CS230

Automated Detection of Epileptic Seizures in Rodents for High-Throughput Analysis

Michael P. Reddick, Alex H. Ruch, Steven J. Tan
mreddick@stanford.edu, aruch@stanford.edu
sjtan@stanford.edu

Abstract

Epilepsy is a neurological disorder that is often associated with seizures due to the increased and uncontrollable firing of neurons in the brain. A team of Stanford researchers and physicians led by Dr. Max Wintermark is developing a targeted drug delivery treatment to avoid surgery. To evaluate the treatment, researchers must manually count the number of seizures in treated rodents over hours of video. Thus, an automated, deep-learning approach to identify and count the number of seizures can save significant hours of labor. Here, we use Inception v4 to encode the frame information and feed the time series to an RNN. Our model exhibits nearly perfect recall and a high degree of specificity in our test sets. In future work, we propose using frame differencing to suppress background and enhance signal due to motion or a 3D convolution network to perhaps improve performance.

1 Introduction

Epilepsy is a neurological disorder prevalent in the United States, with approximately 2 million people affected, and 3% of the population diagnosed by age 80. Epilepsy is often associated with seizures due to the increased and uncontrollable firing of neurons in the brain. Treatments of epilepsy include anticonvulsant/anti-epileptic drugs (AED) and invasive surgery that targets problematic brain tissue. Unfortunately, one-third of patience do not respond to AEDs, and surgery comes with risk of life threatening consequences, such as infection or permanent brain damage. A team of Stanford researchers and physicians led by Dr. Max Wintermark is developing an approach using Magnetic Resonance guided Focus Ultrasound (MRgFus) and intravenously injected microbubbles to selectively incise the Blood Brain Barrier (BBB), allowing the neurotoxin quinolinic acid to cross into the brain and destroy the problematic tissue [1]. MRgFus has great precision such that the neurotoxin only destroys tissue local to the small BBB incision, thus mitigating the limitations of current epilepsy treatments.

Wintermark's team is currently testing this approach in epileptic mouse and rat models, in which they quantify seizure episodes before and after treatment. However, this process is slow and laborious as it requires human operator to visually inspect upwards of 1000 hours of video, limiting how quickly scientists can analyze results and iterate over experimental conditions. Here, we use a combination of Inception v4 feature encoding with an LSTM to detect seizures with nearly perfect recall and specificity. We also propose an alternate model combining frame differencing with 3D convolutions to potentially improve seizure detection. We envision our model can be generalized and deployed for use by the epilepsy and seizure research community.

CS230: Deep Learning, Winter 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

Several papers in the field of motion detection and surveillance video have attempted to address similar problems involving the identification of specific types of motion for a fixed camera position. These approaches focus on various methods of suppressing background signal through techniques such as frame differencing, optical flow detection, or background classification.[2]

ReMotENet developed by Ruichi Yu and colleagues used frame differencing to suppress the background signal of surveillance videos.[3] They then used a series of 3D convolutions to integrate data across channels, followed by a spatial-temporal attention layer to enhance the weighting of specific regions of the image, and finally followed by another series of 3D convolution across the temporal dimension. Their network showed improved performance over simple 3D convolutions, but was mostly focused on gross movements across the field of view.

Optical flow methods such as one developed by Huang *et al.* estimate motion by generating a vector field of pixel movements between frames. [4] The propose a more efficient method of motion detection by detecting the optical flow of the video and then mask out the estimation of the background optical flows to retain the relevant motion. Optical flow methods can be slower than other motion detection algorithms, and their added added benefit of tracking motion even during camera movement was unnecessary for our current task.

Background classification identifies specific patterns to mask out spurious signal and pull out foreground motion. WisenetMD developed by Lee *et al.* used background subtraction method along with a secondary check to help remove additional false positives.[5] Their methods allowed for more robust motion detection even with more dynamic background elements such as moving trees or waves.

3 Dataset and Features

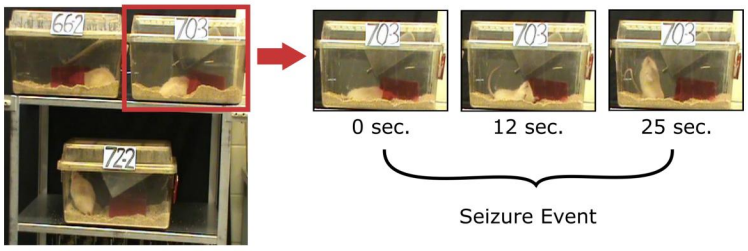

Figure 1: Example frame from a seizure video and select frames corresponding to a seizure event.

