# Understanding cortical computation through the lens of joint-embedding predictive architectures

Ashena Gorgan Mohammadi[1,2,*], Manu Srinath Halvagal[1,2,3,*], and Friedemann Zenke[1,2,†]

[1]Friedrich Miescher Institute for Biomedical Research, 4056 Basel, Switzerland
[2]Faculty of Science, University of Basel, 4033 Basel, Switzerland
[3]Department of Physics, Harvard University, Cambridge MA, United States
[*]These authors contributed equally to this work.
[†]Correspondence: friedemann.zenke@fmi.ch

**Abstract**

Tracking prey or recognizing a lurking predator is as crucial for survival as anticipating their actions. To guide behavior, the brain must extract information about object identities and their dynamics from entangled sensory inputs. How it accomplishes this feat remains an open question. Predictive coding theories propose that this ability arises by comparing predicted sensory signals with actual inputs and reducing the associated prediction errors. However, existing models rely on generative architectures that compute prediction errors directly in the input space, which is hard to reconcile with neural anatomy. Here, we develop a theory that avoids this issue by operating solely on internal representations. Specifically, we introduce recurrent predictive learning (RPL), a recurrent joint-embedding predictive architecture inspired by self-supervised machine learning, that learns abstract representations of object identity and their dynamics and predicts future object motion. Crucially, the model learns sequence representations that resemble successor-like representations observed in the primary visual cortex of humans. The model also develops abstract sequence representations comparable to those reported in the macaque prefrontal cortex. Finally, we outline how RPL's modular feedforward-recurrent organization could map onto canonical cortical microcircuits as a plausible model of cortical representation learning. Our work establishes a circuit-centric theory framework that unifies predictive processing with self-supervised learning, providing a fresh perspective on how the brain acquires an internal model of the world.

## Introduction

It takes us a split second to recognize an oncoming car at an intersection, to anticipate the trajectory of a ball passed to us, or to realize that we are being chased by a dog in the park. Importantly, we have no difficulty assessing these situations and responding to them, despite never having seen this particular car, ball, or dog. To accomplish this feat, our nervous system faces a twofold challenge. On the one hand, it must detect abstract patterns from sensory input and form invariant object representations, allowing us to recognize a dog. This task is hindered by the fact that distinct sensory stimuli are often entangled, requiring complex hierarchical networks to untangle them (Fig. 1). On the other hand, our brain must also infer equivariant

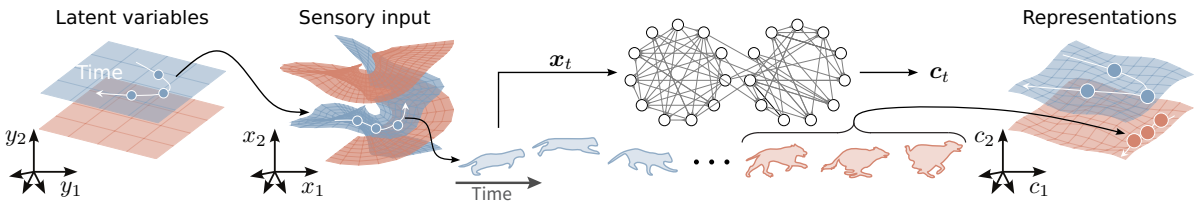


Figure 1: **Extracting latent variables from sensory stimuli.** Animals and humans must infer latent variables $\boldsymbol{y}$ such as object identity, orientation, and velocity from ambiguous and noisy sensory stimuli. This task is computationally demanding because sensory stimuli $\boldsymbol{x}$ are typically entangled (left). Untangling these input manifolds to approximate the underlying latent variables (right) requires multi-layer networks (middle). How our brains achieve this untangling remains a central question in neuroscience.

features of the dog's dynamics, such as its position, velocity, and angle of attack. After all, our actions depend on whether it is moving toward or away from us. Only having both invariant and equivariant representations allows us to respond appropriately in novel situations. While a plethora of experiments have shown that sensory cortices encode this information [1–9], the mechanisms underlying the plasticity that forms these representations remain elusive.

An influential idea of how representation learning works in the cortex is predictive processing [10, 11]. It proposes that the brain learns an internal model of the world, allowing it to anticipate incoming sensory input. According to this view, cortical networks generate predictions about expected sensory stimuli and compute the difference between these predictions and the actual input, so-called prediction errors. Minimizing prediction errors enables the inference of latent variables, learning internal representations, and bootstrapping the internal model [10, 12–20]. While a compelling idea, existing models of predictive processing have one major drawback: they are formulated as generative models that construct predictions in the input space. In other words, a model using this principle needs a dedicated decoder network to predict, for instance, the image of an incoming dog, which is then compared to the actual sensory experience, rather than comparing internal representations directly. This subtle distinction has profound consequences for the required neuronal architecture, rendering existing models largely inconsistent with neural anatomy. For example, there are no back-projections from the visual cortex to the retina, which would be required to compute the prediction errors in "pixel space" for learning a generative model. Consequently, whether and how we can reconcile predictive processing with neural anatomy remains a central outstanding question in circuit neuroscience [11, 21–23].

A possible alternative that seems more aligned with neural anatomy is to abandon the notion of a generative model and instead compute prediction errors solely from internal representations, a strategy employed by recent work in self-supervised machine learning [24]. While dispensing with the need for a decoder network, this strategy comes with its own challenge: "representational collapse," in which the network implements trivial solutions, such as a constant representation, which is perfectly predictable but also utterly useless from an information-processing point of view. Avoiding representational collapse requires additional variance regularization techniques [25–27], biologically implausible negative samples [28, 29], or specific network architectures such as a joint-embedding predictive architecture (JEPA) [30].

A JEPA avoids collapse by computing prediction errors in the representation space using a dedicated predictor network [31–36]. In addition to being more closely aligned with neural anatomy, this strategy offers another distinct advantage. Since it does not rely on a reconstruction objective, it allows focusing network resources on abstract features, like "the dog at my six o'clock is about to pounce," rather than low-level sensory details, such as the particular

sheen of the dog's fur around the ears, which may be irrelevant to the situation at hand [37, 38]. Nevertheless, most JEPAs studied in machine learning are poor models of neurobiology because they use implausible architectural elements, such as transformers, and learning strategies, such as token masking, while focusing mainly on static images, hand-crafted transformations, and invariant representations geared towards object classification. This raises the question of whether a JEPA broadly consistent with cortical circuits exists. A minimal model would have to be based on recurrent neural networks rather than transformers, learn from streaming sensory data without masking, extract invariant and equivariant features, and remain consistent with experimental data.

Here, we introduce such a model. We refer to it as recurrent predictive learning (RPL) thanks to its specific circuit layout. RPL learns linearly decodable object representations, including their associated motion variables, from videos of moving objects without requiring any labels. As we will see, it yields abstract representations spanning multiple timescales, consistent with representations in the macaque prefrontal cortex (PFC) [39]. In addition to representation learning, RPL learns a dynamic world model in the latent space, which represents future stimuli, similar to human V1 [40]. Importantly, RPL comprises distinct feed-forward and recurrent components, each playing a dedicated functional role. In a hierarchical setting, in which prediction errors are computed in local prediction circuits, this organization allows for a direct interpretation in terms of the cortical microcircuit. In summary, we introduce a novel network architecture and learning strategy that extends the notion of predictive processing to internal representation prediction, providing a fresh circuit-centric approach to understanding the computational principles of the cortex.

## Results

To study the putative mechanisms by which sensory networks extract latent variables, such as invariant object identity, and equivariant latent variables, such as position, velocity, and orientation, we simulated deep neural network models with a specific structure. Broadly speaking, these networks consist of three functionally distinct components (Fig. 2a). First, a feed-forward "encoder" network nonlinearly encodes sensory stimuli into the intermediate embedding $\boldsymbol{z}$ with minimal to no history dependence. Second, a recurrent neural network, the "integrator," successively integrates these embeddings to form the internal representations $\boldsymbol{c}$. Third, the predictor network $f$ takes the representation $\boldsymbol{c}_t$ at time $t$ to predict the embedding $\boldsymbol{z}_{t+\Delta}$ an instant later. Finally, the resulting prediction error

$$\boldsymbol{e}_{t+\Delta} = \underbrace{\boldsymbol{z}_{t+\Delta}}_{\text{embedding}} - \underbrace{f(\boldsymbol{c}_t)}_{\text{prediction}} \tag{1}$$

which serves as an instructive learning signal to optimize the model parameters (Methods). Importantly, this unsupervised learning signal is fully specified in the internal neuronal representation space, localized in time, and can be computed without an explicit prediction or reconstruction of $\boldsymbol{x}$ in the input space. We collectively refer to network models trained by minimizing the above prediction error as RPL in the following.

To evaluate RPL's effectiveness in forming untangled representations of both invariant and dynamic variables from object motion sequences, we constructed a synthetic dataset of moving animal images (Fig. 2b; Methods) with *known* latent variables. To that end, we generated short sequences of one of eight cartoon animal images moving smoothly with a randomly chosen linear and angular velocity within a square arena with elastic collisions at the boundaries (Fig. 2c). To obtain individual video frames, we sampled the trajectory at discrete intervals, placed each

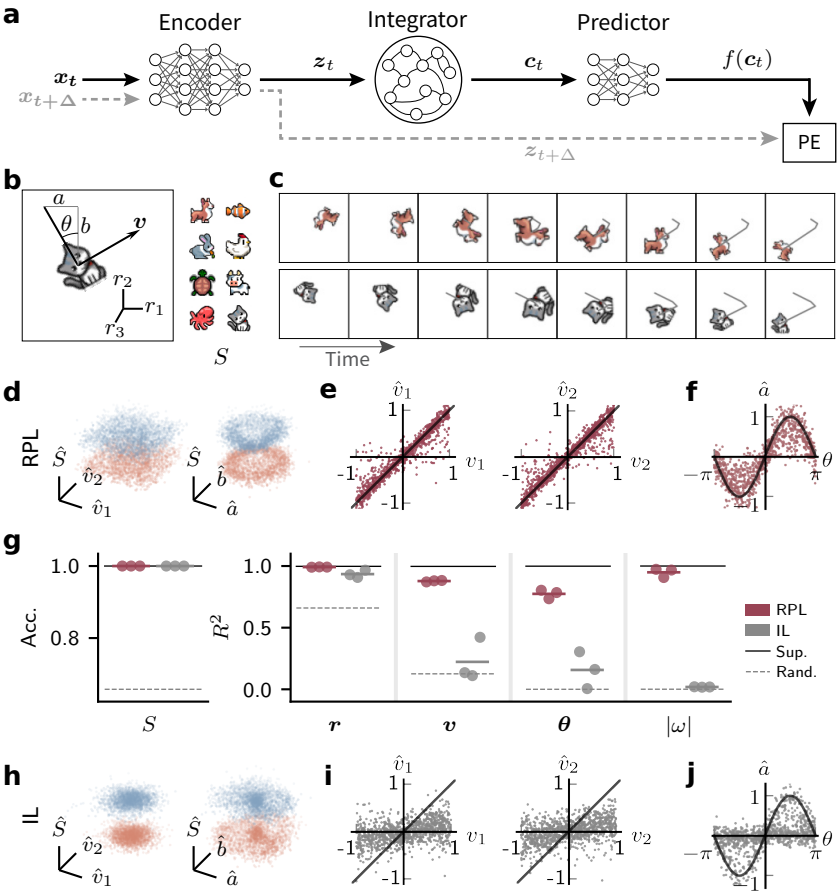


Figure 2: **Learning latent variables from motion stimuli without reconstruction and labels.** **(a)** Schematic of the network model. RPL consists of different network components. First, the "encoder" encodes sensory stimuli $\boldsymbol{x}$ into intermediate embeddings $\boldsymbol{z}$. Subsequently, the "integrator" synthesizes these embeddings into the internal representation $\boldsymbol{c}$. Finally, the "predictor" network predicts future embeddings $\boldsymbol{z}_{t+\Delta}$ from $\boldsymbol{c}_t$ yielding the associated prediction error (PE). The learning objective is to minimize this PE. **(b)** Synthetic data generation paradigm with known latent variables. We generate motion trajectories by moving a 2D animal image, chosen randomly from one of eight categories (right), within a square arena (left). Each frame is characterized by the object position $r_{1/2}$, its scale to mimic position in the $r_3$ direction, and the orientation angle $\theta$. For each trajectory we randomly select a constant velocity $\boldsymbol{v}$ and angular velocity $\omega = \dot{\theta}$. To obtain pixel images that serve as network inputs, we further place these motion trajectories on a gray background and add Gaussian pixel noise, and elastic collisions at the borders (Methods). **(c)** Two example trajectories shown frame-by-frame. **(d)** Scatter plots of linearly decoded latent variables from learned internal representations $\boldsymbol{c}$ for a representative sample of cat (blue) and dog (red) input images. Velocity $(v_1, v_2)$ and object identity $S$ (left). Orientation $(a, b)$ and $S$ (right). **(e)** Decoded velocity components $\hat{v}_{1/2}$ as a function of ground truth on held-out data. **(f)** Same as (e) but for $\hat{a}$ as function of $\theta$. The model (RPL) has formed untangled and structured representations of the latent variables. **(g)** Linear readout accuracy of animal categories $S$ (left) and coefficient of determination ($R^2$) (right) for position $\boldsymbol{r}$, velocity, orientation $\boldsymbol{\theta} = (a, b)$, and angular speed $|\omega|$ for linear decoders trained on the internal representations $\boldsymbol{c}$ for $n = 3$ independent networks, trained with the RPL and invariance learning (IL) objectives. Decoding performance for $n = 1$ supervised end-to-end training (Sup.) and a randomly initialized network (Rand.) are given for reference. RPL decoding accuracy for object identity, and $R^2$ values are close to the performance of networks trained end-to-end with supervised learning, whereas IL fails to accurately represent the velocity, orientation, and angular speed. **(h,i,j)** Same as d-f but for IL.

frame in front of a gray background, and added Gaussian pixel noise (Supplementary Fig. S1; Methods). Each video frame was characterized by the animal's identity $S$, its position $\boldsymbol{r}$, and velocity $\boldsymbol{v}$, with $r_{1/2}$ defining the planar position and $r_3$ defining the scale, orientation $\theta$, and the corresponding angular velocity $\omega$. Finally, to ensure that these latent variables were indeed entangled in stimulus space, we measured how well a linear decoder could predict them from pixel values alone. To that end, we quantified linear classification accuracy and decoding quality ($R^2$) of dynamic variables. Most variables could not reliably be decoded from the raw input image sequences (Supplementary Fig. S2).

