,d). Third, our model adds a variance-dependence that has diverse effects on the sliding threshold when the neuronal output does not accurately predict future activity and, thus, changes rapidly. We will see that these modifications are crucial to representation learning from the temporal structure in sensory inputs. Because the predictive term encourages neurons to predict future activity at their output, and thus in latent space rather than the input space, we

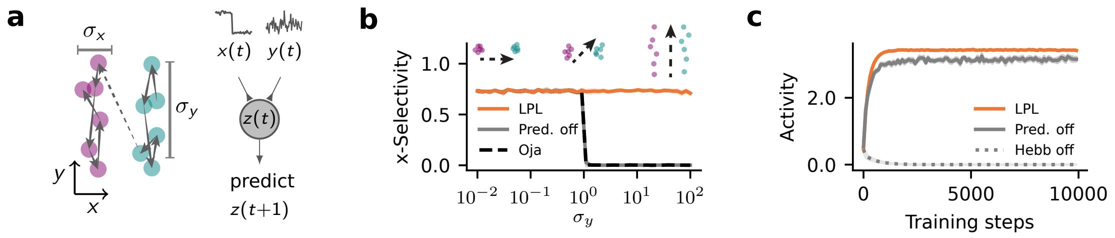


FIGURE 2.2: **LPL learns predictive features. (a)** Illustration of the two-dimensional synthetic data generating process. Consecutive data points stay within the same cluster separated along the x-direction and are drawn independently from the corresponding normal distribution centered in that cluster (left). These data are fed into a linear neuron that learns via LPL (right). **(b)** Cluster selectivity of the features learned by LPL with and without the predictive term and by Oja's rule for different values of σ_y. By varying σ_y, we obtain a family of sequences with different amplitudes of within-cluster transitions (top). LPL selects temporally contiguous features and therefore ensures that the neuron always becomes selective to cluster identity. Oja's rule finds PC1, the direction of highest variance, which switches to the noise direction at $\sigma_y = 1$. LPL without the predictive component shows the same behavior. Selectivity values were averaged over ten random seeds. The shaded area corresponds to one SD. **(c)** Mean output activity of the neuron over training time for $\sigma_y = 1$ under different versions of LPL. LPL initially increases its response and saturates at some activity level, even when the predictive term is disabled. However, without the Hebbian term, the activity collapses to zero.

refer to Eq. (2.1) as the Latent Predictive Learning (LPL) rule.

2.2.1 Invariant features in temporal data

To investigate the functional advantages of LPL over BCM and other classic Hebbian learning rules (Supplementary Note A.3), we designed a synthetic two-dimensional learning task in which we parametrically controlled the proportion of predictable changes between subsequent observations (Fig. 2.2a; Methods). The data sequence consisted of noisy inputs from two clusters separated along the x-axis. Consecutive inputs had a high probability of staying within the same cluster, thus making cluster identity a temporally contiguous feature. By varying the noise amplitude σ_y in the y-direction, we controlled the amount of unpredictable changes. We simulated a single rate neuron with different datasets for varying σ_y, while the two input connections were plastic and evolved according to the LPL rule (Eq. (2.1)) until convergence. We then measured neuronal selectivity to cluster identity (Methods).

We found that LPL rendered the neuron selective to the cluster identity for a large range of σ_y values (Fig. 2.2b). However, without the predictive term, the selectivity to cluster identity was lost for large σ_y values. This behaviour was expected because omitting the predictive term renders the learning rule purely Hebbian, which biases selectivity toward directions of high variance. To illustrate this point, we repeated the same simulation with Oja's rule, a classic Hebbian rule that finds the principal component in the input, and found similar qualitative behaviour. Thus LPL behaves fundamentally different from purely Hebbian rules, by selecting predictable features in the input.

To confirm that the Hebbian term is essential for LPL to prevent representational collapse, we simulated learning without the Hebbian term (cf. Eq. (2.1)). We observed that the neuron's activity collapses to zero firing rate as expected (Fig. 2.2c). Conversely, learning with the Hebbian term but without the predictive term did not result in collapse. Therefore, LPL's Hebbian component is essential to prevent activity collapse.

Moreover, Hebbian plasticity needs to be dynamically regulated to prevent run-away activity [40]. In LPL this regulation is achieved by inversely scaling the Hebbian term by a moving estimate of the variance of the postsynaptic activity $\sigma_z^2(t)$. Without this variance-modulation, neural activity either collapsed or succumbed to runaway activity depending on which term was dominant (Supplementary Note A.4). Either case precluded the neuron from developing cluster selectivity. We verified that these findings generalized to higher-dimensional tasks with more complex co-variance structure (Supplementary Note A.5). Hence, the combination of the predictive with variance-modulated Hebbian metaplasticity in LPL is needed to learn invariant predictive features independently of the co-variance structure in the data.

2.2.2 Disentangled representations in deep hierarchical networks

As we move through the world, we see objects, animals, and people under different angles and contexts (Fig. 2.3a). Therefore, objects themselves constitute temporally contiguous features in normal vision. We thus wondered whether training an artificial DNN with LPL on image sequences with such object permanence results in disentangled representations. To that end, we built a convolutional DNN model in which we "stacked" layers whose synaptic connections evolved according to the LPL rule. Additionally we included a term to decorrelate neurons within each layer. Inhibitory plasticity presumably plays this role in biological neural networks [123–126]. LPL was implemented in a "layer-local" manner, meaning that there was no backpropagation through layers (Methods).

To simulate temporal sequences of related visual inputs, we generated pairs of images sampled from a large dataset, by applying different randomized transformations (Supplementary Fig. A.2; Methods). We trained our network model on these visual data until learning converged and evaluated the linear decodability of object categories from the learned representations using a separately trained linear classifier.

We found that in networks trained with LPL, object categories could be linearly decoded at the output with an accuracy of $(63.2 \pm 0.3)\%$ (Fig. 2.3b; Table 2.1), suggesting that the network has formed partially disentangled representations (Supplementary Fig. A.3). To elucidate the roles of the different learning rule components, we conducted several ablation

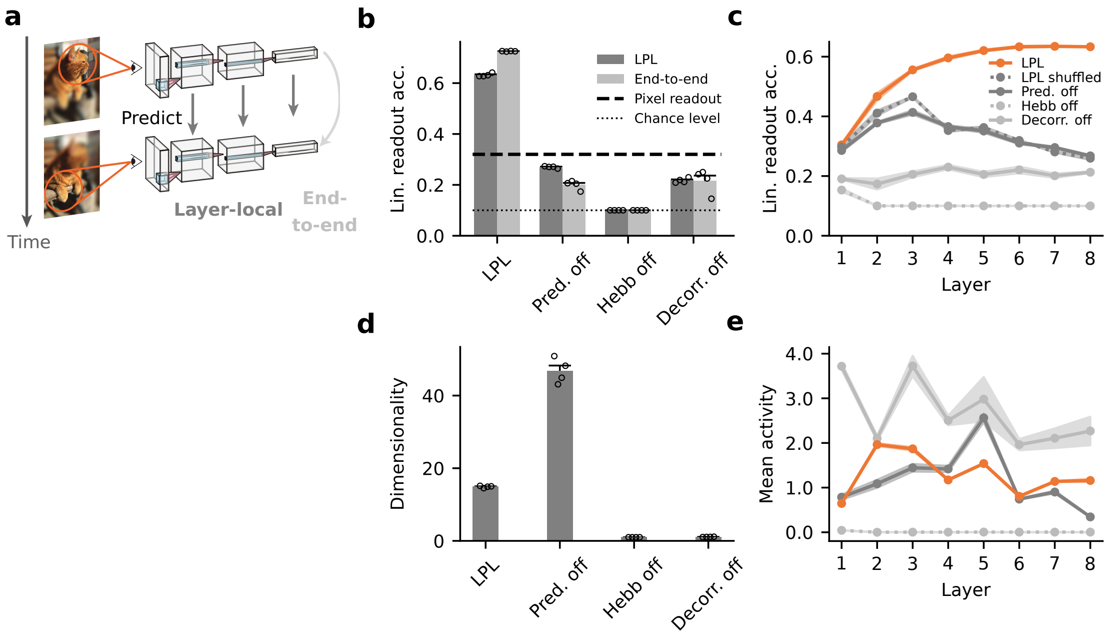


FIGURE 2.3: **LPL disentangles representations in DNNs.** **(a)** Schematic of the DNN trained using LPL. We distinguish two learning strategies: Layer-local and end-to-end learning. In layer-local LPL, each layer's learning objective ($\mathcal{L}_i$) is to predict representations within the same layer, whereas end-to-end training takes into account the output layer representations only ($\mathcal{L}_{\text{out}}$), and updates hidden layer weights using backpropagation. **(b)** Linear readout accuracy of object categories decoded from representations at the network output after training $n = 4$ networks independently on natural image data (STL-10; see Methods for details) with different learning rules in layer-local (dark) as well as the end-to-end configuration (light). Bars are averages ±standard error of the mean (SEM). "Pred. off" corresponds to LPL but without the predictive term in the learning rule (cf. Eq. 2.7). "Hebb off" refers to the configuration without the BCM-like Hebbian term. Finally, "Decorr. off" is the same as the single neuron learning rule (Eq. (2.1)) without the decorrelation term. LPL yields features with high linear readout accuracy. In contrast, when any component of LPL is disabled, linear readout accuracy drops below the pixel-decoding accuracy of ~32% (dashed line). **(c)** Linear readout accuracy of the internal representations at different layers of the DNN after layer-local training. Data points are averages ($n = 4$), error bands indicate SEM. LPL's representations improve up to six layers and then settle at a high level. In contrast, readout accuracy is close to chance level without the Hebbian component, and similarly remains at low levels when the decorrelating mechanism is switched off. Interestingly, when the predictive term is off, the readout accuracy initially increases in early layers, but then ultimately decreases back below the pixel-level accuracy with further increasing depth. Finally, the full LPL learning rule applied to inputs in which temporal contingency is destroyed (LPL shuffled) behaves qualitatively similar to the purely Hebbian rule. **(d)** Dimensionality ±SEM of the internal representations for the different learning rule configurations shown in (b). When either the Hebbian or decorrelation term are disabled, the dimensionality of the representations collapses to one. **(e)** Mean neuronal activity at different layers of the DNN after training with the different learning rule variants shown in (c). Data averaged over networks as in (c). Error bands denote ±SEM. Excluding the Hebbian term (dotted line) leads to collapsed representations in all layers.

experiments. First, we repeated the same simulation but now excluding the predictive term. This modification resulted in an accuracy of $(27.0 \pm 0.2)\%$, which is *lower* than the linear readout accuracy of a classifier trained directly on the pixels of the input images (see Table 2.1), indicating that the network did not learn disentangled representations, consistent with previous studies on purely Hebbian plasticity [20, 119]. We measured a similar drop in accuracy when we disabled either the Hebbian or the decorrelation component during learning (Fig. 2.3b).

Convolutional DNNs trained through supervised learning use depth to progressively separate representations [18]. To understand whether networks trained with LPL similarly leverage depth, we measured the linear readout accuracy of the internal representations at every layer in the network. Crucially, we found that in the LPL-trained networks, the readout accuracy increased with the number of layers until it gradually saturated (Fig. 2.3c), whereas this was not the case when any of the components of LPL was disabled. Similarly, readout accuracy decreased, when the temporal contiguity in the input was broken by shuffling, reminiscent of experiments in developing rats [101]. Together, these results suggest that LPL's combination of Hebbian, predictive, and decorrelating elements is crucial for disentangling representations in hierarchical DNNs.

In SSL, the two most common causes for failure to disentangle representations are representational and dimensional collapse (Supplementary Fig. S1), due to excessively high neuronal correlations [127]. To disambiguate between these two possibilities in our model, we computed the dimensionality of the representations and the mean neuronal activity at every layer (Methods). We found that disabling either the Hebbian or the decorrelation component led to a dimensionality of approximately one, whereas the LPL rule with and without the predictive term resulted in higher dimensionality: ≈ 15 or ≈ 50 respectively (Fig. 2.3d). Disabling the Hebbian term silenced all layers (Fig. 2.3e), demonstrating representational collapse. In contrast, disabling the decorrelation term resulted in non-zero activity levels, indicating that dimensional collapse underlies its poor readout accuracy (Fig. 2.3e). Finally, we verified that excluding LPL's predictive component neither caused representational nor dimensional collapse, suggesting that the decreasing linear readout accuracy with depth was due to the network's inability to learn good internal representations. Taken together, these results show that the predictive term is crucial for disentangling object representations in DNNs (Fig. 2.3) whereas the other terms are essential to prevent different forms of collapse.

It is an ongoing debate whether neurobiology implements some form of credit assignment [20, 71, 107–109]. Above we showed that LPL, as a local learning rule, effectively disentangles representations without the need for credit assignment, provided mechanisms exist that ensure neuronal decorrelation [124]. Naturally, our next question was whether a non-local LPL formulation could improve learning. To that end, we considered the

	STL-10		CIFAR-10	
	Layer-local	End-to-end	Layer-local	End-to-end
DNN with LPL	63.2±0.3	72.5±0.1	59.4±0.4	70.4±0.2
Raw pixel values	31.6		35.9	

Table 2.1: Linear classification accuracy in percent on the STL-10 and CIFAR-10 datasets for LPL and for a linear decoder trained on the raw pixel values (Methods). Error values correspond to SEM over $n = 4$ simulations with different random seeds.

fully non-local case using backpropagation. Specifically, we repeated our simulations with end-to-end training on the LPL objective defined at the network's output (Methods). While we do not know how the brain would implement such a non-local LPL algorithm, it provides an upper performance estimate of what is possible. End-to-end learning reproduced all essential findings of layer-local learning while increasing overall performance (Fig.2.3b; Table 2.1). Thus LPL's performance improves in the non-local setting, further underscoring that biological networks could benefit from credit assignment circuit mechanisms.

The above simulations used pairs of augmented images. To check whether the key findings generalized to more realistic input paradigms and other measures of disentangling, we trained DNNs with LPL on procedurally generated videos from the 3D Shapes dataset [128]. The videos consisted of objects shown under a slowly changing view angle, scale, or hue and occasional discontinuous scene changes, but without additional image augmentation (Supplementary Fig. A.4a,b; Methods). We found that LPL-trained networks reliably disentangle object identity. In contrast, networks trained without predictive learning failed to do so (Supplementary Fig. A.4c). Finally, the ground-truth latent manifold structure in the procedurally generated dataset is known. This knowledge allowed us to probe disentangling of the latent manifold directly instead of using linear classification as a proxy. This analysis revealed that LPL-trained networks faithfully disentangled the underlying objects and factors. At the same time, they also learned the topology of the data-generating manifold from the temporal sequence structure (Supplementary Figs. A.4d–g and A.5). Thus LPL's ability to disentangle representations generalizes to video stimuli and other measures of disentanglement.

2.2.3 Invariance learning in the primate inferotemporal cortex

Changing the temporal contiguity structure of visual stimuli induces neuronal selectivity changes in primate IT, an unsupervised learning effect described by Li and DiCarlo [100]. In their experiment, a macaque freely viewed a blank screen, with objects appearing in the peripheral visual field at one of two alternative locations relative to the (tracked) center of its gaze, prompting the macaque to perform a saccade to this location (Fig. 2.4a). The experimenters differentiated between normal exposures in which the object does not change during the saccade and "swap exposures" in which the initially presented object was

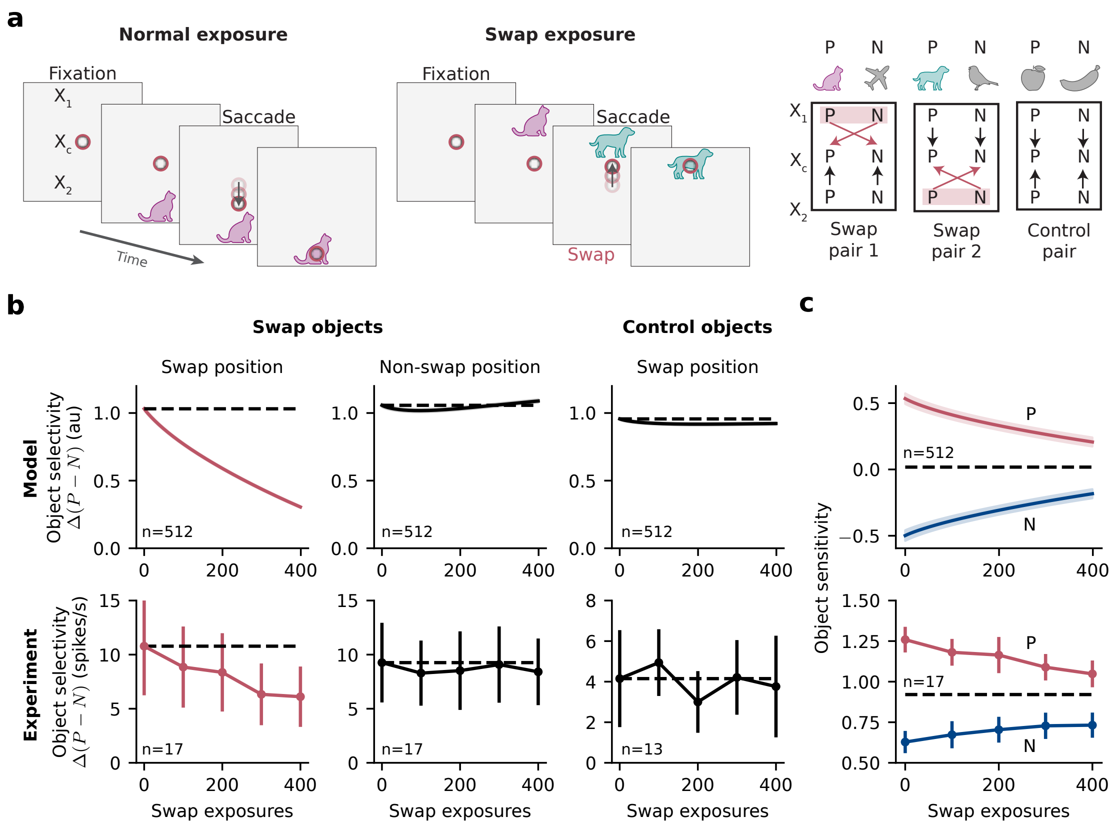


FIGURE 2.4: **LPL captures invariance learning in the primate inferotemporal cortex (IT). (a)** Schematic of the simulation setup modeled after the experiment by Li and DiCarlo [100]. The inputs to the model consist of images of objects presented at three different positions X_1, X_c, and X_2 on a blank canvas. Following the original experiment, we performed a targeted perturbation in the simulated visual experience that the model network was exposed to (left and center). Specifically, we switched object identities during transitions from a specific peripheral position, say X_1, to the central position X_c, while keeping transitions from the other peripheral position to the center unmodified (right). **(b)** Evolution of object selectivity as a function of number of swap exposures in the model (top row) and observed in-vivo (bottom row; data points extracted and replotted from [100], see Methods for details). Data are presented as mean values ±SEM. We differentiate between pairs of swapped objects at the Swap (left) and Non-swap positions (center) as well as control objects at the Swap position (right). LPL qualitatively reproduces the evolution of swap position-specific remapping of object selectivity as observed in IT. Control objects at the Swap position, i.e., images not used during the swap training protocol, show no selectivity changes in agreement with the experiment. **(c)** Average response to objects P and N as a function of number of swap exposures. The change in object selectivity between preferred objects P and non-preferred objects N is due to both increased responses to N and decreased responses to P in both our model (top) and the experimental recordings (bottom). Data are mean values ±SEM.

consistently swapped out for a different one as the monkey saccaded to a specific target location $\mathrm{X_{swap}}$. Hence, swap exposures created an “incorrect” temporal association between one object at position $\mathrm{X_{swap}}$ and a different one at the animal’s center of gaze $\mathrm{X_c}$. For any particular pair of swap objects, either the location above or below the center of gaze was chosen as $\mathrm{X_{swap}}$, and transitions from the opposite peripheral position $\mathrm{X_{nonswap}}$ to the center $\mathrm{X_c}$ were kept consistent as a control. The authors found systematic position- and object-specific changes of neuronal selectivity due to swap exposures which they attributed to unsupervised learning. Specifically, a neuron initially selective to an object P over another object N, reduced or even reversed its selectivity at the swap position $\mathrm{X_{swap}}$ while preserving its selectivity at the non-swap position $\mathrm{X_{nonswap}}$ (Fig. 2.4b).

We wanted to know whether LPL can account for these observations. To that end, we built a DNN model and generated input images by placing visual stimuli on a larger gray canvas to mimic central and peripheral vision as needed for the experiment (cf. Fig. 2.4a; Methods). Importantly, we ensured that the network’s input dimension and output feature map size were large enough to avoid full translation invariance due to the network’s convolutional structure alone. To simulate the animal’s prior visual experience, we trained our network model with LPL on a natural image dataset. After training, the learned representations were invariant to object location on the canvas (Supplementary Fig. S2), a known property of neural representations in the primate IT [104]. Next, we simulated targeted input perturbations analogous to the original experiment. For a given pair of images from different classes, we switched object identities during transitions from a specific peripheral position, say $\mathrm{X_1}$, to the center X_c while keeping transitions from the other peripheral position $\mathrm{X_2}$ to the center unmodified. We used $\mathrm{X_1}$ as the swap position for half of the image pairs and $\mathrm{X_2}$ for the other half. Throughout, we recorded neuronal responses in the network’s output layer while the weights in the network model evolved according to the LPL rule.

We observed that the neuronal selectivity between preferred inputs P, as defined by their initial preference (Methods), in comparison to non-preferred stimuli N in the model qualitatively reproduced the results of the experiment (Fig. 2.4b). Effectively, LPL trained the network’s output neurons to reduce their selectivity to their preferred inputs P at the swap position while preserving their selectivity at the non-swap position. Furthermore, we observed that object selectivity between pairs of control objects did not change, consistent with the experiment (Fig. 2.4b). Further analysis revealed that the origin of the selectivity changes between P and N stimuli at the swap position was due to both increases in responses to N and decreases in responses to P, an effect also observed in the experiments (Fig. 2.4c). Thus, LPL can account for neuronal selectivity changes observed in monkey IT during in-vivo unsupervised visual learning experiments.

2.2.4 Learning in spiking neural networks

So far we considered LPL in discrete-time rate-based neuron models without an explicit separation of excitatory and inhibitory neurons. In contrast, cortical circuits consist of spiking neurons that obey Dale's law, and learn in continuous time. To test whether our theory would extend to such a more realistic setting, we simulated a plastic recurrent SNN model consisting of 100 excitatory and 25 inhibitory neurons (Fig. 2.5a; Methods). We simulated input from five Poisson populations with temporally varying firing rates (Fig. 2.5b; Methods). Input population $P0$ had a constant firing rate, whereas $P1$'s and $P2$'s firing rates followed two independent slowly varying signals. $P1_{\mathrm{ctl}}$ and $P2_{\mathrm{ctl}}$ whose firing rates were temporally shuffled versions of $P1$ and $P2$ served as control populations. The input connections to the excitatory neurons evolved according to the spiking LPL rule (cf. (2.1)), a fully local learning rule. Decorrelation was achieved through inhibitory STDP ([124]; Methods).

After approximately 28 h of simulated time, the network's firing dynamics had settled into an asynchronous irregular activity regime from which the slowly varying input signals could be decoded linearly with high fidelity (Fig. 2.5b). In contrast, $P1_{\mathrm{ctl}}$ and $P2_{\mathrm{ctl}}$ did not have high reconstruction accuracy, consistent with the idea that the network preferentially represents the slowly varying inputs in its activity. This notion was supported by the strong synaptic connectivity to $P1/2$ (Fig. 2.5c). We further computed the relative difference between the average afferent weight from each signal in comparison to its associated control pathway. As expected, we found that neuronal weights were preferentially tuned to the slow input channels (Fig. 2.5d). However, this selectivity was lost when we turned either the predictive or the Hebbian term off. The absence of Hebbian plasticity was further accompanied by activity collapse (Fig. 2.5e), like in the rate-based network.

To investigate the role of inhibition, we next removed the inhibitory population. This manipulation resulted in excessively high firing rates (Fig. 2.5e; Supplementary Fig. A.6) and a notable reduction of the representational dimensionality (Fig. 2.5f; Methods). In the network with plastic inhibition, weights were more decorrelated and purely selective to either $P1$ or $P2$ (Fig. 2.5g). In contrast, removing inhibition resulted in fewer neurons preferentially tuned to either signal (Fig. 2.5h). Finally, a network with fixed inhibitory weights showed comparable dimensionality to the plastic inhibition case (Fig. 2.5f), but with a drop in selectivity (Fig. 2.5d). These results indicate that inhibition is needed to prevent correlated neuronal activity and the ensuing reduction in representational dimensionality. Further, inhibitory plasticity is required to ensure that the slow signals are preferentially represented (Supplementary Fig. A.6). Together, these findings illustrate that LPL learns predictive features in realistic spiking circuits with separate excitatory and inhibitory neuronal populations.

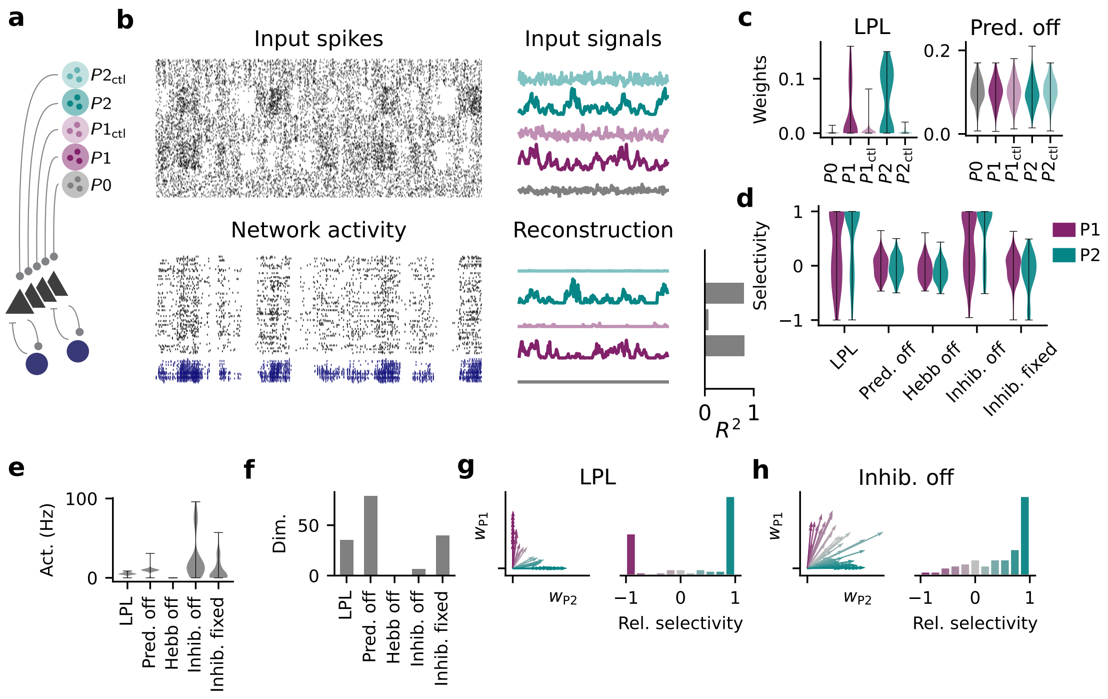


Figure 2.5: **LPL in a spiking neural network (SNN).** **(a)** Wiring diagram of the SNN with five distinct input populations. **(b)** Snapshot of spiking activity over 100 ms after LPL plasticity for the inputs (top left) and the network (bottom left) separated into excitatory (black) and inhibitory neurons (blue). The input spikes are organized in five distinct Poisson populations whose firing rates evolve according to five different temporal input signals (top right). Population activity of two slowly varying signals ($P_{1/2}$) can be linearly reconstructed (Methods) with high R^2 values from the network activity whereas temporally shuffled control signals ("ctl"; Methods) are heavily suppressed (bottom right). **(c)** Distribution of mean afferent synaptic strength per excitatory neuron ($n = 100$) grouped by input population. Input connections from slowly varying signals are larger than those from the shuffle controls (left), but not when learning with the predictive term turned off (right). Error bars show min/max ranges. **(d)** Signal selectivity as relative difference between signal and control pathway for networks trained with different learning rule variations (Methods; $n = 100$ neurons). "LPL" refers to learning with the spiking LPL rule combined with inhibitory plasticity on the inhibitory-to-excitatory connections. "Pred. off" corresponds to learning without the predictive term, and "Hebb off" to learning without the Hebbian term. "Inhib. off" refers to a setting without any inhibitory neurons, whereas "Inhib. fixed" indicates a setting where the inhibitory-to-excitatory weights are held fixed. The network with LPL and inhibitory plasticity acquires high selectivity to both signals. Selectivity is lost if the predictive term, the Hebbian term, or inhibitory plasticity are switched off. **(e)** Average firing rate of excitatory neurons ($n = 100$) in the network for the different configurations in (d). When the Hebbian term is off, spiking activity collapses to low activity levels in contrast to all other configurations in which it settles at intermediate activity levels. **(f)** Dimensionality of the neuronal representations (Methods) for the different configurations in (d). Inhibition prevents dimensional collapse, even in cases where inhibition is not plastic. **(g)** Averaged weight vectors of all excitatory neurons corresponding to input populations $P1$ and $P2$ (left) and the distribution of relative neuronal selectivities between these populations (right). Most neurons become selective either to $P1$ or $P2$, but few to both signals simultaneously. Color indicates relative preference of their weight vectors to either signal (Methods). **(h)** Same as (g), but without an inhibitory population. Most neurons develop selectivity to $P2$ or mixed selectivity to both signals, and their weight vectors are more correlated.

2.2.5 Rate and spike-timing dependence of synaptic plasticity

Next, we wanted to examine whether the spike-based LPL rule is consistent with experimental observations of plasticity induction. Experiments commonly report intertwined rate and spike-timing dependence presumably mediated through nonlinear voltage- and calcium-dependent cellular mechanisms [117, 129]. Theoretical work has further established conceptual links between phenomenological STDP models, SFA, and BCM theory [113, 130–134].

To compare LPL to experiments, we simulated a standard STDP induction protocol. Specifically, we paired 100 pre- and post-synaptic action potentials with varying relative timing Δt for a range of different repetition frequencies ρ. During the entire plasticity induction protocol, the postsynaptic cell was kept depolarized close to its firing threshold and weights evolved according to spike-based LPL. We repeated the simulated induction protocol for different initial values of the slowly moving averages of the postsynaptic firing rate $\bar{S}_i(t)$ and variance $\sigma_i^2(t)$ (Methods). This was done because these variables do not change much over the course of a single induction protocol due to their slow dynamics. Their presence, however, makes LPL a form of metaplasticity, i.e., plasticity depends on past neuronal activity.

We found that for small initial values of σ_i^2, the induced weight changes followed an antisymmetric temporal profile consistent with STDP experiments (Fig. 2.6a). For larger initial values of σ_i^2, the STDP window changed to a more symmetric and then ultimately an anti-Hebbian profile while the plasticity amplitude was suppressed, as expected due to the variance-dependent suppression of the Hebbian term in the learning rule (Fig. 2.6b,c). Next we investigated the effect of different initial values for $\bar{S}_i(t)$, which acts as a moving threshold reminiscent of BCM. Specifically, we recorded plastic changes at two fixed spike timing intervals $\Delta t = \pm 10\,\mathrm{ms}$ for $\sigma_i^2(t=0) = 10^{-5}$. For intermediate threshold values $\bar{S}_i(t=0) = 20\,\mathrm{Hz}$, causal spike timing induced long-term potentiation (LTP) with a nonlinear frequency dependence (Fig. 2.6d) whereas acausal pre-after-post timings showed a characteristic cross-over from LTD to LTP similarly observed in experiments [116]. In contrast, a low initial threshold $\bar{S}_i(t=0) = 0$, which would occur in circuits that have been quiescent for extended periods of time, resulted in LTP induction for both positive and negative spike timings whereas a high initial value ($\bar{S}_i(t=0) \geq 50\,\mathrm{Hz}$), corresponding to circuits with excessively high activity levels, led to LTD (Supplementary Fig. A.7). Importantly such slow shifts in activity-dependent plasticity behavior are consistent with the metaplasticity observed in monocular deprivation experiments [110, 121, 134]. Thus, LPL qualitatively captures key phenomena observed in experiments such as STDP, the rate-dependence of plasticity, and metaplasticity, despite not being optimized to reproduce these phenomena. Rather our model offers a simple normative explanation for the necessity of different plasticity patterns that are also observed experimentally [129].

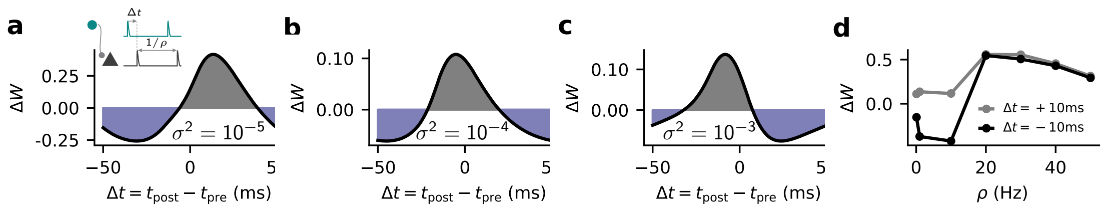


FIGURE 2.6: **LPL accounts for STDP and predicts metaplasticity of the STDP window.** **(a)** Relative weight change due to LPL in response to a standard STDP induction protocol with varying spike timing Δt for 100 pairings at a repetition frequency of $\rho = 10\,\text{Hz}$ (inset) for an initial value of $\sigma^2(t=0) = 10^{-5}$. **(b)** Same as (a), but with initial value of $\sigma^2(0) = 10^{-4}$. **(c)** Same as (a), but with $\sigma^2(0) = 10^{-3}$. **(d)** Relative weight change as a function of repetition frequency ρ for positive and negative relative spike timings ($\Delta t = \pm 10\,\text{ms}$).

2.3 Discussion

We have introduced LPL, a local plasticity rule that combines Hebbian and predictive elements. We demonstrated that LPL disentangles object representations in DNNs through mere exposure to temporal data in which object identity varies slowly. Crucially, we showed that predictive and Hebbian learning are both required to achieve this effect. Moreover, we demonstrated that LPL qualitatively captures the representational changes observed in unsupervised learning experiments in monkey IT [100]. Finally, we found that LPL in SNNs naturally reproduces STDP and its experimentally observed rate-dependence, while further predicting a new form of metaplasticity with distinct variance-dependence of the STDP window.

The idea that sensory networks use temporal prediction as a learning objective has been studied extensively in both machine learning and neuroscience. The model in this article combines elements of classic BCM theory with central ideas of SFA and more recent SSL approaches from machine learning. While SSL has shown great promise in representation learning without labelled data, it is typically formulated as a contrastive learning problem requiring negative samples [22, 23] to prevent representational collapse. Since negative samples break temporal contiguity they are not not biologically plausible. LPL does not require negative samples. Instead, it relies on variance regularization as proposed previously to prevent collapse [28]. Our model uses virtually the same mechanism, albeit with a logarithmic variance dependence (Supplementary Note A.4), and builds a conceptual bridge from variance regularization to Hebbian metaplasticity. Like most SSL approaches, Bardes *et al.* [28] used end-to-end learning whereby the objective function is formulated on the embeddings at the network's output. In contrast, we studied the case of greedy learning in which the objective is applied to each layer individually. Doing so alleviates the need for backpropagation and permitted us to formulate the weight updates as local learning rules, similar to work that combined contrastive objectives with greedy training [118]. Furthermore, recent work showed that greedy contrastive learning is directly linked to plasticity rules that rapidly switch between Hebbian and anti-Hebbian learning through a global third factor [114]. However, both these models required implausible negative samples, whereas LPL does neither require end-to-end training nor negative samples.

LPL shares its basic shape with the BCM rule, which has been qualitatively confirmed in numerous experimental studies both in-vitro [110, 116, 121] and in-vivo [122]. Furthermore, BCM has been linked to STDP [117] and informed numerous phenomenological plasticity models [130–133, 135]. However, unequivocal evidence for the predicted supralinear behavior of the firing rate-dependence of the BCM sliding threshold remains scarce [121] and the fast sliding threshold required for network stability seems at odds with experiments [40, 134]. In contrast, LPL does not require a rapid nonlinear sliding threshold for stability. Instead, it posits a fast-acting variance-dependence of Hebbian plasticity that ensures stability. This suppressive effect allows the sliding threshold, possibly implemented through neuronal or circuit mechanisms [121, 136], to catch up slowly, more consistent with experiments [134]. Hence, LPL offers a possible explanation for the current gap between theory and experiment.

The notion of slowness learning has been studied extensively in the context of the Trace Rule [137], optimal stability [138], and SFA [16, 126] which have conceptual ties to STDP [113]. However, the former enforces a hard constraint on the norm of the weight vector to prevent collapse, while the latter rely on hard variance constraints on the activity. In contrast, LPL implements a soft variance constraint [28] to the same effect. A similar soft constraint on the variance can be derived from statistical independence arguments [139] within a mutual information view of SSL [22]. However, these studies used negative samples, assume rapid global sign switching of the learning rule, and did not connect their work to biological plasticity mechanisms.

Our study has several limitations which we aim to address in future work. First, our study is limited to visual tasks of core object recognition, whereas other sensory modalities may use LPL as a mechanism to form disentangled representations of the external world. For computational feasibility, we restricted ourselves to artificial data augmentation techniques borrowed from SSL and procedurally generated videos with a simple structure, which are only crude proxies of rich real-world stimuli. Finally, there remains a performance gap in classification performance compared to less plausible fully supervised and contrastive approaches (Supplementary Table A.1) showing that there remains room for improvement, possibly by incorporating biological circuit mechanisms and top-down feedback connections into the model. It is left as future work to show how LPL can be extended to the circuit level and to more ethologically realistic sensory modalities [140] and video input while further combining them with plausible models of saccadic eye movement.

Despite the limitations, our model makes several concrete predictions. First, modulating the strength of Hebbian plasticity as a function of the postsynaptic variance is essential to LPL. Therefore, the predictive contribution to plasticity should be best observable for highly variable neuronal activity. While our model does not make quantitative predictions about the time scale of variance estimation, we expect that a quiescent neuron shows stronger Hebbian plasticity than neurons with highly irregular activity. Moreover, LPL should manifest in metaplasticity experiments as a transition from an asymmetric Hebbian STDP window, via a symmetric window to, ultimately, an anti-Hebbian window (cf. Fig. 2.6) when priming the postsynaptic neuron with increasing output variance. Specifically, we expect a neuron which has remained quiescent for a long period of time to display a classic STDP window, whereas a neuron whose activity has undergone substantial fluctuations in the recent past should show an inverted STDP window. Such metaplasticity may account for the diversity of different shapes of STDP windows observed in experiments [129].

To fathom how established data-driven plasticity models are related to theoretically motivated learning paradigms such as SFA and SSL is essential to understanding the brain. A central open question in neuroscience remains: How do the different components of such learning rules interact with the rich local microcircuitry to yield useful representations at the network level? In this article, we have only scratched the surface by proposing a local plasticity rule and illustrating its aptitude for disentangling internal representations. However, a performance gap remains compared to learning algorithms that can leverage top-down feedback. We expect that extending predictive learning to the circuit and network level will narrow this gap and generate deep mechanistic insights into the underlying principles of neural plasticity.

2.4 Methods

2.4.1 Plasticity model

The LPL rule is derived from an objective function approach. It consists of three distinct parts, each stemming from a different additive term in the following combined objective function:

$$\mathcal{L}_{\text{LPL}} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{Hebb}} + \mathcal{L}_{\text{decorr}} \tag{2.3}$$

First, the predictive component $\mathcal{L}_{\text{pred}}$ minimizes neuronal output fluctuations for inputs that occur close in time. Second, a Hebbian component $\mathcal{L}_{\text{Hebb}}$ maximizes variance and thereby prevents representational collapse. Finally, $\mathcal{L}_{\text{decorr}}$ is a decorrelation term that we use in all non-spiking network simulations to prevent excessive correlations between neurons within the same layer in a network. In SNNs decorrelation is achieved without this term through lateral inhibition and inhibitory plasticity.

In the following, we consider a network layer with N input units and M output units trained on batches of B pairs of consecutive stimuli. In all simulations we approximate the temporal derivative ${}^{\mathrm{d}z}/_{\mathrm{d}t}$ which appears in Eqn. (2.1) by finite differences $z(t) - z(t - \Delta t)$ assuming a discrete timestep Δt while absorbing all constants into the learning rate. In this formulation, the LPL rule has a time horizon of two time steps in the sense that only one temporal transition enters into the learning rule directly. We used this insight to efficiently train our models using minibatches of paired consecutive input stimuli which approximates learning on extended temporal sequences consisting of many time steps. Let $x^b(t) \in \mathbb{R}^N$ be the input to the network at time t, $W \in \mathbb{R}^{M\times N}$ be the weight matrix to be learned, $a^b(t) = Wx^b(t) \in \mathbb{R}^M$ be the pre-activations, and $z_i^b(t) = f(a_i^b(t))$ the activity of the ith output neuron at time t. Finally, b indexes the training example within a minibatch of size B.

Predictive component. We define the predictive objective $\mathcal{L}_{\text{pred}}$ as the mean squared difference between neuronal activity in consecutive time steps.

$$\begin{aligned}\mathcal{L}_{\text{pred}}(t) &= \frac{1}{2MB}\sum_{b=1}^{B}\|z^b(t) - \mathrm{SG}(z^b(t-\Delta t))\|^2 \\ &= \frac{1}{2MB}\sum_{b=1}^{B}\sum_{i=1}^{M}\left(z_i^b(t) - \mathrm{SG}(z_i^b(t-\Delta t))\right)^2 \qquad (2.4)\end{aligned}$$

Here SG denotes the Stopgrad function, which signifies that the gradient is not evaluated with respect to quantities in the past.

