Automatic Calcium Imaging Response Detection

Leavitt, Lee
u0591788@utah.edu

Desabathina, Indumathi
u1265976@utah.edu

Cristiano, Deicy
u1228538@utah.edu

Introduction

Please go to this link for the git repository which accompanies this report.

Heterogeneous populations of cells provide massive potential for drug target discovery. Each cell has a multitude of receptors, ion channels, and G protein-coupled receptor (GPCR), each of which has various probabilities of success for drug targeting. The system to be studied in this project is the Dorsal Root Ganglion. This is the sensory organ of the body responsible for the transmission of all sensations to the central nervous system. Within this system, and depending on the experimental design, *each cell is an experiment*.

To understand which ion channels and GCPRs are on the surface of these cells we load them with a dye that binds to calcium we then image these cells every 1-2 seconds while different pharmacological agents are applied. When these channels become activated calcium will flow into the cell. Measuring calcium from these cells (~3000 per run) provides a functional approach to viewing and perturbing these potential drug targets. Once a target is identified this system can either be used as the drug screening platform or provide researchers with the target to further study and develop on higher throughput screens.

In this project we aim to develop tools to automatically identify these artificially provoked responses from the time series of images. In order to do so, we try a variety of architectures and methods within the class of deep learning networks to classify or diagnose patterns. Deep learning methods have gained a great deal of importance over the traditional machine learning techniques for pattern recognition.

First, we establish a baseline using Fully connected networks. Second, we experiment with a Convolutional Neural Network (CNN) input into these fully connected networks. Finally, we use recurrent neural networks (RNN's) specifically Long short term memory (LSTM) as our final approach, and we test two different inputs to this architecture.

Our results suggest that LSTM's are the best performing networks, and we are able to achieve 96% accuracy on each experiment described below.

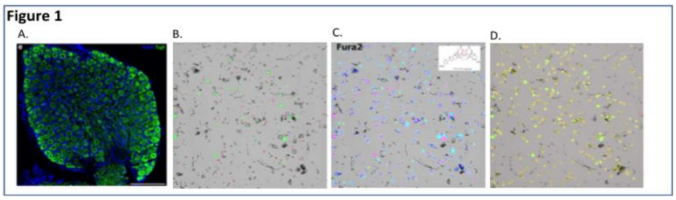
Related Work

Previous work for automatic identification of responses used Support Vector Machines (SVM); Lasso, Ridge, and Elastic Net regression with mathematically derived statistics performed an unsatisfactory success rate. Although the total accuracy rate is high for all detected responses (~96%), the dataset was significantly skewed, where responses only made up ~2-5% of all responses. When observing the success rate of responding cells the accuracy dropped to 60%. Thus, a more sensitive approach is needed to succeed at this problem.

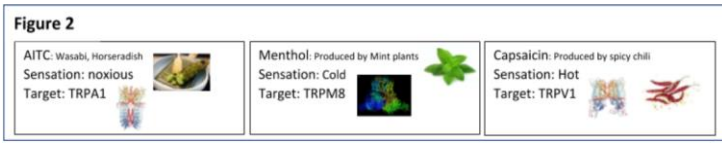
For this approach we will use raw response/traces, instead of calculating statistics that represent the response. This means the signals will be used as input to the deep learning network architecture for classification. This allows for the deep networks to create the statistics for analyzing each cell's pattern of response without calculating clever statistics to include all the features in the classification. Different pattern recognition techniques such as CNN have shown good results after limited pre-processing of the signal. Including CNN to our problem tackle down the low proportion of the cells that respond, for that reason it is expected a better accuracy of CNN.

These signals are also analyzed as a time-series data which are best served by Recurrent Neural Network. This type of neural network is well-suited to time series data allow to consider the dependence over time and take into account the dynamic of the sequence. We will use an RNN to consider this dependence on the time to predict the patter that cells follow when a drug is applied.

