Recurrent Neural Network-based approach for classifying Audio/Visual Stimuli EEG Data

Tyrome Sweet and Taran Rallings

University of California Merced

Introduction

Brain-Computer interfaces are machine learning applications that read in brain waves and uses the information obtained to control a device or machine. This scientific and technological research seeks to improve the lives of individuals suffering from limited mobility. In addition to prenatal disabilities, many people each year suffer accidents causing the loss of a limb, or paralysis. By understanding, and using signals from the brain, this will allow people to operate an exoskeleton, robotic prosthetic, virtual keyboard or mouse, wheelchair and many other devices which will enable them to move around more comfortably in the world.

The problem that Brain-Computer Interfaces currently have is with the classification of the brain signals in real time. The goal of this project is to utilize a recurrent neural network to classify and characterize electroencephalography (EEG) signals on a time-series. The outcome from this work will be fast-scalable algorithms, that are able to determine the stimuli the brain is responding to and identify low-dimensional signals within these noisy time-series that can be used to make predictions of subject actions. More specifically, this project will accomplish the following objectives:

1): Classify EEG signals from subjects receiving either auditory or visual stimuli.
2): Identify low dimensional structures through ICA that are indicative of either auditory or visual modes of stimuli.

The next step after the completion of this project is to make predictions on the stimuli the subject was exposed to. By accurately classifying and predicting stimuli, this will give us more insight on brain waves, and will bring us a step closer to analyzing raw data in real time, which will greatly improve the efficiency of Brain-Computer Interfaces.

The Experiment and Data

Dr. Ramesh Balasubramaniam's lab conducted a two-part experiment that exposed the subject to different stimuli. The two stimuli types were audio and visual, and more specifically tones and flashes. The subject attends a block of isochronous, uninterrupted tones and flashes that total 140 per stimuli in part 1 of the experiment, then attends another block that have occasional omissions in either the tone or flash, but still total up to 140 per stimuli. The blocks vary in frequency of the tones or flashes, and the subjects were asked if the blocks were slower or faster than the block that preceded it.

The Electroencephalography(EEG) is a tool that measures brain waves, and the waveguard (EEG cap) can vary on the number of channels (or nodes) that it uses to cover the surface area of the scalp/head **[5]**. Dr. Ramesh Balasubramaniam's lab utilizes a 32-channel cap for experimentation, and the basic layout of the cap in relation to the regions of the brain are shown in **Figure 1.**

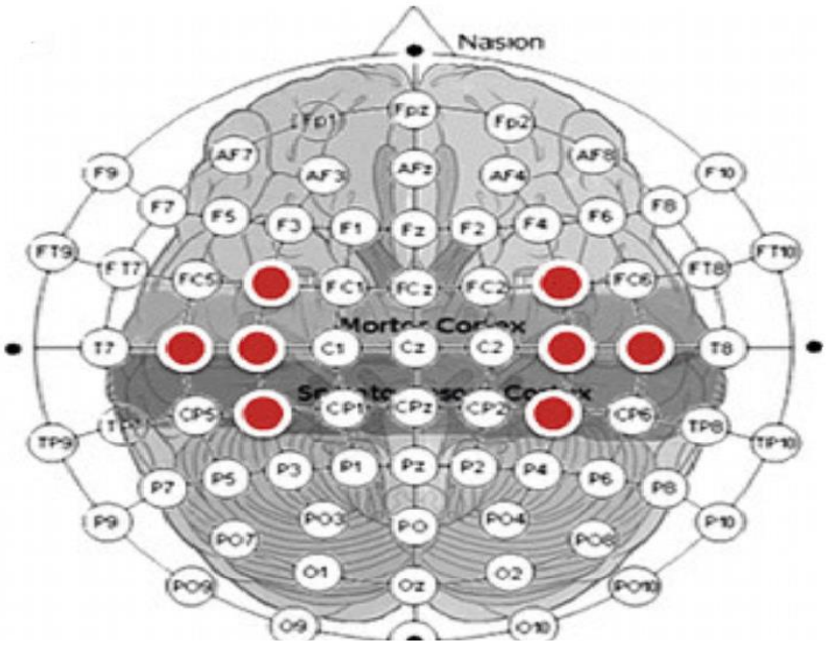


Figure 1: [9] Visual structure of how the EEG Cap channels are positioned the scalp, with the corresponding regions of the brain that is being monitored.

