Assignment 1

CS 375/Psych 249 (Stanford University, Autumn 2018)

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 Assignment Questions

The purpose of this and next assignments is to reproduce aspects of the results of some recent core papers in the mapping from 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. In this assignment, we will mainly focus on the network training and only cover a little bit of network evaluation.

2.1 ResNet18 Training on Subsampled ImageNet

As a beginning, please train a ResNet18 [29] to solve categorization in the ImageNet 1000-way challenge task [24]. We are asking you to train this model from scratch here, because part of this assignment will be to look at evaluations of the network across multiple time points in the training process. However, training a ResNet18 from scratch on the full ImageNet needs a lot of computational resources, which effectively means more credits. To avoid that, we only ask you to train the ResNet18 on a smaller ImageNet, which is stored at "/mnt/data/ImageNet/image_label_quarter" on your Google Cloud instances.

Here are some detailed instructions on training:

- You are **required** to use TFUtils (see section 5.1) to train the network.
- You can mostly follow the example AlexNet training code used in your Google Cloud tutorial and make necessary changes.

- The most important change is to replace the model function for building AlexNet using a function building ResNet18. You should **feel free** to use publicly available codes to build ResNet18 network.
- In addition to this change, you also need to change the data augmentation style in the ImageNet data provider, which can be done through setting parameter "prep_type" to be "resnet" when initializing the ImageNet class if you follow the AlexNet example. Data augmentation methods have been proved critical to get good performances for training deep neural networks on ImageNet. Notice that the data augmentation methods used here are not exactly the same as the methods described in [29], but following `https://github.com/pytorch/examples/tree/master/imagenet`, which has successfully reproduced ResNet results. In order to help you understand the difference between these two data augmentation methods, we would like to ask you to compare the behaviors of ResNet ImageNet and AlexNet ImageNet data provider. In the ipython notebook you are expected to submit, please describe the procedures of getting one augmented image from the original image using both two data augmentation methods.
- For other training settings such as network initializations, weight decay, and learning rate schedule, please try your best to follow the same setting as [29].

In summary, we are expecting you to do the following:

1. Use TFUtils to train a ResNet18 on small ImageNet.
2. Show the validation performance trajectory, smoothed training loss trajectory, and learning rate trajectory for this network in ipython notebook submitted.
3. In the ipython notebook, describe the procedures of getting augmented images from the original images using both data augmentation methods for ResNet and AlexNet.

2.2 Network Evaluation

To evaluate the network for performances on mapping to neural data, please first choose several training intermediate checkpoints which you think are important (must include the final checkpoint). Specifically, we will provide access to the Ventral Neural Dataset (VND), a dataset consisting of responses of inferior temporal (IT) and V4 cortex neurons, on the images shown in [21]. (See more about this dataset in section 3.2 below.)

For all checkpoints, you should utilize TFUtils function "test_from_params" to extract features from **three layers** which are lower layer, middle layer, and higher layer correspondingly in the network on the same images presented to the neurons and then use these features to perform following tasks:

- Categorical 8-way basic categorization task, e.g. Animal vs Car vs Chair, etc.
- One continuous estimation task, you can choose from predicting object position, size, or pose regression.

The data needed for training and evaluating all these tasks is contained in the VND dataset described in section 3.2.

For the categorical tasks, we recommend that you use a regularized linear SVM classifier from `scikit-learn`[1]. For the continuous estimation tasks, we recommend that you use a regularized linear regression procedure[2].

As a comparison point, you should also compute whatever performance evaluations you do on model features, also on IT and V4 neural features. That is, estimate how good IT and V4 neural features are at 8-way object categorization and the continuous estimation task you choose.

You are only asked to perform these analyses for a subset of the data, which is the middle and high variation images — that is, take images from variation level V3 and V6. In order to evaluate the performances of these tasks, you need to randomly split the images into training and evaluating sets. And to get an estimation of noises due to the random splits, you should run these tasks on at least three splits.

