Encoding the natural response of primate retina

Category: Life Sciences

Nandita Bhaskhar[1], Ananth Saran Yalamarthy[2] & Arushi Arora[1]
Email IDs: {nanbhas, ananthy, arushi15}@stanford.edu
[1] Dept. of Electrical Engineering, Stanford University
[2] Dept. of Mechanical Engineering, Stanford University

1. Abstract:

Developing encoding models that estimate the functionality of a retina are immensely valuable, not only to get insight into complex visual pathways in the human brain but also to enable brain-machine-interfaces of the future such as a retinal prosthesis. We aimed to harness approaches from machine learning and neural networks to better model the natural encoding that takes place in the retina. In this project, we implement two models: a Linear Nonlinear Poisson (LNP) model and deep learning models (Convolutional Neural Networks or CNNs) to predict spike trains, given visual stimuli composed of white noise images. We find that the relatively simple baseline LNP model shows a best correlation of ~30% and that CNNs achieve comparable performance ~19%, which we believe is limited by the time we had (two weeks) to experiment with different architectures and to train the network. We believe experimenting with different CNN architectures, using combinations of CNNs and RNNs and using a different dataset may increase model performance.

2. Introduction:

The retina represents the first stage in our highly sophisticated visual processing unit. With only three layers of cells, its neurons transform the entire visual scene into spike patterns. At its simplest, the visual image detected by the *photoreceptors* in the last layer is sent to the *Retinal Ganglion Cells (RGCs)* in the front layer. There are ~20 different types of RGCs; we model two of the most prominent ones – ON cells (spike on detecting light) and OFF cells (spike on detecting darkness). These spike responses are what are sent to the brain via the optic nerve [1].

An accurate model of the retina is essential to understand the early computations performed by our visual system (Fig. 1). It is relatively less complicated to obtain characterization data from the retina as compared to any other processing unit because of its easy accessibility. Apart from just better understanding the visual system, a good encoding model of the retina will allow development of retinal prosthetic devices that could mimic the function of an intact retina in a blind eye by electrically stimulating neurons to transmit artificial signals to the brain.

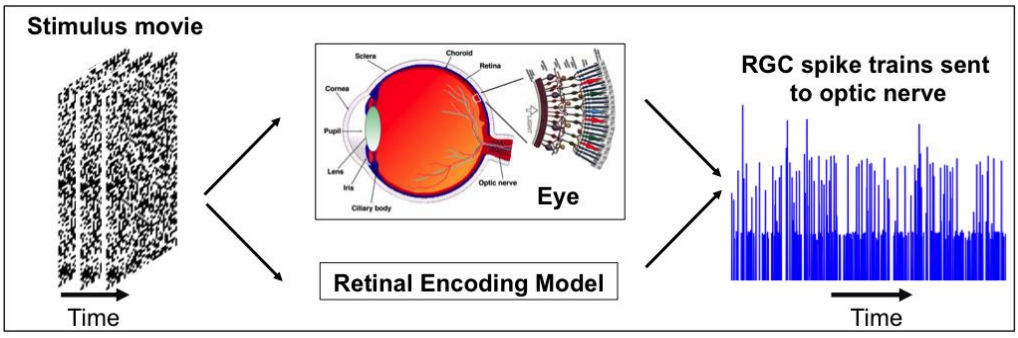

Figure 1. (left) The stimulus movie consists of a series of white noise images each of frame size 80 pixels x 40 pixels shown to the primate retina at a refresh rate of 120 Hz. The Retinal Ganglion Cells in the retina send out spike trains in response to this visual stimulus to the optic nerve. The aim is to create a model that can perform this encoding to eventually help create an artificial retina. (right) a single frame of input.

While in the past, the retina was thought to act just as a "pre-filter" for visual stimuli, recent studies have shown that the retina performs complex non-linear computations [2]. The presence of linear and non-linear computing units in the retina have led to the simplistic LNP model [3] and later the Generalised Linear model (GLM) based on maximum likelihood that considers the complex interactions of a retinal cell with its neighbours [1]. However, though some of these models could provide good approximations for retinal responses to Gaussian white noise, none were able to encode the responses for natural scenes [2]. We hope to establish a model that can accurately predict retinal responses to arbitrary stimuli. The complex and non-linear organization of the retina make deep learning and neural networks strong contenders for such a model.

