Object recognition under occlusion revisited: elucidating algorithmic advantages of recurrent computation

Byungwoo Kang[1,2], Shaul Druckmann[1]*

[1] Department of Neurobiology, Stanford University, Stanford, CA

[2] Department of Physics, Stanford University, Stanford, CA

* Correspondence: shauld@stanford.edu

Abstract

Recurrent connections are ubiquitous in the visual system, yet their role is less understood compared to that of feedforward visual processing. Occluded object recognition, an ethologically critical cognitive capacity, is thought to rely crucially on recurrent processing of visual information. Understanding how occluded object recognition depends on recurrent connections is thus of great importance. Here, using deep neural networks as our main tool, we performed comprehensive, controlled computational experiments to elucidate the potential role and necessity of recurrent computation in this task. While our comparison of neural network architectures revealed that sufficiently deep feedforward networks could easily outperform recurrent ones, contrary to previous studies, at the same time we found evidence that recurrent networks learn to “explain away” occlusion by leveraging information about the occluder, without being explicitly trained to do so. Furthermore, inspired by recent experimental studies, we demonstrated that a deep network model incorporating biologically inspired features was capable of faithfully reconstituting corrupted representations, even for heavily occluded objects, by taking global contexts into account through recurrent iterations. Lastly, we examined how occlusion shapes internal representations of neural networks optimized for object recognition. Although representations of occlusion-trained and non-occlusion-trained networks were nearly indistinguishable for unoccluded images, they exhibited highly divergent representations for occluded images. Our work sheds new light on computational roles of recurrent connections, a pervasive component of biological circuits, and how ethologically relevant conditions may strongly reshape neural representations supporting key cognitive functions.

Introduction

Vision is a key sensory modality for primates, and the study of visual processing has revealed diverse principles of neural computation, circuit organization and function (1, 2). Most classical descriptions of information processing in the visual cortex focus on its feedforward nature (3-5), yet a growing body of literature suggests that recurrent processes also play important computational roles (6-10). In particular, the visual processing of occluded objects—key objects in a scene that are partially obscured from view—has garnered much attention as one of the major tasks for which recurrent computations might be crucial in the visual system. Recognizing occluded objects can be ethologically critical for animals, for instance when identifying predators partially concealed in vegetation.

Evidence for the importance of recurrent computation in recognizing occluded objects has come both from the dynamics of neural responses to such objects and from manipulations of temporal context. First, studies have observed that object-selective responses emerge about 50-100 ms later for occluded stimuli than for unoccluded stimuli across multiple contexts (11-17). Second, backward masking, a condition where a high-contrast noise mask rapidly following an image (typically 25 to 150ms after the image onset) could disrupt potential recurrent processing, has been shown to impair recognition of occluded stimuli much more severely than that of unoccluded stimuli (15, 16, 18). While these types of observations do not necessarily imply that object recognition under occlusion involves recurrent processes within the visual cortex, they do strongly suggest that it involves some form of temporally extended processing.

Here, we performed an extensive, well-controlled computational study into the role of recurrent connections in processing occluded objects, using deep neural networks as our main computational tool. We addressed a number of theoretical questions. First, to what extent, if at all, is recurrent computation superior to feedforward computation for occluded object recognition? Second, to what extent does the intuitive explanation for the utility of feedback processing in occluded object recognition hold: i.e., does the ability to feed back information about the occluding object to “explain away” missing parts of the occluded object improve performance (19-22)? Third, how feasible is it to denoise a high-dimensional representation of visual stimuli, such as that of V4, corrupted by occlusion? This is presumably a more demanding task than simply identifying the category of the occluded object, as the high-dimensional representation may contain more fine-grained visual information about the object, such as its texture and shape. Yet recent experimental evidence suggests prefrontal cortex (PFC) inputs

may be involved in such denoising (17). Fourth, to what extent are visual features that are learned in object recognition affected by occlusion?

Several computational studies have explored the role of recurrent connections in occluded object recognition, finding substantial advantages for recurrent processing compared to strictly feedforward processing (15, 16, 23). Unlike these previous studies, we arrive at a more nuanced answer. Our results show that recurrent computation is not intrinsically superior to feedforward computation on occluded object recognition, as sufficient deep feedforward networks were able to outperform recurrent networks. At the same time, we observed a tendency hinting at a computational advantage unique to recurrent computation: the recurrent models tended to identify occluded objects better when they were trained to also identify the occluding objects before doing so. Such a sequential processing of visual information is difficult or unnatural to implement in standard feedforward networks. We believe that the main sources of difference from previous studies are our more extensive comparisons and controls.

Our findings align with the core intuition behind the utility of recurrent computation for occluded object recognition: that information about the occluding object can be leveraged to "explain away" missing parts of the occluded object as due to occlusion (19-22). We find that even more fine-grained visual representations can be denoised using a model inspired by recent experimental observations. Finally, we show that training on occluded images results in substantial changes in neural network activations, but that these changes may not be revealed without probing the network with appropriate stimuli.

In summary we revisit aspects of occluded object recognition from a computational point of view, shedding new light on why and to what extent recurrent computation might be advantageous and the effect occlusion may have on neural representations of visual stimuli.

Results

A comparison across feedforward and recurrent architectures of occluded object classification

To understand the impact of recurrence on processing occluded stimuli, we performed a large-scale computational experiment in which we trained over a dozen different architectures of artificial neural networks on an occluded object category recognition task (Fig. 1). Due to the lack of an established large-scale dataset consisting of occluded

images, we created a new dataset. It consisted of images of an occluded background object and a foreground object, where the two objects were taken from FashionMNIST, a standard image classification dataset in the machine learning community (24). We first selected four classes (T-Shirt/Top, Pullover, Bag, and Ankle boot) out of the total ten classes in FashionMNIST. We chose these particular object classes since objects belonging to these classes had comparable areas, which made it easier to control occlusion levels. To generate a two-object image with occlusion, we then randomly sampled two FashionMNIST images from the selected classes, applied a random rotation and translation to each individual object, and placed them together on a 50 × 50-pixel square black background. We only kept those images whose occlusion level was between 50% and 75% and ensured that the distribution of the occlusion level was uniform in that range (see Methods). In total, the dataset contained 16 classes, labeling both the foreground- and background-object classes (Fig. 1*A*).

We trained networks on four tasks derived from this dataset (Fig. 1*B*). The first, which we refer to as the full task, requires networks to correctly determine the class of the foreground and background object. The rest of the tasks were designed as controls for different aspects of this task. The second, the foreground-only task, requires the network to determine only the class of the foreground object. The third, the background-only task, requires the network to determine only the class of the background object. The fourth, the unoccluded task, requires the network to determine the class of the foreground and background object, but with unoccluded input. Specifically, the network receives as input not the occluded image but rather two images, one in which only the foreground object is present and another in which only the background object is present. In these images the objects are translated and rotated in exactly the same way as in the occluded image. The foreground- and background-only task controls were be used to more precisely tease apart temporal computations, as explained in more detail below. The unoccluded task kept the same two-object classification but controlled for the existence of occlusion.

We trained three general categories of network architectures: feedforward, recurrent, and top-down recurrent (Fig. 1*C*). The feedforward networks were composed of convolutional layers followed by fully connected readout layers. The recurrent models had ConvLSTM (25) layers, followed by an LSTM (26), and finally fully connected readout layers. The top-down recurrent models similarly had ConvLSTM and LSTMs followed by a fully connected readout layer, but in addition they had connections from higher layers to lower layers.

We trained four feedforward architectures on each task (Fig. 1*D*). The network FF had three convolutional layers and one fully connected layer. The convolutional layers had 64 feature maps in each and the fully connected layer had 1024 units. FF Wide had the same number of layers as FF but twice as many units per layer. FF Wider Taller had twelve

convolutional layers and four times as many units per layer than FF. Finally, FF Weight Shared (WS) Wider Taller had the same number of units and layers as FF Wider Taller, but certain layers shared weights, i.e., had identical connectivity (Fig. 1*D*). The networks had the same architecture across tasks, except for the readout layer. In the full task and unoccluded task, in which the networks were required to predict the classes of both the foreground and background objects, the networks had two fully connected readout layers. In the foreground- and background-only tasks, where only one class was to be predicted, the networks had only one fully connected readout layers.

We trained eleven recurrent network variants which differed in architecture, the time steps in which the class predictions were performed and the number of time steps their dynamics were run for (Fig. 1*D*). The network Rec had three layers of ConvLSTMs with 64 feature maps—the same number as feature maps in the feedforward architecture, each followed by a max-pool, then an LSTM layer with 512 units and finally a readout layer. We chose the LSTM layer to have half the number of units as the fully connected layer of FF because the recurrent networks output one prediction at a time step whereas the feedforward ones produce as output predictions for both objects at the same time. The network was run for two time steps. It outputted the identity of the foreground object in the first time step and the background object in the second. The variants Rec 5 and Rec 10 were run for five and ten time steps respectively, outputting the identity of the foreground object in the next to last time step and the identity of the background object in the last time step. The “Rec Thin” networks contained half the number of units at each layer. Since ConvLSTM and LSTM layers have more parameters per unit than standard feedforward layers, cutting the number of units by half roughly equated the number of parameters between the Rec Thin architectures and the FF architecture above. The “Rec Reverse” networks outputted the object classes in reverse order: the class of the background object in the next to last time step and that of the foreground object in the last time step. This allows a comparison for the utility of identifying the foreground object first. The “Rec Last” networks outputted both object classes simultaneously at the last time step, allowing comparisons to test the utility of sequential predictions. The “Rec Control” network outputted both classes on the very first time step, similar to feedforward networks.

For top-down recurrent networks, there were a large number of possible top-down connection patterns that could be added to a recurrent network to form a top-down recurrent network, e.g., many possible source top layers, target bottom layers, connections from one layer to many, etc. We initially tested a large number of possible combinations and found that in our hands many variants were difficult to train and performed poorly. For clarity we therefore present only the four that we found to be competitive with recurrent networks (Fig. 1*D*): T-D Rec v1 has connections from the

second ConvLSTM layer to the first; T-D Rec v2 from the third ConvLSTM layer to the first; T-D Rec v3 from the third to the second and second to first; T-D Rec v4 from the third to the second (see Methods).

We found networks from both the feedforward and recurrent network architectures that achieved high performance on the full task (Fig. 2*A*). When comparing networks with equal number of layers, however, the recurrent models outperformed the feedforward models. Specifically, FF was outperformed by all the standard recurrent models (i.e., recurrent models that first predicted the foreground then the background image, unlike some of the control recurrent architectures, for more details see architecture descriptions above or in Methods). FF Wide, which has twice as many hidden activations as FF at each layer, performed somewhat better than FF, but still worse than most of the recurrent models.

The recurrent models' superior performance was not due merely to increased number of parameters regardless of temporal iterations, but rather due to their ability to perform recurrent computations. This was highlighted by the performance of Rec Control, which has the same architecture and thus the same number of parameters as the recurrent models but only runs for one time step. Rec Control was outperformed by all the recurrent models as well as the feedforward ones. Also consistent with the above observation, the recurrent models performed significantly better with increased numbers of iterations. For all families of the recurrent architecture (Rec, Rec Reverse, and Rec Last), the models that ran for five time steps outperformed those that ran for two time steps by a clear margin. The same tendency held for the top-down recurrent models as well.

Contrary to previous claims (14-16, 23), however, recurrent computation was not intrinsically superior to the feedforward computation in dealing with occlusion. In particular, a deep, purely feedforward model, FF Wider Taller, clearly outperformed most of the recurrent models and performed nearly as well as the best performing recurrent model, Rec 10.

We found only a weak correlation between performance and raw number of parameters (Fig. 2*A*). Instead, we observed that a distinct property, which we refer to as the "computational depth" of a neural network, is a better predictor. Here we define it to be the maximum number of times an input variable is affine-transformed by the network's weights and then transformed by a nonlinearity until it reaches the output end of the network. For an ordinary feedforward network, its computational depth is simply equal to the number of layers. While for an ordinary recurrent network, it is equal to the sum of the number of layers and the number of recurrent iterations. Performance was highly correlated with computational depth, although it tended to saturate at a depth greater than

eight (Fig. 2*B*). This nicely dovetails with both observations above, since the computational depth can be increased by either recurrent iterations or having more layers.

To more cleanly delineate whether recurrent computations offer advantages at a given computational depth, we considered weight-shared feedforward networks. Feedforward networks typically have different weights across layers, but recurrent networks use the same weights across time. Similarly, weight-shared feedforward networks have the same structure as feedforward networks but the weights at certain layers are forced to be identical to the weights at other layers, mimicking recurrent weight sharing. We found that the standard feedforward network, FF Wider Taller, outperformed its weight-shared counterpart, FF WS Wider Taller, but by a relatively small margin. This suggests that repeating an identical computation over time—a defining characteristic of recurrent networks—has no significant, if any, intrinsic advantage in handling occlusion.

The relation between recurrent architectures and top-down recurrent architectures was more complex. One variant, T-D Rec v2, outperformed the recurrent models, whereas another variant, T-D Rec v1, outperformed them when run for two time steps but not when run for five time steps (Fig. 2*C*). The other two variants, T-D Rec v3 and v4 did not outperform the recurrent models for either choice of recurrent time steps. This demonstrates that performance of a top-down recurrent model is sensitive to the choice of the layers between which top-down feedback exists. In addition, it suggests that top-down feedback does not necessarily improve performance. Similar observations were also recently made in (6) in the context of single-object recognition.

While our discussion so far has focused on the multi-object classification task, we observed similar trends in the background and foreground object classification tasks as well (Fig. S1*B*-*C*). Our observation that the ordering of relative performance across architectures was conserved for the foreground-only task indicates that occluded object recognition is not as unique as was previously suggested for its ability to tease out network architecture effects. In contrast, the performance difference across models in the unoccluded multi-object classification task was smaller than in the other tasks (Fig. S1*D*). This is in line with our expectation that fairly easy tasks will not differentiate between architectures. More importantly, the change in relative performance in this easier task indicates that the differences we find above are not just due to some intrinsic difficulty in training some of the architectures.