Hebbian component. To avoid representational collapse we rely on the Hebbian plasticity rule that results from minimizing the negative logarithm of the variance of neuronal activity:

$$\mathcal{L}_{\text{Hebb}}(t) = \frac{1}{M}\sum_{i=1}^{M} -\log\left(\sigma_i^2(t)\right) \qquad (2.5)$$

where $\bar{z}_i(t) = \mathrm{SG}(\frac{1}{B}\sum_{b=1}^{B} z_i^b(t))$ and $\sigma_i^2(t) = \frac{1}{B-1}\sum_{b=1}^{B}\left(z_i^b(t) - \bar{z}_i(t)\right)^2$ are the current estimates of the mean and variance of the activity of the ith output neuron. Note that we do not compute gradients with respect to the mean estimate, which would require backpropagation through time. Assuming that the mean is fixed allows formulating LPL as a temporally local learning rule (cf. Eq. (2.3)). To minimize the computational burden in DNN simulations, we performed all necessary computations on minibatches, which includes estimating the mean and variance. However, these quantities could also be estimated using stale estimates from previous inputs, a requirement for implementing LPL as an online learning rule. Using stale mean and variance estimates from previous minibatches in our DNN simulations did cause a drop in readout performance (Supplementary Table A.2).

Still, such a drop could possibly be avoided either using larger mini batch sizes, by further reducing the learning rate, or by computing the estimates as running averages over past inputs. All of the above manipulations result in essentially the same learning rule (see Supplementary Note A.2).

Decorrelation component. Finally, we use a decorrelation objective to prevent excessive correlation between different neurons in the same layer as suggested previously [28, 90, 123]. The decorrelation loss function is the sum of the squared off-diagonal terms of the covariance matrix between units within the same layer, which is given as

$$\mathcal{L}_{\text{decorr}}(t) = \frac{1}{(M^2 - M)} \sum_{i=1}^{M} \sum_{k \neq i} \left(\frac{\sum_{b=1}^{B} \left(z_i^b(t) - \bar{z}_i(t)\right) \left(z_k^b(t) - \bar{z}_k(t)\right)}{B - 1} \right)^2 , \tag{2.6}$$

with a scaling factor that keeps the objective invariant to the number of units in the population.

The full learning rule. We obtain the LPL rule as the negative gradient of the total objective $\mathcal{L}_{\text{LPL}}$ plus an added weight decay. For a single network layer, this yields the layer-local LPL rule where we omitted the time argument t from all present quantities for brevity:

$$\begin{aligned} \Delta W_{ij} &= -\eta \left(\frac{\partial \mathcal{L}_{\text{pred}}}{\partial W_{ij}} + \lambda_1 \frac{\partial \mathcal{L}_{\text{Hebb}}}{\partial W_{ij}} + \lambda_2 \frac{\partial \mathcal{L}_{\text{decorr}}}{\partial W_{ij}} \right) - \eta \eta_w W_{ij} \\ &= \eta \frac{1}{MB} \sum_{b=1}^{B} \left(-(z_i^b - z_i^b(t - \Delta t)) + \lambda_1 \frac{\alpha}{\sigma_i^2} (z_i^b - \bar{z}_i) - \lambda_2 \beta \sum_{k \neq i} (z_k^b - \bar{z}_k) C_{ik} \right) f'(a_i^b) x_j^b \\ &\quad - \eta \eta_w W_{ij} \end{aligned} \tag{2.7}$$

Here λ_1 and λ_2 are parameters which control the relative strengths of each objective, α and β are the appropriate normalizing constants for batch size and number of units, η_w is a parameter controlling the strength of the weight decay, and C_{ik} is the covariance between units i and k:

$$C_{ik} = \frac{1}{B - 1} \sum_{b'=1}^{B} \left(z_i^{b'}(t) - \bar{z}_i(t) \right) \left(z_k^{b'}(t) - \bar{z}_k(t) \right) . \tag{2.8}$$

Numerical optimization methods. We implemented all network models learning with LPL using gradient descent on the equivalent objective function in PyTorch (v1.11.0) with the Lightning framework (v1.6.1). DNN simulations were run on five Linux workstations equipped with Nvidia Quadro RTX 5000 graphics processing units (GPUs) and a compute cluster with Nvidia V100 and A100 GPUs. In case of the DNNs, we used the Adam optimizer to accelerate learning. Parameter values used in all simulations are summarized in Supplementary Table A.3. All simulations were run using Python (v3.8). We used Jupyter notebooks (v1.0.0) for all data analysis and plotting. The simulation and analysis code is available online [141].

2.4.2 Learning in the single neuron setup

We considered a simple linear rate-based neuron model whose output firing rate z is given by the weighted sum of the firing rates x_j of the input neurons, i.e, $z = \sum_j W_j x_j$, where W_j corresponds to the synaptic weight of input j. We trained the neuron using stochastic gradient descent (SGD) on the corresponding objective function:

$$\mathcal{L} = \frac{1}{B}\left(z(t) - SG(z(t - \Delta t))\right)^2 - \log\left(\sigma_z^2(t) + \epsilon\right) - \eta_w \sum_j W_j^2 \tag{2.9}$$

Here, and in all following simulations, we fixed the Hebbian coefficient $\lambda_1 = 1$. We also added a small constant $\epsilon = 10^{-6}$ to the estimate of the variance σ_z for numerical stability. In case of a single rate neuron, the LPL rule (Eq. (2.7)) simplifies to Eq. (2.1) without the decorrelation term.

Synthetic two-dimensional dataset generation. The two-dimensional synthetic data sequence (Fig. 2.2a) consists of two clusters of inputs, one centered at $x = -1$, and the other at $x = +1$. Pairs of consecutive data points were drawn independently from normal distributions centered at their corresponding cluster. To generate a family of different datasets, we kept the standard deviation in the x-direction fixed at $\sigma_x = 0.1$ and varied σ_y. Additionally, to account for occasional transitions between clusters with probability p, we included a corresponding fraction of such "crossover pairs" in the training batch. For each value of σ_y, we simulated the evolution of the input connections of a single linear model neuron that received the x and y as its two inputs, and updated its input weights according to LPL. In the simulations in Fig. 2.2 we assumed $p \to 0$, however, the qualitative behavior remained unchanged for noise levels below $p = 0.5$, i.e, as long as the "noisy" pairs of points from different clusters were rare in each training batch (Supplementary Fig. A.8).

Neuronal selectivity measure. After training weights to convergence, we measured the neuron's selectivity to the x-input as the normalized difference between mean responses to stimuli coming from the two respective input clusters. Concretely, let $\langle z_1 \rangle$ be the average output caused by inputs from the $x = 1$ cluster, and $\langle z_2 \rangle$ from the $x = -1$ cluster, then the selectivity χ is defined as:

$$\chi = \frac{|\langle z_1 \rangle - \langle z_2 \rangle|}{z_{\max} - z_{\min}} \tag{2.10}$$

with $z_{\max}$ the maximum and $z_{\min}$ the minimum response across all inputs.

2.4.3 Learning in deep convolutional neural networks

For all network simulations, we used a convolutional DNN based on the VGG-11 architecture [142] (see Supplementary Note A.6 for details). We trained this network on STL-10 and CIFAR-10 (Supplementary Fig. A.9), two natural image datasets (see Supplementary Table A.3 for hyperparameters). To simulate related consecutive inputs, we used two differently augmented versions of the same underlying image, a typical approach in vision-based SSL methods. Specifically, we first standardized the pixel values to zero mean and unit standard deviation within each dataset before using the set of augmentations originally suggested by [23], which includes random crops, blurring, color jitter and random horizontal flips (see Supplementary Fig. A.2 for examples).

Synthetic video generation. To study LPL in settings with more naturalistic transitions between consecutive images and without relying on image augmentation, we procedurally generated videos using images from the 3D Shapes dataset [128]. The dataset has a known latent manifold structure spanned by view angle, object scale, hue, and object type and is commonly used to measure disentangling in variational autoencoders. Using the knowledge of the ground truth factors, we generated continuous video composed of 17-frame clips during which the object shape remained fixed and a randomly chosen factor changed gradually. Specifically, we proceeded as follows: we randomly chose one factor and changed it frame-by-frame such that transitions between adjacent factor values were more likely. For instance, one such clip shows a cube under a smoothly varying camera angle (Supplementary Fig. A.4a). Furthermore, we randomly permuted the order of all three hue factors. This was done to break the orderly ring topology of the hue mappings in the original dataset, which allowed us to test that the structure is restored through LPL, but not other methods (see Supplementary Fig. A.4g). After 17 frames we randomly chose another shape and factor and repeated the above procedure. This sequence generation resulted in video with many consecutive latent manifold traversals as captured by the empirical transition matrices (Supplementary Fig. A.5a). Importantly, due to the nature of the video, which switches between objects periodically, the resulting input sequence also included occasional transitions between different objects that the LPL rule interprets as positive samples. Such transitions also appear in real world stimuli when objects leave or enter the scene. Despite these “false positives” LPL learned disentangled representations of shapes and the underlying factors.

Network training. We trained our network models on natural image data by minimizing the equivalent LPL objective function. For both datasets, we trained the DNN using the Adam optimizer with default parameters and a cosine learning rate schedule that drove the learning rate to zero after 800 epochs. We distinguish between two cases: layer-local and end-to-end learning. End-to-end learning corresponds to training the network by optimizing

$\mathcal{L}_{\mathrm{LPL}}^{(\mathrm{out})}$ at the network's output while using backpropagation to train the hidden layer weights. This is the standard approach used in deep learning. In contrast, in layer-local learning, we minimized the LPL objective $\mathcal{L}_{\mathrm{LPL}}$ at each layer in the network independently without backpropagating loss gradients between layers similar to previous work [114, 118]. In this case, every layer greedily learns predictive features of its own inputs, i.e, its previous layer's representations. To achieve this behavior, we prevented PyTorch from backpropagating gradients between layers by detaching the output of every layer in the forward pass and optimizing the sum of per-layer losses $\sum_l \mathcal{L}_{\mathrm{LPL}}^{(l)}$.

Unless mentioned otherwise, we used global average pooling (GAP) to reduce feature maps to a single vector before applying the learning objective at the output of every convolutional layer for layer-local training, or just at the final output in the case of end-to-end training. Although pooling was not strictly necessary and LPL could be directly applied on the feature maps (Supplementary Fig. A.10), it substantially sped up learning and led to an overall improved linear readout accuracy on CIFAR-10 (Supplementary Table A.2). However, we observed that GAP was essential on the STL-10 dataset for achieving readout accuracy levels above the pixel-level baseline (cf. Table 2.1). This discrepancy was presumably due to the larger pixel dimensions of this dataset and the resulting smaller relative receptive field size in early convolutional layers. Concretely, feature pixels in the first convolutional layer of VGG-11 have a receptive field of 3×3 pixels covering a larger portion of the 32×32 CIFAR-10 images as compared to the 96×96 STL-10 inputs. This hypothesis was corroborated by the fact that when we sub-sampled STL-10 images to a 32×32 resolution, the dependence on GAP was removed and LPL was effective directly on the feature maps (Supplementary Table A.2).

Baseline models. As baseline models for comparison (Supplementary Table A.1), we trained the same convolutional neural network (CNN) network architecture either with a standard cross-entropy supervised objective, which requires labels, or with a contrastive objective, which relies on negative samples. To implement contrastive learning, the network outputs $z(t)$ were passed through two additional dense projection layers $v(t) = f_{\mathrm{proj}}(z(t))$, which is considered crucial in contrastive learning to avoid dimensional collapse [127]. Finally, the following contrastive loss function was applied to these projected outputs

$$\mathcal{L}_{\mathrm{contrast}}(t) = \sum_{b=1}^{B} \left(-\mathrm{sim}(v^b(t), \mathrm{SG}(v^b(t-\Delta t))) + \sum_{b' \neq b}^{B} \mathrm{sim}(v^b(t), v^{b'}(t)) \right) \tag{2.11}$$

where $\mathrm{sim}(v_1, v_2) = \frac{v_1^{\mathrm{T}} v_2}{\|v_1\| \|v_2\|}$ is the cosine similarity between two representations v_1 and v_2. The second term in the loss function is a sum over all pairwise similarities between inputs in a given minibatch. These pairs correspond to different underlying base images and therefore constitute negative samples. During training the network is therefore optimized to reduce the representational similarity between them.

For training the layer-local versions of the supervised and contrastive models, we followed the same procedure as with LPL of optimizing the respective loss function at the output of every convolutional layer l of the DNN without backpropagation between the layers. Because projection networks are necessary for avoiding dimensional collapse in case of contrastive learning, we included two additional dense layers to obtain the projected representations $v^l(t) = f^l_{\text{proj}}(z^l(t))$ at every level of the DNN before calculating the layer-wise contrastive loss $\mathcal{L}^l_{\text{contrast}}$. This meant that gradients were backpropagated through each of these dense layers for training the corresponding convolutional layers of the DNN, but consecutive convolutional layers were still trained independent of each other.

Population activity analysis

We adopted two different metrics in order to analyze the representations learned by the DNN after unsupervised training with LPL on the natural image datasets.

Linear readout accuracy. To evaluate how well the LPL rule trained the DNN to disentangle and identify underlying latent factors in a given image, we measured linear decodability by training a linear classifier on the network outputs in response to a set of training images. Crucially, during this step we only trained the readout weights while keeping the weights of the LPL-pretrained DNN frozen. We then evaluated the linear readout accuracy (Fig. 2.3b) on a held-out test set of images. We used the same procedure to evaluate the representations at intermediate layers (Fig. 2.3c), and for the baseline models.

Representational similarity analysis. To visualize the latent manifold structure in learned network embeddings, we computed average representational similarity matrixs (RSMs). To obtain the RSM for one factor, say object hue, we first fixed the values of all the other factors and calculated the cosine similarity between the network outputs as the object hue was changed. We repeated this procedure for many different values for the other factors to get the final averaged RSM for object hue (Supplementary Fig. A.4f).

Metric for disentanglement. To quantitatively measure disentanglement, we used the metric proposed by [128]. This measure requires full knowledge of the underlying latent factors, as was the case for our procedurally generated videos. In brief, to compute the measure one first identifies the most insensitive neuron to all except one factor. Next, using the indices of these neurons, one trains a simple majority-vote classifier that predicts which factor is being coded for. The accuracy of this classifier on held-out data is the disentanglement score.

Dimensionality and activity measures. To characterize mean activity levels in the network models, we averaged neuronal responses over all inputs in the validation set. To quantify the dimensionality of the learned representations, we computed the participation ratio [143]. Concretely, if $Z \in \mathcal{R}^{B \times N}$ are N-dimensional representations of B input images, and λ_i, $1 \leq i \leq N$ is the set of eigenvalues of $Z^T Z$, then the participation ratio is given by:

$$\text{Dim.} = \frac{\left(\sum_{i=1}^{N} \lambda_i\right)^2}{\sum_{i=1}^{N} \lambda_i^2} \tag{2.12}$$

2.4.4 Model of unsupervised learning in inferotemporal cortex

Network model and pretraining dataset. To simulate the experimental setup of Li and DiCarlo [100], we modeled the animal's ventral visual pathway with a convolutional DNN. To that end, we used the same network architecture as before except that we removed all biases in the convolutional layers in order to prevent boundary effects. This modification resulted in a drop in linear readout accuracy (Supplementary Table A.2). Pre-training of the network model proceeded in two steps as follows. First, we performed unsupervised pre-training for 800 epochs on STL-10 using augmented image views exactly as before. Next, we added a fully-connected dense layer at the network's output, and trained it for 10 epochs with the LPL objective while keeping the weights of the convolutional layers frozen. During this second pre-training phase, we used augmented STL-10 inputs which were spatially extended in order to account for the added spatial dimension of different canvas positions in the experiment [100]. The expanded inputs consisted of images placed on a large black canvas at either the center position X_c or one of two peripheral positions $\mathrm{X}_{1/2}$ at the upper or lower end of the canvas. Concretely, these images had dimensions $(13 \times 96) \times 96$ which resulted in an expanded feature map at the output of the convolutional DNN with spatial dimensions 13×1 (see Supplementary Note A.6 for details). Note that we only expanded the canvas in the vertical dimension instead of using a setup with a 13×13 feature map because it resulted in a substantial reduction of computational and memory complexity. During this second stage of pre-training, the network was only exposed to "true" temporal transitions wherein the image was not altered between time steps apart from changing position on the canvas.

Data generation for simulated swap exposures. To simulate the experiment by [100], we exposed the network to normal and swap temporal transitions. In the latter case the image was consistently switched to one belonging to a different object category at the specific swap position. The swap position for a given pair of images was randomly pre-selected to be either X_1 or X_2, while the other non-swap position was used as a control. Specifically, we switched object identities during transitions from one peripheral swap position, say X_1, to the central position X_c, while keeping transitions from the other peripheral position X_2

to the center unmodified. As in the experiment, we chose several pairs of images as swap pairs, and fixed X_1 as the swap position for half the pairs of images and X_2 as the swap position for the other half. To simulate ongoing learning during exposure to these swap and non-swap input sequences, we continued fine-tuning the convolutional layers. To that end, we used the Adam optimizer we used during pre-training with its internal state restored to the state at the end of pre-training. Moreover, we used a learning rate of 10^{-7} during fine-tuning which was approximately $100\times$ larger than the learning rate reached by the cosine learning rate schedule during pre-training (4×10^{-9}, after 800 epochs). Finally, we trained the newly added dense layers with vanilla SGD with a learning rate of 0.02.

Neuronal selectivity analysis. Before training on the swap exposures, for each output neuron in the dense layer, we identified the preferred and non-preferred member of each swap image pair, based on which image drove higher activity in that neuron. This allowed us to quantify object selectivity on a per-neuron basis as $P - N$, where P is the neuron's response to its initially preferred image, and N to its nonpreferred image at the same position on the canvas. Note that, by definition, the initial object selectivity for every neuron is positive. Finally, we measured the changes in object selectivity $P - N$ during the swap training regimen, at the swap and non-swap positions averaging over all output neurons for all image pairs. As a control, we included measurements of the selectivity between pairs of control images that were not part of the swap set.

Comparison to experimental data. To compare our model to experiments, we extracted the data from [100] using the Engauge Digitizer software (v12.1) and replotted it in Fig. 2.4b.

2.4.5 Spiking neural network simulations

We tested a spiking version of LPL in networks of conductance-based leaky integrate-and-fire (LIF) neurons. Specifically, we simulated a recurrent network of 125 spiking neurons (100 excitatory and 25 inhibitory neurons) receiving afferent connections from 500 input neurons. In all simulations the input connections evolved according to the spike-based LPL rule described below. In our model, neurons actively decorrelated each other through locally connected inhibitory interneurons whose connectivity was shaped by inhibitory plasticity.

Neuron model. The neuron model was based on previous work [17, 35] in which the membrane potential U_i of neuron i evolves according to the ordinary differential equation

$$\tau^{\mathrm{mem}} \frac{dU_i}{dt} = \left(U^{\mathrm{leak}} - U_i\right) + g_i^{\mathrm{exc}}(t)\left(U^{\mathrm{exc}} - U_i\right) + g_i^{\mathrm{inh}}(t)\left(U^{\mathrm{inh}} - U_i\right) \tag{2.13}$$

where τ^{mem} denotes the membrane time constant, U^x are the synaptic reversal potentials (Supplementary Table A.4), and the $g_i^x(t)$ the corresponding synaptic conductances expressed in units of the neuronal leak conductance. The excitatory conductance is the sum of NMDA and AMPA conductances $g_i^{\text{exc}}(t) = 0.5(g_i^{\text{ampa}}(t) + g_i^{\text{nmda}}(t))$. Their dynamics are described by the following differential equations

$$\frac{dg_i^{\text{ampa}}}{dt}(t) = -\frac{g_i^{\text{exc}}(t)}{\tau^{\text{ampa}}} + \sum_{j \,\in\, \text{exc}} w_{ij} S_j(t) \tag{2.14}$$

$$\tau^{\text{nmda}} \frac{dg_i^{\text{nmda}}}{dt}(t) = g_i^{\text{ampa}}(t) - g_i^{\text{nmda}}(t) \tag{2.15}$$

whereas the inhibitory GABA conductance $g_i^{\text{inh}} = g_i^{\text{gaba}}$ evolves as

$$\tau^{\text{gaba}} \frac{dg_i^{\text{gaba}}}{dt} = -g_i^{\text{gaba}} + \sum_{j \,\in\, \text{inh}} w_{ij} S_j(t) \quad . \tag{2.16}$$

In the above expressions $S_j(t) = \sum_k \delta(t_j^k - t)$ refers to the afferent spike train emitted by neuron j, in which t_j^k are the corresponding firing times, and τ^x denotes the individual neuronal and synaptic time constants (Supplementary Table A.4). Neuron i fires an output spike whenever its membrane potential reaches the dynamic firing threshold $\vartheta_i(t)$ that evolves according to

$$\frac{d\vartheta_i}{dt}(t) = \frac{\vartheta^{\text{rest}} - \vartheta_i(t)}{\tau^{\text{thr}}} + \left(\vartheta^{\text{max}} - \vartheta_i(t)\right) S_i(t) \tag{2.17}$$

to implement an absolute and relative refractory period. Specifically, ϑ_i jumps to $\vartheta^{\text{max}} = 50\,\text{mV}$ every time an output spike is triggered after which it exponentially decays back to its rest value of $\vartheta^{\text{rest}} = -50\,\text{mV}$. All neuronal spikes are delayed by 0.8 ms to simulate axonal delay and to allow efficient parallel simulation before they trigger postsynaptic potential in other neurons.

Time varying spiking input model. Inputs were generated from 500 input neurons divided into five populations of 100 Poisson neurons each. All inputs where implemented as independent Poisson processes with the same average firing rate of 5 Hz and neurons within the same group shared the same instantaneous firing rate. Concretely, neurons in $P0$ had a fixed firing rate of 5 Hz, whereas the firing rates in groups $P1$ and $P2$ changed slowly over time. Specifically, we generated periodic template signals $x(t)$ from a Fourier basis

$$x(t) = \sum_k \frac{\theta_k}{\alpha^k} \sin\left(\frac{2\pi t + \phi_k}{T}\right) \tag{2.18}$$

with random uniformly drawn coefficients $0 \leq \theta_k, \phi_k < 1$. The spectral decay constant $\alpha = 1.1$ biased the signals toward slow frequencies and thus slowly varying temporal structure. We chose the period $T = 3\,\mathrm{s}$ for $P1$ and $(3+{}^1\!/_{13})\mathrm{s}$ for $P2$ respectively. The different periods were chosen to avoid phase-locking between the two signals. Both signals were then sampled at 10 ms intervals, centered on 5 Hz, variance-normalized, and clipped below at 0.1 Hz before using them as periodic time varying firing rates for $P1$ and $P2$. Additionally, we simulated control inputs $P1/2_{\mathrm{ctl}}$ of the two input signals by destroying their slowly varying temporal structure. To that end, we repeated the original firing rate profile for 13 periods before shuffling it on a time grid with 10 ms temporal resolution.

Spike-based LPL. To extend LPL to the spiking domain, we build on SuperSpike [36], a previously published online learning rule, which had only been used in the context of supervised learning in SNNs thus far. In this article, we replaced the supervised loss with the LPL loss (Eq. (2.3)) without the decorrelation term. The resulting spiking LPL online rule for the weight w_{ij} is given by

$$\begin{aligned}\frac{\mathrm{d}w_{ij}}{\mathrm{d}t} &= \eta\;\alpha * \left(\epsilon * S_j(t) f'(U_i(t))\right) \\ &\quad \times \left[\alpha * \left(-\left(S_i(t) - S_i(t-\Delta t)\right)\right) + \frac{\lambda}{\sigma_i^2 + \xi}\left(S_i(t) - \bar{S}_i(t)\right)\right)\Bigg] \\ &\quad + \eta \qquad \underbrace{\delta S_j(t)}_{\text{transmitter-triggered}} \end{aligned} \tag{2.19}$$

with the learning rate $\eta = 10^{-2}$, a small positive constant $\xi = 10^{-7}$ to avoid division by zero. Further, α is a double exponential causal filter kernel applied to the neuronal spike train $S_i(t)$. Similarly, ϵ is a causal filter kernel that captures the temporal shape of how a presynaptic spike influences the postsynaptic membrane potential. For simplicity, we assumed a fixed kernel and ignored any conductance-based effects and NMDA dependence. Further, we added the transmitter-triggered plasticity term with $\delta = 10^{-3}$ to ensure that weights of quiescent neurons slowly potentiate in the absence of activity to ultimately render them active [17]. Finally, $\lambda = 10^{-4}$ is a constant that modulates the strength of the Hebbian term. We set it to zero to switch off the predictive term where this is mentioned explicitly.

Further, $f'(U_i) = \beta\left(1 + \beta\left|U_i - \vartheta^{\mathrm{rest}}\right|\right)^{-2}$ is the surrogate derivative with $\beta = 1\,\mathrm{mV}^{-1}$, which renders the learning rule voltage-dependent. Finally, $\bar{S}_i(t)$ and $\sigma_i^2(t)$ are slowly varying quantities obtained online as exponential moving averages with the following dynamics

$$\tau^{\mathrm{mean}}\frac{\mathrm{d}\bar{S}_i(t)}{\mathrm{d}t} = S_i(t) - \bar{S}_i(t) \tag{2.20}$$

$$\tau^{\mathrm{var}}\frac{\mathrm{d}}{\mathrm{d}t}\sigma_i^2(t) = -\sigma_i^2(t) + \left(S_i(t) - \bar{S}_i(t)\right)^2 \tag{2.21}$$

with $\tau^{\text{mean}} = 600\,\text{s}$ and $\tau^{\text{var}} = 20\,\text{s}$. These quantities confer the spiking LPL rule with elements of metaplasticity [121].

In our simulations, we computed the convolutions with α and ϵ by double exponential filtering of all quantities. Generally, for the time varying quantity $c(t)$ we computed

$$\begin{aligned} \frac{\mathrm{d}\bar{c}}{dt}(t) &= -\frac{\bar{c}(t)}{\tau^{\text{rise}}} + c(t) && (2.22) \\ \tau^{\text{fall}} \frac{\mathrm{d}\bar{\bar{c}}}{\mathrm{d}t}(t) &= -\bar{\bar{c}}(t) + \bar{c}(t) && (2.23) \end{aligned}$$

which yields the convolved quantity $\bar{\bar{c}}$. Specifically, we used $\tau_\alpha^{\text{rise}} = 2\,\text{ms}$, $\tau_\alpha^{\text{fall}} = 10\,\text{ms}$, $\tau_\epsilon^{\text{rise}} = \tau_{\text{ampa}} = 5\,\text{ms}$, and $\tau_\epsilon^{\text{fall}} = \tau_{\text{mem}} = 20\,\text{ms}$.

Overall, one can appreciate the resemblance of Eq. (2.19) with the non-spiking equivalent (cf. Eq. (2.1)). As in the non-spiking case the learning rule is local in that it only depends on pre- and postsynaptic quantities. The predictive term in the learning rule can be seen as an instantaneous error signal which is minimized when the present output spike train $S_i(t)$ is identical to a delayed version of the same spike train $S_i(t - \Delta t)$ with $\Delta t = 20\,\text{ms}$. In other words, the past output serves as a target spike train (cf. [36]).

Microcircuit connectivity. Connections from the input population to the network neurons and recurrent connections were initialized with unstructured random sparse connectivity with different initial weight values (Supplementary Table A.5). One exception to this rule was the excitatory-to-inhibitory connectivity which was set up with a Gaussian connection probability profile

$$P_{ij}^{\text{con}} = \exp\left(-\frac{(j - c(i))^2}{\sigma^2}\right) \tag{2.24}$$

with $c(i) = 0.25i$ with $\sigma^2 = 20$ to mimic the dense local connectivity onto inhibitory neurons due to which inhibitory neurons inherit some of the tuning of their surrounding excitatory cells.

Inhibitory plasticity. Inhibitory to excitatory synapses were plastic unless mentioned otherwise. We modeled inhibitory plasticity according to a previously published inhibitory STDP model [124].

$$\frac{dw_{ij}^{\text{inh}}}{dt} = \zeta\left((x_i(t) - 2\kappa\tau^{\text{stdp}})S_j(t) + (x_j(t)S_i(t))\right) \tag{2.25}$$

using pre- and postsynaptic traces

$$\frac{dx_k}{dt} = -\frac{x_j(t)}{\tau^{\text{STDP}}} + S_k(t) \tag{2.26}$$

with time constant $\tau^{\mathrm{STDP}} = 20\,\mathrm{ms}$, learning rate $\zeta = 1 \times 10^{-3}$, and target firing rate $\kappa = 10\,\mathrm{Hz}$.

Reconstruction of input signals from network activity. To reconstruct the input signals, we first computed input firing rates of the five input populations by binning their spikes emitted during the last 100 s of the simulation in 25 ms bins. We further averaged the binned spikes over input neurons to provide the regression targets. Similarly, we computed the binned firing rates of the network neurons but without averaging over neurons. We then performed Lasso regression using SciKit-learn with default parameters to predict each target input signal from the network firing rates. Specifically, we trained on the first 95 s of the activity data, and computed R^2 scores on the Lasso predictions over the last 5 s of held-out data (Fig. 2.5b).

Signal selectivity measures. We measured signal selectivity of each neuron to the two slow signals relative to their associated shuffled controls (Fig. 2.5d) using the following relative measure defined on the weights:

$$\chi^i = \frac{w^i_P - w^i_{P_{\mathrm{ctl}}}}{w^i_P + w^i_{P_{\mathrm{ctl}}}} \tag{2.27}$$

where w^i_P is the average synaptic connection strength from the signal pathways $P1/2$ onto excitatory neuron i, and $w^i_{P_{\mathrm{ctl}}}$ is the same but from the control pathways $P1/2_{\mathrm{ctl}}$.

Representational dimension. To quantify the dimensionality of the learned neuronal representations (Fig. 2.5f), we binned network spikes in 25 ms bins and computed the participation ratio (Eq. (2.12)) of the binned data.

Neuronal tuning analysis of the learned weight profiles. To characterise the receptive fields of each neuron (Fig. 2.5g,h), we plotted w_{P1} against w_{P2} for every neuron in the excitatory population (Figs. 2.5g,h; left), and colored the resulting weight vectors by mapping the cosine of the vectors with the x-axis (w_{P2}) to a diverging color map. Furthermore, we calculated the relative tuning index as follows

$$\chi^i_{\mathrm{rel}} = \frac{w^i_{P2} - w^i_{P1}}{w^i_{P2} + w^i_{P1}} \quad . \tag{2.28}$$

STDP induction protocols. To measure STDP curves, we simulated a single neuron using the spiking LPL rule (Eq. 2.19) with a learning rate of $\eta = 5 \times 10^{-3}$. In all cases, we measured plasticity outcomes from 100 pairings of pre- and postsynaptic spikes at varying repetition frequencies ρ. The postsynaptic neuron's membrane voltage was held fixed between spikes at -51mV for the entire duration of the protocol. To measure STDP curves, we set the initial synaptic weight at 0.5 and simulated 100 different pre-post time delays Δt chosen uniformly from the interval $[-50, 50]$ ms with $\rho = 10$ Hz. To measure the rate-dependence of plasticity, we repeated the simulations for fixed $\Delta t = \pm 10$ ms while varying the repetition frequency ρ.

Numerical simulations. All SNN simulations were implemented in custom C++ code [141] using the Auryn SNN simulator (v0.8.2-dev, commit 36b3c197). Throughout we used a 0.1 ms simulation time step. Simulations were run on seven Dell Precision workstations with eight-core Intel Xeon CPUs.

2.4.6 Statistics & Reproducibility

This article is a simulation study. No statistical method was used to predetermine sample size. No data were excluded from the analyses. The experiments were not randomized. The Investigators were not blinded to allocation during experiments and outcome assessment.

Data Availability

The deep learning tasks used the STL-10 and CIFAR-10 datasets, typically available through all major machine learning libraries. The original releases for these datasets can be found at `http://ai.stanford.edu/%7Eacoates/stl10/`, and `https://www.cs.toronto.edu/~kriz/cifar.html` respectively. For the Supplementary Figures, we further used the 3D Shapes dataset [128] available at `https://github.com/deepmind/3d-shapes/` and the MNIST dataset available at `http://yann.lecun.com/exdb/mnist/`.

Code Availability

- The simulation code to reproduce the key results is publicly available at `https://github.com/fmi-basel/latent-predictive-learning`.
- PyTorch and the Lightning framework are freely available at `https://pytorch.org` and `https://www.pytorchlightning.ai`.
- The Auryn spiking network simulator is available at `https://github.com/fzenke/auryn`.
- The Engauge Digitizer is available at `http://markummitchell.github.io/engauge-digitizer`.

Acknowledgements

We thank all members of the Zenke Group for comments and discussions that shaped this project, and Atul Kumar Sinha for many helpful suggestions. We are particularly grateful to Julian Rossbroich for providing invaluable insights throughout the course of this work. This project was supported by the Swiss National Science Foundation [grant number PCEFP3_202981] (FZ) and the Novartis Research Foundation (MSH & FZ). The funders had no role in study design, data collection and analysis, decision to publish or preparation of the manuscript.

Chapter III

Implicit variance regularization in non-contrastive SSL

Authors: Manu Srinath Halvagal[*], Axel Laborieux[*], Friedemann Zenke

A version of this chapter has been published as:

M. S. Halvagal, A. Laborieux, and F. Zenke. "Implicit variance regularization in non-contrastive SSL". in: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 63409–63436.

Author contributions: FZ concieved the study. MSH and AL performed the analysis, wrote code and ran simulations. All authors discussed the results and contributed to the final manuscript.

Abstract

Non-contrastive SSL methods like BYOL and SimSiam rely on asymmetric predictor networks to avoid representational collapse without negative samples. Yet, how predictor networks facilitate stable learning is not fully understood. While previous theoretical analyses assumed Euclidean losses, most practical implementations rely on cosine similarity. To gain further theoretical insight into non-contrastive SSL, we analytically study learning dynamics in conjunction with Euclidean and cosine similarity in the eigenspace of closed-form linear predictor networks. We show that both avoid collapse through implicit variance regularization albeit through different dynamical mechanisms. Moreover, we find that the eigenvalues act as effective learning rate multipliers and propose a family of isotropic loss functions (IsoLoss) that equalize convergence rates across eigenmodes. Empirically, IsoLoss speeds up the initial learning dynamics and increases robustness, thereby allowing us to dispense with the exponential moving average (EMA) target network typically used with non-contrastive methods. Our analysis sheds light on the variance regularization mechanisms

[*]These authors contributed equally.

of non-contrastive SSL and lays the theoretical grounds for crafting novel loss functions that shape the learning dynamics of the predictor's spectrum.

3.1 Introduction

SSL has emerged as a powerful method to learn useful representations from vast quantities of unlabeled data [23–28, 90, 145]. In SSL, the network's objective is to "pull" together its outputs for two differently augmented versions of the same input, so that they learn representations that are predictive across randomized transformations [146]. To avoid the trivial solution whereby the network output becomes constant, also called representational collapse, SSL methods use either a contrastive objective to "push" apart representations of unrelated images [22, 23, 26, 147–149] or other non-contrastive strategies. Non-contrastive methods comprise explicit variance regularization techniques [28, 90, 103], whitening approaches [91, 150], and asymmetric losses as in Bootstrap Your Own Latent (BYOL) [24] and SimSiam [25]. Asymmetric losses break symmetry between the two branches by passing one of the representations through a predictor network and stopping gradients from flowing through the other "target" branch. How this architectural modification prevents representational collapse is not obvious and has been the focus of several theoretical [151–155] and empirical studies [156–158]. A significant advance was provided by Tian, Chen, and Ganguli [151] who showed that linear predictors align with the correlation matrix of the embeddings, and proposed the closed-form predictor DirectPred based on this insight. However, previous analyses assumed a Euclidean loss at the output [151–153, 155] except [154], whereas practical implementations typically use the cosine loss [24, 25] which yields superior performance on downstream tasks. This difference raises the question whether analysis based on the Euclidean loss provides an accurate account of the learning dynamics under the cosine loss.

In this work, we provide a comparative analysis of the learning dynamics for the Euclidean and cosine-based asymmetric losses in the eigenspace of the closed-form predictor DirectPred. Our analysis shows how both losses implicitly regularize the variance of the representations, revealing a connection between asymmetric losses and explicit variance regularization in VICReg [28]. Yet, the learning dynamics induced by the two losses are markedly different. While the learning dynamics of different eigenmodes decouple in the Euclidean case, dynamics remain coupled for the cosine loss.

Moreover, our analysis shows that for both losses, the predictor's eigenvalues act as learning rate multipliers, thereby slowing down learning for modes with small eigenvalues. Based on our analysis, we craft an isotropic loss function (IsoLoss) for each case that resolves this problem and speeds up the initial learning dynamics. Furthermore, IsoLoss works without an EMA target network possibly because it boosts small eigenvalues, the purported role of the EMA in DirectPred [151]. In summary, our main contributions are the following:

- We analyze the SSL dynamics in the eigenspace of closed-form linear predictors for asymmetric Euclidean and cosine losses and show that both perform implicit variance regularization, but with markedly different learning dynamics.
- Our analysis shows that predictor eigenvalues act as learning rate multipliers which slows down learning for small eigenvalues.
- We propose isotropic loss functions for both cases that equalize the dynamics across eigenmodes and improve robustness, thereby allowing to learn without an EMA target network.

3.2 Eigenspace analysis of the learning dynamics

To gain a better analytic understanding of the SSL dynamics underlying non-contrastive methods such as BYOL and SimSiam [24, 25], we analyze them in the predictor's eigenspace. Specifically we proceed in three steps. First, building on DirectPred, we invoke the neural tangent kernel (NTK) to derive simple dynamic expressions of the predictor's eigenmodes for Euclidean and cosine loss. This formulation uncovers the implicit variance regularization mechanisms that prevent representational collapse. Using the eigenspace framework, we illustrate how removing the predictor or the stop-gradient results in collapse or run-away dynamics. Finally, we find that predictor eigenvalues act as learning rate multipliers for their associated mode, thereby slowing down learning for small eigenvalues. We derive a modified isotropic loss function (IsoLoss) that provides more equalized learning dynamics across modes, which showcases how our analytic insights help to design novel loss functions that actively shape the predictor spectrum. However, before we start our analysis, we will briefly review DirectPred [151] and the NTK [159], a powerful theoretical tool linking representational changes and parameter updates. We will rely on both concepts for our analysis.

3.2.1 Background and problem setup

We begin by reviewing DirectPred [151] and defining our notation. In the following, we consider a Siamese neural network $\boldsymbol{z} = f(\boldsymbol{x}; \boldsymbol{\theta})$ with output $\boldsymbol{z} \in \mathbb{R}^M$, input $\boldsymbol{x} \in \mathbb{R}^N$, and parameters $\boldsymbol{\theta}$. We further assume a linear predictor network $W_{\mathrm{P}} \in \mathbb{R}^{M\times M}$ and use the same parameters for the online and target branches as in SimSiam [25]. We denote pairs of representations as $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}$ corresponding to pairs of inputs $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}$ related through augmentation and implicitly assume that all losses are averaged over many augmented

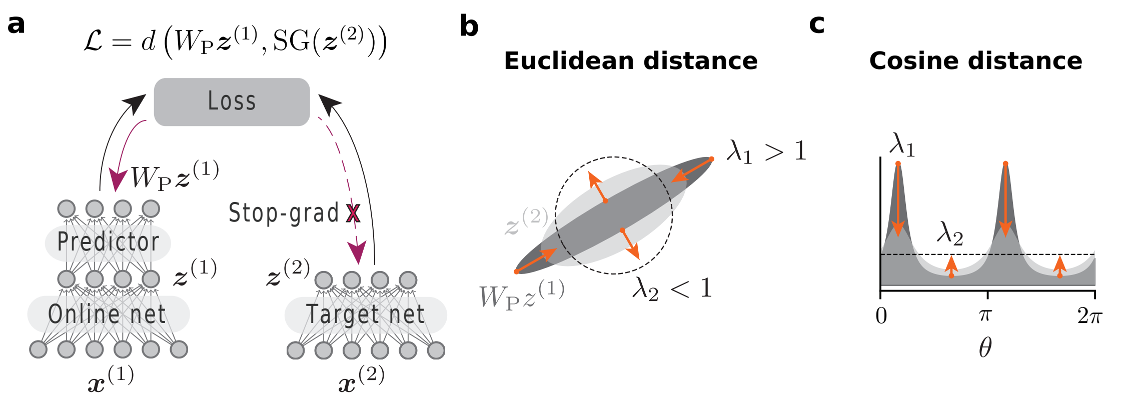


FIGURE 3.1: **(a)** Schematic of a Siamese network with a predictor network and a stop-gradient on the target network branch. The target network can be a copy (SimSiam [25]) or a moving average (BYOL [24]) of the online network. In either case, the target network is not optimized with gradient descent. **(b)** Visualization of learning dynamics under the Euclidean distance metric showing learning update directions along two eigenmodes, with the light cloud representing the distribution of the representations $\boldsymbol{z}$, the darker cloud representing the predictor outputs $W_P \boldsymbol{z}$, and the dotted circle indicates the steady state $\lambda_{1,2} = 1$, reached during learning. All eigenvalues converge to one. **(c)** Same as **(b)**, but for the cosine distance. The dotted line indicates the steady state $\lambda_1 = \lambda_2$.

pairs. The asymmetric loss function (Fig. 3.1a), introduced in BYOL [24], is then given by:

$$\mathcal{L} = d\left(W_P \boldsymbol{z}^{(1)}, \mathrm{SG}(\boldsymbol{z}^{(2)})\right),$$

where SG denotes the stop-gradient operation, and d is either the Euclidean distance metric $d(\boldsymbol{a}, \boldsymbol{b}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{b}\|^2$ or the cosine distance metric $d(\boldsymbol{a}, \boldsymbol{b}) = -\frac{\boldsymbol{a}^\top \boldsymbol{b}}{\|\boldsymbol{a}\|\|\boldsymbol{b}\|}$. We refer to the corresponding loss functions as $\mathcal{L}^{\mathrm{euc}}$ and $\mathcal{L}^{\mathrm{cos}}$ respectively.