3. Data:

We use the *unique* large-scale multi-electrode recordings (dozens of data sets with hundreds of cells each) that ***Professor E. J. Chichilnisky's*** lab in the Dept. of Neurosurgery at Stanford University have collected by stimulating RGCs in primate (macaque monkey) retina with movies of white noise. Primate retina is exceedingly similar to the human retina which makes this data invaluable in making predictions about the human visual system.

The input data shown to the retina consists of sixty 2-D training movies of a certain duration D (Fig. 1) refreshed at 120 Hz making each input movie of size 80 x 40 x D. For every RGC that is shown the input movie, there is a corresponding output vector containing its cell spike times. For the LNP model, we also use a Spike triggered average (STA) matrix for each, which refers to the RGC's receptive field as explained later. The test data consists of a distinct test movie stimulus shown sixty times to obtain sixty repeats of each RGC's response.

4. Problem Formulation:

For each identified RGC, given m training examples $\{x^{(i)}, y^{(i)}\}, i \in \{1,2,\dots,m\}$ where $x^{(i)}$ corresponds to a movie of frame size L x B and duration D seconds ($= N$ frames at refresh rate r) and $y^{(i)} = 1\{t_j$ observed a spike$\}$, where $j \in \{1,2,\dots, D/resolution\}$ represents the observed spike train in the RGC, the goal is to predict $\hat{y}$ for a test $x_{(test)}$ repeated n_r times such that $err(y_{\{test\}}, \hat{y})$ is minimum for a reasonable error metric "err".

5. LNP Model Implementation:

The LNP model was chosen as the baseline because though it is simpler than the GLM, it does consider first order nonlinearities which simpler models do not. It is extremely robust to fluctuations in responses, avoids adaptation to prolonged stimuli and is well suited to simultaneous measurements from multiple neurons [4]. This model is sufficiently interpretable as opposed to classical linear representations of neurons. The LNP model consists of a linear block, followed by a non-linear estimate of the firing rate, subjected to a Poisson process for spike generation (Fig. 2).

Figure 2. Block diagram of LNP model. Input stimulus first transforms into a generator signal using a linear function. The Nonlinear block transforms the generator signal into a spiking rate. The Poisson statistics transform that output into spikes.

The duration D is divided into bins of size Δt (usually chosen as $1/r$ where r is the refresh rate). The neural response f_t is defined as the number of spikes observed in time bin t. For a certain time bin t, vector s_t is defined to be a fixed fraction of the stimulus movie immediately preceding t, $D^*(= N^*$ in frames), chosen to exceed the memory of the neuron, i.e., the period over which a stimulus can affect the response. Thus, the key assumption here is that response f_t at time t depends only on s_t. A successful model should predict the average number of spikes per time bin observed after the presentation of any stimulus [4]. Here, s_t is a k-dimensional vector where $k = L$ x B x N^*.

5.1) The first step in the LNP model is the linear filter which is the spike triggered average (STA) vector, ω. The STA stimulus is defined as the average stimulus preceding a spike in the response i.e. the sum of the stimuli that preceded each spike divided by the total number of spikes [4]. Mathematically, $\omega = (\sum_D s_t f_t)/\sum_D f_t$, where, t is the time bin index over the duration D. Thus, the linear part of the model, ω, can be directly estimated from the given output spike train and is also a k-dimensional vector. This STA vector indicates how the neuron weights different spatial and temporal components of the stimulus. The spatial structure of ω describes the neuron's ***receptive field*** (Fig. 3a, Fig. 3d) that isolates a region of visual space that the neuron is active to. The temporal structure of ω represents the ***impulse response*** of the underlying linear summation (Fig. 3b, Fig. 3e). The memory of the cell can be assessed by examining the duration of the impulse response i.e. the time offset from the spike at which the STA converges to 0 [4]. The neural response to be modelled is the average spike count f in the time bin immediately following a stimulus s, i.e. the expected response to the stimulus $R(s) = \boldsymbol{E}[f \mid s] = \sum_f f\, P(f|s)$. The simplifying assumption made in the LNP model is that R is a static non-linear function of a real valued linear function of the stimulus. Thus, $R(s) = N(\omega \,.\, s)$ where . represents the scalar dot product between two vectors. Thus, stimulus entries s are weighted by the fixed STA vector ω and summed to give the *generator signal* $(g = \omega \,.\, s)$