Finally, we tested one of the central intuitive notions behind the advantage of recurrent computation: “explaining away”. Namely, that the recognition of a foreground object can feed back into the representation and explain why the parts of the background object that are occluded appear different than the standard form for that object, thereby facilitating

classification (19-22). To study this phenomenon, we compared performance on background object recognition in two contexts: (i) when networks had to recognize a background object, but first recognize a foreground object and (ii) when networks predicted only the background object. In line with the idea of explaining away, we found that recurrent models, but not feedforward models, tended to recognize the background object better when they also had to recognize the foreground object (Fig. 3*A*). In more detail, we compared performance in recognizing the background object between networks trained only to predict the background object and those performing the full task. In the background-only task the recurrent networks predicted the background object in the last time step. In the full task the recurrent networks predicted the background object in the last time step as well, but they also predicted the foreground object in the next-to-last time step. In contrast, the feedforward networks have no temporal activation and therefore performed simultaneous prediction, whether it was the background object only or both objects. We find that almost all the recurrent networks predicted the background object better when required to predict the foreground object at the previous time step in the context of the full task (Fig. 3*A*). In contrast, feedforward networks performed slightly worse on categorizing the background object in the context of the full task (Fig. 3A). We note that Rec Last and Rec Control architectures are recurrent but similar to feedforward models in that they also predict both objects simultaneously and thus should be grouped with feedforward models for this comparison.

Importantly, we did not observe an analogous performance gain in foreground object recognition from first predicting the background object (Fig. 3*B*). This would be expected since the entire foreground object is fully visible; thus, there is nothing to explain away. In addition, it demonstrates that the performance gain in background object recognition when first predicting the foreground object cannot be simply due to having more training targets, as the same benefit did not extend to predicting the foreground object. Note that here we could only compare a smaller number of architectures since we only compare architectures that predict the foreground object in the same time step (the last time step) in the full task and foreground-only task. We did this to avoid the confound of having different numbers of time steps before prediction in the two tasks, which would likely skew the results given the importance of computational depth we found above.

As a further test of the "explaining-away" hypothesis, we compared our standard recurrent networks (Rec architecture), which predicted the foreground object first, with recurrent networks that were identical except that they were required to predict the background object first (Rec Reverse architecture). The standard recurrent models outperformed the

reverse ones, both in the two time step and five time step variants, consistent with the explaining-away hypothesis (Fig. 3*C*).

In summary, these results suggest that recurrent models are capable of "explaining away" missing pixels of the occluded object as due to occlusion, using the information about the occluding object processed at a previous time step. Such a sequential processing of visual information seems difficult or at least unnatural for a feedforward network to learn or execute.

PFC-net: denoising an occlusion-corrupted object representation by a biologically-inspired deep network model

The previous subsection focused on overall performance in object classification tasks under occlusion. A recent study examining V4 representations (17) suggested more specific properties of recurrent processing of occluded images. Namely, over time V4 selectivity to occluded shapes is enhanced, likely through feedback from ventrolateral prefrontal cortex (vlPFC) (17). We therefore hypothesized that the "explaining away" hinted at by the results of our previous section may take the form of feedback inputs whose effect is to change–over time–the initial representation of an occluded object to make it more similar to its representation had it not been occluded, thus facilitating downstream recognition. We refer to this process as reconstituting the unoccluded representation. We sought to test the feasibility of reconstituting by feedback a high-dimensional representation of an occluded object, such as that of V4, using a deep network model.

We designed our model, which we will refer to as "PFC-net", to reflect some key qualitative features of the V4-vlPFC interactions suggested by Fyall et al. (17). First, we assume that representation reconstitution is primarily carried out by a distinct neural circuit (vlPFC), although recurrent processes within the visual cortex may also be involved to some extent. This was motivated by the observation of two distinct peaks in response of selective V4 neurons and peak responses of many vlPFC neurons that tend to occur between them. In our simplified picture, vlPFC reconstitutes a V4 representation corrupted by occlusion and then sends back the reconstituted representation to V4. Second, we assume that vlPFC is given a representation of the foreground object that is well disentangled from the background object. This is suggested by the existence of vlPFC neurons that are highly selective to the occlusion level of the stimulus but do not exhibit much selectivity for the occluded object. The last key feature we incorporate in our model is that vlPFC has its own representation of visual objects, distinct from that of V4.

It seems reasonable to suppose such a representation given the shape-selective neurons observed in vlPFC.

PFC-net takes as its input images of an occluded object and a foreground object and then predicts an occlusion-free (reconstituted) image of the background object (Fig. 4*A*). The occlusion-free output image represents the reconstituted V4 representation, and the foreground object image represents a disentangled representation of the occluder, which we assume is given to vlPFC as mentioned above. We note however that a qualitatively similar picture arises even if we train networks without access to an explicit disentangled occluder representation (Fig. S2*E*). The model consists of two key parts: an inference module that predicts an occlusion-free latent representation of the background object from the two inputs and a generative module that decodes the latent representation into the corresponding raw-pixel image output (Fig. 4*A*).

The intuition for this dual architecture is that it will allow the inference module to improve its current estimate for the latent variable by synthesizing an occlusion-free image from it and evaluating how likely it is given the input images. Such an operation is closely related to the notion of "analysis by synthesis" whereby the brain possesses a generative model of the world and the perception is the search for the best explanation of incoming stimuli in terms of this generative model (27-30). Phrased differently, one can view this dual architecture as performing a context-dependent computation with the occluder being the context in which to interpret the occluded image. This is in line with the computational role typically associated with PFC in guiding flexible, context-dependent behavior (31).

The generative module of PFC-net is a generative adversarial network (GAN), a state-of-the-art deep generative model (32) that was trained on original, unoccluded FashionMNIST images of the four classes that form the basis of our study (see Methods and Fig. S2*A*). The inference module iteratively predicts the latent representation based on the inputs and its previous prediction from the inputs (both the predicted latent representation and the image generated from it), refining its prediction over five time steps (see Methods and Fig. S2*B*).

Over the course of our experimentation with PFC-net we discovered that a crucial factor for its performance was the degree to which the background object was spatially transformed. Though PFC-net was able to deal with foreground object translation and rotation, its performance substantially deteriorated when the background object was randomly translated and rotated from its pose in the FashionMNIST dataset. This contrasted strongly with the object classification tasks we considered above, where deep neural networks accurately identified highly transformed occluded objects. This confirms our intuition that reconstituting a high-dimensional representation of occluded objects may

be a significantly more challenging task than just classifying them. To avoid this difficulty becoming a confound, we report results when the background object was not spatially transformed and only the foreground object was rotated and translated (a dataset we refer to as “original” pose, see Methods). In addition, we considered a more complex architecture that does work with the fully translated and rotated images. In this architecture the spatial transformation is inferred and reversed by separately trained networks (see Methods and Fig. S2*D*).

If PFC-net effectively reconstitutes images, then a decoder that has never encountered occlusion should still be able to correctly classify an occluded background object. Thus, we quantified PFC-net's performance by measuring the accuracy of a classifier that never encountered occluded images when it was given as input the output of PFC-net. This way of quantifying representation reconstitution provides a more meaningful measure of how well semantic features of the occluded object are recovered than more naïve measures such as the pixel-space Euclidean distance, which would be strongly affected by uninformative differences such as pixel values in the black background of images.

We found that decoders that have never encountered occlusion are nevertheless able to classify occluded background objects when they are given images that have been reconstituted by PFC-net, but not when given the original occluded images (Fig. 4*B*). This suggests that PFC-net indeed effectively reconstituted the images. In fact, the level of accuracy achieved was almost as high as that of classifying occluded images by decoders trained directly on occluded images. This was particularly surprising given our expectation that PFC-net output would not generate perfectly realistic images, thereby setting up a much more challenging generalization for the decoder than just generalizing from a training set of occluded images to a test set of comparably occluded images. This high level of accuracy was only slightly improved when we used a decoder trained directly on PFC-net outputs (Fig. S2*C*). We found qualitatively similar results when the decoder had a recurrent architecture (Fig. S2*C*). We note that the absolute levels of accuracy presented here are different from those presented in the previous section due to the different nature of the background image (not rotated or translated).

We observed that most of the reconstitution of representation occurs in the first few time steps (Fig. 4*C*). However, in many cases, refinement continued to occur over time. Visual inspection suggests that the initial representation may be imperfectly reconstituted due to the occluder, but over time the global context assists in converging to the correct object (Fig. 4*C*). In some cases, PFC-net at first incorrectly inferred the category of the occluded object before rectifying its representation over time (Fig. 4*D*). Such improvement of PFC-net representation over time reveals in an interpretable manner how recurrent

computations can facilitate the recognition of occluded objects. This is also reminiscent of delayed object-selective responses to occluded stimuli mentioned above (11-17).

Having trained decoders on unoccluded images and tested them on occluded images above, we performed the converse test, which yielded surprisingly asymmetric results. Across architectures, classifiers trained only on heavily occluded images (occlusion level between 50% and 75%) generalized unexpectedly well to unoccluded images, in fact performing better in this generalization than on the images they were trained upon (Fig. S2*F*). This was surprising given the large shift in visual features caused by occlusion. Indeed, earlier in this subsection, we observed that neural networks do not generalize well from unoccluded to occluded images, resulting in about 20% drop in accuracy (Fig. 4*B*). The fact that neural networks generalize well in the opposite direction suggests that features they learn from heavily occluded images are not specialized to heavy occlusion regimes but generalizable to lighter occlusion levels, while the converse is not true.

Effect of occlusion on internal representations of neural networks

To better understand the asymmetry of generalization across different occlusion levels, we compared the internal representations of networks trained on occluded and unoccluded images. We quantified the representational similarity between networks by performing linear regression from activations from one layer in one architecture, the source model, to another layer of a second architecture, the target model (6, 33) (Fig. 5*A*, see Methods). Specifically, we trained a set of five randomly initialized networks on occluded images and another set of five networks on unoccluded images. Then, for each pair of a source and target neural network type, we trained a linear regression from the source model's activations to the target model's at each layer; we did this separately for occluded and unoccluded images. We then quantified representational similarity by the correlation between the predicted and target activations of the linear regression across held-out images, averaged over pairs of randomly initialized instances. Below, we used the non-occlusion-trained architecture as the source model and the occlusion-trained architecture as the target model, but results are qualitatively the same if we reverse the source and target models.

To better interpret this measure of similarity, we need an estimate of the maximum value the measure can take, a "noise ceiling". Previously (6, 33), a noise ceiling for similarity was calculated by the correlation across input images between trial-average responses from two disjoint sets of trials. The noise (and its ceiling) originates from trial-to-trial variability of neural activity in the primate visual cortex. Since our neural networks are

deterministic functions of the input, there is no such trial-to-trial variability. Instead, we used a noise ceiling that originates from the instance-to-instance variability of the random initialization of the network weights. Thus, our noise ceiling was defined to be the average representational similarity between distinct instances of a target neural network type. In a sense, it quantifies how stereotypical learned representations are given the randomness at initialization.

We compared the representations of occlusion-trained and non-occlusion-trained classifiers across our three classes of network architectures—feedforward, recurrent and top-down recurrent, the first two of which were identical to the decoders used in the previous subsection. They were also identical in form to the ones used in the first subsection, but they had more units in convolutional layers to achieve a sufficiently high performance (>70% classification accuracy) on the dataset used for PFC-net. Specifically, the feedforward architecture had 64, 128, and 128 output feature maps in the three convolutional layers, and 512 output units in the fully connected layer. Similarly, the recurrent and top-down recurrent architecture had 64, 128, and 128 output feature maps in the three ConvLSTM layers and 512 hidden units in the LSTM layer. The top-down recurrent architecture's feedback connection was identical to that of T-D Rec v2 in the first subsection, the best-performing top-down recurrent variant. For fully connected and LSTM layers, we used all output unit activations as targets of the linear regression. For convolutional and ConvLSTM layers (which have a much larger number of output unit activations), we randomly sampled 256 units from each such layer of a target architecture to calculate representational similarity (see Methods). The images used had both foreground and background objects transformed (see Methods). Results in the original pose condition (only foreground image transformed, see Methods) are provided in Fig. S6.

We found that the similarity of internal representations in the feedforward network strongly depended on the type of test images used (Fig. 5*B*). For unoccluded images, occlusion-trained networks had as similar representations to non-occlusion-trained ones as to separate instances of themselves. In contrast, for occluded images, we find significant representation dissimilarity between networks trained with and without occlusion. While it is not surprising that networks trained with or without occlusion are dissimilar given the differences in their training, we feel compelled to emphasize that their appearing almost identical for unoccluded images highlights the importance of the test input one uses to evaluate similarities of neural networks.

In other words, the fact that two neural networks share similar representations for one set of inputs does not necessarily imply they will continue to do so on another set of inputs. In addition, we found that the intermediate representations of networks trained with or

without occlusion tended to be more distinguishable toward higher layers in a gradual manner (Fig. 5*B*). This tendency suggests that the presence of occlusion during neural network training primarily affects weights at higher layers. The results for the recurrent networks displayed similar features (Figs. 5*C* and S4). As in the feedforward case, the occlusion-trained and non-occlusion-trained classifiers had significantly more distinct representations for occluded images than for unoccluded images. Also, in parallel with the feedforward case, the representations became progressively more distinguishable toward higher layers. Counter to our expectation, however, the distinguishability of the representations did not tend to change as much over time.

