Assignment 1

CS 375/Psych 249 (Stanford University, Fall 2017)

1 Background

Sparked by the seminal ideas of Hubel and Wiesel, six decades of work in visual systems neuroscience has shown that the ventral visual stream generates invariant object recognition behavior via a hierarchically-organized series of cortical areas that encode object properties with increasing selectivity and tolerance [1, 2, 3, 4, 5]. Early visual areas, such as V1 cortex, capture low-level features such as edges and center-surround patterns [6, 7]. In contrast, neural population responses in the highest ventral visual areas, inferior temporal (IT) cortex, can be used to decode object category, robust to significant variations present in natural images [8, 9, 10]. The featural content of mid-level visual areas such as V2, V3, and V4 is less well understood, but these areas appear to contain intermediate computations between simple edges and complex objects, along a pipeline of increasing receptive field sizes [11, 12, 1, 13, 14, 15, 16, 17, 18].

Many of these observations can be formalized mathematically via the class of computational architectures known as Hierarchical Convolutional Neural Networks (HCNNs), a generalization of Hubel and Wiesel's simple and complex cells that has been developed over the past 30 years [19, 20]. HCNN models are composed of several retinotopic layers combined in series, each of which is very simple. But together, these layers produce a deep, complex transformation of the input data — like the ventral stream itself.

Building on these broad-stroke insights, recent work in *goal-driven optimization* has sought to optimize parameters of deep neural networks to maximize their performance on high-level, ecologically valid visual tasks [21]. Leveraging computer vision and machine learning techniques, together with large amounts of real-world labelled images used as supervised training data, [22, 23, 24], HCNNs have been produced that achieve near-human-level performance on challenging object categorization tasks [25].

Even though these networks were not directly optimized to fit neural data, their top hidden layers are nonetheless highly predictive of neural responses in IT cortex, both in electrophysiological [21, 26], and fMRI data [27, 28]. These deep, goal-optimized neural networks have thus yielded the first quantitatively accurate, predictive model of the IT population response. These HCNN models map not only to IT, but also to other levels of the ventral visual stream. Intermediate HCNN model layers are highly predictive of neural responses in V4 cortex [21], the dominant cortical input to IT. Similarly, lower model layer filters resembling Gabor wavelets naturally emerge, and are effective models of fMRI voxel responses in early visual cortex [27, 28]. In other words: combining two general biological constraints — the behavioral constraint of object recognition performance, and the architectural constraint imposed by the HCNN model class — leads to greatly improved models of multiple areas through the visual pathway hierarchy.

2 This Assignment

The purpose of this assignment is to reproduce aspects of the results of some recent core papers in the application of deep neural networks to neural data in the ventral visual system [21, 27, 28]. Specifically, there are two basic components to the assignment: network training and network evaluation.

2.1 Network Training

For each of several neural network architectures, train the neural network to solve categorization in the ImageNet 1000-way challenge task [24]. We are asking you to train models from scratch here, because part of the assignment will be to look at evaluations of the network across multiple time points in the training process. The network architectures that you should build include:

- The 1-Stream Variant of AlexNet, as discussed in `https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet`. This model has five convolutional layers and 3 fully connected layers. You may or may not care to include the local response normalization layers.
- One or more smaller variants of deep neural network, with fewer layers and fewer filters per layer. We will leave it up to you to choose exactly what smaller network(s) to use. The purpose of this is to show how network depth and complexity effects both performance on ImageNet as well as fit to neural data. For example, you might want to make a network that is inspired by the HMAX architecture [29, 30].
- (Bonus) A more recent deep network with known higher performance on ImageNet than Alexnet, such as, e.g. Inception v3[1] or VGG[2].