Analyzing Cell Responses to Drugs

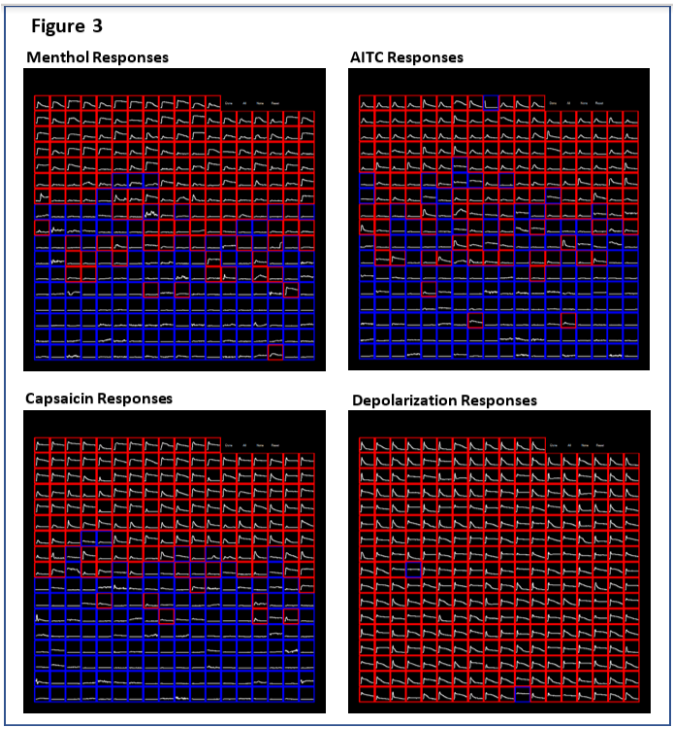


Cells from the dorsal ganglion are gently dissociated and plated in a minimal media. **Figure 1A** shows an intact dorsal root ganglion. Each green dot is an individual neuron. **Figure 1B** shows these cells after dissociation. The red and green colors on the surface of these cells identify two different types of pain sensing neurons. Cells recover for ~16-24 hours, the following day, cells are loaded with Fura2 (**Figure 1C**). This is a dye which binds to calcium. This is the signal for activation of receptors and ion channels on the surface of these cells. A region of interest is then identified using CellProfiler. This is shown in **Figure 1D**. These cells are then imaged every two seconds while different plant derived natural products are applied to these neurons. Follow this link to watch a video of how this data is collected.



During the video three different plant derived natural products are added to the cells. Each molecule activates a different ion channels on the surface of the cells. As can be observed each cell has a different composition of ion channels present on the surface of the cell. Cell 1 senses cold since menthol activate the cold sensing ion channel TRPM8. Cell 3 senses heat due to capsaicin activating the heat sensing ion channel TRPV1. For a summary of these ion channels see **Figure 2**.

The video shows a small example of three cells imaged during these experiments. In practice the field of view is much larger since we use a 4x objective. This increased view of our cells allows us to collect data from ~3000 cells at a time. Each of these 3000 cells are scored by the researchers as responded or did not respond. **Figure 3** shows the tool researchers use to correct the labels, each applications scoring is corrected as either '*yes the cell responded to this compound'* (the trace with a red box surrounding it represents this). Every trace within a blue box represents *'no the cell did not respond to this compound'*.



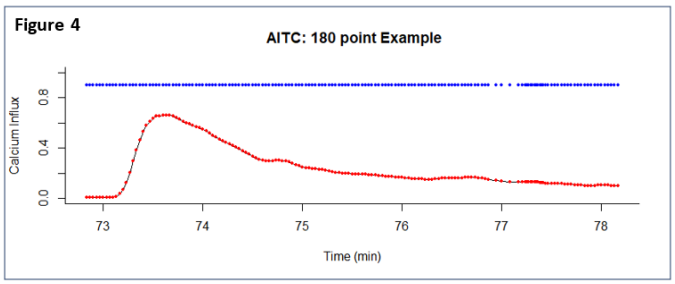
The researchers cell type definition and advanced analysis depend on the correct scoring of these responses and this process take ~2 hours. We have been collecting data for the past few years and have built up a large and clean dataset. Due to the hard work of these researchers our data set has a clean label space.

Dataset

In this section we describe the dataset we use and some of its properties.

We will use the data from the laboratory of Dr. Baldomero Olivera. Each experiments have ~3000 cells, experiments last between one and two hours and up to thirty different challenges can be applied during these experiments. Our dataset is comprised of ten experiments with an approximate total of ~15000 samples. Table 1 shows a break down of these responses.