The top-down recurrent architecture exhibited similar trends (Fig. S*5*). Our expectation was that top-down feedback might make the representations of the occlusion-trained and non-occlusion-trained classifiers more distinguishable at lower layers. We found a small, yet noticeable enhancement in the distinguishability of the representations at lower layers, including the first layer that directly received top-down feedback. The fact that the distinguishability in the lower layers was not as high as higher layers from which top-down feedback is received can be explained by the top-down feedback mostly transmitting components of the representation that are more similar across occluded and non-occluded training.

In addition to occlusion, we also examined the effects of spatial transformations, such as translation and rotation, on representations learned by neural networks (Fig. S8). In parallel with the above results, representations of transformation-trained and non-transformation-trained classifiers tended to be more distinguishable on transformed images and in higher layers (Fig. S8*A* and *C*). Also mirroring the case of occlusion (Fig. S7), the generalization from transformed images to non-transformed images was easy, but the generalization in the reverse direction was not (Fig. S8*B* and *D*). In particular, we note that in our setup, the generalization from unrotated to rotated images seems significantly harder than that from unoccluded to occluded images. Though controlling for difficulty across distinct types of generalizations is challenging, systematically studying neural networks' ability to generalize across distinct types of variations in object appearance and how their representations are affected by such variations is an interesting future direction.

Overall, our results show that the choice of inputs is crucial in comparing computations performed by different neural networks. The inferred similarity of their computations

depended on the inputs used to probe them, and therefore differences in representations may be best dissected through judicious choice of inputs that reflects their training history.

Discussion

There has been significant progress in understanding cortical responses in the context of recognizing single, unoccluded objects (6, 8, 33-35), but occluded object recognition has remained more elusive. One outstanding tension between recent studies of single object recognition and occluded object recognition regards the utility of recurrent computations: even though deep feedforward networks have achieved state-of-the-art performance in various visual tasks including single object recognition, multiple studies claimed an apparent superiority of recurrent networks in occluded object recognition. Resolving this tension and better understanding the significance of recurrent computations in the context of occluded object recognition were primary motivations for our present work.

Our results demonstrate that feedforward architectures are not intrinsically worse at occluded object recognition. At the same time, recurrent architectures have advantages. For example, we observed that “explaining away”—the intuitive notion by which recurrent architectures detect the presence of an occluder and then feed back that information to improve identification of the occluded object—arises naturally in our experiments and yields a modest increase in performance. Moreover, inspired by recent experimental results on the role of V4-vlPFC interactions in assisting occluded object recognition (17), we examined whether it is computationally viable to reconstitute a high-dimensional representation of occluded stimuli. This is a much more computationally demanding task than merely identifying the category of an occluded object, as the high-dimensional representation may encode richer information about the object such as its texture and detailed shape. Nonetheless, our biologically inspired architecture, PFC-net, was able to reconstitute occluded images in a manner that reveals recurrent feedback's potential role in bringing global context into consideration. Lastly, we studied how occlusion may alter the internal representations of a neural network trained to perform an object recognition task. This was an important issue to clarify, since the recent works that argued for the similarity between the artificial neural network and primate visual cortex rely on comparisons of such internal representations.

To the best of our knowledge, a single study has employed recurrent and feedforward models with comparable architectures that were fully trained on occluded images (23). Yet the dataset used in this study consisted of computer-generated digits that lacked

intra-class variability in shape, which may fail to reflect the full complexity of occluded object recognition. In addition, the models considered in the study, all having only two hidden layers, may have been insufficiently deep for their results to generalize to deeper networks that are more relevant for comparison to the visual cortex (33). Other previous studies employed deeper networks and more realistic images, and are thus perhaps less prone to the above concerns. However, their experimental design was not well suited to comparing task performance across network architectures. For example, feedforward models were either not trained on occluded images (15) or the feedforward and recurrent models did not have comparable architectures (16). Moreover, in these studies, occlusion was typically generated by simple geometric masks, which may cause their visual recognition tasks to be qualitatively different from that occurring in the real world, where occlusion is often generated by visually complex objects. By addressing these shortcomings, our approach facilitated fairer and more thorough comparisons across different architectures.

A comparison across feedforward and recurrent architectures of occluded object classification

Our results suggest a nuanced view on the significance of recurrent connections for classification of occluded images. On the one hand, unlike some previous studies (15, 16, 23), we do not find a striking difference in the level of performance achievable on occluded classification between recurrent and feedforward architectures if number of units and parameters is not a factor. On the other hand, for the same number of parameters, recurrent architectures offered far better performance, meaning that they were more efficient in solving the task.

The key intuition behind the utility of recurrent connections lies in their facility with "explaining away". We found this feature emerged in our results even when networks were trained end-to-end in a fully agnostic and unstructured way. Notably, we even found a difference in performance that was consistent with explaining away when comparing the standard recurrent architecture to a recurrent architecture that predicted the two objects in reverse order. This suggests a focus not just on architectures per se, but rather on the temporal structuring of the tasks that networks perform. Though temporal structuring, which requires sequential processing of information, seems difficult or (at least) unnatural for a feedforward network to learn or execute, specialized feedforward architectures that explicitly structure tasks in time can be constructed. Finally, the improved performance when the occluded object is reported after the occluder—or, more generally, when the more complex object recognition is performed

after the simpler one—is likely relevant for understanding the delayed response to occluded and other difficult stimuli in the visual cortex (11-16, 36, 37).

On the other hand, while our analyses provided evidence for the emergence of the explaining- away computation in recurrent networks, it still remains unclear how they specifically perform such computation. While it is notoriously difficult to interpret the computations neural networks perform (38), given the potential significance of the idea of explaining away we predict it will be worthwhile to understand in more concrete terms how the knowledge about the occluder feeds back to inference of the occluded object in a generic recurrent network.

PFC-net: denoising an occlusion-corrupted object representation by a deep network model

Our model, PFC-net, tested the feasibility of implementing "explaining away" for complex objects in a high-dimensional representation space. Though the structure of the model was anchored in recordings from primates, our choice to have the model operate in raw pixel space was a compromise. Ideally, it would have been built upon V4-like representations of occluded stimuli rather than raw pixels. This compromise was prompted by how much is still unknown about the nature of the V4 representation, especially when visual stimuli are occluded; we contend that incorporating incomplete knowledge likely would have only introduced biases. We note, though, that our model is forward compatible: since the V4 representation enters as a given input, it would be straightforward to incorporate a more complete description and rebuild our model once such a description becomes available.

Using the raw-pixel representation had an important advantage in that PFC-net's intermediate representations were interpretable. Through visual inspection, we found that PFC-Net is capable of improving its representation of occluded objects over recurrent iterations. Although it is hard to characterize the strategy PFC-Net employs to do so, as with any other deep neural networks, we speculate that it might leverage its ability to compare its prediction and the actual input: examining the difference between the predicted and actual appearance of the occluded object can help with inferring a more accurate representation of it at the next iteration.

While our modeling assumes that computations needed to reconstitute V4 representations of occluded stimuli occur in vlPFC, it is likely that this assumption holds only approximately. We expect neural computations that compensate for occlusion might also occur within the visual cortex itself (at least for low occlusion levels), given its

ability to maintain invariance in neural responses (e.g. those of Inferior Temporal (IT) cortex) against variation in object appearance. Thus, it will be interesting to study the interplay of occlusion-related computations in the visual cortex and in other areas, including vlPFC. Relatedly, a recent study showed in the context of core object recognition that a top-down feedback from vlPFC to the visual cortex is critical for recurrent processing of “challenging” images, for which IT takes additional time to develop selectivity (39). This might hint at a broader role of recurrent computations in multi-regional circuits in solving challenging visual tasks (40).

Effect of occlusion on internal representations of neural networks

Our comparison between representations of networks trained with occluded and non-occluded images revealed two issues. First, representations become more distinct as one progresses to higher layers. This suggests that good performance on occluded object recognition might be achieved by taking a network trained on non-occluded image datasets and re-tuning just the topmost weights of the network. This is a positive finding from the perspective of training a very deep network; training such networks requires large-scale datasets such as ImageNet (41), but no such large and diverse datasets with occlusion are easily available. We readily admit the notion of only re-learning the topmost layers is not novel and is commonly performed in practice, but our results give support to why this is a sensible strategy at least in the context of occluded object recognition.

The difference we observe in representation similarity across layers suggests that occlusion-induced modulations of neural representations are more likely to be found in higher layers of the visual cortex. We believe this prediction, but we suggest it would be more interesting if it is found to be false. That is, if one found significant occlusion-related modulations in lower layers of the visual cortex (contrary to our prediction), this might hint that learning in the visual cortex or even the overall computational structure of it has more fundamental mechanisms to treat occlusion differently. Alternatively, it might also hint more generally that the visual cortex deviates from our current understanding of deep neural networks.

Second, our comparison revealed that networks with clearly dissimilar representations may nevertheless appear similar under some sets of stimuli. This is of particular importance for comparisons of artificial neural networks to real neural circuits, since we do not have access to the dataset used "to train" (so to speak) the real neural circuit. This finding therefore raises the importance of not just the task and training set used in

training an artificial neural network, but also of the dataset then used for comparison. More specifically, our results suggest that just using natural images of single objects may not be enough for comparing representations since features that may naturally arise when there are multiple objects, such as occlusion, may strongly affect the degree of similarity inferred.

In summary, we performed a comprehensive, realistic evaluation of the potential role of recurrent computation in identifying occluded objects. Our results resolve tensions between the purported computational capabilities of feedforward versus recurrent connections. They also suggest that recurrent networks are capable of leveraging information about the occluder to explain away occlusion thanks to their temporally structured computations, even when they were never explicitly trained to do so. Thus, we set a framework that can be used to structure additional detailed comparisons of neural dynamics during processing of occluded images, how they are shaped by recurrent connections, and the underlying computation.

Materials and Methods

A comparison across feedforward and recurrent architectures of occluded object classification

Image dataset generation

FashionMNIST (24) consists of 70000 28 × 28 grayscale images of fashion products on a black background. It contains 7000 images per each of its 10 classes and is split to a training set of 60000 images and a test set of 10000 images. For each set, we selected a subset of images that belong to 4 classes, T-Shirt/Top, Pullover, Bag, and Ankle boot, yielding a reduced training set of 24000 images and test set of 4000 images. We generated 50000 images of two overlapping FashionMNIST objects for a training set and 10000 for a validation set from the reduced training set of FashionMNIST, and another 10000 for a test set from the reduced test set. Labelling both the foreground and background objects, our dataset contains total 16 classes.

To generate an image of two-overlapping FashionMNIST objects, we randomly sampled a pair of FashionMNIST images, and we zero-padded them symmetrically on all four sides to enlarge their size from 28 × 28 to 50 × 50 (0 corresponds to the black pixel and 1 to the white pixel). We then applied a random rotation and translation to each FashionMNIST image using the grid generator and bilinear sampler of Spatial Transformer Network (42). Spatial Transformer Network regards both the source and target images as grids of pixels and uses normalized coordinates for the pixel locations such that those within the spatial bound of the image are given x and y coordinates between -1 and 1. It parametrizes a 2-dimensional affine transformation from the source to target image by a 2 × 3 matrix acting on these normalized coordinates, which is a concatenation of a 2 × 2 matrix representing a linear transformation and a 2 × 1 matrix representing a 2-dimensional translation. To select a random rotation, we sampled an angle from the uniform distribution between -180 and 180 degrees from which the corresponding 2 × 2 rotation matrix was determined. To select a random translation, we first parametrized it by the radial distance and angle of the polar coordinates, and sampled the radial distance from the uniform distribution between 0.1 and 0.15 and the angle from the uniform distribution between -180 and 180 degrees from which the corresponding 2 × 1 translation matrix was determined. While the rotation angle was sampled independently for each of the two objects, the radial distance and angle for the translation were sampled once for the pair and the corresponding translation was applied to one of them and the diametrically opposite translation was applied to the other. The lower and upper bound for the distribution of the radial distance were chosen to make it easier to control the occlusion level of the resulting two-object image. The

translations of the two objects were restricted to be diametrically opposite for the same reason.

To place the two spatially transformed objects in a single image, we then segmented them from their black backgrounds using an ad hoc algorithm. The algorithm first derives a rough segmentation mask by thresholding the input image at the pixel value of 0.01. The threshold value was chosen to ensure that the resulting mask contains the whole object while not including too many background pixels; if the threshold is higher than this value, the mask tends to miss dark regions of the object and if it is lower, the mask tends to include too many background pixels that are not perfectly black. To get rid of a small number of background pixels included in the rough mask, the algorithm calculates the mean pixel value within the rough mask and thresholds the input image at 10% of this mean pixel value, producing a more accurate mask. The segmentation masks of the two objects from this algorithm were used to overlay one over the other, resulting in an image of the two objects overlapping with each other. Its occlusion level was defined as the ratio of the area of the overlap between the two masks to the area of the background-object mask.

Our dataset only includes such images whose occlusion levels range between 50% and 75%. We ensured that the distribution of the occlusion level in our dataset is uniform such that it contains an equal number of images in each 5% bin in that range. We also ensured that, in each 5% bin, there exists an equal number of images for each of the 16 classes.

For the unoccluded task, where the input is two images, one in which only the foreground object is present and another in which only the background object is present, we concatenated these images in the channel dimension.

Hyperparameter search and neural network training

For each model, we performed 90 random searches to tune hyperparameters. Each random search began by randomly sampling a learning rate, an L2 weight decay coefficient, a batch size, and a dropout probability from the following distributions: the log-uniform distribution in [1e-4, 5e-3] (learning rate), 0 with 15% probability or a value sampled from the log-uniform distribution in [1e-12, 1e-3] with 85% probability (L2 weight decay coefficient), the discrete uniform distribution over {32, 64, 128} (batch size), and the uniform distribution in [0.2, 0.7] (dropout probability).