5.2) The second step, i.e. the nonlinear filter, is given by the static nonlinear function N. This function is estimated by averaging f_t across time bins that give rise to the same generator signal g_t. Since the stimulus was Gaussian white noise, roughly the same generator signal will be observed at many times during the stimulus presentation. If H is a collection of time bins having roughly similar generator signals, i.e. if $g_t \approx \bar{g}, \forall\, t \in H$, then $\bar{f}$ is the average of all f_t where $t \in H$. This also implies that $N(\bar{g}) = \bar{f}$. Thus, by plotting $\bar{f}$ vs $\bar{g}$ for different values, the nonlinear function N can be estimated [4]. The nonlinearity is clearly visible from the plots (Fig. 3c, Fig. 3f) and we found the best sigmoidal function fit. This is used because it indicates that saturation in the firing rate occurs at the extremes of the generator signal. An optimization is run [5] to find the four parameters that lead to the best generalized sigmoidal fit to the data (solid line in Fig. 3 c,f).

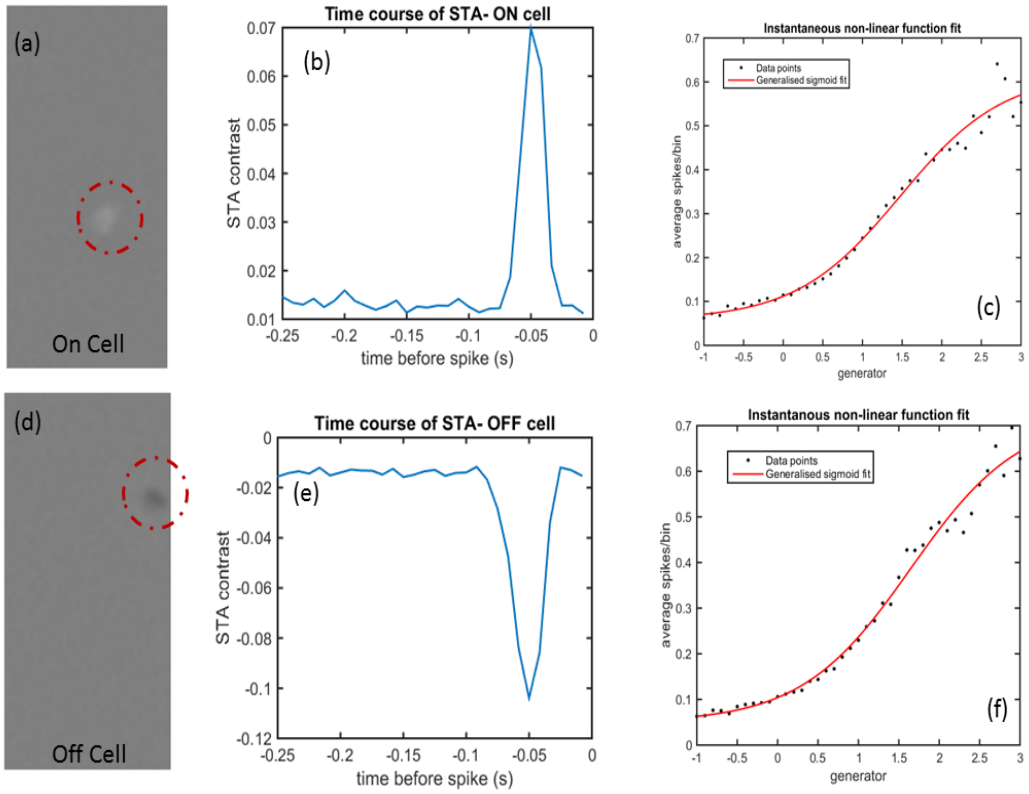

Figure 3. a) Spatial profile of STA for an ON-cell at a time instant (circled region shows centre of receptive field) b) The temporal profile of STA for an ON-cell. c) The scatter plot shows the actual spike rate as a function of generator signal. The solid line shows the optimized sigmoid fit for the generator signal. d-f) Same as (a-c) for an OFF-cell.