We have access to upwards of 1000 hours of video generated by the Wintermark team, spanning 60 days with 10 hours of recording each day and for 4 groups of cages. Each video is a recording of multiple epileptic rats or mice, each housed in individual cages. Annotations consist of time points corresponding to the beginning of a seizure episode by each animal and the corresponding video. Unfortunately, only a small subset of the data is currently annotated, and we are working with the team to continue annotating more data.

4 Methods

Image segmentation. Since each video contains multiple rodents, we first needed to segment around each cage, which each contains one rodent. We did this by manually drawing bounding boxes of the same dimensions in the initial video frame, and cropping along this boundary through all of the frames.

Generating feature vectors from Inception v4. We next converted each frame of the .MP4/.MPG videos files into numpy arrays of dimensions (w x h x 3), which were then reshaped with Tensorflow's bilinear resizing function into frames of size 299x299 using to match Inception v4's input shape. Finally, these arrays are propagated through the Inception v4 NN up through the penultimate layer (i.e. pre-classification), the output of which serve as our feature vectors (1 x 1536) for input into the

RNN. To access the data in more manageable pieces, we split each video file into 10 minute clips of 18000 frames each, resulting in arrays of size (18000, 1536).

Generating label vectors from seizure annotations. The duration of seizures in rodents ranges from 10 secs – 1 min. Consequently, we decided that the network should learn with a resolution of 5, 10 or 20 seconds. In other words, a single example could be a 5 sec video clip, in which each frame is represented as a feature vector as outputted by Inception v4 or as a difference between two frames. Each window is annotated as 1 (positive for seizure) or 0 (negative for seizure). A window is labelled positive when at least 1 second the window overlaps the start of the seizure.

LSTM. Our choice of architecture combines Google's Inception v4 image recognition model with a simple RNN. This approach uses transfer learning by utilizing the pre-trained weights of Inception v4 to transform our time series image data into a time series of "feature vectors" that will serve as inputs into the LSTM model. We reasoned the image recognition capabilities of Inception could also extract important features of each frame (or image) in our videos. This transformation in theory should provide an easier learning task for the RNN, as it has to learn temporal relationships between simpler inputs. We chose a simple LSTM network [7], in which we only care about the output of the final memory block. This output is fed into a fully connected layer before finally propagating through a sigmoid layer for seizure detection.

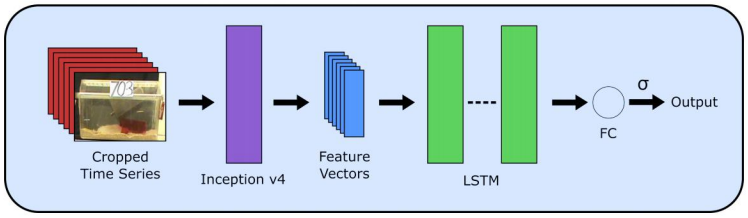

Figure 2: Schematic of LSTM architecture. Our model encodes frame information using Inception v4 and feeds the feature vectors to an LSTM.

Frame differencing. As an alternate processing approach, we convert the images to gray scale and take the absolute difference between frames at a specified frame interval. This allows up to suppress the background and focus on the motion between frames, which may indicate that a seizure has occurred.

3D convolution net. Our alternative approach was inspired by Yu et. al. ReMotENet architecture for motion detection.[3] We chose to stack five layers that consisted of a 3D convolution and max pooling that down samples over the temporal domain. Each 3D convolution was performed with 16 3 x 3 x 3 (time, width, height) filters of stride 1 x 1 x 1 and 'same' padding. Each max pooling operation was performed with a 2 x 1 x 1 filter of stride 2 x 1 x 1. Following the five 3D conv + pool layers, there is a global average pooling layer to yield a 1 x 1 x 1 x 16 tensor. This is passed through 3 layers, each consisting of 2 filters of 1 x 1 x 1 convolutions. This output is fed into a fully connected layer before finally propagating through a sigmoid layer for seizure detection.

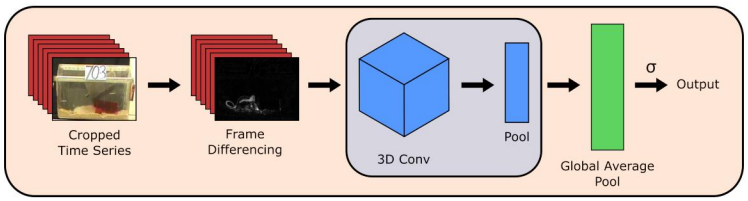

Figure 3: Schematic of proposed frame differencing and 3D convolution architecture. Our model will take the differences between frames at a specified interval and run a 3D convolution net.