Figure 4 shows a representation of a single trace from a single cell. This will be the data fed into the neural networks. Prior to this experiment the traces are smoothed and fit to a zero baseline. The traces are also minimally scaled to the maximal response the neuron receives. The red dots in Figure 4 represent the response at each time point. The x-axis corresponds to time in minutes. The y-axis represents the influx of calcium into a single cell. The data is scaled between zero and one. A value of one is determined from a maximal depolarizing pulse at the end of the experiment. This should tell us what is the maximum at which this cell can respond.



The blue circles represent the timing of the image capture. One thing to note is the timing of the image capture separating at 77 minutes. The represents the inconsistency that can arise during image collection.

We classify the cells during in each experiment with four pharmacological identifiers, AITC, Capsaicin, K40, and Menthol. Table 1. shows the distribution of the responses. We see that there is a lower cell responses to the Menthol pharmacologic.

Table 1: Distribution of cell responses

	Does not Responds	Responds
AITC	12120 79.6%	3105 20.4%
Capsaicin	11441 75.1%	3784 24.9%
K40	7220 47.4%	8005 52.6%
Menthol	13417 88.1%	1808 11.9%

Methods

To improve performance all models were run on a GPU. This significantly improve the speed of training.

Approach 1:

To gain a baseline of success we designed a model with a 1 dimensional convolution layer is 32 filter, size 3 kernel, come padding, and a relu activation function. This was sent into a pooling layer of size 2.

This was then flattened and sent into a fdully connected layer with 5 neurons with relu activation,which was finally sent into a dense layer with a single output with a sigmoid activation function. This was optimized using ADAM, and the binary cross entropy loss was used.

Approach 2:

Our second approach considers the sequential data since every point in the cell response is dependent on previous positions. RNNs process a time series step-by-step, maintaining an internal state summarizing the information they have seen so far. A LSTM, is a type of RNN explicitly designed to avoid the long-term dependency problem. Unlike regular RNN cells, LSTM cells maintain 4 different 'gates' that control the flow of the gradients. All recurrent neural networks have the form of a chain of repeating modules of neural network. LSTMs also have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

For the first approach the entire raw trace is fed into the LSTM. So we have a total of 120 LSTM modules. We set the output of this to be 100 which was fed into dense network with a sigmoid activation. The model was optimized with ADAM, using a binary cross entropy loss.

Approach 3:

A modification of the previous attempt with a convolutional layer before the LSTM cell was added in order to improve the performance. This approach combines convolutional neural networks an LSTM. We add a one-dimensional CNN and max pooling layers which feed the consolidated features to the LSTM.

Approach 4:

Instead of using the raw trace as an input a series of statistics were calculated over a time interval. So the data input into the LSTM was single dimensional. For example [A, B, C] were;

A: Number of samples (~15,000)

B: Number of timesteps (120)

C: Number of features (1)

For this experiment we decided to reduce the timesteps but increase the features, so

A: Number of samples (~15,000)

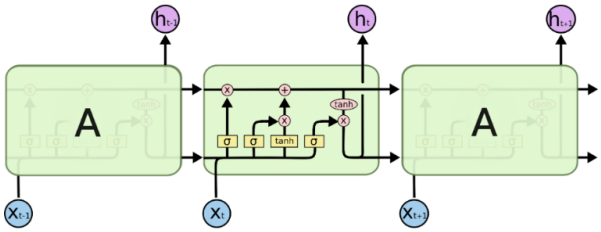
B: Number of timesteps (10)

C: Features (4):

a. Mean: The mean was calculated for every 10 steps.
b. Standard Deviation: The standard deviation was calculated for every 10 steps.
c. Standard Error of the Mean: The standard error of the mean was calculated for every 10 steps.

d. Sum of the Gradients: The gradient was calculated for each timestep, this was then summed over 10 steps.

From there this data was fed into a LSTM with an output of 10 which was fed into a dense layer with a sigmoid activation. The model optimized using ADAM, and the loss was computed with binary cross entropy.