5.3) The third step is spike generation using the Poisson process that takes $R(s)$ as input and predicts a spike at time instant t_i by comparing $R(s)\Delta t$ to a random number between 0 and 1 from a uniform distribution [6].

5.4) Results: The model was tested on data consisting of a test movie having n_r repeats. The trained N and STA ω are used n_r times to produce n_r spike trains. The average spike train is calculated by taking mean of the n_r trials for both the predicted and actual spike trains and the correlation between the predicted and actual outputs ranges from 11.4% to ~30% for different neurons with the best one being 29.7% for an OFF cell and 30.7% for an ON cell (Fig. 4).

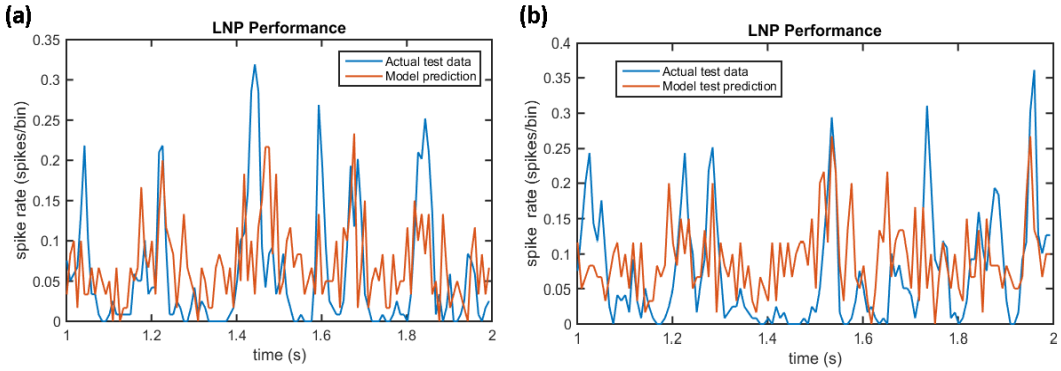

Figure 4. Comparison of predicted and actual spike trains for OFF cell (left) correlation coefficient = 29.7% and ON cell (right) – correlation coefficient = 30.7%

5.5) Discussion: The LNP model is simplistic and makes many fundamental and not completely accurate assumptions about the retina. To mention a few, it assumes a single non-linear function captures all the nonlinearities in the retina and that variability of the spike generation can be modeled with a simple Poisson process. Thus, it is not surprising that even though in terms of correlation coefficient it has a decent performance, the plots in Fig.4 follow the trend only to a certain extent.

6. Deep Learning Implementation:

Recent research on salamander retina has shown that **deep learning** methods can lead to high correlation coefficients ($> 70\%$) for spike train prediction [7] [8]. In view of this, we explored the use of deep learning models to eliminate the inherent assumptions of the LNP model (namely, the existence of separate linear and non-linear blocks, as well as a Poisson process for spike generation). In addition, from all the available modelling options, a neural network seems most likely to be able to capture the complexity of the retina.

