C. *elegans* Visual Implementation of Computational Heuristic Experiments

Rachel St Clair, Ken Dawson-Scully, Elan Barenholtz, William Hahn

Florida Atlantic University, Center for Complex Systems and Brain Sciences
rstclair2012@fau.edu

Abstract

Deep computer vision was used to assess C.elegans seizure activity from recorded data in efforts to predict average time to recovery per seizing assay. Video was cataloged from various inductions of electric shock to C.elegans in aqueous solutions. Methods iterated over several types of pre-trained convolutional architectures for fine-tuning to develop the proposed protocol to detect number of C.elegans seizing per frame during the assay. Furthermore, several hyper-parameters were tested to ensure the networks high accuracy, while post-processing algorithms were then implemented to calculate the average time seizing per video assay. The network, VGG proved to be most accurate (97% validation accuracy) in detecting seizing events. Overall, the developed software compares to human determined seizing averages by a decrease in (¡insert%) variability of pair-wise differences. This report will discuss the data used, model architecture, hyper-parameter tuning, post-processing algorithms, and future directions.

Keywords: C. elegans, Computer Vision, Seizure Detection, Transfer Learning.

I. Introduction

C. elegans have recently accrued interest in behavioral sciences for their ability to model multiple aspects of human neurology. Most recently, researchers have developed high-throughput screening assays to determine induced influences on seizure duration [1]. Work of this kind generates high amounts of data to be analyzed by hand before producing useful results for interpretation. Such assays greatly lend to developing pharmacological tools and understanding the basis of epileptic activity. The need to streamline the data-to-result pipeline could make more efficient use of scientist, resources, and time in not only this application, but many others.

I. Data Processing

Seizure assays were conducted by recording up to six *C. elegans* in small tubes filled with saline fluid. Each tube was connected to an electric node that would deliver a small electric shock to the tube near the beginning of each recording, distinguishable by bubble entry into the tube. At that time, worms were hand-labeled to be seizing and returned to normal locomotion when sine-like wave formations of swimming were observed. The onset of shock and offset of seizing per worm was recorded to calculate the total average time seizing for all worms in each video. While most frames were of 224 pixel height and 256 pixel width, all frames were re-sized to 224 by 224, all three color channels were kept, and no padding was added (unless specified within the model type class) (Fig. 1). This collection of hand-labeled data across 68 different videos was then split up into into individual frames tagged with the number of worms seizing in such frame using python and openCV.

II. Methods

Multiple deep learning model architectures were employed to analyze the video data: ResNet-18, Alexnet, VGG-11, Squeezenet, Densenet, and Inception. The goal to learn features relevant to classifying number of seizing worms was shared across model architectures. However, each model was compared to itself for two different types of data sampling. Finally, fine-tuned model VGG, was chosen for further hyper-parameter testing and to evaluate testing results, due to its high-accuracy score and simplicity(compared to some of the other pre-trained models).

I. Data Sampling

From the 68 videos, seven categories of seizing were created for the PyTorch ImageFolder dataset class (0,1,2,3,4,5, or 6) to represent possible number of worms seizing in each frame. Table one best describes the unequal dis-

tribution of frames per category, with zero worms seizing having an overwhelming amount of frames (775,010) compared to six worms seizing at 900 frames. To avoid over-fitting and under-representation of any one category, two methods were compared for all model types: over-sampling and under-sampling. Image augmentation was implemented in the form of image re-sizing (needed for inception's input layer of size 229), random horizontal flips, and normalization for training; re-sizing and normalization for validation was used throughout all model types and sampling methods.

For the over-sampled trials, 5,783 (the number of data in five worms seizing category) random samples were taken from category zero to four worms seizing, with all of the samples taken from five worms. Since six seizing category only had 900 samples, a random 20% of these were removed from the training set for validation and the remaining were randomly duplicated until 5,783 samples were accumulated. Finally, a random 20% from each other category was removed from training to the validation set. This method ensured each class would be represented fairly while still generating a larger amount of data for fine-tuning, all the while ensuring no instances in the validation set would be found in the training set.

