PNAS
www.pnas.org

Supplementary Information for

Unsupervised Neural Network Models of the Ventral Visual Stream

Chengxu Zhuang, Siming Yan, Aran Nayebi, Martin Schrimpf, Michael C. Frank, James J. DiCarlo, and Daniel L. K. Yamins

Chengxu Zhuang.
E-mail: chengxuz@stanford.edu

This PDF file includes:

Supporting Information Text

Methods.

Neural Network Training. We used ResNet-18 (1) without its final pooling layer and the categorization readout layer as visual backbones for all objectives except PredNet. For each objective, we trained three networks with different network initializations. Most objectives were performed by adding an additional header upon the visual backbone and then training the whole network with the objective-specific loss in addition to a L2-regularization loss of the network weights with a weight decay coefficient 10^{-4}. Unless specified, the input image to the networks was in resolution 224×224 and there were two learning rate drops during training. The learning rate was dropped by 10 times after the validation performance saturates. Most training hyperparameters such as the batch size and the initial learning rate were set according to the papers of these objectives. Most of the networks were trained with batch size of 128.

Auto-Encoder. The image (x) was first projected into a 128-dimension hidden vector (h) through passing through a ResNet-18, which produces a 512-dimension vector output, and then a linear projection layer that produced 128-dimension vector output. An output image ($\bar{x}$) was then generated from this vector using another linear projection layer that produced 512-dimension vector output from h and a reversed ResNet-18. The Auto-Encoder loss optimized ($\mathcal{L}_{AE}$) was the sum of the per-pixel average of the L2 different between the output and the input images and the L1-norm of the hidden vector multiplied by 10^{-4}:

$$\mathcal{L}_{\text{AE}} = \frac{\|x - \bar{x}\|_2^2}{224^2} + 10^{-4}\|h\|_1$$

PredNet (2). PredNet network was trained to predict the responses to the next frame using previous frames. This method used a specifically-designed recurrent neural network architecture consisting of three stacked modules, each of which includes four basic parts: an input convolutional layer (A_l), a recurrent representation layer (R_l), a prediction layer ($\hat{A}_l$), and an error representation (E_l) (2). These layers and the loss function were computed as following:

$$\begin{aligned}
A_l^t &= \begin{cases} x_t, & l = 0 \\ \text{MaxPool}(\text{ReLU}(\text{Conv}(E_{l-1}^t))), & l > 0 \end{cases} \\
\hat{A}_l^t &= \text{ReLU}(\text{Conv}(R_l^t)) \\
E_l^t &= [\text{ReLU}(A_l^t - \hat{A}_l^t); \text{ReLU}(\hat{A}_l^t - A_l^t)] \\
R_l^t &= \text{ConvLSTM}(E_l^{t-1}, R_l^{t-1}, \text{Upsample}(R_{l+1}^t)) \\
\mathcal{L}_{\text{PredNet}} &= \sum_t \lambda_t \sum_l \frac{\lambda_l}{n_l} \sum_{n_l} E_l^t
\end{aligned}$$

where $\lambda_t, \lambda_l, n_l$ are correspondingly the weighting factors by time, the weighting factors by layer, and the number of units in the lth layer. These hyperparameters were directly taken from the original paper (2). We used 10 consecutive frames as inputs. As ResNet-18 is a feedforward network, it cannot be used in PredNet. We also found that PredNet failed to train more layers added to each module. Therefore, we used the same network architecture as the original paper (2). Later neural fitting, object position/pose estimation task, and categorization tasks were all performed using each of the twelve layers and we reported the best number across these layers. As the network architecture is very different from others, we cannot show comparable neural fitting layer trajectories. The PredNet model reported in Fig. 2 and Fig. 1 was trained on Kinetics-400 (3).

Depth Prediction. A multi-layer header was added to the visual backbone to output a per-pixel depth image ($\hat{D}$) (4). This depth image was then compared to the ground truth normalized depth image (D), of which the mean was 0 and the standard deviation was 1 within one image. The per-pixel average L2-norm of the difference was used as the optimization loss: $\mathcal{L}_{\text{Depth}} = \|\hat{D} - D\|_2^2/224^2$. This objective was trained on PBRNet, which is a large-scale synthetic dataset containing 0.4 million images which are physically-based rendered from 45K realistic 3D indoor scenes (5).

Contrastive Predictive Coding (CPC) (6). The input image was first resized into 256×256. Then a grid of small crops in the resolution of 64×64 were taken from the resized input image. As these crops were taken every 32 pixels, this procedure generated 7×7 crops. These crops were then fed into the network to generate a grid of low-dimension embeddings in the shape of $7 \times 7 \times 512$. For each column of the grid, a GRU recurrent head (7) was added to take the first five embeddings (e_{0-4}) to generate a final 128-dimension embedding (c). This embedding was transformed into 512-dimension vector as predictions to the final two embeddings ($e_{5,6}$) in the same column. The loss optimized is:

$$\mathcal{L}_{\text{CPC}} = -\sum_{i=5,6} \log \frac{\exp(c^T W_i e_i)}{\sum_{j=0-6} \exp(c^T W_i e_j)}$$

where W_i is in the shape of 128×512. Although this method has "contrastive" in its name, CPC is very different from deep contrastive embedding methods as it predicts the current embedding in the context of the embeddings of some of the other small crops within the same image, while for deep contrastive embedding methods, the "contrastive" usually represents the context of the embeddings of other images.

Colorization (8). The input image was first converted into Lab color space. The L channel, which is in the shape of $224 \times 224 \times 1$ was then used as the input to the network. As regression problem is typically harder to optimize than categorization problem, Zhang, Isola, and Efros developed a quantization method that quantizes the ab output space in to 313 bins (8). They also downsample the ab channels to 28×28 resolution to reduce the computation requirement. As the spatial resolution of usual ResNet-18 outputs before the final spatial pooling is

only 7×7, we changed the convolution operations in the last 4 residual modules to be dilated convolutions, which did not reduce the spatial dimension but increase the dilation step instead. This change increased the spatial dimension of the output from 7×7 to 28×28. Three additional convolution layers were added upon this output to generate a $28 \times 28 \times 313$ prediction, following settings proposed by Doersch and Zisserman (9). The final loss optimized was the cross entropy loss between the prediction and the ground truth ab channels.

Relative Position (10). Two image crops of shape 96×96 were first randomly chosen from a 2×2 grid of image crops generated from the original input image and then fed into the network. The outputs of ResNet-18 towards these two crops were first sent to two bottlenecked residual blocks separately and then concatenated together. The concatenated vector was sent to three additional bottlenecked residual blocks and a final fully connected layer to produce a 8-dimension output as predictions to what spatial relative position these two crops were. The final loss was therefore the cross entropy loss between the prediction and the ground truth relative position label. We followed Doersch and Zisserman (9) for the structures of the additional residual blocks and the methods to sample image crops.

Deep Cluster (11). All training images were first embedded into a 4,096-dimension space. These embeddings were then clustered into 10,000 small clusters using the KMeans algorithm. The index of the cluster one image belonged to was used as a "category" label for this image. A linear layer to generate a 10,000-way category prediction was added upon the 4,096-dimension layer. The loss optimized was the cross entropy loss between the category prediction and the KMeans cluster label. At the end of every training epoch, the embeddings were regenerated for all training images, the KMeans algorithm was used again to cluster the embeddings into 10,000 clusters, and the weights of the added linear layer were also reset to random values as the new cluster labels were different from the previous cluster labels.

Instance Recognition (12). For each original input image $x \in X$, a random data augmentation $v \in V$ was first applied to generate a view of this image $v(x)$ as the input to the network. The network f was used to embed the input into a 128-dimension unit-sphere ($e = f(v(x))$). Conceptually, this objective optimized the network to minimize the following loss:

$$\mathcal{L}_{\text{IR}}^{\text{concept}} = -\log \frac{\exp(f(v'(x))^T e/\tau)}{\sum_i \exp(f(v_i(x_i))^T e/\tau)}$$

As recomputing $f(v_i(x_i))$ for the whole dataset in each step is intractable for large datasets, a "memory bank" mechanism was used to avoid these computations through replacing $f(v_i(x_i))$ and $f(v'(x))$ with their corresponding items in the "memory bank" ($\text{m}(x_i)$ and $\text{m}(x)$), which were running averages maintained across the training. This memory bank was maintained by updating $\text{m}(x)$ with $0.5\text{m}(x) + 0.5e$ after each optimization step. Moreover, s examples were randomly chosen from the memory bank to compute the actual loss defined as below:

$$h(a) = \frac{\frac{\exp(a^T e/\tau)}{Z}}{\frac{\exp(a^T e/\tau)}{Z} + s/n}, a \in \mathbb{R}^{128}$$

$$\mathcal{L}_{\text{IR}} = -\log h(\text{m}(x)) - \sum_j \log(1 - h(\text{m}(x_j)))$$

where Z was a large constant to approximate the denominator $\sum_i \exp(\text{m}(x_i)^T e/\tau)$, n was the number of training images, and j iterated over s sampled indexes. Following Wu et al. (12), we set $s = 4096$, $\tau = 0.07$, and Z as the initial value of the denominator. The data augmentation included random cropping, random horizontal flip, random color jittering, and random color dropping (transforming to grayscale images).

Contrastive Multiview Coding (CMC) (13). The input image was first converted into Lab color space. The L channel (x^{L}) and ab channels (x^{ab}) were then sent into two ResNet-18 networks (f^{L} and f^{ab}) to get the corresponding 128-dimension embeddings (e^{L} and e^{ab}). Similar to the IR method, two separate memory banks (m^{L} and m^{ab}) were maintained as running averages of L channel embeddings and ab channel embeddings. If $\mathcal{L}_{\text{IR}}$ is viewed as a function of the memory bank m and the current embedding e as $\mathcal{L}_{\text{IR}}(m, e)$, the CMC loss is then computed as following:

$$\mathcal{L}_{\text{CMC}} = \mathcal{L}_{\text{IR}}(m^{\text{L}}, e^{\text{ab}}) + \mathcal{L}_{\text{IR}}(m^{\text{ab}}, e^{\text{L}}) \quad [1]$$

Optimizing this loss would push together the embeddings of corresponding L and ab channels and separating them away from the embeddings of other images. We used the L-ResNet18 to evaluate the neural predictivity and the behavior consistency metric, as the stimulus used there are all gray-scale images. We dropped random color jittering and random color dropping from the data augmentation procedure, as they influenced the ab channels too much.