6.1) Data & Network Architecture: The data used for training the CNN is the same as that used for the LNP model, namely Gaussian binary white noise movies (which can include all spatial and temporal frequencies as well as $2^{80\text{x}40}$ patterns in a given frame). However, the data was structured a little differently to make it compatible for training the neural network. Each training example $x^{(i)}$ for the CNN was an input movie, $L \times B \times D^*(= N^*\text{frames})$, where N^* was chosen to be 10 most recent frames (~0.083s) to optimize for network training time. Note that 10 frames were chosen using the intuition we developed from the STA (this can be seen in Fig. 3b, 3e, where ~0.1s of image time history determines spiking). To make the training outputs amenable for deep learning, the binary spike trains from the output vector spike train data, $y^{(i)}$ was convolved with a Gaussian filter with standard deviation equal to the bin size Δt, as can be seen in Fig. 5(b) to get a smoothed output $y_s^{(i)}$ [7]. These outputs account for the ***inherent variability*** in the spike times that are observed in the retina and are also representative of the test data that we will compare against, since those are averaged over n_r repeats as described in section 5 (which can be thought of as some kind of smoothing). In summary, for training we used 5 minutes of movie data stimuli corresponding to a set of $x^{(i)}, i \epsilon\{1,2,\dots,m'\}$ where $m' = 36{,}000$, trained against smoothed, normalized outputs that represent the spike train $y_s^{(i)}, i \epsilon\{1,2,\dots,m'\}$. Fig. 5(a) discusses an architecture of the Convolutional Neural Network (CNN) that we implemented. The input image is convolved with a series of filters and pooling layers (with ReLU activations), as seen in 5(a), and eventually a dense layer with a sigmoid non-linearity that represents a spiking probability in time bin t.

6.2) Platform, Loss & Evaluation Metrics: We used Keras (with Theano as backend) as our modelling platform which allows for rapid prototyping with neural networks. Since the outputs of our CNN are normalized spiking probabilities trained against $y_s^{(i)} \in [0,1]$, we used $L = \frac{1}{M}\sum_{i=0}^{M}[y_s^{(i)}\log\hat{y}_i + (1 - y_s^{(i)})\log(1 - \hat{y}_i)]$, which is the binary cross-entropy loss (or logloss). This performs much better than using the mean-squared error loss, which often gave us predictions with very low variance (all values around the mean of $y_s^{(i)}$, similar to that observed in [7]). To characterize the performance of our CNN, we used correlation coefficient as our performance metric in accordance with LNP results.

6.3) CNN Hyper-parameters: The most important parameters we tweaked were (a) the CNN architecture (b) the CNN's weight matrices, which were initialized via a uniform distribution with a scale parameter s_{CNN} and (c) learning rate (lr) of the CNN. We found the fitting behaviour to be very susceptible to s_{CNN}, with small values showing low variance in outputs, while large values showing vice-versa. We found scale factors of around ~0.1-0.3 to work well in practice. Higher learning rates (~$5 \times 10^{-4} - 10^{-3}$) were found to well in practice. We did not experiment with varying lr as a function of epoch.

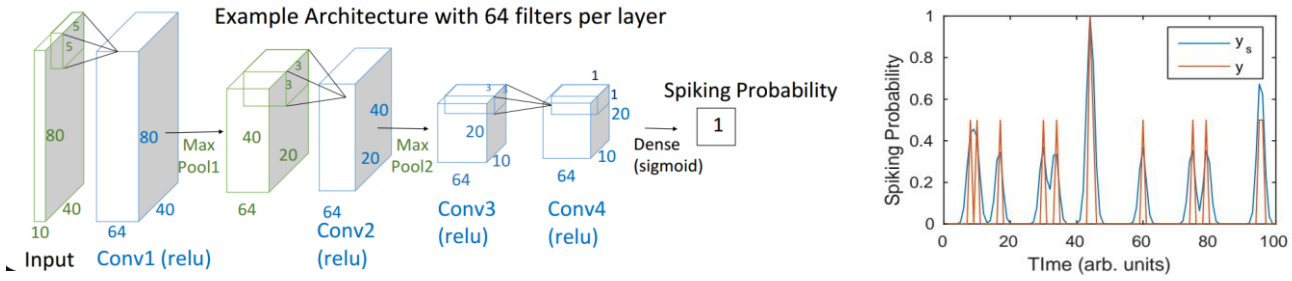

Figure 5(a). Example CNN architecture used. The 1st layer uses 64 (5×5) filters. In later layers, the depth of the CNN is preserved, while the spatial dimensions are reduced using pooling layers. A dense sigmoid layer at the end produces a single output that represents the spiking probability. (b) Image showing the raw spike trains y recorded, as well as the smoothed output y_s which is used for training the CNN weights.