Once the hyperparameters were selected, the model was trained using the Adam optimizer, a standard variant of the mini-batch gradient descent algorithm, with its default hyperameters (43) and its classification accuracy (the total classification

accuracy, when there were two class labels to predict) on the validation set was recorded after each epoch of training. The learning rate was decayed by a factor of 2, if the number of 'bad' epochs in which the validation set accuracy did not improve by more than 0.5% was greater than 4. The number of bad epochs was reset to 0 when a more than 0.5% improvement occurred, and once the learning rate was decayed, a cooldown period of 4 epochs had to be waited before it could be decayed again. If either the validation set accuracy did not reach 60% in the first 10 epochs or the number of bad epochs reached 10, the search stopped immediately, or otherwise, the search continued for 100 epochs. Once the search was completed, the best validation set accuracy thus far was recorded and the corresponding model was saved for the current set of hyperparameters. After 90 such random searches, we identified the hyperparameters with the highest validation set accuracy.

After identifying the optimal hyperparameters for each model, we trained and saved five instances of each model with its optimal hyperparameters exactly in the same way as above. The performances of the five saved instances on the test set were used to calculate the mean and standard deviation of each model's performance.

In all the feedforward models, all the weights were initialized by the Kaiming initialization (44) with the factor of the square root of 2 removed from the standard deviation of the Kaiming normal distribution when the weights were not followed by a ReLU. All the biases were initialized to 0.

In all the (top-down) recurrent models, all the weights were initialized by the normal distribution whose mean is 0 and standard deviation is the inverse of the square root of the number of input units. All the biases were initialized to 0, except for the forget biases of the (Conv)LSTMs, which were initialized to 1, following the common practice (45). The initial hidden states of the (Conv)LSTMs were initialized to 0 and learned in the same way as the other parameters.

<u>Details on feedback connections in the top-down recurrent models</u>

In all feedback connections, we first performed upsampling, to match the difference in spatial scale between these layers (due to max-pooling), and applied a 1×1 convolution whose number of output feature maps equals the number of feedforward feature maps incoming to the target layer of the feedback connection. The output feature maps of the 1×1 convolution were then concatenated in the channel dimension with the incoming feedforward feature maps before being fed into the target layer.

PFC-net: denoising an occlusion-corrupted object representation by a biologically-inspired deep network model

Image dataset generation

The datasets used in this subsection are very analogous to the one described above. The only difference is in the implementation of spatial transformations applied to FashionMNIST images. Instead of first zero-padding 28 × 28 images and then transforming them, we directly transformed them to 50 × 50 images using Spatial Transformer Network (42). Since we did not zero-pad source images, we needed to apply scale transformations to them as well as rotations and translations. For each source image, the scale factor was always fixed to 2.2 (this is the factor by which the object in the source image shrinks after the transformation), the rotation angle was sampled from the uniform distribution between -180 and 180 degrees, and the x- and y-translations were sampled from the uniform distribution between -0.55 and 0.55. Unlike in the above dataset, the translations for the foreground and background objects were not coupled. A 2 × 2 matrix was formed by multiplying the scale factor with the rotation matrix corresponding to the sampled angle, and concatenated with a 2 × 1 matrix representing the sampled x- and y-translations to yield a 2 × 3 affine transformation matrix for Spatial Transformer Network. The spatially transformed individual-object images were segmented and composed in exactly the same way as above.

The core dataset used for the comparison of PFC-net and end-to-end trained classifiers consists of images whose occlusion levels range between 10% and 60%, with an equal number of images in each 10% bin in that range (50000 training and 10000 test images). The occluded dataset used in the occluded-to-unoccluded generalization consists of images whose occlusion levels range between 50% and 75%, with an equal number of images in each 5% bin in that range (50000 training and 10000 test images). The unoccluded test set consists of unoccluded background-object images (i.e. the images of background objects before being occluded by the foreground objects) from test images in the core dataset.

Hyperparameter search and neural network training

For all models other than the generative module of PFC-net, the learning rate of 1e-4 and batch size of 64 were used, the L2 weight decay coefficient was always set to zero, and the dropout probability was set to 0.5 if a model contained a dropout layer. Each model was trained for 100 epochs without a learning rate decay. The model parameters were initialized in exactly the same way as in the comparison study.

The generative module of PFC-net was trained following a standard training practice proposed in (46): the learning rates for both the generator and discriminator were 2e-4, the momentum hyperparameter of the Adam optimizer was set to 0.5, the batch size was 64, and the number of training epochs was 25. Its weights were initialized by the normal distribution with mean 0 and standard deviation 0.02 and biases were initialized to zero, also as suggested in (46).

We trained five instances of each model with randomly initialized parameters, and their performances on the test set were used to calculate the mean and standard deviation of each model's performance.

Details of the architectures

We considered two versions of PFC-net: one in which the background object is in its original pose (as in FashionMNIST dataset) and only the foreground object is randomly rotated and translated ("original pose" condition), and one in which both of them are randomly transformed but the transformation is inferred and reversed by separately trained networks ("predicted pose" condition).

In the original pose condition, the 50×50 images generated as described above were transformed back to the original pose of the background object and downsampled to 28×28 using Spatial Transformer Network (42). The resulting images of an occluded background object were used as one of the inputs to PFC-net, or "x" mentioned above. Also, for each of the 50×50 images, We applied the same transformation and downsampling to the corresponding foreground-object-only image, to obtain the other input to PFC-net, "x_f".

In the predicted pose condition, a separately trained network, "Disentangler", first disentangled the images of a foreground and background object into foreground- and background-object-only images. Then, another network, "PoseNet", inferred the transformation applied to the background object, taking as input the background-object-only images predicted by Disentangler. The inferred transformation was applied to the original 50×50 images using Spatial Transformer Network producing x, and similarly to the inferred foreground-object-only images, producing x_f. As in the original pose condition, x and x_f were also simultaneously downsampled to 28×28.

Details of the architectures of PFC-net, Disentangler, PoseNet, and end-to-end trained decoders are described below.

Let FC[N] be a fully connected layer with N output units, Conv[N, k, s] a convolutional layer with N output feature maps, kernel size k, and stride s, TransConv[N, k, s] a

transposed convolutional layer with N output feature maps, kernel size k, and stride s (47), BN a batch normalization layer (48), Dropout[p] a dropout layer with dropout probability p (49), ReLU a ReLU nonlinearity (50-52), LReLU[s] a Leaky ReLU nonlinearity with negative slope s (53), Sigmoid a sigmoid nonlinearity, and Tanh a tanh nonlinearity.

PFC-net:

The generative module of PFC-net is the generator of a GAN (32) trained on original FashionMNIST images of the 4 selected classes (Fig. S2*A*). It takes as input a noise vector sampled from the uniform distribution over the 62-dimensional unit interval. Its architecture is FC[1024]-BN-ReLU-FC[128 × 7 × 7]-BN-ReLU-TransConv[64, 4, 2]-BN-ReLU-TransConv[1, 4, 2]-Sigmoid. The discriminator of the GAN has the following architecture: Conv[64, 4, 2]-LReLU[0.2]-Conv[128, 4, 2]-BN-LReLU[0.2]-FC[1024]-BN-LReLU[0.2]-FC[1]-Sigmoid. This GAN architecture is inspired by (46).

The inference module of PFC-net is a recursive feedforward network that takes images of an occluded background object and a foreground object, x and x_f, and iteratively predicts a 62-dimensional latent vector for the background object in the input image over 5 time steps (Fig. S2*B*). At each time step t, it is given its prediction of the latent vector from the previous time step, z_{t-1}, and the corresponding image generated by the generative module, $G(z_{t-1})$. $G(z_{t-1})$ is first channel-wise concatenated with x and x_f. The concatenated tensors are then fed into a block of convolutional and fully connected layers, which has the following architecture: Conv[256, 4, 2]-BN-LReLU[0.2]-Conv[256, 4, 2]-BN-LReLU[0.2]-Conv[128, 3, 1]-BN-LReLU[0.2]-FC[512]-BN-LReLU[0.2]. Its output is concatenated with z_{t-1} and then fed into another black of fully connected layers with the following architecture: FC[512]-BN-ReLU-FC[62]-Tanh. The output of the second block is a correction to z_{t-1}, which is added to z_{t-1} and then clamped at 0 and 1 to yield z_t. The input latent vector at the beginning of the iteration, z_0, is the sigmoid of a learned parameter initialized to 0.

Disentangler:

Disentangler consists of an encoder, decoder, and classifier. The encoder encodes the input image x into a 512-dimensional latent vector and has the following architecture: Conv[128, 4, 2]-LReLU[0.2]-Conv[128, 4, 2]-BN-LReLU[0.2]-Conv[128, 4, 2]-BN-LReLU[0.2]-FC[512]. The latent vector from the encoder is split in half, with the first half representing the foreground object in x and the second half the background object. The decoder reconstructs a single-object image of the foreground (background) object from the latent vector of the foreground (background) object and has the following

architecture: FC[64 × 6 × 6]-BN-ReLU-TransConv[64, 4, 2]-BN-ReLU-TransConv[64, 4, 2]-BN-ReLU-TransConv[1, 4, 2]-Sigmoid. The classifier predicts the class label of the foreground (background) object from the latent vector of the foreground (background) object and has the following architecture: FC[128]-BN-ReLU-Dropout[0.5]-FC[64]-BN-ReLU-Dropout[0.5]-FC[4].

PoseNet:

PoseNet consists of a component estimating the scale transformation and translation and another component estimating the rotation applied to the FashionMNIST object in the input image. In our experiment, the input image is a background object image predicted by Disentangler. The first component estimates the three parameters specifying the scale transformation and translation from the input image and has the following architecture: Conv[128, 4 , 2]-BN-ReLU-Conv[128, 4, 2]-BN-ReLU-Conv[128, 4, 2]-BN-ReLU-FC[256]-BN-ReLU-FC[128]-BN-ReLU-FC[3]. The input image is transformed by the inverse of the estimated scale transformation and translation using Spatial Transformer Network (42), and the second component estimates the angle of the rotation from the transformed image. It has the following architecture: Conv[128, 4, 2]-BN-ReLU-Conv[128, 4, 2]-BN-ReLU-Conv[128, 4, 2]-BN-ReLU-FC[256]-BN-ReLU-FC[128]-BN-ReLU-FC[1].

End-to-end trained decoders:

The feedforward decoders have the same architecture as FF in the previous subsection with the following differences: the second and third convolutional layers have 128 output feature maps and the first fully connected layer has 512 output units. Similarly, the recurrent decoders have the same architecture as Rec 5 in the previous subsection with the following differences: the second and third ConvLSTM layers have 128 hidden feature maps. These decoders were either trained on unoccluded images, occluded images, or images predicted by PFC-net. The same architectures were also used for the occluded-to-unoccluded generalization test.

Effect of occlusion on internal representations of neural networks

Image dataset generation

For comparison of occlusion-trained and non-occlusion-trained classifiers, we used the same dataset as in the subsection “PFC-net: denoising an occlusion-corrupted object representation by a biologically-inspired deep network model”. In particular, images in

the original pose condition were identical to those images used in the original pose condition in that subsection. Images in the random pose condition were the original 50 × 50 two-object images in that subsection.

The dataset used for comparison of translation-trained and non-translation-trained classifiers was generated in the same manner as the above dataset, except that we only had one object per image and did not randomly rotate it.

Similarly, the dataset used for comparison of translation-trained and non-translation-trained classifiers was generated in the same manner as the above occlusion dataset, except that we only had one object per image and did not randomly translate it.

Details of calculation of representational similarity

As explained in the main text, we quantified the representational similarity between occlusion-trained and non-occlusion-trained classifiers by performing linear regression from activations of one layer in one classifier (the source model), to those of the same layer in the other classifier (the target model). The representational similarity is defined as the correlation between the predicted and target activations of the linear regression across held-out images, averaged over pairs of randomly initialized instances of the source and target models. Specifically, the correlation between the predicted and target activations of the linear regression in each layer was calculated separately for each target unit. The median of this correlation over units (which was further averaged over pairs of instances, and for convolutional layers also averaged over different random samplings of target units) was taken to be the measure of the representational similarity in that layer.

Acknowledgments

We thank Brett Larsen and Tyler Benster for discussions and Jon-Michael Knapp for comments on the manuscript. This work was supported by the National Institute of Health (NS113110, EB02871) and the Simons Collaboration on the Global Brain (SPI542969).

Author Contributions: B.K. and S.D. designed research; B.K. performed research; B.K. and S.D. wrote the paper.

Competing Interest Statement: The authors declare no competing interest.

References

1. J. S. Werner, L. M. Chalupa, *The Visual Neurosciences* (MIT Press, 2004).
2. J. S. Werner, L. M. Chalupa, *The New Visual Neurosciences* (MIT Press, 2014).
3. T. Serre, A. Oliva, T. Poggio, A feedforward architecture accounts for rapid categorization. *Proceedings of the National Academy of Sciences* **104**, 6424 (2007).
4. M. Riesenhuber, T. Poggio, Hierarchical models of object recognition in cortex. *Nature neuroscience* **2**, 1019-1025 (1999).
5. K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics* **36**, 193-202 (1980).
6. A. Nayebi *et al.* (2018) Task-driven convolutional recurrent models of the visual system. in *Advances in Neural Information Processing Systems*, pp 5290-5301.
7. T. C. Kietzmann *et al.*, Recurrence is required to capture the representational dynamics of the human visual system. *Proceedings of the National Academy of Sciences* **116**, 21854-21863 (2019).
8. J. Kubilius *et al.* (2019) Brain-like object recognition with high-performing shallow recurrent ANNs. in *Advances in Neural Information Processing Systems*, pp 12805-12816.
9. L. Gwilliams, J.-R. King, Recurrent processes support a cascade of hierarchical decisions. *eLife* **9**, e56603 (2020).
10. Y. Mohsenzadeh, S. Qin, R. M. Cichy, D. Pantazis, Ultra-Rapid serial visual presentation reveals dynamics of feedforward and feedback processes in the ventral visual pathway. *eLife* **7**, e36329 (2018).
11. G. Kovács, R. Vogels, G. A. Orban, Selectivity of macaque inferior temporal neurons for partially occluded shapes. *Journal of Neuroscience* **15**, 1984-1997 (1995).
12. K. J. Nielsen, N. K. Logothetis, G. Rainer, Dissociation between local field potentials and spiking activity in macaque inferior temporal cortex reveals diagnosticity-based encoding of complex objects. *Journal of Neuroscience* **26**, 9639-9645 (2006).
13. Y. Kosai, Y. El-Shamayleh, A. M. Fyall, A. Pasupathy, The role of visual area V4 in the discrimination of partially occluded shapes. *Journal of Neuroscience* **34**, 8570-8584 (2014).