SimCLR (14). The loss optimized by SimCLR was close to the conceptual loss of the IR method. However, it did not use the memory bank mechanism, but used an extremely large batch size so that the images in one batch can be treated as reasonable representatives of the training dataset. More specifically, SimCLR sampled 4,096 images as one batch, noted as $\{x_i\}$. For each image x_i, two data augmentation instances (v_i^0 and v_i^1) were generated. A three-layer Multi-Layer-Perceptron (MLP) was also used to produce the corresponding final embeddings e_i^0 and e_i^1 for these two views, from the ResNet-18 output (the output of the spatial pooling layer). This is different from other deep contrastive embedding methods, which only use one linear layer to produce the final embedding from the ResNet-18 output. The optimized loss is defined as following:

$$\mathcal{L}_{\text{SimCLR}} = \frac{1}{\text{bs}} \sum_{i=0}^{\text{bs}} \sum_{j=0,1} \frac{\exp(e_i^{jT} e_i^{1-j}/\tau)}{\sum_{k=0}^{\text{bs}} \sum_{l=0,1} \exp(e_i^{jT} e_k^l/\tau)}$$

where bs is the batch size, typically 4,096. Through a hyper-parameter grid search, SimCLR also found that a higher $\tau = 0.15$ and a Guassian blur augmentation both helped improve the performance.

Local Aggregation (LA) (15). As described in the main text, this method first identified close neighbors ($\text{C}(x) \subset X$) and background neighbors ($\text{B}(x) \subset X$) for each image x and optimized the following loss:

$$\mathcal{L}_{\text{LA}} = -\log \frac{\sum_{x_a \in \text{C}(x)} \exp(\text{m}(x_a)^T f(v(x))/\tau)}{\sum_{x_b \in \text{B}(x) \cup \text{C}(x)} \exp(\text{m}(x_b)^T f(v(x))/\tau)}$$

$\mathrm{C}(x)$ was identified through aggregating 10 independent KMeans clustering results on the embeddings of all the training images. Each of the KMeans result clustered the embeddings into 30,000 clusters and the aggregation was done through computing the union of the 10 clusters x belonged, i.e. any image that was in the same cluster with x in any of the 10 clustering results would be identified as a close neighbor of x. This aggregation helped avoid the randomness in KMeans results. Given the computation efficiency of KMeans clustering procedure, we only performed the clustering at the end of every epoch. The background neighbors were identified through getting the nearest 4,096 embeddings in the embedding space for $f(v(x))$. They typically also included all the close neighbors, which usually only had 200 to 300 images.

VIE (16). VIE extends the deep contrastive embedding methods into videos through treating samples from the same video at different time points as data augmentation instances. For example, VIE was mostly trained on Kinetics-400 in the original paper (16). This video dataset contains around 240,000 videos, each of which is 10s long. VIE then treated each video as one data example and added the temporal sampling from the same video into the data augmentation functions, which means that one view v could be from the beginning part of the video while another view v' could be from the ending part of the video. As for SAYCam, of which the videos are usually in minutes or even hours, we split them into 10s short videos and treated each of the 10s videos as one data example. This helped VIE leverage the temporal context information in the video to better train networks. The final loss optimized was still the same as $\mathcal{L}_{\mathrm{LA}}$. However, as the number of videos is lower than the number of images in ImageNet, we changed how $\mathrm{C}(c)$ was identified to use only one clustering result with 10,000 clusters. On both SAYCam and Kinetics, we trained two pathways using VIE loss: static pathway with ResNet-18 and dynamic pathway with 3D-ResNet-18, which receives 16 consecutive frames and applies temporal and spatial convolutions (16). The pretrained two pathways were concatenated across the channel dimension in each layer as the final network. When testing on static stimuli, we repeated the images for 16 times and averaged the responses of the dynamic pathway across the temporal dimension.

Mean Teacher (MT) (17). This method maintained another ResNet-18 as a "teacher" network $\hat{f}$, whose weights were not trainable and were running averages of the main ResNet-18 weights across the training. Assuming the weights of the main ResNet-18 as θ and the weights of the teacher network as $\hat{\theta}$, the teacher network was then maintained through updating $\hat{\theta}$ with $0.999\hat{\theta} + 0.001\theta$ after every optimization step. At every step, a batch of images from the labeled dataset as well as a batch of images from the unlabeled dataset were sent to both the main and the teacher networks, from which four category predictions were generated. These predictions were used to define the loss as following:

$$\begin{aligned}\mathcal{L}_{\mathrm{MT}} =& \frac{1}{\|B^L\|}\sum_{x\in B^L}(\mathrm{CE}(\hat{f}(x), f(x)) + \mathrm{CE}(f(x), y)) \\ &+ \frac{1}{\|B^U\|}\sum_{x\in B^U}\mathrm{CE}(\hat{f}(x), f(x))\end{aligned}$$

where B^L is the batch from the labeled dataset, B^U is the batch from the unlabeled dataset, $\|B\|$ is the number of images in B, $\mathrm{CE}(a, b)$ is the cross entropy loss between a and b, and y is the category label for x.

Local Label Propagation (LLP) (18). As described in the main text, two steps were iterated in LLP: a label propagation step and a representation optimization step. The first step generated pseudolabels for unlabeled images. We then combined these pseudolabels with the ground truth labels for labeled images and used the combined labels to optimize the representation using the following loss function:

$$\mathcal{L}_{\mathrm{LLP}} = (\mathrm{CE}(c, y) + \frac{\sum_{x_a}\exp(\mathrm{m}(x_a)^T e/\tau)}{\sum_{x_b}\exp(\mathrm{m}(x_b)^T e/\tau)}) \cdot \mathrm{cf}(x)$$

where c is the 1000-way category prediction and e is the 128-dimension embedding output, both of which were generated by adding one linear layer to the ResNet-18 output, y is the label for the current input x, $\mathrm{cf}(x)$ is the confidence of the label of x defined in the label propagation step, x_a iterates over all images sharing the same labels, and x_b iterates over all training images. In the label propagation step, LLP infers the pseudolabels for the unlabeled images from the labeled images. This step was adopted from the conceptually simpler weighted K-Nearest-Neighbor classification algorithm. In weighted-KNN, the nearest K labeled embeddings ($\mathcal{N}_K(x)$) of the current embedding e were used to decide the pseudolabels through a "vote" process of these labeled examples. The vote of each $x_i \in \mathcal{N}_K(x)$ is weighted by $\exp(\mathrm{m}(x_i)^T e/\tau)$. Assuming S classes, the total weight for x as class j is thus:

$$w_j(x) = \sum_{i\in I^{(j)}}\exp(\mathrm{m}(x_i)^T e/\tau), \quad \text{where} \quad I^{(j)} = \{i | x_i \in \mathcal{N}_K(x), y_i = j\}$$

Therefore, the probability $p_j(x)$ that datapoint x is of class j, the associated inferred pseudo-label y, and the corresponding confidence $\mathrm{cf}(x)$, are defined as:

$$p_j(x) = w_j(x)/\sum_{k=1}^{S} w_k(x), \quad y = \arg\max_j p_j(x), \quad \text{and } \mathrm{cf}(x) = p_y(x).$$

However, this simple KNN procedure "introduced a spurious positive correlation between the local density of a labeled example and the number of unlabeled examples whose pseudo-labels are propagated from this labeled example". (18) To correct this correlation, LLP modified $w_j(x)$ to the following:

$$w_j(v) = \sum_{i\in I^{(j)}}\frac{\exp(\mathrm{m}(x_i)^T e/\tau)}{\sum_{k\in\mathcal{N}_T(x_i)}\exp(\mathrm{m}(x_i)^T \mathrm{m}(x_k)/\tau)}$$

where $\mathcal{N}_T(x_i)$ are T nearest examples of x_i in the embedding space and the denominator is a measure of the local density of x_i in the embedding space.

Neural Response Datasets. *Neural response dataset for V1 area*. This dataset was collected by presenting stimulus to two awake and fixating macaques, where responses of 166 neurons in V1 area were collected by a linear 32-channel array (19). The stimulus consisted of 1450 images from ImageNet and texture-like synthesized images matching the outputs of different layers of a ImageNet trained deep neural networks. The images were presented for 60ms each in one trial without blanks and centered on the population receptive field of the neurons in each session. A total of 262 neurons were isolated in 17 sessions. The response latency of these neurons is typically 40ms. Therefore, spike counts between 40-100ms were extracted and averaged across trials to get the final responses. Neurons were further selected based on whether at least 15% of their total variance could be attributed to the stimuli, which left 166 neurons. Among these neurons, we computed the median of the split-half correlations of all neurons as a measure of reliability. The split-half correlation of one neuron was computed through randomly splitting all the trials of one stimuli into two parts of the same size and computing the Pearson correlation between the averages of two parts (20). We found that the reliability of this V1 area neural response dataset is 0.428. This reliability value was also used to correct neural predictivity of the models.

Neural response dataset for V4 and IT areas. This dataset was collected by presenting stimuli to two fixating macaques, on which three arrays of electrodes were implanted with one array in area V4 and the other two arrays in area IT (21). The stimuli were constructed by rendering one of 64 3-dimensional objects at randomly chosen position, pose, and size, on a randomly chosen naturalistic photograph as background. These objects belonged to 8 categories (tables, planes, fruits, faces, chairs, cars, boats, animals), each of which consisted of 8 unique objects. According to the scale of variances position, pose, and size are sampled from, three datasets were generated, corresponding to low, medium, and high variations. For example, low variation images had objects placed in the center of the images with a fixed position, pose, and size, while objects in high variation images are placed with a highly varied setting. There were in total 5,760 images, of which 2,560 were high variation images and 2,560 were medium variation images. These images were presented to the primates for 100ms with 100ms of gap between images. During presentation, a circular mask was applied to each image, which subtended around 8 degree of visual angle. From the three arrays, the neural responses of 168 IT sites and 88 V4 sites were collected (21, 22). It is unknown exactly which cortical layers these sites were in, but because the macaque brain is gyrencephalic, while the Utah array used in the recordings was rigid and contained electrodes of a fixed length (1.5mm), it is possible that the dataset contained units from a mixture of cortical layers. Following the previous studies (22), we used the averaged responses between 70-170ms after stimuli presentation as this window contained most of object category-related information. The low and medium variation images were used to select hyperparameters of neural fitting and only the prediction results on high variation images were reported. We computed the reliability of neurons in both V4 and IT area and found that the reliability of V4 area is 0.895 and that of IT area is 0.822.

Downstream Task Performance Evaluation. *ImageNet object categorization task*. This task is a standard procedure to evaluate the quality of unsupervised representations in previous studies (12, 14, 15, 23). Results of this task are shown in Fig. S1. A linear readout layer was added to the pretrained visual backbones. This layer was trained on ImageNet training set to perform the ImageNet categorization task through a cross-entropy loss. The performance was then evaluated on the official ImageNet validation set. The initial learning rate was 0.01 and the training took 160 epochs. We dropped the learning rate at 40th, 80th, and 140th epochs. Each learning rate drop was by 10 times. A L2 regularization loss on the linear readout weights was used with the regularization coefficient 10^{-4}. We used the same data augmentations used in previous studies (12, 15): random cropping, random horizontal flip, random color jittering, and random gray-scale transform. We reported the best categorization performance on the official ImageNet validation set throughout the training.