DirectPred. Tian, Chen, and Ganguli [151] showed that a linear predictor in the BYOL setting aligns during learning with the correlation matrix of representations $C_{\boldsymbol{z}} := \mathbb{E}_{\boldsymbol{x}}\left[\boldsymbol{z}\boldsymbol{z}^\top\right]$, where the expectation is taken over the data distribution. Since the correlation matrix is a real symmetric matrix, one can diagonalize it over $\mathbb{R}$: $C_{\boldsymbol{z}} = U D_C U^\top$, where U is an orthogonal matrix whose columns are the eigenvectors of $C_{\boldsymbol{z}}$ and D_C is the real-valued diagonal matrix of the eigenvalues s_m with $m \in [1, M]$. Given this eigendecomposition, the authors proposed DirectPred, in which the predictor is not learned via gradient descent but directly set to:

$$W_P = f_\alpha(C_{\boldsymbol{z}}) = U D_C^\alpha U^\top \quad , \tag{3.1}$$

where α is a positive constant exponent applied element-wise to D_C. The eigenvalues λ_m of the predictor matrix W_P are then $\lambda_m = s_m^\alpha$. We use D to denote the diagonal matrix containing the eigenvalues λ_m. While DirectPred used $\alpha = 0.5$, the follow-up study DirectCopy [152] showed that $\alpha = 1$ is also effective while avoiding the expensive diagonalization step. While Tian, Chen, and Ganguli [151] based their analysis on the

Euclidean loss $\mathcal{L}^{\text{euc}}$, most practical models, including Tian et al.'s large-scale experiments, relied on the cosine similarity loss $\mathcal{L}^{\text{cos}}$. This discrepancy raises the question to what extent setting the predictor to the above expression is justified for the cosine loss. Empirically, we find that a trainable linear predictor *does* align its eigenspace with that of the representation correlation matrix also for the cosine loss (see Fig. B.1 in Appendix B.1).

Neural tangent kernel (NTK). The NTK is a powerful analytical tool characterizing the learning dynamics of neural networks [159, 160]. Here, we recall the definition of the *empirical* NTK [160] corresponding to a single instantiation of the network's parameters $\boldsymbol{\theta}$. If $|\mathtt{D}|$ denotes the size of the training dataset, $\mathcal{L} : \mathbb{R}^M \to \mathbb{R}$ an arbitrary loss function, $\mathcal{X}$, the training data concatenated into one vector of size $N|\mathtt{D}|$, and $\mathcal{Z} = z(\mathcal{X})$, the concatenated output of size $M|\mathtt{D}|$, then the empirical NTK is the $(M|\mathtt{D}| \times M|\mathtt{D}|)$-sized matrix:

$$\Theta_t(\mathcal{X}, \mathcal{X}) = \nabla_{\boldsymbol{\theta}} \mathcal{Z} \nabla_{\boldsymbol{\theta}} \mathcal{Z}^\top,$$

and the continuous-time gradient-descent dynamics [160] of the representations $\boldsymbol{z}$ are given by:

$$\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}t} = -\eta \Theta_t(\boldsymbol{x}, \mathcal{X}) \nabla_{\mathcal{Z}} \mathcal{L} \quad . \tag{3.2}$$

In other words, the empirical NTK links the representational dynamics $\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}t}$ under gradient descent on the parameters $\boldsymbol{\theta}$, and the "representational gradient" $\nabla_{\mathcal{Z}} \mathcal{L}$.

3.2.2 Implicit variance regularization in non-contrastive SSL

As a starting point for our analysis, we first express the relevant loss functions in the eigenbasis of the predictor network. We do this using a closed-form linear predictor as prescribed by DirectPred. In the following, we use $\hat{\boldsymbol{z}} = U^\top \boldsymbol{z}$ to denote the representation expressed in the eigenbasis.

Lemma 1. (Euclidean and cosine loss in the predictor eigenspace) *Let W_P be a linear predictor set according to DirectPred with eigenvalues λ_m, and $\hat{\boldsymbol{z}}$ the representations expressed in the predictor's eigenbasis. Then the asymmetric Euclidean loss $\mathcal{L}^{\text{euc}}$ and cosine loss $\mathcal{L}^{\text{cos}}$ can be expressed as:*

$$\mathcal{L}^{\text{euc}} = \tfrac{1}{2} \sum_m^M |\lambda_m \hat{z}_m^{(1)} - \mathrm{SG}(\hat{z}_m^{(2)})|^2 \quad , \tag{3.3}$$

$$\mathcal{L}^{\text{cos}} = -\sum_m^M \frac{\lambda_m \hat{z}_m^{(1)} \mathrm{SG}(\hat{z}_m^{(2)})}{\|D\hat{\boldsymbol{z}}^{(1)}\| \|\mathrm{SG}(\hat{\boldsymbol{z}}^{(2)})\|} \quad . \tag{3.4}$$

for which we defer the simple proof to Appendix B.2. Rewriting the losses in the eigenbasis makes it clear that the asymmetric loss with DirectPred can be viewed as an *implicit* loss function in the predictor's eigenspace, where the variance of each mode naturally appears through the λ_m terms. In the following analysis, we will show how the learning dynamics implicitly regularize these variances λ_m. From Eq. (3.3) we directly see that $\mathcal{L}^{\text{euc}}$ is a sum of M terms, one for each eigenmode, which decouples the learning dynamics, a fact first noted by Tian, Chen, and Ganguli [151]. In contrast, the form of $\mathcal{L}^{\text{cos}}$ yields coupled dynamics due to the $\|D\hat{\boldsymbol{z}}^{(1)}\| = \sqrt{\sum_k (\lambda_k \hat{z}_k^{(1)})^2}$ term in the denominator. This coupling arises from the normalization of the representation vectors to the unit hypersphere when calculating the cosine distance. The normalization effectively removes one degree of freedom and, in the process, adds a dependence between all the representation dimensions (Fig. 3.1b and 3.1c).

To get an analytic handle on the evolution of the eigen-representations $\hat{\boldsymbol{z}}$ as the encoder learns, we first note that if training were to update the representations directly, instead of indirectly through updating the weights $\boldsymbol{\theta}$, they would evolve along the following "representational gradients":

$$\nabla_{\hat{\boldsymbol{z}}^{(1)}} \mathcal{L}^{\text{euc}} = \left(D\hat{\boldsymbol{z}}^{(1)} - \hat{\boldsymbol{z}}^{(2)} \right) D \quad , \tag{3.5}$$

$$\nabla_{\hat{\boldsymbol{z}}^{(1)}} \mathcal{L}^{\text{cos}} = -\frac{D\hat{\boldsymbol{z}}^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\| \|\hat{\boldsymbol{z}}^{(2)}\|} + \frac{(D\hat{\boldsymbol{z}}^{(1)})^\top \hat{\boldsymbol{z}}^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} D^2 \hat{\boldsymbol{z}}^{(1)} \quad . \tag{3.6}$$

In practice, however, representations of different samples do not evolve independently along these gradients, but influence each other through parameter changes in $\boldsymbol{\theta}$. This interdependence of representations and parameters are captured by the empirical NTK $\Theta_t(\mathcal{X}, \mathcal{X})$ (cf. Eq. (3.2)). Because the NTK is positive semi-definite, loosely speaking, gradient descent on the parameters changes representations "in the direction" of the above representational gradients.

To see this link more formally, we express the NTK in the eigenbasis as $\hat{\Theta}_t(\mathcal{X}, \mathcal{X}) = \nabla_{\boldsymbol{\theta}} \hat{\mathcal{Z}} \nabla_{\boldsymbol{\theta}} \hat{\mathcal{Z}}^\top$ where $\hat{\mathcal{Z}} = \hat{z}_t(\mathcal{X}) = U^\top z_t(\mathcal{X})$. Since we are concerned with the learning dynamics in this rotated basis, we will rewrite Eq. (3.2) for continuous-time gradient descent for a generic loss function $\mathcal{L}$ as:

$$\frac{\mathrm{d}\hat{\boldsymbol{z}}}{\mathrm{d}t} = -\eta \hat{\Theta}_t(\boldsymbol{x}, \mathcal{X}) \nabla_{\hat{\mathcal{Z}}} \mathcal{L} \quad . \tag{3.7}$$

Note, that structurally these dynamics are the same as the embedding space dynamics in Eq. (3.2) but merely expressed in the predictor eigenbasis (see Lemma 2 in Appendix B.2 for a derivation). Although $\hat{\Theta}_t$ changes over time and is generally intractable in finite-width networks, it is positive semidefinite. This property guarantees that the cosine angle between the representational training dynamics under the parameter-space optimization of a neural network $\frac{\mathrm{d}}{\mathrm{d}t}\hat{\mathcal{Z}} \propto -\hat{\Theta}_t \nabla_{\hat{\mathcal{Z}}} \mathcal{L}$ and the dynamics that would result from optimizing the representations $\frac{\mathrm{d}}{\mathrm{d}t}\hat{\mathcal{Z}} \propto -\nabla_{\hat{\mathcal{Z}}}\mathcal{L}$ is non-negative:

$$\left\langle -\nabla_{\hat{\mathcal{Z}}}\mathcal{L}, \frac{\mathrm{d}\hat{\mathcal{Z}}}{\mathrm{d}t} \right\rangle = \eta \left\langle \nabla_{\hat{\mathcal{Z}}}\mathcal{L}, \hat{\Theta}_t \nabla_{\hat{\mathcal{Z}}}\mathcal{L} \right\rangle \geq 0.$$

In other words, the representational updates due to network training lie within a 180-degree cone of the dynamics prescribed by Eqs. (3.5) and (3.6). This guarantee makes it possible to draw qualitative conclusions about asymptotic collective behavior, e.g., whether a network is bound to collapse or not, from analyzing the more tractable dynamics that follow the representational gradients $\frac{\mathrm{d}}{\mathrm{d}t}\hat{\mathcal{Z}} \propto -\nabla_{\hat{\mathcal{Z}}}\mathcal{L}$ of the transformed BYOL/SimSiam loss. For ease of analysis, we now consider linear networks with Gaussian i.i.d inputs, an important limiting case amenable for theoretical analysis [161]. In this setting the empirical NTK becomes the identity and the simplified representational dynamics are exact, allowing us to fully characterize the representational dynamics for $\mathcal{L}^{\text{euc}}$ and $\mathcal{L}^{\cos}$ in the following two theorems. In the proofs for these theorems, we show that the assumption of Gaussian inputs can be relaxed further.

Theorem 1. (Representational dynamics under $\mathcal{L}^{\text{euc}}$) *For a linear network with i.i.d Gaussian inputs learning with $\mathcal{L}^{\text{euc}}$, the representational dynamics of each mode m independently follow the gradient of the loss $-\nabla_{\hat{z}}\mathcal{L}^{\text{euc}}$. More specifically, the dynamics uncouple and follow M independent differential equations:*

$$\frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t} = -\eta \frac{\partial \mathcal{L}^{\text{euc}}}{\partial \hat{z}_m^{(1)}}(t) = \eta \lambda_m \left(\hat{z}_m^{(2)} - \lambda_m \hat{z}_m^{(1)} \right) \quad , \tag{3.8}$$

which, after taking the expectation over augmentations yields the dynamics:

$$\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = \eta \lambda_m (1 - \lambda_m) \hat{z}_m \quad . \tag{3.9}$$

We provide the proof in Appendix B.2 and appreciate that $\frac{\mathrm{d}}{\mathrm{d}t}\hat{z}_m$ has the same sign as $\hat{z}_m$ whenever $\lambda_m < 1$ and the opposite sign whenever $\lambda_m > 1$. These dynamics are convergent and approach an eigenvalue λ_m of one, thereby preventing collapse of mode m. Since the eigenmodes are orthogonal and uncorrelated, and the condition simultaneously holds for all modes, this ultimately prevents both representational and dimensional collapse [127]. Since the eigenvalues also correspond to the variance of the representations, the underlying mechanism constitutes an *implicit* form of variance regularization. Finally, we note that the above decoupling of the dynamics for the Euclidean loss has been described previously in Tian, Chen, and Ganguli [151].

Nevertheless, the representational dynamics are different for the commonly used cosine loss $\mathcal{L}^{\cos}$.

Theorem 2. (Representational dynamics under $\mathcal{L}^{\cos}$) *For a linear network with i.i.d Gaussian inputs trained with $\mathcal{L}^{\cos}$, the dynamics follow a system of M coupled differential equations:*

$$\frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t} = \eta \frac{\lambda_m}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \sum_{k \neq m} \lambda_k \left(\lambda_k (\hat{z}_k^{(1)})^2 \hat{z}_m^{(2)} - \lambda_m \hat{z}_m^{(1)} \hat{z}_k^{(1)} \hat{z}_k^{(2)} \right) \quad , \tag{3.10}$$

and reach a regime in which the eigenvalues are comparable in magnitude. In this regime, the expected update over augmentations is well approximated by:

$$\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} \approx \eta \lambda_m \cdot \mathbb{E}\left[\frac{\hat{z}_m^2}{\|D\hat{\boldsymbol{z}}\|^3} \right] \cdot \mathbb{E}\left[\frac{\hat{z}_m}{\|\hat{\boldsymbol{z}}\|} \right] \cdot \sum_{k \neq m} \lambda_k \left(\lambda_k - \lambda_m \right), \tag{3.11}$$

where we have assumed averages over augmentations. See Appendix B.2 for the proof. Theorem 2 states that $\frac{\mathrm{d}}{\mathrm{d}t}\hat{z}_m$ has the same or different sign as $\hat{z}_m$ depending on the sign of the aggregate sum $\sum_{k \neq m} \lambda_k(\lambda_k - \lambda_m)$. This relation suggests that a steady state is only reached through mutual agreement when the non-zero eigenvalues are all equal. In contrast to the Euclidean case, there is no pre-specified target value (see Fig. B.2 in Appendix B.1). Thus, the cosine loss also induces implicit variance regularization, but through a markedly different mechanism in which eigenmodes cooperate.

3.2.3 Importance of stop-grad and predictor networks

We now extend our analysis to explain the known failure modes due to ablating the predictor or the stop-gradient for each distance metric. When we omit the stop-grad operator from $\mathcal{L}^{\text{euc}}$, we have:

$$\mathcal{L}^{\text{euc}}_{\text{noSG}} = \tfrac{1}{2}\|W_{\text{P}}\boldsymbol{z}^{(1)} - \boldsymbol{z}^{(2)}\|^2 \quad \Rightarrow \quad \frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = -\eta \left(1 - \lambda_m\right)^2 \hat{z}_m \quad , \tag{3.12}$$

so that $\frac{\mathrm{d}}{\mathrm{d}t}\hat{z}_m$ and $\hat{z}_m$ always have opposite signs (see Appendix B.3 for the derivation). This drives the representations toward zero with exponentially decaying eigenvalues, causing the notorious representational collapse [25]. Omitting the stop-grad operator from $\mathcal{L}^{\cos}$ yields a nontrivial expression for the dynamics causing the largest eigenmode to diverge (see Appendix B.3) . Interestingly, this is different from the collapse to zero inferred for the Euclidean distance.

Similarly, when removing the predictor network in the Euclidean loss case, the dynamics read:

$$\mathcal{L}^{\text{euc}}_{\text{noPred}} = \tfrac{1}{2}\|\boldsymbol{z}^{(1)} - \text{SG}(\boldsymbol{z}^{(2)})\|^2 \quad \Rightarrow \quad \frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = 0 \quad , \tag{3.13}$$

meaning that no learning updates occur. When the predictor is removed in the cosine loss case, the dynamics are:

$$\mathcal{L}^{\mathrm{cos}}_{\mathrm{noPred}} = -\frac{\left(\boldsymbol{z}^{(1)}\right)^{\top} \mathrm{SG}(\boldsymbol{z}^{(2)})}{\|\boldsymbol{z}^{(1)}\| \|\mathrm{SG}(\boldsymbol{z}^{(2)})\|}$$
$$\Rightarrow \frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = \eta \sum_{k\neq m} \left(\mathbb{E}\left[\frac{\hat{z}_k^2}{\|\hat{\boldsymbol{z}}\|^3}\right] \mathbb{E}\left[\frac{\hat{z}_m}{\|\hat{\boldsymbol{z}}\|}\right] - \mathbb{E}\left[\frac{\hat{z}_m \hat{z}_k}{\|\hat{\boldsymbol{z}}\|^3}\right] \mathbb{E}\left[\frac{\hat{z}_k}{\|\hat{\boldsymbol{z}}\|}\right]\right). \tag{3.14}$$

As we show in Appendix B.3, these dynamics also avoid collapse. However, the effective learning rates become impractically small without the eigenvalue factors from Eq. (3.11). We summarized the predicted dynamics of all settings in Table 3.1. Thus, our analysis provides mechanistic explanations for why stop-grad and predictor networks are required for avoiding collapse in non-contrastive SSL.

TABLE 3.1: Summary of eigendynamics as predicted by our analysis for linear networks.

Loss	$\mathrm{d}\hat{z}_m/\mathrm{d}t \propto$	Predicted dynamics
$\mathcal{L}^{\mathrm{euc}}$	$\lambda_m(1-\lambda_m)$	λs converge to 1, large ones faster.
$\mathcal{L}^{\mathrm{euc}}_{\mathrm{noSG}}$	$-(1-\lambda_m)^2$	All λs collapse.
$\mathcal{L}^{\mathrm{euc}}_{\mathrm{noPred}}$	0	No learning updates.
$\mathcal{L}^{\mathrm{euc}}_{\mathrm{iso}}$	$(1-\lambda_m)$	λs converge to 1 at homogeneous rates.
$\mathcal{L}^{\mathrm{cos}}$	$\lambda_m \sum_{k\neq m} \lambda_k(\lambda_k-\lambda_m)$	λs converge to equal values.
$\mathcal{L}^{\mathrm{cos}}_{\mathrm{noSG}}$	Appendix B.3	All λs diverge.
$\mathcal{L}^{\mathrm{cos}}_{\mathrm{noPred}}$	Appendix B.3	λs converge to equal values at low rates.
$\mathcal{L}^{\mathrm{cos}}_{\mathrm{iso}}$	$\sum_{k\neq m} \lambda_k(\lambda_k-\lambda_m)$	λs converge to equal values at homogeneous rates.

3.2.4 Isotropic losses

In Eqs. (3.9) and (3.11) the eigenvalues appear as multiplicative learning rate modifiers in front of the difference terms that determine the fixed point. Hence, modes with larger eigenvalues converge faster than modes with smaller eigenvalues, reminiscent of previous theoretical work on supervised learning [161]. We hypothesized that the anisotropy in learning dynamics could lead to slow convergence for small eigenvalue modes or instability for large eigenvalues. To alleviate this issue, we designed alternative isotropic loss functions that equalize relaxation dynamics for all eigenmodes by exploiting the stop-grad function. Put simply, this involves taking the dynamics from Eqs. (3.8) and (3.10), removing the leading λ_m term, and deriving the loss function that would result in the desired dynamics.

One such isotropic “IsoLoss” function for the Euclidean distance is:

$$\mathcal{L}_{\text{iso}}^{\text{euc}} = \tfrac{1}{2}\|\boldsymbol{z}^{(1)} - \text{SG}(\boldsymbol{z}^{(2)} + \boldsymbol{z}^{(1)} - W_{\text{P}}\boldsymbol{z}^{(1)})\|^2. \tag{3.15}$$

We note that this IsoLoss has the same numerical value as $\mathcal{L}^{\text{euc}}$, but the gradient flow is modified by placing the prediction inside the stop-grad and also adding and subtracting $\boldsymbol{z}^{(1)}$ inside and outside of the stop-grad. The associated idealized learning dynamics in our analytic framework are given by:

$$\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = \eta\,(1-\lambda_m)\,\hat{z}_m, \tag{3.16}$$

where the λ_m factor (cf. Eq. (3.9)) disappeared (Table 3.1). Similarly, for the cosine distance,

$$\mathcal{L}_{\text{iso}}^{\text{cos}} = -(\boldsymbol{z}^{(1)})^\top \text{SG}\left(\frac{\boldsymbol{z}^{(2)}}{\|W_{\text{P}}\boldsymbol{z}^{(1)}\|\|\boldsymbol{z}^{(2)}\|}\right) + \frac{1}{2}\text{SG}\left(\frac{(W_{\text{P}}\boldsymbol{z}^{(1)})^\top \boldsymbol{z}^{(2)}}{\|W_{\text{P}}\boldsymbol{z}^{(1)}\|^3\|\boldsymbol{z}^{(2)}\|}\right)\|W_{\text{P}}^{1/2}\boldsymbol{z}^{(1)}\|^2 \tag{3.17}$$

is one possible IsoLoss, in which $W_{\text{P}}^{1/2} = UD^{1/2}U^\top$ with the square-root applied element-wise to the diagonal matrix D. While this IsoLoss does not preserve numerical equality with the original loss $\mathcal{L}^{\text{cos}}$, it achieves the desired effect of removing the leading λ_m learning-rate modifier (cf. Table 3.1).

3.3 Numerical experiments

To validate our theoretical findings (cf. Table 3.1), we first simulated a small linear Siamese neural network as shown in Fig.3.1a, for which Theorems 1 and 2 hold exactly. We fed the network with independent standard Gaussian inputs, and generated pairs of augmentations using isotropic Gaussian perturbations of standard deviation $\sigma = 0.1$. We then trained the linear encoder with each configuration described above. Training the network with $\mathcal{L}_{\text{noSG}}^{\text{euc}}$ resulted in collapse with exponentially decaying eigenvalues, whereas $\mathcal{L}_{\text{noSG}}^{\text{cos}}$ succumbed to diverging eigenvalues as predicted (Fig. 3.2a). Training without the predictor caused vanishing updates for $\mathcal{L}_{\text{noPred}}^{\text{euc}}$ and slow learning for $\mathcal{L}_{\text{noPred}}^{\text{cos}}$, in line with our analysis (Fig. 3.2b). Optimizing $\mathcal{L}^{\text{euc}}$, the representations become increasingly isotropic with all the eigenvalues λ_m converging to one (Fig. 3.2c, top), whereas optimizing $\mathcal{L}^{\text{cos}}$ also resulted in the eigenvalues converging to the same value, but different from one (Fig. 3.2c, bottom). The anisotropy in the dynamics of different eigenmodes noted above is particularly striking in the case of the Euclidean distance (Fig. 3.2c). Training with $\mathcal{L}_{\text{iso}}^{\text{euc}}$ and $\mathcal{L}_{\text{iso}}^{\text{cos}}$ resulted in similar convergence properties as their non-isotropic counterparts, but the eigenmodes converged at more homogeneous rates (Fig. 3.2d). Finally, we confirmed that these findings were qualitatively similar in the corresponding nonlinear networks with ReLU activation

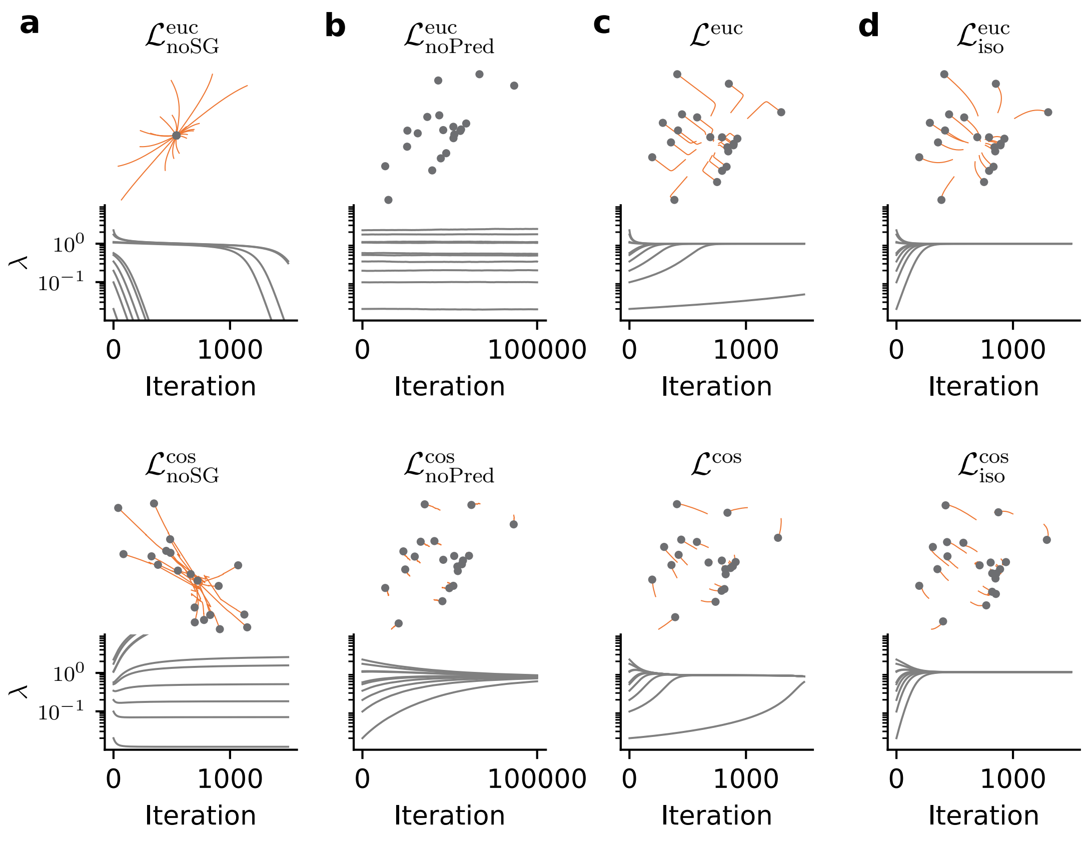


FIGURE 3.2: Evolution of representations (top) and eigenvalues (below) of W_P throughout training with different loss functions. The representational trajectories correspond to training with $M = 2$ for visualization and the points signify the final network outputs. The eigenvalues were computed with dimensions $N = 15$ and $M = 10$. **(a)** Omitting the stop-grad leads to representational collapse in the Euclidean case (top), and diverging eigenvalues for the cosine case (bottom). **(b)** No learning occurs without the predictor with the Euclidean distance, but learning does occur with the cosine distance, although at low rates. Note the change in scale of the time-axis. **(c)** Optimizing the BYOL/SimSiam loss leads to isotropic representations under both distance metrics. **(d)** Optimizing IsoLoss has the same effect, but with uniform convergence dynamics for all eigenvalues for both distance metrics.

(see Fig. B.3 in Appendix B.1). Thus, our theoretical findings hold up in simple Siamese networks.

3.3.1 Nonlinear networks and real-world datasets

To investigate how well our theoretical analysis holds up in non-toy settings, we performed several self-supervised learning experiments on CIFAR-10, CIFAR-100 [162], STL-10 [163], and TinyImageNet [164]. We based our implementation[1] on the Solo-learn library [165], and used a ResNet-18 backbone [166] as the encoder and the cosine loss, unless mentioned otherwise (see Appendix B.4 for details). As baselines for comparison, we trained the same

[1]Code is available at `https://github.com/fmi-basel/implicit-var-reg`

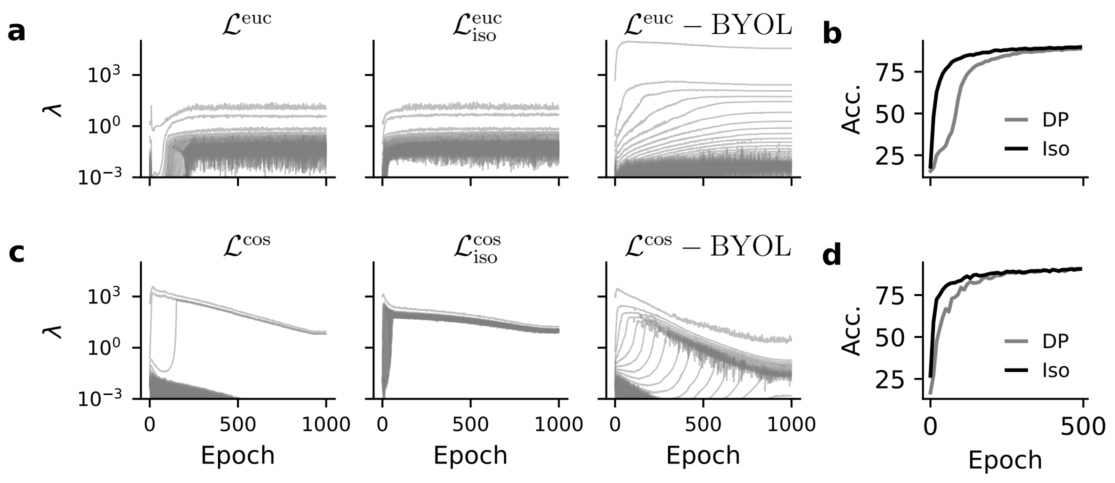


FIGURE 3.3: Learning dynamics for a ResNet-18 network trained with different loss functions. **(a)** Evolution of the eigenvalues of the representation correlation matrix during training for closed-form predictors as prescribed by DirectPred (left) and IsoLoss (center). Right: Standard BYOL with the nonlinear trainable predictor. For clarity, we plot only one in ten eigenvalues. Both $\mathcal{L}^{\text{euc}}$ and $\mathcal{L}^{\text{euc}}_{\text{iso}}$ drive the eigenvalues to converge quickly and remain constant thereafter with relatively small fluctuations (note the logarithmic scale). BYOL results in the eigenvalues being spread across a large range of magnitudes. **(b)** Linear readout validation accuracy for $\mathcal{L}^{\text{euc}}$ and $\mathcal{L}^{\text{euc}}_{\text{iso}}$ during the first 500 training epochs. IsoLoss accelerates the initial learning dynamics as predicted by the theory. **(c)** Same as in (a) but for the cosine distance. $\mathcal{L}^{\text{cos}}$ recruits few large eigenvalues, but drives them gradually to the same magnitude, whereas $\mathcal{L}^{\text{cos}}_{\text{iso}}$ quickly recruits *all* eigenvalues and causing them to converge to an isotropic solution. In contrast, BYOL recruits eigenvalues in a step-wise manner. **(d)** Same as (b) but for the cosine distance.

backbone using BYOL with the nonlinear predictor and DirectPred with the closed-form linear predictor. We recorded the online readout accuracy of a linear classifier trained on frozen features following standard practice, evaluated either on the held-out validation or test set where available.

We found that the eigenvalue dynamics of the representational correlation matrix in the ResNet-18 closely mirrored the analytical predictions for the closed-form predictor. For Euclidean distances (Fig. 3.3a), the eigenvalues for DirectPred and IsoLoss converged to a small range of values around one. However, the dynamics for BYOL with a learnable nonlinear predictor deviated significantly with the eigenvalues distributed over a larger range. Consistent with our analysis, IsoLoss had faster initial dynamics for the eigenvalues which also resulted in a faster initial improvements in model performance (Fig. 3.3b). The faster learning with IsoLoss was even more evident for the cosine distance (Fig. 3.3c). Surprisingly, BYOL, which uses a nonlinear predictor also closely matched the predicted dynamics in the case of the cosine distance. Furthermore, the dynamics showed a stepwise learning phenomenon wherein eigenvalues are progressively recruited one-by-one, consistent with recent findings for other SSL methods [167]. Finally, IsoLoss exhibited faster initial learning (Fig. 3.3d), in agreement with our theoretical analysis. Thus, our theoretical analysis accurately predicts key properties of the eigenvalue dynamics in nonlinear networks trained on real-world datasets.

3.3.2 Practical benefits of IsoLoss

To further investigate the impact of IsoLoss on learning, we first verified that it does not have any adverse effects on downstream classification performance. We found that IsoLoss matched or outperformed DirectPred on all benchmarks (Table 3.2) when trained with an EMA target network as used in the original studies. Yet, it performed slightly worse than BYOL, which uses a nonlinear predictor and an EMA target network. Because EMA target networks are thought to amplify small eigenvalues [151], we speculated that IsoLoss may work without it. We repeated training for the closed-form predictor losses without EMA to test this idea. We found that $\mathcal{L}^{\cos}_{\text{iso}}$ was indeed robust to EMA removal. However, it caused a slight drop in performance (Table 3.2) and a notable reduction in the recruitment of small eigenvalues (see Fig. B.4 in Appendix B.1). In contrast, optimizing the standard BYOL/SimSiam loss $\mathcal{L}^{\cos}$ with the symmetric linear predictor was unstable, as reported previously [151]. Finally, we confirmed the above findings also hold for $\alpha = 1$ (cf. Eq. (3.1)) as prescribed by DirectCopy [152] (see Table B.1 in Appendix B.1). Thus, IsoLoss allows training without an EMA target network.

Table 3.2: Linear readout validation accuracy in % ± stddev over five random seeds. The † denotes crashed runs, known to occur with symmetric predictors like DirectPred [151]. Starred values * were taken from the Solo-learn library [165].

Model	EMA	CIFAR-10	CIFAR-100	STL-10	TinyImageNet
BYOL	Yes	92.6*	70.5*	91.7 ± 0.1	38.3 ± 1.5
SimSiam	No	90.7 ± 0.2	66.3 ± 0.4	87.5 ± 0.7	39.8 ± 0.6
DirectPred ($\alpha = 0.5$)	Yes	92.0 ± 0.2	66.6 ± 0.5	88.8 ± 0.3	40.1 ± 0.5
	No	$12.1 \pm 1.3^{\dagger}$	$1.6 \pm 0.6^{\dagger}$	$10.4 \pm 0.1^{\dagger}$	$1.3 \pm 0.2^{\dagger}$
IsoLoss (ours)	Yes	91.5 ± 0.2	69.0 ± 0.2	89.0 ± 0.3	44.8 ± 0.4
	No	91.5 ± 0.2	64.3 ± 0.3	87.4 ± 0.1	40.4 ± 0.4

The above result suggests that IsoLoss promotes the recruitment of small eigenvalues in closed-form predictors. Another factor that has been implicated in suppressing recruitment is weight decay [152]. To probe how weight decay and IsoLoss affect small eigenvalue recruitment, we repeated the above simulations with EMA and different amounts of weight decay. Indeed, we observed less eigenvalue recruitment with increasing weight decay for DirectPred (Appendix B.1, Fig. B.5a), but not for IsoLoss (Fig. B.5b). However, for IsoLoss larger weight decay resulted in lower magnitudes of *all* eigenvalues. Hence, IsoLoss reduces the impact of weight decay on eigenvalue recruitment.

3.4 Discussion

We provided a comprehensive analysis of the SSL representational dynamics in the eigenspace of closed-form linear predictor networks (i.e., DirectPred and DirectCopy) for both the Euclidean loss and the more commonly used cosine similarity. Our analysis revealed how asymmetric losses prevent representational and dimensional collapse through *implicit* variance regularization along orthogonal eigenmodes, thereby formally linking predictor-based SSL with explicit variance regularization approaches [28, 90, 103]. Our work provides a theory framework which further complements the growing body of work linking contrastive and non-contrastive SSL [158, 168–171].

We empirically validated the key predictions of our analysis in linear and nonlinear network models on several datasets, including CIFAR-10/100, STL-10, and TinyImageNet. Moreover, we found that the eigenvalues of the predictor network act as learning rate multipliers, causing anisotropic learning dynamics. We derived Euclidean and cosine IsoLosses, which counteract this anisotropy and enable closed-form linear predictor methods to work without an EMA target network, thereby further consolidating its presumed role in boosting small eigenvalues [151].

To our knowledge, this is the first work to comprehensively characterize asymmetric SSL learning dynamics for the cosine distance metric widely used in practice. However, our analysis rests on several assumptions. First, the analytic link through the NTK between gradient descent on parameters and the representational changes is an approximation in nonlinear networks. Moreover, we assumed Gaussian i.i.d inputs for proving Theorems 1 and 2. Although these assumptions generally do not hold in nonlinear networks, our analysis qualitatively captures their overall learning behavior and predicts how networks respond to changes in the stop-grad placement.

In summary, we have provided a simple theoretical explanation of how asymmetric loss configurations prevent representational collapse in SSL and elucidate their inherent dependence on the placement of the stop-grad operation. We further demonstrated how the eigenspace framework allows crafting new loss functions with a distinct impact on the SSL learning dynamics. We provided one specific example of such loss functions, IsoLoss, which equalizes the learning dynamics in the predictor's eigenspace, resulting in faster initial learning and improved stability. In contrast to DirectPred, IsoLoss learns stably without an EMA target network. Our work thus lays out an effective framework for analyzing and developing new SSL loss functions.

Acknowledgments

This project was supported by the Swiss National Science Foundation [grant numbers PCEFP3_202981 and TMPFP3_210282], by EU's Horizon Europe Research and Innovation Programme (grant agreement number 101070374) funded through SERI (ref 1131-52302), and the Novartis Research Foundation. The authors declare no competing interests.

Chapter IV

Representation learning of sequences through recurrent predictor networks

Authors: Manu Srinath Halvagal[*], Ashena Gorgan Mohammadi[*], Friedemann Zenke

At the time of writing, the materials in this chapter are being prepared for submission to a peer-reviewed journal.

Author contributions: FZ, MSH, and AGM designed the study. MSH and AGM performed the analysis, wrote code, ran simulations, and wrote the initial manuscript. FZ supervised the project.

Abstract

Extracting disentangled and predictive representations of both invariant and dynamic latent factors underlying sensory data is crucial for understanding a changing world. Various computational models have been proposed to explain how the brain learns such representations, but they often rely on input-space predictions, are limited to invariant features, or require biologically implausible negative samples. We present a novel predictive learning model that captures both invariant and dynamic factors with recurrent network modules, by optimizing representation-space predictions in architectures without relying on negative samples. Through experiments on synthetic moving objects, handwritten digit sequences, and natural speech, we demonstrate that predictive learning consistently outperforms invariance-based approaches in capturing dynamic temporal properties. Beyond representing factors of the data, the learned representations enable accurate simulation through autoregressive rollouts and can be used to predict the future. Finally, we show that predictive learning remains effective in a biologically plausible layer-wise training setting without end-to-end backpropagation. These findings contribute to our understanding of how predictive principles may underlie representation learning in the brain.

[*]These authors contributed equally.

4.1 Introduction

Inferring the factors underlying sensory data is a critical step in how the brain understands and models the world. In natural environments, sensory data consist of sequences of observations such as a series of visual snapshots of an animal running or auditory signals of a bird singing. These sequences are governed by the underlying dynamics of the world, which can be described by abstract latent factors such as the kind of animal or bird, its location, and velocity. Although the latent factors may implicitly lie on simple and structured manifolds, they are often highly entangled in the sensory data (Fig. 4.1, left), making them hard to decode. The brain is thought to disentangle these factors into interpretable internal representations [104] that can be used, first and foremost, to make predictions about the future and inform behavior. The mechanisms by which the brain learns to extract such *predictive representations* from sequences of sensory stimuli remain poorly understood.

A diverse body of research has explored mechanisms for learning structured predictive representations from sensory data [11, 13, 15, 16, 20, 172]. Predictive coding frameworks [15] posit that neural circuits continuously generate predictions of future sensory stimuli and update representations based on prediction errors. Learning models in this framework typically compute prediction errors in raw sensory input-space [56, 173]. However, low-level details of the sensory data are often irrelevant for perceptual tasks [98]. Other work on slow feature analysis [16] has proposed that temporal continuity could serve as a learning signal for extracting slowly varying or invariant features that often correspond to meaningful latent factors. These ideas have inspired computational models of learning and synaptic plasticity centered around prediction errors in the internal representation-space [103, 114, 174]. To avoid collapsing to trivial representations, these models tend to require contrastive objectives to explicitly push apart unrelated representations through negative samples [114, 174] or regularization based on Hebbian plasticity [103]. However, they remain incomplete descriptions of biological representation learning. On the one hand, negative samples require temporally non-local comparisons of representations [22] or rapid switching of the sign of plasticity [114], for which there is as yet little experimental evidence. On the other hand, non-contrastive models [103] do not require negative samples but have thus far been limited to learning invariant features and have not been shown to learn more dynamic and fine-grained latent factors within invariant object categories. For instance, apart from distinguishing between types of animals, it is also important to represent their position, velocity, and body pose.

In this article, we propose a model for learning predictive representations from sequences of sensory observations (Fig. 4.1, center) that bridges these approaches. Our approach, which we refer to as RePL, explicitly optimizes representation-space prediction without requiring reconstruction of sensory inputs or negative samples. RePL leverages a combination of feedforward encoding to extract instantaneous features, recurrent processing to capture sequence dynamics and context via temporal integration, and a predictor network to generate predictions of future representations. Furthermore, RePL uses a predictive learning objective based on a more general notion of predicting future representations beyond pure invariance. We contrast this against a purely invariance learning objective [103], applied within the same architecture. We expect that RePL, with the predictive objective, can capture both invariant and dynamic latent factors underlying the observed data (Fig. 4.1, right).

We demonstrate our model on a diverse set of tasks related to recognizing spatiotemporal patterns in synthetic videos of moving objects, handwritten digit sequences, and natural speech. We find that predictive learning consistently forms disentangled representations of dynamic properties such as velocity and orientation in synthetic videos and phonemes in speech, whereas invariance learning does not. In particular, invariance learning fails to capture dynamic factors that require temporal integration, possibly due to generating spuriously predictive or invariant features using the recurrence. We also find that the representations learned under RePL with the predictive learning objective enable accurate simulation of the future through autoregressive rollouts. Finally, we show that RePL can learn predictive representations even without end-to-end backpropagation in a more biologically plausible layer-wise training setting. These findings suggest that RePL is a promising approach to learning both invariant and dynamic latent factors underlying sensory data and provides a plausible explanation of representation learning in the brain.