[1] See `https://github.com/tensorflow/models/blob/master/research/slim/nets/inception_v3.py`

[2] See `https://github.com/tensorflow/models/blob/master/research/slim/nets/vgg.py`

- Inspired by the V1-like model in [31][3], you should also implement a hard-coded 1-layer CNN where the filter kernels are a fixed (untrained) Gabor filterbank (see e.g. `http://scikit-image.org/docs/dev/api/skimage.filters.html#skimage.filters.gabor_kernel`), implemented as a Tensorflow model. After the convolution step, this model should have three nonlinearities, including (i) a rectified linear (**ReLu**) step, (ii) maxpooling, and (iii) a local response normalization step. You might also consider trying this model, with a local response normalization as a *first* step, as described in [31].

2.2 Network Evaluation

For each of several time points during training, as well as the final trained point, you should evaluate the network for performance on several metrics associated with neural data. Specifically, we will provide access to the Neural Response Benchmark (NRB), a dataset consisting of responses of inferior temporal (IT) cortex neurons, on the dataset shown in [21]. This dataset consists of approximately 6,000 images of 64 3-dimensional objects in 8 categories (Animals, Boats, Cars, Chairs, Faces, Fruits, Planes, Tables), on photographic backgrounds. The data is collected for three levels of image variation: Low variation ("V0") in which objects are at a fixed position, pose and size; medium variation ("V3"), in which objects are shown at an intermediate level position, pose, and size variability; and high variation ("V6") in which objects are shown at a high level of position, pose, and size variability. (See more about this dataset in section 3.2 below.)

At each evaluation timepoint, you should extract features from the network on all the test images and on these features, compute the metrics described in the sections that follow.

2.2.1 Performance on Test Tasks

Performance, for features from various layers of the trained network on categorization and estimation tasks that are defined by the labels of dataset, including:

1. Categorical 8-way basic categorization task, e.g. Animal vs Car vs Chair &c.
2. Within-category 8-way object identification task, e.g. BMW vs Ford vs Alfa &c.
3. One or more continuous estimation tasks, e.g. object position, size, or pose regression.

The labels for evaluating all these tasks are contained in the NRB dataset package discussed below in section 3.2 below.

For the categorical tasks, we recommend that you use a regularized linear SVM classifier from `scikit-learn`[4]. For the continuous estimation tasks, we recommend that you use a regularized linear regression procedure[5].

As a comparison point, you should also compute whatever performance evaluations you do on model features, also on IT neural features. That is, estimate how good IT neural features are at 8-way object categorization. (After all, IT data is just feature data, just like the model outputs.)

You should perform all these analyses for two subsets of the data:

- All the images — that is, take classifier/regressor training and testing images from variation levels V0, V3, and V6.
- Just the high variation images — that is, take classifier/regressor training and test images from variation level V6 alone.

2.2.2 Representational Similarity Analysis (RSA)

Background: The Representational Dissimilarity Matrix (RDM) of a feature representation F with n images and m feature dimensions is the $n \times n$ matrix M defined by

$$M(i, j) = 1 - corr(F[i], F[j]),$$

where $F[i]$ is the length-m feature vector response to image i, and $corr$ is the pearson product-moment correlation[6]. RDMs describe how the m-dimensional feature representation lays out images relative to each other. To compare RDMs between

[3] And see code `https://github.com/npinto/v1like`

[4] See e.g. `http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html`.

[5] See `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeCV.html`

[6] `https://en.wikipedia.org/wiki/Pearson_correlation_coefficient`

two different representations F_1 and F_2, each $n \times n$ symmetric matrix M_1, M_2 is flattened into a length $n \cdot (n-1)/2$-dimensional vector, and then the Spearman rank-order correlation[7] between the two flattened vectors is computed.[8] This type of analysis comparing two RDMs is sometimes called "Representational Similarity Analysis", and it measures how similar the two representations treat the images. For more about RSA, see [32], which introduced its use in neuroscience.

