Training neural networks to have brain-like representations improves object recognition performance

Abstract

The current state-of-the-art object recognition algorithms, deep convolutional neural networks (DCNNs), are inspired by the architecture of the mammalian visual system [8], and capable of human-level performance on many tasks [15]. However, even these algorithms make errors. As DCNNs improve at object recognition tasks, they develop representations in their hidden layers that become more similar to those observed in the mammalian brains [24]. This led us to hypothesize that teaching DCNNs to achieve even more brain-like representations could improve their performance. To test this, we trained DCNNs on a composite task, wherein networks were trained to: a) classify images of objects; while b) having intermediate representations that resemble those observed in neural recordings from monkey visual cortex. Compared with DCNNs trained purely for object categorization, DCNNs trained on the composite task had better object recognition performance. Our results outline a new way to regularize object recognition networks, using transfer learning strategies in which the brain serves as a teacher for training DCNNs.

1 Introduction

Deep convolutional neural networks (DCNNs) have recently led to a rapid advance in the state-of-the-art object recognition systems [15]. At the same time, there remain critical shortcomings in these systems [18]. For example: DCNNs are much more sensitive to image manipulations like gray-scaling than are humans [3]; DCNNs have a strong bias towards textural information over shape [4]; and DCNNs are subject to adversarial examples (small pixel-level image manipulations, imperceptible to a human, that nevertheless cause the DCNN to change the category label it assigns to the image [5]). Given these challenges, we asked whether training DCNNs to respond to images in a more brain-like manner could lead to better performance. Motivating us, DCNN architectures are directly inspired by that of the mammalian visual system (MVS) [8], and as DCNNs improve at object recognition tasks, they learn representations that are increasingly similar to those found in MVS [10][23][6][17]. Consequently, we expected that forcing the DCNNs to have image representations that were even more similar to those found in MVS, could lead to better performance.
Previous work showed that the performance of smaller “student” DCNNs could be improved by training them to match the image representations of larger “teacher” DCNNs [17, 19, 7], and that DCNNs could be directly trained to reproduce image representations formed by the V1 area of monkey visual cortex [9]. These studies provide a foundation for the current work, in which we used monkey V1 as a teacher network for training DCNNs to categorize images. This is a form of transfer learning, where we used the monkey brain to constrain the image representations within the DCNN. DCNNs trained with the monkey V1 as a teacher outperformed those trained without this teacher signal, by several relevant metrics. We did not aim to achieve state-of-the-art classification performance, as we are studying neural networks that are small relative to the current state-of-the-art (the largest networks we considered have the VGG-16 architecture). However, our results indicate that, even for moderately powerful DCNNs, performance can be improved by using data from the monkey brain as a teacher signal. We anticipate that future studies could apply this training method to larger networks, thereby improving on the current current state-of-the-art object recognition systems.

2 Methods

2.1 Monkey Visual Cortex Data

We used publicly-available multielectrode recordings from anesthetized monkeys presented with a series of images while experimenters recorded the spiking activity of neurons in primary visual cortex (V1) with a multielectrode array [2]. These recordings were conducted in 10 experimental sessions with 3 different animals, resulting in recordings from 392 neurons. In addition to these V1 data, which our main focus, we also studied recordings from cortical areas V4 and IT [16]. Similar to the V1 data, these recordings were performed with multielectrode arrays implanted in the cortex.

2.2 Representational Similarity Matrices (RSMs)

To compare image representations in the monkey brain with those in a DCNN, we used representational similarity matrices (RSMs) [11]. For each pair of images (i & j) shown to the monkey, we computed the similarity between measured neural responses ($v_i$ & $v_j$): these vectors contain the firing rates of all of the observed neurons. We assessed the similarity by the cosine similarity between those vectors (Fig. 1). These values ($RSM_{ij}$) were assembled into matrices, describing the representational similarities [11] in monkey V1, for all image pairs (i & j). We averaged the representational similarity matrices over the 10 experiments to yield a single RSM that was used for training the neural networks. During DCNN training, we input the same pairs of images (i & j) into our DCNNs as were displayed to the monkeys, and computed representational similarity matrices for the chosen layer of hidden units (Fig. 1): the DCNN’s RSM is denoted by $\hat{RSM}$.