Object categorization task. For this and the following three tasks, we used the same dataset, on which the V4 and IT neural data was collected. For the pretrained visual backbones except the PredNet models, we took the outputs from the first pooling layer and all eight residual blocks. For the PredNet models, we took the outputs from all twelve layers. A PCA-based dimension reduction method was applied to the network output of one layer to reduce its dimension to 1000. This is done through first extracting the responses of the same network and the same layer to 1000 ImageNet validation images and then computing the PCA components for this layer, see (20). After this dimension reduction step, we fit a linear Support Vector Classifier for each layer to predict the category of the object in the input image. As the regularization parameter can significantly influence the final performance, we chose the best parameter from "1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1" using the low and medium variation subsets separately for each network and each layer of the network. We then did the fitting on the training split of the high variation subset and computed the categorization performance on the test split of the same subset. Four different train-test splits were randomly selected. The performance of the best layer was finally reported.

Object position estimation task. After getting the dimension-reduced representations using the same method as the object categorization task, we then fit a linear Support Vector Regression model from these representations to predict both the vertical and the horizontal locations of the object center in the image. We chose the regularization parameters from "1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1" on the low and medium variation subsets and tested the fitting on the high variation subset. The Pearson correlations between the predicted and the ground truth positions were computed for all layers. For each network, we picked the best layer and reported the correlation averaged across both locations.

Object pose estimation task. The only thing different in this task compared to the position prediction task was that the prediction target was the z-axis (vertical axis) and the y-axis (horizontal axis) rotations, both of which ranged between -90 degrees and 90 degrees. The (0, 0, 0) angle was defined in a per-category base and was chosen to make the (0, 0, 0) angle "semantically" consistent across different categories. Hong et al. (24) described these choices as the following:

1. Animals: animal is facing forward, with its head upright.
2. Boats: boat is oriented with bow facing forward and keel point downward.
3. Cars: car grille is facing forward, while tires on the bottom.
4. Chairs: chair legs are facing downward, with the seat facing forward.

5. Faces: looking straight the viewer, with top of the head oriented upward.

6. Fruits: stem attachment at the top. Note that many of the fruits possess a rough rotational symmetry around the vertical axis.

7. Planes: cockpit facing forward, with plane in upright position.

8. Tables: table legs facing straight downward, with longest side along the horizontal axis.

More details can be found in (24). This category-specific definition of the canonical pose, combined with the results shown in Fig. S3 that these object-centric tasks are better supported by higher layers than lower layers, possibly explains the gap between the unsupervised and supervised models in performing this pose estimation task, as the higher layers of the supervised model are more category-related.

Object size estimation task. The prediction target was the three-dimensional object scale, which was used to generate the image in the rendering process. This target varied between 0.625 to 1.6, which meant a relative measure to a fixed canonical size (=1). When objects were at the canonical size, they occluded around 40% of image on longest axis.

Neural response fitting procedure. We mostly followed the fitting method proposed in (25) but slightly modified it to run it more efficiently. This method is different from what is used by Schrimpf et al. (20) as the latter uses a PCA-based dimension reduction method to reduce the dimensions of the network activations and then runs a PLS-Regression, while the former factorizes the linear weight matrix into spatial and channel weight matrices to reduce the number of fitting parameters. We found that the method we used achieves better absolute neural predictivity compared to the method used by Schrimpf et al., though we have also found that the relative order of different models are very similar across different fitting methods.

For the neural network response n_o of one layer whose output shape is $[s_x, s_y, c]$, we fit a spatial mask m_s of shape $[s_x, s_y]$ and a channel mask m_c of shape $[c]$ for each neuron to predict its response r. The predicted response can be written as:

$$\hat{r} = \sum_{i=1}^{s_x} \sum_{j=1}^{s_y} \sum_{k=1}^{c} m_s[i,j] m_c[k] n_o[i,j,k]$$

The optimized loss is then:

$$L = (\hat{r} - r)^2 + w(\|m_s\|_2^2 + \|m_c\|_2^2)$$

We chose w from "0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1" on low and medium variation images used in collecting V4 and IT neural responses. The weights were trained on the training split (0.8) and evaluated on the validation set (0.2). The Pearson correlation was computed between $\hat{r}$ and r and further corrected by the noise ceiling of that neuron. This correction is done through dividing the raw correlation by the square root of cross-trial neural response correlation (20). The median value of the corrected correlations of all neurons within one cortical area was reported for one layer as its neural predictivity for this area. For each neuron, the neural predictivity is first averaged across all three networks and all four training-validation splits. The error bars are the standard deviations of means generated through bootstrapping from neural predictivity of all neurons for 200 times.

Optimal Stimuli Computation. Following a previous study (26), we optimized the 2D discrete Fourier transform of the input image to maximize the spatially averaged responses of one channel in a given layer. We used Adam optimizer with learning rate 0.05 and trained for 512 steps. In each training step, random augmentations including jitterring, scaling, and rotations were applied to the input image (26).

Human Behavior Consistency Metric. The behavior dataset consists of 2400 images generated by putting 24 objects in front of high-variant and independent naturalistic backgrounds (27). For each pretrained network, a linear classifier was trained from the penultimate layer on 2160 training images to predict the category. As the number of features may be too large, we tested three dimension reduction methods and report the best consistency among them. These methods are: 1. averaging across spatial dimensions; 2. PCA projections to 1000 components using ImageNet validation images; 3. Training a 1000-dimension linear category-centric projection through adding a linear layer outputting 1000 units upon the current layer and another linear readout upon this linear layer and then training only these two added layers to do ImageNet categorization tasks. The resulted confusion matrix on 240 validation images was compared to that of human subjects, for which 1,472 human subjects were recruited from Amazon Mechanical Turk (details can be found in Rajalingham et al. (27) and Schrimpf et al. (20)). As we directly took the data from (27), IRB approval is not required for us. The Pearson correlation between the matrixes was computed and then corrected by dividing it using the square root of human split-half correlation (20).

Data Availability. ImageNet can be downloaded from http://www.image-net.org/. SAYCam can be downloaded from https://nyu.databrary.org/volume/564. Neural data is available through the public BrainScore repo.

Code Availability. Our codes can be found at https://github.com/neuroailab/unsup_vvs.

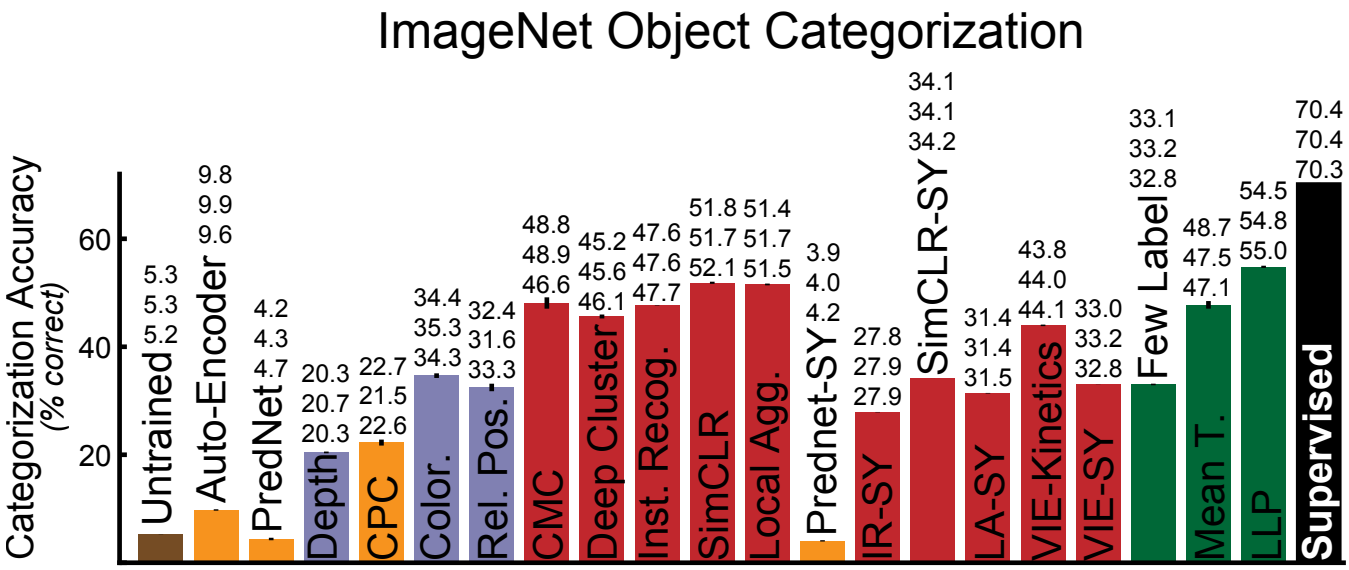

Fig. S1. Transfer performance on ImageNet object categorization task. Three numbers on top of each bar represent the transfer performance from three networks of different initializations. Models ending with "SY" were trained on SAYCam. Green bars are for semi-supervised models.

