Assignment 3

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

1 Background & Purpose

The retina comprises the first component of visual processing, and even at this level, the retina must compress visual information from 100 million photoreceptors down to 1 million ganglion (output) cells. In just a few layers, the retina predicts object motion [1], predicts complex spatiotemporal patterns [2], and can reduce spatiotemporal redundancy in natural scenes [3].

In class, we fit an LN model [4] using spike-triggered analysis [5] to a simple white noise bar stimulus and saw that it allowed us to interpret the spatiotemporal receptive field of the ganglion cell (RGC) and characterize its ON/OFF response. However, the LN model is not without its drawbacks. There are many cell types in the retina, and the LN model ignores cell types as it consists of a single linear filter which pools over the input stimulus. Moreover, the retina is multilayered (e.g. photoreceptor/bipolar layer, amacrine cell layer, ganglion cell layer), so it is not clear *a priori* that we have adequately modeled this functionally-relevant anatomy with an LN model. In fact, LN models are poor predictors of the retina's response to natural scenes.

The purpose of this assignment is to reproduce aspects of the results in [6], in which it is shown that deeper CNN models are in fact better models of the retinal responses than shallower LN models. The CNN architecture you will be training is based on the three layer architecture in [6]. By having multiple filters at each convolutional layer, the network is not only multilayered, but could potentially learn multiple "cell types" in its architecture. This allows the models to generalize better across stimuli, capture interunit responses in the retina, as well as form a general feature basis for capturing retinal phenomena over long timescales.

2 Datasets

As with previous assignments, there are two basic components to the assignment: network training and network evaluation. Unlike in the previous datasets, here the training and testing are both on neural data responses (as opposed to training the network to solve a task, and then just testing against neural data).

The stimuli you will be working with are movies where each image is 50 x 50 consisting of 40 frames each (recall that the 40 frames corresponds to the temporal integration time of the RGC at a 100 Hz frame rate). In particular, you will be training models on a white noise stimulus, which consists of binary checkers, and a natural scene stimulus which consists of a sequence of jittered natural images sampled from a natural image database [7]. The labels will be the response of 5 OFF-type tiger salamander RGC responses recorded in the Baccus Lab at Stanford.

The datasets provided consist of the white noise and natural scene imagesets, along with the firing rates of 5 tiger salamander RGCs (binned using 10 ms bins and smoothed with a 10 ms Gaussian filter). The whitenoise stimulus consists of 50 x 50 spatial checkers, each of which spans 55 μm x 55 μm on the retina. At this resolution, they can differentially activate nonlinear subunits [8, 9] within the $\sim$ 250 μm salamander ganglion cell receptive field center. The images for the natural scene dataset were taken from [7]. Both whitenoise and natural scene datasets consist of 40 frames and are already pre-installed on the Google Cloud setups that you will be given.

The whitenoise and natural scene datasets are located at `/datasets/deepretina_data` on your Google Cloud setup (in the `tf_records/whitenoise/` and `tf_records/naturalscene/` subdirectories, respectively). **Please do not distribute this data with anyone outside of this class.** We have converted both the whitenoise and natural scene datasets into TFRecord format[1]. The whitenoise dataset consists of 323762 train images and 5957 test images and the natural scene dataset consisting of 323756 train images and 5956 test images. Each record has two fields: an "images" field, which contains array data of the actual images the RGC was shown (which are of shape 50 x 50 x 40), of type float32; and a "labels" field, which contains the firing rate of each of the 5 RGCs (also of type float32). The retinal dataset is loaded into TFUtils using the `retinaTF` data provider. `train*` and `test*` filenames for the TFRecord filenames in each directory are the training and heldout test TFRecords, respectively.

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

3 Network Training

For each of the two stimulus sets (whitenoise and natural scenes), you will train an LN model and a three-layer CNN. You will then be asked to evaluate these models on both stimuli, namely each network will be trained on whitenoise and tested on both whitenoise and natural scene test data, and will also be trained on natural scenes and tested on both whitenoise and natural scene test data. Specifically, the network architectures that you should implement, as separate model functions in TFUtils, include:

- **The LN model:** As we discussed in class, this can be viewed simply as a one layer neural network that consists of a fully connected layer followed by a nonlinearity. Rather than using spike-triggered analysis [5] to fit the model (like we did in the tutorial for the simple bar stimulus), we will instead train this model via SGD for efficiency. In this case, in order to ensure non-negative predicted firing rates for the model, the nonlinearity we will use will be the softplus nonlinearity. The recommended optimizer is Adam with a learning rate of 1e-3, and L2 regularization of 1e-3 on the fully connected layer weights, though you are more than welcome to choose your own hyperparameters.
- **The three layer CNN:**
  - The first convolutional layer consists of 16 filters of size 15x15, with stride 1 and VALID padding, with L2 regularization of 1e-3, followed by a ReLU.
  - You will add Gaussian noise to the first layer activations (prior to the ReLU), with mean 0 and standard deviation 0.1, only during training and not during validation. This is just a way to regularize the model and is similar in idea to Dropout.
  - The second convolutional layer consists of 8 filters of size 9x9 with stride 1 and VALID padding, with L2 regularization of 1e-3, followed by a ReLU.
  - Again, you will add Gaussian noise to the second layer activations (prior to the ReLU), with mean 0 and standard deviation 0.1, only during training and not during validation.
  - The final layer is a fully connected layer which has an output dimension of 5 units, followed by a softplus nonlinearity, with L2 regularization of 1e-3.
  - It is recommended to use Xavier initialization for the convolutional layers with the biases zero initialized. For the final fully connected layer, it is recommended to use a random normal initialization with mean 0 and standard deviation of 0.05, with the biases zero initialized. When training on whitenoise, it is recommended you use Adam with a learning rate of 1e-3, and when training on natural scenes, it is recommended that you use Adam with a learning rate of 1e-4. In both cases, the recommended training batch size is 5000. Again, you are more than welcome to choose your own hyperparameters.