14. H. Tang *et al.*, Spatiotemporal dynamics underlying object completion in human ventral visual cortex. *Neuron* **83**, 736-748 (2014).
15. H. Tang *et al.*, Recurrent computations for visual pattern completion. *Proceedings of the National Academy of Sciences* **115**, 8835-8840 (2018).
16. K. Rajaei, Y. Mohsenzadeh, R. Ebrahimpour, S.-M. Khaligh-Razavi, Beyond core object recognition: Recurrent processes account for object recognition under occlusion. *PLoS computational biology* **15**, e1007001 (2019).
17. A. M. Fyall, Y. El-Shamayleh, H. Choi, E. Shea-Brown, A. Pasupathy, Dynamic representation of partially occluded objects in primate prefrontal and visual cortex. *Elife* **6**, e25784 (2017).
18. D. Wyatte, T. Curran, R. O'Reilly, The limits of feedforward vision: Recurrent processing promotes robust object recognition when objects are degraded. *Journal of Cognitive Neuroscience* **24**, 2248-2261 (2012).
19. A. Yuille, D. Kersten, Vision as Bayesian inference: analysis by synthesis? *Trends in cognitive sciences* **10**, 301-308 (2006).
20. N. Kriegeskorte, Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual review of vision science* **1**, 417-446 (2015).
21. H. Tang, G. Kreiman, "Recognition of occluded objects" in Computational and Cognitive Neuroscience of Vision. (Springer, 2017), pp. 41-58.
22. N. C. Rust, A. A. Stocker, Ambiguity and invariance: two fundamental challenges for visual processing. *Current opinion in neurobiology* **20**, 382-388 (2010).
23. C. J. Spoerer, P. McClure, N. Kriegeskorte, Recurrent convolutional neural networks: a better model of biological object recognition. *Frontiers in psychology* **8**, 1551 (2017).
24. H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).
25. S. H. I. Xingjian *et al.* (Convolutional LSTM network: A machine learning approach for precipitation nowcasting. pp 802-810.
26. S. Hochreiter, J. Schmidhuber, Long short-term memory. *Neural computation* **9**, 1735-1780 (1997).
27. D. M. MACKAY, TOWARDS AN INFORMATION-FLOW MODEL OF HUMAN BEHAVIOUR*. *British Journal of Psychology* **47**, 30-43 (1956).
28. P. Dayan, G. E. Hinton, R. M. Neal, R. S. Zemel, The Helmholtz Machine. *Neural Computation* **7**, 889-904 (1995).
29. V. Nair, J. Susskind, G. E. Hinton (2008) Analysis-by-Synthesis by Learning to Invert Generative Black Boxes. (Springer Berlin Heidelberg, Berlin, Heidelberg), pp 971-981.
30. I. Yildirim, T. D. Kulkarni, W. A. Freiwald, J. B. Tenenbaum (2015) Efficient and robust analysis-by-synthesis in vision: A computational framework, behavioral tests, and modeling neuronal representations. in *Annual conference of the cognitive science society*.
31. E. K. Miller, J. D. Cohen, An Integrative Theory of Prefrontal Cortex Function. *Annual Review of Neuroscience* **24**, 167-202 (2001).
32. I. Goodfellow *et al.* (2014) Generative adversarial nets. in *Advances in neural information processing systems*, pp 2672-2680.
33. D. L. K. Yamins *et al.*, Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences* **111**, 8619-8624 (2014).
34. S. A. Cadena *et al.*, Deep convolutional models improve predictions of macaque V1 responses to natural images. *PLoS computational biology* **15**, e1006897 (2019).

35. P. Bao, L. She, M. McGill, D. Y. Tsao, A map of object space in primate inferotemporal cortex. *Nature* **583**, 103-108 (2020).
36. K. Kar, J. Kubilius, K. Schmidt, E. B. Issa, J. J. DiCarlo, Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. *Nature Neuroscience* **22**, 974-983 (2019).
37. C. J. Spoerer, T. C. Kietzmann, J. Mehrer, I. Charest, N. Kriegeskorte, Recurrent neural networks can explain flexible trading of speed and accuracy in biological vision. *PLoS computational biology* **16**, e1008215 (2020).
38. L. H. Gilpin *et al.* (2018) Explaining explanations: An overview of interpretability of machine learning. in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)* (IEEE), pp 80-89.
39. K. Kar, J. J. DiCarlo, Fast Recurrent Processing via Ventrolateral Prefrontal Cortex Is Needed by the Primate Ventral Stream for Robust Core Visual Object Recognition. *Neuron* https://doi.org/10.1016/j.neuron.2020.09.035 (2020).
40. E. Froudarakis *et al.*, The visual cortex in context. *Annual review of vision science* **5**, 317-339 (2019).
41. J. Deng *et al.* (2009) ImageNet: A large-scale hierarchical image database. in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp 248-255.
42. M. Jaderberg, K. Simonyan, A. Zisserman (2015) Spatial transformer networks. in *Advances in neural information processing systems*, pp 2017-2025.
43. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
44. K. He, X. Zhang, S. Ren, J. Sun (2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. in *Proceedings of the IEEE international conference on computer vision*, pp 1026-1034.
45. R. Jozefowicz, W. Zaremba, I. Sutskever (2015) An empirical exploration of recurrent network architectures. in *International conference on machine learning*, pp 2342-2350.
46. A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).
47. M. D. Zeiler, G. W. Taylor, R. Fergus (2011) Adaptive deconvolutional networks for mid and high level feature learning. in *2011 International Conference on Computer Vision* (IEEE), pp 2018-2025.
48. S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).
49. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* **15**, 1929-1958 (2014).
50. V. Nair, G. E. Hinton (2010) Rectified linear units improve restricted boltzmann machines. in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp 807-814.
51. K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, Y. LeCun (2009) What is the best multi-stage architecture for object recognition? in *2009 IEEE 12th international conference on computer vision* (IEEE), pp 2146-2153.
52. X. Glorot, A. Bordes, Y. Bengio (2011) Deep sparse rectifier neural networks. in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp 315-323.
53. A. L. Maas, A. Y. Hannun, A. Y. Ng (2013) Rectifier nonlinearities improve neural network acoustic models. in *Proc. icml*, p 3.

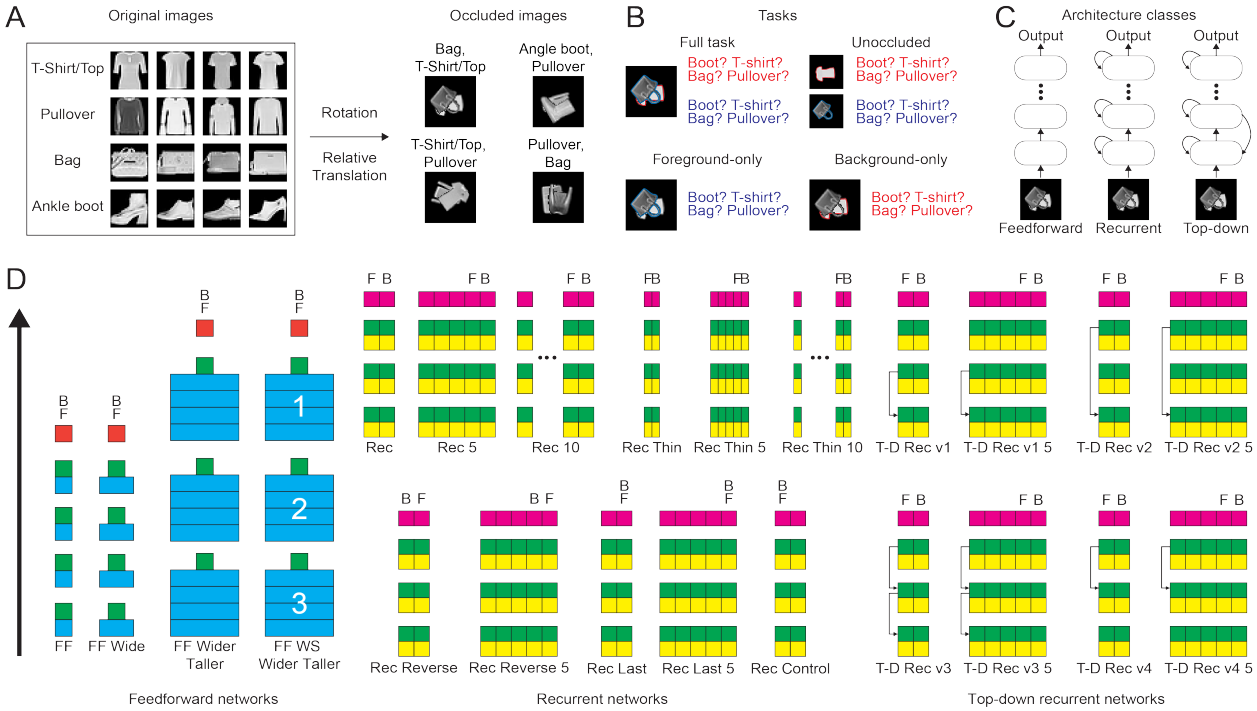

Fig. 1. Task description and network architectures. (*A*) Example images from the fashion MNIST dataset and the dataset used for training. Left: original fashion MNIST images. Each row is a separate class and each column has a different image within the class. Right: images used in our study. (*B*) Schematics of the different tasks. In the full task, the category of both the foreground (occluding) and background (occluded) objects needs to be predicted. In the foreground- and background-only tasks, the class of only one object needs to be predicted. In the unoccluded task, the network receives an input containing each object separately on a black background. (*C*) Schematics of the three architecture classes. (*D*) Schematics of specific architectures. Width of rectangle represents number of units, with the smallest width corresponding to 64 units. Blue corresponds to convolutional layers, green to max-pool operations, red to fully connected layers, yellow to ConvLSTM layers and magenta to LSTM layers. All convolutional layers have a kernel size of 5, and max-pooling a kernel size of 2. Columns indicate number of time steps networks are run (feedforward networks are run for one time step, recurrent and top-down for more). Letters indicate the time step at which a category classification is outputted for the foreground object (denoted by F) and background object (denoted by B). Letters stacked vertically indicate simultaneous output of both categories. Roman numerals indicate weight sharing in the weight-shared feedforward network. (*Left*) feedforward networks. (*Middle*) recurrent networks. (*Right*) top-down recurrent networks.

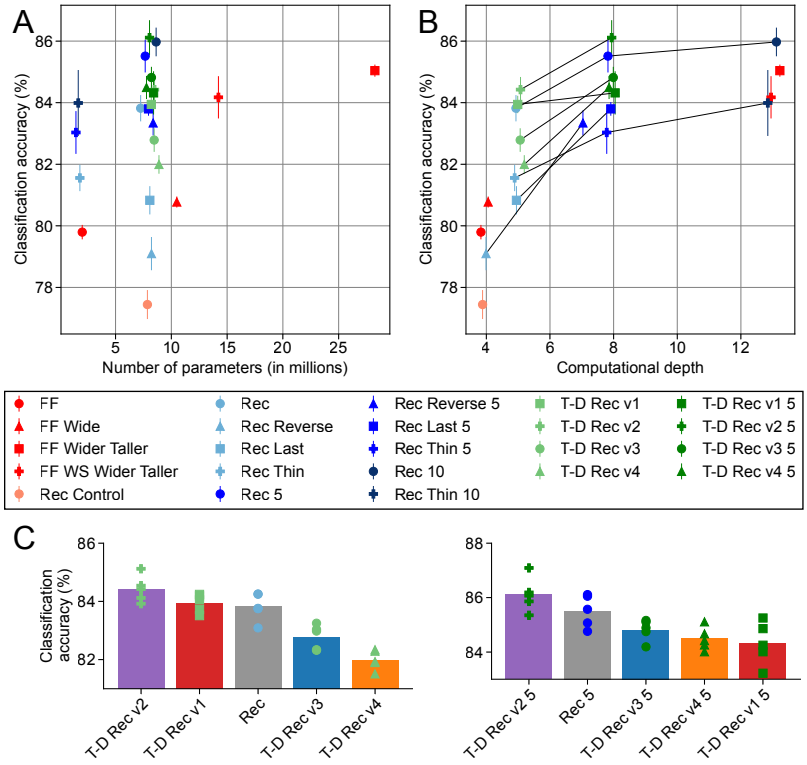

Fig. 2. Comparison of occluded image classification across architectures. (*A*) Task performance vs. number of parameters. Each symbol corresponds to a network architecture, with the average test performance of that architecture across five independent initializations on the y-axis and the number of parameters on the x-axis. The error bars indicate the standard deviations across the initializations. A random Gaussian noise of zero mean and 0.5 standard deviation (in units of millions, consistent with the x-axis) was added to each dot's x-coordinate to separate overlapping dots. For Rec, instead of adding a random noise, we displaced its dot's x-coordinate by 1 toward the left to avoid overlap with other nearby dots. The symbol legend is given in the box below. (*B*) Task performance vs. computational depth. Symbols indicate network architectures as in (*A*). The x-axis corresponds to the computational depth, defined as the maximum number of times an input variable is affine-transformed by the network's weights and then transformed by a nonlinearity until it reaches the output end of the network. The y-axis indicates the average test performance. Lines connect recurrent models of the same architecture that only differ in the number of time steps they are run for. We added a random Gaussian noise of zero mean and 0.15 standard deviation to each dot's x-coordinate to separate out overlapping dots. (*C*) Comparison of task performance of recurrent and top-down recurrent models. n=5 independent initializations for each model. (*Left*) models that run for two time steps. (*Right*) models that run for five time steps.