To test whether RPL recovers the underlying invariant and dynamic latent variables, we trained our model on approximately 5.7 h of synthetic videos (Methods). We then visualized the learned representations by plotting two example classes (Cat and Dog) onto their best linear projections of the latent linear velocities $v_{1/2}$. The resulting point clouds faithfully recovered the underlying uniform velocity distribution of the data (Fig. 2d,e; Supplementary Fig. S3a). Similarly, when regressing on $a = \sin\theta$ and $b = \cos\theta$, the resulting linear projections revealed an explicit orientation encoding in the internal representations (Fig. 2d,f). Finally, we quantified classification accuracy and decoding quality of position, velocity, orientation $\boldsymbol{\theta} = (a, b)$, and angular speed $|\omega|$, and saw high decoding performance for all latent variables (Fig. 2g). Importantly, the decoding performance of RPL was substantially better than a random untrained baseline model and close to a reference model trained using supervised learning. Thus, RPL learns untangled representations of object identity and motion variables that can be read out linearly, a computation that downstream neurons can easily implement.

We were curious whether this dual untangling of object identity and motion variables depends on the particular type of JEPA we used, in which the recurrent pathway predicts the feedforward signal. To address this question, we kept the network architecture unchanged and used invariance learning (IL) with a symmetric learning objective similar to previous work [25, 26] in which the predictor network predicts its own output instead of the encoder output (Methods). Concretely, this IL objective minimizes the prediction error $\boldsymbol{e}^{\mathrm{IL}}_{t+\Delta} = f(\boldsymbol{c}_{t+\Delta}) - f(\boldsymbol{c}_t)$. Networks trained to minimize such IL prediction error formed untangled representations of object identity and position. Still, their representations did not allow reliable decoding of other latent variables such as velocity, orientation, and angular speed (Fig. 2g–j; Supplementary Figs. S3c,d, S4, S5). Similarly, optimizing the network using an IL objective at the level of the embedding $\boldsymbol{z}$ or the representation $\boldsymbol{c}$ (Eqs. (4), (5)), resulted in worse decoding performance than the RPL objective (Supplementary Fig. S2). Thus RPL's asymmetric recurrent prediction circuit is essential to learn representations of static and dynamic latent variables and specifically for representing direction of motion or sequence order.

We next wondered whether an alternative asymmetric JEPA implementation, in which the prediction target is the integrator output rather than the encoder output, could also learn to extract both invariant and dynamic latent variables. If so, this would indicate that the asymmetric prediction character of a JEPA is key to learn both invariant and equivariant representations, while being insensitive to where the prediction errors are computed in the circuit, i.e., before or after the integrator. To investigate this question, we defined context-predictive learning (CL) as a third learning paradigm which minimizes prediction errors resulting from predicting the future representations at the level of the integrator $\boldsymbol{c}_t$ by minimizing prediction errors $\boldsymbol{e}^{\mathrm{CL}}_{t+\Delta} = \boldsymbol{c}_{t+\Delta} - f(\boldsymbol{c}_t)$. We then evaluated the representations using linear probing as before. Again, decoding accuracy from the resulting network representations was worse than RPL, but better than the IL objectives (Supplementary Fig. S2). Hence, RPL's learning objective of predicting embeddings from a context-aware internal representation is crucial for learning untangled representations of invariant and dynamic latent variables.

We speculated that CL's performance gap may be due to the integrator being able to in-

ternally "hallucinate" predictable signals that minimize the prediction error, even though they are unrelated to the input. We reasoned that this issue should become more pronounced when extracting latent variables from noisy input data, thereby requiring the network to rely more heavily on the integrator to infer them by integrating over time. To test this idea, we generated synthetic videos as before while systematically increasing pixel noise and frame dropout, i.e., introducing different numbers of consecutively blank frames at random locations in the video (Supplementary Fig. S6a). Consistent with the above hypothesis, we found that the performance gap between RPL and CL widened with increasing noise levels and number of dropped frames (Supplementary Fig. S6b,c), while RPL's overall performance remained relatively stable. Moreover, CL's performance variability between independently trained models was higher, suggesting that the CL paradigm is less robust. Thus, RPL's particular asymmetric architecture is essential to robustly untangle dynamic and invariant latent variables. Consequently, we will now focus on comparing RPL to IL for the remainder of the article.

### RPL circuits learn an internal world model.

In addition to representing objects and their motion trajectories, animals need to anticipate future states of the world based on past observations. We wondered whether RPL implicitly learns a world model capable of such mental simulation. We reasoned that if the recurrent circuit indeed learns a world model, it should simulate motion trajectories in latent space in an autoregressive way. To test this idea, we introduced a dynamic switch in the network setup that routes the predicted embeddings $\tilde{\boldsymbol{z}}$ back as input to the integrator, replacing the feed-forward input from the environment (Fig. 3a; Methods). Next, we primed a network in feed-forward mode with the first eight frames of an animal motion stimulus to allow the model to infer the underlying latent variables. We then switched to feed-back mode for 16 time steps. Throughout, we decoded all latent variables from the internal representations as before. We found that the networks in feed-back mode simulated motion trajectories that closely resembled the position, scale, and object identity of the actual trajectory (Fig. 3b; Supplementary Fig. S7). Similarly, the velocity and orientation estimates resembled the ground-truth dynamics (Fig. 3c). Finally, to check how errors evolved over time, we calculated the average mean squared error over many simulated trajectories for different dynamic variables. This analysis showed that errors dropped rapidly during the inference phase, as expected, and slowly increased during the simulation phase (Fig. 3d). Together, these findings show that the learned internal representations can reliably simulate future object motion and that RPL learns a world model.

### RPL learns at different levels of abstraction

In the moving animal task studied above, the latent-variable dynamics were simple and predominantly linear, and each animal category was associated with a single image reducing the need for representation learning. We wondered whether RPL's specific asymmetric recurrent architecture would enable it to simultaneously learn representations at different levels of abstraction for more diverse stimulus sets and more intricate sequence structures.

To investigate this question, we generated structured sequences of handwritten digits from the MNIST dataset [41]. Specifically, we generated the sequences using the following grammar. Sequences consisted of distinct digit triplets interspersed by zeros (Fig. 4a). Only triplets within the clusters of digits $\{1, 2, 3\}$, $\{4, 5, 6\}$, or $\{7, 8, 9\}$ were allowed, covering all possible permutations within each cluster. To generate a sequence, we first selected a random cluster, sampled a random permutation within it, and then sampled random instances of the corresponding digits from the MNIST dataset (Fig. 4b). We continued by sampling the next cluster and triplets

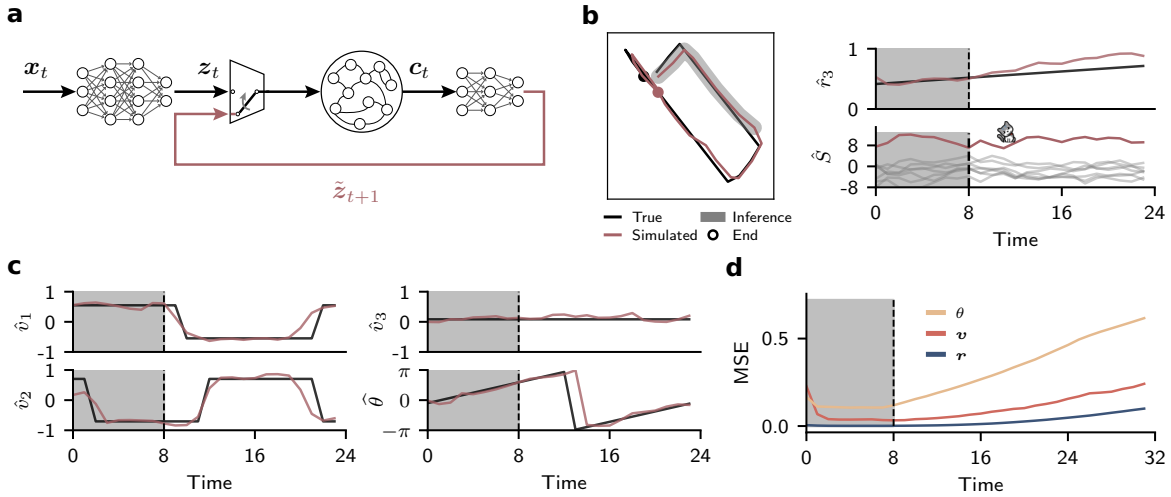


Figure 3: **RPL circuits learn an internal world model.** **(a)** Setup for simulating trajectories using internal representations. We added a dynamic switch allowing to replace feed-forward input to the integrator with internal predictions (cf. Fig. 1). To simulate future trajectories, we provide the network with external-world sensory input during an initial inference period. Afterwards, we replaced feed-forward input to the integrator with predicted embeddings in an autoregressive manner. **(b)** Example of a decoded trajectory (red) compared to the ground truth (black) in the $r_1$-$r_2$ plane (left) and $r_3$ as a function of time (top right; cf. Fig. 2a) as well as the object decoder logits (bottom right). In this example we used an initial inference period of eight frames of a cat trajectory (shaded) followed by 16 steps of internal simulation. **(c)** Decoded velocity and orientation during the inference period (shaded) and simulation. The decoded values closely follow the ground truth. **(d)** Mean-squared error of the decoded quantities compared to ground-truth as a function of time. The error increases monotonically over simulation time.

according to our defined grammar (Methods). We then fed the resulting image sequences to a network model for training with the RPL and IL learning objectives.

To probe the information encoded in the learned representations, we further trained three sets of linear classifiers to decode digit identity, cluster identity, and triplet identity (Fig. 4c; Methods). Encoding triplet identity requires a sense of temporal direction and a specific memory of recent digits. In contrast, cluster identity is direction-invariant and does not require distinguishing digits within a cluster. We thus expected RPL, an asymmetric JEPA, to yield high accuracy across all three tasks, in contrast to IL which, due to its symmetric objective, should only perform well at representing cluster identity. This is indeed what we found. RPL-trained network representations solved all three decoding tasks (Fig. 4d) with accuracy comparable to that of a fully supervised model.

In contrast, IL resulted in lower-fidelity representations of digit and triplet identity, on par with an untrained random network, while cluster identity was still encoded reliably as expected. Nonetheless, we also found that networks trained with IL could still represent individual digits somewhat faithfully at the encoder level by training a separate set of linear classifiers on the encoder outputs $\boldsymbol{z}$ (Supplementary Note S1). However, this improved digit-decoding accuracy was still lower than for RPL. This analysis highlights again the importance of asymmetric prediction error computation through the recurrent pathway for learning time-resolved and abstract representations, even when these representations themselves do not depend on temporal context, as is the case for individual digit stimuli.