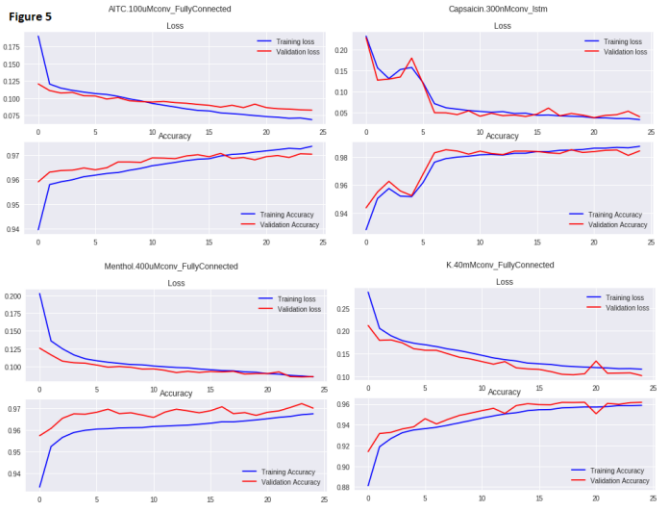


The repeating module in an LSTM contains four interacting layers.

Results

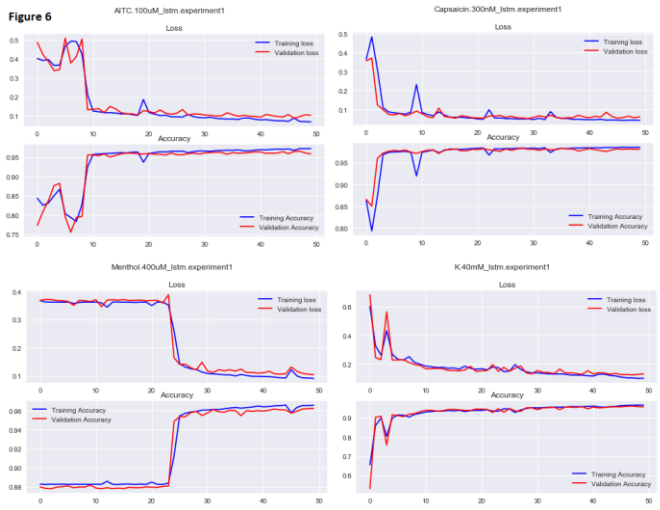
Approach 1:
This approach showed a surprisingly good success rate, and with more epoch the model looks to continue to learn. Although overfitting is apparent in the AITC

Figure 5



Approach 2:

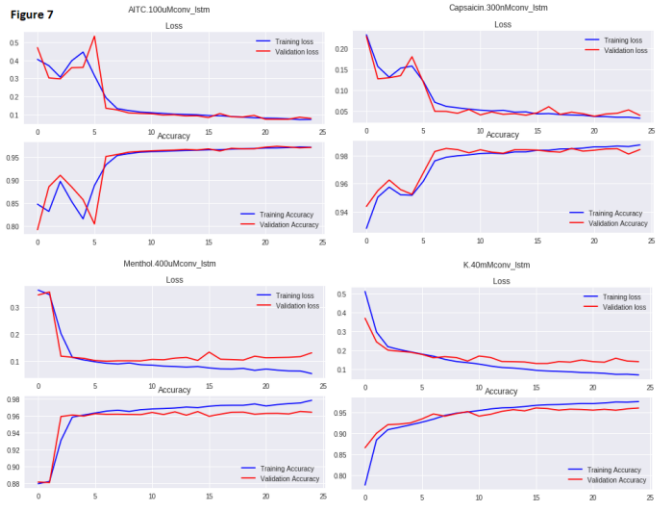
The results of this approach are shown in **figure 6.** The loss function from training (blue) and validation (red) at each epoch for the four pharmacological identifiers in the order AITC, Capsaicin, K40, and Menthol. These graphs suggest that there are differences in the convergence according to the pharmacologic substances. While convergence is achieved in 5 epochs for K40, it takes 25 for menthol, which may be explained by the low percentage of cells that respond, or the input feature not being informative enough.



Approach 3:

Figure 7. present the results for attempt 3 which include convolutional neural networks fed into LSTM cells. Spatial features are learned and a LSTM layer is later applied. Although the training loss is good, this model is not able to generalize well as we see in the loss curves. Curves for training loss and the training and validation loss separate at the end. That could be explained by an overfit problem.