2.3 Deep Convolutional Neural Networks and Cost Functions

We performed most of our experiments on the CORNet-Z DCNN architecture, described below. We also performed some experiments on the larger VGG-16 architecture. Results were similar for both architectures. The CORNet-Z DCNN architecture [14] is a trimmed-down version of the AlexNet [12] object recognition algorithm (Fig. 1). The layers in CORNet-Z have been identified with the brain areas at the corresponding depths within the mammalian visual hierarchy [14] (Fig. 1).

We trained the networks on the CIFAR100 [13] task, which consists of classifying images of objects from 100 different categories. Regardless of the network architecture, we randomly initialized all weights, and trained the DCNN to minimize a cost function consisting of two terms: classification error, and mismatch between the network’s hidden representations and those in monkey V1. Classification error was computed as the cross entropy between the network’s final outputs and the true object labels. Representation mismatch was computed as the mean-squared error between the monkey V1 representational similarity matrix (RSM), and that of the relevant layer of the DCNN ($\hat{RSM}$). For CORNet-Z, this was the V1 block (Fig. 1), and for VGG-16, this was the third convolutional layer. That layer was chosen because, in VGG-16 networks trained for object recognition tasks, it has representations most similar to those seen in monkey V1 [1].
natural images

CORNet-Z Model

V1 V2 V4 IT
decoder
CORNet-Z Model

1 2

image1
image2

𝑹𝑺𝑴

Figure 1: CORNet-Z has multiple blocks, each of which consists of a convolution followed by a ReLU nonlinearity and max pooling. The blocks are identified with cortical areas V1, V2, V4 and IT, which exist at the corresponding depths in primate visual system. The cost function is a weighted combination of classification error (cross entropy) and representational similarity, weighted by \( \lambda \).

A trade-off parameter, \( \lambda \), determines the relative weighting of the two terms in the cost function

\[
\text{cost} = \lambda \sum_{i,j} (RSM_{ij} - \hat{RSM}_{ij})^2 - \sum_i \hat{y}_i \log(y_i).
\]  

We updated the trade-off parameter \( \lambda \) throughout training, so as to keep the ratio between the two terms in the loss function constant. I.e., \( \lambda \) was updated so that \( r \) was constant, with \( r = \lambda \left[ \sum_{i,j} (RSM_{ij} - \hat{RSM}_{ij})^2 \right] / \left[ \sum_i \hat{y}_i \log(y_i) \right] \). We studied networks with several different values of this ratio, \( r \). We also experimented with using a constant \( \lambda \) throughout training but found this constant-ratio method leads to better object recognition performance (data not shown).

2.4 Training Procedures

We trained each CORNet-Z network for 100 epochs (250 for VGG-16). For most of our experiments, networks trained with neural data (i.e., with \( r > 0 \)), were trained to minimize the composite cost (Eq. 1) for the first 10 epochs, and thereafter were trained on just the cross entropy loss (this increased from 10 to 100 epochs for the VGG-16 experiments). In other words, we set \( r \to 0 \) after these first 10 epochs. This procedure reduces the computational cost due to evaluating the representational similarity matrices. We performed some experiments in which the neural data regularizer was applied at all training epochs, and saw similar results (not shown).

We kept a static training rate of 0.01 for all networks and a batch size of 128 for CORNet-Z or 256 for VGG-16. We used dropout regularization \cite{21} with 0.5 retention probability for the 3 fully connected layers of all networks. For the training images, we centred the pixels globally across channels. The held-out testing images were not preprocessed, nor were the natural images that were presented to the monkeys in the neuroscience experiments. The trained network weights, and code associated with this paper, can be downloaded at github.com/cfederer/TrainCNNsWithNeuralData.

For each architecture and choice of parameters, we repeated the training from 10 different random initial conditions; results reported are mean ± SEM. This approach has a larger computational cost than does reporting the result of a single training run, but makes it more likely that our findings will generalize because they do not depend on the idiosyncracies of weight initializations.

2.5 Randomized RSMs

For control experiments, to test whether monkey V1 RSMs confer some benefit in neural network training, above that which could be obtained from randomly-drawn RSMs (e.g., to test whether arbitrary RSM constraints were just as good), we repeated our experiments with randomly-generated RSMs in place of the monkey V1 ones. We generated the random RSMs in several different ways.
First, we drew 39-element i.i.d. random vectors from a Gaussian distribution with \( \mu = 5 \) and \( \sigma = 0.582 \); different random vectors were drawn for each image. The number of elements (39) matches the average number of neurons simultaneously observed in one of our neuroscience experiments, and the variance (0.582) matches the variance of the measured neural data. The mean of these randomly-generated vectors differs from the neural data; we later consider random data with the same mean as in the V1 dataset. Given these random vectors, we then calculated the representational similarity matrices from the randomly drawn vectors. In Figs. 4 and 5, that RSM is dubbed “random”.