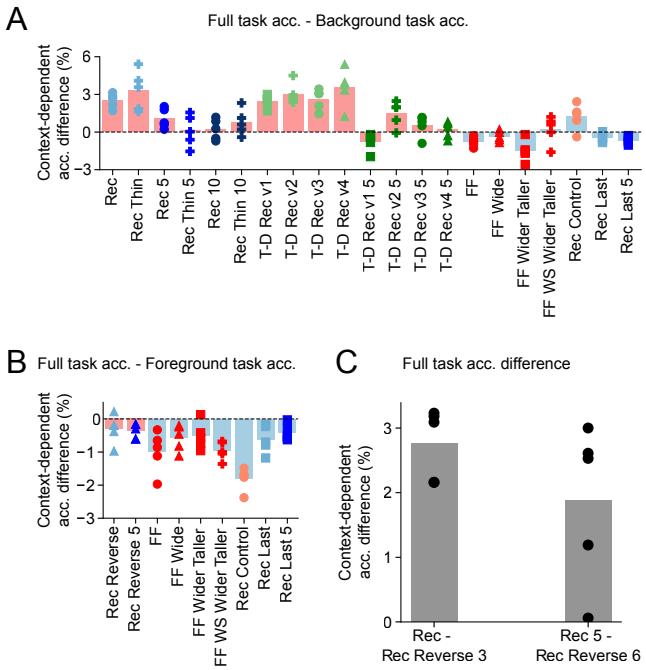

Fig. 3. Evidence for an “explaining-away” computation in occluded object recognition. (*A*) Comparison of classifying the background object in the context of full classification and background-only classification is consistent with explaining away. For each network architecture, we plot the difference between the performance of classifying the background object when it is performed in the context of the full (background and foreground) task and in the background-only task. Positive values indicate better performance in the context of the full task. Bars indicate the average difference across multiple independent networks, symbols show differences between individual networks (n=5). Light red bars indicate network architectures that first output the foreground class and then the background class (e.g. Rec and Rec 5). Light blue bars indicate networks that simultaneously output the foreground and background classes (e.g., feedforward networks). (*B*) Control analysis for "explaining away". Same analysis and plotting conventions as in (*A*), but now comparing the performance of classifying the foreground object in the context of the full task and foreground-only task. Note that the smaller number of architectures is due to the fact that to match the number of processing steps we only take architectures that in the full task predict the foreground object in the last time step. (*C*) The difference in performance of the background object classification between full-task models with identical computational depth. Note that although the two models have identical computational depth, one of them predicts the foreground object first (e.g. Rec) while the other predicts the background object first (e.g. Rec Reverse 3). n=5 independent networks.

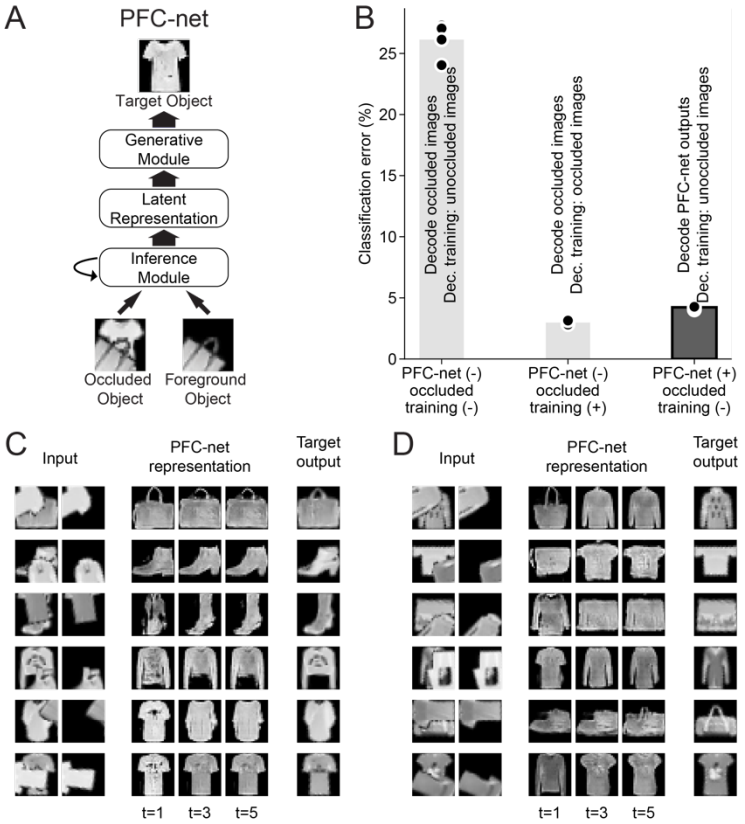

Fig. 4. Representation reconstitution by PFC-net. (*A*) A schematic of PFC-net. PFC-net consists of an inference module and generative module. The inference module takes as input images of an occluded object and a foreground object, and predicts an occlusion-free latent representation of the occluded object through recurrent iterations over five time steps. The generative module maps the predicted latent representation to the corresponding raw-pixel image. (*B*) PFC-net allows successful classification of occluded images by decoders not trained with occlusion. Each bar corresponds to a different combination of network architecture and training. Height of bar indicates the average test classification error of predicting the identity of the background object, and black dots indicate individual independently initialized networks (n=5). Left bar: performance of decoders trained on unoccluded images when classifying occluded images. Middle bar: performance of decoders trained on occluded images when classifying occluded images. Right bar: performance of decoders trained on unoccluded images when classifying occluded images first processed through PFC-net. The performance of a decoder that has never seen occlusion is almost as good as a decoder fully trained with occlusion when given PFC-net output. (*C*) PFC-net tends to refine its representation over time. Left columns indicate the two inputs: original occluded object image and foreground object image. Middle columns show the images corresponding to PFC-net's representations at three time points. Right column indicates the target output for PFC-net, i.e. the image of the background object if unoccluded. (*D*) Similar to (*C*), but examples where PFC-net initially gets the category of the occluded object incorrect and then rectifies it over time.

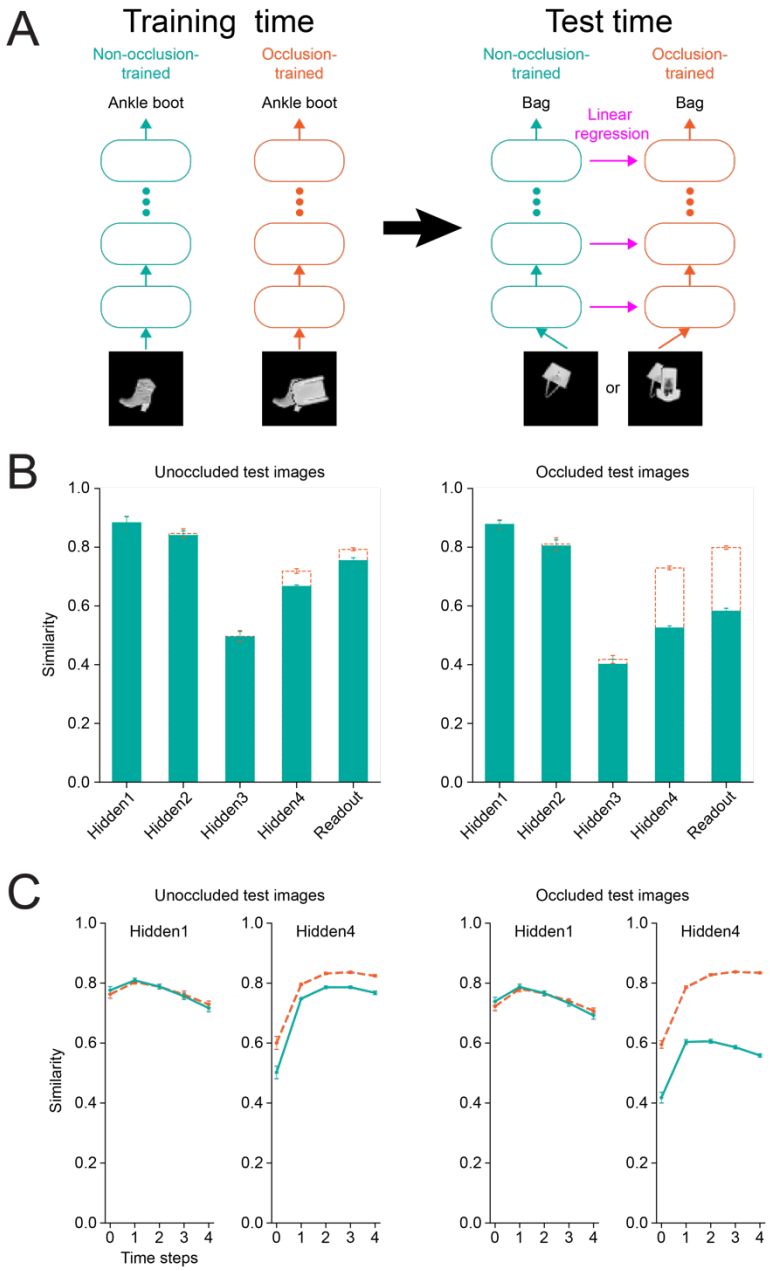

Fig. 5. Degree of representational similarity strongly depends on the type of images in the test dataset. (*A*) A schematic for computation of representational similarity. We trained neural networks of the same architecture to classify either unoccluded images ("non-occlusion-trained" network) or occluded images ("occlusion-trained" network). We then presented identical unoccluded or occluded held-out test images to both networks, and compared their intermediate representations at each layer. (*B*) Representational similarity of occlusion-trained feedforward networks across layers. Dotted orange bars show the average representational similarity of occlusion-trained networks to other occlusion-trained networks. Filled green bars show the average representational similarity of occlusion-trained networks to non-occlusion-trained networks. The error bars indicate the standard deviations across different pairs of randomly initialized instances of the networks (n=25 pairs for the filled green bars, and n=20 pairs for the dotted orange bars, excluding the pairs of the same instances) (*Left*) similarity when assessed on unoccluded images. (*Right*)

similarity when assessed on occluded images. (*C*) Representational similarity of the first and fourth hidden layers of occlusion-trained recurrent neural networks. Dotted orange lines indicate the average representational similarity of occlusion-trained networks to other occlusion-trained networks across time steps. Solid green lines indicate the average representational similarity of occlusion-trained networks to non-occlusion-trained networks. The error bars indicate the standard deviations across different pairs of randomly initialized instances of the networks (n=25 pairs for the solid green lines, and n=20 pairs for the dotted orange lines, excluding pairs of the same instances). (*Left*) similarity when assessed on unoccluded images. (*Right*) similarity when assessed on occluded images.

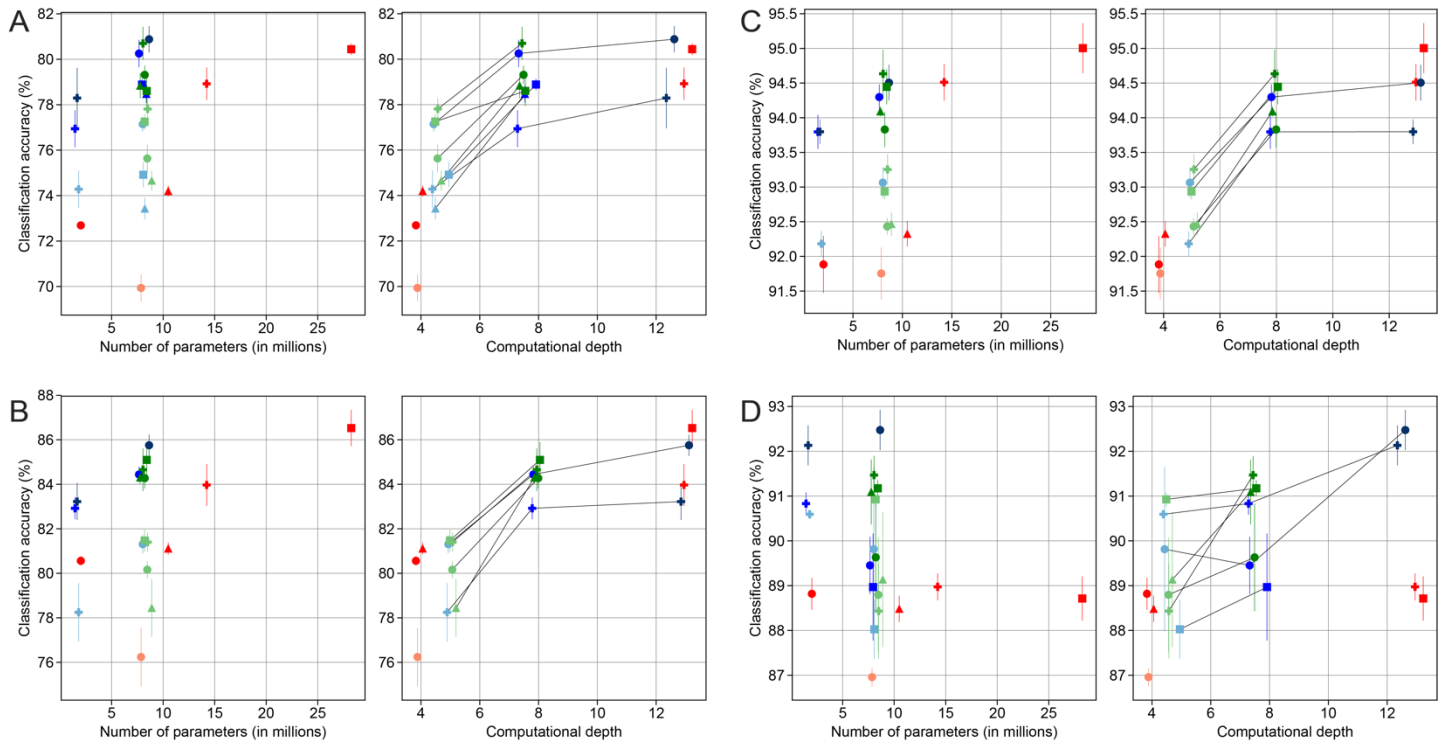

Fig. S1. Comparison of performance in the background-only, foreground-only, and unoccluded task. (*A*) Full task total accuracy vs. number of parameters and vs. computational depth. The total accuracy refers to the accuracy of predicting both the foreground and background object correctly. In case computational depths are different for the foreground and background prediction (e.g. as in the recurrent models that predict the two objects sequentially), we averaged them. n=5 independent initializations for each model. We added a random Gaussian noise of zero mean and 0.5 (for the number-of-parameters plot) or 0.15 (for the computational-depth plot) standard deviation to each dot's x-coordinate to separate out overlapping dots. (*B*) Same plots as in (*A*), but for the background-only task accuracy. (*C*) Same plots as in (*A*), but for the foreground-only task accuracy. (*D*) Same plots as in (*A*), but for the unoccluded task accuracy (the accuracy of predicting both objects correctly).