What you should actually measure: What you should measure is the similarity of the for the trained neural network features from various layers and the RDM of the IT neural data. That is, (i) compute the RDM for the model features, (ii) compute it for IT features from the neural data, and (iii) compare the RDMs as described above (using the spearmanr correlation metric). Because there are approximately 6,000 images in the test dataset, RDMs directly computed from that data are large a bit hard to work with and/or visualize. For this reason, you can reduce the RDM on a per-object basis by taking the mean of the features for all images of a given object. Since there are 64 objects in the dataset, this leads to a 64 x 64 RDM matrix, which is much easier to work with. In this case, the 64x64 is also nicely visualizable, if the objects are ordered by category, with category block-diagonals emerging. You should compute these 64x64 RDMs from:

- All images in the dataset,
- just the Variation 6 alone (that is, subselect the stimuli with "var" attribute equal to $V6$ before computing the average vectors and then the RDM). [And of course, do the same thing for both the model and neural features.]

2.2.3 Neural Response Regression

The ability of the neural network features at various layers to regress IT neural data responses. That is, for each neuron n_j in the 168-neuron sample, use linear regression to identify weights w_{ij} and bias b_j on model features f_i from various network layers, such that

$$n \sim b_j + \sum_{i=0}^{n} w_i f_i.$$

We recommend you use PLS regression[9] or Ridge regression[10] for this purpose. (If you do use PLS regression in `scikit-learn`, be sure to set the scale parameter to False.)

2.2.4 Visualization of Layer 1 Filters

At each training timepoint, you should also visualize the filters of the first layer of network, essentially so that you can see whether (and how) Gabor-like and center-surround patterns arise. For the purposes of this assignment, merely visualizing the filters is sufficient. As a bonus, you could quantify how "Gabor-like" each filter is, plot the distribution of preferred orientations vs frequencies, or the extent to which color opponency[11] is present.

3 Datasets

There are two basic datasets in this assignment: ImageNet [24], on which you will train models and the Neural Representation Benchmark (NRB) dataset [33], on which you will test models. Both datasets are already pre-installed on the Google Cloud setups that you will be given.

3.1 ImageNet

The ImageNet dataset is located at `/datasets/TFRecord_Imagenet_standard` on your google cloud setup. We have converted the dataset into TFRecord format[12]. The dataset consists of approximately 1.2M images, broken down into approximately 1150 TFRecords files each containing aproximately 1,000 records. Each record has three fields: an "images" field, which contains array data of the actual imagenet images, of type float64; a "labels" field, which contains the category label for each image, as an integer from 1 to 1000; and a "labels_0" field, which is identical to the "labels" field but is 0-indexed. The Imagenet dataset is loaded into TFUtils using the `ImageNetDataProvider`.

[7] `https://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient`

[8] If starting with a square-form version of the correlation matrix, be careful to only use the upper or lower triangular portion of the flattened matrix when computing the spearman. Otherwise you'll be double-counting all but the diagonal, which is trivially 0 anyhow. Have a look at `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html`, `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.squareform.html`, and `https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.triu_indices.html`.

[9] `http://scikit-learn.org/stable/modules/generated/sklearn.cross_decomposition.PLSRegression.html`

[10] See `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html` and `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeCV.html`.

[11] see e.g. `http://courses.washington.edu/psych333/handouts/coursepack/ch14-Color_vision.pdf`.

[12] Details are on this page: `https://www.tensorflow.org/api_guides/python/python_io`

3.2 Neural Data

The data associated with the NRB is located at `/datasets/neural_data/tfrecords_with_meta` on your google cloud install. Like the ImageNet images, we have converted the data into TFRecords format. The NRB consists of 5,760 images, broken down into 23 TFRecords each containing 256 records. Each image in the NRB was constructed by rendering one of 64 3-dimensional objects at some chosen position, pose, and size, on a randomly chosen background photograph. The fields in the tfrecords contain:

- "images": containing the actual images of the dataset, in array format with dataset float64
- "category": containing the string name of the category of the object, e.g. one of Animal, Boat, Car, Chair, Face, Fruit, Plane or Table.
- "obj": containing the string name of the object (of 64 possible objects) in the image.
- "var": containing the string name of the variation level of the image parameters from which the image was drawn. Specifically, these include "V0" (low variation), "V3" (medium variation), and "V6" (high variation). There are 640 V0 images (10 images per object), 2560 V3 images (40 images per object), and 2560 V6 images (40 images per object), for a total of 90 images per object.
- "ty": the horizontal position of the object in the image, in type float64.
- "tz": the vertical position of the object in the image, in type float64.
- "size": the size of the object in the image, in type float64.
- "ryz—rxy—rxz_semantic": the rotation of the object in the r_{yz}, r_{xy} or r_{xz} planes (corresponding to in-plane rotation, rotation around the horizontal axis, and rotation around the vertical axis, respectively).
- And a variety of other data about the image, including object centroid, volume, bounding box, and so on.

All the data needed to evaluate the metrics described in section 2.2 above are contained in these fields.

4 Code

4.1 TFUtils

The primary interface to tensorflow that we will use in this project is the TFUtils library[13]. TFUtils is a package we have designed to standardize the storage of results for large-scale training of Tensorflow models. We will discuss the API and usage of TFUtils in class, and illustrate how it works during tutorials.

The interface to TFUtils is basically to provide a *configuration*. This is a python dictionary of information about what you want to train and/or test in any given session. Your job in this assignment is to figure out how to fill in the blanks of the TFUtils configuration file to describe both training a neural network and then testing it on neural data.

As we will discuss in class, TFUtils stores results in a MongoDB database[14]

4.2 Training

Sample code to get you going on the training phase of the project is located at on CS375 code repository at `https://github.com/neuroailab/cs375/blob/master/2017/assignment1/train_imagenet.py`.

4.3 Testing

Sample code to get you going on the training phase of the project is also located at on CS375 code repository at `https://github.com/neuroailab/cs375/blob/master/2017/assignment1/train_imagenet.py`.

4.4 Strategy

Because there are two phases of the project, and the first phase (training the neural networks) is time consuming, we have made a partially pre-trained AlexNet model available so that you can get going on phase two (writing and perfecting your evaluation code) while your network is training. The TFUtils database address for accessing the pre-trained AlexNet is given as the default parameteres in the sample testing code.

[13] `https://github.com/neuroailab/tfutils`
[14] `http://mongodb.com`

4.5 Lab Reports

We expect results to be presented as a kind of "Lab Report" in the form of a Jupyter notebook[15]. We've provided an example of what your Lab Report notebook should look like on the CS375 repo at `https://github.com/neuroailab/cs375/blob/master/2017/assignment1/cs375-assignment-1.ipynb`.