To test whether having ‘V1-like’ data would suffice for the training, we drew 39-element vectors from i.i.d. Gaussian distributions with the same mean and variance as were seen in the monkey data: \( \mu = 0.495 \) and \( \sigma = 0.582 \). We then used these random vectors to generate a RSM, as described above. This RSM was dubbed “random (V1-stats)”.

Finally, we applied a shuffling procedure to the V1 data, where we randomly permuted the image labels associated with each recorded vector of neural firing rates. As a result of this procedure, the neural responses no longer matched the images that were shown to the monkey. The random permutation was done independently for each neuron. This leads to vectors of firing rates that match (for each neuron, and to all orders) the distributions seen in the monkey data, but removes information about the specific image features those neurons represent (i.e., the neurons’ receptive fields). Similar to the above experiments, we assembled these vectors into a RSM. This is referred to as “V1 shuffled”.

3 Results

We trained neural networks on the composite cost (Eq. 1), with varying ratios \( r \) describing the trade-off between representational similarity cost and categorization cost. We evaluated the trained networks based on categorization accuracy achieved on held-out data (not used in training) from the CIFAR100 dataset. In our figures, black lines indicate networks trained purely for categorization \( (r = 0) \), while red lines indicate networks trained using monkey V1 as a teacher \( (r > 0) \); see Methods). Higher weightings of neural data in the loss function correspond to darker red lines.

3.1 Neural Networks Trained with Monkey V1 as a teacher show improved accuracy

We first present results from the CORNet-Z architecture, and we discuss results from the larger VGG-16 model in Sec. 3.7. We tested the trained models’ ability to classify previously-unseen images from the CIFAR100 dataset (i.e., images not used in training), and quantified the fraction of images correctly labeled. Networks trained to both classify objects and match neural representations (i.e., those with \( r > 0 \)) had better object recognition performance (Fig. 2).

Figure 2: Accuracy in categorizing previously-unseen CIFAR100 images for the CORNet-Z architecture trained with different weighting ratios, \( r \), applied to monkey V1 representation similarity for the first 10 epochs of training. A) Test-set accuracy at each epoch during training. Chance accuracy indicated by dashed black line. Shaded areas are +/- SEM over 10 different random initializations of each model. B) Test-set accuracy for previously-unseen CIFAR100 images at the end of training, as a function of the weight \( (r) \) given to neural representational similarity in the cost function.
3.2 Networks Trained with Monkey V1 as a Teacher Have More Diverse Unit Activations

Why might the networks trained to mimic the monkey brain have better object recognition performance? To gain some insights into this question, we used t-SNE [22] to visualize the unit activations from the first convolutional layer of the networks from the preceding section (i.e., the layer of CORNet-Z that aligns with monkey V1 in terms of depth in the visual pathway). We input images from the CIFAR100 test set into the networks, and used t-SNE to embed those data into two dimensions. We repeated this procedure for networks trained with different ratios $r$ that dictate the trade-off between representational similarity cost and categorization cost.

While the representations of images from different categories remain co-mingled at this low level of the neural network, the representations are more varied for the network trained with a neural representation weighting of $r = 0.1$ (Fig. 3b) than for the network trained no neural data ($r=0$: Fig. 3a). This motivated us to compute the average variance per unit within the V1-like layer of the CORNet-Z networks (variance in activations over the test-set images, averaged over all units in that hidden layer). As the weighting ratio $r$ of neural similarity in the cost function increases, so too does the activation variance (Fig. 3c). While this increase in activation variance is initially associated with increasing object-recognition performance (e.g., up to $r = 0.1$), at higher values of $r$, that increased unit activation variance no longer correlates with higher accuracy (e.g., for $r=3.0$ or $r=4.0$: Fig. 3c).

These data suggest that the improved generalization performance obtained by using neural data in the training procedure (i.e., Fig. 2) could arise in part because the training procedure that uses neural data forces the networks to have more diverse activations in their low-level units.