4.2 Results

To find disentangled representations of latent factors from sequences of sensory stimuli, we considered a predictive learning objective for the RePL model (Fig. 4.1), including both a feedforward and recurrent encoder followed by a predictor network. The predictive learning objective is designed to optimize the prediction of future feedforward representations $\boldsymbol{z}_{t+1}$ from the recurrent representations $\boldsymbol{c}_t$ at the current time step t. The predictive objective function is thus defined as:

$$\mathcal{L}_{\text{pred}} = \|p(\boldsymbol{c}_t) - \text{SG}(\boldsymbol{z}_{t+1})\|^2 \quad , \tag{4.1}$$

where p is the feedforward predictor network, and SG is the stop-gradient operator. An alternative promising approach to learn predictive representations is invariance learning, which suggests that predictive features are the ones that are invariant through time. We examine the invariance learning objective [103]:

$$\mathcal{L}_{\text{inv}} = \|p(\boldsymbol{c}_t) - SG(p(\boldsymbol{c}_{t+1}))\|^2 + \mathcal{L}_{\text{reg}} \quad , \tag{4.2}$$

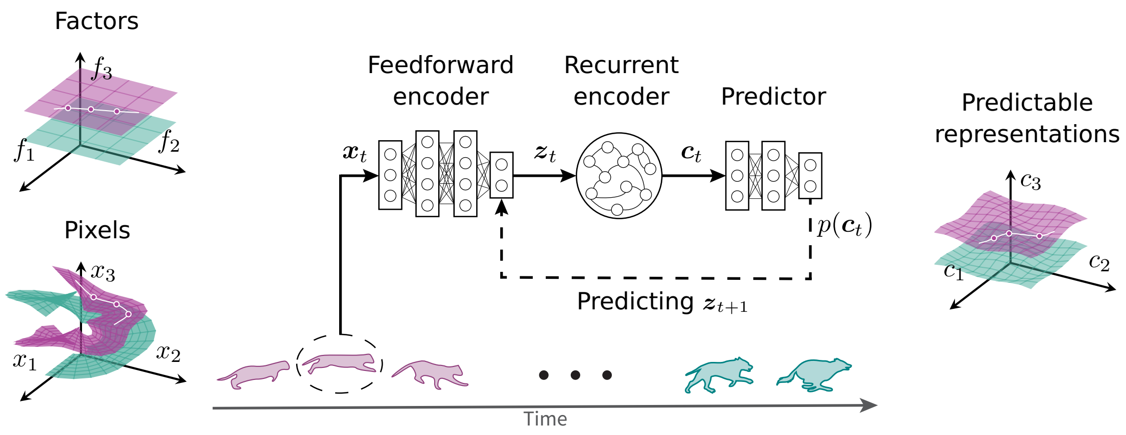


FIGURE 4.1: **The Recurrent Predictive Learning (RePL) model for learning latent factors from sequences.** Abstract latent factors $\boldsymbol{f}$ such as object identity, location, and velocity underlie sequences of observed images or sounds. Although such behaviorally relevant factors may implicitly lie on simple and structured manifolds, they are often highly entangled in the observed data $\boldsymbol{x}$ (left), making it difficult to decode them. We propose a predictive learning model that enables linear decoding of latent factors by learning to predict future representations (center). The model consists of a feedforward encoder that extracts instantaneous features of input stimuli, a recurrent encoder that additionally captures contextual and dynamical features, and a predictor network that generates predictions of future feedforward representations. By learning to extract features that are maximally predictive of the future, the network naturally disentangles the latent factors in the representation space $\boldsymbol{c}$ (right).

where $\mathcal{L}_{\text{reg}}$ is defined to prevent the representations to be collapsed into trivial solutions (Methods).

4.2.1 Predictive learning of motion factors from synthetic videos

Tracking the motion of objects in the environment is crucial for animals to respond and interact with their surroundings, escape predators, and catch prey. Motion parameters, therefore, constitute an important class of latent factors that the brain must learn to represent. To evaluate the effectiveness of predictive learning in forming disentangled representations of motion factors from sequences, we first considered a synthetic dataset of moving sprites (Fig. 4.2a; Methods). The dataset consisted of videos of sprites moving with constant translation and rotation speeds, and perfectly elastic collisions at the boundaries (Fig. 4.2b). The speeds were chosen randomly for each video. The sprites were characterized by several latent factors, namely the sprite identity class, location, velocity, planar orientation, and the corresponding rotation velocity. We trained a DNN (Fig. 4.1) to predict future representations using the two different objective functions: invariance learning and predictive learning (Fig. 4.2c,d; Methods). Additionally, we used a supervised baseline model trained to predict the true latent factors from the input images, and a randomly initialized network as controls. We then evaluated how well we could decode the latent factors from the learned recurrent representations using linear readouts.

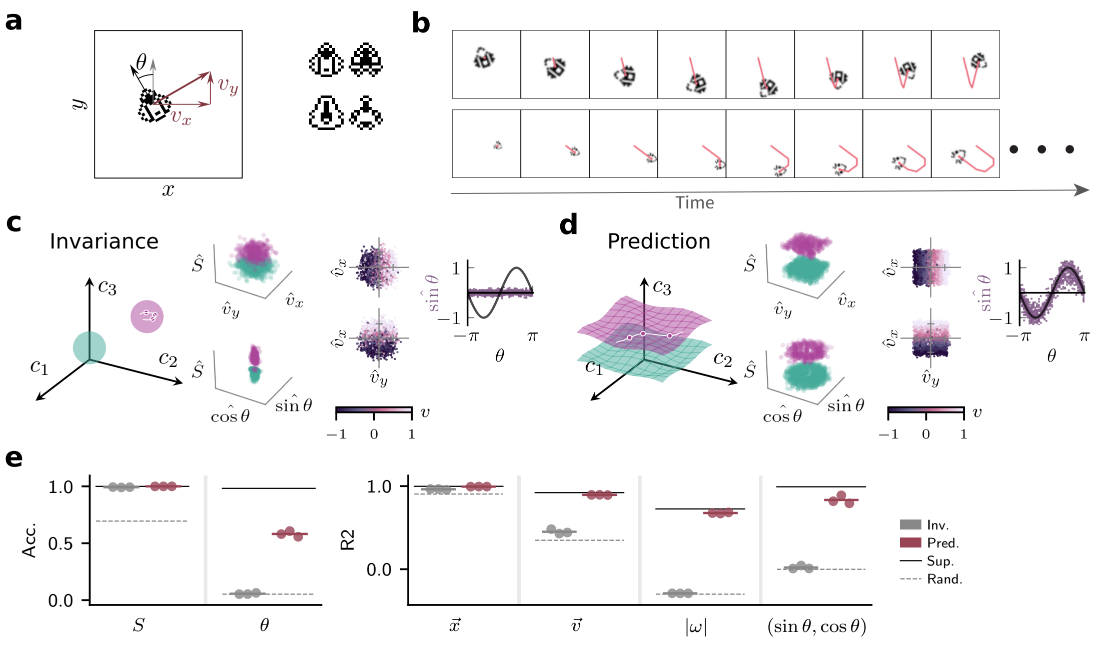


Figure 4.2: **Predictive learning develops disentangled representations of motion factors from videos of moving sprites.** **(a)** Video generation. The videos consist of sprites moving in a 2D plane with constant randomly chosen translation and rotation speeds, with perfectly elastic collisions at the boundaries. Several latent factors determine the appearance and motion of the sprites (left), including the sprite identity class S (right), location (x, y), velocity (v_x, v_y), and orientation θ. **(b)** Example frames from two videos. **(c)** The invariance learning objective, which aims to create compact manifolds (left), fails to learn the latent factors of velocity and orientation. A 3D view of the learned representations (center-left) projected along directions that are maximally predictive of sprite identity S and pairs of latent factors (v_x, v_y) (top) and $(\sin\theta, \cos\theta)$ (bottom), colored by S, shows that the different sprites are well-separated. However, the velocity projections colored by the true velocity (center-right) indicate that they are not informative of the true velocity values. Similarly, the sine of the orientation estimated from the representations fails to capture the true orientation (right). **(d)** Same as (b) but for the predictive learning objective (left). Predictive learning captures sprite identity (center-left), velocity (center-right), and orientation (right). **(e)** Linear readout accuracies and coefficients of determination (R^2) for several latent factors, decoded from the representations after invariance learning (Inv.), predictive learning (Pred.), supervised learning (Sup.), and random initialization (Rand.). We show decoding accuracies for sprite identity and discretized orientation, and R^2 values for planar position, translation velocity, rotation speed, and the sine and cosine of the orientation. Predictive learning yields high accuracies and R^2 values for all latent factors, whereas invariance learning yields high accuracy only for sprite identity and position.

For each latent factor, we found the maximally predictive direction in the learned representation space of the recurrent encoder using linear regression or classification and plotted the projections of the representations along these directions (Fig. 4.2c,d). We found that both invariance and predictive objectives learned to separate the different sprite identity classes in the representation space. However, the velocity projections colored by the true velocity values showed that invariance learning failed to encode the velocity latent factors, whereas predictive learning encoded them accurately. Similarly, the sine of the orientation estimated from the representations captured the true orientation after predictive learning, but not invariance learning. Importantly, the maximally predictive directions for the latent factors were all nearly orthogonal to each other after predictive learning (Supplementary Fig. C.1), indicating that the learned representations were disentangled [175]. The full set of projections coding for the latent factors is shown in Supplementary Fig. C.2.

We quantified the decoding of latent factors (Fig. 4.2e) from the learned recurrent representations with the linear readout accuracies and coefficients of determination (R^2 values). Predictive learning yielded high accuracies and R^2 values for all latent factors, in most cases close to the supervised baseline, whereas invariance learning yielded high accuracy only for sprite identity and position. For most factors, invariance learning did not even outperform the network at initialization. Sprite identity is an invariant factor that is preserved across time, explaining why invariance learning was successful in capturing it. The position was also captured by invariance learning but is not an invariant factor per se. We hypothesize two reasons for this observation: the position remains somewhat invariant over time in slow trajectories and, moreover, the position is trivially accessible due to the convolutional architecture used for the feedforward encoder. While the argument of limited invariance also holds for the orientation and velocities, these are not trivially accessible from the architecture. On the one hand, orientation is generally a more difficult feature to learn for convolutional architectures. On the other hand, velocities are dynamical properties, and invariance learning appeared to be unable to capture dynamical features in any task we tested, despite the inclusion of the recurrent encoder in the architecture. Overall, these results demonstrate that predictive learning can develop disentangled representations of motion factors from sequences of moving sprites.

Predictive simulation of motion trajectories in latent space In addition to representing the motion factors of objects in the environment, another important ability for intelligent behavior is to predict the future state of the environment based on past observations. We therefore wondered whether the learned predictive representations could be used to simulate the future trajectory of a sprite. To test this, we provided the network with the first 8 frames of a sprite trajectory as context and iteratively simulated the representations for future time steps using autoregressive rollouts (Fig. 4.3a; Methods). We found that the simulated trajectories closely matched the ground truth trajectories (Fig. 4.3b; Supplementary Fig. C.3). We also calculated the prediction error of the simulated rollouts for different latent factors over time (Fig. 4.3c). The prediction error was calculated as the mean squared error between the predicted and true values of the latent factors. We found that the prediction error was low for all latent factors but increased gradually over the time horizon of prediction. In particular, the position predictions remained accurate for many more frames than the velocity and orientation predictions. These results demonstrate that the learned predictive representations can be used to simulate future motion of a sprite.

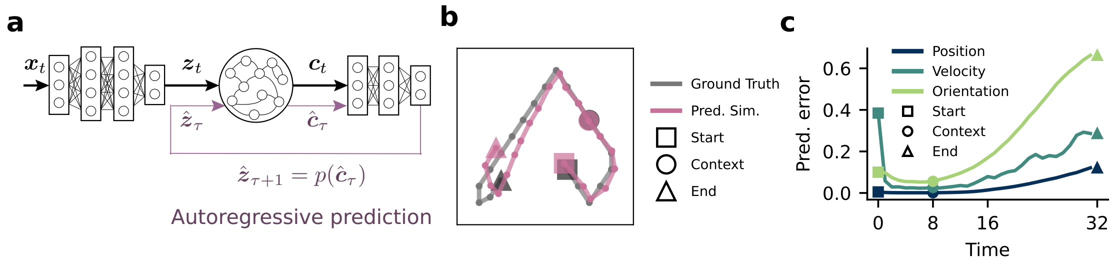


FIGURE 4.3: **Predictive representations can be used to simulate rollouts of future trajectories.** **(a)** Procedure for simulating sprite trajectory rollouts. The network is provided with the first 8 frames of a sprite video as context and is tasked with autoregressively predicting representations of the future frames. During the context window, the network is provided with the true sprite images, and during the rollout, the network uses the predictor network's outputs as input to the recurrent encoder for the next time step. **(b)** Trajectory estimated from the learned representations (colored) compared with the ground truth trajectory (grey) for 20 time steps. The 12 time steps after the context window are simulated. The simulated trajectory closely matches the ground truth trajectory. **(c)** Mean-squared prediction error of the simulated rollouts for different latent factors over time. The prediction error is low for all latent factors but increases gradually over time.

4.2.2 Predictive learning of abstract representations

Although predictive learning was successful in disentangling the motion factors of moving sprites, the images of the sprites were derived from only four underlying images, making the recognition task relatively easy. We next tested the effectiveness of predictive learning in developing abstract representations of categories and temporal properties of sequences of more complex data, such as images of handwritten digits and speech recordings.

Predictive learning of representations of digit sequences

We first exposed the network to sequences of triplets of handwritten digit images sampled from one of three digit clusters, with the underlying cluster changing with a probability of 0.2 (Fig. 4.4a; Methods). “Zero" digits were interspersed in the sequence to indicate cluster changes. Each digit cluster consisted of three digits, so every triplet constituted some unique permutation of the digits in the cluster. Additionally, the zero digit was considered as a separate cluster. We trained a DNN to predict future representations from current representations using the same invariance and predictive objectives as before, along with the supervised and random baselines (Methods). We then evaluated the learned representations by decoding the digit, digit cluster, and triplet identity using linear readouts (Fig. 4.4b).

The digit, cluster, and triplet labels constitute three different levels of abstraction at different timescales in the data. The digit label is an instantaneous property of the data lasting for only one time step, the triplet label is a property that persists for precisely three time steps and depends on temporal order, and the cluster label is a more invariant property that persists for the duration of an underlying cluster, 15 time steps on average.

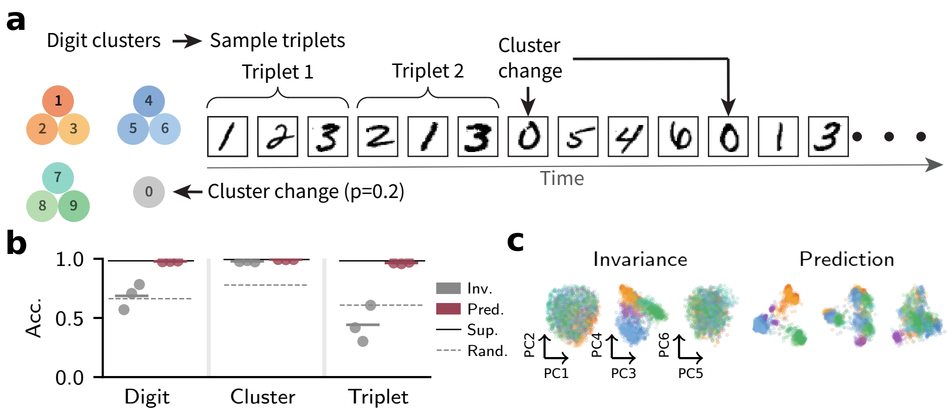


FIGURE 4.4: **Simultaneous representation and sequence learning.** **(a)** Schematic of the digit triplet sequence structure. The sequence consists of triplets of digits sampled from one of three digit clusters (left). The underlying cluster changes with a probability of 0.2, indicated by an intervening zero, or remains the same for the next triplet. For any given digit specified by the sequence generation process, a randomly chosen handwritten image of that digit is presented to the network (right). **(b)** Linear decoding accuracy of the digit, digit cluster, and triplet identity from the representations after invariance learning (Inv.), predictive learning (Pred.), supervised learning (Sup.), and random initialization (Rand.). Predictive learning yields high accuracy for all three tasks on par with supervised learning, whereas invariance learning yields reliably high accuracy only for the digit cluster. **(c)** Principal component projections of the learned representations colored by the digit identity. Predictive learning shows a clear separation between the different digit classes, whereas invariance learning shows a clear separation between digit clusters but not between individual digits.

To be able to predict the future as prescribed by the predictive objective, the network must learn to represent all three properties.

We found that predictive learning yielded high accuracy for all three tasks for readout from the recurrent encoder, indicating that the network had learned to represent abstract features for both individual input categories as well as the temporal structure of the sequences (Fig. 4.4b). In contrast, invariance learning yielded reliably high accuracy only for the digit cluster but not for the individual digits or triplets. The first six principal component projections of the learned representations showed that this is because invariance learning failed to separate the individual digits and formed coarse groupings of the digit clusters, whereas predictive learning developed finer-grained representations (Fig. 4.4c). This indicates that invariance learning tends to focus on the most invariant properties of the data, whereas predictive learning can capture both invariant and dynamic properties. Similar observations hold for the feedforward encoder representations (Supplementary Note C.2). Overall, these results demonstrate that predictive learning can develop abstract representations of categories as well as simple temporal properties from sequences of handwritten digits.

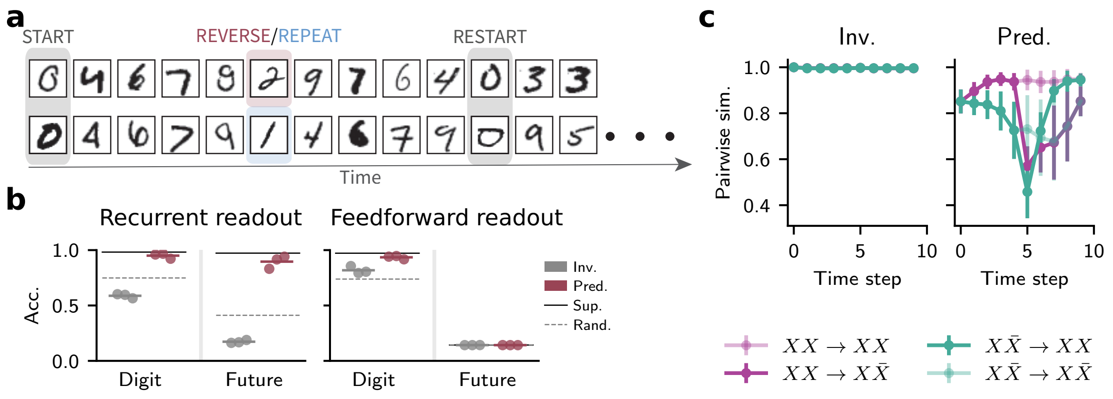


FIGURE 4.5: **Predictive learning can capture contextual properties of complex sequences. (a)** Sequences of digit images in the working memory repeat/reversal task. The sequences consist of a 4-digit prefix followed by a cue to indicate either a repeat or reversal condition, and a suffix that is identical to the prefix in the repeat condition and reversed in the reversal condition. **(b)** Comparison of linear decoding accuracy of the digit and the future digit sequence from the feedforward and recurrent representations after invariance learning (Inv.), predictive learning (Pred.), supervised learning (Sup.), and random initialization (Rand.). The digit label is decoded at every time step, and the future digit sequence is decoded from the representation at the time step when the reverse/repeat cue is presented. Predictive learning yields high accuracy for both tasks, close to the supervised baseline. Invariance learning yields very low accuracies, below the random initialization baseline. **(c)** Similarities of the learned representations between pairs of sequences with the same (or reversed) digit prefix that are each cued for repetition or reversal resulting in identical (or reversed) suffixes. We show representational similarities for both methods in (b). Predictive learning shows similarity values that reflect whether the pair of sequences share the same prefix or suffix, whereas invariance learning shows high similarity between all pairs of sequences at all time steps.

Predictive learning of contextual sequence properties

We wanted to further test the ability of predictive learning to capture more complex temporal properties of sequences. We therefore designed a contextual repeat/reversal task, in which sequences consisted of digit prefixes X followed by a repetition X or a reversed version $\bar{X}$ depending on a cue presented in the middle of the sequence (Fig. 4.5a). We trained a DNN on this sequence using the same invariance and predictive objectives as before and evaluated the learned representations by decoding the current digit and the future digit sequence using linear readouts (Fig. 4.5b). We found that predictive learning yielded high accuracy for both tasks, close to the supervised baseline, whereas invariance learning yielded very low accuracies, below the random initialization baseline.

We also compared the average pairwise representational similarities for four pairs of sequences (Methods): (i) pairs sharing both prefix and suffix, (ii) pairs sharing the same prefix but reversed suffixes, (iii) pairs with reversed prefixes and the same suffix, (iv) pairs with both the prefixes and suffixes reversed with respect to each other. For the representations learned under the predictive objective, we saw that pairs of sequences with the same prefix were similar until the cue was presented, after which they remained similar or became dissimilar depending on the cue (Fig. 4.5c). Similarly, the representations of pairs of sequences with reversed prefixes were dissimilar until the cue, after which they became similar or remained dissimilar depending on the cue. In contrast, the representations learned by invariance learning showed high similarity between all pairs of sequences at all time steps, regardless of the cue. These results demonstrate that predictive learning can capture complex temporal properties of sequences with in-context cues, whereas invariance learning cannot.

Predictive learning of representations for natural speech

Having established the effectiveness of predictive learning in capturing abstract representations of categories as well as temporal properties from sequences of synthetic data, we next tested how predictive learning fares on naturalistic data sequences. Specifically, we trained a DNN on audio recordings of speech utterances from the Librispeech [176] corpus with the same invariance and predictive objectives as before. Next, we evaluated the learned representations by decoding the speaker identity and phoneme sequence using linear readouts (Fig. 4.6b; Methods).

Speaker identity is an invariant property that is preserved across an entire audio recording, whereas the phoneme sequence is a temporal property that varies dynamically. We found that predictive learning yielded high accuracy for both tasks, whereas invariance learning yielded high accuracy only for the speaker identity. The phoneme sequence decoding accuracy was significantly lower for invariance learning compared to predictive learning and similar to the decoding accuracy for random initialization, indicating that invariance learning failed to capture the temporal properties of the phoneme sequences. These results demonstrate that predictive learning can capture both invariant and dynamical properties of natural speech.

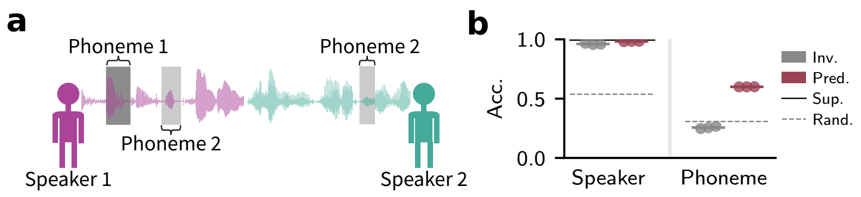


FIGURE 4.6: **Predictive learning of speech representations. (a)** A simple schematic of the speech dataset. The dataset consists of audio recordings of speech utterances from the Librispeech corpus, with labels indicating the speaker identity and phoneme sequence. **(b)** Linear decoding accuracy of the speaker identity and phoneme sequence from the representations after invariance learning (Inv.), predictive learning (Pred.), supervised learning (Sup.), and random initialization (Rand.). Predictive learning yields high accuracy for both tasks. Invariance learning yields high accuracy only for the speaker identity.

4.2.3 Predictive learning requires non-contextual prediction targets

For every task we have considered, predictive learning is more effective than invariance learning at learning dynamical factors underlying sequential data. We hypothesize that this arises due to the contextual information being present in both the predictor function and target function in the case of invariance learning. This effectively makes it possible for the networks to "construct" their own invariant features using recurrence that then satisfy the learning objective, and largely ignore the input stimuli. To validate this hypothesis, we considered a context-predictive objective, where the recurrent representation $\boldsymbol{c}$ was used as the target for prediction instead of the feedforward representation $\boldsymbol{z}$ (Methods). When we trained the same DNNs with this objective on the sprite videos and LibriSpeech, we observed that the networks failed to learn most latent factors. The decoding performance was below even that of invariance learning for most tasks that required any representation learning (Table 4.1).

TABLE 4.1: Linear readout validation accuracy or R^2 ± stddev over three random seeds.

Objective	Sprite Videos		Librispeech	
	Sprite ID (Acc.)	$\lvert\omega\rvert$ (R^2)	Speaker (Acc.)	Phoneme (Acc.)
Pred.	$> 0.99 \pm 0.01$	0.68 ± 0.01	0.95 ± 0.01	0.60 ± 0.01
Ctx. Pred.	$> 0.99 \pm 0.01$	-0.26 ± 0.04	0.65 ± 0.02	0.30 ± 0.01
Inv.	$> 0.99 \pm 0.01$	-0.29 ± 0.01	0.96 ± 0.01	0.26 ± 0.01

Layer-wise predictive learning

So far, we considered predictive learning in single networks that are trained end-to-end with backpropagation to optimize one global predictive objective. However, the anatomical constraints required for backpropagation are thought to be biologically implausible. We therefore considered a layer-wise predictive learning procedure that eases these restrictions, similar to recent work on predictive plasticity [103, 114]. Here, each layer of the network is trained independently to predict future representations of the layer's feedforward representation $\boldsymbol{z}$ (Fig. 4.7a). We trained a network with this layer-wise predictive learning procedure on the sprite videos from Fig. 4.2 and evaluated the learned representations by decoding the latent factors from the recurrent representations of each layer (Fig. 4.7b).

We found that the linear decoding accuracies and R^2 values for the latent factors increased with the depth of the network, reaching similar levels as the network trained end-to-end with backpropagation for most latent factors. Importantly, every layer of this network was identical with the same number of units in the encoders and predictor, indicating that the increase in decoding performance was solely due to the network extracting progressively better latent factor representations, and not due to differences in the layer architectures. These results demonstrate that predictive learning is effective in developing representations of instantaneous and dynamic latent factors even in a layer-wise local training setting and pave the way for biologically plausible circuit-level models of predictive learning.

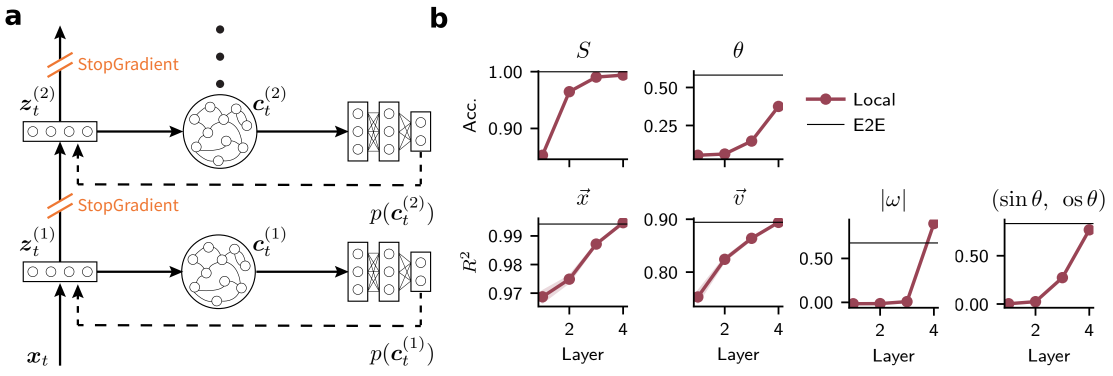


FIGURE 4.7: **Layer-wise predictive learning. (a)** Schematic of the layer-wise predictive learning procedure. The network consists of stacked identical gradient-isolated layers. Every layer l has its own recurrent encoder and predictor network, and is trained to predict future representations of that layer's feedforward representation $\boldsymbol{z}^{(l)}$. **(b)** Linear decoding accuracies and coefficients of determination (R^2) for the latent factors of the sprite videos from Fig. 4.2e decoded from the recurrent representations $\mathbf{c}^{(l)}$ of each layer after layer-wise local predictive learning. The decoding accuracies and R^2 values are plotted over the hierarchy of the locally trained network (Local), and compared with the network trained end-to-end with backpropagation (E2E; Pred. from Fig. 4.2e).

4.3 Discussion

We have shown that RePL, a novel model of predictive learning in representation-space, can develop disentangled representations of latent factors from sequences of synthetic and naturalistic stimuli. RePL developed abstract representations of invariant and dynamical properties of the stimulus sequences using a predictive learning objective and a recurrent network architecture, consistently outperforming invariance learning in capturing the dynamical latent factors. We found that the prediction target in the predictive objective is crucial for learning useful representations. Specifically, the prediction target should be context-free non-recurrent features of the data in order to avoid constructing spurious predictive features that are uninformative of the input data. We also showed that RePL can be used to reliably simulate the future. We also found that RePL can develop abstract representations of latent factors even in a layer-wise local training setting. These results suggest that predictive learning is a powerful framework for learning abstract representations from sensory data and can be a useful model for understanding how the brain learns to represent the world.

The predictive learning objective presented here is one particular implementation of the general idea that the brain is constantly making predictions about the world and updating its internal representations to minimize the prediction error. This notion is supported by recent experimental evidence pointing to prediction as a general principle of cortical structure and function [47, 48, 52, 57, 177–179] and is at the core of long-standing predictive coding theories of brain function [15]. In particular, experimental findings suggesting that the visual cortex *straightens* representational trajectories relative to raw sensory inputs [180, 181], the so-called straightening hypothesis, is of particular relevance to our work. The layer-wise predictive learning architecture (Fig. 4.7a) we use is also reminiscent of the dual counterstream architecture that has been proposed as a description of feedforward and feedback pathways in the cortical structure [182]. In particular, our work suggests that the feedforward pathway may play the role of prediction targets for the recurrent feedback pathway and that the recurrent pathway may be responsible for updating the feedforward pathway to minimize prediction errors.

Several computational models have been proposed to explain how the brain might learn to make predictions and update its internal representations to minimize prediction error [103, 114, 183]. Our work differs from and improves upon each of these models in important ways. First, the model presented here moves beyond learning invariant or slow features [16, 103] to learning predictive features [99] that are more generally informative of the future. Second, our model is non-contrastive and requires no negative samples, unlike plasticity models based on contrastive learning [114]. Finally, we focus on representation-space prediction as opposed to input-space prediction [183]. Representation-space prediction is more compelling from both a biological perspective, as it eliminates the need for decoder networks to compute prediction errors at the sensory periphery, and a computational perspective, as it allows for focusing on abstract representations rather than low-level details of sensory data [98]. Overall, our work is the first to highlight the importance of non-recurrent prediction targets in predictive learning models, and to show that predictive learning can simultaneously develop abstract representations of both invariant categories and dynamic properties from sensory data.

Why does invariance learning, and more generally context-predictive learning, fail to capture dynamic factors? We hypothesize that this occurs due to shared access to contextual information for both the predictor and target functions. In this case, the network can satisfy the learning objective by using the recurrent encoder to construct artificially invariant or predictive representations of the shared context without actually encoding instantaneous input-derived latent factors. This problem is particularly pronounced for invariance learning and dynamic features like velocity that require temporal integration. Although velocity remains physically constant between collisions, its neural representation requires maintaining information across time steps—precisely the processing that invariance learning discourages by collapsing temporally adjacent states.

Our work is also closely related to the broader field of SSL. SSL aims to learn useful representations from unlabeled data by training neural networks on pretext tasks that involve predicting parts of the data from other parts, to then use the learned representations for downstream tasks [22–25, 28, 89]. Our work can be seen as a particular instance of the JEPA framework [94] for SSL, with a specific focus on predicting future representations from current representations as a pretext task. Several instantiations of the JEPA framework have recently been proposed [29, 84, 87, 97, 184], with the seq-JEPA model [97] and the VJ-VCR model [87] being the most similar to our work. While VJ-VCR required regularization objectives in order to prevent representation collapse, the predictive objective in RePL relies solely on the asymmetry between prediction and target functions [144, 151] that would naturally arise in a microcircuit implementation of predictive learning with specialized prediction neurons. In turn, seq-JEPA considered arbitrary transformations between consecutive stimuli and not predictable sequences as in our work. Therefore, the model requires additional action inputs specifying the transformation between consecutive time steps in order to learn equivariant features such as orientation, which are naturally learned in our model due to their predictable evolution. That said, incorporating self-motion or other self-generated actions [185] into our model could be an interesting future direction.

This work has several limitations that should be addressed in future work. The link between the predictive learning model proposed here and the brain is still largely speculative. We expect that further work tying this computational model to detailed biologically plausible circuit-level models provides a concrete path towards understanding whether and how the brain uses predictive principles to learn and represent the world. In particular, it would be interesting to investigate how the predictive learning model can be explicitly mapped onto the dual counterstream architecture [182]. Additionally, the predictive learning model presented here is a feedforward-recurrent architecture that is trained with backpropagation, which is thought to be biologically implausible. While we have shown that layer-wise local training can be effective for predictive learning, this still requires backpropagation within each layer. It would be interesting to investigate how biologically plausible approaches to credit assignment [72, 107–109, 186–189] could be used within these gradient-isolated layers. Notably, such approaches tend to be more competitive with backpropagation for shallow networks, so they may remain effective in replacing the few steps of backpropagation required for the layer-wise predictive learning procedure presented here.

Extracting abstract representations of invariant and dynamic features of the world is a crucial function performed by the brain to enable intelligent behavior. We have presented a predictive learning model that is effective at learning such representations from sequences of a variety of complex data types. Beyond providing a computational model of how the brain might learn from sequences, our work offers broader implications for understanding how predictive representations support adaptive behavior. By learning to anticipate the future based on current and past observations, animals can prepare appropriate responses to dynamic environmental challenges and opportunities.

4.4 Methods

Predictive and invariance objective functions

The different objective functions considered in this work all aim to predict future representations $\boldsymbol{z}_{t+1}$ from the current representation $\boldsymbol{z}_t$ in the latent space, and take the form:

$$\mathcal{L} = \|f_{\text{pred}}(\boldsymbol{z}_t) - \text{SG}(f_{\text{tgt}}(\boldsymbol{z}_{t+1}))\|^2 + \mathcal{L}_{\text{reg}} \quad , \tag{4.3}$$

where f_{pred} is a predictor function, f_{tgt} is a target function, $\mathcal{L}_{\text{reg}}$ encapsulates any regularisation losses required to prevent representation collapse [28], and SG is the stop-gradient operator. The stop-gradient operator ensures that the weight update is not evaluated with respect to future representations. To obtain accurate predictions of future representations, the predictor function should be able to access contextual information in addition to the current representation. This is achieved by using a recurrent encoder $f_{\text{rec}}(\boldsymbol{z}_t)$ to include a context-informed recurrent representation $\boldsymbol{c}_t$ in the predictor function:

$$\begin{aligned} f_{\text{pred}}(\boldsymbol{z}_t) &= p \circ f_{\text{rec}}(\boldsymbol{z}_t) \\ &= p(\boldsymbol{c}_t) \quad , \end{aligned} \tag{4.4}$$

where p is a feedforward predictor network.

Predictive learning objective. While the predictor function requires context-informed recurrent representations, the target function can simply be the future feedforward representation, i.e., $f_{\text{tgt}}(\boldsymbol{z}_{t+1}) = \boldsymbol{z}_{t+1}$. This leads to the predictive learning objective function used in the RePL model:

$$\begin{aligned} \mathcal{L}_{\text{pred}} &= \|f_{\text{pred}}(\boldsymbol{z}_t) - \text{SG}(\boldsymbol{z}_{t+1})\|^2 \\ &= \|p(\boldsymbol{c}_t) - \text{SG}(\boldsymbol{z}_{t+1})\|^2 \quad . \end{aligned} \tag{4.5}$$

Context-predictive objective. Alternatively, the recurrent representation of future inputs can be used as the target, i.e., $f_{\text{tgt}}(\boldsymbol{z}_{t+1}) = f_{\text{rec}}(\boldsymbol{z}_{t+1})$. This leads to the context-predictive learning objective function:

$$\begin{aligned} \mathcal{L}_{\text{pred ctx.}} &= \|f_{\text{pred}}(\boldsymbol{z}_t) - \text{SG}(f_{\text{rec}}(\boldsymbol{z}_{t+1}))\|^2 \\ &= \|p(\boldsymbol{c}_t) - \text{SG}(\boldsymbol{c}_{t+1})\|^2 \quad . \end{aligned} \tag{4.6}$$

Invariance learning objective. The invariance learning objective is obtained by setting the target function $f_{\text{tgt}}(\boldsymbol{z}_{t+1}) = f_{\text{pred}}(\boldsymbol{z}_{t+1})$. These objectives typically need additional regularization losses in order to avoid collapsing to trivial representations [28]. Therefore, the objective function in Eq. 4.3 now yields the invariance objective:

$$\begin{aligned}\mathcal{L}_{\text{inv}} &= \|f_{\text{pred}}(\boldsymbol{z}_t) - SG(f_{\text{pred}}(\boldsymbol{z}_{t+1}))\|^2 + \mathcal{L}_{\text{reg}} \\ &= \|p(\boldsymbol{c}_t) - SG(p(\boldsymbol{c}_{t+1}))\|^2 + \mathcal{L}_{\text{reg}} \quad .\end{aligned} \tag{4.7}$$

Regularization loss. In all our experiments, we define $\mathcal{L}_{\text{reg}}$ similar to [103], as the sum of a push and decorrelation loss:

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{push}} + \mathcal{L}_{\text{decorr}} \quad , \tag{4.8}$$

applied on $p(\boldsymbol{c}_t) \in \mathbb{R}^M$. The push loss encourages the representations to be spread out in the latent space:

$$\mathcal{L}_{\text{push}} = -\frac{1}{M}\sum_{i=1}^{M} \log \sigma_i^2 \quad , \tag{4.9}$$

where σ_i^2 is the overall variance of the activity of neuron i in $p(\boldsymbol{c}_t)$. The decorrelation loss ensures that not all the dimensions of the latent space are used to encode the same information:

$$\mathcal{L}_{\text{decorr}} = \sum_{i=1}^{M}\sum_{j\neq i}^{M} C_{ij}^2 \quad , \tag{4.10}$$

where C_{ij} is the correlation coefficient between the activities of neurons i and j.

4.4.1 Learning from synthetic videos

Sprite video dataset generation. We generated the moving-sprites video dataset from a set of discretized latent factors, by creating 20,000 sequences of 32 frames of 64x64 pixel images containing spaceship sprites rendered on a black background. The latent factors included the sprite identity S, location $\vec{x} = (x, y)$, velocity $\vec{v} = (v_x, v_y)$, planar orientation θ, and the corresponding rotation velocity ω. Additionally, we scaled the sprite images throughout the videos with different rates of change to represent motion in z-direction with velocity v_z. The scaling factor was limited to $[0.7, 2]$ to ensure the sprite is perceptible but not larger than a quarter of the frame. The initial scale and its rate of change were sampled accordingly from uniform distributions $z \sim \mathcal{U}(-0.7, 2)$ and $v_z \sim \mathcal{U}(0.125, 0.125)$. Similarly, we sampled the initial location from a uniform distribution: $x, y \sim \mathcal{U}(8, 56)$, where 8 is the padding around the image, ensuring that the sprite remained fully visible within the frame. We sampled the initial orientation from a uniform distribution $\theta \sim \mathcal{U}(0, 2\pi)$, the initial translation velocity from $v_x, v_y \sim \mathcal{U}(-8, 8)$, and rotation velocity from $\omega \sim \mathcal{U}(-\frac{\pi}{6}, \frac{\pi}{6})$. Additionally, the sprite bounced off the walls of the frame by reversing the velocity component when it reached the edge of the frame. Throughout each video, the sprite identity remained constant, while the other factors changed over time.

Network architecture and training. To obtain feedforward representations of the video frames, we used a CNN with three convolutional layers, each with 32 filters of size 5×5 and stride and padding of size 2, followed by a batch normalization layer and a rectified linear unit (ReLU) activation function. The output of the third convolutional layer was flattened to obtain the 1D feedforward representation $\boldsymbol{z}_t$. This representation was then fed to the recurrent encoder with a single layer of 512 long short-term memory (LSTM) units to obtain recurrent representation $\boldsymbol{c}_t$. Finally, the recurrent representations passed through a feedforward predictor network with a single hidden layer of 512 units and the ReLU activation function, followed by a linear output layer. The network was trained using the Adam optimizer with learning rate 3×10^{-4}, weight decay 10^{-5}, and batch size 128 for 500 epochs using the aforementioned objective functions on 90% of the videos. In the case of the predictive objective variants (Eqs. (4.5) & (4.6)), the predictor network had a learning rate of 0.003, i.e., a 10 times faster weight update rate than the encoders [25].

Supervised and random controls. For the supervised and random baseline, the predictor network was replaced with a fully connected linear readout network with units corresponding to different tasks and categories. The resulting network was then trained jointly for all the tasks with supervised learning to decode the latent factors of the dataset. In the case of the random control, the feedforward and recurrent encoder weights were frozen after random initialization and only the readout weights were updated.

Evaluation. The trained network was evaluated on the remaining 10% of the videos to assess the quality of the learned representations. This was done by pretraining, then freezing the network, and then training linear readouts on the standardized recurrent representations to predict the latent factors. The linear readouts were trained using the Adam optimizer with a learning rate of 10^{-3}, weight decay of 10^{-5}, and a batch size of 128 for 100 epochs. Additionally, the network was evaluated by simulating rollouts of the sprite trajectories. For this purpose, the first 8 frames of the video were fed to the networ, and the predicted feedforward representations, i.e., output of the predictor network, were fed to the recurrent encoder autoregressively for the next 12 frames to generate simulated recurrent representations of the future frames. The predicted recurrent representations were then passed through the previously trained linear readouts to estimate latent factors of the future frames. The estimated latent values were then used to generate the predicted sprite trajectories (Fig. 4.3a). The predicted sprite trajectories were compared to the ground truth trajectories to compute the mean squared error of position, velocity, and orientation through time (Fig. 4.3).

4.4.2 Learning from sequences of handwritten digits

Digit triplets dataset generation. The digit triplets dataset was generated by creating 10,000 64-frame sequences of 28x28 pixel images of handwritten digits, sampled from the MNIST dataset. We clustered the digits into four groups: $\{1, 2, 3\}$, $\{4, 5, 6\}$, $\{7, 8, 9\}$, and $\{0\}$. The permutations of the digits within each of the first three groups constituted the triplets, making a total of 18 triplet labels. Each sequence consisted of sequences of triplets from different digit clusters, separated by a zero digit. Specifically, zero digits signaled a change in the cluster group of triplets and appeared with a probability of 0.2 at the end of each triplet sequence (Fig. 4.4a). The latent factors in this case were the digit identity, the cluster group, and the triplet identity.

Repeat/reversal dataset generation. The repeat/reversal dataset was generated by creating 10,000 sequences of MNIST digits. Each sequence was constructed from 5 smaller sequences of length 10. Each length-10 sequence started with a zero, followed by digit prefixes X of length 4 where $X_i \sim \mathrm{U}(3, 9)$, $\forall i \in \{1, 2, 3, 4\}$, a cue digit (1 for repeat and 2 for reverse), and a 4-digit long suffix which was either X or its reverse $\bar{X}$ based on the cue (Fig. 4.5a).