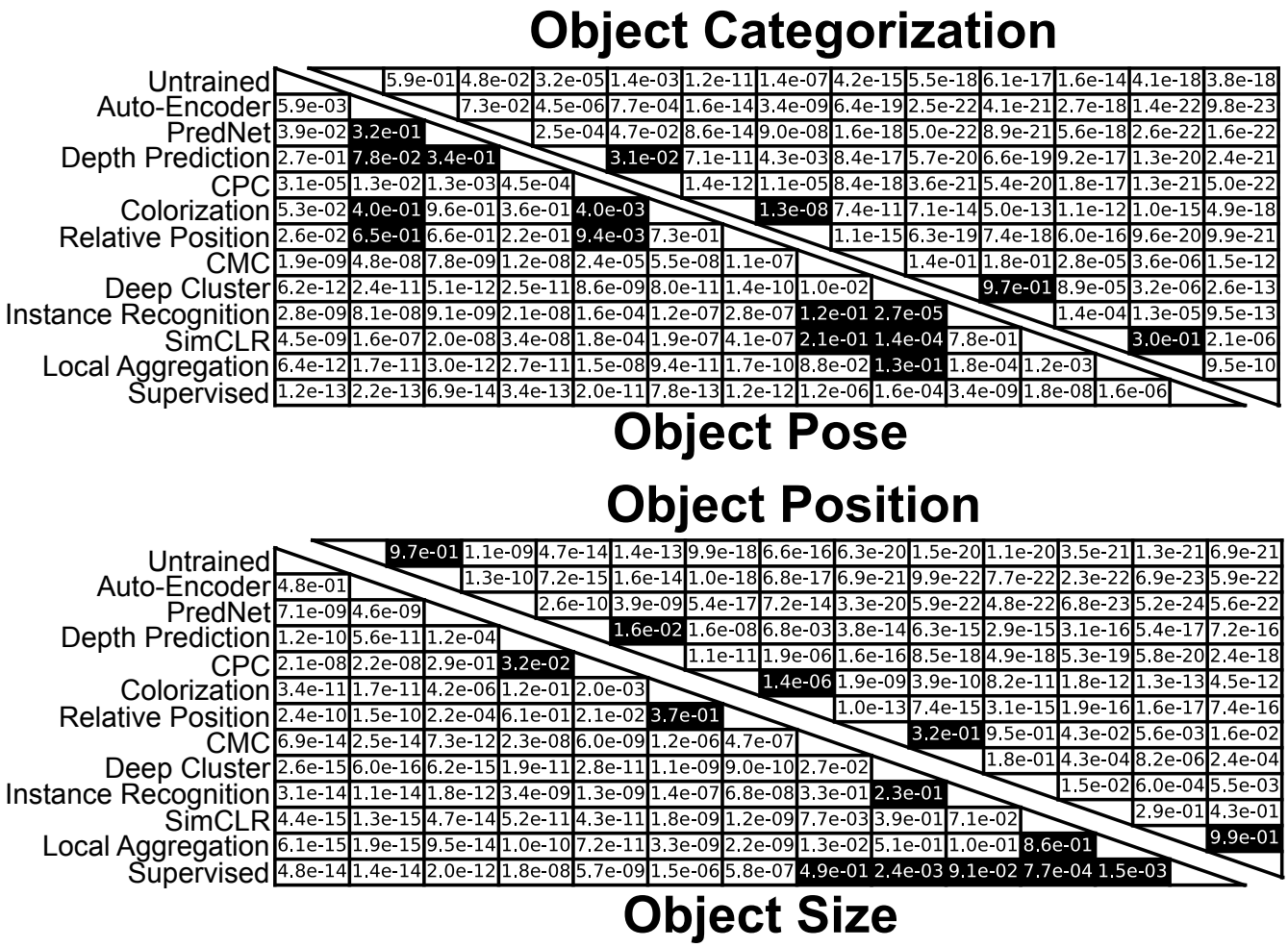

Fig. S2. T-test results of transfer task performance. If the performance of the method in the i-th row is better than that of the method in the j-th column, the number in the i-th row and j-th column is then in black color and means the unpaired and two-tailed t-test p-value between two methods. Otherwise, if the i-th method is worse than the j-th method, then the number is in white color with black background. The degree of freedom is 11, as there were 4 train-validation splits for each of the three models and we used all 12 numbers to perform the t-tests.

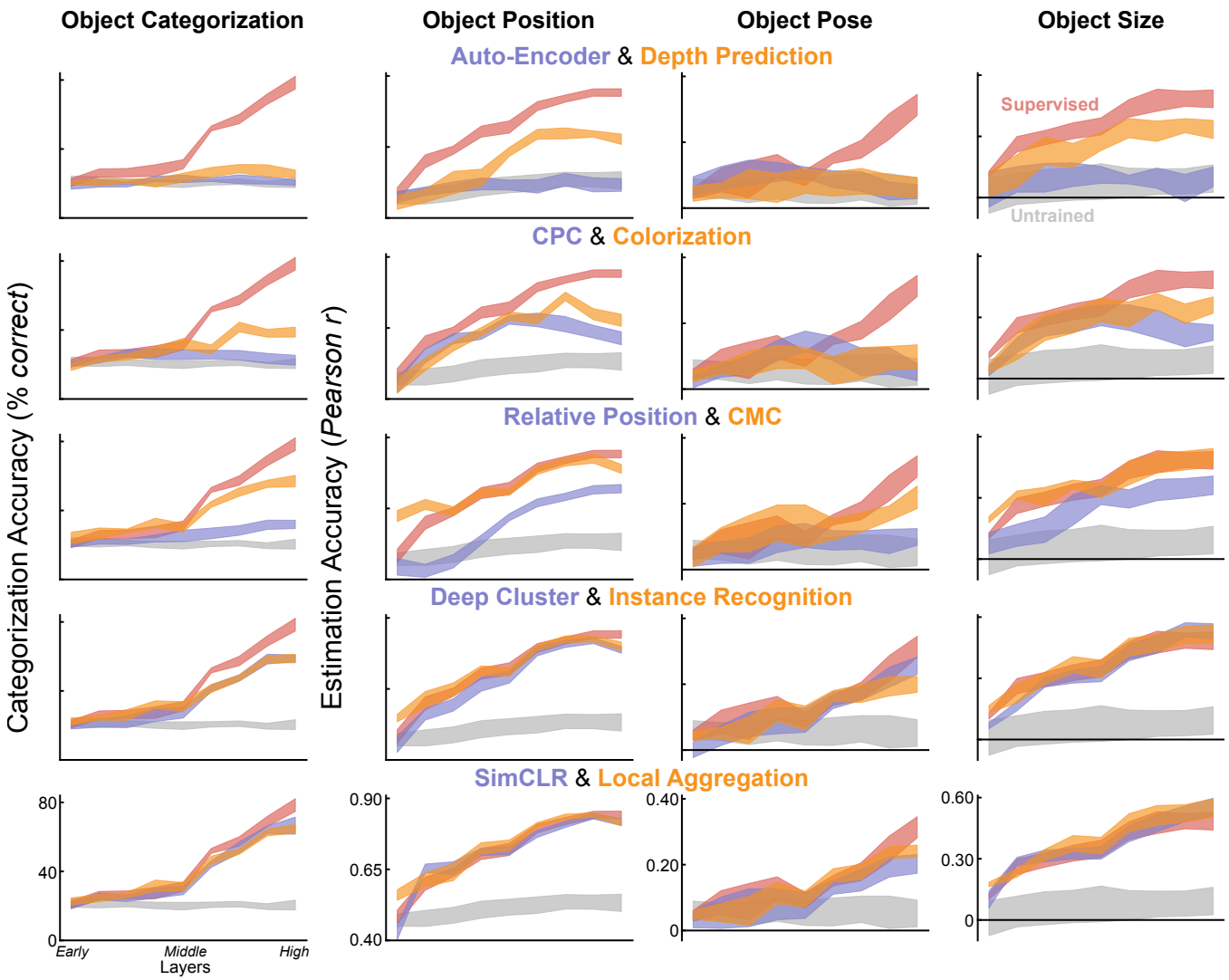

Fig. S3. The per-layer transfer task performance of all methods. This figure shows that the methods which perform good in transfer tasks achieve their best performance in high or middle layers, instead of early layers. The line width is the standard deviation across three models with different initializations and four train-val splits.

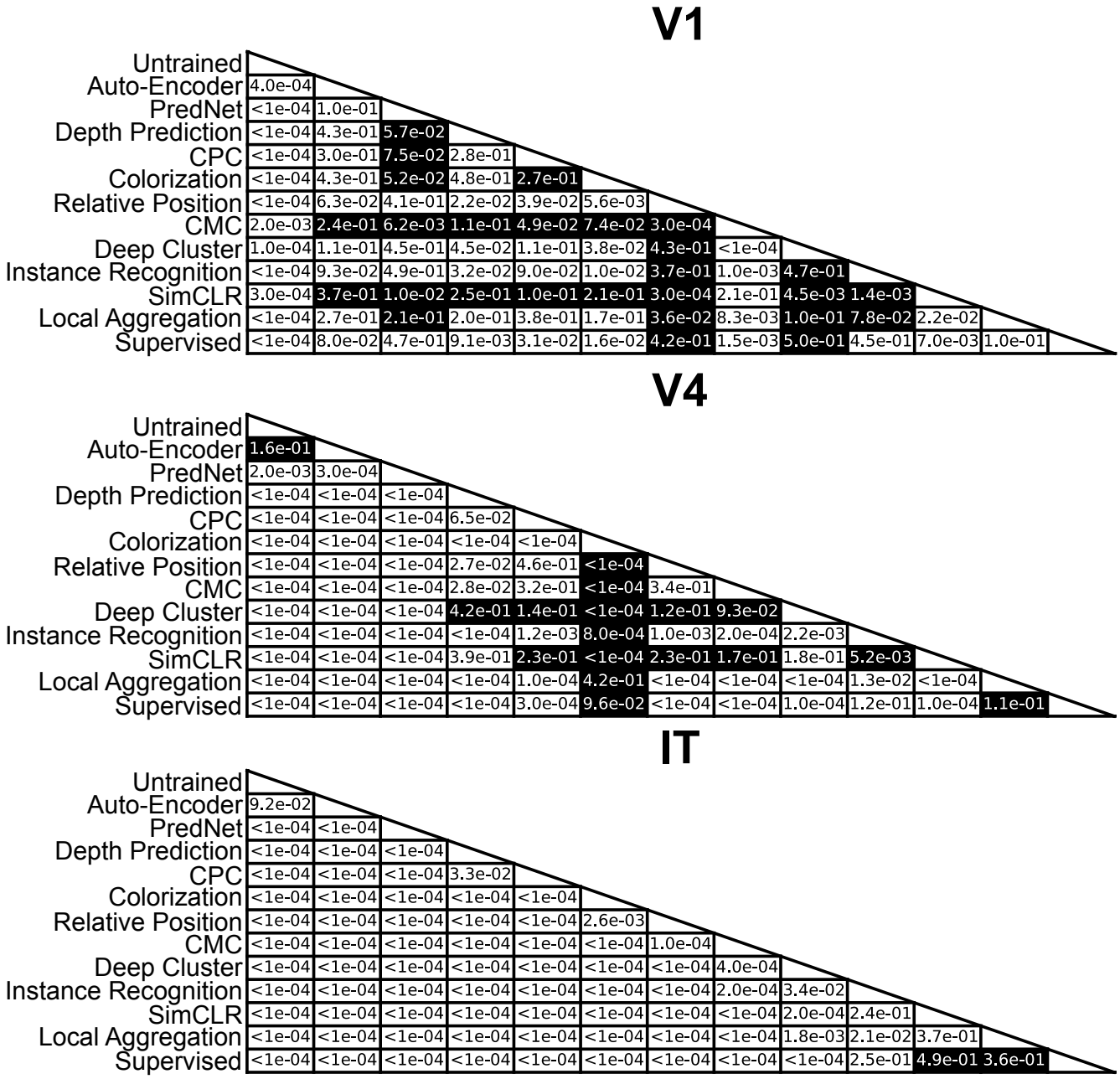

Fig. S4. Statistical test results of neural predictivity results. We computed statistical significance through a bootstrapping method. The meaning of the text color and the background color is the same as that in Fig. S2. When using the bootstrapping method, we repeatedly sampled neurons with replacement in each area for 10,000 times. For each sampled group of neurons, we computed the median of neural predictivity of models across these sampled neurons and compared the medians of two methods. The numbers were computed by counting the number of samples one method is better/worse than another method and dividing this number by 10,000. "<1e-04" means that this counting returns 0. This bootstrapping method simulates the situations where new neurons are sampled from the same distribution. Therefore the resulting numbers represent the probability of one method being better than another method for a newly sampled group of neurons.