![Figure 3: Visualizing the hidden unit activations. In panels A and B, we input the same 1280 randomly-selected images from the CIFAR100 test set into DCNNs trained with either $r=0$ or $r=0.1$. We then used t-SNE to embed those high-dimensional unit activations into two dimensions. X- and Y-axes represent the two dimensions of this t-SNE embedding. Colors indicate the object categories for each input image. A) t-SNE embedding of hidden-unit activations from networks trained with no neural data ($r=0$). B) t-SNE embedding of hidden-unit activations from networks trained with neural data $r=0.1$ for the first 10 epochs of training. C. Average variance per unit in the V1-like layer of trained CORNet-Z networks with different weightings, $r$, of neural representations in the cost function. Inset zooms in on the first data points for $r = 0$, 0.01, 0.1.](image)

3.3 The Details of the Teacher Representation Matter

Training neural networks to categorize objects, while matching the image representations seen in monkey V1, leads to improved object recognition performance (Fig. 2). Is that result specific to the image representations in the monkey brain, or would having any arbitrary added RSM constraint in the cost function yield similar results? To answer this question, we performed the same neural network experiment described above (Fig. 2), but using randomly-generated matrices in place of the monkey V1 representational similarity matrices. For these experiments, we used the optimal cost function weighting ratio, $r$, found from our experiments with V1 data: $r=0.1$.

We created randomly-generated RSMs in several different ways (see Methods), used them in place of the monkey V1 RSM in the training procedure, and compared the trained networks’ object categorization performance. Networks trained with the Gaussian random-data RSMs (either with or without matching the mean and standard deviation of the monkey data) underperformed ones trained
with real V1 data for the teacher RSM. Networks trained with shuffled V1 data get nearly the same testing accuracy as those trained with real V1 data (Fig. 4), although the real V1 data still appears to form (by a small margin) the best teacher representation. The teacher RSM may be instructing the DCNN about the distribution of activations (see Sec. 3.2), making the statistics of the data important while not requiring the exact V1 data.

Figure 4: Accuracy in categorizing previously un-seen CIFAR100 images for networks trained with different teacher RSMs: the real monkey V1 RSM (dark red); the image-shuffled V1 data RSM (light red); RSM from random Gaussian vectors drawn with the same mean and standard deviation as the V1 data (purple); and RSM from random Gaussian vectors drawn with different mean than the neural data (blue). These were all trained with a weighting of \( r = 0.1 \) applied to the representational similarity in the loss function. For comparison, the baseline network (trained with no representational similarity cost) is shown in black. A) Testing accuracy over epochs of training. Shaded areas on plot are +/- SEM over 10 different random initializations of each model. B) Test accuracy plotted (same as in A) by type of data used in forming the teacher RSM.

3.4 The DCNN Layer at Which the Representational Similarity Cost is Applied Matters

Above, we found that the details of the RSM used as a teacher for training the DCNN, matter. This led us to wonder whether it matters where in the network that representational similarity cost is applied. To test this, we repeated the experiments from Fig. 2 (training CORNet-Z architecture DCNNs with different weightings for the mismatch between network RSM and monkey V1 RSM), but computed the representation similarity cost for layers other than the V1-identified CORNet-Z layer. The resultant networks all had lower training and testing accuracy than did networks trained with no neural data (data not shown). These results, and those in the preceding section, demonstrate that the performance benefits of the monkey V1 representation teacher require that the monkey V1 representation similarity cost be assigned to the appropriate layer in the DCNN.

3.5 Neural Networks Trained with Monkey V1 as a Teacher Make Errors that are More Sensible

We demonstrated that teaching neural networks to respond to images in a more brain-like manner boosts accuracy in categorizing held-out testing data (Fig. 2). All networks, regardless of regularization, still make frequent errors. However, some errors are worse than others. For example, confusing a mouse for a hamster is more reasonable than confusing a mouse for a skyscraper. This intuition led us to quantify the quality of the errors made by each of our trained networks. For this purpose, we exploited the fact that the 100 classes of labels in the CIFAR100 dataset are grouped into 20 superclasses. One example is “small mammals”, which encompasses mouse, squirrel, rabbit, shrew, and squirrel.

We thus asked, for each trained network, what fraction of their categorization errors were within the correct superclass (e.g., confusing a hamster for a mouse) vs in the wrong superclass (e.g., confusing a mouse for a skyscraper). We performed this test on the network trained with the monkey V1 data as a teacher, with the weighting ratio that yielded the best categorization performance \( (r = 0.1) \). For this network, errors were within the correct superclass 23.9% of the time (Fig. 5). For comparison, for the network trained with no neural data, errors were within the correct superclass 22.8% of the time (Fig. 5).
Networks trained to categorize images, using monkey V1 as a teacher make fewer categorization errors (Fig. 2) and, when they do make errors, those errors are more often within the correct superclass and thus more reasonable (Fig. 5). We find a similar pattern in networks trained with other weighting ratios $r$ (Fig. 5B). Moreover, results with randomized RSMs mirror their impact on categorization: when a teacher RSM improves categorization accuracy, it increases the fraction of errors that are within the correct superclass.