Network architecture and training. To obtain feedforward representations of the MNIST images in the triplets dataset, we used a multi-layer perceptron (MLP) with two hidden layers of 64 units, each followed by batch normalization and ReLU activation functions. The resulting feedforward representations were then fed into a single-layer recurrent neural network (RNN) with 256 units to generate the recurrent representations. The recurrent representations were further passed through a feedforward predictor network with a single hidden layer of 256 units and ReLU activation function, followed by a linear output layer. Given the complexity of the repeat/reversal dataset, we used a network with the same architecture as the digit triplets dataset but with double the number of units in every layer. Both networks were trained using the Adam optimizer with learning rate 3×10^{-4}, weight decay 10^{-3}, and batch size 64 for 200 epochs using the different objective functions on 90% of the sequences.

Supervised and random controls. To obtain the supervised and random baseline, the predictor network was replaced with a fully connected linear readout network with units corresponding to different tasks in each of the datasets. For the digit triplets dataset, the tasks included identifying the digit, the cluster, and the triplet. The triplet identity was decoded only after seeing its second digit. In the case of the repeat/reversal dataset, the tasks include identifying each digit in the sequence and the digit suffixes at the cue point. The network was then trained jointly using supervision as done for the sprite video dataset with or without freezing the encoders to obtain the random and supervised controls respectively.

Evaluation. The trained network was evaluated on the remaining 10% of the sequences to assess the quality of the learned representations. This was done by training linear readouts on the standardized feedforward and recurrent representations in both the digit triplets and repeat/reversal datasets after pretraining and freezing the network, to predict the latent factors (similar to random control). All linear readouts were trained using the Adam optimizer with a learning rate of 10^{-3}, weight decay of 10^{-5}, and a batch size of 128 for 50 epochs. In the case of the repeat/reversal dataset, we additionally calculated the cosine similarity between recurrent representations of four kinds of sequence pairs consisting of (i) two sequences with the same digit prefix and repeat cue (resulting in identical suffixes), (ii) two sequences with the same digit prefix, one followed by a repeat cue and the other by a reversal cue (reversed suffixes), (iii) two sequences with digit prefixes reversed with respect to each other, and both followed by different cues (identical suffixes), and (iv) two sequences with digit prefixes reversed with respect to each other, and both followed by the same cue (reversed suffixes).

4.4.3 Learning from natural speech data

LibriSpeech dataset. Similar to [22], we used a 100-hour subset of the LibriSpeech dataset [176] for training and evaluating the network. LibriSpeech contains English audio book files from 251 speakers, with 41 force-aligned phoneme labels. The audio files were sampled at 16 kHz and there is a phoneme label for roughly every 10 ms of audio.

Network architecture and training. To obtain feedforward representations of the audio frames, we used the same 1D CNN architecture as in [22], with 512 output units. The feedforward representations were then fed into a single-layer LSTM with 256 units to generate the recurrent representations. The recurrent representations were then passed through a feedforward predictor network with a single hidden layer of 512 units and ReLU activation function, followed by a linear output layer. The network was trained using the Adam optimizer with learning rate 3×10^{-4}, weight decay 10^{-5}, and batch size 256 for 500 epochs using the different objective functions, with the exception that the target representation was the 8th frame ahead in the audio sequence instead of the immediate next representation. This value was experimentally determined to be the optimal prediction horizon for the predictive objective.

Supervised and random controls. To obtain the supervised and random baseline, the predictor network was replaced with a fully connected linear readout network. The linear readout contained 251 units for speaker classification and 41 units for phoneme classification. The resulting network was then trained separately for each classification task using the cross-entropy loss. For the random control, the feedforward and recurrent encoder weights were frozen, and only the readout weights were updated. Since phoneme labels are force-aligned and no start/end point is indicated for the audio part corresponding to each phoneme, a batch size of 1 was used for training readouts of the phoneme classification task.

Evaluation. The trained network was evaluated on the test subset of the dataset by training linear readouts on the standardized recurrent representations after pretraining and freezing the network to predict the phoneme and speaker labels. The linear readouts were trained using the Adam optimizer with learning rate 10^{-3}, weight decay 10^{-5}, and batch size 64 (for speaker) for 50 epochs.

Layer-wise pretraining with the predictive learning objective

To assess the effectiveness of layer-wise pretraining with the predictive learning objective, we used the sprite video dataset. The network was constructed of 4 layers, each of which included feedforward and recurrent encoders, and a predictor network (Fig. 4.7a). The feedforward encoder in each area was a single-layer 2D CNN with 32 filters of size 5×5 and stride and padding of size 2, followed by a batch normalization layer and a ReLU activation function. The output of each feedforward encoder was flattened to obtain a 1D feedforward representation, each of which was fed into a different single-layer LSTM with 256 units to generate the recurrent representations. The recurrent representations were then passed through feedforward predictor networks with single hidden layers of 512 units and ReLU activation function, followed by linear output layers. The network was trained using the Adam optimizer with learning rate 3×10^{-4}, weight decay 10^{-5}, and batch size 128 for 500 epochs using the predictive learning objective in Eq. 4.5 on 90% of the videos. Importantly, the different layers were gradient isolated and independently optimized the predictive learning objective in each layer, without backpropagating the gradients across layers.

Evaluation. To evaluate the representations learned by this network, we trained linear readouts on the standardized recurrent representations of each layer after pretraining and freezing the network to decode the latent factors. The linear readouts were trained using the Adam optimizer with learning rate 10^{-3}, weight decay 10^{-5}, and batch size 128 for 100 epochs.

Chapter V

Conclusion

In this thesis, we have developed a biologically plausible computational model for predictive representation learning that extracts learning signals from the passage of time and learns abstractions of sensory data. In this chapter, I summarize our main findings, discuss their implications, and suggest directions for future work.

5.1 Summary of findings

In Chapter II, we developed a normative synaptic plasticity rule combining the principles of predictive learning with classical Hebbian plasticity models like the BCM rule [11]. We showed that this rule, termed Latent Predictive Learning (LPL), can learn invariant object representations in hierarchical sensory networks by leveraging temporal persistence. Notably, LPL learned without the need for backpropagation and remained effective in spiking neural network models. LPL also reproduced several known plasticity phenomena at both population- and single-synapse levels and made new experimental predictions regarding synaptic metaplasticity [121]. The LPL rule built on one class of non-contrastive representation-space SSL methods, specifically variance-regularized algorithms [28, 90], which were originally designed to obtain image representations that are invariant to various augmentations. Our work extended these methods to the temporal domain and adapted them to a predictive and backpropagation-free learning setup [118]. We also highlighted a surprising connection between the variance regularization strategy for preventing representation collapse and classical Hebbian plasticity.

In Chapter III, we analyzed the learning dynamics of asymmetric predictor-based SSL models. These methods rely on somewhat opaque architectural tricks in order to avoid representational collapse. Their learning dynamics have been the focus of several theoretical [151–155] and empirical studies [156–158]. We built on this previous work to develop a powerful yet intuitive eigenspace analysis of these learning dynamics. Our theoretical framework revealed that, surprisingly, the same variance-regularization mechanism that prevents representation collapse in LPL and similar models [28] is implicitly enforced by the asymmetry induced by the predictor and stop-gradient. The theory also showed that the learning dynamics are typically anisotropic, with larger eigenvalues learning faster, reminiscent of previous theoretical work on supervised learning [161]. We further designed isotropic loss functions (IsoLoss) that can counteract this anisotropy, providing faster initial learning and robustness to architectural changes.

Finally, in Chapter IV, we introduced RePL, a novel predictive learning framework that extends the asymmetric predictor-based SSL models to the temporal domain and includes a recurrent encoder module for learning dynamic properties of sequential data. Although Chapter III showed that such models implicitly share the same collapse-preventing mechanism as LPL, we found that they differ in their representation learning capabilities. Specifically, we found that RePL is capable of learning representations that include but go beyond temporal persistence. Through experiments on synthetic moving objects, handwritten digit sequences, and natural speech, we demonstrated that RePL consistently outperforms an invariance-based approach similar to LPL, in capturing dynamic properties of sequential data. The learned representations also enabled accurate simulation through autoregressive rollouts and could potentially be leveraged for planning and control tasks. A key insight from this work is that the prediction target is crucial for learning dynamic factors - specifically, we found that non-contextual prediction targets are essential for learning useful representations of properties like velocities of moving objects or phoneme sequences in speech. Finally, we showed that RePL remains effective even in a layer-wise training setup without end-to-end backpropagation, taking an initial step towards biological plausibility.

5.2 The path forward

We set out to develop and examine a computational model of predictive representation learning that is both biologically plausible and functionally effective for training DNNs. We leveraged the body of engineering knowledge in machine learning to build predictive SSL models that implement the predictive learning principle thought to be a foundational aspect of biological learning. These models remained effective even upon imposing some set of biological constraints, such as the absence of backpropagation or the presence of spiking dynamics. They also seem to be consistent with some known experimental findings in neuroscience and are beginning to generate new experimental predictions.

However, it is clear that this work is only a preliminary step towards a full understanding of predictive learning in the brain. The models we developed are still quite abstract and far from the detailed circuit-level models that would be required to make more precise experimental predictions. In the following, I outline some key directions for future work that could help bridge this gap.

5.2.1 Predictive SSL in brains

The map is not the territory, and neural networks, no matter how detailed, are not brains. The only real test of whether predictive SSL is a good model of biological learning is to compare the predictions generated by the model to the actual behavior of biological systems. This requires more work on both the experimental and theoretical fronts across levels of analysis [1].

On the experimental side, it would be interesting to see if the learning dynamics of the LPL rule can be observed in real neurons. We have presented one experimental prediction in Chapter II: the STDP window should be modulated by the history of the postsynaptic neuron. This could be tested *in vitro* by measuring the plasticity of synapses onto neurons that have been artificially depolarized or hyperpolarized in the preceding time window.

Beyond this specific prediction, the distinction between feedforward and recurrent streams of information [182] in the RePL model is suggestive of layer-specific roles in predictive learning. However, more modeling work is needed to develop a detailed cortical-layer-specific implementation of predictive learning mechanisms incorporating detailed biophysical constraints such as Dale's law. Longitudinal studies of neural activity in different layers of the cortex during exposure to stimuli with precisely controlled temporal properties [190] could both inform and validate such detailed models. Disrupting development with a visual diet of temporally scrambled stimuli [101] to test the necessity of temporal contiguity for the development of representations in different layers could do the same.

Finally, the interaction between predictive learning and other known plasticity mechanisms deserves exploration. For instance, how does predictive SSL interact with reward-based learning? Experimental paradigms that dissociate these different learning signals could help determine their relative contributions to representational learning in different brain regions and under different behavioral conditions.

5.2.2 Predictive SSL in machines

While our work has demonstrated the effectiveness of temporally predictive SSL for representation learning, there remain open questions and opportunities for further development in machine learning applications.

The scalability of these methods to larger and more complex datasets remains to be fully explored. Although our experiments with natural speech and moving object datasets showed promising results, extending these approaches to natural videos or multimodal inputs may require additional architectural innovations or training strategies. For example, the use of hierarchical or multi-scale models could help capture the complex spatiotemporal structure of natural videos, while incorporating additional modalities such as audio or text could enable more generalized representations of multimodal data.

The temporal aspect of our models suggests natural applications in domains requiring temporal reasoning and planning. The ability of RePL to capture dynamic properties and enable simulation through autoregressive rollouts could be particularly valuable for planning in reinforcement learning tasks. Exploring how these representations can be leveraged for planning and control tasks represents a promising avenue for future work.

5.3 Concluding remarks

In accordance with Feynman's dictum, "What I cannot create, I do not understand," that opens this thesis, we have attempted to understand and test a particular theory about the brain by creating computational models implementing that theory. The models we have developed are far from the final word on predictive learning and only a starting point for further investigation. Future work will require close interplay between theoretical, experimental, and computational approaches, and will likely involve a combination of techniques from machine learning, neuroscience, and cognitive science.

Appendix A

Supplements to Chapter II

A.1 Supplementary Figures and Tables

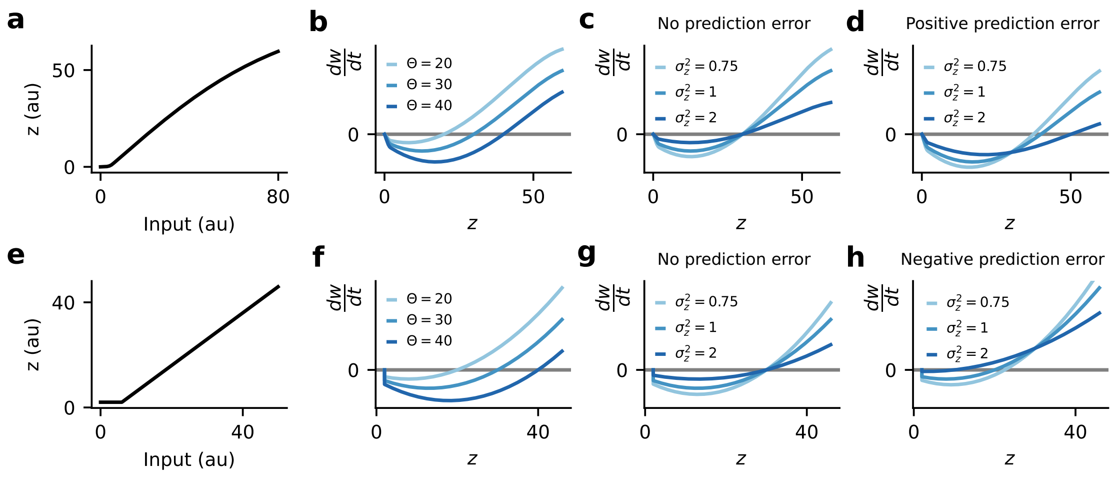


FIGURE A.1: **LPL extends BCM theory by adding a variance- and rate-of-change dependence.** **(a)** Example of a typical neuronal input-output function with postsynaptic activity z. **(b)** Weight change induced by the LPL rule for co-varying input and the postsynaptic activity z for different values of the plasticity threshold Θ, with $\sigma_z^2 = 1$ and $\frac{dz}{dt} = 0$. The functional shift of the threshold is reminiscent of the BCM rule. **(c)** Same as (b) but for different values of the variance of the postsynaptic activity with zero prediction error $\frac{dz}{dt} = 0$ and fixed mean activity $\bar{z} = 30$. **(d)** Same as (c) but with a positive prediction error $\frac{dz}{dt} = +10$. **(e)** Same as (a), but for a ReLU activation function with positive threshold. **(f–g)** Same as above but for ReLU. **(h)** Same as in (d) but for ReLU and a negative prediction error $\frac{dz}{dt} = -10$.

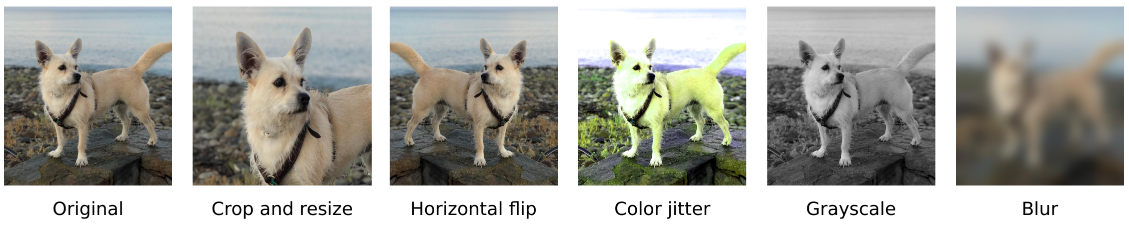


FIGURE A.2: **Image augmentation model.** Illustration of the image transformations used to generate natural image sequences as suggested by Chen *et al* [23].

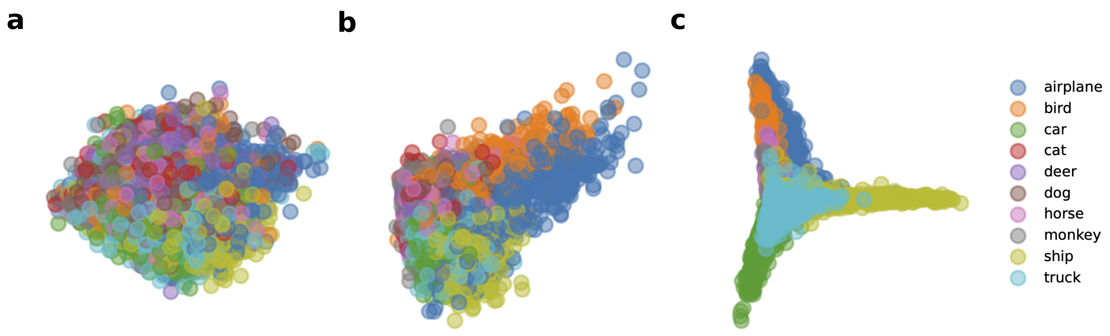


FIGURE A.3: **Disentangling of object representations in the DNN.** **(a)** Data distribution of the STL-10 validation set along the first two principal components in pixel-space. Data corresponding to different object classes are highly entangled. **(b)** Same as (a) but along the principal components of representations in Layer 3 of the DNN after learning with LPL. Object classes are somewhat disentangled. **(c)** Same as (a) but along the principal components of representations in Layer 8 of the DNN. Object classes are highly disentangled.

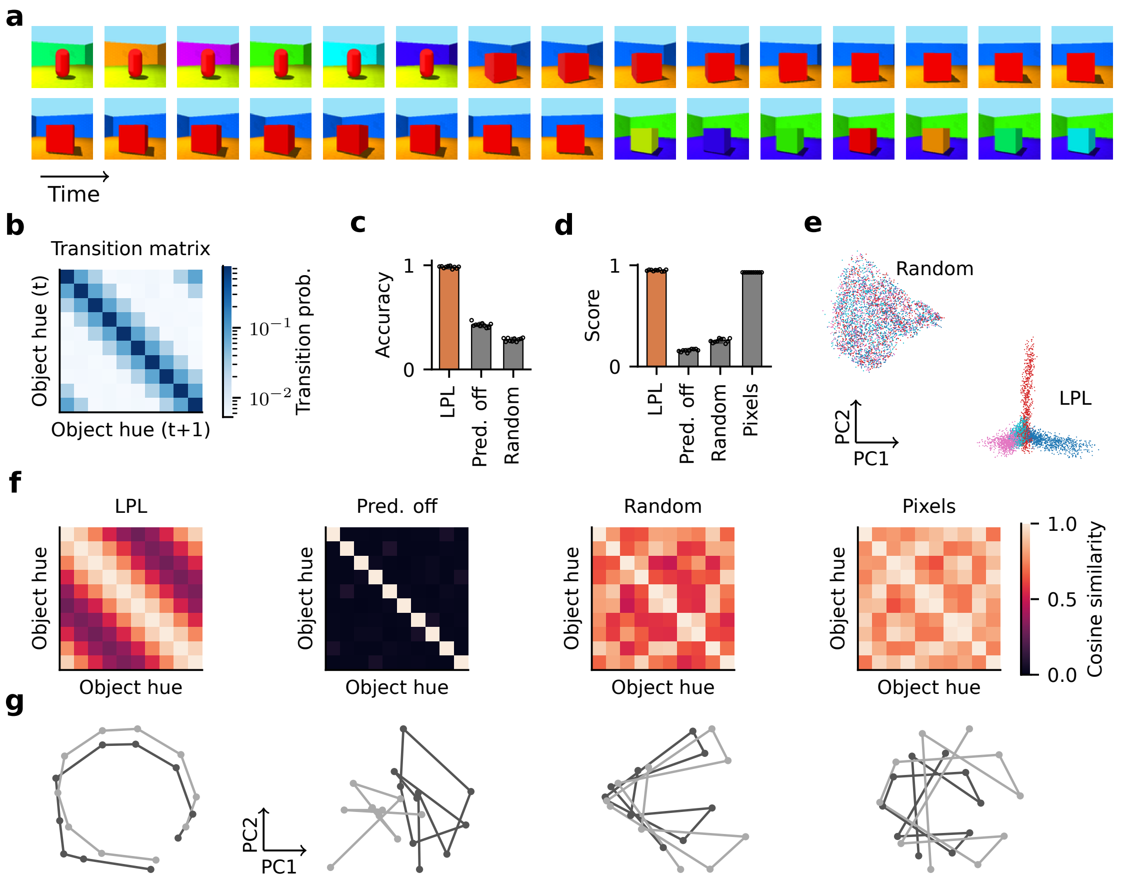


FIGURE A.4: **LPL finds latent manifold structure of simulated video data.** **(a)** Input frames from the procedurally generated video using the 3D Shapes dataset [128]. **(b)** The empirically measured transition matrix of object hue with latent structure (see Extended Data Fig. A.5 for the complete set of transition matrices). **(c)** Object classification accuracy of a linear classifier trained on network outputs of a network with LPL, without the predictive term (Pred. off), and the randomly initialized network (Random). Values represent averages from cross validation ($n = 10$ folds). Error bars indicate ±SEM. The accuracy is close to 100% for LPL, but lower at initialization or when trained without the predictive term. **(d)** Disentanglement scores computed according to the metric proposed by Kim and Mnih [128] for the final-layer representations of the three networks in (c) compared to the input pixels (Pixels). LPL yields close to maximum scores (95.0% ± 0.8%), higher than a randomly initialized network or after training without the predictive term. However, evaluating the metric on the pixels directly also yields high scores (93.0% ± 0.0%), albeit still slightly lower than LPL. The high scores in pixel space can partially be explained by the high input dimension and the small number of classes in the dataset. Importantly, the metric is insensitive to the manifold topology (see below). Different data points correspond to averages over $n = 10$ independent evaluations of the metric, with error bars ±SEM. **(e)** Projections of the representations onto the first two principal components before (Random) and after training (LPL). Each point corresponds to one input image, and the color represents the object type. The object class is entangled at initialization and disentangled after learning. **(f)** Averaged RSM computed from representations of different object colors in (d). LPL's RSM closely resembles the transition structure shown in (b). Without the predictive term, the RSM becomes diagonal, while the random network's RSM does not have this structure and roughly follows the input pixel similarity structure. **(g)** Network output projected onto the first two principal components for changing hue sequentially while keeping all other factors fixed. The two lines correspond to two different object sizes. The trajectories are disentangled for LPL and preserve the topology of the data manifold (cf. b), whereas this is not the case when the predictive term is off, at initialization (random), or at the input (pixels).

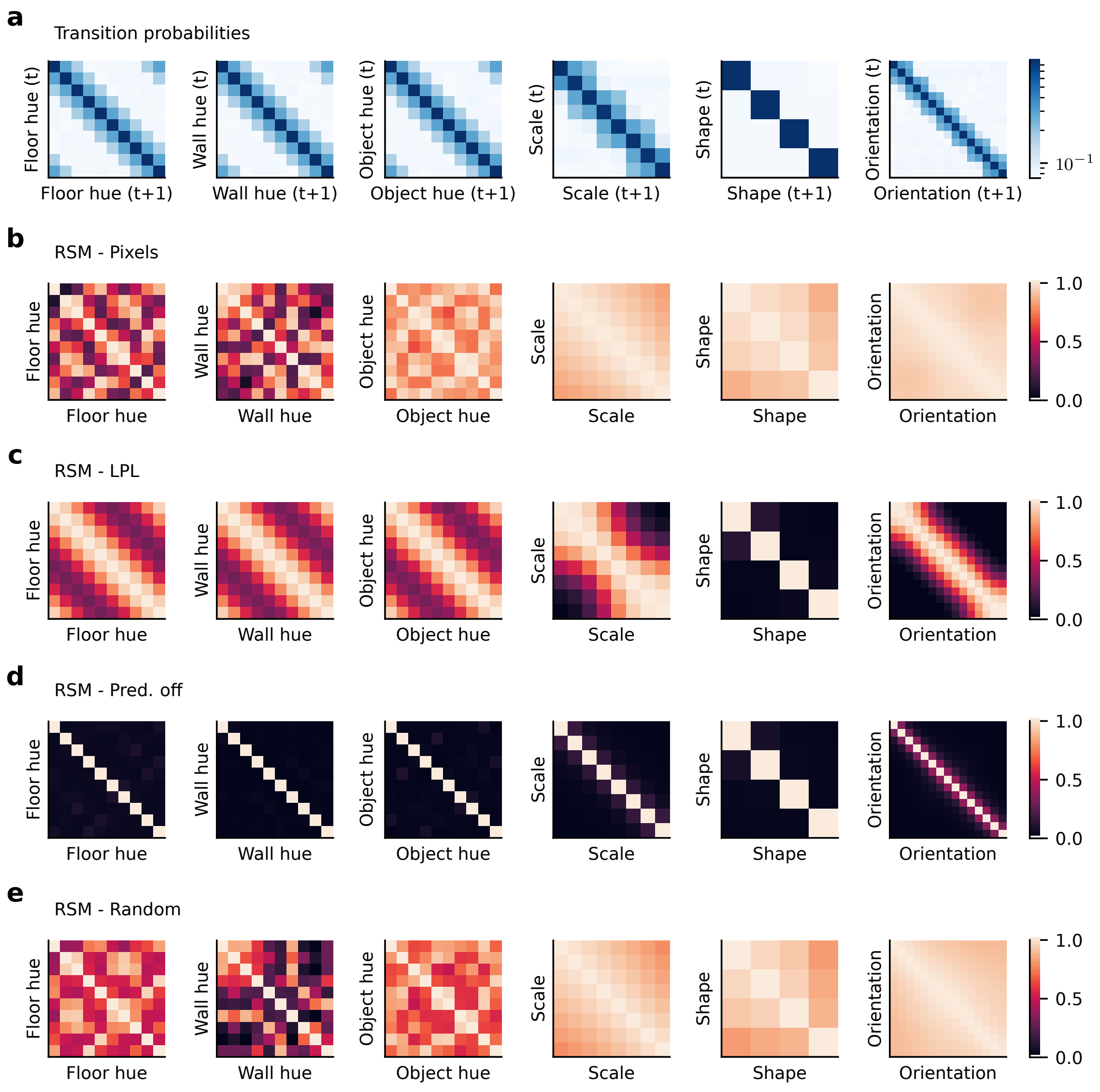


FIGURE A.5: **Transition matrices and RSMs for all latent factors of the 3D shapes dataset.** **(a)** Transition probabilities estimated from the generated video. The high values on the diagonal reflect the fact that within a 17-frame clip, only one factor changes while the others remain fixed. The off-diagonal values reflect the transition probabilities when a specific factor is changing. For instance, within a clip cycling through all the object hues, the color may only change to the next or previous assignments in the color map with a smaller probability for a two-step transition. The hue mapping was randomly chosen with respect to the original dataset to ensure an entangled topology at the input (cf. Fig. A.4g). The orientation and scale factors are not allowed to transition from the smallest to the largest values, and vice versa. Furthermore, the direction of change for any factor is fixed within a given clip, but may reverse for orientation and scale at the extreme values (cf. Fig. A.4a). **(b)** Same as Fig. A.4f, but for all factors at the pixel level. RSM values represent average cosine similarity between the pixels of images differing only in one factor with all other factors fixed. Some similarity structure exists along the scale and orientation factors only. **(c)** Same as (b) but for the final-layer representations learned by LPL. The RSM closely resembles the transition probability structure that characterizes the temporal properties of the video sequence. **(d)** Same as (c) but for learning without the predictive term. The RSM is diagonal, which shows that the network represents different factors in almost orthogonal directions. **(e)** Same as (b), but at random initialization before training. The RSM for all factors is reflective of the pixel RSM.

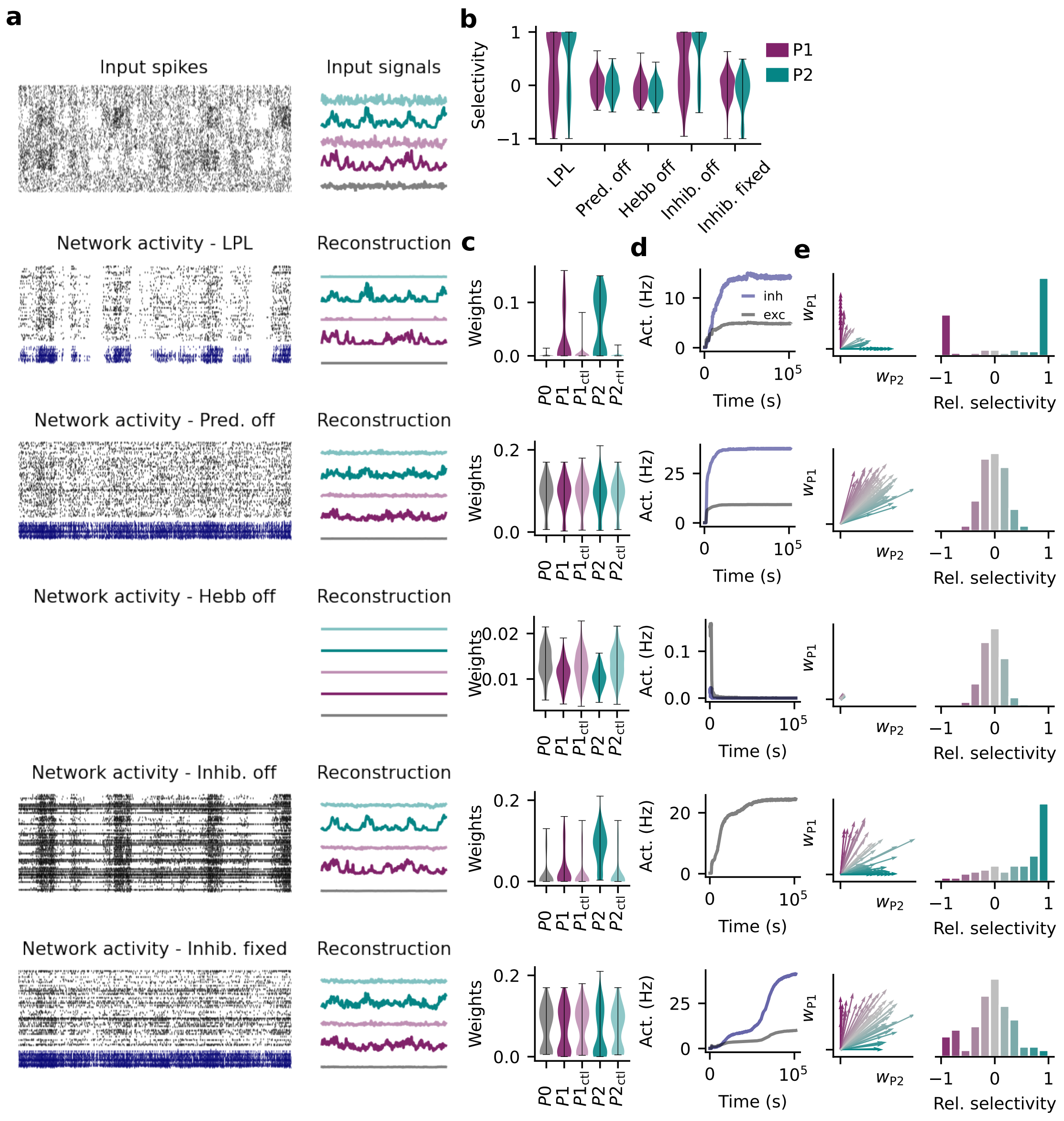


FIGURE A.6: **Same as Figure 2.5 but with detailed controls.** **(a)** Snapshot of spiking activity (left) and underlying firing rate signals or their reconstructions (right) over 100 ms for the input and network populations. **(b)** Same as Fig. 2.5d showing signal selectivity learned with the different variations of spiking LPL given in (a). **(c)** Average synaptic connection strength grouped by input population for the different configurations in (a). LPL with plastic inhibition results in higher weights on the slowly varying signals relative to the shuffled controls, but not when the predictive or Hebbian term are disabled. Without inhibition or without inhibitory plasticity, connections from all populations are strong with a small preference for $P2$. **(d)** Average firing rates over 100 s bins throughout training for the configurations in (a). Firing rates saturate with the inhibitory neurons settling at a higher firing rate when learning with spiking LPL with inhibition, even when the predictive term is disabled or the inhibition is not plastic. Activity collapses without the Hebbian term, whereas firing rates diverge without inhibition. **(e)** Averaged weight vectors from populations $P1$ and $P2$ onto each excitatory neuron (left) and distribution of the excitatory neurons' relative selectivity between the two populations (right). Different neurons are exclusively selective to either $P1$ pr $P2$ under spiking LPL with inhibitory plasticity. Without the predictive term, or the Hebbian term, few if any neurons are selective to one population over the other. Moreover, weights collapse to small values without the Hebbian term. When inhibition is removed altogether, a few neurons become exclusively selective to $P2$, but the weight vectors are not well-decorrelated. Without inhibitory plasticity, a few weight vectors are well-decorrelated, but most neurons are not preferentially selective to either signal.

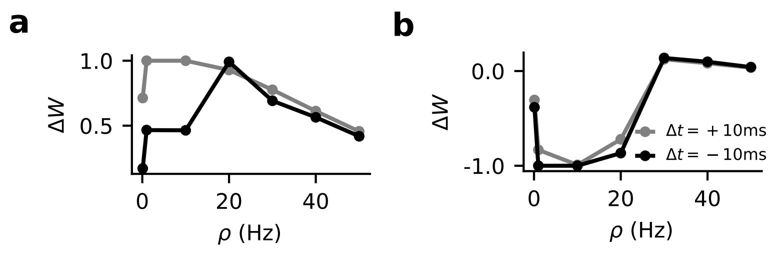


FIGURE A.7: **Learning threshold determines the sign of plasticity. (a)** Weight changes as a function of repetition frequency ρ for positive and negative relative spike timings ($\Delta t = \pm 10\,\mathrm{ms}$) with $\sigma^2(t=0) = 10^{-5}$ and $\bar{S}_i(t=0) = 0$. **(b)** Same as (a) but for $\bar{S}_i(t=0) = 50$.

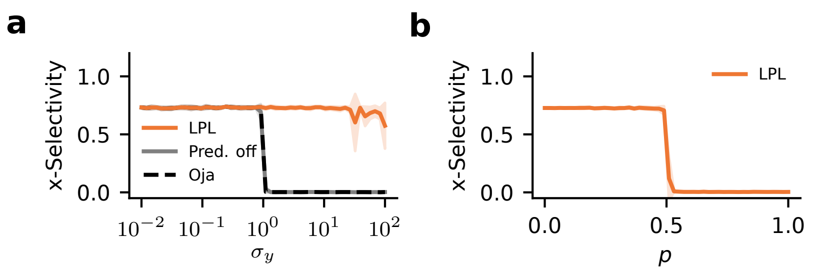


FIGURE A.8: **LPL is robust to noise. (a)** Same as Figure A.1b but for high rates of noisy transitions between clusters in the training data sequence with $p = 0.2$ (Methods). A neuron learning with LPL still consistently becomes selective to cluster identity even with noisy transitions. Data are averages ±SEM ($n = 10$ random seeds). **(b)** Cluster selectivity as a function of the probability of noisy cross-cluster transitions in the data sequence with $\sigma_y = 1$. LPL drives selectivity to cluster identity only below $p = 0.5$, i.e, only as long as cluster identity remains the slow feature. Values are averages ±SEM ($n = 10$ random seeds).

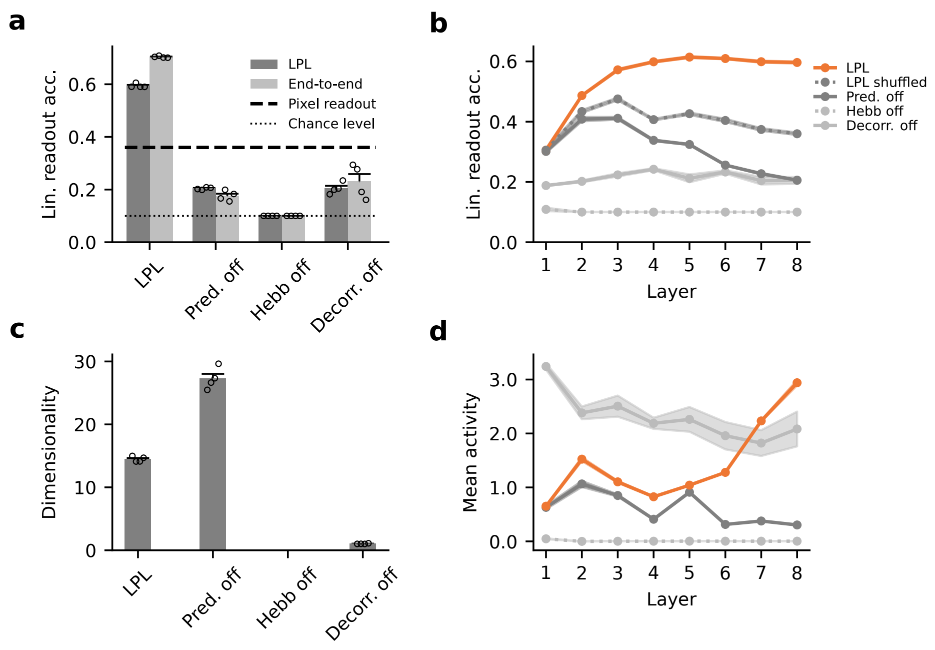


FIGURE A.9: **Same as Figure 2.3 but for the CIFAR-10 dataset.** **(a)** Linear readout accuracy of object categories decoded from representations at the network output after training it on natural image data for different learning rules in layer-local (dark) as well as the end-to-end configuration (light). **(b)** Linear readout accuracy of the internal representations at different layers of the DNN after layer-local training. **(c)** Dimensionality of the internal representations for the different learning rule configurations shown in (b). **(d)** Mean neuronal activity at different layers of the DNN after training for the different learning rule variants shown in (b).

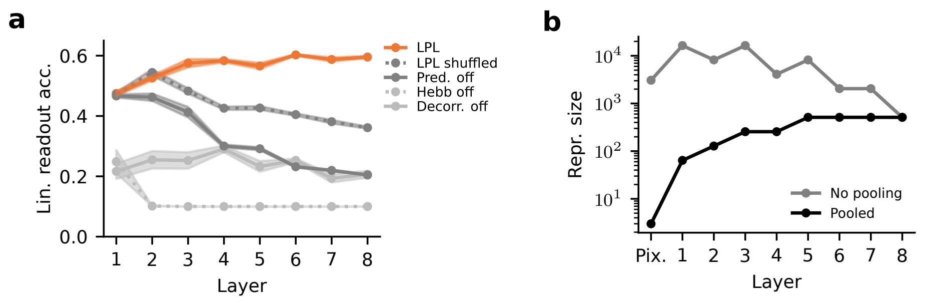


FIGURE A.10: **Evaluating readout accuracy without pooling.** **(a)** Same as Extended Data Fig. A.9b above, but with linear readout accuracies evaluated from the full feature map at each layer instead of after pooling. Results are qualitatively the same as before, but the starting accuracy at early layers is substantially higher. **(b)** Effective representation size that would be the input to the linear classifier at each layer with or without pooling. Without pooling, the number of features at early layers is very large, and may explain the higher early-layer accuracies in (b).

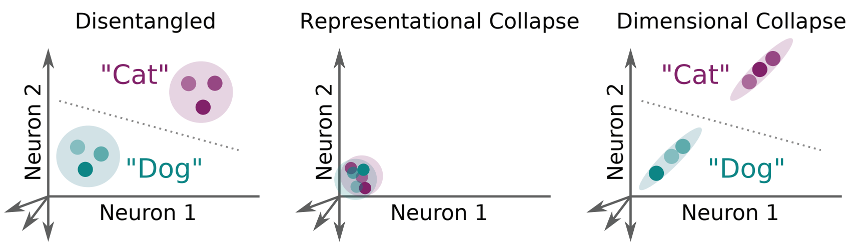


FIGURE A.11: **Illustration of collapse modes typifying poorly disentangled features** Effectively disentangled representations (left) separate categories well with different representational directions encoding different relevant features. Purely predictive learning without counteracting Hebbian plasticity leads to collapsed representations (center), typically to zero activity levels. Dimensional collapse (right) is characterized by highly correlated activity across all neurons, indicating that only one relevant feature is encoded by all neurons, which is unlikely to be conducive to hierarchical feature extraction for non-trivial tasks such as object recognition.

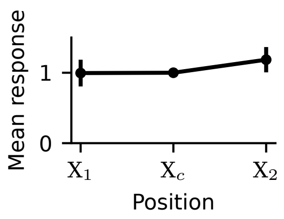


FIGURE A.12: **Learned network representations are invariant to object position on the canvas.** Activity of neurons in the pretrained DNN's output layer in response to images at three positions on the canvas, normalized by each neuron's response to the center position, and averaged over n =neurons and over images.

	STL-10		CIFAR-10	
	Layer-local (%)	End-to-end (%)	Layer-local (%)	End-to-end (%)
LPL	63.2±0.3	72.5±0.1	59.4±0.4	70.4±0.2
Neg. samples	77.0±0.2	81.0±0.3	67.4±0.3	76.5±0.1
Supervised	70.8±0.3	77.8.5±0.3	81.7±0.2	87.1±0.2
Pixel-space decoding	31.6		35.9	

Table A.1: Extended version of Table 2.1 showing linear classification accuracy on the STL-10 and CIFAR-10 datasets for LPL and different baseline models (Methods). Error values correspond to SEM over four simulations with different random seeds.

	STL-10	CIFAR-10
VGG-11	63.2±0.3	59.4±0.4
VGG-11 without bias terms	58.7	54.6
VGG-11 with stale mean and variance estimates	59.3±0.6	47.7±0.6
VGG-11 without GAP	29.2±1.8	47.7±0.3
VGG-11 without GAP (down-sampled inputs)	45.3±0.5	-

Table A.2: Linear classification accuracy on the STL-10 and CIFAR-10 datasets for layer-locally trained LPL with the base VGG-11 architecture, and several modified versions. VGG-11 without bias terms refers to the standard architecture including GAP but without any bias terms in the convolutional layers. Stale mean and variance estimates denote calculating the representational mean and variance from the previous batch. VGG-11 without GAP refers to computing and optimizing the LPL objective on the unpooled feature maps. Downsampled inputs correspond to the architecture without GAP, but now using STL-10 images subsampled to a lower resolution of 32×32. Reported error values correspond to SEM over four simulations with different random seeds.

	Single neuron (2D dataset)	Single neuron (digit dataset)	Network simulations
λ_1	1	1	1
λ_2	-	-	10
Optimizer	SGD	SGD	Adam[a]
Learning rate η	$\min(10^{-2}, 10^{-2}/\sigma_y)$	10^{-2}	10^{-3} [b]
Weight decay η_w	0.15	0.15	1.5×10^{-6}
Batch size B	200	200	1024
Training steps	$\max(10000, 100\sigma_y)$	1000	∼78000[c]

[a] With default parameters from [191].
[b] Reduced to zero during training using a cosine learning rate schedule.
[c] ∼35000 for CIFAR-10.