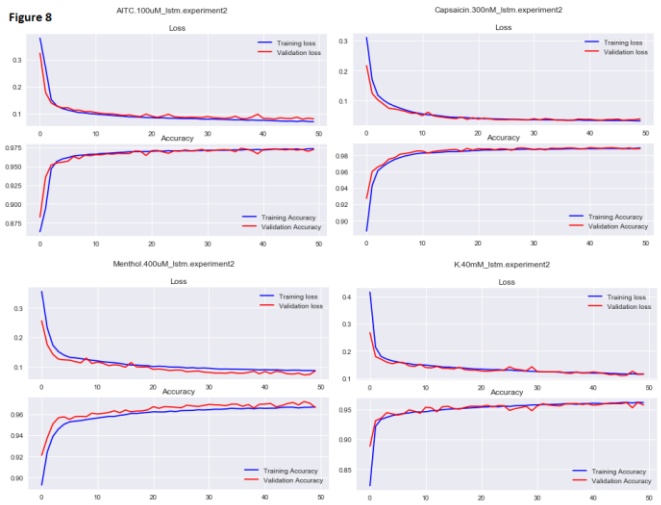
Table 2 shows the misclassified responses for this approach. It presents the number and percentage of cells that are misclassified within do not respond, respond, and overall. We get a low percentage of misclassification overall, which ranges from 1.4%. to 2.77%.



Attempt 4:

The results of this approach are represented in **figure 8.** Compared to attempt 3, learning was much smoother, and the menthol response learned at the same rate as the other responses. Interestingly K40mM shows an increasing

slope at the 50th epoch suggesting that we could get a better model if we increased the epoch.



The next part of this project was to visualize what he networks were getting right and what they were getting wrong. The git repository has all visualizations shows the mismatches of our four identifiers. One question researchers should ask is should we continue to improve the model. For example, is it possible to get better than 97%? To view this we looked at attempt 4. **Figure 9** shows the visualization of the responses that were labeled as 0/non-responders, but the model labeled it as a 1/responder. What is fascinating with this is that every single misclassified response is actually correctly classified.