6.4) Results: During our initial trials, our network would converge at outputs that predicted the mean of y_s with very low variance, which is characteristic of **underfit**. Thus, we focused our initial efforts on finding sets of hyper-parameters to **overfit** the training data and produce high training correlations. We chose this strategy knowing that

once we had **overfit** the training data, we could carefully tweak our CNN to best handle our training data. Figure 6 depicts this process.

Scale	#Neurons	Learning Rate (x1e-4)		
		10	5	1
0.2	64	0.98	0.993	0.44
	32	0.978	0.996	0.817
0.25	64	0.9795	0.991	0.478
	32	0.9845	0.935	0.355
0.3	64	0.989	0.947	0.368
	32	-0.012	-0.012	-0.012

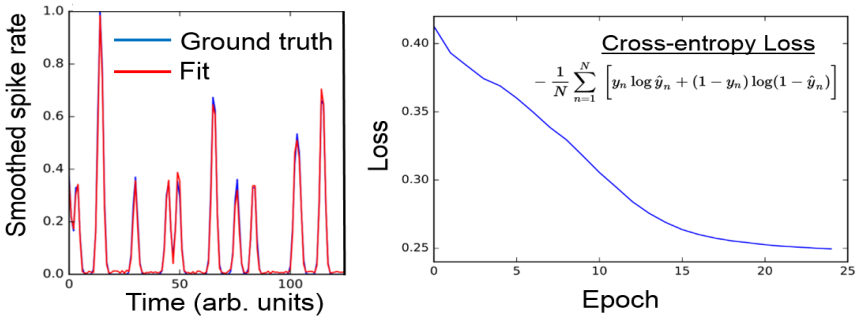

Figure 6(a). Table showing subset of experiments to determine CNN hyperparameters for overfit. The values correspond to correlation coefficients observed after 25 epochs. Best fits are observed at low s_{CNN} and high lr values. b) A magnified section of the fit on the training data for the best parameters in Table (a). Note that the CNN can overfit the data very well. c) Corresponding loss function over 25 epochs.

Once we obtained CNN parameters that were representative of overfit behaviour, we tested the CNN performance on the test data set (same one used for the LNP) and retrained the parameters on validation loss. Results for the same ON cell are shown in Fig. 7, with ρ as the correlation between y_s and $\hat{y}$. In the top panel, we present some results which support relatively small s_{CNN} values and high learning rates. The bottom panel shows our best result, with ρ_{test}~20% and a corresponding ρ_{train}~75%.