![Table showing error within correct superclass (%) for different weighting ratios](image)

Figure 5: The percentage of errors that are within the correct superclass in the CIFAR100 dataset. A) Percentage of errors within the correct superclass for the network trained with no neural data ($r=0$), with the RSM formed by: monkey V1 data; image-shuffled V1 data (as described in Sec. 3.3); data drawn from Gaussian distribution with the same mean and standard deviation as V1 data; and data drawn from Gaussian distribution with a different mean than the neural data. B) Percentage of errors within the correct superclass for networks trained with the monkey V1 RSM as a teacher, and various weightings of neural similarity in the cost function (Eq. 1).

### 3.6 Neural Networks Trained with Monkey V1 as a Teacher are More Robust to Label Corruption in the Training Data

Given that networks trained using neural data made fewer, and more reasonable, errors, we hypothesized that they generalized more robustly. To test that hypothesis, we did experiments in which some image labels in the training dataset were incorrect (i.e., in the presence of label noise). Datasets will often contain misclassified images, and ideally computer vision networks would not be heavily affected by these misclassifications. To generate a corrupted training set, we switched labels of 10% to 50% of the training dataset, keeping the distribution of classes equal. We only corrupted the training data, and left testing data intact. We then trained CORNet-Z networks with $r=0.1$, and networks without neural data, $r=0$. By the end of the 100 training epochs, networks trained with neural data achieved better testing accuracy than did those without neural data (Fig. 6B). The generalization error (training cost - testing cost) was also lower for networks trained with neural data (Fig. 6C).

![Figure 6: Training CORNet-Z networks with corrupted labels. A) Test set accuracy on networks trained with (red) and without (black) neural data, with different fractions of corrupted labels in the training set. Labels for the test set were not corrupted. B) Generalization error (training cost - testing cost) on networks trained with (red) and without (black) neural data and different fractions corrupted training labels. Line patterns indicate different fractions of corrupted labels in the training set.](image)
3.7 Similar Results are Obtained with the Larger VGG-16 Network

All of the above experiments were obtained on the relatively small CORNet-Z network. This led us to wonder whether monkey V1 data could serve as a similarly effective teacher for deeper neural networks with higher baseline performance. To answer this question, we repeated our above experiments, but using the VGG-16 architecture \cite{20} in place of CORNet-Z. We applied the monkey V1 representational similarity cost at the third convolutional layer of the VGG-16 (see Methods). We found that, with the VGG-16 architecture, the monkey V1 teacher signal improves categorization performance, increases robustness to corrupted labels in the training set, and reduces generalization error, similar to what was observed with the smaller CORnet-Z architecture (Fig. 7).

Figure 7: Accuracy in categorizing previously-unseen CIFAR100 images for the VGG-16 architecture trained on the composite tasks, with different weighting ratios, $r$, applied to monkey V1 representational similarity in the cost function. A) Testing accuracy for each epoch of training. Chance accuracy indicated by dashed black line. Shaded areas on plot are +/- SEM over 10 different random initializations of each model. B) Test accuracy plotted (same as in A) vs the weight of the emphasis on neural representation similarity. C) Test set accuracy on networks trained with (red) and without (black) neural data and 0.1 to 0.5 fraction corrupted training labels. Labels for the testing set have not been corrupted. D) Generalization error (training cost - testing cost) on networks trained with (red) and without (black) neural data and 0.1 fraction corrupted training labels.

4 Discussion

Training the early layers of convolutional neural networks with monkey V1 image representations as a teacher improves those networks’ ability to categorize previously-unseen images. Moreover, when networks had the monkey V1 representation teacher, their mistakes were more reasonable than they were for networks with no such teacher representation. These results were consistent for two different network architectures of vastly different sizes (CORNet-Z and VGG-16), suggesting that they will generalize well to other networks. We emphasize that our goal was not to achieve state-of-the-art classification performance, but rather to determine whether and how we could use the brain as a teacher network for object recognition tasks. We tested this with powerful but relatively small networks. Given our positive findings, future work could fruitfully apply this neural data teaching procedure to larger networks to make better object recognition systems.