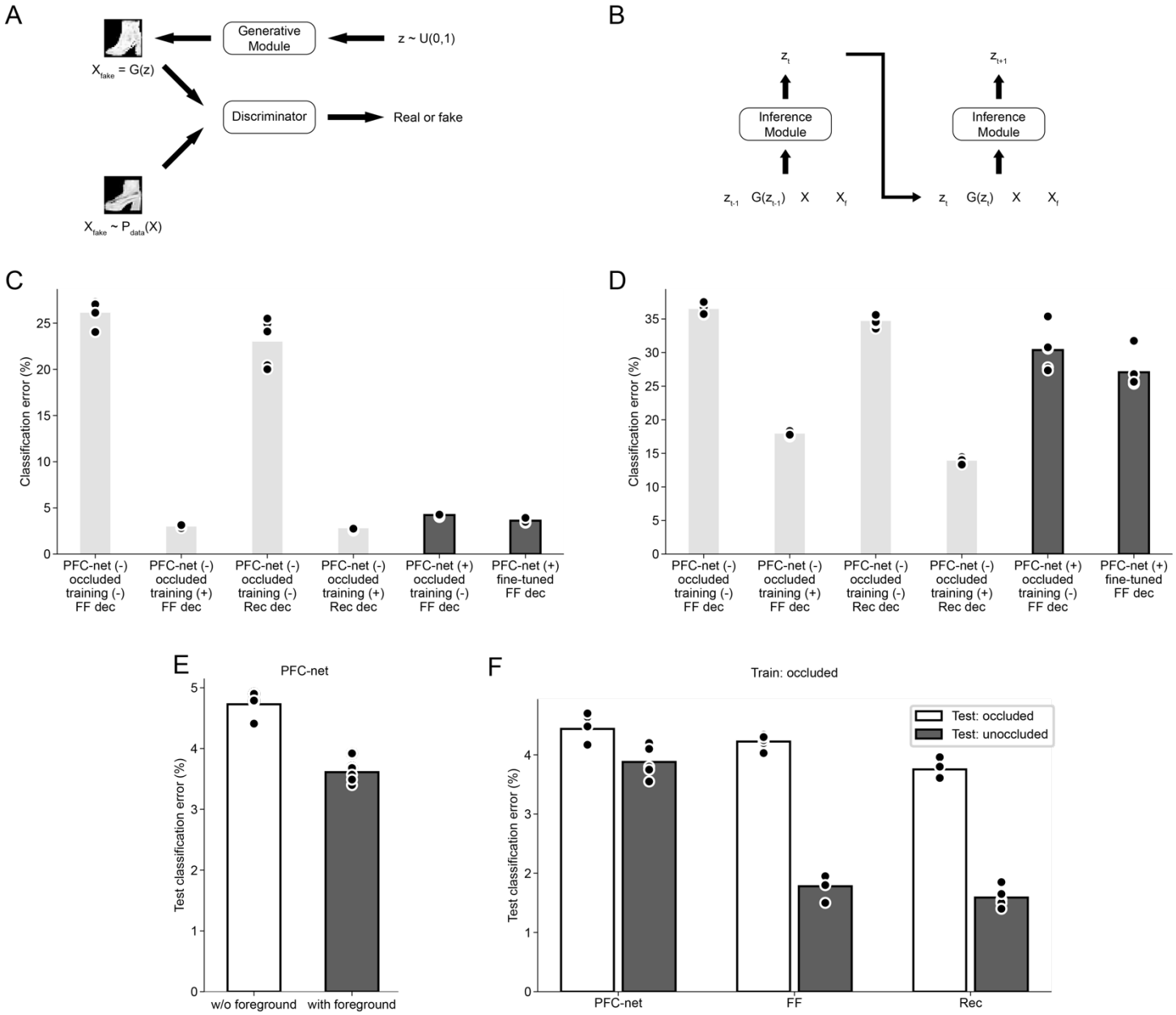

Fig. S2. Further details of PFC-net architecture and performance. (*A*) Schematics of generative module of PFC-net. (*B*) Schematics of inference module of PFC-net. (*C*) Same as Fig. 4*B*, but includes more models. As in Fig. 3*B*, "PFC-net (+ or -)" indicates whether the decoder takes PFC-net outputs or raw occluded images as input, and "occluded training (+ or -)" indicates whether the decoder is trained on occluded or unoccluded images. "fine-tuned" means that the decoder is fine-tuned on the outputs of PFC-net. "FF or Rec dec" indicates whether the decoder is feedforward or recurrent. n=5 independent initializations for each model. (*D*) Same as (*C*) for the predicted pose condition, where the spatial transformations to the background object is inferred and reversed by separately trained networks. (*E*) Comparison of models with and without the foreground object image as an additional input (in the original pose condition). "PFC-net" refers to the PFC-net combined with the downstream (fine-tuned) FF decoder, and "FF" and "Rec" the FF and Rec decoders trained on occluded images, respectively. For each of these architectural classes, we considered two types of models, one which is given the foreground object image as an additional input and one which is not. Their architectures were otherwise identical. n=5 independent initializations for each model. (*F*) Occluded-to-unoccluded generalization. All three classes of models were trained on occluded images whose occlusion level ranged between 50% and 75% (see Methods), and tested either on held-out occluded images or unoccluded images. n=5 independent initializations for each model.

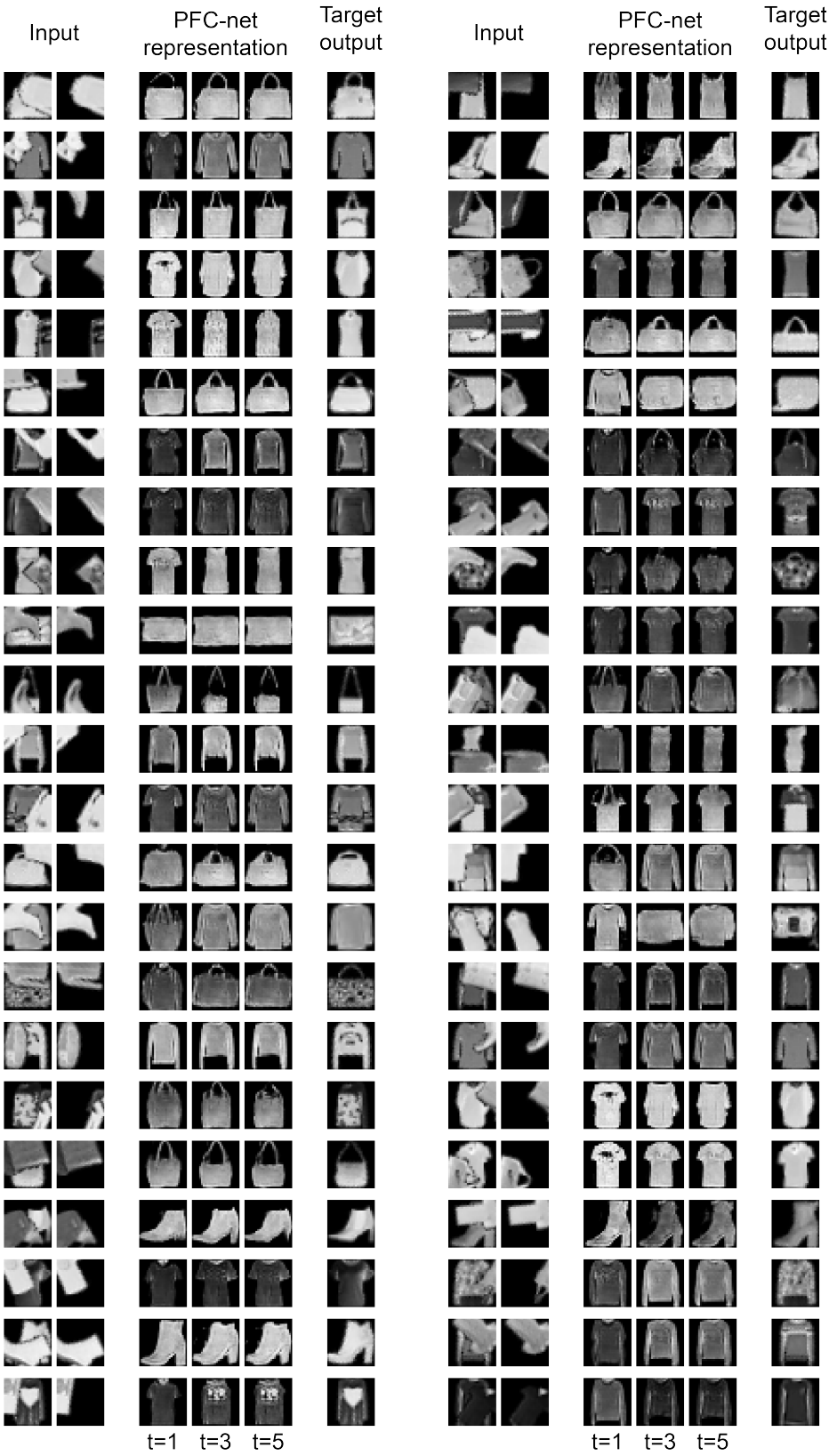

Fig. S3. More examples of PFC-net representation over time.

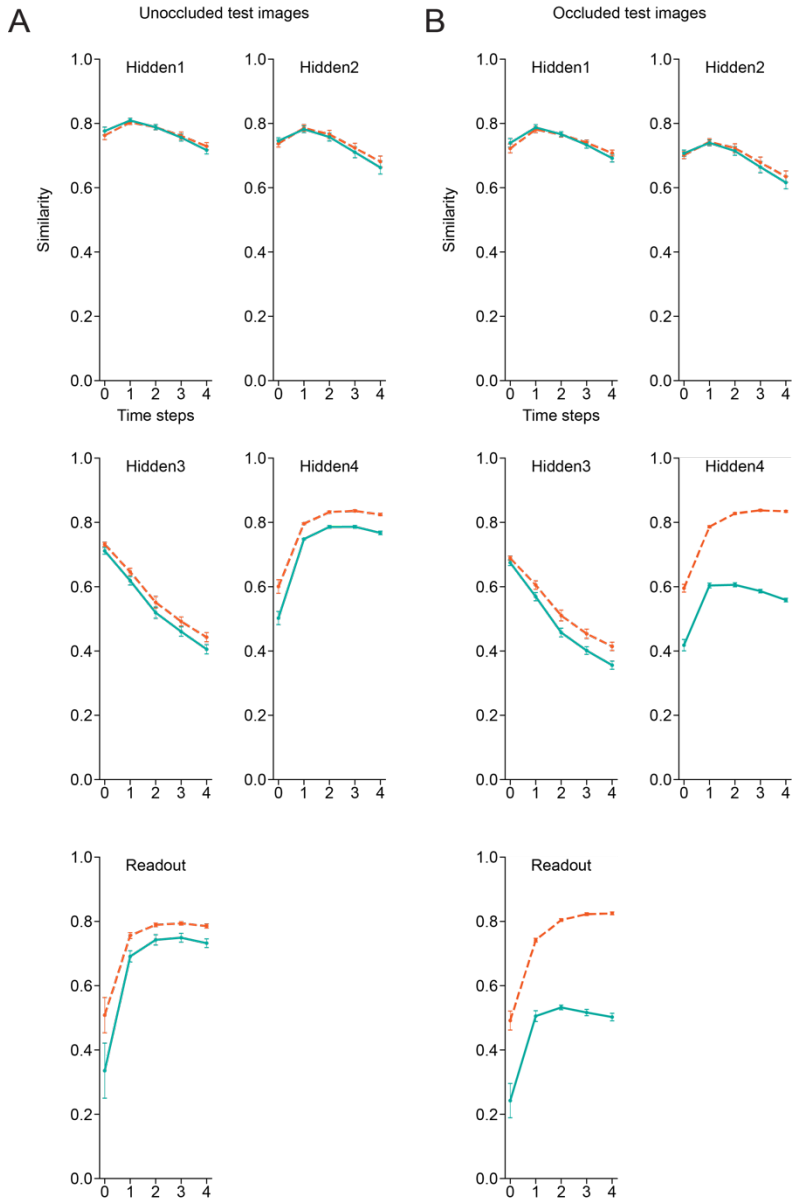

Fig. S4. Representational similarity between occlusion-trained and non-occlusion-trained recurrent networks in the random pose condition. Dotted orange lines indicate average representational similarity of occlusion-trained networks to other occlusion-trained networks across time steps. Solid green lines indicate average representational similarity of occlusion-trained networks to non-occlusion-trained networks. The error bars indicate the standard deviations across different pairs of randomly initialized instances of the networks (n=25 pairs for the solid green lines, and n=20 pairs for the dotted orange lines, excluding the pairs of the same instances). (*A*) When tested on unoccluded images. (*B*) When tested on occluded images.