5 Experiments/Results/Discussion

Generating Minibatches. In order to facilitate easier training, we randomly loaded 80 feature vector files representing 10 minute video clips, such that at least half of the video files contained at least 1 seizure event. We then created minibatches using the 80 loaded feature vector files to generate

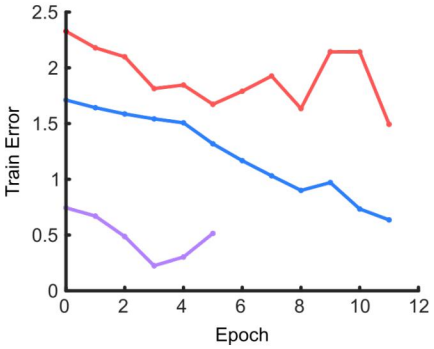

Figure 4: Training error over epoch number for select training hyperparameters using Inception frame encoding and an LSTM.

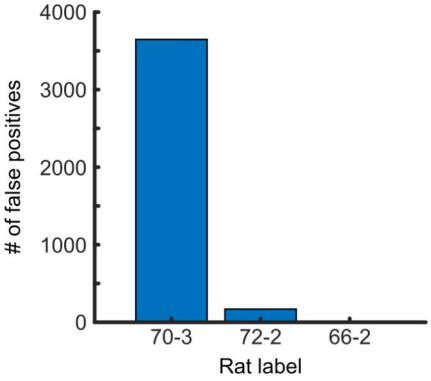

Figure 5: Number of false positive events per specific rat for a selected training run.

mini-batches of 128 5 second clips with the corresponding annotation. Each minibatch contains a minimum of 1/64 seizure examples to ensure the model is exposed to some positive examples in each minibatch.

Hyperparameter search. In order to tune the performance of our models, we also performed random hyperparameter search. Since we desired to tune over 7 hyperparameters, we chose random search over grid search as grid search would be impractical, especially without some prior knowledge to minimize the search domain. For the LSTM architecture, we tuned the following: size of LSTM cell, learning rate, proportion of positive examples per minibatch (note our data predominantly negative examples), minibatch size, sliding window size, class weights, and L2 regularization loss scalar.

RNN Results. We used a simple LSTM with up to 256 hidden units to train on the minibatches with up to 128 examples of the 5 second video clips. We chose to use a sigmoid output function for binary classification of a seizure event and used the cross-entropy loss as our loss function. Below we show three examples showing the range of performance with different hyperparameters. Generally, we see training error trends downward, indicating the model is learning (Figure 4). Interestingly, though Run 3 has significantly lower training error, it also has the worst F1 score. We found Run 3 simply learned to label most of the examples as negative, which is likely a result of the class imbalance in the data set. Notably, Run 2 has the highest false negative penalty among the 3 examples shown, and also yielded the best performance of all hyperparameter combinations tested. These observations suggest the importance of the class weight parameter to counteract the class imbalance of our data. In addition to the class weight parameter, we also found learning rate to be important for performance. Runs 1 and 3 have higher learning rates and training error profiles that do not monotonically decrease smoothly with each iteration. This is in contrast with Run 2, which has a lower learning rate, suggesting that learning rates too far above 2.5e-5 will adversely affect learning efficiency. Figure 8 shows small L2 regularization scalar values (between 0 and 0.1) can generally improve F1 scores for 5 and 20 second windows; however, no regularization gave the best performance for window duration of 10 seconds. LSTM sizes between 200 and 300 generally performed best for 5 and 10 second windows, where as LSTM sizes over 500 worked best for the 20 second window. Beyond the effect of hyperparameters, we found the model was sensitive to any correlation in frequency of seizures and a specific rodent. For example, Rodent 70-3 has over an order of magnitude more false positives compared to Rodents 72-2 and 66-2, but was only was twice as likely to have a seizure (Figure 5).

	Parameters					Metrics	
Run	RNN Size	Learning Rate	Minibatch Size	Minibatch Positive %	False Negative Penalty	Recall	F1 Score
1 (Red)	256	4.40E-05	64	12.5	62.6	100	0.064
2 (Blue)	256	2.52E-05	64	3.125	105	100	0.18
3 (Purple)	128	1.00E-04	128	3.125	18.6	0	4.40E-09

Figure 6: Parameters and scores for selected Inception + RNN runs.

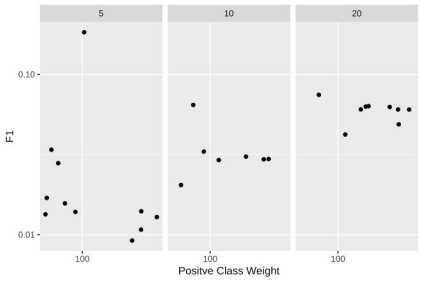

Figure 7: Positive class weight vs. F1 score for different window sizes.