As for the under-sampled trials, 900 (the total number of frames in six seizing category) random samples were taken from each category zero - five, while all samples were taken from six worms seizing. Again, a random sampling of 20% from each category was moved to the validation set and the 80% remaining were kept for training. The advantage of this technique is the absence of multiple same-samples that would otherwise be produced in the over-sampling technique. However, the dataset was then limited to only 720 frames for each category in the training set.

II. Model Architectures

Each model type was imported after being pre-trained on ImageNet database and executed using in PyTorch using a hard-coded loop that iterated through each model type with the same hyper-parameters [2, 3]. Since each model type is a vast architecture with many layers, parameters, and adjunct algorithms, only VGG-11 will be discussed in detail here. The network is comprised of individual 3x3 convolutional kernal layers stacked with intermittent max-pooling (2x2 kernal size with stride of 2), 11 layers deep, totalling 133 million parameters [4]. The final three fully connected layers for the output are then reshaped in this model to accompany the feature class size of seven. For all model types, SGD was used as the optimizer, while cross-entropy loss was used for the criterion.

III. Hyper-parameter Optimization

Initially, the iterative model was trained and validated with a batch size of 16 for 25 epochs for both under and over-sampling cases. These hyper-parameters were arbitrarily chosen as the base comparison between sampling techniques. To further test VGG-11, the best performing model, epochs were increased to 50 since the accuracy curve (generated in Visdom during training/validation) was only just beginning to reach its asymptote at 25 epochs with oversampling. Furthermore, for another trial the batch size was decreased to 16 to determine its influence on validation accuracy, further increase of batch size would of been preferable but computing power capabilities maxed out at at batch size of 32. Final trials were executed with batch size of eight for 25 and 100 epochs. The final model was saved after fine-tuning for 100 epochs with a batch-size of 16.

IV. Post-Processing

The model was developed to infer the number of worms seizing per frame. To make this approach fulfill the desired goal of determining average seizing time for all worms per video, post-processing of the model outputs are necessary. While the fine-tuned model may not appear as end-to-end (input to solution) only a few calculations are needed. Once the trained model is saved, meaning all weights for all nodes and layers are stored, un-seen (during training) test videos can be fed into the model in evaluation mode, frame by frame. The model will take each frame as input and calculate the number of worms seizing without updating the parameters in the network. This information is then plotted and smoothed using median filter, kernel size 91, from python based library, Scipy [5]. The kernel sized was determined through trial and error by examining the output predictions and smoothed predictions until most video's predictions appeared consistent with the target results. Figure two shows a comparison between the raw predictions of the model, filtered predictions, and human-labeled seizing events. As mentioned before, the frame rate of the video is 29 frames per second, there is an abundance of data to be analyzed for one second decisions on starting and stopping of seizing in the worms behavior. The median filter looks before and after the current position in the output trajectory and replaces mistakes (i.e. noise) in the model's predictions. The max of the output plot gives the total number of worms seizing; while the sum of the filtered output plot gives the total number of frames seizing occurred for all the worms combined. Thus, the average seizing is calculated by the following equation.

$$\frac{total\ frames\ seizing\ in\ filtered\ predictions\ for\ all\ frames}{\frac{total\ frames\ predicted}{total\ frames}} * \frac{fps}{number\ of\ worms\ seizing}$$

To compute the error of the models predicted average time spent seizing per video to the human determined average time, percent error was computed using

$$((human\ avg. - model\ avg.)/human\ avg.) * 100$$

. However, when the average time spent seizing for such video was zero, the percent error was computed simply by

$$human\ (avg. - model\ avg.) * 100.$$

This procedure on average took

$$insertavgtimetoprocess$$

for videos ranging from

$$insertrangeoflengthofvideos$$

.

III. Results

I. Training

VGG received the highest validation accuracy for over-sampling with 25 epochs and a batch size of 16 out of the tested model architectures (Fig.3). Table two reflects the best validation accuracy scores for each model across the sampling conditions. Under-sampling generally preformed worse than oversampling and increasing epochs rather than the batch dimension improved accuracy more effectively for the limited trials preformed here. For all conditions, best validation accuracy was used as the comparison. This metric represents the best performance of the model (by type) on predicting the target label (number of worms seizing) for each single frame in the overall validation dataset. Finally, table three shows the comparison metric for each hyper-parameter condition used in further tuning VGGnet. Therefor, the best network and hyper-parameter combination tested here was VGG with batch size of 8 and oversampling for 100 epochs.