To gain further insight into how the temporal context is encoded in the trained networks, we visualized the principal components (PCs) of the learned representations (Fig. 4e,f; Supplemen-

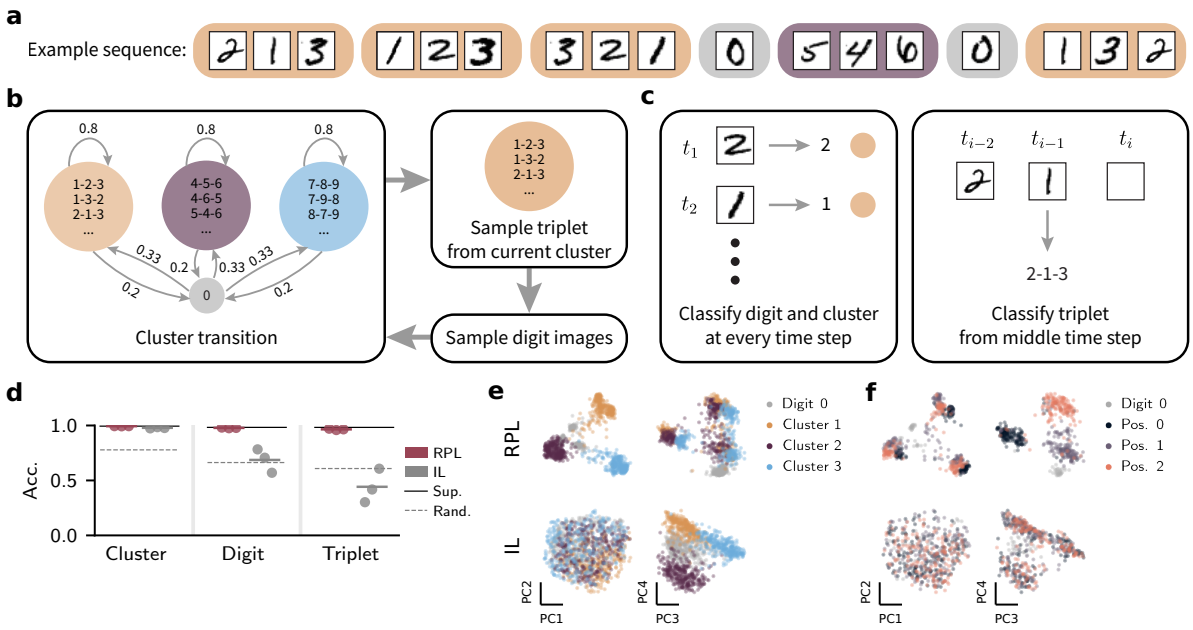


Figure 4: **RPL learns at different levels of abstraction.** **(a)** An example sequence of digit triplets separated by zeros. **(b)** Schematic of the sequence generation paradigm. We generated triplet sequences containing randomly ordered handwritten digits sampled from one of three digit clusters. Transitions between clusters took place with probability 0.2 accompanied by a "zero". For any given sequence, each digit instance was randomly chosen from the MNIST dataset. **(c)** Schematic of the tasks used to investigate the emergence of abstraction by asking how well the network representations encode cluster, digit, and triplet identity respectively. Solving these tasks requires both a mapping from images to abstract digit and cluster representations as well as knowledge about the transition structure. To quantify the representational quality, we trained a linear classifier for each task (Methods). **(d)** Linear decoding accuracy of the cluster, digit, and triplet identities for RPL and IL for $n = 3$ independently trained networks. Decoding accuracy for end-to-end supervised learning (solid line) and a randomly initialized network (dashed line) are given for reference. RPL achieves high accuracy on all tasks, whereas IL results in a preference for cluster identity. **(e)** Principal component (PC) projections of the learned representations colored by cluster identity. Both RPL and IL clearly separate the clusters. **(f)** Same as (d) but colored by temporal position within a triplet. Only RPL yields representations of the abstract temporal transition structure.

tary Fig. S8). While cluster identity was clearly separated by both methods, e.g., along the first two PCs for RPL, and third and fourth PCs for IL (Fig. 4e), it was evident that RPL resulted in more finely structured representations. For instance, RPL representations encoded the temporal position within a triplet in the third and fourth PCs, essentially creating an internal clock that tracks sequence progression (Fig. 4f). Furthermore, PCs five through eight displayed a rich, though less interpretable, organization (Supplementary Fig. S8), presumably encoding the digit identity. In contrast, IL lacked such a fine-grained organization beyond clearly separating cluster identity, lending further support to our hypothesis that RPL's asymmetric architecture is essential for learning abstract sequence structure that requires a sense of direction, whereas IL's symmetric setup is limited to representing invariant features. Thus, RPL's asymmetric recurrent architecture is necessary to simultaneously learn complex representations at different abstraction levels from abstract sequence stimuli.

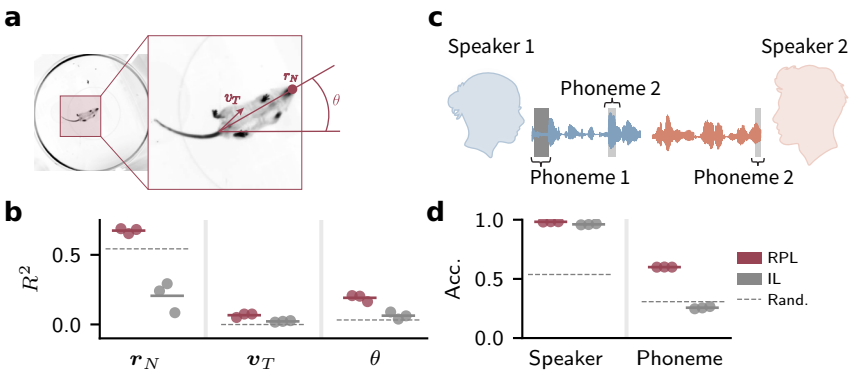


Figure 5: **RPL extracts latent variables from real-world data. (a)** Schematic of real-world video data of behaving mice in an open-field arena [42]. We used annotated key points to compute position $\boldsymbol{r}_N$, velocity $\boldsymbol{v}_T$, and orientation $\theta$ as proxies of latent variables for evaluation (Methods). **(b)** $R^2$ values of decoded position, velocity, and orientation for RPL and IL from $n = 3$ independently trained networks. Decoding performance for a randomly initialized network (dashed line) shown for reference. **(c)** Schematic of real-world speech data from the Librispeech corpus [44]. We relied on speaker identity and phoneme labels to assess model performance. **(d)** Linear decoding accuracy of speaker identity and phoneme sequence for RPL and IL for $n = 3$ independently trained models. Decoding from a randomly initialized network given for reference (dashed line). RPL has high decoding accuracy for speaker identity and phonemes. In contrast, IL yields high accuracy only for the speaker identity.

## Representation learning on real-world data

Having established RPL's effectiveness in extracting latent variables from synthetic datasets, we wondered to what extent these findings would generalize to real-world data. To address this question, we used published video data of behaving mice in an open-field arena [42] in which nose position $\boldsymbol{r}_N$ and tail root position $\boldsymbol{r}_T$ had previously been annotated with DeepLabCut [43]. Based on these annotations, we further estimated animal orientation $\theta$ and velocity $\boldsymbol{v}_T$ (Fig. 5a; Methods) as latent variables to evaluate the quality of the learned representations.

We then trained RPL networks on the videos and used linear probing for evaluation as before. For comparison, we also trained the same network using the IL objective. We found that decoding from RPL representations yielded higher $R^2$ values across all variables (Fig. 5b), consistent with previous findings, that RPL untangles representations during learning from real-world videos. We observed that velocity was more difficult to decode, possibly because mice remained stationary in large portions of the videos, leading to a velocity distribution with a peak near zero (Supplementary Fig. S9).

Since the mouse videos were structurally quite similar to our synthetic data above (cf. Fig. 2), we wanted to see whether RPL would yield similarly useful latent representations in a different data domain. To investigate this question, we trained both RPL and IL models on spoken-language recordings from the Librispeech corpus [44]. This dataset includes labels for speaker identity and the phonemes of the spoken text (Fig. 5c). Since phonemes change within and across sentences whilst speaker identity remains unchanged, we expected that RPL would learn representations that allow decoding both, whereas symmetric IL objectives would focus on speaker identity. To check whether this is the case, we evaluated the learned representations by linearly decoding speaker identity and phoneme sequence (Methods). Our results confirmed our expectation. RPL yielded higher accuracy for both speaker identity and phonemes, whereas IL failed to represent phoneme information (Fig. 5d). Thus, RPL extracts meaningful latent variables across different real-world data domains.

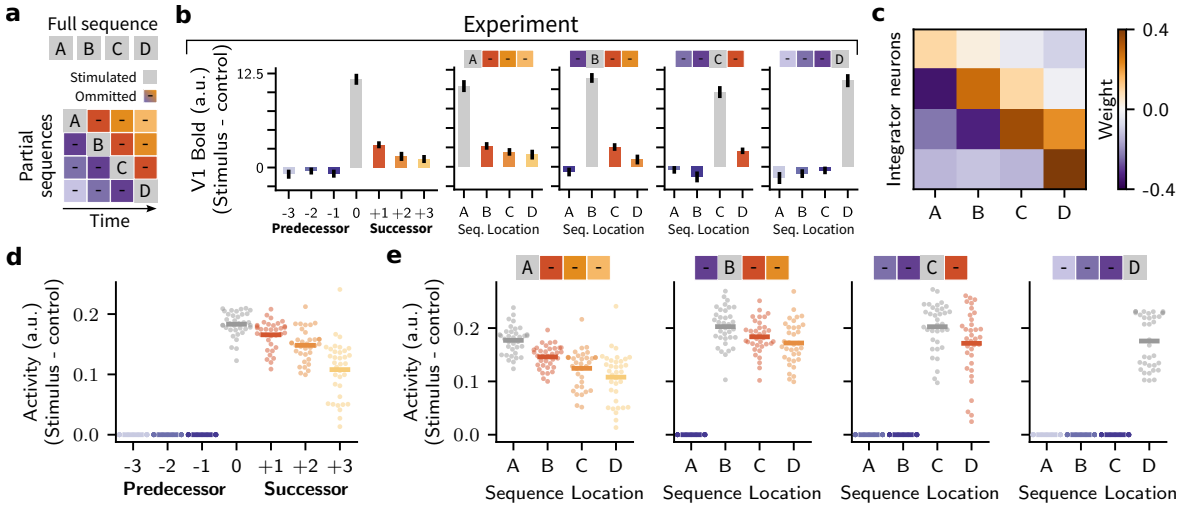


Figure 6: **RPL learns successor-like representations comparable to human V1. (a)** Schematic of the full and partial sequences used in the experiment. **(b)** Relative BOLD signal responses reproduced in human V1 [40]. Relative stimulus response to partial sequence stimuli (left). Individual responses to partial sequences (right). Partial sequence stimulation caused brain activity encoding successor locations but not predecessor locations. Error bars represent the standard error of the mean. Data extracted and reproduced from Ekman, Kusch, and Lange [40]. **(c)** Relative neuronal selectivity matrix extracted from the RPL model representations ($n = 35$ models). The negative values below the diagonal indicate inhibition of activity at predecessor locations. Positive values above the diagonal indicate that the successors' representations are driven by the stimulus. **(d)** Average neural activity for all partial sequences at non-stimulated and stimulated locations after removing the baseline activity as in (b). Each dot is a trained model and the solid lines represent the average activity over all models. The plot indicates positive neural activity at stimulated and successor locations, and no activity at predecessor locations. **(e)** Neural activity at all sequence locations in response to each partial sequence. These plots indicate the decaying positive response at successor locations for individual partial sequences, hinting at successor-like representations in response to each partial sequence.

## RPL learns successor-like representations comparable to V1 in humans

We next wondered to what extent RPL network models predicted *in vivo* data. To that end, we first focused on signatures of successor representations, whereby a neuronal population encodes a predictive representation of future states, such that each state representation reflects the expected future occupancy of other states under a given policy [45]. Recent work described successor-like representations in fMRI recordings from the primary visual cortex of humans [40]. During these experiments, humans were exposed to a sequence of four non-overlapping items (A-B-C-D) across multiple trials. After an initial learning phase, the participants were exposed to either the entire sequence or only one of the four sequence items, referred to as partial sequences (Fig. 6a). The data from V1 indicated activation at the successor but not the predecessor locations upon partial sequence exposures (Fig. 6b).

We wanted to know whether RPL networks exhibit similar activation patterns at successor locations after being exposed to comparable sequential stimuli. To investigate this question, we trained minimal RPL circuit models ($n = 35$; Methods) on an (A-B-C-D) sequence to mirror the experimental protocol. After sorting the neurons in the integrator based on their selectivity, we visualized the average weight matrix associated with the neural selectivity to each sequence item (Fig. 6c). This matrix showed the largest entries on the main diagonal, positive values on the two upper off-diagonals, and negative values below the main diagonal, a clear signature of

successor representations.