TABLE A.3: Hyperparameter values for DNN simulations.

Parameter	Value	Parameter	Value
τ^{mem}	20 ms	U^{exc}	0 mV
τ^{ampa}	5 ms	U^{inh}	-80 mV
τ^{gaba}	10 ms	U^{leak}	-70 mV
τ^{nmda}	100 ms		
τ^{thr}	5 ms		

TABLE A.4: Table summarizing the neuronal parameters.

Source	Destination	Connection probability	Initial weight value
input	exc	0.1	0.15
exc	inh	local (see Methods)	0.4
inh	exc	0.5	0.1
inh	inh	0.1	0.4

TABLE A.5: Summary of SNN connectivity parameters.

Supplementary Notes

A.2 Equivalence of the objective function and learning rule formulations

Here, we show that the objective functions defined in Eqs. (2.4), (2.5), and (2.6) indeed result in the LPL rule (Eq. (2.7)).

Predictive component. We recall that the predictive objective $\mathcal{L}_{\text{pred}}$ is the mean squared difference between neuronal activity in consecutive time steps.

$$\begin{aligned}\mathcal{L}_{\text{pred}} &= \frac{1}{2MB}\sum_{b=1}^{B}\|z^b - SG(z^b(t-\Delta t))\|^2 \\ &= \frac{1}{2MB}\sum_{b=1}^{B}\sum_{i=1}^{M}\left(z_i^b - SG(z_i^b(t-\Delta t))\right)^2\end{aligned}$$

Taking the derivative with respect to the network weights results in the following learning rule

$$\frac{\partial\mathcal{L}_{\text{pred}}}{\partial W_{ij}} = \frac{1}{MB}\sum_{b=1}^{B}\left(z_i^b - z_i^b(t-\Delta t)\right) f'(a_i^b)x_j^b \tag{A.1}$$

which does not require backpropagation through time due to the Stopgrad function.

Hebbian component. The Hebbian component minimizes the negative logarithm of the variance of neuronal activity:

$$\mathcal{L}_{\text{Hebb}} = \frac{1}{M}\sum_{i=1}^{M} -\log\left(\sigma_i^2\right)$$

where $\bar{z}_i = \text{SG}(\frac{1}{B}\sum_{b=1}^{B} z_i^b)$ and $\sigma_i^2 = \frac{1}{B-1}\sum_{b=1}^{B}\left(z_i^b - \bar{z}_i\right)^2$ are the mean and variance of the activity of the ith output neuron over the minibatch. The corresponding learning rule is obtained as the negative gradient of this loss function with respect to the weights W. The gradient itself is given by:

$$\frac{\partial\mathcal{L}_{\text{Hebb}}}{\partial W_{ij}} = -\frac{2}{M(B-1)\sigma_i^2}\sum_{b=1}^{B}(z_i^b - \bar{z}_i)f'(a_i^b)x_j^b \tag{A.2}$$

Note that the objective and the resulting gradient is essentially unchanged upto a scaling factor when we use running estimates of the variance with a time constant τ instead of batch estimates.

$$\begin{aligned}\mathcal{L}_{\text{Hebb}}(t) &= \frac{1}{M}\sum_{i=1}^{M} -\log\left(\sigma_i^2(t)\right)\\ &= \frac{1}{M}\sum_{i=1}^{M} -\log\left((1-\tau)(z_i(t)-\bar{z}_i(t))^2+\tau\text{SG}(\sigma_i^2(t-1))+\epsilon\right)\\ \frac{\partial\mathcal{L}_{\text{Hebb}}}{\partial W_{ij}}(t) &= -\frac{2(1-\tau)}{M\sigma_i^2(t)}(z_i^b(t)-\bar{z}_i(t))f'(a_i^b(t))x_j^b(t)\end{aligned}$$

Decorrelation component. Finally, the decorrelation objective is the decorrelation loss function as the sum of the squared off-diagonal terms of the covariance matrix between units.

$$\mathcal{L}_{\text{decorr}} = \frac{1}{(M^2-M)}\sum_{i=1}^{M}\sum_{k\neq i}\left(\frac{\sum_{b=1}^{B}\left(z_i^b(t)-\bar{z}_i(t)\right)\left(z_k^b(t)-\bar{z}_k(t)\right)}{B-1}\right)^2 ,$$

which gives the gradient:

$$\frac{\partial\mathcal{L}_{\text{decorr}}}{\partial W_{ij}} = \frac{4}{(B-1)(M^2-M)}\sum_{b=1}^{B} f'(a_i^b)x_j^b\sum_{k\neq i}\left(z_k^b(t)-\bar{z}_k(t)\right)C_{ik} , \tag{A.3}$$

where C_{ik} is the covariance between units i and k (Eq. (2.8)).

The full learning rule. The combined weight updates for a descent along the sum of the three gradients in Eqs. (A.1), (A.2), and (A.3) in a single-layer network finally yields the LPL rule including the decorrelation component:

$$\begin{aligned}\Delta W_{ij} &= -\eta\left(\frac{\partial\mathcal{L}_{\text{pred}}}{\partial W_{ij}}+\lambda_1\frac{\partial\mathcal{L}_{\text{Hebb}}}{\partial W_{ij}}+\lambda_2\frac{\partial\mathcal{L}_{\text{decorr}}}{\partial W_{ij}}\right) \qquad (A.4)\\ &= \frac{\eta}{MB}\sum_{b=1}^{B}\left(-(z_i^b-z_i^b(t-\Delta t))+\lambda_1\frac{\alpha}{\sigma_i^2}(z_i^b-\bar{z}_i)-\lambda_2\beta\sum_{k\neq i}(z_k^b-\bar{z}_k)C_{ik}\right)f'(a_i^b)x_j^b\end{aligned}$$

where $\alpha = \frac{2B}{B-1}$ and $\beta = \frac{4B}{(B-1)(M-1)}$ are the appropriate normalizing constants, and λ_1 and λ_2 are the loss coefficients. Including weight decay in the weight update finally yields the LPL rule for a network given in Eq. (2.7).

A.3 Relating the Hebbian component of LPL to Oja's rule

To see the relation of the Hebbian component of LPL with the classic Oja's rule [41], we consider the case of a single output neuron ($M = 1$), with no nonlinearity ($f'(a) = 1$), along with the assumption that the input is zero-centered ($\bar{x}_j = 0$). Consequently, $\bar{z} = \sum_j W_j \bar{x}_j = 0$ and $\sigma_z^2 = \left\langle (z - \bar{z})^2 \right\rangle = \langle z^2 \rangle$, which yields a very simple Hebbian learning rule for descending the gradient in Eq. (A.2):

$$\Delta W_j(t) = -\frac{\partial \mathcal{L}_{\text{Hebb}}(t)}{\partial W_{ij}} = \frac{z(t) x_j(t)}{\langle z^2 \rangle}$$

This update rule along with a weight decay (with coefficient η_w) yields a learning rule that, on average, is equivalent to Oja's rule up to a scaling factor, and in fact has exactly the same non-zero fixed point when $\eta_w = 1$, but with different convergence dynamics because of the multiplication by $1/\langle z^2 \rangle$.

$$\begin{aligned}
\Delta W_j(t) &= \frac{z(t) x_j(t)}{\langle z^2 \rangle} - \eta_w W_j \\
\langle \Delta W_j \rangle &= \frac{\langle z x_j \rangle}{\langle z^2 \rangle} - \eta_w W_j \\
&= \frac{1}{\langle z^2 \rangle} \left(\langle z x_j \rangle - \eta_w W_j \langle z^2 \rangle \right) \qquad \text{(A.5)}
\end{aligned}$$

Oja's rule is presented below for reference

$$\begin{aligned}
\Delta W_j^{\text{Oja}}(t) &= z(t) x_j(t) - W_j z^2 \\
\left\langle \Delta W_j^{\text{Oja}} \right\rangle &= \langle z x_j \rangle - W_j \langle z^2 \rangle \qquad \text{(A.6)}
\end{aligned}$$

A.4 Importance of the variance-dependent modulation of Hebbian learning

To analytically understand the importance of the variance-dependent scaling of the Hebbian term in the learning rule, we first looked at the synthetic two-dimensional learning task from Fig. 2.2a, and modeled the behaviour of the LPL loss functions under a particular distribution of the representations. Specifically, we considered the case where the two input clusters map to a mixture of two normal distributions in *representation space* with each Gaussian component corresponding to the representations of one input cluster. Each of the two Gaussians are assumed to have a standard deviation of r, with their means symmetrically located on either side of zero with a distance of $D = 4r$ between their centers. We used this setting to investigate how the predictive and Hebbian loss terms of LPL behave under different values of the representational variance by co-varying r and D.

The predictive loss in this case is proportional to the expected squared difference

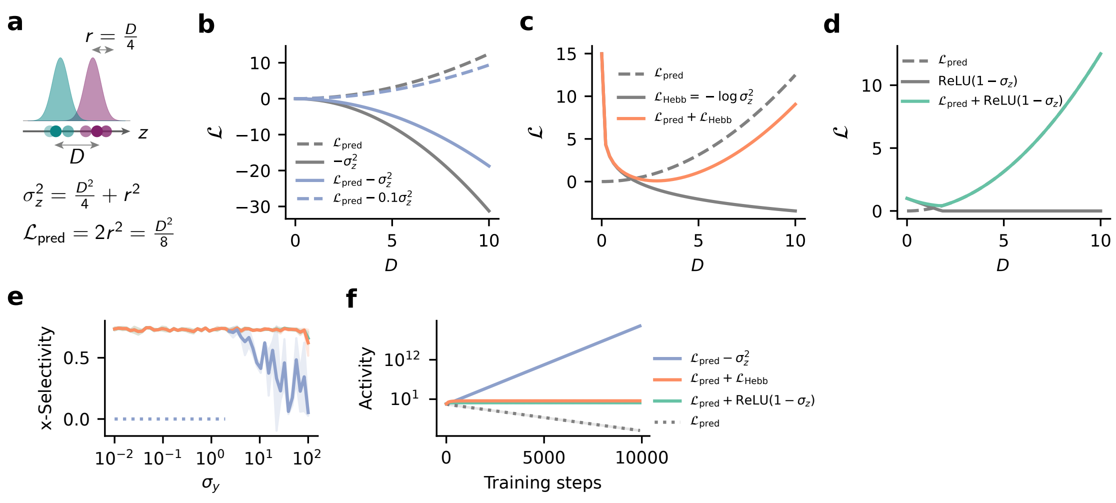


FIGURE A.13: **Variance-dependent modulation of Hebbian learning objective is crucial for stable learning. (a)** Example setting with two clusters distance D apart in representation space. The size of each cluster $r = \frac{D}{4}$ is assumed to scale with D. In this case, representational variance is $\frac{D^2}{4} + r^2$. **(b)** Total loss combining the predictive loss with a naive variance maximising loss $(-\sigma_z^2)$ as a function of cluster separation D. Depending on the coefficient of the variance loss, the global minimum of the loss is either at $D = \infty$ (solid blue) or at $D = 0$ (dashed blue), implying runaway LTP or collapse respectively. **(c)** Same as (b) but using $\mathcal{L}_{\text{Hebb}} = -\log \sigma_z^2$, the Hebbian objective optimized by LPL instead of the naive variance objective. Here, the predictive loss starts dominating at higher values of D inducing a global minimum at $D \approx 3$. **(d)** Same as (b) but the variance loss is now $\text{ReLU}(1 - \sigma_z)$, the objective that was optimized in the VICReg model [28]. Here as well, the predictive loss dominates at higher values of D, inducing a global minimum at $D \approx 2$. **(e)** Cluster selectivity learned by LPL on the two-dimensional synthetic sequence from Fig. 2.2a with the standard predictive loss combined with the different variance losses from (a), (b) and (c). Dotted sections indicate simulations where learning diverged. **(f)** Mean output activity over training time for LPL on the two-dimensional synthetic sequence with $\sigma_y = 1$ for the different cases from (a), (b) and (c).

$\langle (z_1 - z_2)^2 \rangle$ between two independently drawn samples $z_{1/2}$ from the same Gaussian, i.e, $\mathcal{L}_{\text{pred}} = 2r^2$. The overall variance of the representations is $\sigma_z^2 = r^2 + (\frac{D}{2})^2$ (variance of the Gaussian mixture). Under this representational distribution, we studied the overall loss function obtained by combining the predictive loss with different variance-maximising losses, each a different decreasing function of the variance, i.e, $\mathcal{L}_{\text{var}} = f(\sigma_z^2)$. Specifically, we considered the cases $\mathcal{L}_{\text{var}} = -\sigma_z^2$, $\mathcal{L}_{\text{var}} = -\log \sigma_z^2$, and $\mathcal{L}_{\text{var}} = \text{ReLU}(1 - \sigma_z)$. These loss functions are plotted in Fig. A.13b-d respectively along with $\mathcal{L}_{\text{pred}}$, and the resulting full LPL objective in each case. With the naive variance maximization objective $\mathcal{L}_{\text{var}} = -\sigma_z^2$, the full objective is dominated by the variance term at large values of D (Fig. A.13b), and therefore inherently drives unstable learning. It is not possible to remedy this situation by simply using a smaller weight for the variance loss, for instance, by weighting $\mathcal{L}_{\text{var}}$ with a small weight of 0.1. This is because downweighting the variance objective simply moves the loss minimum at $D = \infty$ to $D = 0$, the exact situation of collapse the variance objective is meant to prevent (Fig. A.13b). In contrast, using $\mathcal{L}_{\text{var}} = -\log \sigma_z^2$ (Fig. A.13c), or $\mathcal{L}_{\text{var}} = \text{ReLU}(1 - \sigma_z)$ (Fig. A.13d) along with $\mathcal{L}_{\text{pred}}$ constitute loss landscapes with minima at finite non-zero values of D. This is because these variance objectives only dominate at low values of D, but have diminishing influence with growing D allowing the predictive term to dictate learning, and preventing runaway activity.

We validated these scaling arguments with learning simulations using each of the three proposed learning objectives on the synthetic two-dimensional sequence learning task from Fig. 2.2a. We found that using the naive variance maximization objective $\mathcal{L}_{\text{var}} = -\sigma_z^2$ results in poor learning of cluster selectivity, whereas $\mathcal{L}_{\text{var}} = -\log \sigma_z^2$ and $\mathcal{L}_{\text{var}} = \text{ReLU}(1 - \sigma_z)$ prove effective (Fig. A.13e). Furthermore, the naive variance objective indeed suffers from runaway instability (Fig. A.13f).

A.5 Predictive feature selected by LPL strictly depends on temporal contiguity properties of the input sequence

The slow or "predictive" feature picked up by a single neuron learning with LPL purely depends on the temporal order of stimuli that it is exposed to. One would expect, then, that it is possible to manipulate the learned feature by altering the temporal sequence of stimuli.

To illustrate that this is indeed the case, we designed a predictive learning task similar to that in Fig. 2.2a using a subset of images from the MNIST handwritten digit dataset [192]. Specifically, we sampled 2000 images from this dataset corresponding to the digits "five" and "six", equally distributed between the two classes. We generated sample inputs by embedding these 28×28 grayscale images in a 56×56 blank canvas at either the top-left or bottom-right location. Because we sought to demonstrate that changing the temporal transition structure qualitatively changes neuronal selectivity, we considered two types of sequences with distinct temporal contiguity properties (Fig. A.14). In the Digit Sequence we preserved digit identity (either five or six) across subsequent input images, while changing their position on the canvas, whereas in the Location Sequence, we presented different digits at the same location in successive inputs. Therefore, the predictive feature is location in the Location Sequence and digit identity in the Digit Sequence. Furthermore, digit identity and digit location were approximately aligned with the first two principal components of the data which account for 30 % and 5 % of the explained variance respectively (Fig. A.14).

We again exposed a single rate neuron model to these two sequence types, while allowing the plastic input connections to evolve according to the LPL rule. After convergence, we measured neuronal selectivity to digit identity and location. We measured selectivity to location and digit identity with the same measure defined in Eq. (2.10), only changing what inputs fall into clusters 1 and 2 in each case. Concretely, we measured selectivity to digit identity by setting $\langle z_1 \rangle$ to be the mean response to the digit five (at any location) and $\langle z_2 \rangle$ the mean response to the digit six. Finally, we set $\langle z_1 \rangle$ and $\langle z_2 \rangle$ to the mean responses to digits at the two locations regardless of digit identity in order to measure location selectivity.

At initialization with random weights, the neuron was partially selective to both location and digit identity (Fig. A.14f). However, subsequent training with LPL rendered the neuron purely selective to either location or digit identity depending on which sequence it was exposed to during training. Yet, when the predictive term was turned off, the specific sequence did not matter and the neuron always became selective to location, which coincides with the direction of highest variance in the data (PC1; Supplementary Fig. A.14). Finally, we confirmed that Oja's rule showed the same behavior (Fig. A.14f). Thus, a neuron learning with LPL finds temporally contiguous features in high-dimensional sequential data rather than the direction of largest variance, and the the temporally contiguous feature that is learned is strictly determined by the temporal sequence of the stimuli the neuron is exposed to.

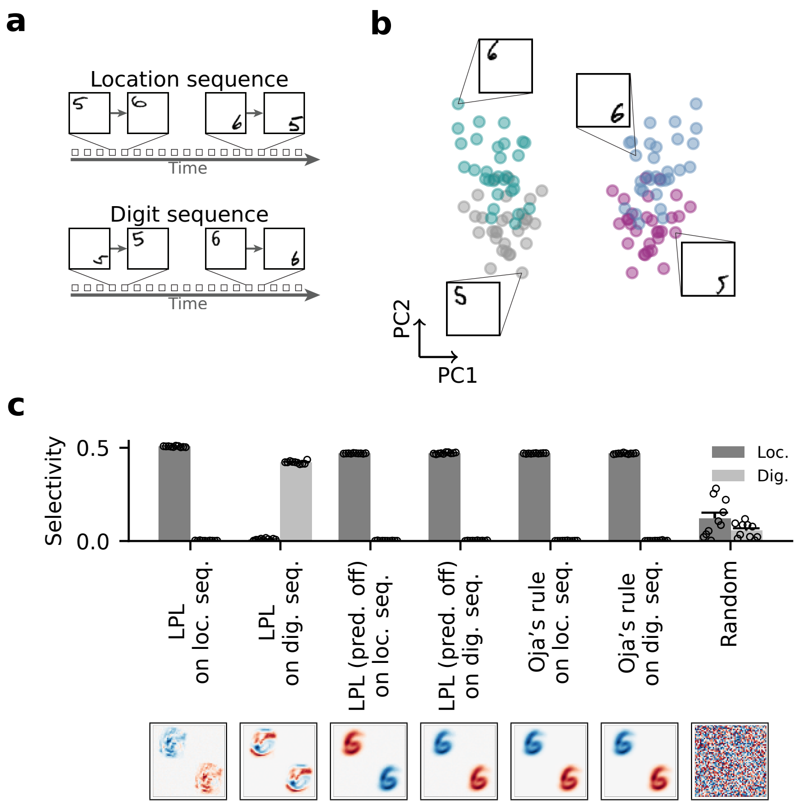


FIGURE A.14: **Temporal contiguities determine features learned under LPL.** **(a)** Schematic of the two kinds of temporal sequences (Methods) in which subsequent inputs were either different digits shown at the same location ("Location Sequence") or the same digit presented at a different location ("Digit Sequence"). **(b)** Scatter plot of the first two principal components of the synthetic digit dataset consisting of randomly sampled handwritten digits in one of the two locations. The principal components closely correspond to location (PC1) and digit identity (PC2). The four digit-position categories are indicated by color. Insets show representative examples from each category. **(c)** Emergent feature selectivity of a single neuron exposed to the two sequence modalities while learning under different rules (top), and the resulting input weights learned in each case (bottom). Under LPL, the neuron's selectivity mirrors the temporally preserved (predictive) property in each sequence, i.e., location in the Location Sequence and digit identity in the Digit Sequence. However, the specific sequence does not matter for Oja's rule or for LPL without the predictive term. In these cases, the neuron always becomes selective to location, the direction of maximum variance. Error bars indicate SEM over ten random seeds.

A.6 Details of the deep neural network architecture

For all DNN simulations, we used the convolutional layers of the VGG-11 architecture consisting of eight blocks each containing 3×3 convolutions, the ReLU activation function followed by a 2×2 max-pool operation in some blocks (detailed architecture description provided below).

```
VGG11Encoder(
  (blocks): ModuleList(
    (0): ConvBlock(
      (module): Sequential(
        (0): Conv2d(3, 64, kernel_size=(3, 3),
               stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0,
                  dilation=1, ceil_mode=False)
      )
    )
    (1): ConvBlock(
      (module): Sequential(
        (0): Conv2d(64, 128, kernel_size=(3, 3),
               stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0,
                  dilation=1, ceil_mode=False)
      )
    )
    (2): ConvBlock(
      (module): Sequential(
        (0): Conv2d(128, 256, kernel_size=(3, 3),
               stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): Identity()
      )
    )
    (3): ConvBlock(
      (module): Sequential(
        (0): Conv2d(256, 256, kernel_size=(3, 3),
               stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0,
                  dilation=1, ceil_mode=False)
      )
    )
    (4): ConvBlock(
      (module): Sequential(
        (0): Conv2d(256, 512, kernel_size=(3, 3),
               stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): Identity()
      )
    )
    (5): ConvBlock(
      (module): Sequential(
        (0): Conv2d(512, 512, kernel_size=(3, 3),
                    stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0,
                       dilation=1, ceil_mode=False)
      )
    )
    (6): ConvBlock(
      (module): Sequential(
        (0): Conv2d(512, 512, kernel_size=(3, 3),
                    stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): Identity()
      )
    )
    (7): ConvBlock(
      (module): Sequential(
        (0): Conv2d(512, 512, kernel_size=(3, 3),
                    stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0,
                       dilation=1, ceil_mode=False)
      )
    )
  )
  (pooler): AdaptiveAvgPool2d(output_size=(1, 1))
)
```

Furthermore, for the simulations modeling unsupervised learning in the IT, we used an adaptive average pooling layer with spatial output dimensions of 13×1. This ensured that the final pooling layer preserved spatial separation along the canvas itself so that the final feature map consisted of 13×1 512-dimensional vectors. We added a fully connected layer on top of these feature maps to finally get a single 512-dimensional feature vector per image.

Appendix B

Supplements to Chapter III

B.1 Supplementary Figures and Tables

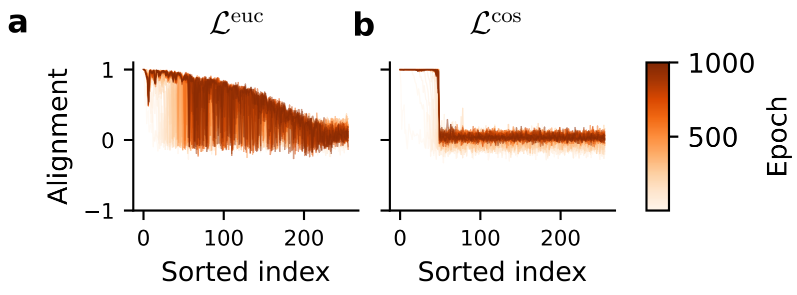


FIGURE B.1: Eigenspace alignment between $C_{\boldsymbol{z}} = \mathbb{E}_{\boldsymbol{x}}\left[\boldsymbol{z}\boldsymbol{z}^\top\right]$ and a linear predictor W_{P} trained with gradient descent. Following [151], we measure eigenspace alignment as the cosine between $\boldsymbol{u}_i$ and $W_{\mathrm{P}}\boldsymbol{u}_i$ for every eigenvector $\boldsymbol{u}_i$ of $C_{\boldsymbol{z}}$. **(a)** Measured alignment for every eigenvector of $C_{\boldsymbol{z}}$ ordered by sorted eigenvalue indices over training epochs, when the network is trained with $\mathcal{L}^{\text{euc}}$. **(b)** Same as **(a)** but for training with $\mathcal{L}^{\text{cos}}$.

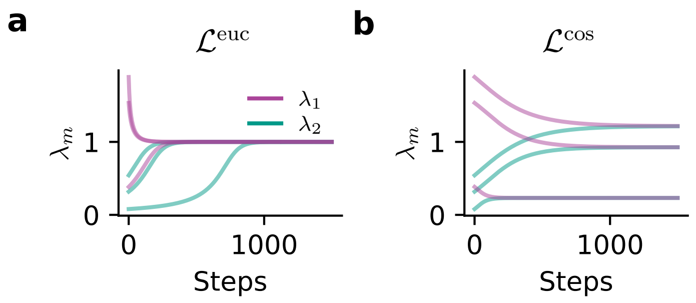
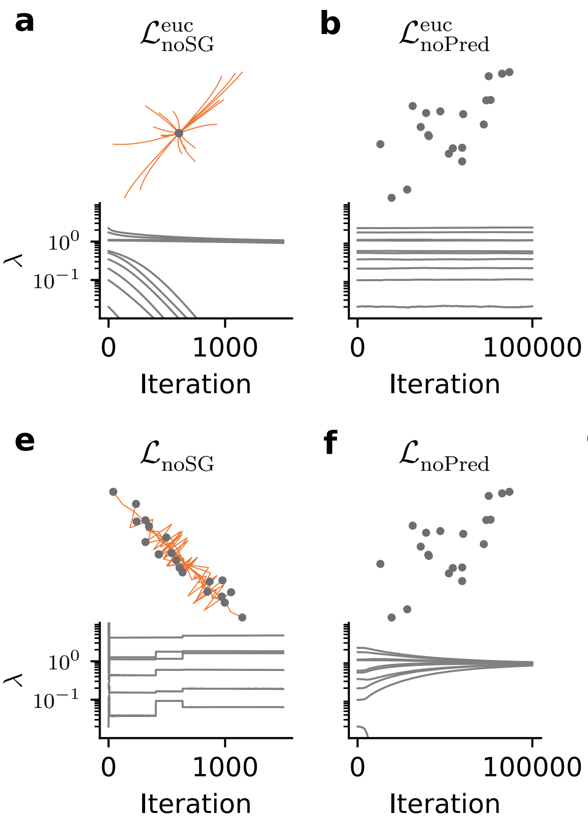


FIGURE B.2: Comparison between the dynamics under Euclidean and cosine asymmetric losses for different initializations in a network with $M = 2$ output neurons. **(a)** Observed dynamics of the eigenvalues in the two-neuron toy network under three different initializations. Both eigenvalues always converge to 1 regardless of the initialization. **(b)** Same as **(a)**, but for the cosine distance. Under different initializations, the two eigenvalues converge to arbitrary, but equal, values.



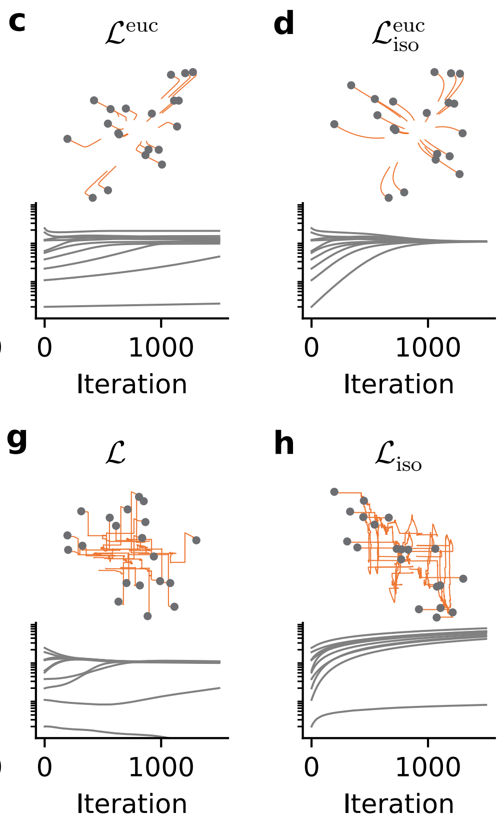


FIGURE B.3: Same as Fig. 3.2 but with a ReLU nonlinearity on the embeddings. We observe learning dynamics qualitatively similar to the linear network.

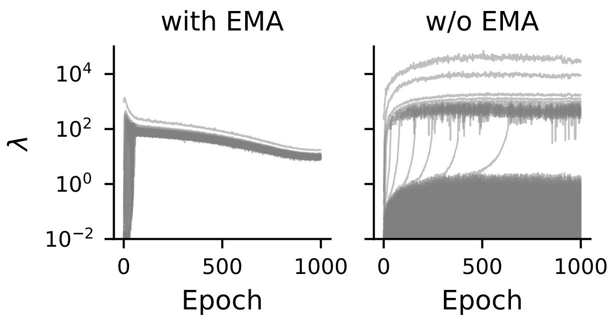


FIGURE B.4: Eigenvalue dynamics for learning under IsoLoss ($\mathcal{L}^{\text{cos}}_{\text{iso}}$) with and without the EMA target network. Removing the EMA results in markedly different dynamics with fewer eigenmodes recruited during training.

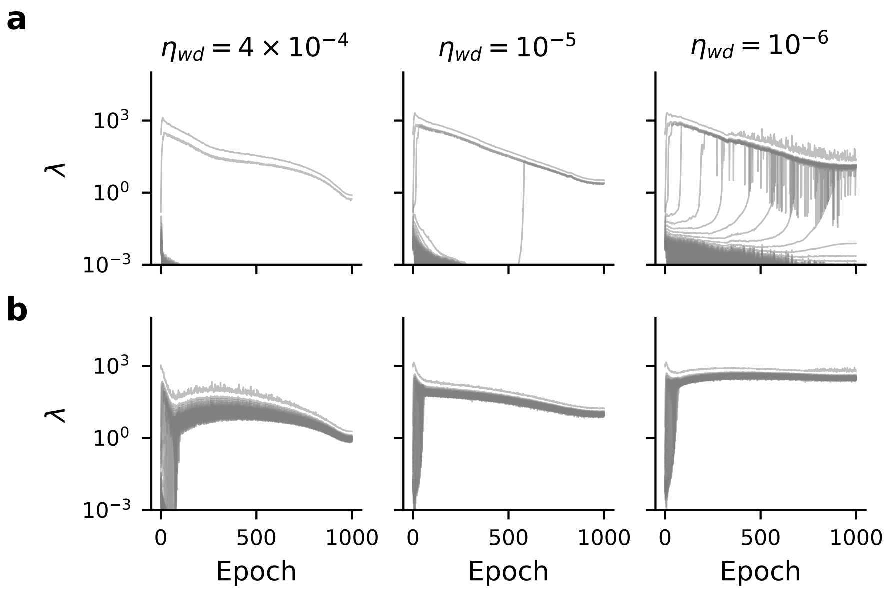


FIGURE B.5: Effect of weight decay on eigenvalue recruitment for DirectPred and IsoLoss. **(a)** Evolution of eigenvalues during learning under DirectPred ($\mathcal{L}^{\text{cos}}$) with EMA and different amounts of weight decay. Decreasing weight decay correlates with the number of eigenvalues recruited during learning. **(b)** Same as **(a)** but for IsoLoss ($\mathcal{L}^{\text{cos}}_{\text{iso}}$). All eigenvalues are recruited independent of the strength of weight decay. However, the magnitude of the eigenvalues inversely correlates with the magnitude of weight-decay.

TABLE B.1: Linear readout validation accuracy in % ± stddev over five random seeds. The † denotes crashed runs, known to occur with symmetric predictors like DirectPred [151].

Model	α	EMA	CIFAR-10	CIFAR-100	STL-10	TinyImageNet
DirectCopy	1	Yes	91.3 ± 0.2	68.7 ± 0.3	89.3 ± 0.2	45.3 ± 0.3
		No	12.1 ± 0.6†	1.5 ± 0.5†	10.3 ± 0.1†	0.6 ± 0.1†
DirectPred	0.5	Yes	92.0 ± 0.2	66.6 ± 0.5	88.8 ± 0.3	40.1 ± 0.5
		No	12.1 ± 1.3†	1.6 ± 0.6†	10.4 ± 0.1†	1.3 ± 0.2†
IsoLoss (ours)	1	Yes	91.5 ± 0.2	69.3 ± 0.2	89.1 ± 0.3	45.6 ± 0.9
		No	91.4 ± 0.2	63.0 ± 0.3	86.9 ± 0.4	38.5 ± 0.5
	0.5	Yes	91.5 ± 0.2	69.0 ± 0.2	89.0 ± 0.3	44.8 ± 0.4
		No	91.5 ± 0.2	64.3 ± 0.3	87.4 ± 0.1	40.4 ± 0.4

B.2 Proofs

Lemma 1. (Euclidean and cosine loss in the predictor eigenspace) *Let W_P be a linear predictor set according to DirectPred with eigenvalues λ_m, and $\hat{\boldsymbol{z}}$ the representations expressed in the predictor's eigenbasis. Then the asymmetric Euclidean loss $\mathcal{L}^\mathrm{euc}$ and cosine loss $\mathcal{L}^\mathrm{cos}$ can be expressed as:*

$$\mathcal{L}^\mathrm{euc} = \tfrac{1}{2}\sum_m^M |\lambda_m \hat{z}_m^{(1)} - \mathrm{SG}(\hat{z}_m^{(2)})|^2 \quad , \tag{3.3}$$

$$\mathcal{L}^\mathrm{cos} = -\sum_m^M \frac{\lambda_m \hat{z}_m^{(1)} \mathrm{SG}(\hat{z}_m^{(2)})}{\|D\hat{\boldsymbol{z}}^{(1)}\|\|\mathrm{SG}(\hat{\boldsymbol{z}}^{(2)})\|} \quad . \tag{3.4}$$

Proof. Under DirectPred, the predictor is a symmetric matrix with eigendecomposition $W_\mathrm{P} = UDU^\top$. Since U is an orthogonal matrix, we also have $UU^\top = I$ so that we can simplify the losses as follows:

$$\begin{aligned}
\mathcal{L}^\mathrm{euc} &= \tfrac{1}{2}\|W_\mathrm{P}\boldsymbol{z}^{(1)} - \mathrm{SG}(\boldsymbol{z}^{(2)})\|^2 \\
&= \tfrac{1}{2}\|UDU^\top\boldsymbol{z}^{(1)} - \mathrm{SG}(UU^\top\boldsymbol{z}^{(2)})\|^2 \\
&= \tfrac{1}{2}\|D\hat{\boldsymbol{z}}^{(1)} - \mathrm{SG}(\hat{\boldsymbol{z}}^{(2)})\|^2 \\
&= \tfrac{1}{2}\sum_m^M |\lambda_m \hat{z}_m^{(1)} - \mathrm{SG}(\hat{z}_m^{(2)})|^2
\end{aligned}$$

$$\begin{aligned}
\mathcal{L} &= -\frac{\left(W_\mathrm{P}\boldsymbol{z}^{(1)}\right)^\top \mathrm{SG}(\boldsymbol{z}^{(2)})}{\|W_\mathrm{P}\boldsymbol{z}^{(1)}\|\|\mathrm{SG}(\boldsymbol{z}^{(2)})\|} \\
&= -\frac{(\boldsymbol{z}^{(1)})^\top UDU^\top \mathrm{SG}(\boldsymbol{z}^{(2)})}{\|UDU^\top\boldsymbol{z}^{(1)}\|\|\mathrm{SG}(UU^\top\boldsymbol{z}^{(2)})\|} \\
&= -\frac{(\hat{\boldsymbol{z}}^{(1)})^\top D\, \mathrm{SG}(\hat{\boldsymbol{z}}^{(2)})}{\|D\hat{\boldsymbol{z}}^{(1)}\|\|\mathrm{SG}(\hat{\boldsymbol{z}}^{(2)})\|} \\
&= -\sum_m^M \frac{\lambda_m \hat{z}_m^{(1)}\, \mathrm{SG}(\hat{z}_m^{(2)})}{\|D\hat{\boldsymbol{z}}^{(1)}\|\|\mathrm{SG}(\hat{\boldsymbol{z}}^{(2)})\|} \quad ,
\end{aligned}$$

where we used the fact that U is orthogonal and therefore does not change the Euclidean norm. $\hat{\boldsymbol{z}} = U^\top\boldsymbol{z}$ is the representation rotated into the eigenbasis. $\square$

Lemma 2. (Learning dynamics in a rotated basis) *Assuming that a given loss $\mathcal{L}$ is optimized by gradient descent on the parameters of a neural network with network outputs $\boldsymbol{z}$, a given orthogonal transformation $\hat{\boldsymbol{z}} = U^\top z$ and learning rate η, then the rotated representations $\hat{\boldsymbol{z}}$ evolve according to the dynamics:*

$$\frac{\mathrm{d}\hat{\boldsymbol{z}}}{\mathrm{d}t} = -\eta \hat{\Theta}_t(\boldsymbol{x}, \mathcal{X}) \nabla_{\hat{\mathcal{Z}}} \mathcal{L} \quad ,$$

where $\hat{\Theta}_t(\mathcal{X}, \mathcal{X}) = \nabla_{\boldsymbol{\theta}} \hat{\mathcal{Z}} \nabla_{\boldsymbol{\theta}} \hat{\mathcal{Z}}^\top$ is the empirical NTK expressed in the rotated basis.

Proof. Let $\boldsymbol{\theta}$ be the parameters of the neural network. Then we obtain the representational dynamics using the chain rule in the continuous-time gradient-flow setting [160]:

$$\begin{aligned} \frac{\mathrm{d}\hat{\boldsymbol{z}}}{\mathrm{d}t} &= \nabla_{\boldsymbol{\theta}} \hat{\boldsymbol{z}} \frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t} \\ &= \nabla_{\boldsymbol{\theta}} \hat{\boldsymbol{z}} \left(-\eta \nabla_{\boldsymbol{\theta}} \mathcal{L}\right) \\ &= \nabla_{\boldsymbol{\theta}} \hat{\boldsymbol{z}} \left(-\eta \nabla_{\boldsymbol{\theta}} \hat{\mathcal{Z}}^\top \nabla_{\hat{\mathcal{Z}}} \mathcal{L}\right) \\ &= -\eta \hat{\Theta}_t(x, \mathcal{X}) \nabla_{\hat{\mathcal{Z}}} \mathcal{L} \quad . \end{aligned}$$

The above is a reiteration of the derivation of Eq. (3.2) given by Lee et al. [160], with an additional orthogonal transformation on the network outputs. □

We proceed by proving the following Lemma which we will use in our proofs of Theorems 1 and 2.

Lemma 3. *The NTK for a linear network is invariant under orthogonal transformations of the network output.*

Proof. We first note that for a linear network, the parameters $\boldsymbol{\theta}$ are just the feedforward weights W. Therefore, for any orthogonal transformation U of the network output:

$$\begin{aligned} \hat{\boldsymbol{z}} &= U^\top f(\boldsymbol{x}) = U^\top W \boldsymbol{x} \\ \Rightarrow \nabla_\theta \hat{\boldsymbol{z}} &= \nabla_W \hat{\boldsymbol{z}} = \nabla_W \left(U^\top W \boldsymbol{x}\right) = \boldsymbol{x}^\top \otimes U^\top, \end{aligned} \tag{B.1}$$

where $\otimes$ is the Kronecker product resulting from the fact that every input vector component appears in the update once for each output component.