We also tried training networks with V4 and IT recorded data from macaque monkeys [16]. Using the CORNetZ architecture, we experimented with forcing the deeper layers of the network to match representations from those recordings. This resulted in lower testing accuracy than the baseline CORNetZ model. However, we cannot conclusively state that V4 or IT data would not be helpful in
training object recognition networks. The decrease in accuracy may be because we did not find the optimal combination of how many epochs for which to train the network on the joint loss function, and how much to emphasize matching the neural representations. It could also be because the data normalization done within the lab that collected those data [16] was different from the lab from which our V1 data came [2]. However, it remains possible that V1 data may be actually more useful for training DCNNs because the specificity of neural responses increases in the deeper layers of the mammalian visual system: the V4 and IT data may not include recordings from neurons that would be useful in our particular object recognition task, whereas V1 neurons carry more generalist representations. Finally, features in DCNNs are only transferable in the early layers and become much less so in deeper layers such as the equivalent to V4 and IT [25], further suggesting that V1 data could be the most useful brain data for training object recognition networks.

References

[1] S.A. Cadena, G.H. Denfield, E.Y. Walker, L.A. Gatys, A.S. Tolias, M. Bethge, and A.S. Ecker. Deep convolutional models improve predictions of macaque V1 responses to natural images. bioRxiv, 2017.
[2] R. Coen-Cagli, A. Kohn, and O. Schwartz. Flexible gating of contextual influences in natural vision. Nat. Neuro., 2015.
[3] R. Geirhos, D.H.J. Janssen, H.H. Sch, J. Rauber, M. Bethge, and F.A. Wichmann. Comparing deep neural networks against humans: object recognition when the signal gets weaker. arXiv, 2018.
[4] R. Geirhos, C. Michaelis, and P. Rubisch. ImageNet-Trained CNNs are Biased Towards Texture; Increasing Shape Bias Improves Accuracy and Robustness. ICLR, 2019.
[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. ICLR, 2015.
[6] U. Gu and M.A.J. van Gerven. Deep Neural Networks Reveal a Gradient in the Complexity. J. Neurosci., 2015.
[7] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. arXiv, 2015.
[8] D.H. Hubel and T.N. Wiesel. Receptive Fields and Functional Architecture of Monkey Striate Cortex. J Physiol., 1968.
[9] W. Kindel, E. Christensen, and J. Zylberberg. Using deep learning to probe the neural code for images in primary visual cortex. J. Vis., 2019.
[10] N. Kriegeskorte. Deep Supervised, but Not Unsupervised, Models May Explain IT Cortical Representation. PLOS, 2014.
[11] N. Kriegeskorte, M. Mur, and P. Bandettini. Representational similarity analysis – connecting the branches of systems neuroscience. Front Syst Neurosci., 2008.
[12] A. Krizhevsky and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. NIPS, 2015.
[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.
[14] J. Kubilius, M. Schrimpf, A. Nayebi, D. Bear, D.L.K Yamins, and J.J. DiCarlo. CORnet: Modeling the Neural Mechanisms of Core Object Recognition. bioRxiv, 2018.
[15] Y. Lecun, Y. Bengio, and G. Hinton. Deep Learning. Nature, 2015.
[16] NJ. Majaj, H. Hong, E.A. Solomon, and J.J. DiCarlo. Simple Learned Weighted Sums of Inferior Temporal Neuronal Firing Rates Accurately Predict Human Core Object Recognition Performance. J. Neurosci., 2015.
[17] P. Mcclure and N. Kriegeskorte. Representational Distance Learning for Deep Neural Networks. Front. in Comp. Neuro., 2016.
[18] R. Rajalingham, E.B. Issa, P. Bashivan, K. Kar, K. Schmidt, and J.J. DiCarlo. 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., 2018.
[19] A. Romero, N. Ballas, S.E. Kahou, and A. Chassang. FitNets: Hints for Thin Deep Nets. ICLR, 2015.
[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. ICLR, 2015.
[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. J Mach Learn Res, 2014.
[22] L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. J. Mach. Learn. Res., 2008.
[23] D.L.K. Yamins and J.J. Dicarlo. Using goal-driven deep learning models to understand sensory cortex. *Nat. Neuro.*, 2016.

[24] D.L.K. Yamins, H. Hong, C.F. Cadieu, E.A. Solomon, D. Seibert, and JJ. Dicarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *PNAS*, 2014.

[25] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *NIPS*, 2014.