We then computed the average neural activity at the stimulated, predecessor, and successor locations in response to partial sequences, after removing baseline neural activity from all neural responses. We observed that, as in the fMRI recordings [40], the stimulated location showed the highest activity, while activity at successor locations followed a decaying pattern. In contrast, there was no activity at the predecessor locations consistent with the experimental data (Fig. 6b,d). Similarly, we plotted neural activity in response to each partial sequence to see whether this pattern also existed in selective neurons for non-stimulated sequence items at successor locations. We observed a consistent pattern of activity at stimulus and successor but not predecessor locations (Fig. 6e). Thus, RPL entails successor-like representations as observed in V1 of humans experiencing repeating sequence stimuli.

## RPL learns abstract sequence representations similar to macaque PFC

Having observed that RPL learns predictive representations consistent with those in a primary sensory cortex, we asked whether our model could also capture more abstract representations ascribed to higher-order cortical areas, such as the PFC. A widely used experimental setup to investigate abstract sequence learning is the local-global oddball paradigm (Fig. 7a). Here, an animal is exposed to trials or sequence presentations consisting of four stimuli one after the other, either repeating the same stimulus (xxxx) or an oddball in the last slot (xxxY), constituting a local deviant. Repeating trials are organized into blocks, in which the animal is conditioned to expect a particular sequence type, e.g., xxxx or xxxY, as the standard by dint of frequent presentation at the beginning of the block. Later trials in the block will be either global standards or infrequent global deviants, depending on whether they match the context set at the start of the block.

One such experiment was performed by Bellet et al. [39] who recorded neurons from the macaque PFC while exposing them to visual local-global oddball sequences. They found that populations of neurons in the PFC spontaneously learned to encode several abstract sequence properties, such as their serial position within a trial of four stimuli, stimulus identity, global context, and the presence of local and global deviants.

Since we had already observed that RPL is capable of abstract sequence learning (cf. Fig. 4), we hypothesized that RPL should also be able to learn the sequence properties of local-global oddball sequences. To test this hypothesis, we trained an RPL circuit model on the identical visual stimulus sequences that Bellet et al. [39] presented to the macaques, closely mirroring their experimental protocol. Then, following their analysis (Methods), we extracted neuronal population vectors by training linear readouts to decode: serial position, stimulus identity, global context, and local/global deviance.

We found that the serial position of an item in a sequence and stimulus identity could be decoded reliably (Fig. 7b,c) as in the experiments, albeit without the experimentally observed accentuation of the first and the last item, which are presumably due to primacy and recency effects in the experiment [39]. To test whether the RPL network had learned an internal model of the sequence structure, we decoded the global context variable from the internal representations. As observed in the authors' original analysis, we identified neural subspaces in our model that coded for the global context inferred from previous sequence observations (Fig. 7d). Finally, we assessed our model's responses to local and global deviants. We found qualitatively similar separation of population responses corresponding to the presence of local deviance in both predicted (xY|xY) as well as unpredicted (xY|xx) local deviant stimuli (Fig. 7e). Similarly, the model showed an explicit encoding of the presence of global deviance (xx|xY and xY|xx; Fig. 7f) in its population responses, as in the experiment. However, our model did not exhibit the onset

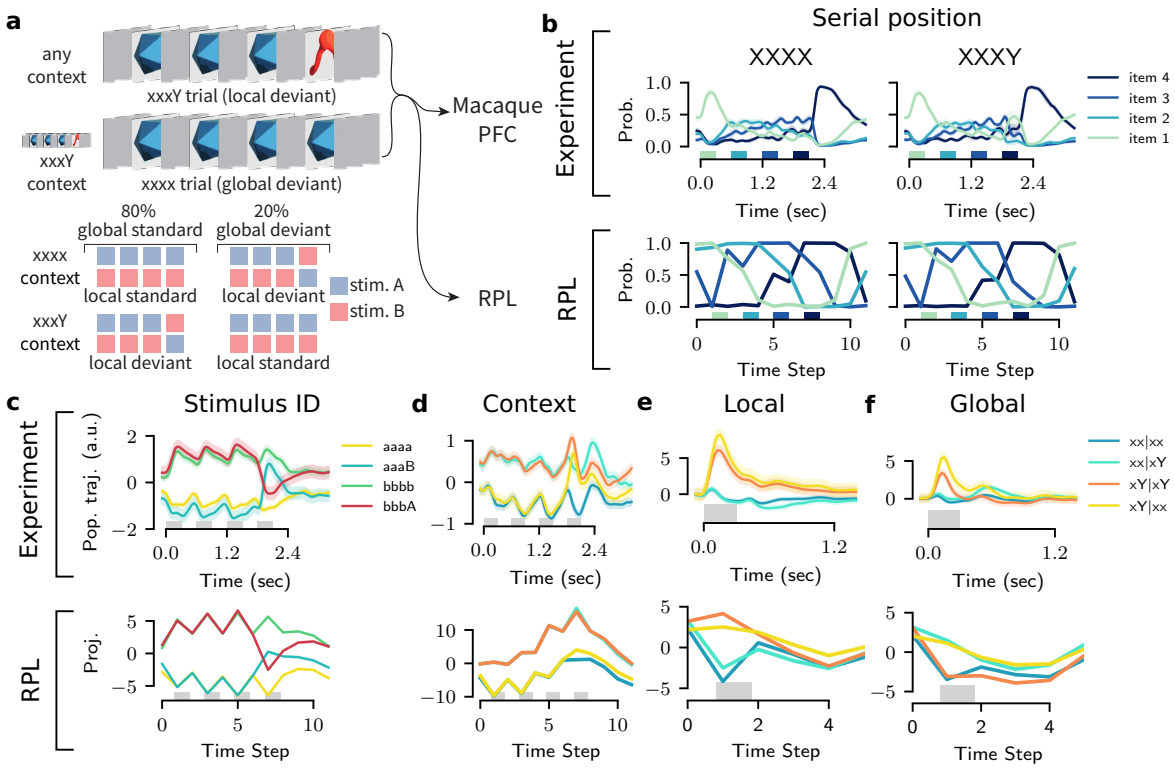


Figure 7: **RPL learns abstract sequence representations similar to macaque PFC.** **(a)** Schematic of the local-global oddball paradigm used by Bellet et al. [39]. Example trials consisting of four sequential stimuli show an "xxxY" and an "xxxx" trial in the context of frequent "xxxY" presentations (top). A given trial can be a local standard or deviant depending on the local sequence structure of the four stimuli and a global standard or deviant depending on the context set for a block of several trials (bottom). Figure adapted from [39]. **(b)** Comparison of temporal serial position representation in experiment (top) and the model (bottom). Plots show best linearly decoded probabilities of the four serial positions (items 1 through 4) averaged over all xxxx and xxxY trials. RPL learns representations that encode serial position more robustly than the macaque PFC. The experimental data corresponds to Monkey A and has been reproduced from publicly available data and code. **(c)** Comparison of stimulus identity representation in the experiment (top) and the model (bottom). Plots show projections of neural activity or RPL representations on directions that best separate pairs of input stimuli (Item a vs b) that comprise x and Y, averaged across trials (Methods). RPL representations encode stimulus identity similar to the experiment. **(d)** Same as (c) but for representation of the global context set by the frequent trial. **(e)** Same as (c) but for the presence or absence of a local deviant at the fourth sequence position. **(f)** Same as (c) but for the presence or absence of a global deviant, decoded from the fourth sequence position.

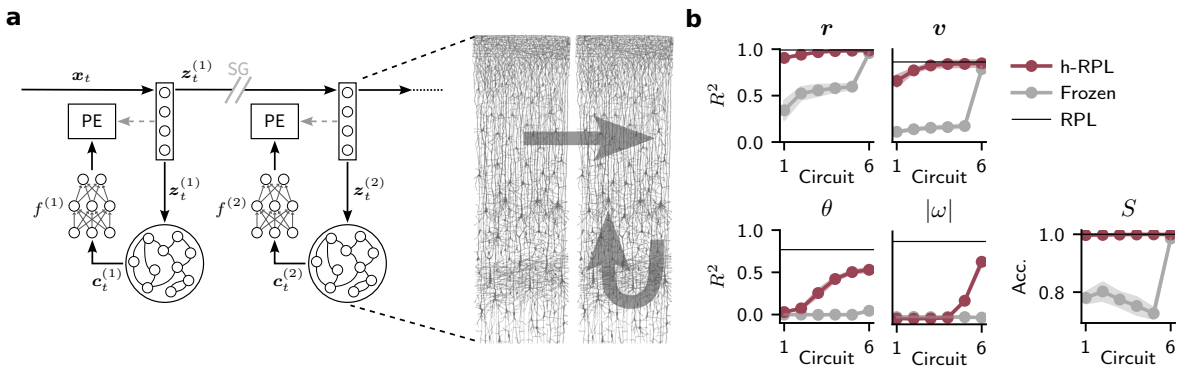


Figure 8: **Untangling in a cortical model with local prediction error circuits. (a)** Schematic of the network model consisting of six identical RPL circuits in a hierarchy, each containing a shallow encoder, an integrator and a predictor network. Each circuit computes its own prediction error while Stop Grad (SG) operations prevented gradients from flowing between the individual circuits. This hierarchical RPL (h-RPL) model is consistent with the notion of hierarchical predictive processing based on a stereotypical cortical microcircuit. Cortical anatomy drawing (right) adapted from Cajal [47]. **(b)** Decoding $R^2$ values and accuracy of the latent variables (cf. Fig. 2b,c) for each RPL circuit in h-RPL (red) and a variant in which only the last circuit in the hierarchy was trained (gray). Shading corresponds to one standard deviation over $n = 3$ independent models. The solid black line denotes the performance of RPL with a deep encoder (cf. Fig 2j). The h-RPL model progressively learns latent variable representations comparable to end-to-end trained RPL.

and primacy effects observed in the data, presumably because it lacked neuronal adaptation mechanisms. Nevertheless, we found that the representations learned via RPL qualitatively matched the relative encoding in PFC. Thus, RPL learns abstract sequence representations whose relative encoding structure is consistent with PFC activity.

## Untangling in a cortical model with local prediction error circuits

So far, we have used deep encoder networks, computing prediction errors at the top-level output, using backpropagation of error (backprop) to update all weights. However, experimental evidence points to local mismatch computation across the hierarchy as early as primary sensory cortex [11]. Moreover, how the brain approximates backprop to solve the credit assignment problem remains an open question [46]. Hence, we wondered whether RPL remains effective without end-to-end optimization and to what extent.

To investigate this question, we built a hierarchical cortical model in which we implemented different cortical areas as local RPL circuits that learn independently. Each circuit consists of a shallow encoder, an integrator, and a local prediction-error circuit. One possible mapping of the model onto cortical anatomy is to view the encoder as being implemented by the superficial Layers 2/3 (L2/3), while the deeper Layer 5 (L5) implements the integrator (Fig. 8a). Finally, information flows laterally between cortical areas, and predictor circuits are implemented vertically within each "cortical column". This mapping is consistent with observed mismatch responses in L2/3 [11], but other mappings onto the cortical microcircuit are possible.

We trained this hierarchical RPL (h-RPL) model on synthetic moving animal data (cf. Fig. 2). For simplicity, we trained every circuit using backprop, but we prevented gradients from flowing between circuits across the hierarchy (Methods). Doing so confines credit assignment to individual circuits with a few synaptic transmission steps, a setting in which biologically plausible backprop approximations are effective [48–51]. In consequence, each circuit $l$ learned

independently in this hierarchical network model.

To assess the quality of the learned representations, we trained linear decoders for each circuit and extracted the latent variables as before (Fig. 8b). This analysis revealed that the linear decoding accuracies and $R^2$ values for the dynamic variables increased with depth along the hierarchy, plateauing at performance levels comparable to those of the end-to-end trained RPL setup. To check whether this linear decodability of the evolving latent variables also implied "perceptual straightening" as described in the visual system [52, 53], we further quantified straightness along the model's hierarchy. We found, that straightness indeed increased along the hierarchy and was due to learning (Supplementary Fig. S10).

One potential caveat with this analysis is that the hierarchical encoders jointly constitute a deep convolutional network with increasing receptive field sizes. To check that the observed increase in linear decodability of latent variables was not merely due to the progressively larger receptive fields, we trained another instance of the same hierarchical model in which we kept the weights in all but the last circuit of the hierarchy frozen. This resulted in an overall reduction in linear decodability across the hierarchy (Fig. 8b). While the drop in $R^2$ for position, velocity, and object category could be almost entirely compensated for by the optimized last circuit, this was not the case for orientation and angular speed. Additionally, we observed a steady increase in the autocorrelation time of the representations over the hierarchy for h-RPL but not in the frozen counterpart (Supplementary Fig. S11). These findings are consistent with the idea that better decodability across circuits is not merely a result of increasing receptive field size, but rather the progressive untangling of representations across the hierarchy. Thus, RPL is a powerful learning algorithm for finding dynamic latent variables even in a hierarchical setting, which resonates with the core ideas of predictive processing and the experimentally observed mismatch neurons in primary sensory cortices.