Figure 9: Menthol 0's classified as 1's

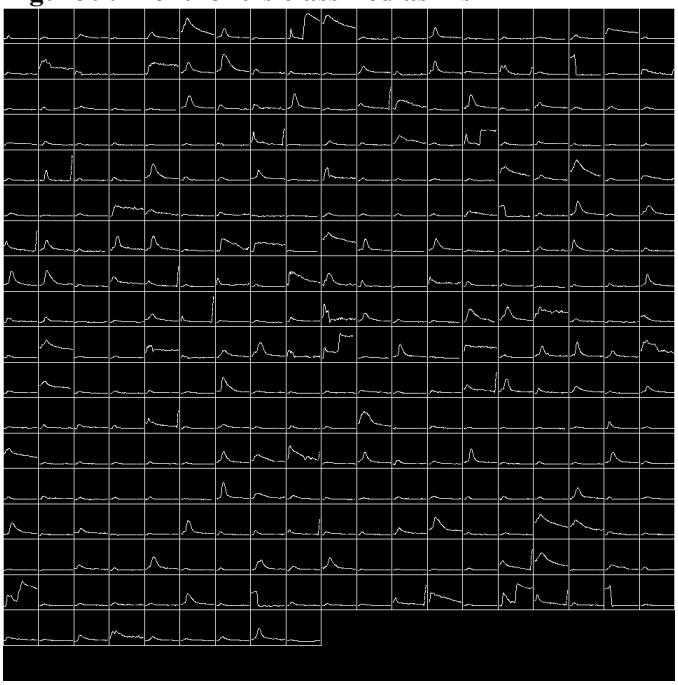


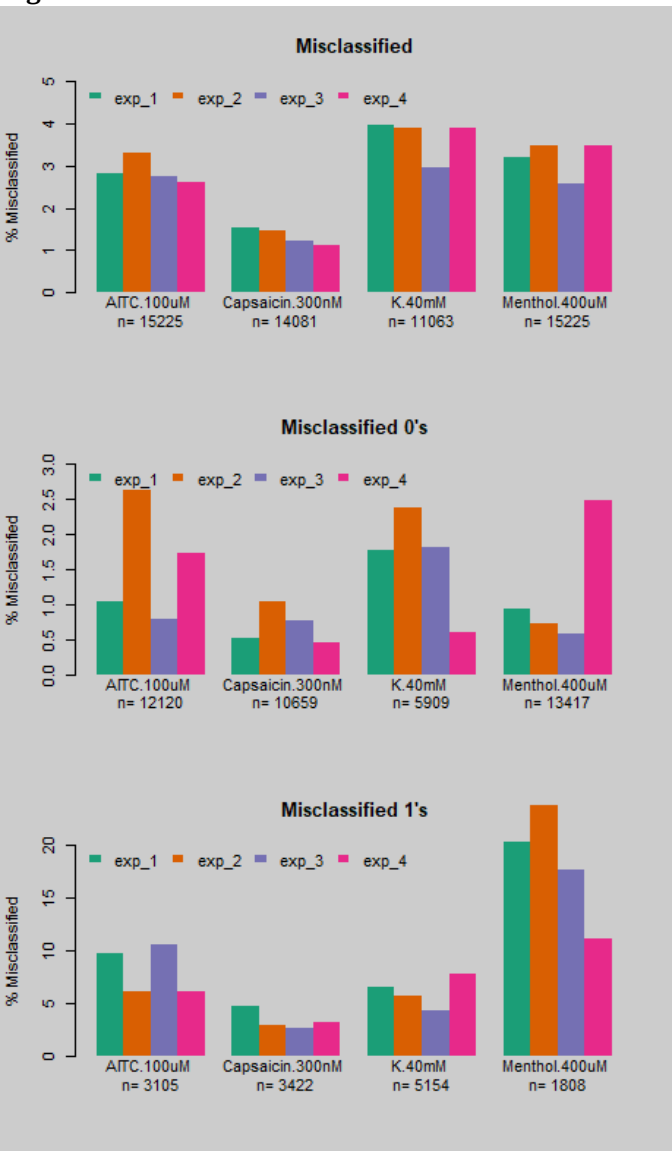
Figure 10 shows menthols responses where 1's are scored as 0's. This also shows very good performance, where only about 15 actually show correct initial classification, the other 190 could either be successfully classified as 0's from the beginning, or are too close to call.

Figure 10: Menthol 1's classified as 0's

Experiment Comparisons

Figure 11 shows the breakdown of the misclassified responses. Capsaicin response showed the best rate of classification with all models, and all models performed very well. But, breaking down the success based on the classification types we can see some models outperform others. No single model was the best at any one response. For example architecture 4 performed the best on the K.40mM response, whereas it significantly underperformed on the menthol response. The misclassified 1's were the place where most scrutiny should be focused. As discussed before this data set was heavily biased to containing significantly more 0 classes than 1 classes. So, the true success should be focused here. Architecture 4, showed very good performance on this series, and the capsaicin response was easily learned. Menthol on the other hand severely struggled to classify with a 20% failure rate. This could have to do with the lowest number of samples (n=1808) compared to the others, but architecture 4 was able to succeed better than the other attempts. Additionally, as discussed model 4 also seemed to classify these responses better than the human as shown in figure 9 and 10.

Figure 11



Conclusions

This work uses some techniques within deep learning networks to automatically identify responses from the time series of images. We try different architectures and methods which perform better both responding cells and no responding cells.

All architectures performed as very good success rates, but architecture four seemed to outperform all the others. Also, this architecture was able to train much faster than the others.

The next steps for this project are to increase the sample size for the 1 class in the menthol response types. Additionally incorporating this architecture into data pipeline we will be able to save a significant amount of

time, and mayu be able to look at a large legacy dataset to tackle a big data opportunity.

Our current data pipeline is developed in the R language. We have had success in the past incorporating python into this workflow by utilizing the reticulate package. Fortunately this will ease our implementation of this solution for the entire team smoothly.

References

Brownlee, J. (2019). Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras. Machine Learning.https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/.

Colah.github.io. (2019). Understanding LSTM Networks -- colah's blog. [online] at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

H, C. (2019). Convolutional neural networks for event-related potential detection: impact of the architecture. - PubMed - NCBI. https://www.ncbi.nlm.nih.gov/pubmed/29060295.