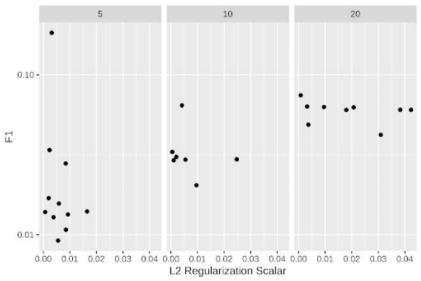

Figure 8: L2 regularization parameter vs. F1 score for different window sizes.

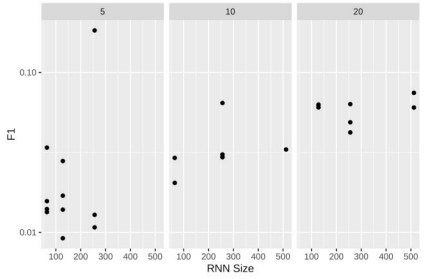

Figure 9: RNN size vs. F1 score for different window sizes.

3D Convolution Results. Due to the amount and nature of our data set, we were not able to complete the preprocessing required for us to fully train our 3D conv model. Specifically, accessing individual frames within the highly encoded .mpg and .mpeg proved highly time intensive using standard python packages.

6 Conclusion/Future Work

The Inception + RNN architecture performed quite well with the best architecture obtaining 100% recall and 96.4% specificity in the test set. However, due to the large class imbalance our data exhibit the precision was very low. For this architecture the most important hyper-parameters were the class weight – false negative penalty – and the learning rate. This accuracy was obtained with < 1,000 positive examples with < 250 distinct seizure events in the training set. The dataset is actively being annotated, so we plan on re-running the hyperparameter search restricted to a more fine-grained area with more annotations once they become available. We expect the inclusion of more annotations and rats in our dataset to boost performance. One additional area we would like to explore with the Inception + RNN architecture is only reusing up to an earlier layer of the Inception network. The Inception architecture was optimized for image classification, where our network needs to be more optimized for pose detection, so we would expect that re-training the final few layers of the Inception net would yield higher accuracy. Unfortunately, given our current computational resources this is infeasible since any earlier layer would yield > 40 TB of feature data to feed into the RNN.

Due to a data-processing bottleneck and instability of our Sherlock, our data storage system, we were not able to fully train a network with frame differencing followed by a 3D convolution. Our preliminary method resulted in a model that would have taken 5+ days to train, so we plan on developing an optimized data pipeline for this architecture and performing a hyper parameter search to tune the model. Ultimately we expect some variation of the frame differencing architecture to obtain superior results to the Inception + RNN network due to the usefulness of frame differencing as a preprocessing step as shown in 3. The main feature needed to detect a seizure is the type of motion of the animal inside the cage, and because there is a fixed point of view and no other objects moving, frame differencing reveals the animal's motion quite clearly.

7 Contributions

All members contributed equally to the original data pipeline and minibatch generation/labeling. Alex was the main implementer for the Inception + RNN tensorflow network, Steven implemented the frame differencing approach and Michael primarily developed the 3d convolutional network. Steven and Michael also contributed to a greater extent in the report preparation.

8 Code

Code available at: https://github.com/aruch/seizure-detector

References

[1] Y. Zhang, H. Tan, E. Bertram, J.-F. Aubry, M.-B. Lopes, J. Roy, E. Dumont, M. Xie, Z. Zuo, A. Klibanov, K. Lee, M. Wintermark. "Non-invasive, focal disconnection of brain circuitry using magnetic resonance-guided low-intensity focused ultrasound to deliver a neurotoxin." Ultrasound in Medicine & Biology. 42.9 (2016): 2261-2269.

[2] K. Sehairi, C. Fatima, J. Meunier, "Comparative study of motion detection methods for video surveillance systems." J. Electron. Imaging. 26, 23025 (2017).

[3] R. Yu, H. Wang, L. S. Davis, "ReMotENet: Efficient Relevant Motion Event Detection for Large-scale Home Surveillance Videos." arXiv:1801.02031v1. (2018).

[4] J. Huang, W. Zou, Z. Zhu, and J. Zhu. "An Efficient Optical Flow Based Motion Detection Method for Non-stationary Scenes." arXiV:1811.08290v2 (2018).

[5] S.-H. Lee, S.-C. Kwon, J.-W. Shim, J.-E. Lim, J. Yoo, "WisenetMD: Motion Detection Using Dynamic Background Region Analysis." arXiV:1805.09277 (2018).

[6] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning" arXiV:602.07261 (2016).

[7] S. Sidor, "Simple Implementation of LSTM in Tensorflow" https://gist.github.com/siemanko/b18ce332bde37e156034e5d3f60f8a23.