## Discussion

We have introduced recurrent predictive learning (RPL), a minimal model of a plastic network that learns untangled representations of objects and their dynamics from sequential observations alone, without requiring labels, an implausible decoder, or a reconstruction objective. Formally, RPL constitutes a JEPA, a machine learning architecture that can extract invariant and equivariant representations at multiple levels of abstraction from synthetic and real-world data. In addition to learning representations, the resulting networks also acquire an internal world model capable of simulating future states purely from their internal representations. Importantly, we showed that the internal representations in our models resemble neuronal activity in the macaque PFC and exhibit signatures of successor-like representations, comparable to those found in human V1. To learn, RPL relies on internal prediction errors generated within a specific circuit motif: an encoder, a network with little temporal memory, feeds into an integrator-predictor circuit combination that, in turn, tries to predict the future embeddings of the encoder. Crucially, this motif is functionally universal and allows hierarchical networks with local predictor circuits to progressively uncover latent variables. Finally, our model's specific structure offers a new way to interpret canonical cortical microcircuits, while making concrete, testable predictions.

The notion that the brain builds and progressively updates an internal model of the world through comparison with observations is a central tenet of systems neuroscience [10, 11, 15, 17, 22, 29, 54–60]. Classic predictive coding models based on this idea assume an underlying generative model that *reconstructs* sensory input [10, 13, 14, 16]. The generative model is learned and updated by computing prediction errors in the input space. However, predictive coding is facing increasing pressure due to its difficulty in explaining recent experimental data

[21–23]. These difficulties might stem from its core assumption of an underlying generative model and it remains elusive whether and how neural circuits could reconstruct expected stimuli for comparison to the observations. For instance, there are no back-projections from the visual cortex to the retina. Instead, the cortical anatomy seems well poised to operate on its internal representations and use internal prediction errors, which may manifest as mismatch responses in cortical circuits [11, 61]. Moreover, recent work in machine learning suggests that reconstruction, as required by generative models, is inefficient for learning high-level abstract representations because it tends to focus on behaviorally irrelevant low-level stimulus details [37, 62]. Together, these insights call for a revised predictive processing framework that neither relies on a generative model nor on a reconstruction objective.

In this article, we studied such an alternative model. RPL is a JEPA, meaning it is a self-supervised machine learning model [28, 31, 63, 64] that neither relies on a reconstruction objective nor on negative samples. While the idea that the brain, and specifically the cortex, relies on self-supervised learning to build internal representations is not new [18, 65], previous models made assumptions that are difficult to reconcile with neurobiology. For example, they either relied on an autoencoder [18] pretrained with a reconstruction objective or a contrastive learning approach [29] requiring negative samples, which are difficult to reconcile with the requirement of temporal locality for learning in the brain. Finally, previous self-supervised learning approaches had a limited focus on invariance learning [26] or "slow" features [66] while ignoring the fact that biological agents also have to learn equivariant representations, allowing to encode information such as from *where* an incoming car is approaching. JEPAs avoid all of the above issues, all the while being consistent with the fundamental notion of predictive processing that the brain builds an internal model.

A plethora of different JEPAs have demonstrated their effectiveness in machine learning [26, 27, 31–37, 62]. However, not all JEPAs are biologically plausible. For instance, existing studies made strong assumptions about the allowed transformations between consecutive stimuli [34] or were not applied to real-world problems and lacked a circuit-level interpretation [67]. In part, this lack stems from current JEPAs typically relying on biologically implausible transformer networks, predicting randomly masked parts of the stimulus, and optimizing a single global prediction error for training the whole network, all of which are difficult to reconcile with neural anatomy. Finally, the majority of research has focused on invariance learning, in which representational quality is assessed primarily through object classification tasks, e.g., via linear decoding or probing. Quantifying model performance at the level of the internal representations is a welcome and necessary departure from evaluating representational quality solely based on a model's ability to reconstruct stimuli at the pixel level, the strategy commonly used in past predictive coding models [13, 18, 19, 68]. However, focusing on pure invariance learning, seems too much of an oversimplification. An increasing number of machine learning studies are acknowledging the importance of equivariant representations [27, 34, 69–72], albeit with the limitations mentioned above and mainly assuming direct access to the underlying transformations.

Our work is the first to overcome these issues. Specifically, we propose a temporal JEPA, i.e., one that works without masking and is based on recurrent neural networks rather than transformers. Moreover, to the best of our knowledge, h-RPL also constitutes the first implementation of a hierarchical JEPA optimizing multi-level prediction errors as originally proposed by LeCun [30]. Functionally, RPL learns both invariant and equivariant representations consistent with the idea of active filtering [73], as well as a latent generative model that enables it to simulate plausible dynamics in the latent space when queried with its own predictions. Interestingly, the necessity of separating the feedforward prediction target from the integrator-predictor branch directly stems from this reliance on a plausible recurrent neural network architecture, which opens the door to further investigation of the putative links between our model and the

cortical microcircuit.

There are several exciting connections between our work and neuroscience experiments, both related to circuit function and anatomy. At the functional level, we observed that RPL exhibits signatures of cortical activity in both primary sensory and higher-order areas. Firstly, RPL develops successor-like sequence representations, where current stimuli induce neural activity at future but not at predecessor locations, comparable to those recently described in human V1 [40]. Secondly, RPL also learns abstract sequence representations reminiscent of those found in the macaque PFC in a local-global oddball sequence paradigm [39]. Moreover, our work is consistent with studies suggesting that the brain develops representations optimized for prediction [12, 20, 74] with increasing levels of invariance [75]. For instance, h-RPL showed a similar increase in temporal stability across its hierarchy of circuits (Supplementary Fig. S11). Finally, our work sheds new light on putative plasticity mechanisms to explain experimental findings of perceptual straightening in the visual cortex [52, 53].

At the level of neural circuits, our work on h-RPL suggests new possible interpretations of the cortical microcircuit. In our h-RPL model, prediction errors are computed and minimized locally within each circuit in the hierarchy. While this notion is consistent with the observed prediction-error-like neurons in L2/3 of the sensory cortex [11, 61], it is also consistent with classic models of predictive coding [10, 13, 76]. What is different in a JEPA like the one described here is that representations and prediction errors are not necessarily functionally segregated, and prediction errors are not communicated to higher-level areas. Instead, the JEPA prescribes a situation in which one part of the network generates a teaching signal for another, driving representation learning in both. What is communicated to other areas are representations rather than error signals, albeit increasingly untangled (cf. Fig. 8). In the context of our model, one possibility is that L2/3 primarily implements the encoder stream via lateral connectivity.

In contrast, L5 implements the integrator, and possibly parts of the predictor network. In this model, both L2/3 and L5 generate instructive teaching signals for each other, while L5 also performs inference by integrating over consecutive embeddings from L2/3. Importantly, abstract stimulus representations, although with different properties, exist in both areas in this model. Moreover, acute changes to the coupling between the encoder and integrator would primarily affect L5 representations, while leaving L2/3 activity largely untouched. Such a decoupling between L2/3 and L5 seems broadly consistent with synchrony changes observed in L5 but not in other neurons, during general anesthesia [77]. Chronic changes to this coupling, however, would permanently impair representation learning in this model.

Our work has several limitations that we plan to address in the future. First, the link between RPL and concrete brain circuits is still under-constrained. For instance, in our h-RPL model, we assumed that the predictor is entirely local, with no top-down input. This is a simplification of our model, and cortical predictor circuits may have access to higher-level abstract representations. This could allow modulating the predictions within each circuit with global context from higher-level circuits, consistent with experimental findings on the role of feedback in sensory cortices [22, 60]. A similar combination of a feedforward sensory pathway with a separate recurrent feedback pathway has been previously proposed as the "dual counterstream" architecture, an organizational principle of cortical structure [78]. Exploring this connection and its functional implications is an exciting avenue for future research.

Another limitation of our model is that we relied on backprop, a biologically implausible training method. While we have shown that circuit-local training can be effective with h-RPL (cf. Fig. 8), we still used backprop to train each circuit. In the future, we will investigate how biologically plausible credit assignment strategies could make the learning algorithm more credible [46, 50, 51, 79–82]. Notably, such approaches tend to be more competitive with backprop for shallow networks, so they may remain effective for replacing the few backpropagation steps

required by the hierarchical learning procedure presented here.

Moreover, latent-variable learning in RPL was sensitive to the prediction time horizon and the underlying distribution of the latent variable. In this article, we have only explored this connection to a limited extent (cf. Supplementary Fig. S9). A more systematic investigation of how the time horizon influences the learned representations will be the goal of future research.

Finally, we relied on linear decodability as a proxy to quantify the quality of learned representations. To that end, we extended the commonly used linear classification strategy to linear regression of dynamic latent variables. While we saw that RPL embeds unrelated latent variables in orthogonal subspaces and improves overall linear decodability, a key question is why. In fact, there is no immediately apparent reason why the learned representations should be embedded on linear manifolds, and presumably they are not. One reason behind the improved linear decodability may be that linear decoding benefits from the orthogonal embedding of unrelated factors, combined with the high embedding dimension. An important question for future research is thus to gain a deeper theoretical understanding of how representations of continuous latent variables emerge in self-supervised learning JEPAs and which inductive biases lead to representations conducive for their linear decodability.

Extracting abstract representations of the world's invariant and dynamic features is a crucial function of the brain that enables intelligent behavior. Here, we have presented a plausible computational model of self-supervised representation learning of both object identity and other latent variables. Our framework extends a long tradition of predictive processing models while going beyond mere predictive coding and pixel space prediction. Specifically, our work builds a conceptual bridge between cortical architecture and JEPAs, a promising machine learning architecture that does not rely on a reconstruction objective. While many important questions remain open, we think our model provides a fresh approach to understanding the computational principles of the cortex, and that exploring the research avenues outlined above holds great promise for gaining deeper insights into the computational principles underlying intelligence in both biological and artificial systems.

## Data and code availability

The code to reproduce all modeling results in this article is available on Github[1]. We further used the following publicly available datasets: The Librispeech dataset is freely available for download[2]. We used train and test splits and force-aligned phoneme annotations as defined previously [28] and made available by the authors[3]. The mouse arena videos are available on Figshare[4]. Further, the two down-sampled videos used in this article are included in our Github repository. Finally, the plots for the macaque PFC experiment [39] were reproduced using data[5] and analysis code[6] shared publicly by the authors.

## Acknowledgments

We thank all Zenke Lab members for comments and discussions throughout this project. Special thanks to Peter Buttaroni for contributing illustrations for figures. This project was supported

[1] https://github.com/fmi-basel/recurrent-predictive-learning

[2] https://www.openslr.org/resources/12/train-clean-100.tar.gz

[3] https://drive.google.com/drive/folders/1BhJ2umKH3whguxMwifaKtSra0TgAbtfb

[4] https://doi.org/10.6084/m9.figshare.19213272.v1

[5] https://figshare.com/articles/dataset/Electrophysiological_data_from_the_local-global_experiment/24311767/1

[6] https://github.com/mebellet/PFCSequenceModels

by the Swiss National Science Foundation (Grant Number PCEFP3_202981), EU's Horizon Europe Research and Innovation Programme (Grant Agreement No. 101070374, CONVOLVE) funded through SERI (Ref. 1131–52302), and the Novartis Research Foundation.

## Author contributions

Ashena Gorgan Mohammadi & Manu Srinath Halvagal: Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review and editing; Friedemann Zenke: Conceptualization, Funding acquisition, Methodology, Supervision, Writing – original draft, Writing – review and editing.

## Declaration of interests

The authors declare no competing interests.

# Methods

In this article, we studied different representation learning paradigms that rely on prediction errors computed in internal representation space. We quantified their ability to extract latent variables like object identity, position, velocity, and orientation from streaming data such as video or audio. In all cases, we used deep neural networks comprised of three distinctly connected network elements: a feed-forward encoder network, a recurrent integrator network, and a feed-forward predictor network.

## Learning objectives

All networks were trained using backprop to optimize one of the following learning objectives defined on the internal representations.

**Recurrent predictive learning (RPL).** While most predictive processing models optimize for reconstructing stimuli in input space, RPL optimizes a predictive objective function defined entirely on internal representations. Concretely, RPL is a particular JEPA which predicts the embeddings $\boldsymbol{z}_{t+\Delta}$ of future stimuli using a predictor network $f$ that acts on the representations $\boldsymbol{c}_t$ from the integrator (cf. Fig. 2a). So, the RPL objective is defined as:

$$\mathcal{L}_{\text{RPL}} = \|f(\boldsymbol{c}_t) - \text{SG}(\boldsymbol{z}_{t+\Delta})\|^2, \tag{2}$$

where SG corresponds to the Stop-Gradient operator. The RPL objective avoids representational collapse by combining the SG with an asymmetric prediction architecture and, consequently, does not require explicit variance regularization [32, 83, 84].