II. Post-Processing

Further testing was performed on a separate set of videos than used in the training and validation data pool to asses further accuracy of the model. After post-processing procedures were run on such videos, metrics could be determined for comparing the human decisions with model decisions. Figure four demonstrates, most notably, the consistency between true positives and true negatives for each seizing class. Once an average time per seizing could be obtained, we could compare the error between the human average seizing time to the model's inference. The average error across all 40 videos was

$$insertavgerror$$

. Since the model is taking in a relatively high frame rate, it has the potential to have a higher resolution of accuracy. We've found the model consistently over-estimates the seizing time when compared to human-labeled, as seen in figure five. To compare it's variability to human counterparts, we calculated a small inter-rater-reliability report. The pair-wise error of rater averages was 4.90% while the range was more dramatic, 0.56% - 26.23%. The model had an average

$$insertavgofmodelerrors$$

of and range of

$$insertrangeofmodelerrors$$

. These metrics allow a delicate forgiveness of model error, as would naturally be found in human analysis. Figure 6 shows the plots of human and model assessments.

IV. Discussion

As this work has demonstrated, human-like accuracy can be provided from autonomous data processing programs, such as CeVICHE. Deep computer vision architectures, as proposed here, have potential to rapidly advance the way we conduct research by minimizing the monotonous data analysis process and creating less variable than human processed thresholds of event detection. By automating as much possible data-analysis, we can next begin scrutinizing the decision-making of the network to employ a wider range of analytic techniques (ig. changes in network saliency for different assay treatments). Applying deep learning automation to this high-throughput *C. elegans* assay is allowing researchers to make quicker knowledge discoveries in this domain, leading to advances in seizure control and alleviation.

I. Future Works

While accuracy of this model was high, the need for scrutinous efficacy in real-world implementation is needed. To further improve the accuracy and ensure redundancy of such metrics, further techniques can be employed to improve the model. Additionally, a more end-to-end approach can be implemented to deploy on edge-devices. These modifications would suggest training more temporal data instances and fine-tuning the data distribution balance. Ultimately, final directions of this work will seek to analyze live video stream of seizure assays.

Acknowledgment

We would like to thank the Machine Perception and Cognitive Robotics Laboratory for their use of Nvidia graphics processing units, Naven Parthasarathy, Michael Teti and Micha Klopukh for their insightful help in data collection and debugging the model. This research was supported by

insertgrantsuffshere

.

References

[1] Monica G Risley, Stephanie P Kelly, and Ken Dawson-Scully. Electroshock induced seizures in adult c. elegans. *PLOS ONE*, 2016.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[3] Nathan Inkawhich. Finetuning Torchvision Models.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[5] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed ¡today¿].

V. Appendix

Worms Seizing	Labeled Frames
zero	757,010
one	157,108
two	68,574
three	27,677
four	12,689
five	5,783
six	900

Table 1: *Number of frames in each dataset category: number of worms seizing per all 68 videos.*

Model Type	Under-sampled Best Accuracy	Oversampled Best Accuracy
VGG-11	0.8881	0.9527
Densenet	0.8944	0.9469
Resnet-18	0.8897	0.9410
Inception	0.9079	0.9341
Alexnet	0.8294	0.8945
Squeezenet	0.8317	0.8990

Table 2: *Best validation accuracy post-training for each model type for over and under-sampled datasets for 25 epochs with batch size of 16.*

Epochs	Batch-size	Best Validation Accuracy
25	8	0.8881
25	16	0.9527
50	8	0.9579
100	8	0.9719
100	16	0.9707

Table 3: *VGG hyper-parameter optimization best validation accuracy results with oversampling.*

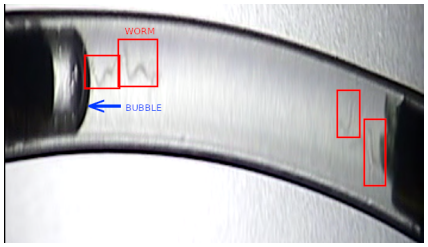

Figure 1: *Raw frame of seizing assay video data marked to show worms and bubble for human interpretation (refer to table one for actual target label categories).*