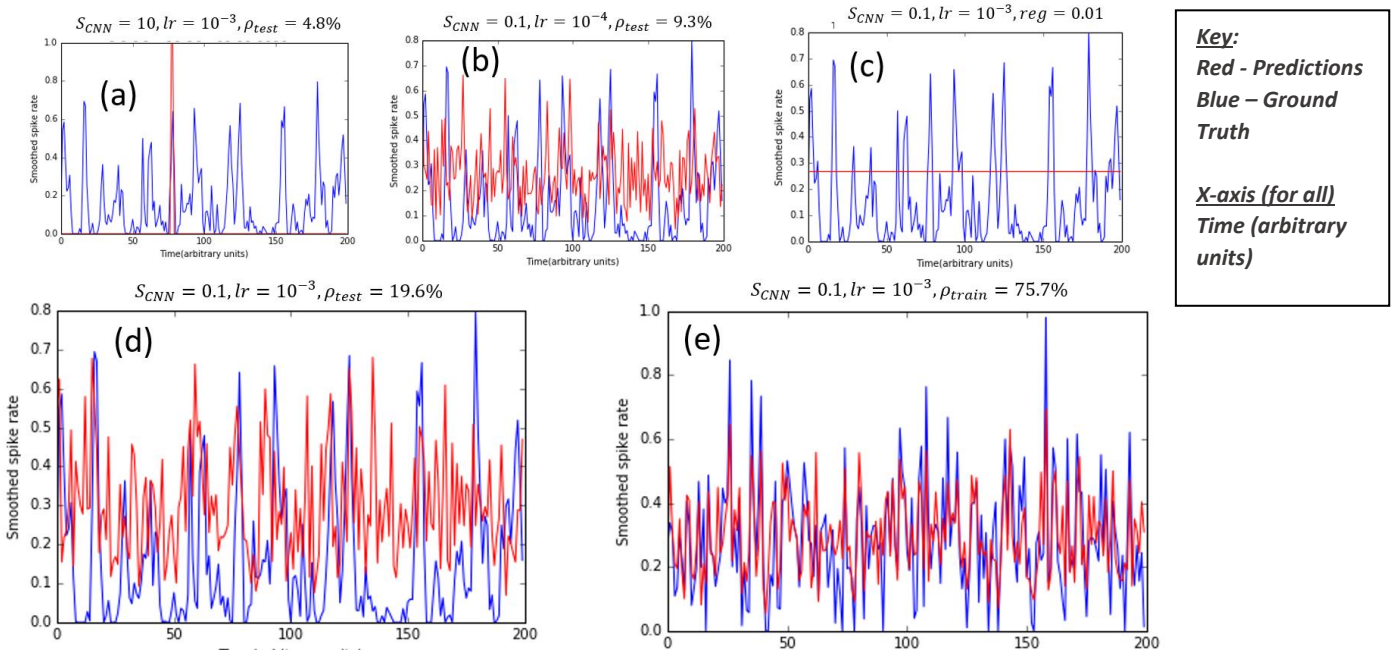

Figure 7. Upper panel: A depiction of results where poor test set performance was observed due to (a) large scale factors, (b) low learning rates and (c) regularization which consistently lead to prediction around the mean of the ground truth. Lower panel: (d) Best test set correlation performance (ρ_{test}~20%). (e) The corresponding fit for the training data where training correlation is about 75% indicating reasonable fit. We trained the network for ~75 epochs.

7. Conclusions and Future Work:

We have implemented two models to encode the retina – the traditional Linear NonLinear Poisson model as well as a deep learning neural network model. The simplistic LNP model shows better test results ($\rho = \sim 30\%$) when compared to the CNN as of now ($\rho = \sim 20\%$). As discussed in section 5.5, there are several limitations to LNP but given enough time, the CNN can be tuned to minimise test error considerably since CNNs have the necessary complexity to encode the retina. There are several architecture changes that can lead to better results such as adding 3D convolution, including an LSTM layer in the end, exploiting the advantages of RNNs given the time dependence of our data. Another thing to try is to encode different cells in the retina, literature shows [7] that different RGCs give different correlations when trained on the same model. It could be that we've used a neuron that has an intrinsic low correlation coefficient.

Acknowledgements: We'd like to thank Professor EJ Chichilnisky for sharing the dataset with us and Nora Brackhill, Vishakh Hegde and Ashwin Paranjape for immensely useful discussions. Thanks to Bo Wang (our TA) for valuable feedback.

References:

[1] Pillow JW, Shlens J, Paninski L, Sher A, Litke AM, Chichilnisky EJ, Simoncelli EP, "Spatiotemporal correlations and visual signalling in a complete neuronal population," *Nature*, pp. 454-995, 2008.

[2] Zrenner E, "Fighting blindness with microelectronics," *Sci Transl Med*, vol. 5, no. 210, p. 126.

[3] Field GD, Chichilnisky EJ, "Information processing in the primate retina: circuitry and coding," *Annu Rev Neurosci*, vol. 30, no. 1.f, 2007.

[4] Chichilnisky EJ, "A simple white noise analysis of neuronal light responses," *Comput. Neural Syst.*, vol. 12, pp. 199-213, 2001.

[5] https://www.mathworks.com/matlabcentral/fileexchange/42641-sigm-fit, "Sigmoid fit from Mathworks".

[6] Heeger D, "Poisson Model of Spike Generation," *http://www.cns.nyu.edu/~david/handouts/poisson.pdf*, 2000.

[7] McIntosh L, Maheswaranathan N, "A Deep Learning Model of the Retina," *http://cs231n.stanford.edu/reports/lanemc_final.pdf*.

[8] McIntosh L, Maheswaranathan N, Nayebi A, Ganguli S, Baccus SA, "Deep Learning Models of the Retinal Response to Natural Scenes," *Advances in Neural Information Processing Systems 29 (NIPS 2016) pre-proceedings*, 2016.