**Invariance learning (IL).** Invariance learning is an alternative approach for representation-space predictive learning that uses a symmetric prediction scheme with explicit regularization terms [25, 26] for preventing collapse. This is also sometimes referred to as a joint-embedding architecture [30, 33]. The IL objective defined on the output of the predictor network in an architecture identical to the one used in RPL (cf. Eq. (2)) is one such approach. We use this, along with a regularization objective $\mathcal{L}_{\text{reg}}$ similar to the one used by Halvagal and Zenke [26], as a baseline for comparison with RPL. The overall IL objective function is then defined as:

$$\begin{aligned}\mathcal{L}_{\text{IL}} &= \|f(\boldsymbol{c}_t) - \text{SG}(f(\boldsymbol{c}_{t+\Delta}))\|^2 + \mathcal{L}_{\text{reg}}\left(f(\boldsymbol{c}_t)\right) \\ &= \|f(\boldsymbol{c}_t) - \text{SG}(f(\boldsymbol{c}_{t+\Delta}))\|^2 - \lambda_1 \frac{1}{M}\sum_{i=1}^{M} \log \sigma_i^2 + \lambda_2 \sum_{i=1}^{M}\sum_{j\neq i}^{M} C_{ij}^2 \quad ,\end{aligned} \tag{3}$$

where $\sigma_i^2$ is the overall variance of the activity of neuron $i$ in the output layer (i.e. $f(\boldsymbol{c}_t)$), $C_{ij}$ is the correlation coefficient between the activities of neurons $i$ and $j$, and $\lambda_1$ and $\lambda_2$ are constants that control the strength of each regularization term. The two terms in $\mathcal{L}_{\text{reg}}$ prevent the neurons from activity collapse and learning identical information, respectively.

Other variations of the IL objective can be defined to enforce invariance on the embeddings $\boldsymbol{z}_t$ or representations $\boldsymbol{c}_t$ of the RPL network. Mathematically, the objective functions corresponding to these variations can be written as:

$$\mathcal{L}_{\text{IL}^{\text{z}}} = \|\boldsymbol{z}_t - \text{SG}(\boldsymbol{z}_{t+\Delta})\|^2 + \mathcal{L}_{\text{reg}}(\boldsymbol{z}_t) \quad , \tag{4}$$

and

$$\mathcal{L}_{\text{IL}^{\text{c}}} = \|\boldsymbol{c}_t - \text{SG}(\boldsymbol{c}_{t+\Delta})\|^2 + \mathcal{L}_{\text{reg}}(\boldsymbol{c}_t) \quad , \tag{5}$$

respectively with the target for $\mathcal{L}_{\text{reg}}$ changed in each case as indicated.

**Context-predictive learning (CL).** Finally, the RPL objective is not the only possible implementation of a JEPA given our three-component network architecture. Alternatively, one could also optimize the same network for predicting future representations $\boldsymbol{c}_{t+\Delta}$ from the integrator instead of $\boldsymbol{z}_{t+\Delta}$, yielding the context-predictive learning (CL) objective function:

$$\mathcal{L}_{\mathrm{CL}} = \|f(\boldsymbol{c}_t) - \mathrm{SG}(\boldsymbol{c}_{t+\Delta})\|^2 \quad . \tag{6}$$

## Datasets and network models

We investigated RPL's performance on multiple synthetic and real-world video and audio datasets with corresponding deep neural network architectures for the encoder, integrator, and predictor.

### Synthetic moving animal video dataset

We generated the synthetic 32-frame moving animal videos by first generating sequences of the continuous latent variables. To that end, we randomly sampled an animal category and simulated its motion with a random initial velocity $v_1, v_2 \sim \mathcal{U}(-8, 8)$ and rotation with a constant randomly sampled angular velocity $\omega \sim \mathcal{U}(-\frac{\pi}{6}, \frac{\pi}{6})$. We also simulated motion in a third depth dimension by changing the scale of the animal starting with a random initial velocity $v_3 \sim \mathcal{U}(-0.125, 0.125)$. Furthermore, we assumed elastic collisions at the position boundaries in all three dimensions, which changed the direction of motion. Overall, the latent variables included the animal category $S$, position $(r_1, r_2)$, scale $r_3$, velocity $\boldsymbol{v} = (v_1, v_2, v_3)$, planar orientation $\theta$, and the corresponding angular velocity $\omega$. We sampled the initial orientation from a uniform distribution $\theta \sim \mathcal{U}(0, 2\pi)$. The initial position in the $r_1$-$r_2$ plane was sampled from a uniform distribution: $r_1, r_2 \sim \mathcal{U}(8, 56)$, where 8 is the padding around the image, ensuring that the animal remained visible within the frame. The scale $r_3$ was randomly initialized from and remained limited to the interval $[0.2, 1]$.

We then sampled the latent variable trajectory at discrete intervals and constructed the corresponding $64 \times 64$ RGB video frames. Each frame contained one antialiased animal image scaled and rendered on a gray background. Here, a scale $r_3 = 1$ corresponded to an image size of $32 \times 32$ pixels. Finally, we added independent Gaussian noise $\sim \mathcal{N}(0, 0.1)$ to each pixel and color channel (Supplementary Fig. S1). Using this procedure, we built a fixed corpus of 4000 32-frame videos for validation and testing, and randomly generated training sets of 16000 videos for each network model we trained.

**Network architecture and training.** We used a convolutional neural network (CNN) with six convolutional layers for the encoder, each with 32 filters of size $5 \times 5$ and padding of size 2, followed by a batch normalization layer and a rectified linear unit (ReLU) activation function. We used a stride of two for the first two layers to reduce the spatial dimensions, and stride one for later layers. The output of the last convolutional layer was flattened to obtain the embedding vector $\boldsymbol{z}_t$. This embedding was then fed to the integrator with a single layer of 512 long short-term memory (LSTM) units to obtain the internal representation $\boldsymbol{c}_t$. Finally, this representation was passed through the predictor, a feedforward Multi-Layer Perceptron (MLP) network with one hidden layer of 512 units and the ReLU activation function, followed by a linear output layer. We trained the networks using the AdamW optimizer [85] with learning rate $3 \times 10^{-4}$, weight decay $10^{-5}$, and batch size 128 for 1000 epochs using the aforementioned objective functions with $\Delta = 1$ on the 16000 training videos, i.e., a total of 512000 frames, using a cosine learning rate schedule. In the case of RPL and CL (Eqs. (2) and (6)), we used a learning rate 10 times higher for the predictor network relative to the encoder, an established protocol for

JEPA training [32]. For IL variants, we used $\lambda_1 = 20$ and $\lambda_2 = 200$ for the regularization strengths.

**Evaluation.** We evaluated the trained network models using linear probing as follows. First, we froze all network parameters and computed the internal representations of the training and testing datasets. We then standardized these representations to zero mean and unit variance. We used the resulting representations of the training dataset to fit an ordinary least square linear regression model for each continuous latent variable and a linear classifier for animal category. Since orientation is not a linear variable, we used $a = \sin\theta$ and $b = \cos\theta$ as target variables for the linear readout, which was trained by minimizing the mean-squared-error. We additionally optimized a phase offset for each animal category to correct for orientation offset biases per animal in the shared representation space. Moreover, we decoded the angular speed $|\omega|$ from time-averaged video representations. We trained the linear classifier on time-averaged video frames of the training set using the AdamW optimizer with a learning rate of $10^{-3}$, weight decay of $10^{-5}$, and a batch size of 128 for 250 epochs. Throughout the article, we reported the classification accuracies and coefficient of determination $R^2$ for regression on the held-out test data. Finally, to provide a lower reference value for comparison, we evaluated the readout performance as described above for an untrained, randomly initialized network as a control. Similarly, to compare with supervised learning, we also trained the same network end-to-end using the regression and classification readouts as supervised training objectives for optimizing the encoder and integrator. Note that the predictor is unused during evaluation in all cases, and also during supervised training.

**Internal simulation of trajectories.** To investigate the model's ability to accurately simulate trajectories in latent space, we primed the network with the first eight frames of a video. After the initial priming phase, the predicted embeddings were fed back to the integrator autoregressively for the following sixteen time steps thereby generating simulated internal representations (Fig. 3a). To evaluate the quality of the simulation, we then estimated the latent variables from the internal representations using the linear decoders described above and compared them to the actual latent variables of the ground truth trajectory (cf. Fig. 3 and S7).

### Synthetic sequences of handwritten digits

To test RPL on more abstract stimulus sequences, we generated synthetic sequences of handwritten digit images consisting of digit triplets separated by zeros (Fig. 4a). We generated the digit sequences by sampling clusters, triplets and digits for 64 time steps according to a structured grammar (cf. Fig. 4b). Specifically, 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 possible triplets. Each digit sequence consisted of such triplets interspersed with zeros, starting with a triplet drawn from a randomly chosen cluster. Each triplet was either followed by another random triplet from the same cluster with a probability of 0.8, or a zero with a probability of 0.2. After a zero, a cluster group was chosen uniformly at random to continue the sequence. Overall, the latent variables in this case were the digit identity, the cluster group, and the triplet identity. Given the generated sequence of digits, we finally sampled corresponding $28 \times 28$ grayscale images from the MNIST training and test sets for training and testing respectively. Using this approach, we generated 10000 training sequences and 10000 test sequences for each model.

**Network architecture and training.** For the encoder, we used an MLP with two hidden layers consisting of 32 units, each followed by batch normalization and ReLU activation functions. We fed the resulting embeddings into a single-layer recurrent neural network (RNN) integrator with 128 units to generate the internal representations. The representations were further passed through a feedforward MLP predictor network with a single hidden layer of 128 units and ReLU activation function, followed by a linear output layer. We trained all networks using the AdamW optimizer with a learning rate $3 \times 10^{-4}$, weight decay $10^{-3}$, and batch size 64 for 200 epochs using the different objective functions with $\Delta = 1$ on the training sequences. For IL, we used $\lambda_1 = 1$ and $\lambda_2 = 10$ for the regularization strength.

**Evaluation.** To assess the quality of the learned representations, we first froze the parameters of the trained networks. Then, we extracted the representations of the training and test sets and standardized them as before. We subsequently trained a set of linear classifiers on these representations to decode digit identity and cluster identity at each time step, and triplet identity after observing the first two digits of any triplet subsequence (cf. Fig. 4b). We trained all three linear classifiers for 50 epochs using AdamW with a learning rate of $10^{-3}$, weight decay of $10^{-5}$, and a batch size of 128. As before, we also computed performance estimates for a random untrained baseline network as well as end-to-end supervised training.

### Real-world videos of behaving mice

To study RPL's ability to extract latent variables from real-world data, we used videos from the open source dataset of behaving mice in an open-field arena provided by Luxem et al. [42]. Concretely, we used two one-hour videos with a 25 Hz frame rate resulting in a total of $2 \times 90000$ frames. To split the data into training and test sets, we sliced the videos into one-minute segments from which we used the first fifty seconds for training and the last 10 seconds for testing. We downscaled all frames to $96 \times 96$ pixel frames using HandBrake [86] and subsampled the video segments at 5 Hz, i.e., only using every fifth frame, to decrease computational training cost. We further computed the proxy latent variables velocity, orientation, and position based on the dataset's DeepLabCut [43] estimates of tail root and nose position. To reduce label noise in the velocity estimates of each sample (Supplementary Fig. S9), we smoothed them using a cumulative summation over time before standardizing the values. We further noted that the dataset contained erroneous nose position outside of the arena, for instance, whenever the nose was not visible when the animal was rearing. To prevent this label noise from entering our orientation estimates, we replaced the affected data points with linearly interpolated values.

**Network architecture and training.** We used a CNN with six convolutional layers, each with 32 filters as the encoder. We used $5 \times 5$ filters with padding and strides of size two for the first three layers, and $3 \times 3$ filters with padding and strides of size one for the following three layers. All convolutional layers were followed by a batch normalization layer and ReLU activation functions. The output of the last layer was flattened to obtain the embedding vector $\boldsymbol{z}_t$. This embedding was then fed to the integrator consisting of a single layer of 256 LSTM units to obtain the internal representation $\boldsymbol{c}_t$. Finally, the internal representations passed through an MLP predictor network with a single hidden layer comprised of 256 units followed by ReLU activation functions and a dense linear output layer. We trained the networks using the AdamW optimizer with learning rate $3 \times 10^{-4}$, weight decay $10^{-5}$, and batch size 128 for 500 epochs using the aforementioned objective functions with $\Delta = 1$ on the training set. In the case of RPL (Eq. (2)), we trained the predictor network with a 10 times faster learning rate than the

rest of the network. When training using an IL objective, we additionally set the regularization strengths to $\lambda_1 = 20$ and $\lambda_2 = 200$.