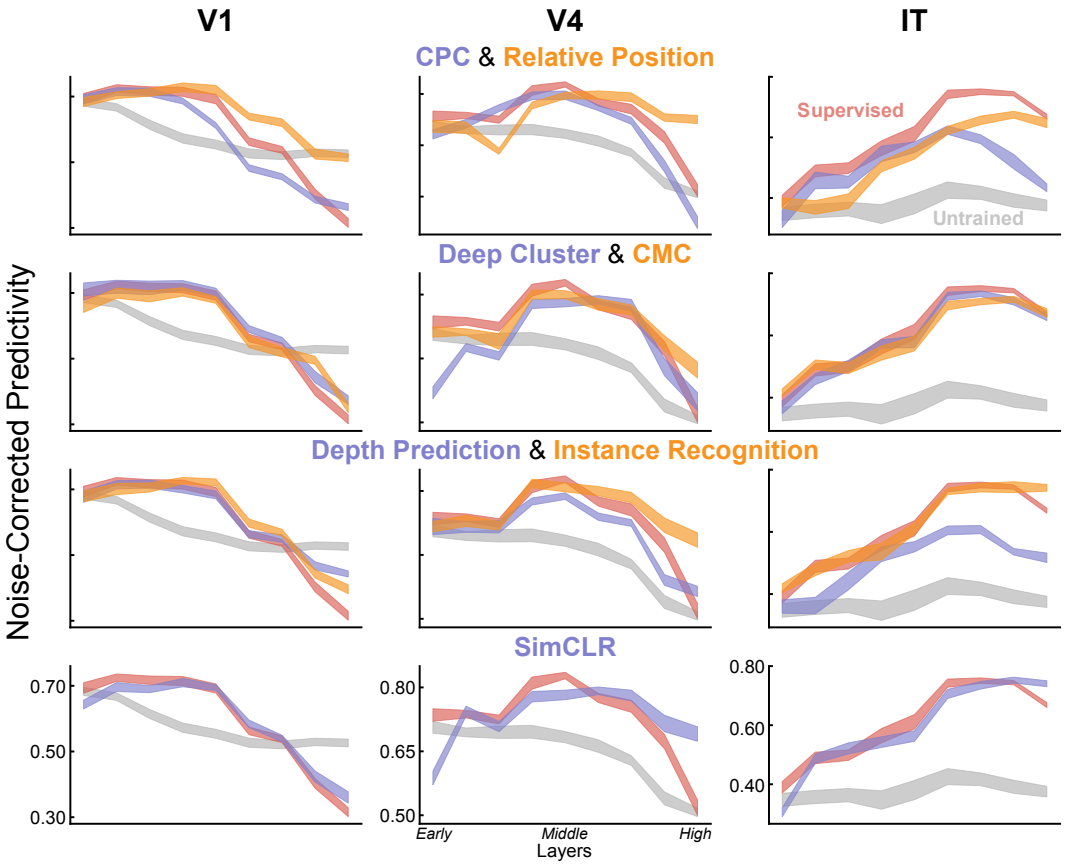

Fig. S5. Neural predictivity of DCNNs across layers. Most methods have correct model-layer-to-brain-area correspondence, e.g. V1 area is best predicted by early layers, V4 area is best predicted by middle layers, and IT area is best predicted by high layers. This correspondence is more correct for models with good neural predictivity for one area. For example, depth prediction models have lower IT neural predictivity, its best layer for IT neurons is also lower than supervised or SimCLR models. The line width is the standard deviation of the neural predictivity medians generated from the bootstrapping method.

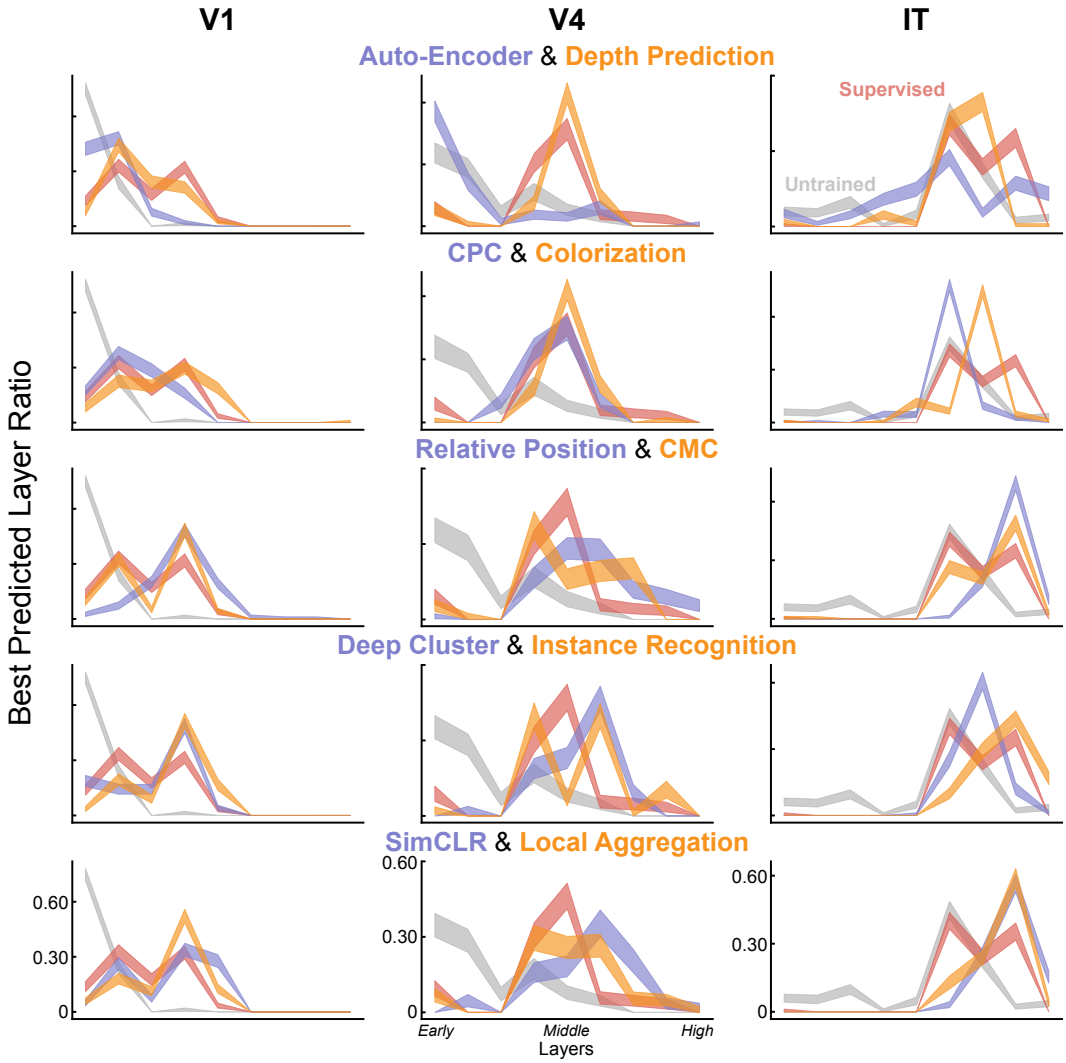

Fig. S6. Best predicted layer ratio for neurons. This ratio is computed in the following way: for each DCNN layer, we compute the number of neurons that are best predicted by this layer and then divide this number by the number of neurons of this area. Similar to Fig. S5, this figure also shows that most methods have correct model-layer-to-brain-area correspondence, though models with better neural correspondence for one area also have more correct correspondence for that area. For example, SimCLR and DeepCluster, both of which have lower V4 neural predictivity, are also peaking at later layers for V4 area compared to the supervised and the Local Aggregation models. The line width is the standard deviation of the ratios generated from the bootstrapping method.

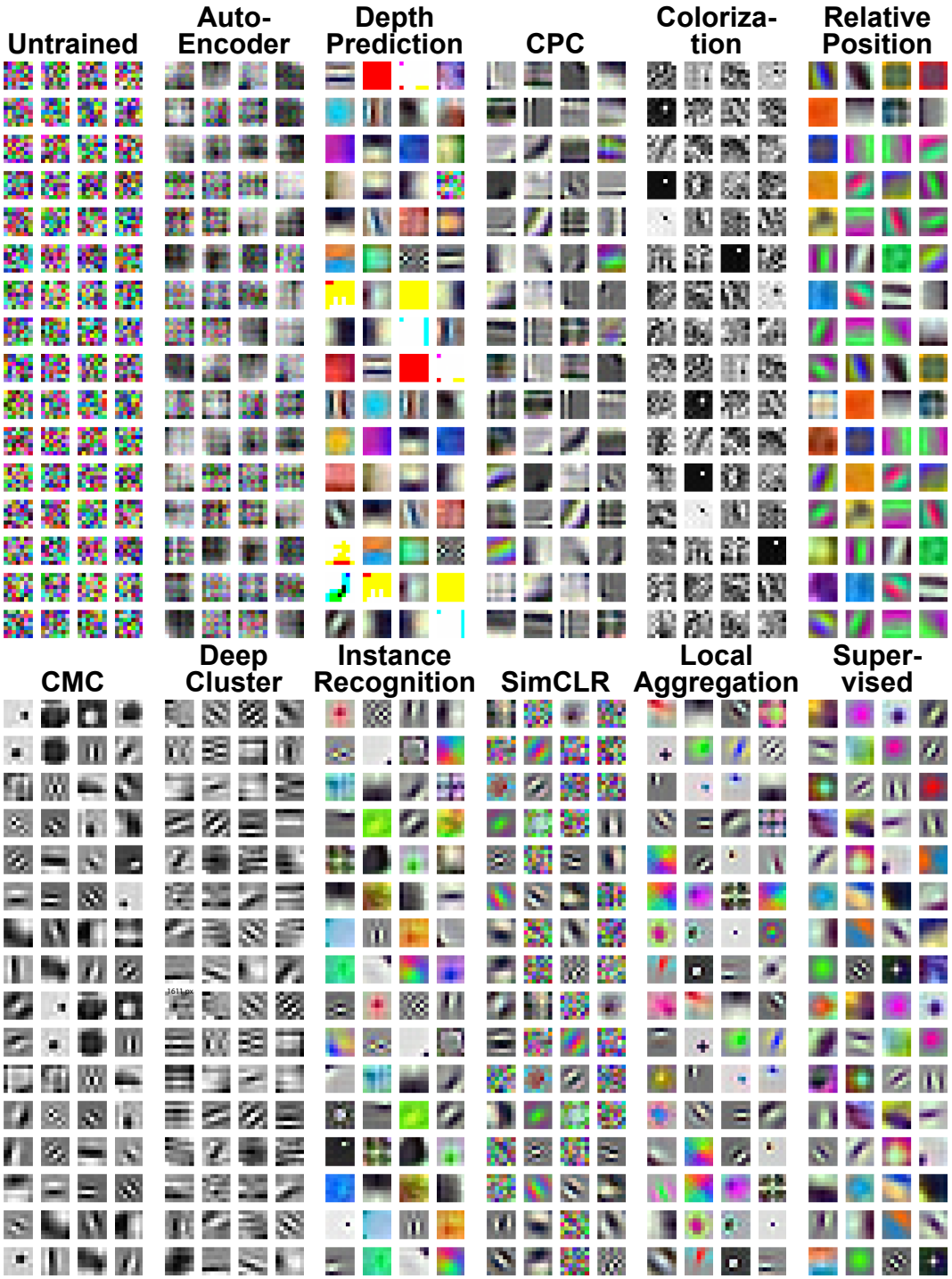

Fig. S7. First layer filters of DCNNs. Most of the models have Gabor-like and center-surrounding filters, corresponding to their comparable V1 neural predictivity to the supervised model.