We now study $\hat{\Theta}_t(\mathcal{X}, \mathcal{X})$, the transformed empirical NTK (cf. Lemma 2). The $(M \times M)$ diagonal blocks in the full $(M|\mathtt{D}| \times M|\mathtt{D}|)$ empirical NTK $\hat{\Theta}_t(\mathcal{X}, \mathcal{X})$ correspond to single samples and the off-diagonal blocks are cross-terms between samples, where $|\mathtt{D}|$ denotes the size of the training dataset and M the dimension of the outputs. We can develop a generic expression for each $(M \times M)$ block $\hat{\Theta}_t(\boldsymbol{x}_i, \boldsymbol{x}_j)$ corresponding to the interactions between samples i and j as:

$$\begin{aligned}
\hat{\Theta}_t(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \nabla_W \hat{\boldsymbol{z}}_i \nabla_W \hat{\boldsymbol{z}}_j^\top \\
&= \left(\boldsymbol{x}_i^\top \otimes U^\top\right) \left(\boldsymbol{x}_j^\top \otimes U^\top\right)^\top \\
&= \left(\boldsymbol{x}_i^\top \otimes U^\top\right) (\boldsymbol{x}_j \otimes U) \\
&= \left(\boldsymbol{x}_i^\top \boldsymbol{x}_j\right) \otimes \left(U^\top U\right) \\
&= \left(\boldsymbol{x}_i^\top \boldsymbol{x}_j\right) \otimes I_\mathrm{M} \\
&= \left(\boldsymbol{x}_i^\top \boldsymbol{x}_j\right) I_\mathrm{M}.
\end{aligned} \tag{B.2}$$

where we have used the fact that $(A \otimes B)^\top = A^\top \otimes B^\top$ and $(A \otimes B)(C \otimes D) = AC \otimes BD$. Here, I_M is the identity matrix of size M. Noting that Eq. (B.2) is unchanged when U is just the identity matrix completes the proof. $\square$

Theorem 1. (Representational dynamics under $\mathcal{L}^{\text{euc}}$) *For a linear network with i.i.d Gaussian inputs learning with $\mathcal{L}^{\text{euc}}$, the representational dynamics of each mode m independently follow the gradient of the loss $-\nabla_{\hat{\boldsymbol{z}}}\mathcal{L}^{\text{euc}}$. More specifically, the dynamics uncouple and follow M independent differential equations:*

$$\frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t} = -\eta \frac{\partial \mathcal{L}^{\text{euc}}}{\partial \hat{z}_m^{(1)}}(t) = \eta \lambda_m \left(\hat{z}_m^{(2)} - \lambda_m \hat{z}_m^{(1)} \right) \quad , \tag{3.8}$$

which, after taking the expectation over augmentations yields the dynamics:

$$\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = \eta \lambda_m \left(1 - \lambda_m\right) \hat{z}_m \quad . \tag{3.9}$$

Proof. For a linear network with weights $W \in \mathbb{R}^{M \times N}$, we have from Lemma 3 that the empirical NTK $\hat{\Theta}(\mathcal{X}, \mathcal{X})$ in the orthogonal eigenbasis is equal to the empirical NTK $\Theta(\mathcal{X}, \mathcal{X})$ in the original basis. Furthermore from the proof for the lemma (see Eq. (B.2) above), each $(M \times M)$ block of the full $(M|\mathtt{D}| \times M|\mathtt{D}|)$ empirical NTK is given by:

$$\hat{\Theta}_t(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\boldsymbol{x}_i^\top \boldsymbol{x}_j \right) I_{\mathrm{M}}. \tag{B.3}$$

where $I_M \in \mathbb{R}^{M \times M}$ is the identity. Eq. (B.3) gives the total effective interaction between the samples i and j from the dataset. For high-dimensional inputs $\boldsymbol{x}$ drawn from an i.i.d standard Gaussian distribution, we have $\boldsymbol{x}_i^\top \boldsymbol{x}_j \approx \delta_{ij}$ by the central limit theorem. Therefore, in the special case of a linear network with Gaussian i.i.d inputs, the representational dynamics (Lemma 2) simplify as follows:

$$\begin{aligned}
\frac{\mathrm{d}\hat{\boldsymbol{z}}_i^{(1)}}{\mathrm{d}t} &= -\eta \hat{\Theta}_t(\boldsymbol{x}_i, \mathcal{X}) \nabla_{\hat{\mathcal{Z}}} \mathcal{L} \\
&= -\eta \hat{\Theta}_t(\boldsymbol{x}_i, \boldsymbol{x}_i) \nabla_{\hat{\boldsymbol{z}}_i} \mathcal{L} - \eta \sum_{j \neq i} \hat{\Theta}_t(\boldsymbol{x}_i, \boldsymbol{x}_j) \nabla_{\hat{\boldsymbol{z}}_j} \mathcal{L} \\
&= -\eta \left(\boldsymbol{x}_i^\top \boldsymbol{x}_i \right) \nabla_{\hat{\boldsymbol{z}}_i} \mathcal{L} - \eta \sum_{j \neq i} \left(\boldsymbol{x}_i^\top \boldsymbol{x}_j \right) \nabla_{\hat{\boldsymbol{z}}_j} \mathcal{L} \\
&= -\eta \nabla_{\hat{\boldsymbol{z}}_i} \mathcal{L} \quad .
\end{aligned} \tag{B.4}$$

While the assumption of Gaussian i.i.d inputs is quite restrictive, we offer a generalizing interpretation here. Specifically, the above argument also holds when the inputs $\boldsymbol{x}$ are not all mutually orthogonal, but fall into P orthogonal clusters in the input dataset. Then, we would have $\boldsymbol{x}_i^\top \boldsymbol{x}_j \approx \delta_{p_i = p_j}$ where p_i is the "label" of the cluster corresponding to sample i. If $\mathcal{P}_i$ is the number of all the samples with the same label p_i, then Eq. (B.4) would simply be scaled to give $\frac{\mathrm{d}\hat{\boldsymbol{z}}_i^{(1)}}{\mathrm{d}t} = -\eta \mathcal{P}_i \nabla_{\hat{\boldsymbol{z}}_i} \mathcal{L}$.

For brevity, we proceed with the simplest case Eq. (B.4) in which every input is orthogonal. For $\mathcal{L}^{\text{euc}}$, the representational gradient $\nabla_{\hat{\boldsymbol{z}}_i}\mathcal{L}$ is then given by:

$$\nabla_{\hat{\boldsymbol{z}}_i}\mathcal{L}^{\text{euc}} = \left(D_t\hat{\boldsymbol{z}}_i^{(1)} - \hat{\boldsymbol{z}}_i^{(2)}\right) D_t$$

Noting that D_t is just a diagonal matrix containing the eigenvalues λ_m and dropping the sample subscript i for notational ease, we obtain for the m-th component of $\nabla_{\hat{\boldsymbol{z}}_i}\mathcal{L}^{\text{euc}}$:

$$\frac{\partial\mathcal{L}^{\text{euc}}}{\partial\hat{z}_m} = \lambda_m(\lambda_m\hat{z}_m^{(1)} - \hat{z}_m^{(2)}) \quad .$$

Substituting this result in Eq. (B.4) gives us Eq. (3.8), the expression we were looking for. Finally, introducing $\hat{z}_m \equiv \mathbb{E}[\hat{z}_m^{(1)}] = \mathbb{E}[\hat{z}_m^{(2)}]$ as the expectation over augmentations, we find that each eigenmode evolves independently in expectation value as:

$$\begin{aligned}\mathbb{E}\left[\frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t}\right] = \frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} &= \eta\lambda_m\left(\mathbb{E}[\hat{z}_m^{(2)}] - \lambda_m\mathbb{E}[\hat{z}_m^{(1)}]\right) \\ &= \eta\lambda_m\left(1-\lambda_m\right)\hat{z}_m \quad .\end{aligned}$$

□

Theorem 2. (Representational dynamics under $\mathcal{L}^{\cos}$) *For a linear network with i.i.d Gaussian inputs trained with $\mathcal{L}^{\cos}$, the dynamics follow a system of M coupled differential equations:*

$$\frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t} = \eta \frac{\lambda_m}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \sum_{k \neq m} \lambda_k \left(\lambda_k (\hat{z}_k^{(1)})^2 \hat{z}_m^{(2)} - \lambda_m \hat{z}_m^{(1)} \hat{z}_k^{(1)} \hat{z}_k^{(2)} \right) \quad , \tag{3.10}$$

and reach a regime in which the eigenvalues are comparable in magnitude. In this regime, the expected update over augmentations is well approximated by:

$$\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} \approx \eta \lambda_m \cdot \mathbb{E}\left[\frac{\hat{z}_m^2}{\|D\hat{\boldsymbol{z}}\|^3} \right] \cdot \mathbb{E}\left[\frac{\hat{z}_m}{\|\hat{\boldsymbol{z}}\|} \right] \cdot \sum_{k \neq m} \lambda_k \left(\lambda_k - \lambda_m \right), \tag{3.11}$$

Proof. We can retrace the steps from the proof for Theorem 1 until Eq. (B.4):

$$\frac{\mathrm{d}\hat{\boldsymbol{z}}_i^{(1)}}{\mathrm{d}t} = -\eta \nabla_{\hat{\boldsymbol{z}}_i} \mathcal{L} \quad .$$

$\nabla_{\hat{\boldsymbol{z}}_i} \mathcal{L}$ is a vector of dimension M. Ignoring the sample subscript i for simplicity, and focusing on the m-th component of $\nabla_{\hat{\boldsymbol{z}}_i} \mathcal{L}$, we get:

$$\begin{aligned}
\mathcal{L}^{\cos} &= -\sum_m^M \frac{\lambda_m \hat{z}_m^{(1)} \mathrm{SG}(\hat{z}_m^{(2)})}{\|D\hat{\boldsymbol{z}}^{(1)}\| \|\mathrm{SG}(\hat{\boldsymbol{z}}^{(2)})\|} \\
\Rightarrow \frac{\partial \mathcal{L}}{\partial \hat{z}_m^{(1)}} &= -\frac{\lambda_m \hat{z}_m^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\| \|\hat{\boldsymbol{z}}^{(2)}\|} + \frac{\sum_k \lambda_k \hat{z}_k^{(1)} \hat{z}_k^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \cdot \lambda_m^2 \hat{z}_m^{(1)} \\
&= -\frac{\lambda_m}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \left[\|D\hat{\boldsymbol{z}}^{(1)}\|^2 \hat{z}_m^{(2)} - \lambda_m \hat{z}_m^{(1)} \left(\sum_k \lambda_k \hat{z}_k^{(1)} \hat{z}_k^{(2)} \right) \right] \\
&= -\frac{\lambda_m}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \left[\left(\sum_k \lambda_k^2 (\hat{z}_k^{(1)})^2 \right) \hat{z}_m^{(2)} - \lambda_m \hat{z}_m^{(1)} \left(\sum_k \lambda_k \hat{z}_k^{(1)} \hat{z}_k^{(2)} \right) \right] \\
&= -\frac{\lambda_m}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \sum_{k \neq m} \lambda_k \left(\lambda_k (\hat{z}_k^{(1)})^2 \hat{z}_m^{(2)} - \lambda_m \hat{z}_m^{(1)} \hat{z}_k^{(1)} \hat{z}_k^{(2)} \right) \\
\Rightarrow \frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t} &= -\eta \frac{\partial \mathcal{L}}{\partial \hat{z}_m^{(1)}} \\
&= \frac{\eta \lambda_m}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \sum_{k \neq m} \lambda_k \left(\lambda_k (\hat{z}_k^{(1)})^2 \hat{z}_m^{(2)} - \lambda_m \hat{z}_m^{(1)} \hat{z}_k^{(1)} \hat{z}_k^{(2)} \right) \quad ,
\end{aligned}$$

proving Eq. (3.10). Assuming sufficiently small augmentations, $\hat{z}_k^{(1)}$ and $\hat{z}_k^{(2)}$ carry the same sign, and the net sign of both terms inside the parenthesis is fully determined by

$\gamma_m \equiv \text{sign}(\hat{z}_m^{(1)})$. Hence, we may write:

$$\frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t} = \frac{\eta\lambda_m\gamma_m}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3\|\hat{\boldsymbol{z}}^{(2)}\|}\sum_{k\neq m}\left(\lambda_k^2(\hat{z}_k^{(1)})^2|\hat{z}_m^{(2)}| - \lambda_m\lambda_k|\hat{z}_m^{(1)}||\hat{z}_k^{(1)}||\hat{z}_k^{(2)}|\right) \quad .$$

It is useful to separate out γ_m in this manner because every other term in the expression is now non-negative. Then $\text{sign}(\gamma_m \cdot \frac{\mathrm{d}\hat{z}}{\mathrm{d}t}) = \text{sign}(\hat{z}_m \cdot \frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t})$ tells us whether $\hat{z}_m$ tends to increase or decrease in magnitude, as we have argued in the main text.

Asymptotic analysis. To get a handle on how the different eigenvalues influence each other, we consider two important limiting cases. First, we consider the asymptotic regime dominated by one eigenvalue, and show that it tends towards a more symmetric solution in which the gap between different eigenvalues decreases. Second, we derive asymptotic expressions for the near-uniform regime in which all eigenvalues are comparable in size and show that this solution tends toward the uniform solution (cf. Eq. (3.11)).

To facilitate our analysis, we define each mode's relative contribution $\chi_m \equiv \frac{|\hat{z}_m|}{\|\hat{\boldsymbol{z}}\|}$ and evaluate Eq. (3.10) taking the expectation value over augmentations:

$$\begin{aligned}
\mathbb{E}\left[\frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t}\right] &= \eta\lambda_m\sum_{k\neq m}\left(\lambda_k^2\cdot\mathbb{E}\left[\frac{(\hat{z}_k^{(1)})^2\hat{z}_m^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3\|\hat{\boldsymbol{z}}^{(2)}\|}\right] - \lambda_m\lambda_k\cdot\mathbb{E}\left[\frac{\hat{z}_m^{(1)}\hat{z}_k^{(1)}\hat{z}_k^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3\|\hat{\boldsymbol{z}}^{(2)}\|}\right]\right)\\
\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} &= \eta\lambda_m\sum_{k\neq m}\left(\lambda_k^2\cdot\mathbb{E}\left[\frac{\hat{z}_k^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{\hat{z}_m}{\|\hat{\boldsymbol{z}}\|}\right] - \lambda_m\lambda_k\cdot\mathbb{E}\left[\frac{\hat{z}_m\hat{z}_k}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{\hat{z}_k}{\|\hat{\boldsymbol{z}}\|}\right]\right)\\
&= \eta\lambda_m\gamma_m\sum_{k\neq m}\left(\lambda_k^2\cdot\mathbb{E}\left[\chi_k^2\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\chi_m\right] - \lambda_m\lambda_k\cdot\mathbb{E}\left[\chi_m\chi_k\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\chi_k\right]\right)
\end{aligned} \tag{B.5}$$

In the second equality we used the fact that the expectation value taken over augmentations is conditioned on the input sample, which makes them conditionally independent.

One dominant eigenvalue. First, we consider the low-rank regime in which one eigenvalue dominates. Without loss of generality, we assume $\lambda_1 \gg \lambda_k \; \forall k \neq 1$. We then have:

$$\begin{aligned}
&\chi_1 \sim 1\\
&\chi_k \sim \epsilon \quad (0 < \epsilon \ll 1) \quad \forall \quad k \neq 1
\end{aligned}$$

Plugging these values into Eq. (B.5) gives the following dynamics for the dominant eigenmode:

$$\begin{aligned}
\frac{\mathrm{d}\hat{z}_1}{\mathrm{d}t} &\approx \eta\lambda_1\gamma_1 \sum_{k\neq 1} \left(\lambda_k^2 \cdot \mathbb{E}\left[\epsilon^2 \frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \cdot \mathbb{E}\left[1\right] - \lambda_1\lambda_k \cdot \mathbb{E}\left[\epsilon \frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \mathbb{E}\left[\epsilon\right] \right) \\
&= \eta\lambda_1\gamma_1 \sum_{k\neq 1} \left(\lambda_k^2\epsilon^2 \cdot \mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \cdot \mathbb{E}\left[1\right] - \lambda_1\lambda_k\epsilon^2 \cdot \mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \right) \\
&= \eta\lambda_1\gamma_1\epsilon^2 \mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \sum_{k\neq 1} \lambda_k\left(\lambda_k - \lambda_1\right) \quad .
\end{aligned}$$

These updates are always opposite in sign to the representation component, which corresponds to decaying dynamics for the leading eigenmode because $\gamma_1 \frac{\mathrm{d}\hat{z}_1}{\mathrm{d}t} < 0$.

For all other modes we have:

$$\begin{aligned}
\frac{\mathrm{d}\hat{z}_{m\neq 1}}{\mathrm{d}t} &\approx \eta\lambda_m\gamma_m \sum_{k\notin\{m,1\}} \left(\lambda_k^2\epsilon^2 \cdot \mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \cdot \mathbb{E}\left[\epsilon\right] - \lambda_m\lambda_k\epsilon^2 \cdot \mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \cdot \mathbb{E}\left[\epsilon\right] \right) \\
&\quad + \eta\lambda_m\gamma_m \left(\lambda_1^2 \cdot \mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \cdot \mathbb{E}\left[\epsilon\right] - \lambda_m\lambda_1\epsilon \cdot \mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \cdot \mathbb{E}\left[1\right] \right) \\
&= \eta\lambda_m\gamma_m\epsilon \cdot \mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \left(\lambda_1(\lambda_1 - \lambda_m) + \epsilon^2 \sum_{k\notin\{m,1\}} \lambda_k(\lambda_k - \lambda_m) \right) \\
&\approx \eta\lambda_m\gamma_m\lambda_1\epsilon \cdot \mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] (\lambda_1 - \lambda_m) \quad ,
\end{aligned}$$

so that $\gamma_m \frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} > 0$, i.e, the updates have the *same* sign as the representation component, which corresponds to growth dynamics. In other words: The dominant eigenvalue "pulls all the other eigenvalues up," a form of implicit cooperation between the eigenmodes. We also note that the non-dominant eigenmodes increase at a rate proportional to ϵ, whereas the dominant eigenmode decreases at a slower rate proportional to ϵ^2. Thus, for sensible initializations with at least one large and many small eigenvalues, the modes will tend toward an equilibrium at some non-zero intermediate value, without a dominant mode. Next we study this other limiting case in which all eigenvalues are of similar size.

Near-uniform regime. To study the dynamics in a near-uniform regime, we note that all χ_m are of order $\mathcal{O}(1)$ in $\hat{z}_m$, whereas the eigenvalues λ_m are of order $\mathcal{O}(\hat{z}_m^2)$. In this setting, the effect of the eigenvalue terms λ_m on the dynamics is stronger than the χ_m terms which are bounded between 0 and 1. With a sufficiently high-dimensional representation, all χ_m terms will be centered around $1/\sqrt{M}$. Based on these observations, we may make the simplifying assumption that the contributions are all approximately equal, i.e, $\chi_i = \chi$

for all i. Substituting this value in Eq (B.5) gives:

$$\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = \eta\lambda_m\gamma_m \cdot \mathbb{E}\left[\chi^2 \frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \cdot \mathbb{E}\left[\chi\right] \cdot \sum_{k\neq m} \lambda_k \left(\lambda_k - \lambda_m\right) \quad . \tag{B.6}$$

Finally, substituting for χ, which by assumption are all approximately equal:

$$\chi \approx \chi_m = \frac{|\hat{z}_m|}{\|\hat{\boldsymbol{z}}\|} \quad ,$$

and absorbing back the sign from γ_m, we obtain the approximate dynamics in Eq (3.11):

$$\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} \approx \eta\lambda_m \cdot \mathbb{E}\left[\frac{\hat{z}_m^2}{\|D\hat{\boldsymbol{z}}\|^3}\right] \cdot \mathbb{E}\left[\frac{\hat{z}_m}{\|\hat{\boldsymbol{z}}\|}\right] \cdot \sum_{k\neq m} \lambda_k \left(\lambda_k - \lambda_m\right)$$

□

B.3 Derivation of idealized learning dynamics for different loss variations

B.3.1 Removing the stop-grad from the Euclidean loss $\mathcal{L}^{\text{euc}}$

Omitting the stop-grad operator from $\mathcal{L}^{\text{euc}}$ gives:

$$\begin{aligned}\mathcal{L}^{\text{euc}}_{\text{noSG}} &= \frac{1}{2}\|W_{\text{P}}\boldsymbol{z}^{(1)} - \boldsymbol{z}^{(2)}\|^2 \\ &= \frac{1}{2}\sum_m^M |\lambda_m \hat{z}_m^{(1)} - \hat{z}_m^{(2)}|^2 \quad .\end{aligned}$$

Tracing the steps to prove Theorem 1 and assuming Gaussian i.i.d inputs for a linear network, we write:

$$\begin{aligned}\frac{\partial \mathcal{L}^{\text{euc}}_{\text{noSG}}}{\partial \hat{z}_m} &= \frac{\partial \mathcal{L}^{\text{euc}}_{\text{noSG}}}{\partial \hat{z}_m^{(1)}} + \frac{\partial \mathcal{L}^{\text{euc}}_{\text{noSG}}}{\partial \hat{z}_m^{(2)}} \\ &= \left(\lambda_m \hat{z}_m^{(1)} - \hat{z}_m^{(2)}\right)\lambda_m - \left(\lambda_m \hat{z}_m^{(1)} - \hat{z}_m^{(2)}\right) \\ &= \left(\lambda_m \hat{z}_m^{(1)} - \hat{z}_m^{(2)}\right)(\lambda_m - 1) \\ \Rightarrow \frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} &= -\eta \mathbb{E}\left[\frac{\partial \mathcal{L}^{\text{euc}}_{\text{noSG}}}{\partial \hat{z}_m}\right] \\ &= -\eta\left(\lambda_m \mathbb{E}[\hat{z}_m^{(1)}] - \mathbb{E}[\hat{z}_m^{(2)}]\right)(\lambda_m - 1) \\ &= -\eta\,(1 - \lambda_m)^2\,\hat{z}_m \quad ,\end{aligned}$$

which results in decaying representations and thus collapse.

B.3.2 Removing the stop-grad from the cosine loss $\mathcal{L}^{\text{cos}}$

Following the same arguments as above, omitting the stop-grad operator from $\mathcal{L}^{\text{cos}}$ gives:

$$\begin{aligned}\mathcal{L}^{\text{cos}}_{\text{noSG}} &= -\frac{\left(W_{\text{P}}\boldsymbol{z}^{(1)}\right)^\top \boldsymbol{z}^{(2)}}{\|W_{\text{P}}\boldsymbol{z}^{(1)}\|\|\boldsymbol{z}^{(2)}\|} \\ \Rightarrow \frac{\partial \mathcal{L}^{\text{cos}}_{\text{noSG}}}{\partial \hat{z}_m} &= \frac{-\lambda_m}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3\|\hat{\boldsymbol{z}}^{(2)}\|}\sum_{k\neq m}\big(\lambda_k^2(\hat{z}_k^{(1)})^2\hat{z}_m^{(2)} - \lambda_m\lambda_k\hat{z}_m^{(1)}\hat{z}_k^{(1)}\hat{z}_k^{(2)} \\ &\qquad + \lambda_k^2\lambda_m(\hat{z}_k^{(1)})^3 - \lambda_k\hat{z}_m^{(2)}\hat{z}_k^{(2)}\hat{z}_k^{(1)}\big) \\ &\quad + \frac{-\lambda_m}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3\|\hat{\boldsymbol{z}}^{(2)}\|}\left(\lambda_m^3(\hat{z}_m^{(1)})^3 - \lambda_m(\hat{z}_m^{(2)})^2\hat{z}_m^{(1)}\right) \quad ,\end{aligned}$$

so that, when taking the expectation value over augmentations, the dynamics follow:

$$\begin{aligned}
\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} &= -\eta\mathbb{E}\left[\frac{\partial\mathcal{L}^{\cos}_{\text{noSG}}}{\partial\hat{z}_m}\right] \\
&= \eta\lambda_m\gamma_m\sum_{k\neq m}\lambda_k\left(\lambda_k\cdot\mathbb{E}\left[\chi_k^2\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\chi_m\right]-\lambda_m\cdot\mathbb{E}\left[\chi_m\chi_k\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\chi_k\right]\right) \\
&\quad+\eta\lambda_m\gamma_m\sum_{k\neq m}\lambda_k\left(\lambda_m\lambda_k\mathbb{E}\left[\chi_k^3\frac{\|\hat{\boldsymbol{z}}\|^3}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{1}{\|\hat{\boldsymbol{z}}\|}\right]-\mathbb{E}\left[\chi_k\frac{\|\hat{\boldsymbol{z}}\|}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\chi_m\chi_k\|\hat{\boldsymbol{z}}\|\right]\right) \\
&\quad+\eta\lambda_m^2\gamma_m\left(\lambda_m^2\mathbb{E}\left[\chi_m^3\frac{\|\hat{\boldsymbol{z}}\|^3}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{1}{\|\hat{\boldsymbol{z}}\|}\right]-\mathbb{E}\left[\chi_m\frac{\|\hat{\boldsymbol{z}}\|}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\chi_m^2\|\hat{\boldsymbol{z}}\|\right]\right) \quad .
\end{aligned}$$

In the asymptotic regime with dominant eigenvalue λ_1, we get the dynamics:

$$\begin{aligned}
\frac{\mathrm{d}\hat{z}_1}{\mathrm{d}t} &= \eta\lambda_1\gamma_1\sum_{k\neq m}\lambda_k\left(\lambda_k\epsilon^2\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]-\lambda_m\epsilon^2\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]\right) \\
&\quad+\eta\lambda_1\gamma_1\sum_{k\neq m}\lambda_k\left(\lambda_1\lambda_m\epsilon^3\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^3}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{1}{\|\hat{\boldsymbol{z}}\|}\right]-\epsilon^2\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\|\hat{\boldsymbol{z}}\|\right]\right) \\
&\quad+\eta\lambda_1^2\gamma_1\left(\lambda_1^2\cdot\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^3}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{1}{\|\hat{\boldsymbol{z}}\|}\right]-\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\|\hat{\boldsymbol{z}}\|\right]\right) \\
&\approx \eta\lambda_1^4\gamma_1\cdot\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^3}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{1}{\|\hat{\boldsymbol{z}}\|}\right] \\
\frac{\mathrm{d}\hat{z}_{m\neq 1}}{\mathrm{d}t} &= \eta\lambda_m\gamma_m\sum_{k\notin\{m,1\}}\lambda_k\left(\lambda_k\epsilon^3\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]-\lambda_m\epsilon^3\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]\right) \\
&\quad+\eta\lambda_m\gamma_m\lambda_1\left(\lambda_1\epsilon\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]-\lambda_m\epsilon\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]\right) \\
&\quad+\eta\lambda_m\gamma_m\sum_{k\notin\{m,1\}}\lambda_k\left(\lambda_m\lambda_k\epsilon^3\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^3}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{1}{\|\hat{\boldsymbol{z}}\|}\right]-\epsilon^3\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\|\hat{\boldsymbol{z}}\|\right]\right) \\
&\quad+\eta\lambda_m\gamma_m\lambda_1\left(\lambda_m\lambda_1\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^3}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{1}{\|\hat{\boldsymbol{z}}\|}\right]-\epsilon\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\|\hat{\boldsymbol{z}}\|\right]\right) \\
&\quad+\eta\lambda_m^2\gamma_m\left(\lambda_m^2\epsilon^3\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^3}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{1}{\|\hat{\boldsymbol{z}}\|}\right]-\epsilon^3\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\|\hat{\boldsymbol{z}}\|\right]\right) \\
&\approx \eta\lambda_m^2\gamma_m\lambda_1^2\cdot\mathbb{E}\left[\frac{\|\hat{\boldsymbol{z}}\|^3}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{1}{\|\hat{\boldsymbol{z}}\|}\right],
\end{aligned}$$

Thus, all eigenmodes diverge because $\gamma_m\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} > 0$.

Similarly, we find divergent dynamics when starting in the near-uniform regime:

$$\begin{aligned}\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} &= \eta\lambda_m\gamma_m\mathbb{E}\left[\chi^2\frac{\|\hat{\boldsymbol{z}}\|^2}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\chi\right]\sum_{k\neq m}\lambda_k\left(\lambda_k-\lambda_m\right)\\
&+\eta\lambda_m^2\gamma_m\mathbb{E}\left[\chi^3\frac{\|\hat{\boldsymbol{z}}\|^3}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{1}{\|\hat{\boldsymbol{z}}\|}\right]\sum_k\lambda_k^2\\
&-\eta\lambda_m\gamma_m\mathbb{E}\left[\chi\frac{\|\hat{\boldsymbol{z}}\|}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\chi^2\|\hat{\boldsymbol{z}}\|\right]\sum_k\lambda_k\\
&\approx\eta\lambda_m^2\gamma_m\mathbb{E}\left[\chi^3\frac{\|\hat{\boldsymbol{z}}\|^3}{\|D\hat{\boldsymbol{z}}\|^3}\right]\cdot\mathbb{E}\left[\frac{1}{\|\hat{\boldsymbol{z}}\|}\right]\sum_k\lambda_k^2 \quad ,\end{aligned}$$

selecting the terms with the highest power in the eigenvalues.

Thus, omission of stop-grad precludes successful representation learning for both the Euclidean and the cosine loss, but due to different mechanisms. Euclidean loss yields collapse, whereas the cosine loss succumbs to run-away activity.

B.3.3 Removing the predictor from the Euclidean loss $\mathcal{L}^{\text{euc}}$

To analyze the representational dynamics in the absence of the predictor network, we consider $\mathcal{L}^{\text{euc}}_{\text{noPred}}$:

$$\begin{aligned}\mathcal{L}^{\text{euc}}_{\text{noPred}} &= \frac{1}{2}\|\boldsymbol{z}^{(1)}-\text{SG}(\boldsymbol{z}^{(2)})\|^2\\
&=\frac{1}{2}\sum_m^M|\hat{z}_m^{(1)}-\text{SG}(\hat{z}_m^{(2)})|^2 \quad .\end{aligned}$$

The dynamics resulting from this loss function are a special case of the dynamics derived in Theorem 1 with all the eigenvalues equal to one ($\lambda_k = 1$). In particular Eq. (3.8) becomes:

$$\frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t} = -\eta\frac{\partial\mathcal{L}^{\text{euc}}_{\text{noPred}}}{\partial\hat{z}_m^{(1)}}(t) = \eta\left(\hat{z}_m^{(2)}-\hat{z}_m^{(1)}\right),$$

which evaluates to 0 under expectation over augmentations. Hence there is no learning without the predictor.

B.3.4 Removing the predictor from the cosine loss $\mathcal{L}^{\text{cos}}$

Similarly, when we remove the predictor from $\mathcal{L}^{\text{cos}}$ it yields:

$$\mathcal{L}^{\text{cos}}_{\text{noPred}} = -\sum_m^M\frac{\hat{z}_m^{(1)}\text{SG}(\hat{z}_m^{(2)})}{\|\hat{\boldsymbol{z}}^{(1)}\|\|\text{SG}(\hat{\boldsymbol{z}}^{(2)})\|} \quad ,$$

so that Eq. (3.10) becomes:

$$\frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t} = \frac{\eta}{\|\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \sum_{k \neq m} \left((\hat{z}_k^{(1)})^2 \hat{z}_m^{(2)} - \hat{z}_m^{(1)} \hat{z}_k^{(1)} \hat{z}_k^{(2)} \right) \quad . \tag{B.7}$$

Here, the near-uniform approximation (Eq. (3.11)) of ignoring the differences in χ between different eigenmodes is not valid. This is because the λ-terms are no longer present, and the effects of the χ-terms on the dynamics cannot be treated as negligible. In particular, setting $W_{\mathrm{P}} = I$ in the dynamics derived in Theorem 2 would yield zero dynamics. However taking the expectation of Eq. (B.7) over augmentations yields the non-zero dynamics:

$$\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = \eta \sum_{k \neq m} \left(\mathbb{E}\left[\frac{\hat{z}_k^2}{\|\hat{\boldsymbol{z}}\|^3} \right] \mathbb{E}\left[\frac{\hat{z}_m}{\|\hat{\boldsymbol{z}}\|} \right] - \mathbb{E}\left[\frac{\hat{z}_m \hat{z}_k}{\|\hat{\boldsymbol{z}}\|^3} \right] \mathbb{E}\left[\frac{\hat{z}_k}{\|\hat{\boldsymbol{z}}\|} \right] \right), \tag{B.8}$$

which consists of terms of order $\mathcal{O}\left(\frac{1}{\hat{z}_m}\right)$ and $\mathcal{O}(1)$ in $\hat{z}_m$, hinting at slower dynamics compared to Eq. (3.10). To study these granular effects, we would need to explicitly model the effect of the augmentations, for which we do not have a good statistical model. In lieu of deriving these dynamics analytically, we make an observation which restricts the possible dynamical behavior. Specifically, the sum of the eigenvalues remains constant throughout training. To show this, we begin by writing out the expression for the derivative over time of the sum of all the eigenvalues:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \sum_m \lambda_m = \sum_m \frac{\mathrm{d}\lambda_m}{\mathrm{d}t} &= \sum_m \frac{\mathrm{d}}{\mathrm{d}t} \mathbb{E}_{\mathrm{data}}\left[\hat{z}_m^2 \right] \\ &= \mathbb{E}_{\mathrm{data}}\left[\sum_m \frac{\mathrm{d}\hat{z}_m^2}{\mathrm{d}t} \right] \\ &= \mathbb{E}_{\mathrm{data}}\left[\sum_m \hat{z}_m \frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} \right] . \end{aligned}$$

We can derive the term inside the expectation by adding up the dynamics given by Eq. (B.8) for all the different eigenmodes:

$$\begin{aligned} \hat{z}_m^{(1)} \frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t} &= \frac{\eta}{\|\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \sum_{k \neq m} \left((\hat{z}_k^{(1)})^2 \hat{z}_m^{(1)} \hat{z}_m^{(2)} - (\hat{z}_m^{(1)})^2 \hat{z}_k^{(1)} \hat{z}_k^{(2)} \right) \\ \implies \sum_m \hat{z}_m^{(1)} \frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t} &= \frac{\eta}{\|\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \sum_m \sum_{k \neq m} \left((\hat{z}_k^{(1)})^2 \hat{z}_m^{(1)} \hat{z}_m^{(2)} - (\hat{z}_m^{(1)})^2 \hat{z}_k^{(1)} \hat{z}_k^{(2)} \right) \\ &= 0 \\ \implies \mathbb{E}_{\mathrm{data}}\left[\mathbb{E}_{\mathrm{aug}}\left[\sum_m \hat{z}_m^{(1)} \frac{\mathrm{d}\hat{z}_m^{(1)}}{\mathrm{d}t} \right] \right] &= \mathbb{E}_{\mathrm{data}}\left[\sum_m \hat{z}_m \frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} \right] = 0 \quad , \end{aligned}$$

proving that $\frac{\mathrm{d}}{\mathrm{d}t}\sum_m \lambda_m = 0$, i.e, the sum of the eigenvalues is conserved. This precludes collapsing dynamics where all eigenvalues go to zero as well as diverging dynamics where at least one eigenvalue diverges.

B.3.5 Isotropic losses for equalized convergence rates

In Expressions (3.9) and (3.11) we see that the overall learning dynamics have a quadratic dependence on the eigenvalues with a root near collapsed solutions, which causes these modes to learn slower. We reasoned that this anisotropy could be detrimental for learning. To address this issue, we sought to derive alternative loss functions that encourage isotropic learning dynamics for all modes.

Euclidean IsoLoss.

We start by deriving an IsoLoss function for the Euclidean case $\mathcal{L}^{\text{euc}}$. To avoid the unwanted quadratic dependence, we first note that we would like to arrive at the following expression for the dynamics:

$$\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = \eta\,(1-\lambda_m)\,\hat{z}_m \quad .$$

By recalling the Euclidean loss and corresponding dynamics:

$$\mathcal{L}^{\text{euc}} = \tfrac{1}{2}\sum_m^M |\lambda_m \hat{z}_m^{(1)} - \text{SG}(\hat{z}_m^{(2)})|^2 \Rightarrow \frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = \eta\lambda_m\,(1-\lambda_m)\,\hat{z}_m \quad ,$$

we note that the leading λ_m term has no influence on the overall sign of the dynamics, and is introduced by the second step in the chain rule:

$$\frac{\partial \mathcal{L}^{\text{euc}}}{\partial \hat{z}_m^{(1)}} = (\lambda_m \hat{z}^{(1)} - \hat{z}^{(2)}) \cdot \frac{\partial}{\partial \hat{z}_m^{(1)}}(\lambda_m \hat{z}^{(1)} - \hat{z}^{(2)}) \quad .$$

Based on this realization we see that this second step needs to be modified. To that end, we start with the desired derivative:

$$\frac{\partial \mathcal{L}^{\text{euc}}_{\text{iso}}}{\partial \hat{z}_m^{(1)}} = (\lambda_m \hat{z}^{(1)} - \hat{z}^{(2)}) \cdot \frac{\partial}{\partial \hat{z}_m^{(1)}}(\hat{z}^{(1)} - \hat{z}^{(2)}) \quad ,$$

and see that several loss functions are possible. The one we have reported in Eq. (3.15) we derived by applying an appropriate stop-grad while integrating:

$$\frac{\partial \mathcal{L}^{\text{euc}}_{\text{iso}}}{\partial \hat{z}_m^{(1)}} = (\hat{z}_m^{(1)} + \lambda_m \hat{z}^{(1)} - \hat{z}^{(2)} - \hat{z}_m^{(1)}) \cdot \frac{\partial}{\partial \hat{z}_m^{(1)}}(\hat{z}^{(1)} - \hat{z}^{(2)}) \quad .$$

to give:

$$\mathcal{L}^{\text{euc}}_{\text{iso}} = \tfrac{1}{2}\sum_m^M |\hat{z}_m^{(1)} - \text{SG}(\hat{z}_m^{(2)} + \hat{z}_m^{(1)} - \lambda_m \hat{z}_m^{(1)})|^2$$

Another alternative loss with the same desired isotropic learning dynamics, but using a different placement of the stop-gradient operators, is given by:

$$\mathcal{L}_{\text{iso}}^{\text{euc}} = \sum_m^M \text{SG}\left(\lambda_m \hat{z}_m^{(1)} - \hat{z}_m^{(2)}\right) \cdot \left(\hat{z}_m^{(1)} - \text{SG}(\hat{z}_m^{(2)})\right)$$

Cosine Similarity IsoLoss.

Since most practical SSL approaches rely on cosine similarity, which suffers from a similar anisotropy of the learning dynamics, we sought to find IsoLosses in this setting. With the same goal as above, we would like to arrive at the dynamics:

$$\frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = \eta \frac{\hat{z}_m^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\| \|\hat{\boldsymbol{z}}^{(2)}\|} - \eta \frac{\sum_k \lambda_k \hat{z}_k^{(1)} \hat{z}_k^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \cdot \lambda_m \hat{z}_m^{(1)}$$

starting from the cosine loss and corresponding dynamics:

$$\mathcal{L}^{\cos} = -\sum_m^M \frac{\lambda_m \hat{z}_m^{(1)} \,\text{SG}(\hat{z}_m^{(2)})}{\|D\hat{z}^{(1)}\| \|\text{SG}(\hat{z}^{(2)})\|} \tag{B.9}$$

$$\Rightarrow \frac{\mathrm{d}\hat{z}_m}{\mathrm{d}t} = \eta \frac{\lambda_m \hat{z}_m^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\| \|\hat{\boldsymbol{z}}^{(2)}\|} - \eta \frac{\sum_k \lambda_k \hat{z}_k^{(1)} \hat{z}_k^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \cdot \lambda_m^2 \hat{z}_m^{(1)} \quad . \tag{B.10}$$

The IsoLoss in this case can be derived by noting how λ_m arises in each of the two terms in Eq. (B.10), and engineering an alternative loss function corresponding to each term separately.

In the first term, λ_m arises from the partial derivative of the numerator $\lambda_m \hat{z}_m^{(1)} \text{SG}(\hat{z}_m^{(2)})$ in the original loss (Eq. (B.9)). This can be remediated by using $\hat{z}_m^{(1)} \text{SG}(\hat{z}_m^{(2)})$ as the numerator instead.

In the second term in Eq. (B.10), λ_m^2 arises from the partial derivative of $\|D\hat{\boldsymbol{z}}^{(1)}\| = \sqrt{\sum_k (\lambda_k \hat{z}_m^{(1)})^2}$ in the denominator. We can reduce λ_m^2 to λ_m by instead taking the partial derivative of $\|D^{1/2}\hat{\boldsymbol{z}}^{(1)}\| = \sqrt{\sum_k (\lambda_k^{1/2} \hat{z}_m^{(1)})^2}$.

Putting these insights together, we arrive at the desired partial derivative:

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{iso}}^{\cos}}{\partial \hat{z}_m^{(1)}} &= \frac{-1}{\|D\hat{\boldsymbol{z}}^{(1)}\| \|\hat{\boldsymbol{z}}^{(2)}\|} \cdot \frac{\partial \hat{z}_m^{(1)} \hat{z}_m^{(2)}}{\partial \hat{z}_m^{(1)}} + \frac{\sum_k \lambda_k \hat{z}_k^{(1)} \hat{z}_k^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \cdot \frac{1}{2} \frac{\partial \lambda_m (\hat{z}_m^{(1)})^2}{\partial \hat{z}_m^{(1)}} \\ &= \frac{-1}{\|D\hat{\boldsymbol{z}}^{(1)}\| \|\hat{\boldsymbol{z}}^{(2)}\|} \cdot \frac{\partial (\hat{\boldsymbol{z}}^{(1)})^\top \hat{\boldsymbol{z}}^{(2)}}{\partial \hat{z}_m^{(1)}} + \frac{\sum_k \lambda_k \hat{z}_k^{(1)} \hat{z}_k^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3 \|\hat{\boldsymbol{z}}^{(2)}\|} \cdot \frac{1}{2} \frac{\partial \|D^{1/2}\hat{\boldsymbol{z}}^{(1)}\|^2}{\partial \hat{z}_m^{(1)}} \quad ,\end{aligned}$$

and the integrated IsoLoss in eigenspace:

$$\mathcal{L}_{\text{iso}}^{\cos} = -(\hat{\boldsymbol{z}}^{(1)})^\top \text{SG}\left(\frac{\hat{\boldsymbol{z}}^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\|\|\hat{\boldsymbol{z}}^{(2)}\|}\right) + \frac{1}{2}\text{SG}\left(\frac{(D\hat{\boldsymbol{z}}^{(1)})^\top \hat{\boldsymbol{z}}^{(2)}}{\|D\hat{\boldsymbol{z}}^{(1)}\|^3\|\hat{\boldsymbol{z}}^{(2)}\|}\right)\|D^{1/2}\hat{\boldsymbol{z}}^{(1)}\|^2 \quad .$$

Rotating all terms back to the original space gives the desired IsoLoss for cosine similarity as reported (Eq. (3.17)):

$$\mathcal{L}_{\text{iso}} = -(\boldsymbol{z}^{(1)})^\top \text{SG}\left(\frac{\boldsymbol{z}^{(2)}}{\|W_{\text{P}}\boldsymbol{z}^{(1)}\|\|\boldsymbol{z}^{(2)}\|}\right) + \frac{1}{2}\text{SG}\left(\frac{(W_{\text{P}}\boldsymbol{z}^{(1)})^\top \boldsymbol{z}^{(2)}}{\|W_{\text{P}}\boldsymbol{z}^{(1)}\|^3\|\boldsymbol{z}^{(2)}\|}\right)\|W_{\text{P}}^{1/2}\boldsymbol{z}^{(1)}\|^2 \quad .$$

B.4 Experimental methods

Self-supervised pretraining. We used the CIFAR-10, CIFAR-100 [162], STL-10 [163], and TinyImageNet [164] datasets for self-supervised learning with a ResNet-18 [166] encoder and the SimCLR set of transformations [23]. We also adopted several modifications of ResNet-18 which have been proposed to deal with the low resolution of the images in these datasets [23]. The ResNet modifications comprise using 3×3 convolutional kernels instead of 7×7 kernels and skipping the first max-pooling operation. The Solo-learn library [165] also provides specialized sets of augmentations that work well for these datasets, which we adopted as well. The configurations we used for each dataset are summarized in Table B.2. We used BatchNorm in the backbone and the projector MLP in the hidden layer for all methods. For BYOL, we included BatchNorm also in the hidden layer of the predictor MLP.

We used a projection dimension of 256 for the projection MLP using one hidden layer with 4096 units, and the same architecture for the nonlinear predictor for the BYOL baseline. For networks using EMA target networks, we used the LARS optimizer with learning rate 1.0 whereas for networks without the EMA, we used stochastic gradient descent with momentum 0.9 and learning rate 0.1. Furthermore, we used a warmup period of 10 epochs for the learning rate followed by a cosine decay schedule and a batch size of 256. We also used a weight decay 4×10^{-4} for the closed-form predictor models and 10^{-5} for the nonlinear predictor models. For evaluation, we removed the projection MLP and used the embeddings at the pooled output of the ResNet convolutional layers following standard practice. For the EMA, we started with $\tau_{\text{base}} = 0.99$ and increased τ_{EMA} to 1 with a cosine schedule exactly following the configuration reported in [24]. For DirectPred, we used $\alpha = 0.5$ and $\tau = 0.998$ for the moving average estimate of the correlation matrix updated at every step.