For both models, the loss function you will use to fit the model to the data is the Poisson loss function (as a TFUtils loss per case function), which is given by

$$\ell(y^{(i)}, \hat{y}^{(i)}) = \frac{1}{N}\left(\sum_{k=1}^{N}\left(\hat{y}_k^{(i)} - y_k^{(i)} \cdot \log(\hat{y}_k^{(i)} + \varepsilon)\right)\right) \tag{1}$$

where N is the number of ganglion cells (in this case 5), $\hat{y}_k^{(i)}$ is the predicted firing rate of the given ganglion cell, and $y_k^{(i)}$ is the true firing rate of the ganglion cell (k indexes the ganglion cells and (i) indexes the examples in the batch). Note that we have an extra "fudge" factor ε to avoid running into numerical issues (set $\varepsilon =$ 1e-8 in your implementation).

To be clear, in all you will run four trainings: the two models (LN and CNN) each on the two training sets (white noise and natural scenes).

4 Network Evaluation

At the end of training, you should evaluate the both networks for performance on both the whitenoise test data and natural scene test data, by choosing the point in training that resulted in the lowest test set loss (on the same stimulus class you trained on). It is up to you to decide how often to save weights and metrics: we recommend saving every 50 steps for both models. The metric will be the Pearson correlation coefficient for each of the 5 ganglion cells, between the model's predicted firing rate for that ganglion cell and its true firing rate. You will implement this as a TFUtils agg func, after concatenating the responses together along the batch dimension in your online agg func.

To help you, it is recommended that you actually implement the Pearson correlation coefficient in NumPy using the formula:

$$r = \frac{\sum_{i=1}^{n}(\hat{y}^{(i)} - s\left(\hat{y}^{(i)}\right))(y^{(i)} - s\left(y^{(i)}\right))}{std(\hat{y}^{(i)}) \cdot std(y^{(i)})}, \tag{2}$$

where n is the number of validation examples (5957 for whitenoise and 5956 for natural scenes test data), $y^{(i)}$ is the 5 dimensional vector of true firing rates for each of the 5 ganglion cells, and $\hat{y}^{(i)}$ is the 5 dimensional vector of predicted firing rates for each of the 5 ganglion cells, $s(\cdot)$ represents the sample mean along the batch, and $std(\cdot)$ represents the sample standard deviation across the batch, for each of the 5 ganglion cells.

When performing the evaluation for any network at any timepoint, you should run correlations on both natural scenes and white noise stimuli. That is, in all you will will have eight results at the end: for each of the two models (LN and CNN), and each of the two training regimes (white noise and natura scenes), you will results testing the trained model on both testing sets. Ideally, you should see that the three-layer CNN generalizes better than the LN model both within stimuli (e.g. train on whitenoise and test on whitenoise; train on natural scenes and test on natural scenes) as well as across stimuli classes (e.g. train on whitenoise and test on natural scenes; train on natural scenes and test on whitenoise).

5 Code

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/assignment3/train_retina.py`.

Training: When training, it is recommended to title either your TFUtils `dbname` or `collname` that you are saving to with the stimulus type you are training on, e.g. `white_noise_train` or `natural_scenes_train`.

Testing: In this assignment you will run your training and testing code simultaneously. In other words, you will supply your `AggFunc` to the *training* specification in TFUtils, as part of a validation to be periodically during the training process. This way, you do not have to run a separate `test_from_params` process after your training is finished — all your testing will already be done for you, and you simple have to look up and plot the results. Note that the `AggFunc` should return a dictionary, with results for the two correlation evaluations discussed above (e.g. equation 2 on both white noise and natural scenes). We recommend that the kys in this dictionary clearly label which correlation result is which, e.g. like `white_noise_testcorr` and `natural_scenes_testcorr`.

As will previous assignments, you will turn in your results in the form of a Jupyter notebook Lab report, with the train/test loss curves, as well as the performance on all 4 train-test comparisons for each of the two models.

References

[1] Baccus, S. A., Olveczky, B. P., Manu, M. & Meister, M. A retinal circuit that computes object motion. *The Journal of Neuroscience* **28**, 6807–6817 (2008).

[2] Hosoya, T., Baccus, S. A. & Meister, M. Dynamic predictive coding by the retina. *Nature* **436**, 71–77 (2005).

[3] Srinivasan, M. V., Laughlin, S. B. & Dubs, A. Predictive coding: a fresh view of inhibition in the retina. *Proceedings of the Royal Society of London. Series B. Biological Sciences* **216**, 427–459 (1982).

[4] Chichilnisky, E. A simple white noise analysis of neuronal light responses. *Network: Computation in Neural Systems* **12**, 199–213 (2001).

[5] Schwartz, O., Pillow, J. W., Rust, N. C. & Simoncelli, E. P. Spike-triggered neural characterization. *Journal of Vision* **6**, 484–507 (2006).

[6] McIntosh, L. T., Maheswaranathan, N., Nayebi, A., Ganguli, S. & Baccus, S. A. Deep learning models of the retinal response to natural scenes. *Advances in Neural Information Processing Systems* **29**, 1369–1377 (2016).

[7] Tkačik, G. *et al.* Natural images from the birthplace of the human eye. *PLoS One* **6**, e20409 (2011).

[8] Hochstein, S. & Shapley, R. Linear and nonlinear spatial subunits in y cat retinal ganglion cells. *The Journal of Physiology* **262**, 265 (1976).

[9] Gollisch, T. Features and functions of nonlinear spatial integration by retinal ganglion cells. *ournal of Physiology-Paris* **107**, 338–348 (2013).