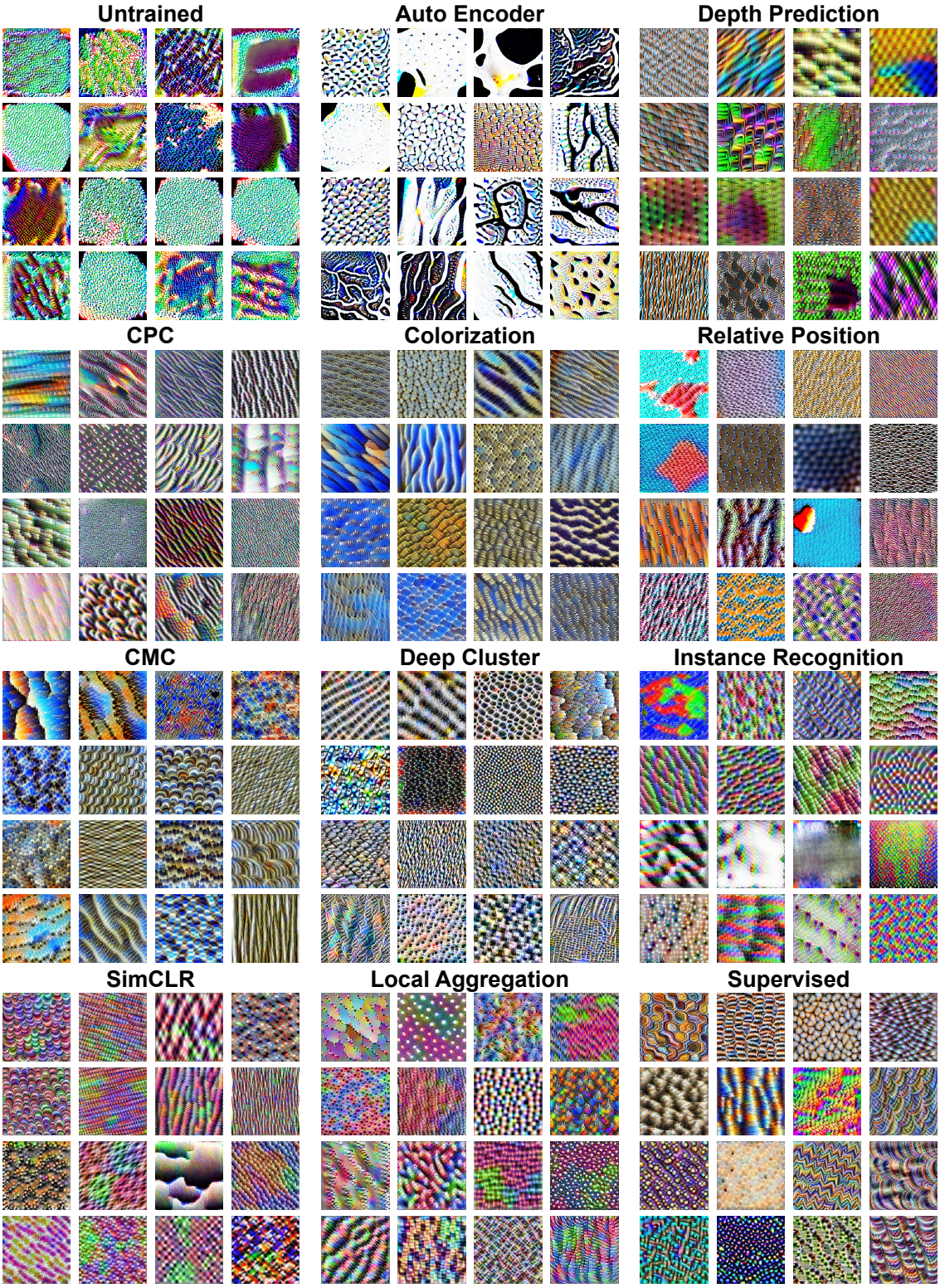

Fig. S8. Optimal stimuli for intermediate layers of DCNNs. The optimal stimuli for models with good V4 neural predictivity, such as the supervised and the Local Aggregation models, are also more texture-like, compared to models with worse V4 neural predictivity, like the DeepCluster and the Relative Position models.

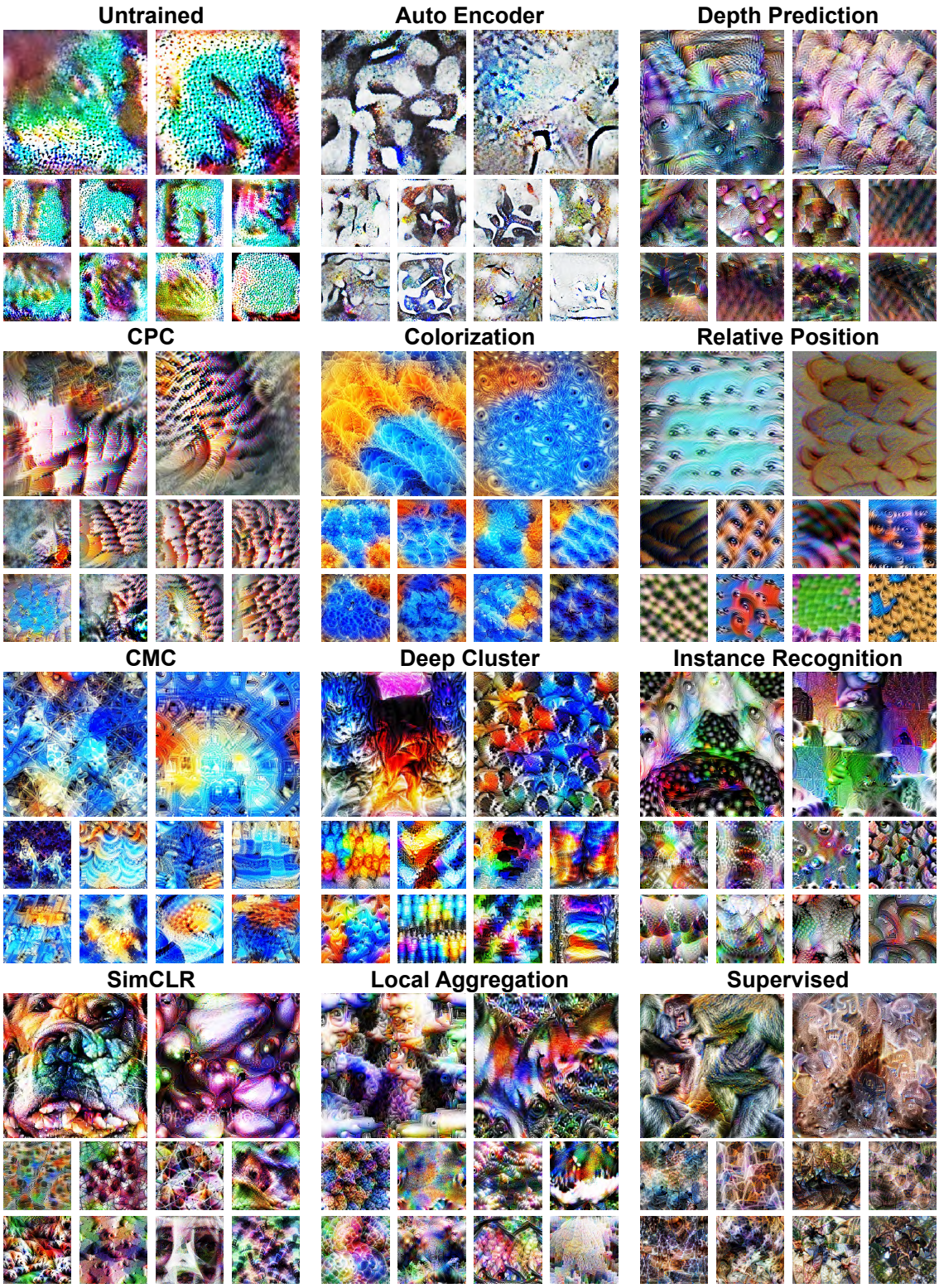

Fig. S9. Optimal stimuli for high layers of DCNNs. Best viewed when scaled. The optimal stimuli for models with good IT neural predictivity, such as the supervised and the Local Aggregation models, are also more object-like, compared to models with worse IT neural predictivity, like the Colorization and the Relative Position models.

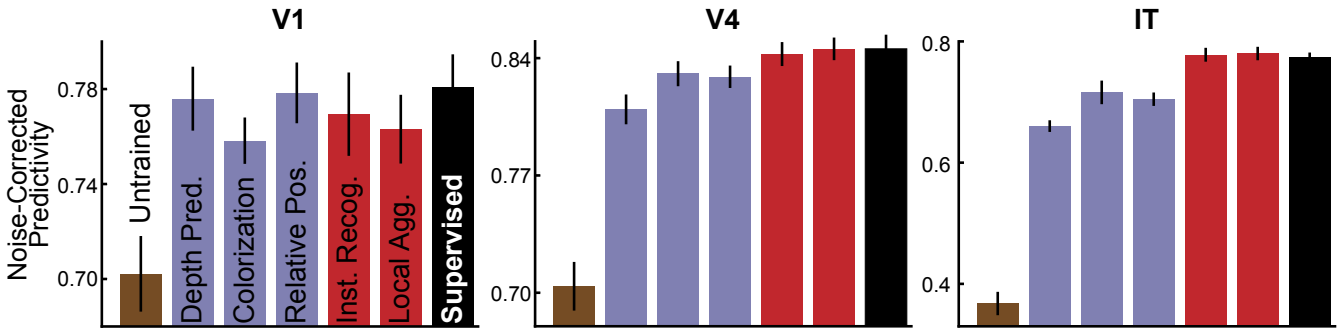

Fig. S10. Neural predictivity results of ResNet-50 models, which are significantly deeper than ResNet-18 models and have different block designs compared to ResNet-18 (1). These models were trained with the same hyperparameters and the same loss functions used for the ResNet-18 models. Similar to the neural predictivity results of ResNet-18 models, the deep contrastive embedding methods (red bars) also achieve comparable neural predictivity to the supervised model and are better than the self-supervised methods (blue bars).