**Evaluation.** We evaluated the networks on the test subset of the dataset using linear least squares with L2 regularization, i.e., ridge regression, with regularization coefficient of 1.0 on the trained internal representations to decode the latent variables $\boldsymbol{r}_N$, $\boldsymbol{v}_T$, and $\theta$.

### Real-world natural speech recordings

To test our representation learning framework in a different data domain, we used a 100-hour audio recording subset of the Librispeech dataset [44] 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.** Following the work of Oord, Li, and Vinyals [28], we used a 1D CNN architecture acting on the raw audio waveform with 512 output units as the encoder. The resulting embeddings were then fed into the integrator, a single LSTM layer with 256 units, to generate the internal representations. The representations were then passed through an MLP predictor network with a single hidden layer of 512 units and ReLU activation function, followed by a dense linear output layer. The network was trained using the AdamW optimizer with learning rate $3 \times 10^{-4}$, no weight decay, and batch size 256 for 500 epochs using the different objective functions with $\Delta = 8$. The hyperparameter $\Delta$ was determined via grid search to be the optimal prediction horizon for RPL. In the case of IL objectives, we used $\lambda_1 = 2$ and $\lambda_2 = 20$ as regularization strengths.

**Evaluation.** We evaluated the trained networks on the test set following the same steps as above and by training linear readouts to decode the phoneme and speaker labels. The linear readout contained 251 units for speaker classification and 41 units for phoneme classification. We trained the linear classifiers using the AdamW optimizer with learning rate $10^{-3}$, weight decay $10^{-5}$, and batch size 64 (for speaker identity) for 50 epochs. Following the same procedure as in Oord, Li, and Vinyals [28], we use a batch size of one for training readouts for the phoneme classification task, as the start and end point annotations for phoneme labels are not provided. As before, we evaluated a randomly initialized network for comparison.

## Learning successor-like representations

To investigate whether RPL learns successor-like representations, we modeled a stimulus sequence A-B-C-D as in the experiment [40] with one-hot vectors. Since the one-hot encoding of the inputs alleviated the need for learning stimulus representations, we replaced the encoder with an identity matrix. For simplicity, we used a vanilla RNN with four units as integrator, whose weights we initialized with an identity matrix, and a dense linear predictor whose weights we initialized with a Glorot uniform distribution [87]. We trained $n = 35$ networks for 352 epochs on the A-B-C-D sequence using the RPL objective with a learning rate of $10^{-3}$ for the integrator and 0.01 for the predictor. Further, we used a weight decay of $10^{-4}$ for all the components.

We trained logistic regression models on the internal representations of the full sequences to classify the sequence locations. We then sorted the integrator units based on their selectivity to sequence locations. We computed the average weight matrix of all logistic regression models and

visualized it to see if it resembles a successor matrix (Fig. 6b). To generate the partial sequences, we used zero input vectors for the non-stimulated sequence locations and the corresponding one-hot vector for the stimulated location. We then computed the activity at stimulated and non-stimulated locations for each partial sequence by recording the mean population activity of the integrator units at each sequence location and subtracting the population response to a zero input (Fig. 6d). These values were averaged over all partial sequences for each stimulated and non-stimulated location to obtain the activity at individual predecessor, stimulated, and successor locations (Fig. 6c).

### Learning abstract sequence representations from local-global oddballs

To study whether RPL learns abstract sequence representations similar to the primate PFC, we modeled the local-global oddball experiment conducted by Bellet et al. [39]. We first extracted images of the ten visual stimuli used in the experiment and downsampled them to $32 \times 32$ pixel color images. We then constructed the oddball sequences from [39] consisting of pairs of stimuli organized in blocks of twenty sequences (scaled down from the original block size of two hundred). Each sequence comprised four stimulus presentations, either repeating the same stimulus (xxxx or xx for short) or with a local deviant at the last position (xxxY or xY for short). Here, x stands for either of the two items in each pair of stimuli and y for the other, fixed within one block. The first five sequences in each block were consistently either xx or xY, setting the global standard or "context" for that block. The next fifteen sequences were randomly set to either global standards (80%) or rare global deviants (20%). The global standards were repeats of the standard sequence, i.e., xx in an xx block (xx|xx) or xY in an xY block (xY|xY). The global deviants differed from the standard sequence only in the last position (xY|xx or xx|xY). The full sequence of stimuli also contained blank frames, one between each pair of consecutive stimuli in a given sequence, and four between sequences in the same block. We generated 2000 such blocks with the stimulus pairs and contexts uniformly chosen at random for each block for training the network model, and 2000 more for testing.

**Network architecture and training.** Since the stimulus set consisted of only ten images, we used a simple MLP with two hidden layers and batch normalization for the encoder. The MLP took in the full vector of image pixels as input and had 64 units in each hidden layer and its output layer. We then fed the embedding from the output of the encoder to an integrator consisting of a single LSTM layer with 256 units to generate the internal representations. These were then passed through an MLP predictor with one hidden layer with 256 units. Both the encoder and predictor MLPs used the ReLU activation function. For training, each block of 20 sequence presentations was given as one training sample sequence. We trained the network using the AdamW optimizer with learning rate $3 \times 10^{-4}$, batch size 128, and weight decay $10^{-3}$ for 2000 epochs using the RPL objective function with $\Delta = 1$. As before, we also used a 10 times higher learning rate for the predictor.

**Evaluation.** For evaluating the trained network, we closely followed the analysis of PFC population recordings done by Bellet et al. [39]. This consisted of training linear decoders for serial sequence position, stimulus identity, global context and presence of local and global deviance as above, but with specific sequence subtypes used for training the readout (cf. Fig 3 and Methods in [39]) as summarized below:

**Serial position:** For sequence position, we trained binary classifiers using logistic regression for each position in the sequence using only xx trials for training. This yielded predicted

probabilities of items 1 through 4 at each time point of a sequence, plotted for the held-out test set in Fig. 7b.

**Stimulus identity:** Stimulus identity was defined as a fixed binary label for the two items in each of the five stimulus pairs. We trained a binary classifier for decoding this binary label using only the first three stimuli in each sequence for training. The projections of the representations onto the weights found by the binary classifier are plotted in Fig. 7c for the held-out test set of sequences.

**Global context:** For decoding the context, we first averaged the representations over the first three stimulus pairs in every sequence attaching a binary label corresponding to whether the global standard was a repeat sequence (xx) or a sequence with a local oddball (xY). As above, we trained a binary classifier with logistic regression and plotted the projections of the representations from held-out sequences onto the weights found by training the binary classifier.

**Deviance:** For decoding both global and local deviance, we used the representation corresponding to the last stimulus presentation to train a binary classifier with logistic regression. Then, we plotted the projections of the representations for the test set onto the weights of the binary classifier.

All linear decoders were trained with AdamW using the learning rate $10^{-3}$ and weight decay $10^{-5}$.

### Hierarchical RPL (h-RPL)

To assess the effectiveness of RPL in a hierarchical setting, we constructed a network from six RPL circuits each with a shallow single-layer 2D CNN with 32 filters of size $5 \times 5$ and padding of size 2, followed by a batch normalization layer and a ReLU activation function. The first two convolutional layers had a stride of two while the convolutional layers in higher circuits had a stride of 1. Note that this stack of shallow instantaneous encoders in h-RPL results in the same deep convolutional encoder used in RPL for the same dataset. The encoder output was sent to the next circuit in the hierarchy and a flattened version was sent to the local integrator circuit, an LSTM with 512 units (Fig. 8a) for all circuits. Each circuit also had its own MLP predictor with a single hidden layer comprised of 512 units with ReLU activation functions and followed by a dense linear output. Importantly, the different circuits were gradient isolated and independently optimized the RPL objective (Eq. 2) at each level of the hierarchy, without backpropagating the gradients between circuits. Finally, we trained the model on the synthetic moving animal videos (cf. Fig. 2) using the AdamW optimizer with learning rate $3 \times 10^{-4}$, weight decay $10^{-5}$, and batch size 128 for 500 epochs using the RPL objective. As before, we evaluated the representations using linear probing albeit now for each circuit individually.

## References


[1] Nikos K Logothetis and David L Sheinberg. "Visual object recognition." In: *Annual review of neuroscience* 19 (1996), pp. 577–621.

[2] David H Hubel and Torsten N Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of physiology* 160.1 (1962), p. 106.

[3] Jean-René Duhamel, Carol L Colby, and Michael E Goldberg. "The updating of the representation of visual space in parietal cortex by intended eye movements". In: *Science* 255.5040 (1992), pp. 90–92.

[4] David Whitney et al. "Flexible retinotopy: motion-dependent position coding in the visual cortex". In: *Science* 302.5646 (2003), pp. 878–881.

[5] Philippa Anne Johnson et al. "Position representations of moving objects align with real-time position in the early visual response". In: *eLife* 12 (Jan. 2023). Ed. by Clare Press, Tirin Moore, Clare Press, and Samson Chota, e82424. ISSN: 2050-084X. DOI: 10.7554/eLife.82424.

[6] John H Maunsell and David C Van Essen. "Functional properties of neurons in middle temporal visual area of the macaque monkey. I. Selectivity for stimulus direction, speed, and orientation". In: *Journal of neurophysiology* 49.5 (1983), pp. 1127–1147.

[7] Eivind Hennestad, Aree Witoelar, Anna R Chambers, and Koen Vervaeke. "Mapping vestibular and visual contributions to angular head velocity tuning in the cortex". In: *Cell Reports* 37.12 (2021).

[8] Nicholas J. Priebe, Stephen G. Lisberger, and J. Anthony Movshon. "Tuning for Spatiotemporal Frequency and Speed in Directionally Selective Neurons of Macaque Striate Cortex". In: *Journal of Neuroscience* 26.11 (2006), pp. 2941–2950. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.3936-05.2006. eprint: https://www.jneurosci.org/content/26/11/2941.full.pdf.

[9] Alexander V. Roitman, Siavash Pasalar, Michael T. V. Johnson, and Timothy J. Ebner. "Position, Direction of Movement, and Speed Tuning of Cerebellar Purkinje Cells during Circular Manual Tracking in Monkey". In: *Journal of Neuroscience* 25.40 (2005), pp. 9244–9257. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.1886-05.2005. eprint: https://www.jneurosci.org/content/25/40/9244.full.pdf.

[10] 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 (Jan. 1999), pp. 79–87. DOI: 10.1038/4580.

[11] Georg B. Keller and Thomas D. Mrsic-Flogel. "Predictive Processing: A Canonical Cortical Computation". In: *Neuron* 100.2 (Oct. 2018), pp. 424–435. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2018.10.003.

[12] Yosef Singer et al. "Sensory cortex is optimized for prediction of future input". In: *eLife* 7 (June 2018). Ed. by Jack L Gallant and Sabine Kastner, e31557. ISSN: 2050-084X. DOI: 10.7554/eLife.31557.

[13] William Lotter, Gabriel Kreiman, and David Cox. "A neural network trained for prediction mimics diverse features of biological neurons and perception". In: *Nature machine intelligence* 2.4 (2020), pp. 210–219.

[14] Loreen Hertäg and Henning Sprekeler. "Learning prediction error neurons in a canonical interneuron circuit". In: *eLife* 9 (Aug. 2020). Ed. by Srdjan Ostojic, Richard B Ivry, and Srdjan Ostojic. Publisher: eLife Sciences Publications, Ltd, e57541. ISSN: 2050-084X. DOI: 10.7554/eLife.57541.

[15] Stefano Recanatesi et al. "Predictive learning as a network mechanism for extracting low-dimensional latent space representations". en. In: *Nature Communications* 12.1 (Mar. 2021). Number: 1 Publisher: Nature Publishing Group, p. 1417. ISSN: 2041-1723. DOI: 10.1038/s41467-021-21696-1.

[16] Fabian A. Mikulasch, Lucas Rudelt, Michael Wibral, and Viola Priesemann. "Where is the error? Hierarchical predictive coding through dendritic error computation". en. In: *Trends in Neurosciences* 46.1 (Jan. 2023). Publisher: Elsevier, pp. 45–59. ISSN: 0166-2236, 1878-108X. DOI: 10.1016/j.tins.2022.09.007.

[17] Daniel Levenstein, Aleksei Efremov, Roy Henha Eyono, Adrien Peyrache, and Blake A. Richards. "Sequential predictive learning is a unifying theory for hippocampal representation and replay". In: *bioRxiv* (Apr. 2024). DOI: 10.1101/2024.04.28.591528.