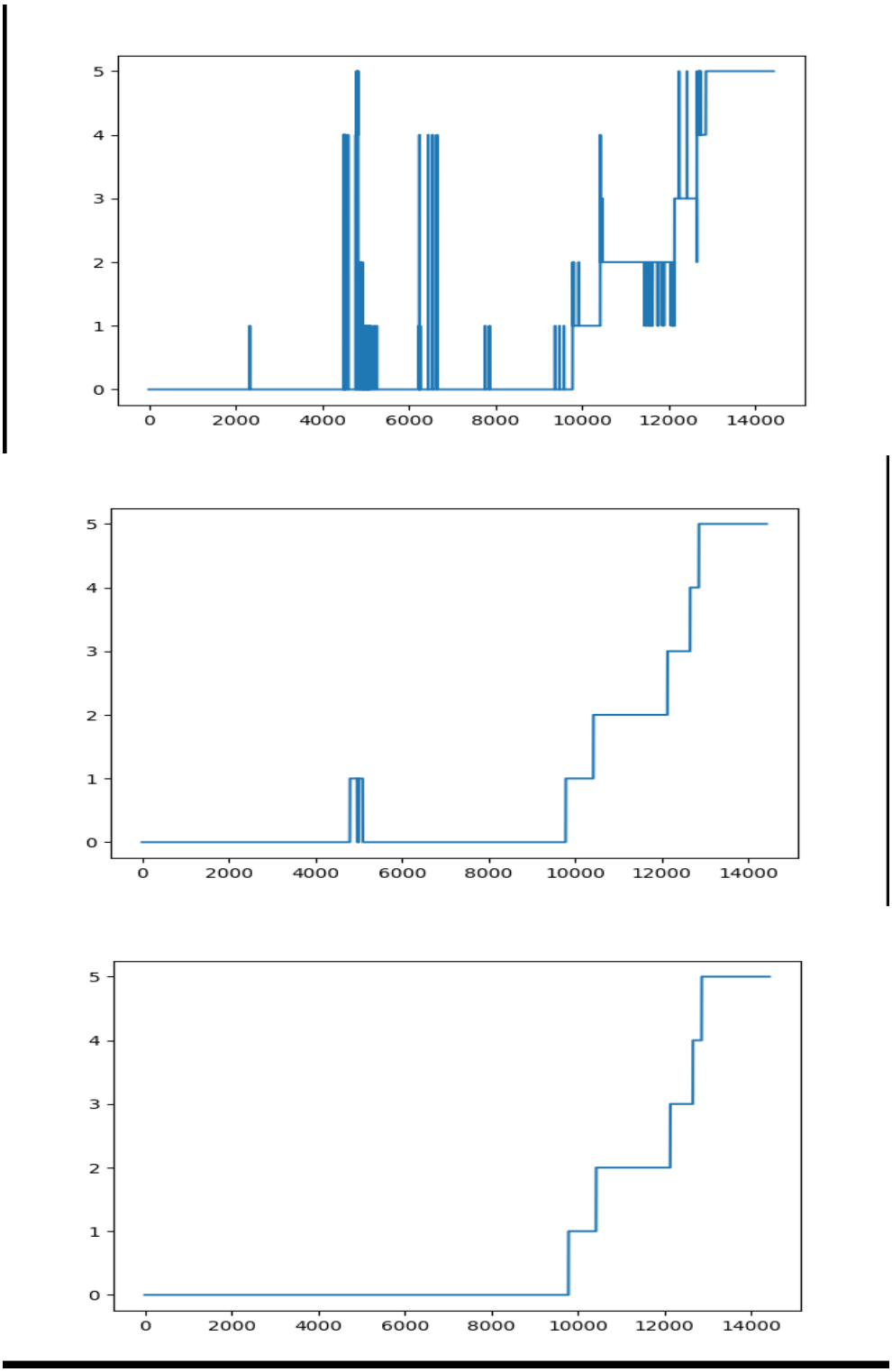

Figure 2: *Predictions of worms seizing per frame from raw model output (top), filtered model output (middle), and human-labeled decisions (bottom), (*this video was used in the training class).*