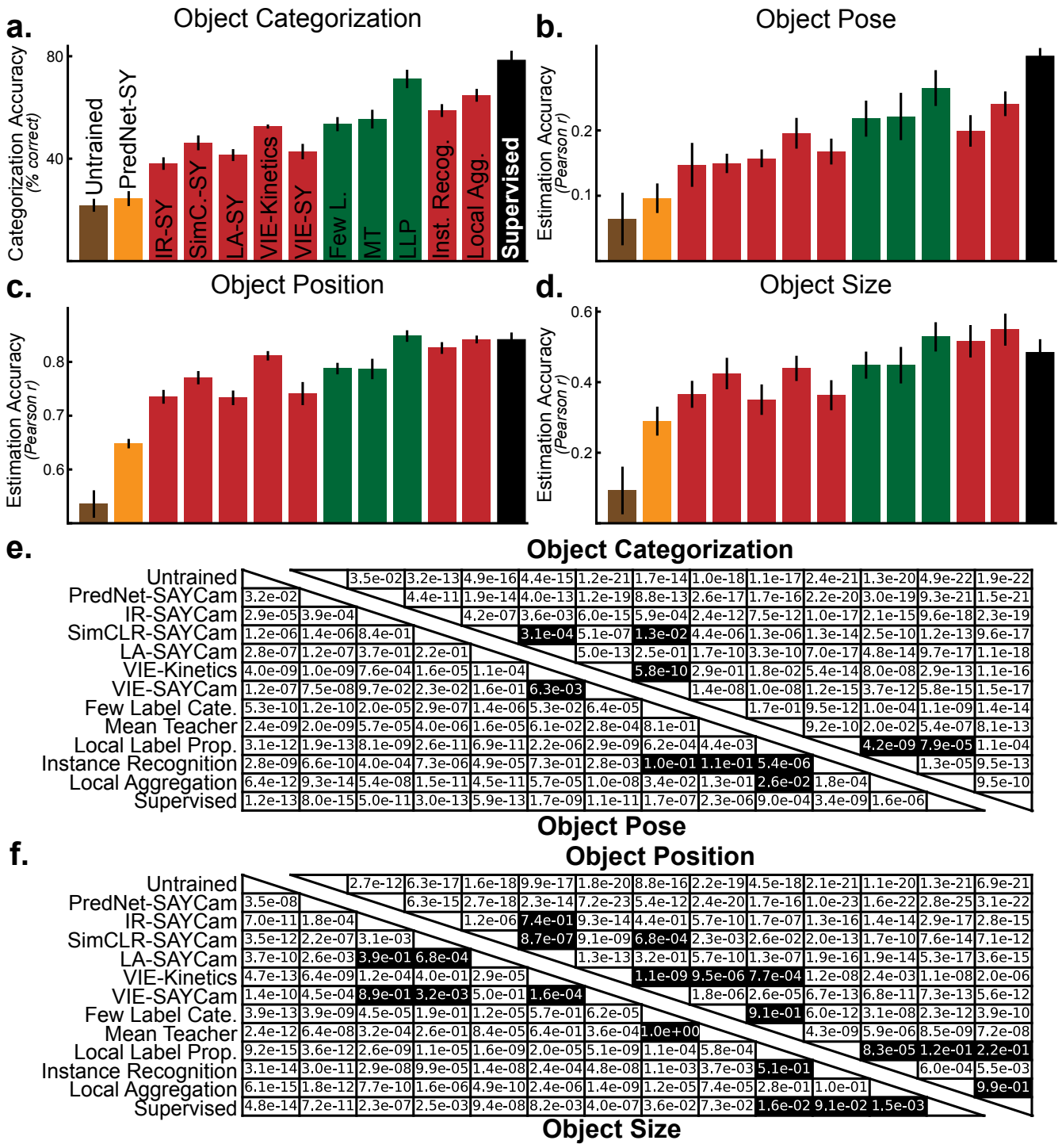

Fig. S11. Transfer task performance (**a** to **d**) and the corresponding t-test results (**e** and **f**) of models not shown in Fig. 1, including models trained on SAYCam (SY) and semi-supervised models trained with 36K labels (green bars). VIE-Kinetics is the same architecture as the VIE-SAYCam model, but trained on Kinetics. The t-test results are in the same format as Fig. S2.

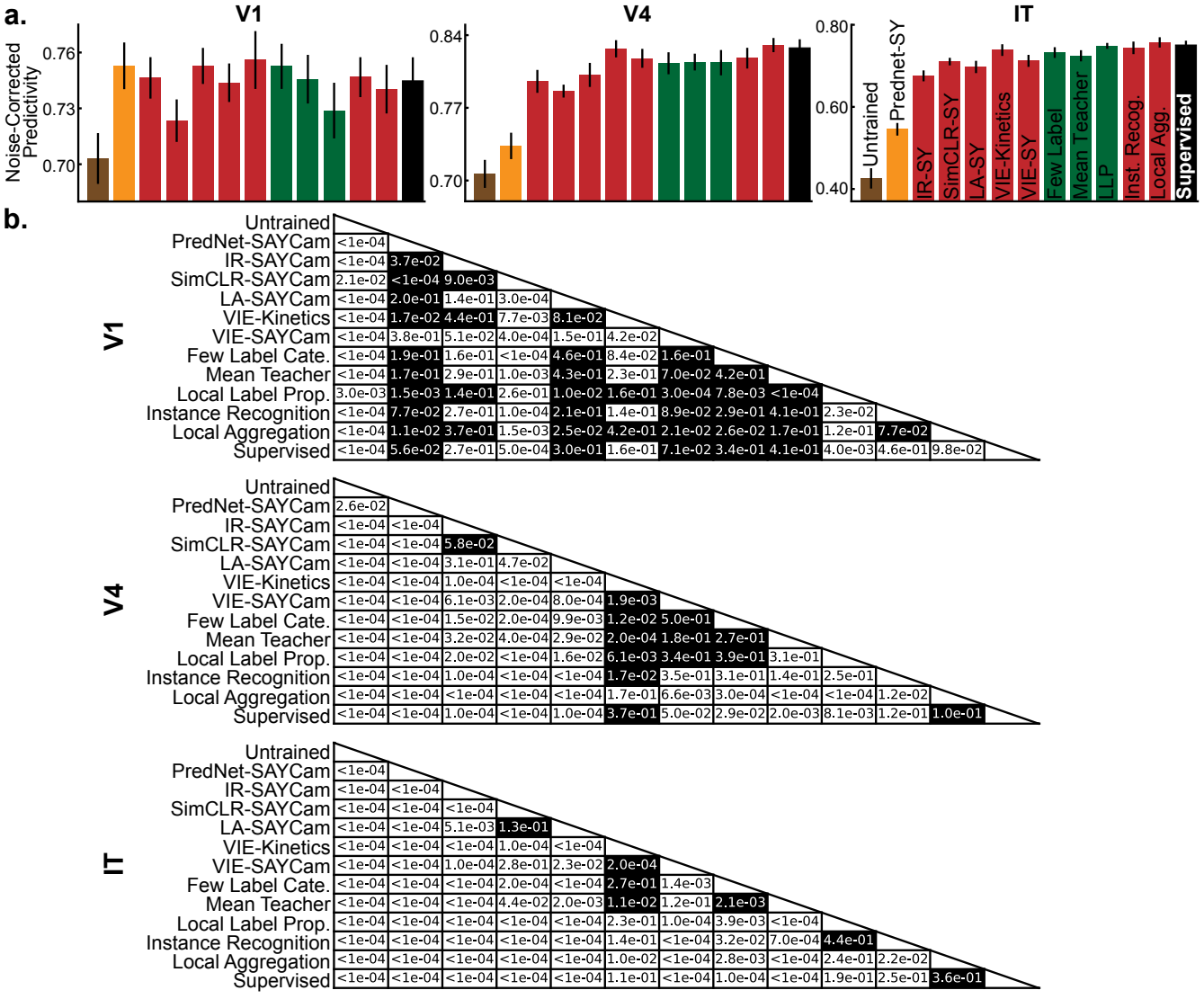

Fig. S12. Neural predictivity (**a**) and the corresponding statistical test results (**b**) of models not shown in Fig. 1, including models trained on SAYCam (SY) and semi-supervised models trained with 36K labels (green bars). The statistical test results are in the same format as Fig. S4.

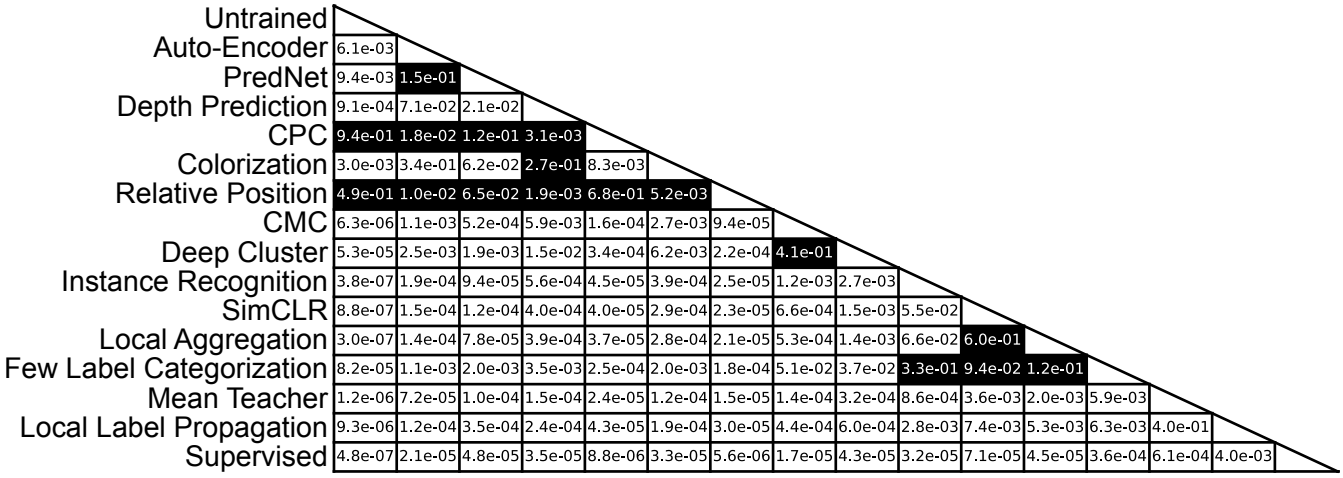

Fig. S13. T-test results of human behavior consistency results (Fig 4**d**). The format is similar to that in Fig. S2. The t-test performed is unpaired and two-tailed. The degree of freedom is 2, as each method has three models of different initializations.

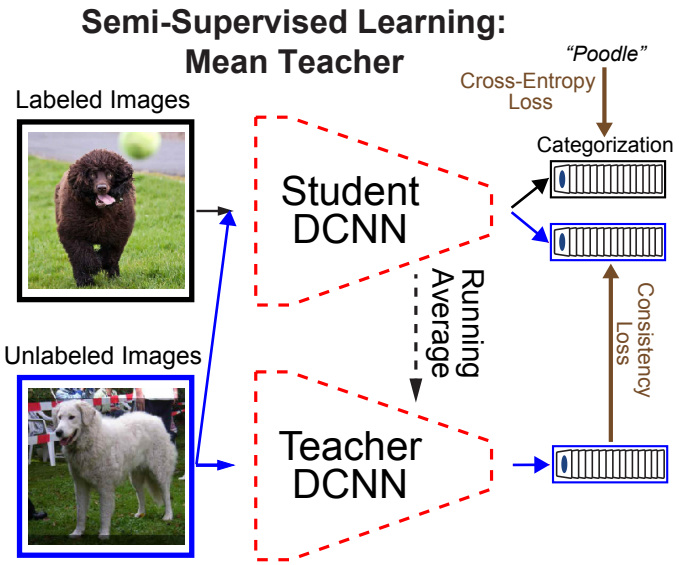

Fig. S14. Schematic for the Mean Teacher (MT) method. In addition to the optimized "student DCNN", MT maintained a "teacher DCNN", whose weights were running averages of the student DCNN. The loss was the sum of the categorization loss on labeled images and the consistency loss between two DCNNs on unlabeled images.