Table B.2

	CIFAR-10	CIFAR-100	STL-10	TinyImageNet
Resolution	32×32	32×32	96×96	64×64
Kernel size	3×3	3×3	7×7	3×3
First max-pool	No	No	Yes	Yes
Blur	No	No	Yes	No

Linear evaluation protocol. We reported the held-out classification accuracy on the test sets for CIFAR-10/100 and STL-10, and the validation set for TinyImageNet, after online training of the gradient-isolated linear classifier on each labeled example in the training set during pretraining.

Compute resources All simulations were run on an in-house cluster consisting of 5 nodes with 4 V100 NVIDIA GPUs each, one node with 4 A100 NVIDIA GPUs, and one node with 8 A40 NVIDIA GPUs. Runs on CIFAR-10/100 took about 8 hours each, and the STL-10 and TinyImagenet runs took about 24 hours each.

Appendix C

Supplements to Chapter IV

C.1 Supplementary Figures

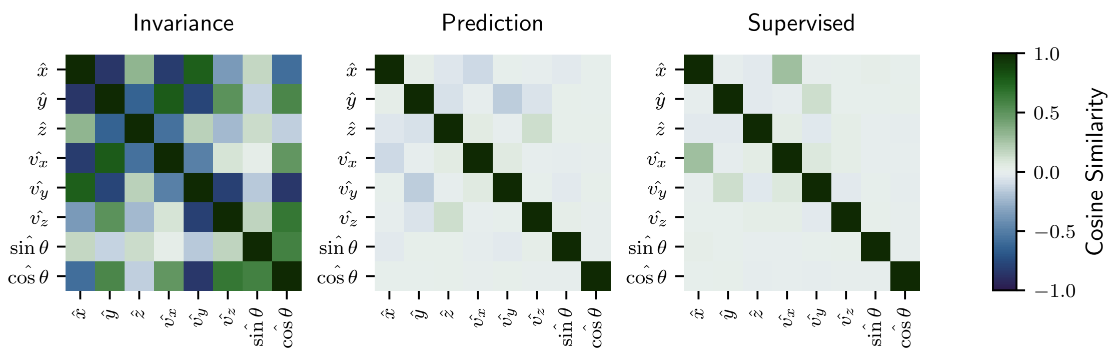


Figure C.1: **Orthogonality of learned representations of motion factors from videos of moving sprites.** Pairwise cosine similarities between the maximally predictive directions of the learned representations for the latent factors. The cosine similarity is calculated as the cosine of the angle between the two directions.

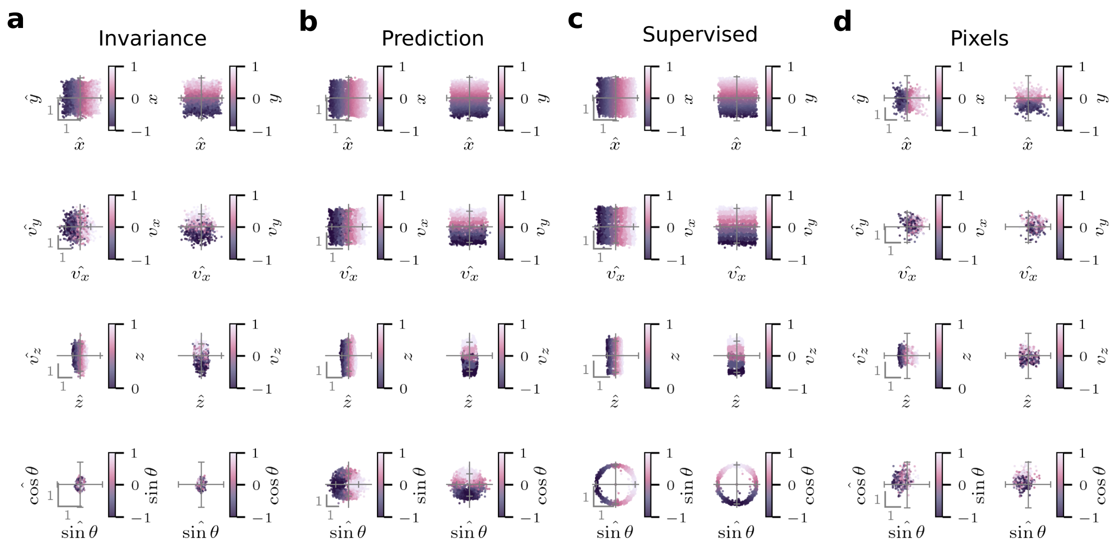


FIGURE C.2: **Projections of learned representations of motion factors from videos of moving sprites.** **(a)** Projections of the representations learned with the invariance objective along directions that are maximally predictive of pairs of latent factors (x, y), (v_x, v_y), (z, v_z) and $(\sin\theta, \cos\theta)$. For each pair, the projections are plotted twice, colored by the true values of the first and second latent factor each time. **(b)** Same as (a) but for the representations learned with the predictive objective. **(c)** Same as (a) but for the representations learned with supervised training. **(d)** Same as (a) but for the raw pixel values.

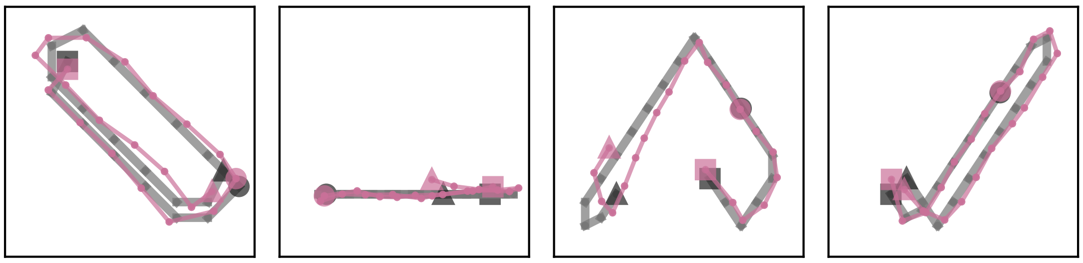


FIGURE C.3: **Simulated rollouts of sprite trajectories.** More examples of simulated rollouts of sprite trajectories using the learned predictive representations (same as Fig. 4.3b).

Supplementary Notes

C.2 Feedforward representations on the MNIST triplet sequence tasks

Readouts from the feedforward encoder showed low accuracy for the triplet identity for both methods (Supplementary Fig. C.4). This is expected because the feedforward encoder does not have access to the temporal context of the sequence. However, the feedforward readout was able to decode the digit labels with high accuracy for both methods, although the accuracy was slightly lower for invariance learning. This seems to be at odds with the low digit classification from the recurrent encoder seen for invariance learning (Fig. 4.4). However, the discrepancy is likely due to the fact that the recurrent representations are more directly optimized for the invariance objective and therefore collapse together digits from the same cluster. The feedforward encoder, on the other hand, could still retain some information about the individual digits, which may suffice for the relatively easy MNIST digit classification task. We validated this hypothesis by training the same network, but without the recurrent encoder, and found that the readout accuracies from the encoder now matched the recurrent encoder readouts (Supplementary Fig. C.4c).

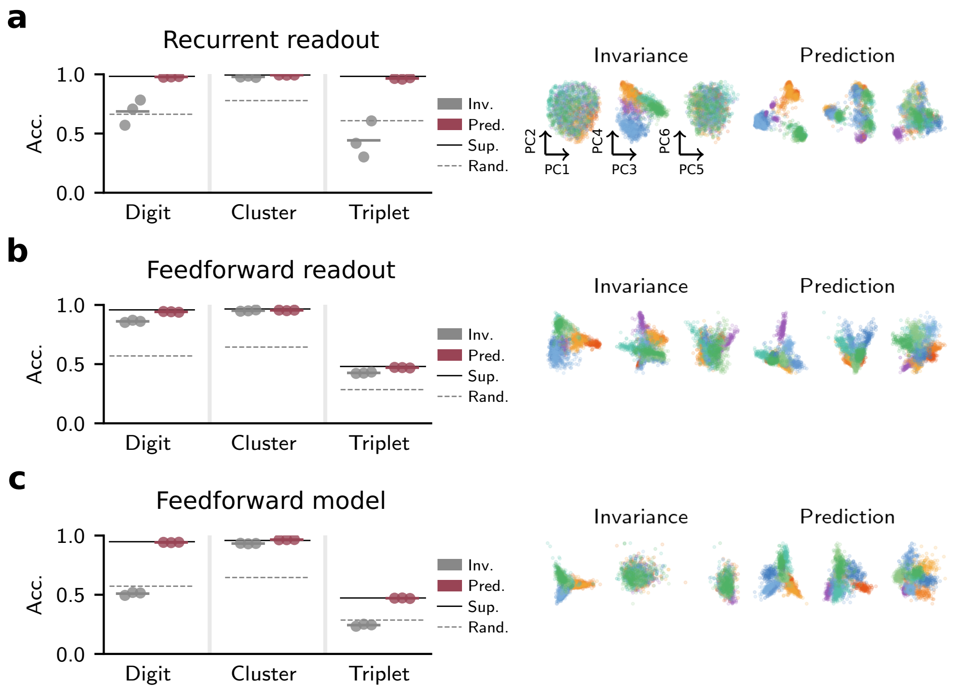


FIGURE C.4: **Readouts from recurrent and feedforward encoders.** **(a)** Same as Fig. 4.4b,c. **(b)** Same as (a) but for the feedforward encoder. **(c)** Same as (a) but for the feedforward encoder trained without a recurrent encoder.

Bibliography

[1] D. Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010.

[2] H. B. Barlow et al. “Possible principles underlying the transformation of sensory messages”. In: *Sensory communication* 1.01 (1961), pp. 217–233.

[3] D. O. Hebb. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

[4] T. V. Bliss and T. Lømo. “Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path”. In: *The Journal of physiology* 232.2 (1973), pp. 331–356.

[5] P. S. Eriksson et al. “Neurogenesis in the adult human hippocampus”. In: *Nature medicine* 4.11 (1998), pp. 1313–1317.

[6] G.-q. Bi and M.-m. Poo. “Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type”. In: *Journal of neuroscience* 18.24 (1998), pp. 10464–10472.

[7] W. Schultz, P. Dayan, and P. R. Montague. “A neural substrate of prediction and reward”. In: *Science* 275.5306 (1997), pp. 1593–1599.

[8] G. G. Turrigiano and S. B. Nelson. “Homeostatic plasticity in the developing nervous system”. In: *Nature reviews neuroscience* 5.2 (2004), pp. 97–107.

[9] J. M. Levenson and J. D. Sweatt. “Epigenetic mechanisms in memory formation”. In: *Nature Reviews Neuroscience* 6.2 (2005), pp. 108–118.

[10] R. S. Sutton and A. G. Barto. “Toward a modern theory of adaptive networks: expectation and prediction.” In: *Psychological review* 88.2 (1981), p. 135.

[11] E. L. Bienenstock, L. N. Cooper, and P. W. Munroe. “Theory of the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex”. In: *J. Neurosci.* 2 (1982), pp. 32–48. DOI: 10.1523/JNEUROSCI.02-01-00032.1982.

[12] J. J. Hopfield. “Neural networks and physical systems with emergent collective computational abilities.” In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[13] A. J. Bell and T. J. Sejnowski. “An information-maximization approach to blind separation and blind deconvolution”. In: *Neural computation* 7.6 (1995), pp. 1129–1159.

[14] W. Gerstner et al. “A neuronal learning rule for sub-millisecond temporal coding”. In: *Nature* 383.6595 (1996), pp. 76–78.

[15] R. P. Rao and D. H. Ballard. “Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects”. In: *Nature Neuroscience* 2.1 (1999), pp. 79–87. DOI: 10.1038/4580.

[16] L. Wiskott and T. J. Sejnowski. “Slow Feature Analysis: Unsupervised Learning of Invariances”. In: *Neural Computation* 14.4 (2002), pp. 715–770. DOI: 10.1162/089976602317318938.

[17] F. Zenke, E. J. Agnes, and W. Gerstner. “Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks”. In: *Nature Communications* 6 (2015), p. 6922. DOI: 10.1038/ncomms7922.

[18] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[19] A. H. Marblestone, G. Wayne, and K. P. Kording. “Toward an integration of deep learning and neuroscience”. In: *Frontiers in computational neuroscience* 10 (2016), p. 94.

[20] B. A. Richards et al. “A deep learning framework for neuroscience”. In: *Nature Neuroscience* 22.11 (2019), pp. 1761–1770. DOI: 10.1038/s41593-019-0520-2.

[21] A. Saxe, S. Nelli, and C. Summerfield. “If deep learning is the answer, what is the question?” In: *Nature Reviews Neuroscience* 22.1 (2021), pp. 55–67.

[22] A. van den Oord, Y. Li, and O. Vinyals. “Representation Learning with Contrastive Predictive Coding”. In: *arXiv preprint arXiv:1807.03748* (2019). DOI: 10.48550/arXiv.1807.03748v2.

[23] T. Chen et al. “A Simple Framework for Contrastive Learning of Visual Representations”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 1597–1607.

[24] J.-B. Grill et al. “Bootstrap your own latent-a new approach to self-supervised learning”. In: *Advances in neural information processing systems* 33 (2020), pp. 21271–21284.

[25] X. Chen and K. He. “Exploring simple siamese representation learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15750–15758.

[26] M. Caron et al. “Unsupervised learning of visual features by contrasting cluster assignments”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9912–9924.

[27] M. Assran et al. “Masked siamese networks for label-efficient learning”. In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXI*. Springer. 2022, pp. 456–473.

[28] A. Bardes, J. Ponce, and Y. LeCun. "VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning". In: *International Conference on Learning Representations*. 2022.

[29] M. Assran et al. "Self-supervised learning from images with a joint-embedding predictive architecture". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 15619–15629.

[30] H. Markram et al. "Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs". In: *Science* 275.5297 (1997), pp. 213–215.

[31] G. H. Seol et al. "Neuromodulators control the polarity of spike-timing-dependent synaptic plasticity". In: *Neuron* 55.6 (2007), pp. 919–929.

[32] A. L. Hodgkin and A. F. Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve". In: *The Journal of physiology* 117.4 (1952), p. 500.

[33] P. Dayan and L. F. Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press, 2005.

[34] W. Gerstner et al. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[35] W. Gerstner and W. M. Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[36] F. Zenke and S. Ganguli. "Superspike: Supervised learning in multilayer spiking neural networks". In: *Neural computation* 30.6 (2018), pp. 1514–1541. DOI: 10.1162/neco_a_01086.

[37] E. O. Neftci, H. Mostafa, and F. Zenke. "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks". In: *IEEE Signal Processing Magazine* 36.6 (2019), pp. 51–63.

[38] N. Frémaux and W. Gerstner. "Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-Factor Learning Rules". In: *Frontiers in Neural Circuits* 9 (2016). DOI: 10.3389/fncir.2015.00085.

[39] F. Zenke, G. Hennequin, and W. Gerstner. "Synaptic Plasticity in Neural Networks Needs Homeostasis with a Fast Rate Detector". In: *PLOS Computational Biology* 9.11 (2013), e1003330. DOI: 10.1371/journal.pcbi.1003330.

[40] F. Zenke and W. Gerstner. "Hebbian plasticity requires compensatory processes on multiple timescales". In: *Philosophical Transactions of the Royal Society B* 372.1715 (Mar. 2017), p. 20160259. DOI: 10.1098/rstb.2016.0259.

[41] E. Oja. "Simplified neuron model as a principal component analyzer". In: *Journal of Mathematical Biology* 15.3 (1982), pp. 267–273. DOI: 10.1007/BF00275687.

[42] C. Bredenberg and C. Savin. "Desiderata for normative models of synaptic plasticity". In: *Neural computation* 36.7 (2024), pp. 1245–1285.

[43] D. C. Knill and A. Pouget. "The Bayesian brain: the role of uncertainty in neural coding and computation". In: *TRENDS in Neurosciences* 27.12 (2004), pp. 712–719.

[44] K. Friston. "The free-energy principle: a unified brain theory?" In: *Nature reviews neuroscience* 11.2 (2010), pp. 127–138.

[45] A. Clark. "Whatever next? Predictive brains, situated agents, and the future of cognitive science". In: *Behavioral and brain sciences* 36.3 (2013), pp. 181–204.

[46] K. J. Friston et al. "Action and behavior: a free-energy formulation". In: *Biological cybernetics* 102 (2010), pp. 227–260.

[47] C. J. Gillon et al. "Responses to Pattern-Violating Visual Stimuli Evolve Differently Over Days in Somata and Distal Apical Dendrites". In: *Journal of Neuroscience* 44.5 (Jan. 31, 2024). DOI: 10.1523/JNEUROSCI.1009-23.2023. pmid: 37989593.

[48] D. Richter, T. C. Kietzmann, and F. P. de Lange. "High-Level Visual Prediction Errors in Early Visual Cortex". In: *PLOS Biology* 22.11 (Nov. 11, 2024), e3002829. DOI: 10.1371/journal.pbio.3002829.

[49] G. G. Parras et al. "Neurons along the auditory pathway exhibit a hierarchical organization of prediction error". In: *Nature communications* 8.1 (2017), p. 2148.

[50] D. M. Wolpert, R. C. Miall, and M. Kawato. "Internal models in the cerebellum". In: *Trends in cognitive sciences* 2.9 (1998), pp. 338–347.

[51] G. B. Keller, T. Bonhoeffer, and M. Hübener. "Sensorimotor mismatch signals in primary visual cortex of the behaving mouse". In: *Neuron* 74.5 (2012), pp. 809–815.

[52] S. Han and F. Helmchen. "Behavior-Relevant Top-down Cross-Modal Predictions in Mouse Neocortex". In: *Nature Neuroscience* (Jan. 4, 2024), pp. 1–11. DOI: 10.1038/s41593-023-01534-x.

[53] W. Lotter, G. Kreiman, and D. Cox. "Deep predictive coding networks for video prediction and unsupervised learning". In: *arXiv preprint arXiv:1605.08104* (2016). DOI: 10.48550/arXiv.1605.08104.

[54] Z. Straka, T. Svoboda, and M. Hoffmann. "PreCNet: Next-frame video prediction based on predictive coding". In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[55] R. P. Rane et al. "Prednet and predictive coding: A critical review". In: *Proceedings of the 2020 international conference on multimedia retrieval*. 2020, pp. 233–241.

[56] R. P. Rao. "A sensory–motor theory of the neocortex". In: *Nature neuroscience* 27.7 (2024), pp. 1221–1235.

[57] G. B. Keller and T. D. Mrsic-Flogel. "Predictive Processing: A Canonical Cortical Computation". In: *Neuron* 100.2 (2018), pp. 424–435. DOI: 10.1016/j.neuron.2018.10.003.

[58] R. Jordan and G. B. Keller. "Opposing influence of top-down and bottom-up input on excitatory layer 2/3 neurons in mouse primary visual cortex". In: *Neuron* 108.6 (2020), pp. 1194–1206.

[59] S. Furutachi et al. “Cooperative thalamocortical circuit mechanism for sensory prediction errors”. In: *Nature* 633.8029 (2024), pp. 398–406.

[60] J. Achiam et al. “Gpt-4 technical report”. In: *arXiv preprint arXiv:2303.08774* (2023). DOI: 10.48550/arXiv.2303.08774.

[61] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133.

[62] F. Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).

[64] A. Graves and N. Jaitly. “Towards end-to-end speech recognition with recurrent neural networks”. In: *International conference on machine learning*. PMLR. 2014, pp. 1764–1772.

[65] V. Nair and G. E. Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[66] Y. Bengio, A. Courville, and P. Vincent. “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[67] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[68] S. Sardi et al. “New types of experiments reveal that a neuron functions as multiple independent threshold units”. In: *Scientific reports* 7.1 (2017), p. 18036.

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

[70] A. Nøkland. “Direct feedback alignment provides learning in deep neural networks”. In: *Advances in neural information processing systems* 29 (2016).

[71] T. P. Lillicrap et al. “Backpropagation and the brain”. In: *Nature Reviews Neuroscience* (2020). Publisher: Nature Publishing Group, pp. 1–12. DOI: 10.1038/s41583-020-0277-3.

[72] J. C. Whittington and R. Bogacz. “An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity”. In: *Neural computation* 29.5 (2017), pp. 1229–1262.

[73] G. I. Parisi et al. “Continual lifelong learning with neural networks: A review”. In: *Neural networks* 113 (2019), pp. 54–71.

[74] R. Geirhos et al. “Shortcut learning in deep neural networks”. In: *Nature Machine Intelligence* 2.11 (2020), pp. 665–673.

[75] B. M. Lake et al. “Building machines that learn and think like people”. In: *Behavioral and brain sciences* 40 (2017), e253.

[76] S. Chung and L. F. Abbott. “Neural population geometry: An approach for understanding biological and artificial neural networks”. In: *Current Opinion in Neurobiology*. Computational Neuroscience 70 (2021), pp. 137–144. DOI: 10.1016/j.conb.2021.10.010.

[77] B. A. Richards and K. P. Kording. “The study of plasticity has always been about gradients”. In: *The Journal of Physiology* 601.15 (2023), pp. 3141–3149.

[78] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.

[79] Y. Bengio. “Deep learning of representations for unsupervised and transfer learning”. In: *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings. 2012, pp. 17–36.

[80] H. B. Barlow. “Unsupervised learning”. In: *Neural computation* 1.3 (1989), pp. 295–311.

[81] D. Pathak et al. “Context encoders: Feature learning by inpainting”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

[82] K. He et al. “Masked Autoencoders Are Scalable Vision Learners”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022), pp. 15979–15988. DOI: 10.1109/CVPR52688.2022.01553.

[83] R. Zhang, P. Isola, and A. A. Efros. “Colorful image colorization”. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer. 2016, pp. 649–666.

[84] A. Bardes et al. “Revisiting Feature Prediction for Learning Visual Representations from Video”. In: *arXiv preprint arXiv:2404.08471* (2024). DOI: 10.48550/arXiv.2404.08471.

[85] Z. Wu et al. “Unsupervised feature learning via non-parametric instance discrimination”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3733–3742.

[86] M. Mathieu, C. Couprie, and Y. LeCun. “Deep multi-scale video prediction beyond mean square error”. In: *arXiv preprint arXiv:1511.05440* (2015). DOI: 10.48550/arXiv.1511.05440.

[87] K. Drozdov, R. Shwartz-Ziv, and Y. LeCun. “Video Representation Learning with Joint-Embedding Predictive Architectures”. In: *arXiv preprint arXiv:2412.10925* (2024). DOI: 10.48550/arXiv.2412.10925.

[88] J. Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the 2019 conference of the North American chapter*

of the association for computational linguistics: human language technologies, volume 1 (long and short papers). 2019, pp. 4171–4186.

[89] R. Balestriero et al. "A Cookbook of Self-Supervised Learning". In: *arXiv preprint arXiv:2304.12210* (2023). DOI: 10.48550/arXiv.2304.12210.

[90] J. Zbontar et al. "Barlow Twins: Self-Supervised Learning via Redundancy Reduction". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 12310–12320.

[91] A. Ermolov et al. "Whitening for self-supervised representation learning". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 3015–3024.

[92] T. Yerxa et al. "Learning efficient coding of natural images with maximum manifold capacity representations". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 24103–24128.

[93] N. Tomasev et al. "Pushing the limits of self-supervised ResNets: Can we outperform supervised learning without labels on ImageNet?" In: *arXiv preprint arXiv:2201.05119* (2022). DOI: 10.48550/arXiv.2201.05119.

[94] Y. LeCun. "A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27". In: *Open Review* 62.1 (2022), pp. 1–62.

[95] D. Ha and J. Schmidhuber. "World models". In: *arXiv preprint arXiv:1803.10122* (2018). DOI: 10.48550/arXiv.1803.10122.

[96] D. Hafner et al. "Mastering atari with discrete world models". In: *arXiv preprint arXiv:2010.02193* (2020). DOI: 10.48550/arXiv.2010.02193.

[97] H. Ghaemi, E. B. Muller, and S. Bakhtiari. "Seq-JEPA: Autoregressive Predictive Learning of Invariant-Equivariant World Models". In: *NeurIPS 2024 Workshop: Self-Supervised Learning - Theory and Practice*. 2024. URL: https://openreview.net/forum?id=MO1OLAKcsJ.

[98] R. Balestriero and Y. LeCun. "Learning by Reconstruction Produces Uninformative Features For Perception". In: *arXiv preprint arXiv:2402.11337* (2024). DOI: 10.48550/arXiv.2402.11337.

[99] S. Richthofer and L. Wiskott. "Predictable feature analysis". In: *2015 IEEE 14th international Conference on machine learning and applications (ICMLA)*. IEEE. 2015, pp. 190–196.

[100] N. Li and J. J. DiCarlo. "Unsupervised Natural Experience Rapidly Alters Invariant Object Representation in Visual Cortex". In: *Science* 321.5895 (2008), pp. 1502–1507. DOI: 10.1126/science.1160028.

[101] G. Matteucci and D. Zoccolan. "Unsupervised experience with temporal continuity of the visual environment is causally involved in the development of V1 complex cells". In: *Science Advances* 6.22 (2020), eaba3742. DOI: 10.1126/sciadv.aba3742.

[102] Y. Chen et al. "Predictive sequence learning in the hippocampal formation". In: *Neuron* 112.15 (2024), pp. 2645–2658.

[103] M. S. Halvagal and F. Zenke. "The combination of Hebbian and predictive plasticity learns invariant object representations in deep sensory networks". In: *Nature neuroscience* 26.11 (2023), pp. 1906–1915.

[104] J. J. DiCarlo, D. Zoccolan, and N. C. Rust. "How Does the Brain Solve Visual Object Recognition?" In: *Neuron* 73.3 (2012), pp. 415–434. DOI: 10.1016/j.neuron.2012.01.010.

[105] D. L. K. Yamins et al. "Performance-optimized hierarchical models predict neural responses in higher visual cortex". In: *Proceedings of the National Academy of Sciences* 111.23 (2014), pp. 8619–8624. DOI: 10.1073/pnas.1403112111.

[106] A. Nayebi et al. "Explaining heterogeneity in medial entorhinal cortex with task-driven neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12167–12179.

[107] J. Guerguiev, T. P. Lillicrap, and B. A. Richards. "Towards deep learning with segregated dendrites". In: *eLife* 6 (2017), e22901. DOI: 10.7554/eLife.22901.

[108] J. Sacramento et al. "Dendritic cortical microcircuits approximate the backpropagation algorithm". In: *Advances in Neural Information Processing Systems* 31 (2018).

[109] A. Payeur et al. "Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits". In: *Nature Neuroscience* 24.7 (2021), pp. 1010–1019. DOI: 10.1038/s41593-021-00857-x.

[110] L. N. Cooper and M. F. Bear. "The BCM theory of synapse modification at 30: interaction of theory with experiment". In: *Nature Reviews Neuroscience* 13.11 (2012), pp. 798–810. DOI: 10.1038/nrn3353.

[111] Y. Singer et al. "Sensory cortex is optimized for prediction of future input". In: *eLife* 7 (June 2018). Ed. by J. L. Gallant and S. Kastner, e31557. DOI: 10.7554/eLife.31557.

[112] C. J. Gillon et al. "Learning from unexpected events in the neocortical microcircuit". In: *bioRxiv* (Jan. 2021), p. 2021.01.15.426915. DOI: 10.1101/2021.01.15.426915.

[113] H. Sprekeler, C. Michaelis, and L. Wiskott. "Slowness: An Objective for Spike-Timing–Dependent Plasticity?" In: *PLoS Comput Biol* 3.6 (2007), e112. DOI: 10.1371/journal.pcbi.0030112.

[114] B. Illing et al. "Local plasticity rules can learn deep representations using self-supervised contrastive predictions". In: *Advances in Neural Information Processing Systems* 34 (2021).

[115] L. Kusmierz, T. Isomura, and T. Toyoizumi. "Learning with three factors: modulating Hebbian plasticity with errors". In: *Current Opinion in Neurobiology* 46 (2017), pp. 170–177. DOI: 10.1016/j.conb.2017.08.020.

[116] P. J. Sjöström, G. G. Turrigiano, and S. B. Nelson. "Rate, Timing, and Cooperativity Jointly Determine Cortical Synaptic Plasticity". In: *Neuron* 32.6 (2001), pp. 1149–1164. DOI: 10.1016/S0896-6273(01)00542-6.

[117] D. E. Feldman. "The Spike-Timing Dependence of Plasticity". In: *Neuron* 75.4 (2012), pp. 556–571. DOI: 10.1016/j.neuron.2012.08.001.

[118] S. Löwe, P. O'Connor, and B. S. Veeling. "Putting an end to end-to-end: Gradient-isolated learning of representations". In: *Advances in Neural Information Processing Systems* 32.NeurIPS (2019).

[119] T. Miconi. "Multi-layer Hebbian networks with modern deep learning frameworks". In: *arXiv preprint arXiv:2107.01729* (2021). DOI: 10.48550/arXiv.2107.01729.

[120] A. Artola, S. Bröcher, and W. Singer. "Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex". In: *Nature* 347.6288 (1990), pp. 69–72. DOI: 10.1038/347069a0.

[121] W. C. Abraham. "Metaplasticity: Tuning synapses and networks for plasticity". In: *Nature Reviews Neuroscience* 9.5 (2008), pp. 387–387. DOI: 10.1038/nrn2356.

[122] S. Lim et al. "Inferring learning rules from distributions of firing rates in cortical neurons". In: *Nature Neuroscience* 18.12 (2015). Number: 12 Publisher: Nature Publishing Group, pp. 1804–1810. DOI: 10.1038/nn.4158.

[123] P. Földiak. "Forming sparse representations by local anti-Hebbian learning". In: *Biological cybernetics* 64.2 (1990), pp. 165–170. DOI: 10.1007/BF02331346.

[124] T. P. Vogels et al. "Inhibitory Plasticity Balances Excitation and Inhibition in Sensory Pathways and Memory Networks". In: *Science* 334.6062 (2011), pp. 1569–1573. DOI: 10.1126/science.1211095.

[125] P. D. King, J. Zylberberg, and M. R. DeWeese. "Inhibitory Interneurons Decorrelate Excitatory Cells to Drive Sparse Code Formation in a Spiking Model of V1". In: *The Journal of Neuroscience* 33.13 (2013), pp. 5475–5485. DOI: 10.1523/JNEUROSCI.4188-12.2013.

[126] D. Lipshutz et al. "A biologically plausible neural network for slow feature analysis". In: *Advances in neural information processing systems* 33 (2020), pp. 14986–14996.

[127] L. Jing et al. "Understanding Dimensional Collapse in Contrastive Self-supervised Learning". In: *International Conference on Learning Representations* (2022).

[128] H. Kim and A. Mnih. "Disentangling by Factorising". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by A. K. Jennifer Dy. Vol. 80. PMLR, July 2018, pp. 2649–2658.

[129] Y. Inglebert et al. "Synaptic plasticity rules with physiological calcium levels". In: *Proceedings of the National Academy of Sciences* (2020). Publisher: National Academy of Sciences Section: Biological Sciences. DOI: 10.1073/pnas.2013663117.

[130] H. Z. Shouval, M. F. Bear, and L. N. Cooper. “A Unified Model of NMDA Receptor-Dependent Bidirectional Synaptic Plasticity”. In: *Proceedings of the National Academy of Sciences* 99.16 (June 2002), pp. 10831–10836. DOI: 10.1073/pnas.152343099.

[131] J.-P. Pfister and W. Gerstner. “Triplets of Spikes in a Model of Spike Timing-Dependent Plasticity”. In: *The Journal of Neuroscience* 26.38 (2006), pp. 9673–9682. DOI: 10.1523/JNEUROSCI.1425-06.2006.

[132] C. Clopath et al. “Connectivity reflects coding: a model of voltage-based STDP with homeostasis”. In: *Nature Neuroscience* 13.3 (2010), pp. 344–352. DOI: 10.1038/nn.2479.

[133] J. Gjorgjieva et al. “A triplet spike-timing–dependent plasticity model generalizes the Bienenstock–Cooper–Munro rule to higher-order spatiotemporal correlations”. In: *Proceedings of the National Academy of Sciences* 108.48 (2011), pp. 19383–19388. DOI: 10.1073/pnas.1105933108.

[134] T. Toyoizumi et al. “Modeling the Dynamic Interaction of Hebbian and Homeostatic Plasticity”. In: *Neuron* 84.2 (2014), pp. 497–510. DOI: 10.1016/j.neuron.2014.09.036.

[135] M. Graupner and N. Brunel. “Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location”. In: *Proceedings of the National Academy of Sciences* 109.10 (2012), pp. 3991–3996. DOI: 10.1073/pnas.1109359109.

[136] G. Hennequin, E. J. Agnes, and T. P. Vogels. “Inhibitory Plasticity: Balance, Control, and Codependence”. In: *Annual Review of Neuroscience* 40.1 (2017), pp. 557–579. DOI: 10.1146/annurev-neuro-072116-031005.

[137] E. T. Rolls and S. M. Stringer. “Invariant visual object recognition: A model, with lighting invariance”. In: *Journal of Physiology-Paris*. Theoretical and Computational Neuroscience: Understanding Brain Functions 100.1 (July 2006), pp. 43–62. DOI: 10.1016/j.jphysparis.2006.09.004.

[138] R. Wyss, P. König, and P. F. M. J. Verschure. “A Model of the Ventral Visual System Based on Temporal Stability and Local Memory”. en. In: *PLOS Biology* 4.5 (Apr. 2006), e120. DOI: 10.1371/journal.pbio.0040120.

[139] Y. Li et al. “Self-supervised learning with kernel dependence maximization”. In: *Advances in Neural Information Processing Systems* 34 (2021).

[140] J. Mehrer et al. “An ecologically motivated image dataset for deep learning yields better models of human vision”. In: *Proceedings of the National Academy of Sciences* 118.8 (2021). Publisher: National Academy of Sciences Section: Biological Sciences. DOI: 10.1073/pnas.2011417118.

[141] M. S. Halvagal and F. Zenke. *fmi-basel/latent-predictive-learning: LPL*. Version v1.0. Aug. 2023. DOI: 10.5281/zenodo.8252888. URL: https://doi.org/10.5281/zenodo.8252888.

[142] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[143] A. Litwin-Kumar et al. "Optimal Degrees of Synaptic Connectivity". In: *Neuron* 93.5 (2017), 1153–1164.e7. DOI: 10.1016/j.neuron.2017.01.030.

[144] M. S. Halvagal, A. Laborieux, and F. Zenke. "Implicit variance regularization in non-contrastive SSL". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 63409–63436.

[145] M. Caron et al. "Emerging properties in self-supervised vision transformers". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 9650–9660.

[146] Y. Tian et al. "What makes for good views for contrastive learning?" In: *Advances in neural information processing systems* 33 (2020), pp. 6827–6839.

[147] J. Bromley et al. "Signature verification using a" siamese" time delay neural network". In: *Advances in neural information processing systems* 6 (1993).

[148] T. Wang and P. Isola. "Understanding contrastive representation learning through alignment and uniformity on the hypersphere". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9929–9939.

[149] T. Chen, C. Luo, and L. Li. "Intriguing properties of contrastive losses". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 11834–11845.

[150] X. Weng et al. "An investigation into whitening loss for self-supervised learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 29748–29760.

[151] Y. Tian, X. Chen, and S. Ganguli. "Understanding self-supervised learning dynamics without contrastive pairs". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10268–10278.

[152] X. Wang et al. "Towards demystifying representation learning with non-contrastive self-supervision". In: *arXiv preprint arXiv:2110.04947* (2021).

[153] K.-J. Liu, M. Suganuma, and T. Okatani. "Bridging the gap from asymmetry tricks to decorrelation principles in non-contrastive self-supervised learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 19824–19835.

[154] C. Tao et al. "Exploring the equivalence of siamese self-supervised learning via a unified gradient framework". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 14431–14440.

[155] P. H. Richemond et al. "The Edge of Orthogonality: A Simple View of What Makes BYOL Tick". In: *arXiv preprint arXiv:2302.04817* (2023).

[156] P. H. Richemond et al. "Byol works even without batch statistics". In: *arXiv preprint arXiv:2010.10241* (2020).

[157] X. Wang et al. “On the Importance of Asymmetry for Siamese Representation Learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16570–16579.

[158] C. Zhang et al. “How does simsiam avoid collapse without negative samples? a unified understanding with self-supervised contrastive learning”. In: *arXiv preprint arXiv:2203.16262* (2022).

[159] A. Jacot, F. Gabriel, and C. Hongler. “Neural tangent kernel: Convergence and generalization in neural networks”. In: *Advances in neural information processing systems* 31 (2018).

[160] J. Lee et al. “Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent”. en. In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 8572–8583.

[161] A. M. Saxe, J. L. McClelland, and S. Ganguli. “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks”. In: *arXiv:1312.6120* (Dec. 2013). DOI: 10.48550/arXiv.1312.6120.

[162] A. Krizhevsky, G. Hinton, et al. “Learning multiple layers of features from tiny images”. MA thesis. University of Toronto, 2009.

[163] A. Coates, A. Ng, and H. Lee. “An Analysis of Single-Layer Networks in Unsupervised Feature Learning”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by G. Gordon, D. Dunson, and M. Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 215–223.

[164] Y. Le and X. Yang. “Tiny imagenet visual recognition challenge”. In: *CS 231N* 7.7 (2015), p. 3.

[165] V. G. T. da Costa et al. “solo-learn: A Library of Self-supervised Methods for Visual Representation Learning”. In: *Journal of Machine Learning Research* 23.56 (2022), pp. 1–6.

[166] K. He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[167] J. B. Simon et al. “On the stepwise nature of self-supervised learning”. In: *arXiv preprint arXiv:2303.15438* (2023).

[168] Q. Garrido et al. “On the duality between contrastive and non-contrastive self-supervised learning”. In: *arXiv preprint arXiv:2206.02574* (2022).

[169] R. Balestriero and Y. LeCun. “Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods”. In: *arXiv preprint arXiv:2205.11508* (2022).

[170] Y. Dubois et al. “Improving self-supervised learning by characterizing idealized representations”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 11279–11296.

[171] A. Pokle et al. “Contrasting the landscape of contrastive and non-contrastive learning”. In: *arXiv preprint arXiv:2203.15702* (2022).

[172] W. J. Johnston and S. Fusi. “Abstract representations emerge naturally in neural networks trained to perform multiple tasks”. In: *Nature Communications* 14.1 (2023), p. 1040.

[173] K. Kermani Nejad et al. “Self-supervised predictive learning accounts for layer-specific cortical observations”. In: *BioRxiv* (2024), pp. 2024–04.

[174] J. S. Prince, G. A. Alvarez, and T. Konkle. “Contrastive learning explains the emergence and function of visual category-selective regions”. In: *Science Advances* 10.39 (2024), eadl1776.

[175] S. Bernardi et al. “The Geometry of Abstraction in the Hippocampus and Prefrontal Cortex”. In: *Cell* 183.4 (Nov. 2020), 954–967.e21. DOI: 10.1016/j.cell.2020.09.031.

[176] V. Panayotov et al. “Librispeech: An ASR corpus based on public domain audio books”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 5206–5210. DOI: 10.1109/ICASSP.2015.7178964.

[177] I. K. Brunec and I. Momennejad. “Predictive Representations in Hippocampal and Prefrontal Hierarchies”. In: *Journal of Neuroscience* 42.2 (Jan. 12, 2022), pp. 299–312. DOI: 10.1523/JNEUROSCI.1327-21.2021. pmid: 34799416.

[178] B. H. Price et al. “Expectation violations produce error signals in mouse V1”. In: *Cerebral Cortex* 33.13 (2023), pp. 8803–8820.

[179] M. F. Tang et al. “Expectation violations enhance neuronal encoding of sensory information in mouse primary visual cortex”. In: *Nature Communications* 14.1 (2023), p. 1196.

[180] O. J. Hénaff, R. L. Goris, and E. P. Simoncelli. “Perceptual straightening of natural videos”. In: *Nature neuroscience* 22.6 (2019), pp. 984–991.

[181] O. J. Hénaff et al. “Primary visual cortex straightens natural video trajectories”. In: *Nature communications* 12.1 (2021), p. 5982.

[182] N. T. Markov et al. “Cortical High-Density Counterstream Architectures”. In: *Science (New York, N.Y.)* 342.6158 (Nov. 1, 2013), p. 1238406. DOI: 10.1126/science.1238406. pmid: 24179228.

[183] D. Levenstein et al. *Sequential Predictive Learning Is a Unifying Theory for Hippocampal Representation and Replay*. Apr. 29, 2024. DOI: 10.1101/2024.04.28.591528. Pre-published.

[184] A. Bardes, J. Ponce, and Y. LeCun. “Mc-jepa: A joint-embedding predictive architecture for self-supervised learning of motion and content features”. In: *arXiv preprint arXiv:2307.12698* (2023). DOI: 10.48550/arXiv.2307.12698.

[185] E. Boonstra and H. Slagter. *Learning to Predict Based on Self- versus Externally Induced Prediction Violations: A Direct Comparison Using a Bayesian Inference*

Modelling Approach. preprint. Neuroscience, Nov. 15, 2022. DOI: 10.1101/2022.11.15.516578.

[186] J. C. Whittington and R. Bogacz. “Theories of error back-propagation in the brain”. In: *Trends in cognitive sciences* 23.3 (2019), pp. 235–250.

[187] A. Meulemans et al. “Credit assignment in neural networks through deep feedback control”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 4674–4687.

[188] J. Rossbroich and F. Zenke. “Dis-inhibitory neuronal circuits can control the sign of synaptic plasticity”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 64059–64082.

[189] Y. Song et al. “Inferring neural activity before plasticity as a foundation for learning beyond backpropagation”. In: *Nature Neuroscience* 27.2 (2024), pp. 348–358.

[190] E. Piasini et al. “Temporal Stability of Stimulus Representation Increases along Rodent Visual Cortical Hierarchies”. In: *Nature Communications* 12.1 (July 21, 2021), p. 4448. DOI: 10.1038/s41467-021-24456-3.

[191] D. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv preprint arXiv:1412.6980* (2014). DOI: 10.48550/arXiv.1412.6980.

[192] Y. LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.