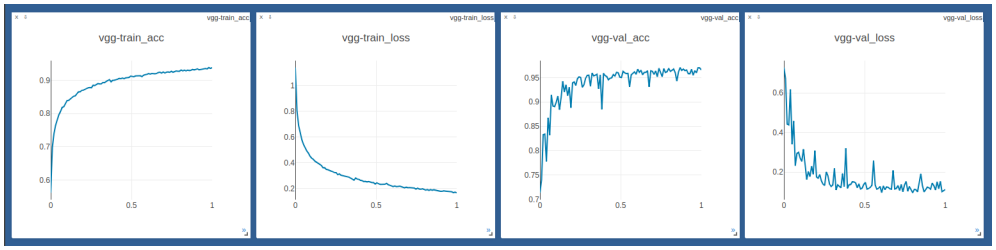

Figure 3: *Visdom plots of training and validation accuracy/losses for VGG network of batch-size 8 for 100 epochs.*

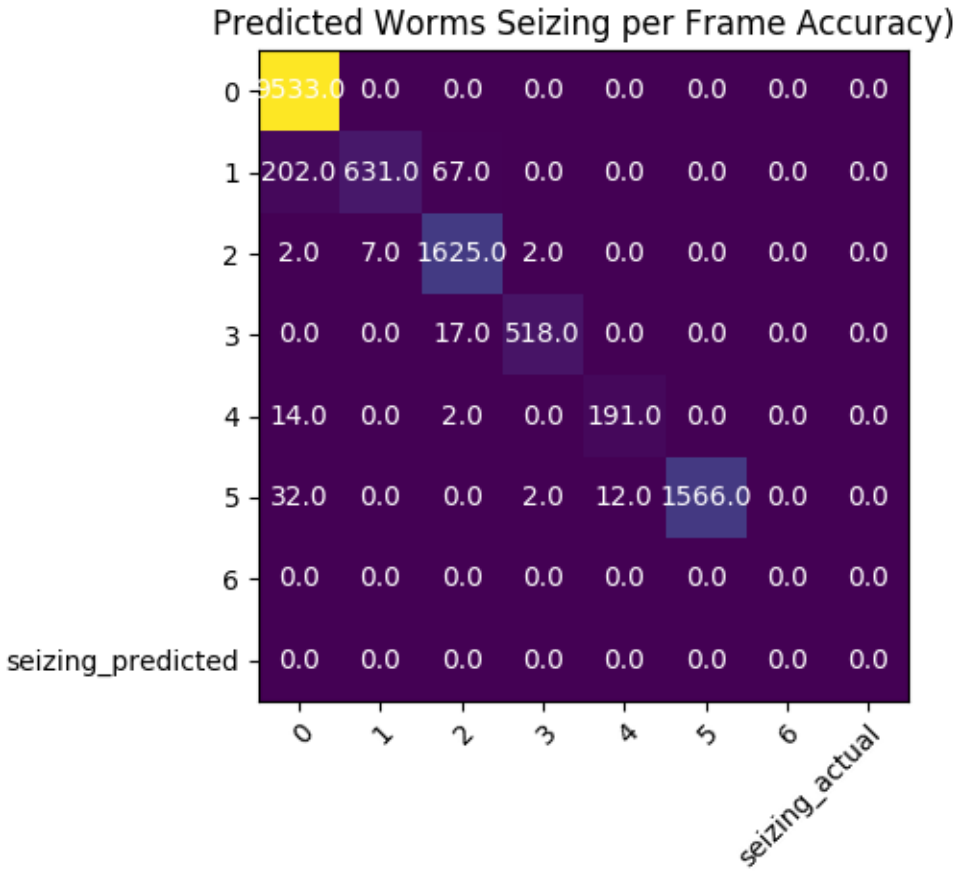

Figure 4: *Confusion matrix of raw model predictions (y-axis) vs. human -labels (x-axis) per number of worms seizing being predicted for each frame of one video (*this video was used in the training class).*

Figure 5: *Over-estimation bias of model predictions shown by comparing human-labeled to model-labeled average seizing times per test video.*

Figure 6: *Comparison of human (x axis) vs. model (y axis) predicted average time seizing per video (z axis).*