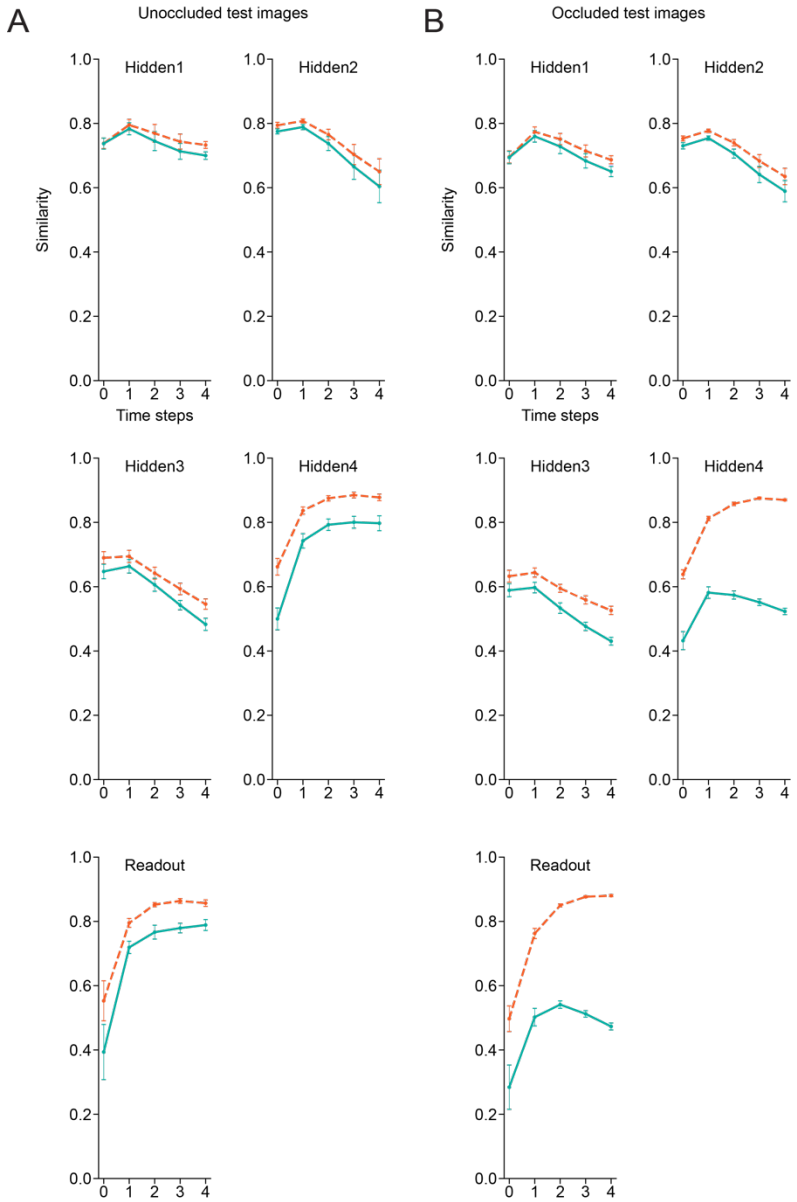

Fig. S5. Similar to Fig. S4 for top-down recurrent networks.

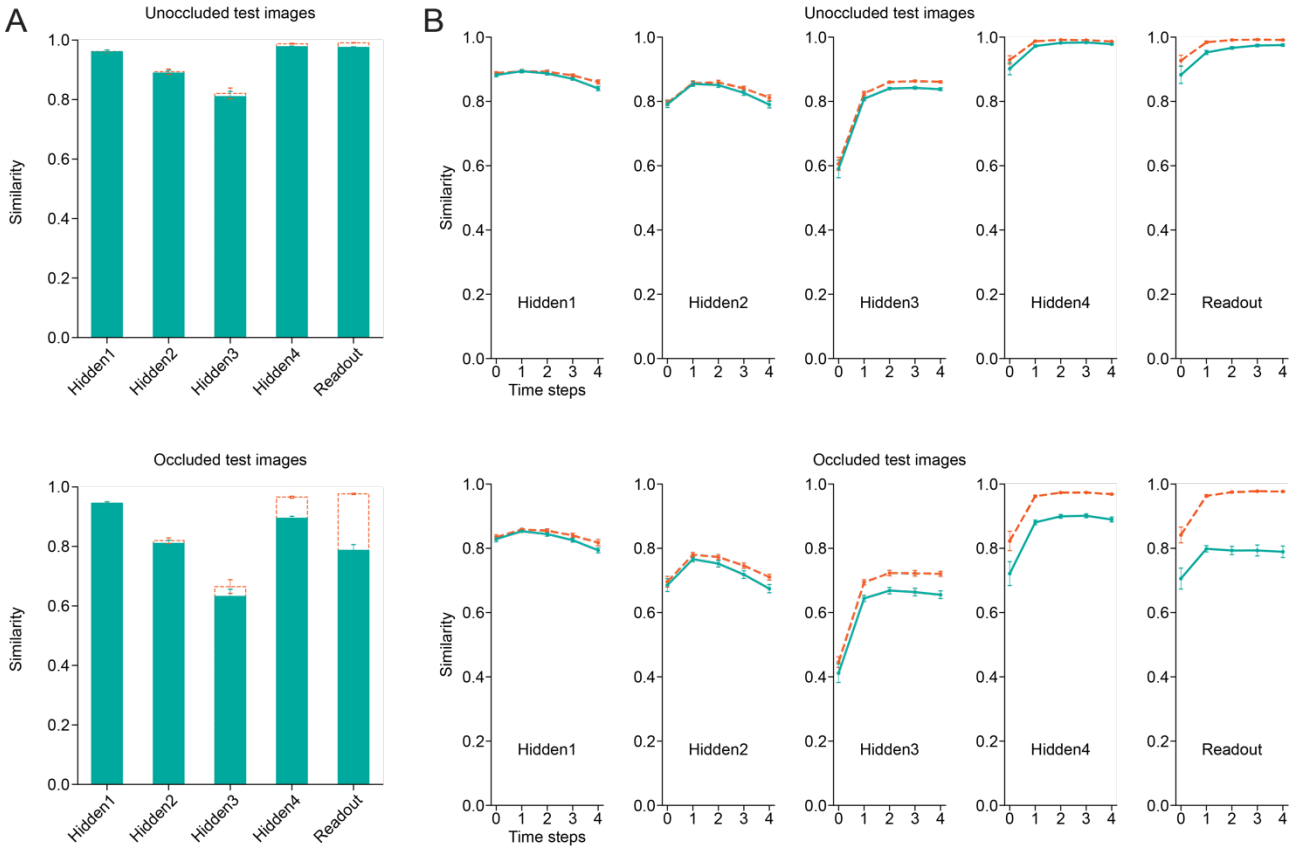

Fig. S6. Same as Fig. 5 for the original pose condition (with additional plots for the second and third hidden layers and readout layer of the recurrent architecture). (*A*) Representational similarity between occlusion-trained and non-occlusion-trained feedforward networks. (*Top*) when tested on unoccluded images. (*Bottom*) when tested on occluded images. (*B*) Same as (*A*) for recurrent networks.

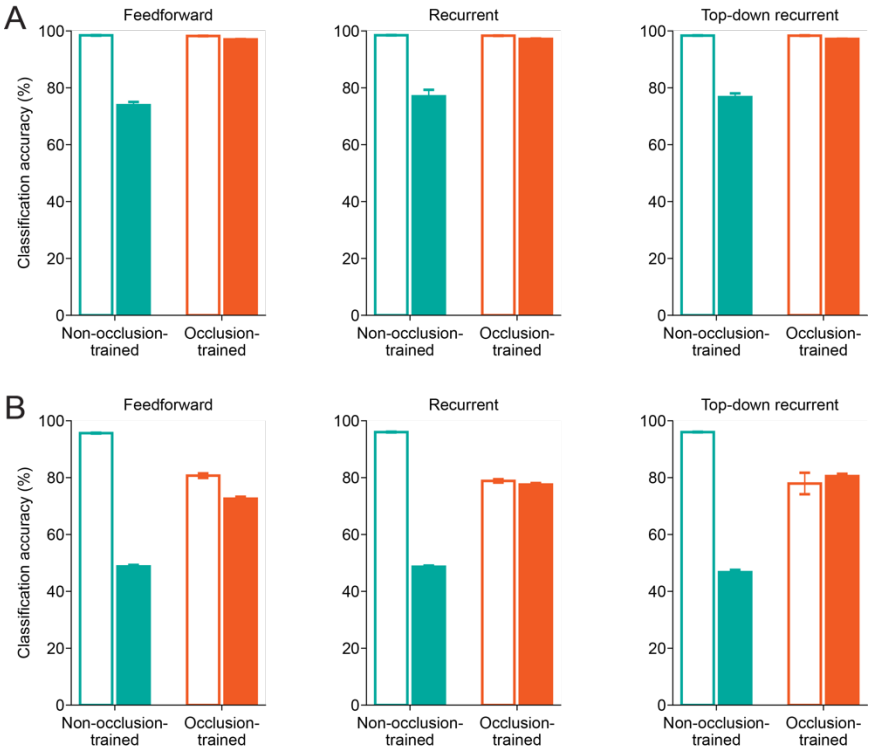

Fig. S7. Generalization performance of occlusion-trained and non-occlusion-trained networks. (*A*) Classification accuracy of occlusion-trained and non-occlusion-trained networks on occluded and unoccluded images in the original pose condition. In each bar graph, the green and orange bars represent non-occlusion-trained and occlusion-trained networks, respectively, and the empty and filled bars accuracy on unoccluded and occluded images, respectively. The error bars indicate the standard deviations across five independent initializations. (*Left*) feedforward architecture. (*Middle*) recurrent architecture. (*Right*) top-down recurrent architecture. (*B*) Same as (*A*) for the random pose condition.

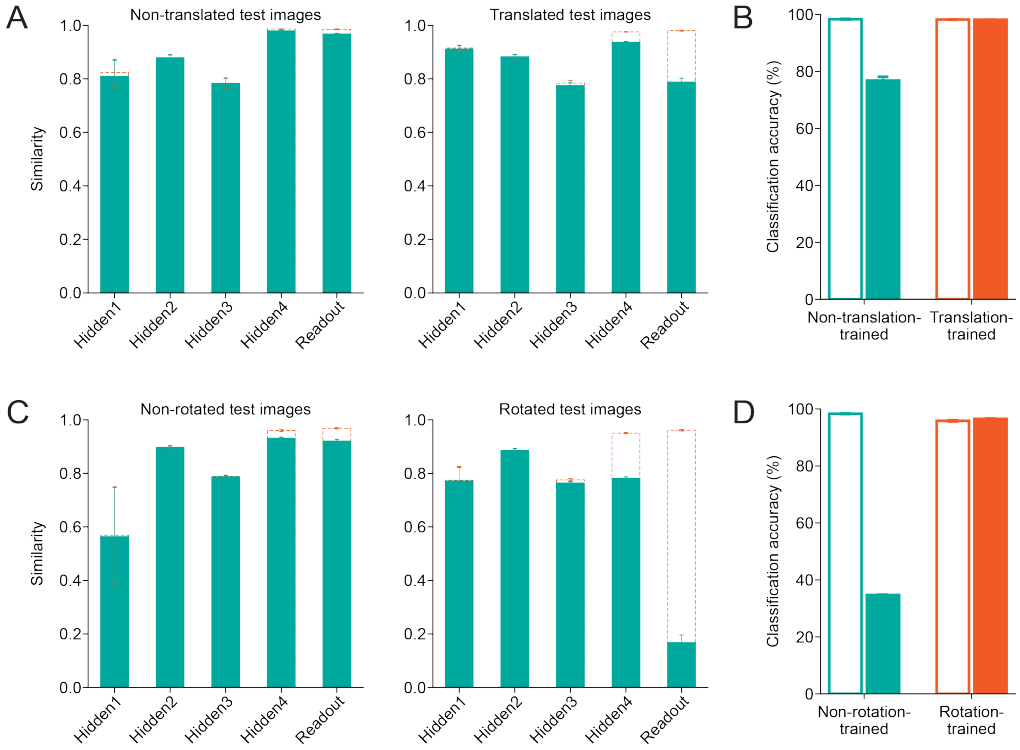

Fig. S8. Effect of spatial transformations on representational similarity. (*A*) Representational similarity between translation-trained and non-translation-trained feedforward networks.Dotted orange bars show average representational similarity of translation-trained networks to other translation-trained networks. Filled green bars show average representational similarity of translation-trained networks to non-translation-trained networks. The error bars indicate the standard deviations across different pairs of randomly initialized instances of the networks (n=25 pairs for the filled green bars, and n=20 pairs for the dotted orange bars, excluding the pairs of the same instances). (*Left*) when tested on non-translated images. (*Right*) when tested on translated images. (*B*) Classification accuracy of translation-trained and non-translation-trained networks on translated and non-translated images. The green and orange bars represent non-translation-trained and translation-trained networks, respectively, and the empty and filled bars accuracy on non-translated and translated images, respectively. The error bars indicate the standard deviations across five independent initializations. (*C*) Same as (*A*) for rotation. (*D*) Same as (*B*) for rotation.