[18] Kevin Kermani Nejad, Paul Anastasiades, Loreen Hertäg, and Rui Ponte Costa. "Self-supervised predictive learning accounts for cortical layer-specificity". en. In: *Nat Commun* 16.1 (July 2025), p. 6178. ISSN: 2041-1723. DOI: 10.1038/s41467-025-61399-5.

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

[20] Luke Taylor, Friedemann Zenke, Andrew J. King, and Nicol S. Harper. "Temporal prediction captures key differences between spiking excitatory and inhibitory V1 neurons". en. In: (May 2024). DOI: 10.1101/2024.05.12.593763.

[21] Selina S Solomon, Huizhen Tang, Elyse Sussman, and Adam Kohn. "Limited Evidence for Sensory Prediction Error Responses in Visual Cortex of Macaques and Humans". In: *Cerebral Cortex* 31.6 (Mar. 2021), pp. 3136–3152. ISSN: 1047-3211. DOI: 10.1093/cercor/bhab014. eprint: https://academic.oup.com/cercor/article-pdf/31/6/3136/41163156/bhab014.pdf.

[22] David Richter, Tim C. Kietzmann, and Floris P. de Lange. "High-Level Visual Prediction Errors in Early Visual Cortex". In: *PLOS Biology* 22.11 (Nov. 11, 2024), e3002829. ISSN: 1545-7885. DOI: 10.1371/journal.pbio.3002829.

[23] Shohei Furutachi, Alexis D. Franklin, Andreea M. Aldea, Thomas D. Mrsic-Flogel, and Sonja B. Hofer. "Cooperative thalamocortical circuit mechanism for sensory prediction errors". en. In: *Nature* 633.8029 (Sept. 2024). Publisher: Nature Publishing Group, pp. 398–406. ISSN: 1476-4687. DOI: 10.1038/s41586-024-07851-w.

[24] Randall Balestriero et al. *A Cookbook of Self-Supervised Learning*. arXiv:2304.12210 [cs]. Apr. 2023.

[25] Adrien Bardes, Jean Ponce, and Yann LeCun. "VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning". In: *International Conference on Learning Representations*. 2022.

[26] Manu Srinath Halvagal and Friedemann Zenke. "The combination of Hebbian and predictive plasticity learns invariant object representations in deep sensory networks". en. In: *Nat Neurosci* 26.11 (Nov. 2023). Number: 11 Publisher: Nature Publishing Group, pp. 1906–1915. ISSN: 1546-1726. DOI: 10.1038/s41593-023-01460-y.

[27] Julie Xueyan Niu, Cristina Savin, and Eero Simoncelli. "Learning predictable and robust neural representations by straightening image sequences". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 40316–40335.

[28] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation Learning with Contrastive Predictive Coding". In: *arXiv:1807.03748 [cs, stat]* (July 2018). arXiv: 1807.03748.

[29] Bernd Illing, Jean Ventura, Guillaume Bellec, and Wulfram Gerstner. "Local plasticity rules can learn deep representations using self-supervised contrastive predictions". In: *Advances in Neural Information Processing Systems* 34 (2021).

[30] Yann LeCun. "A Path Towards Autonomous Machine Intelligence Version 0.9.2, 2022-06-27". en. In: (2022).

[31] Jean-Bastien Grill et al. "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning". In: *arXiv:2006.07733 [cs, stat]* (June 2020). arXiv: 2006.07733.

[32] Xinlei Chen and Kaiming He. "Exploring simple siamese representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15750–15758.

[33] Mahmoud 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.

[34] Hafez Ghaemi, Eilif Benjamin Muller, and Shahab Bakhtiari. "Seq-JEPA: Autoregressive Predictive Learning of Invariant-Equivariant World Models". In: *NeurIPS 2024 Workshop: Self-Supervised Learning - Theory and Practice*. 2024.

[35] Adrien Bardes et al. *V-JEPA: Latent Video Prediction for Visual Representation Learning*. 2024.

[36] Mido Assran et al. "V-jepa 2: Self-supervised video models enable understanding, prediction and planning". In: *arXiv preprint arXiv:2506.09985* (2025).

[37] Randall Balestriero and Yann LeCun. *Learning by Reconstruction Produces Uninformative Features For Perception*. Feb. 2024. DOI: 10.48550/arXiv.2402.11337. arXiv: 2402.11337 [cs.CV].

[38] Etai Littwin et al. "How jepa avoids noisy features: The implicit bias of deep linear self distillation networks". In: *arXiv preprint arXiv:2407.03475* (2024).

[39] Marie E Bellet et al. "Spontaneously emerging internal models of visual sequences combine abstract and event-specific information in the prefrontal cortex". In: *Cell reports* 43.3 (2024).

[40] Matthias Ekman, Sarah Kusch, and Floris P de Lange. "Successor-like representation guides the prediction of future events in human visual cortex and hippocampus". In: *eLife* 12 (Feb. 2023). Ed. by Morgan Barense, Chris I Baker, and Helen Barron, e78904. ISSN: 2050-084X. DOI: 10.7554/eLife.78904.

[41] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[42] Kevin Luxem et al. "Identifying behavioral structure from deep variational embeddings of animal motion". In: *Communications Biology* 5.1 (2022), p. 1267.

[43] Alexander Mathis et al. "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning". In: *Nature Neuroscience* 21.9 (Aug. 20, 2018), pp. 1281–1289. ISSN: 1546-1726. DOI: 10.1038/s41593-018-0209-y.

[44] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. "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.

[45] Peter Dayan. "Improving generalization for temporal difference learning: The successor representation". In: *Neural computation* 5.4 (1993), pp. 613–624.

[46] Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. "Backpropagation and the brain". In: *Nature Reviews Neuroscience* (Apr. 2020). Publisher: Nature Publishing Group, pp. 1–12. ISSN: 1471-0048. DOI: 10.1038/s41583-020-0277-3.

[47] Santiago Ramon y Cajal. *Comparative study of the sensory areas of the human cortex.* Wikimedia Commons. Public Domain. 1899.

[48] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. "Random synaptic feedback weights support error backpropagation for deep learning". In: *Nature communications* 7.1 (2016), p. 13276.

[49] Arild Nøkland. "Direct feedback alignment provides learning in deep neural networks". In: *Advances in neural information processing systems* 29 (2016).

[50] James CR Whittington and Rafal 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.

[51] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. "Dendritic cortical microcircuits approximate the backpropagation algorithm". In: *Advances in neural information processing systems* 31 (2018).

[52] Olivier J Hénaff, Robbe LT Goris, and Eero P Simoncelli. "Perceptual straightening of natural videos". In: *Nature neuroscience* 22.6 (2019), pp. 984–991.

[53] Olivier J Hénaff et al. "Primary visual cortex straightens natural video trajectories". In: *Nature communications* 12.1 (2021), p. 5982.

[54] Andre M. Bastos et al. "Canonical Microcircuits for Predictive Coding". en. In: *Neuron* 76.4 (Nov. 2012), pp. 695–711. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2012.10.038.

[55] Jeff Hawkins. *A Thousand Brains: A New Theory of Intelligence.* en. Google-Books-ID: YMrvDwAAQBAJ. Hachette UK, Mar. 2021. ISBN: 978-1-5416-7580-3.

[56] Iva K. Brunec and Ida Momennejad. "Predictive Representations in Hippocampal and Prefrontal Hierarchies". In: *Journal of Neuroscience* 42.2 (Jan. 12, 2022), pp. 299–312. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.1327-21.2021. pmid: 34799416.

[57] Byron H Price, Cambria M Jensen, Anthony A Khoudary, and Jeffrey P Gavornik. "Expectation violations produce error signals in mouse V1". In: *Cerebral Cortex* 33.13 (2023), pp. 8803–8820.

[58] Matthew 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.

[59] Colleen J. Gillon et al. "Responses to Pattern-Violating Visual Stimuli Evolve Differently Over Days in Somata and Distal Apical Dendrites". en. In: *J. Neurosci.* 44.5 (Jan. 2024). Publisher: Society for Neuroscience Section: Research Articles. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.1009-23.2023.

[60] Shuting Han and Fritjof Helmchen. "Behavior-Relevant Top-down Cross-Modal Predictions in Mouse Neocortex". In: *Nature Neuroscience* (Jan. 4, 2024), pp. 1–11. ISSN: 1546-1726. DOI: 10.1038/s41593-023-01534-x.

[61] Pawel Zmarz and Georg B. Keller. "Mismatch Receptive Fields in Mouse Visual Cortex". In: *Neuron* 92.4 (Nov. 2016), pp. 766–772. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2016.09.057.

[62] Hugues Van Assel, Mark Ibrahim, Tommaso Biancalani, Aviv Regev, and Randall Balestriero. "Joint Embedding vs Reconstruction: Provable Benefits of Latent Space Prediction for Self Supervised Learning". In: *arXiv preprint arXiv:2505.12477* (2025).

[63] Mehdi Noroozi and Paolo Favaro. "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Cham: Springer International Publishing, 2016, pp. 69–84. ISBN: 978-3-319-46466-4.

[64] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. "Barlow Twins: Self-Supervised Learning via Redundancy Reduction". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 12310–12320.

[65] Bernd Illing, Wulfram Gerstner, and Johanni Brea. "Biologically plausible deep learning - But how far can we go with shallow networks?" eng. In: *Neural Netw* 118 (Oct. 2019), pp. 90–101. ISSN: 1879-2782. DOI: 10.1016/j.neunet.2019.06.001.

[66] Laurenz Wiskott and Terrence J. Sejnowski. "Slow Feature Analysis: Unsupervised Learning of Invariances". In: *Neural Computation* 14.4 (Apr. 2002), pp. 715–770. ISSN: 0899-7667. DOI: 10.1162/089976602317318938.

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

[68] Boris Vladimirskiy, Robert Urbanczik, and Walter Senn. "Hierarchical Novelty-Familiarity Representation in the Visual System by Modular Predictive Coding". In: *PLOS ONE* 10.12 (Dec. 2015), pp. 1–19. DOI: 10.1371/journal.pone.0144636.

[69] Alexandre Devillers and Mathieu Lefort. "Equimod: An equivariance module to improve self-supervised learning". In: *arXiv preprint arXiv:2211.01244* (2022).

[70] Jung Yeon Park, Ondrej Biza, Linfeng Zhao, Jan Willem van de Meent, and Robin Walters. "Learning symmetric embeddings for equivariant world models". In: *arXiv preprint arXiv:2204.11371* (2022).

[71] Quentin Garrido, Laurent Najman, and Yann Lecun. "Self-supervised learning of split invariant equivariant representations". In: *arXiv preprint arXiv:2302.10283* (2023).

[72] Yifei Wang et al. "Understanding the role of equivariance in self-supervised learning". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 127483–127510.

[73] Mark H. Histed. *Active filtering: a predictive function of recurrent circuits of sensory cortex*. arXiv:2501.10521 [q-bio]. Jan. 2025. DOI: 10.48550/arXiv.2501.10521.

[74] Stephanie E Palmer, Olivier Marre, Michael J Berry, and William Bialek. "Predictive information in a sensory population". In: *Proceedings of the National Academy of Sciences* 112.22 (2015), pp. 6908–6913.

[75] Eugenio Piasini et al. "Temporal stability of stimulus representation increases along rodent visual cortical hierarchies". In: *Nature communications* 12.1 (2021), p. 4448.

[76] T. Ed Li, Mufeng Tang, and Rafal Bogacz. "Predictive Coding Model Detects Novelty on Different Levels of Representation Hierarchy". In: *Neural Computation* 37.8 (July 2025), pp. 1373–1408. ISSN: 0899-7667. DOI: 10.1162/neco_a_01769. eprint: https://direct.mit.edu/neco/article-pdf/37/8/1373/2531738/neco_a_01769.pdf.

[77] Arjun Bharioke et al. "General anesthesia globally synchronizes activity selectively in layer 5 cortical pyramidal neurons". en. In: *Neuron* (Apr. 2022). ISSN: 0896-6273. DOI: 10.1016/j.neuron.2022.03.032.

[78] Nikola T. Markov et al. "Cortical High-Density Counterstream Architectures". In: *Science (New York, N.Y.)* 342.6158 (Nov. 1, 2013), p. 1238406. ISSN: 0036-8075. DOI: 10.1126/science.1238406. pmid: 24179228.

[79] Jordan Guerguiev, Timothy P Lillicrap, and Blake A Richards. "Towards deep learning with segregated dendrites". In: *elife* 6 (2017), e22901.

[80] Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake A Richards, and Richard Naud. "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.

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

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

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

[84] Manu Srinath Halvagal, Axel Laborieux, and Friedemann Zenke. "Implicit variance regularization in non-contrastive SSL". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 63409–63436.

[85] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

[86] Eric Petit. *HandBrake: The open source video transcoder*. 2003.

[87] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.