In summary, we are expecting you to do the following:

1. Choose several important checkpoints saved during training ResNet18.
2. From the ResNet18 network, choose three layers that corresponds to lower, middle, and higher layers.
3. For each checkpoint, use "test_from_params" function in TFUtils to extract features from all three layers. In the same function, train SVM classifiers or regularized linear regressors to perform the tasks using these features on training images and then validate the classifiers or regressors trained on validation images. You should have at least three splits to have a noise estimation about the metrics reported for these two tasks.
4. Perform the same tasks using IT and V4 neural features. Show all these results in ipython notebook submitted, and compare network and neural performances. Describe your findings.

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

2.3 Smaller Network Training on Subsampled ImageNet

Design one smaller variant of **AlexNet** (at ResNet18 is already very deep), with fewer layers and fewer filters per layer. We will leave it up to you to choose exactly what smaller network 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.

Therefore, after finishing training this smaller AlexNet, please run the same neural data evaluations in the previous section.

In summary, what you are expected to do is similar as that in previous two sections. But just for smaller AlexNet.

3 Datasets

There are two basic datasets in this assignment: ImageNet [24], on which you will train models and the Ventral Neural Dataset (VND) dataset [30], on which you will test models. Both datasets are already pre-installed on your google cloud instances.

3.1 ImageNet

The full ImageNet dataset is located at "/mnt/data/ImageNet/image_label_full" on your google cloud instances. We have converted the dataset into TFRecord format[3]. The dataset consists of approximately 1.2M images, broken down into 1024 TFRecords files each containing aproximately 1,200 records. Each record has two fields: an "images" field, which contains array data of the actual imagenet images stored in jpeg format, of type string; a "labels" field, which contains the category label for each image, as an integer from 0 to 999.

As mentioned in section 2.1, we ask you to train the networks on a smaller ImageNet dataset located at "/mnt/-data/ImageNet/image_label_quarter". This dataset is generated by subsampling around 300 training TFRecords from the full dataset and keeping all validation TFRecords.

An example for loading the ImageNet dataset is at `https://github.com/neuroailab/tfutils/blob/master/tfutils/imagenet_data.py`. If you are following AlexNet training example, this data provider is already used. However, as mentioned in previous section 2.1, you need to change parameter "prep_type" for training ResNet18. Besides this change, you also need to use a different number of steps per epoch for the smaller ImageNet.

3.2 Neural Data

In this section, we will introduce the Ventral Neural Dataset [30], which will be used throughout the quarter. As described in class, this dataset contains neuronal responses of 296 neurons from two animals Chabo and Tito through presenting 5760 images using standard rapid visual stimulus presentation (RSVP). Each image was constructed by rendering one of 64 3-dimensional objects at some chosen position, pose, and size, on a randomly chosen background photograph. Specifically, each image was only presented for 100ms to the animals.

We have stored this dataset in one hdf5 file located at "/mnt/data/neural_data/ventral_neural_data.hdf5" in your google cloud instances. An example function loading it using Tensorflow is at `https://github.com/neuroailab/cs375/blob/master/2018/assignment1/neural_data.py`. You can also use this function directly in TFUtils.

Besides, we can also load the data outside of Tensorflow to take a look. In order to do that, we will use package `h5py`. If you find `h5py` is not installed on your google cloud instances, you can install that through command "pip install h5py". Below is an example of loading the data and showing its contents.

```
In [1]: import h5py

In [2]: data_path = '/mnt/data/neural_data/ventral_neural_data.hdf5'

In [3]: fin = h5py.File(data_path, 'r')

In [4]: fin.keys()
Out[4]:
[u'image_meta',
u'images',
u'neural_meta',
u'time_averaged',
```

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

```
u'time_averaged_trial_averaged',
u'time_binned',
u'time_binned_trial_averaged']
```

These keys mean:

- "images": an array containing the actual images of the dataset, of which the shape is [5760, 256, 256]
- "image_meta": a group of arrays describing the meta data for each image:
  - "category": an array containing the string name of the category of the object for each image, e.g. one of Animal, Boat, Car, Chair, Face, Fruit, Plane or Table.
  - "object_name": an array containing the string name of the object (of 64 possible objects) in the image.
  - "variation_level": 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.
  - "translation_y": the horizontal position of the object in the image, in type float64.
  - "translation_z": the vertical position of the object in the image, in type float64.
  - "size": the size of the object in the image, in type float64.
  - "rotation_[xy,yz,yz]": 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).
- "neural_meta": a group of arrays describing the meta data for each neuron:
  - "V4_NEURONS": containing indexes of neurons belonging to V4 area
  - "IT_NEURONS": containing indexes of neurons belonging to IT area
  - "AIT_NEURONS": containing indexes of neurons belonging to anterior IT area
  - "CIT_NEURONS": containing indexes of neurons belonging to central IT area
  - "PIT_NEURONS": containing indexes of neurons belonging to posterior IT area
  - "ANIMAL_INFO": containing animal names for all neurons
  - "ARRAY_INFO": containing array information for all neurons. There are three arrays for each animal. These arrays are named as "P" (posterior), "M" (middle), and "A" (anterior). (see Figure 2.B in [30] for details)
- "time_averaged_trial_averaged": an array containing time averaged and trial averaged neural responses in shape of [5760, 296], in type float64
- "time_binned_trial_averaged": an array containing trial averaged neural responses for each of the 20ms time bins starting from 0ms to 200ms, in shape of [5760, 11, 296] and type float64
- "time_averaged": a group of arrays containing time averaged neural responses for different trials. As numbers of trials are different for images in different variation levels, so the responses are separated by the variation levels:
  - "variation_level_0": an array containing the neural responses in type float64 and shape of [28, 640, 296], where 28 is the number of trials
  - "variation_level_3": an array containing the neural responses in type float64 and shape of [51, 2560, 296], where 51 is the number of trials
  - "variation_level_6": an array containing the neural responses in type float64 and shape of [47, 2560, 296], where 47 is the number of trials
- "time_binned": a group of groups of arrays containing neural responses for different trials and different 20ms time bins. The neural responses are first stored by different starting time points (0ms, 20ms, 40ms, ..., 200ms). Each 20ms time bin will contain a group of arrays structured in the same way as "time_averaged" data.

4 Lab Reports

We expect results to be presented as a kind of "Lab Report" in the form of a Jupyter notebook[4], together with the codes used for getting these results. In general, with the ipython notebook and submitted codes, other readers should be able to reproduce your results. Therefore, except the results required and explanations needed, you should also have some guidelines about how to reproduce these results using the codes submitted.

We've provided an example of what your Lab Report notebook can look like on the CS375 repo at `https://github.com/neuroailab/cs375/blob/master/2017/assignment1/cs375-assignment-1.ipynb`.

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

5 Tools and Examples

5.1 TFUtils

The primary interface to tensorflow that we will use in this project is the TFUtils library[5]. TFUtils is a package we have designed to provide general framework for large-scale Tensorflow network training and also to standardize the storage of training and validation results. Please refer to the "Tools 1a" tutorial for examples about the API and usage of TFUtils. You can also check the documentation of TFUtils hosted in `http://neuroailab.stanford.edu/tfutils/index.html`.

As we have shown in tutorials, TFUtils stores results in a MongoDB database[6], which should automatically run in the backend on your Google Cloud instances at port 29101.

5.2 Example for Network Training

Sample code for the network training is located at tutorials of TFUtils code repository at `https://github.com/neuroailab/tfutils/blob/master/tutorials/train_alexnet.py`.

5.3 Example for Evaluating on Neural Data

We will release an helping script for evaluating on neural data **very soon**.

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).

[5] `https://github.com/neuroailab/tfutils`
[6] `http://mongodb.com`

[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] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778 (2016).
[30] 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. (*In press.*). *J. Neurosci.* (2015).