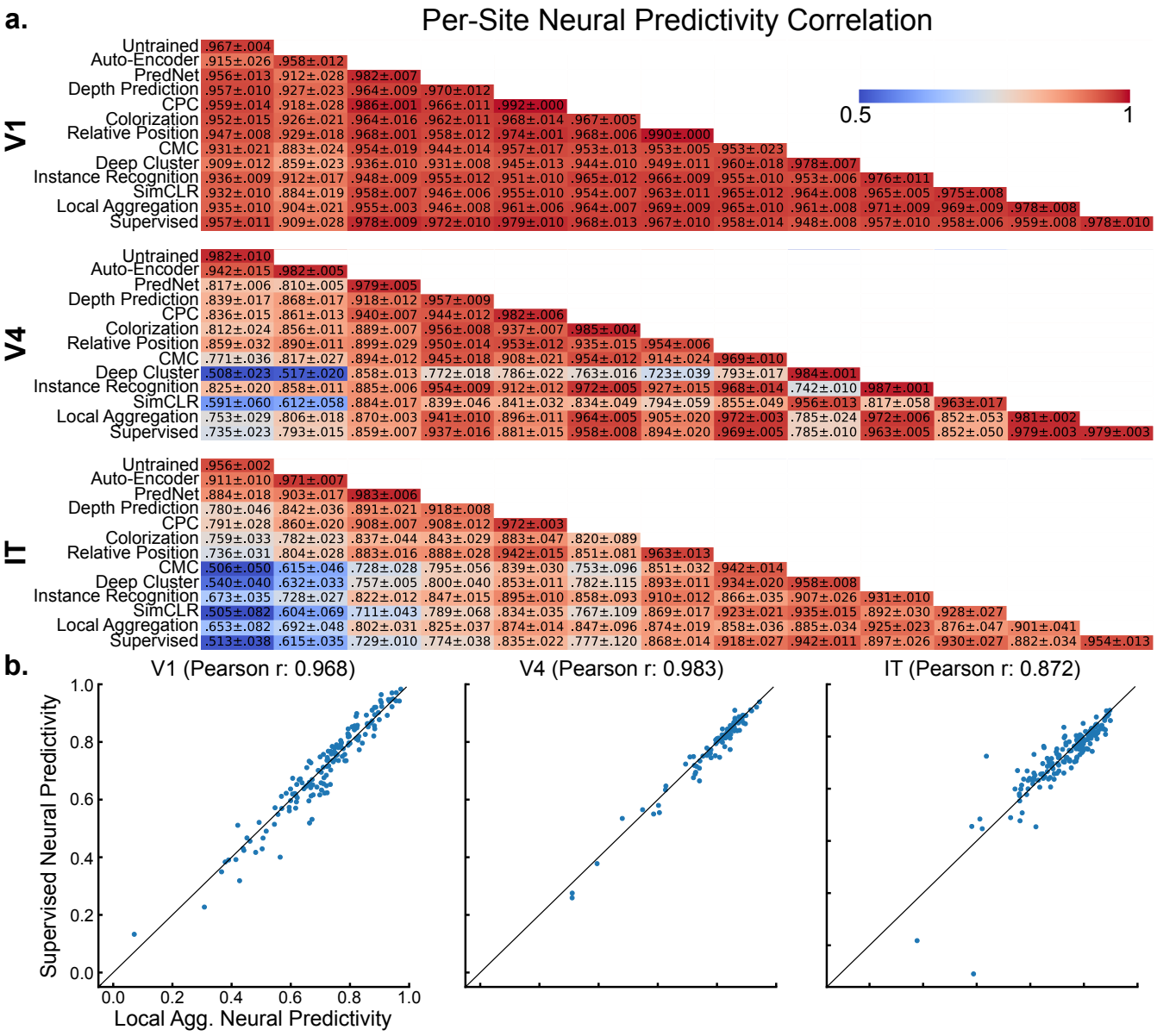

Fig. S15. Correlation of per-site neural predictivity results of DCNNs and scatter plots. (**a**) The per-site neural predictivity result is first computed for each of the three networks of one method. Then, for each pair of methods, the correlation (the number before the $\pm$ symbol) is computed by averaging all possible pairs of networks chosen from each method (for two different methods, there are 9 pairs; for the same method, there are 3 pairs). Therefore, the diagonal numbers represent noise ceiling of this measure. The number after the $\pm$ symbol is the standard deviation of all possible correlations for one pair of methods. (**b**) Scatter plots of neural predictivity of one Local Aggregation network and one supervised network for all three areas. Each dot represents one neuron in that area.

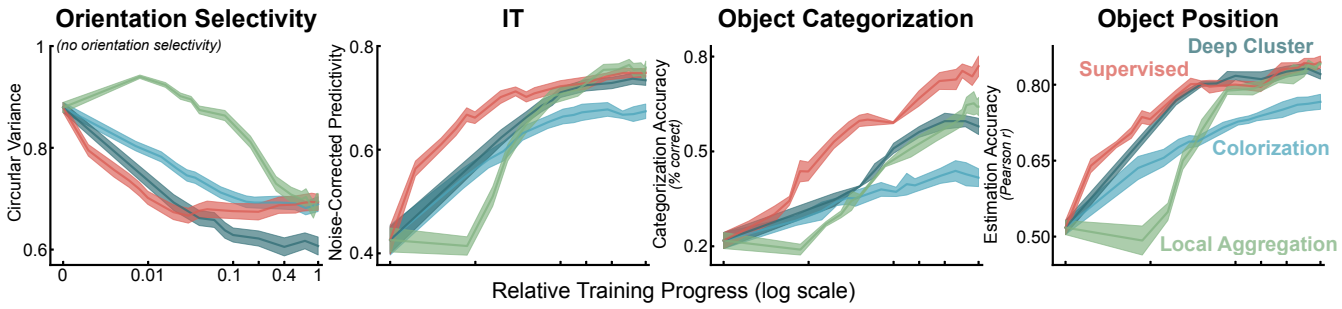

Fig. S16. Training trajectories for (left to right): orientation selectivity measured by circular variance on early layer of DCNNs, IT neural predictivity of best layer, object categorization, object position predicting performance. Local Aggregation networks show slower learning speed at the beginning stage, mainly as they need to bootstrap their representations from scratch and slowly accumulate more accurate running averages of the embeddings in the memory banks used in these networks. Although the learning progress of IR and SimCLR is not shown here, this slowness at the beginning stage is also the case for both of these two methods.

References

1. K He, X Zhang, S Ren, J Sun, Deep residual learning for image recognition in *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 770–778 (2016).
2. W Lotter, G Kreiman, D Cox, Deep predictive coding networks for video prediction and unsupervised learning in *arXiv preprint arXiv:1605.08104*. (2016).
3. J Carreira, A Zisserman, Quo vadis, action recognition? a new model and the kinetics dataset in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 6299–6308 (2017).
4. I Laina, C Rupprecht, V Belagiannis, F Tombari, N Navab, Deeper depth prediction with fully convolutional residual networks in *2016 Fourth 3DV*. (IEEE), pp. 239–248 (2016).
5. Y Zhang, et al., Physically-based rendering for indoor scene understanding using convolutional neural networks in *2017 CVPR*. (IEEE), pp. 5057–5065 (2017).
6. Avd Oord, Y Li, O Vinyals, Representation learning with contrastive predictive coding in *arXiv preprint arXiv:1807.03748*. (2018).
7. K Cho, et al., Learning phrase representations using rnn encoder-decoder for statistical machine translation in *arXiv preprint arXiv:1406.1078*. (2014).
8. R Zhang, P Isola, AA Efros, Colorful image colorization in *ECCV*. (Springer), pp. 649–666 (2016).
9. C Doersch, A Zisserman, Multi-task self-supervised visual learning in *The IEEE International Conference on Computer Vision (ICCV)*. (2017).
10. C Doersch, A Gupta, AA Efros, Unsupervised visual representation learning by context prediction in *Proceedings of the IEEE International Conference on Computer Vision*. pp. 1422–1430 (2015).
11. M Caron, P Bojanowski, A Joulin, M Douze, Deep clustering for unsupervised learning of visual features in *ECCV*. pp. 132–149 (2018).
12. Z Wu, Y Xiong, SX Yu, D Lin, Unsupervised feature learning via non-parametric instance discrimination in *CVPR*. pp. 3733–3742 (2018).
13. Y Tian, D Krishnan, P Isola, Contrastive multiview coding in *ECCV*. (2020).
14. T Chen, S Kornblith, M Norouzi, G Hinton, A simple framework for contrastive learning of visual representations in *ICML*. (2020).
15. C Zhuang, AL Zhai, D Yamins, Local aggregation for unsupervised learning of visual embeddings in *Proceedings of the IEEE International Conference on Computer Vision*. pp. 6002–6012 (2019).
16. C Zhuang, T She, A Andonian, MS Mark, D Yamins, Unsupervised learning from video with deep neural embeddings in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 9563–9572 (2020).
17. A Tarvainen, H Valpola, Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results in *Advances in neural information processing systems*. pp. 1195–1204 (2017).
18. C Zhuang, X Ding, D Murli, D Yamins, Local label propagation for large-scale semi-supervised learning in *arXiv preprint arXiv:1905.11581*. (2019).
19. SA Cadena, et al., Deep convolutional models improve predictions of macaque v1 responses to natural images. *PLoS computational biology* **15**, e1006897 (2019).
20. M Schrimpf, et al., Brain-score: Which artificial neural network for object recognition is most brain-like? in *BioRxiv*. (Cold Spring Harbor Laboratory), p. 407007 (2018).
21. NJ Majaj, H Hong, EA Solomon, JJ DiCarlo, Simple learned weighted sums of inferior temporal neuronal firing rates accurately predict human core object recognition performance. *J. Neurosci.* **35**, 13402–13418 (2015).
22. DL Yamins, et al., Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proc. Natl. Acad. Sci.* **111**, 8619–8624 (2014).
23. K He, H Fan, Y Wu, S Xie, R Girshick, Momentum contrast for unsupervised visual representation learning in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 9729–9738 (2020).
24. H *Hong, DL *Yamins, NJ Majaj, JJ DiCarlo, Explicit information for category-orthogonal object properties increases along the ventral stream. *Nat. neuroscience* **19**, 613–622 (2016).
25. D Klindt, AS Ecker, T Euler, M Bethge, Neural system identification for large populations separating "what" and "where" in *Advances in Neural Information Processing Systems*. pp. 3506–3516 (2017).
26. C Olah, A Mordvintsev, L Schubert, Feature visualization in *Distill*. (2017) https://distill.pub/2017/feature-visualization.
27. R Rajalingham, et al., Large-scale, high-resolution comparison of the core visual object recognition behavior of humans, monkeys, and state-of-the-art deep artificial neural networks. *J. Neurosci.* **38**, 7255–7269 (2018).