References

[1] DiCarlo, J. J., Zoccolan, D. & Rust, N. C. How does the brain solve visual object recognition? *Neuron* **73**, 415–34 (2012).
[2] Malach, R., Levy, I. & Hasson, U. The topography of high-order human object areas. *Trends in cognitive sciences* **6**, 176–184 (2002).
[3] Felleman, D. & Van Essen, D. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex* **1**, 1–47 (1991).
[4] Rust, N. C. & DiCarlo, J. J. Selectivity and tolerance ("invariance") both increase as visual information propagates from cortical area v4 to it. *J Neurosci* **30**, 12978–95 (2010).
[5] Connor, C. E., Brincat, S. L. & Pasupathy, A. Transformation of shape information in the ventral pathway. *Curr Opin Neurobiol* **17**, 140–7 (2007).
[6] Carandini, M. *et al.* Do we know what the early visual system does? *J Neurosci* **25**, 10577–97 (2005).
[7] Movshon, J. A., Thompson, I. D. & Tolhurst, D. J. Spatial summation in the receptive fields of simple cells in the cat's striate cortex. *The Journal of physiology* **283**, 53–77 (1978).
[8] Majaj, N. J., Hong, H., Solomon, E. A. & DiCarlo, J. J. Simple learned weighted sums of inferior temporal neuronal firing rates accurately predict human core object recognition performance. *The Journal of Neuroscience* **35**, 13402–13418 (2015).
[9] Yamane, Y., Carlson, E. T., Bowman, K. C., Wang, Z. & Connor, C. E. A neural code for three-dimensional object shape in macaque inferotemporal cortex. *Nat Neurosci* (2008).
[10] Hung, C. P., Kreiman, G., Poggio, T. & Dicarlo, J. J. Fast readout of object identity from macaque inferior temporal cortex. *Science* **310**, 863–866 (2005).
[11] Freeman, J. & Simoncelli, E. Metamers of the ventral stream. *Nature Neuroscience* **14**, 1195–1201 (2011).
[12] DiCarlo, J. J. & Cox, D. D. Untangling invariant object recognition. *Trends Cogn Sci* **11**, 333–41 (2007).
[13] Schmolesky, M. T. *et al.* Signal timing across the macaque visual system. *J Neurophysiol* **79**, 3272–8 (1998).
[14] Lennie, P. & Movshon, J. A. Coding of color and form in the geniculostriate visual pathway (invited review). *J Opt Soc Am A Opt Image Sci Vis* **22**, 2013–33 (2005).
[15] Schiller, P. Effect of lesion in visual cortical area v4 on the recognition of transformed objects. *Nature* **376**, 342–344 (1995).
[16] Gallant, J., Connor, C., Rakshit, S., Lewis, J. & Van Essen, D. Neural responses to polar, hyperbolic, and cartesian gratings in area v4 of the macaque monkey. *Journal of Neurophysiology* **76**, 2718–2739 (1996).
[17] Brincat, S. L. & Connor, C. E. Underlying principles of visual shape selectivity in posterior inferotemporal cortex. *Nat Neurosci* **7**, 880–6 (2004).
[18] Yau, J. M., Pasupathy, A., Brincat, S. L. & Connor, C. E. Curvature processing dynamics in macaque area v4. *Cerebral Cortex* bhs004 (2012).
[19] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybernetics* (1980).
[20] LeCun, Y. & Bengio, Y. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 255–258 (1995).
[21] Yamins*, D. *et al.* Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences* (2014).
[22] Krizhevsky, A., Sutskever, I. & Hinton, G. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* (2012).
[23] Bergstra, J., Yamins, D. & Cox, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of The 30th International Conference on Machine Learning*, 115–123 (2013).
[24] Deng, J., Li, K., Do, M., Su, H. & Fei-Fei, L. Construction and analysis of a large scale image ontology. In *Vision Sciences Society* (2009).
[25] Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[26] Cadieu, C. F. *et al.* Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS computational biology* **10**, e1003963 (2014).
[27] Khaligh-Razavi, S. M. & Kriegeskorte, N. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLOS Comp. Bio.* (2014).
[28] Güçlü, U. & van Gerven, M. A. Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *The Journal of Neuroscience* **35**, 10005–10014 (2015).
[29] Serre, T., Oliva, A. & Poggio, T. A feedforward architecture accounts for rapid categorization. *Proc Natl Acad Sci U S A* **104**, 6424–9 (2007). 0027-8424 (Print) Journal Article.
[30] Riesenhuber, M. & Poggio, T. Models of object recognition. *Nat Neurosci* **3 Suppl**, 1199–204. (2000).
[31] Pinto, N., Cox, D. D. & DiCarlo, J. J. Why is Real-World Visual Object Recognition Hard. *PLoS Comput Biol* (2008).
[32] Kriegeskorte, N. *et al.* Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron* **60**, 1126–41 (2008).
[33] Cadieu, C. *et al.* The neural representation benchmark and its evaluation on brain and machine. In *International Conference on Learning Representations* (2013).

[15] `http://jupyter.org/`