Shown by **Figure 1** is a 32-Channel EEG cap that is displayed on top of a basic model of the brain, which shows the Somatosensory cortex and motor cortex (2 main regions of the brain). The channels, or nodes, are labeled as such: Odd numbers are on the left, Even numbers on the right. Furthermore, letters are used to specify the region of the brain and are labeled as such: F = Frontal Lobe, T = Temporal lobe, C = Central lobe, P = Parietal lobe, O = Occipital lobe, and Z = electrode placed on mid-line that is shown by the Nasion (indention between forehead and nose) and Inion (Are between back of the neck and skull).

In response to the stimuli received, the subject’s brain waves changed in response to it. These signals were picked up by the EEG, and recorded. The recorded data was captured Data is stored as a. set and associated. fdt file. The .set file can be opened via EEGLAB within MATLAB. Once the file is loaded with EEGLAB, the EEG data is in the structured array EEG. There are two main points of interest, the EEG data itself, and the corresponding event codes and event latencies.

Electroencephalography(EEG) and Independent Component Analysis

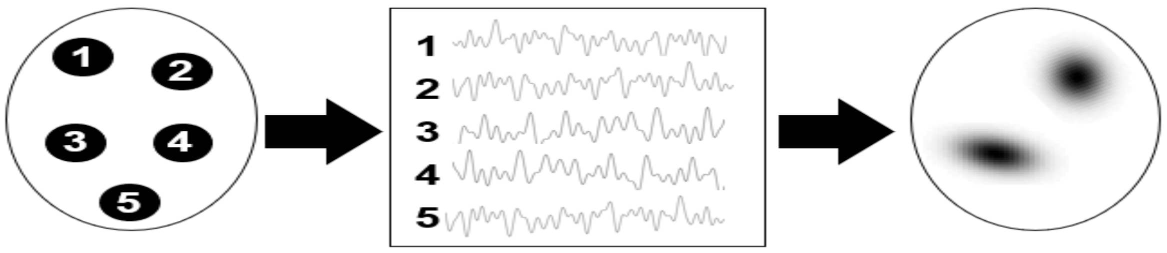


Figure 2: (Left) Subjects wear a waveguard (EEG Cap) that captures electric signals from the surface of their scalp at different spatial locations. **(Center)** EEG signals are correlated and noisy, but when transformed with signal processing techniques like independent component analysis (ICA) reveal low-dimensional coherent structures within the brain **(Right)**. (EEG consists of 32 independent channels, for simplicity show only 5.)

As shown in **Figure 2**, EEG data consists of independent recordings of electric signals from the surface of a subjects' scalp. The data collected reflects noise from interference, (muscular/eye movement) and underlying electronic signals within the brain of the subject. A standard approach towards processing of EEG data is to attempt to resolve the 32 channels measured externally into independent signals internally. One popular technique in this area is the use of independent component analysis (ICA) **[1,2]**. This technique has developed in numerical linear algebra and seeks to represent high dimensional data into a smaller set of components. Unlike the more commonly used principal component analysis (PCA), ICA does not produce orthogonal components or require the underlying signals to be Gaussian. As such, ICA is particularly well suited towards datasets, like EEG, that consist of multiple simultaneous recordings **[8]**. The first step in my analysis will be to translate EEG data into lower dimensional signals with ICA using the MATLAB codebase EEGLab **[3]**.

After transferring the EEG data to independent components, we will develop a recurrent neural network (RNN) that seeks to classify subject data according to whether individuals received either an auditory or visual stimulus. I will train my RNN by first providing it with a small data set (4 subjects in each class) and then attempt to classify the remaining 24 subject readings (12 in each class) into their appropriate class (auditory or visual) **[6/7]**. There are many choices for neural networks, but I will begin by exploring fully connected networks that employ rectified linear units (ReLu). These approaches have been previously successfully applied to EEG data **[4]**.

To facilitate real-time classification and analysis of subject behavior from EEG data, it is necessary to identify and exploit low-dimensional signals. After being able to fully characterize individuals according to their audio or visual stimuli, we proceeded to identify characteristic signals in the independent components indicative of either stimulus. More specifically, we used cluster analysis to seek common components or relationships between components that indicate subject exposure. It is our expectation to retain the ability to discriminate between subjects while using substantially less data.

The two MATLAB datasets were loaded into two csv files to visualize the data as just numeric values instead of graphs and charts. This allowed us to see exact values, at exact time stamps of the events, which ensured that the model was being trained on the correct events and times. The csv file containing the EEG signals are 1.7GB in size, and with the time scale being on Milliseconds, this amount of data was sufficient in training the model.

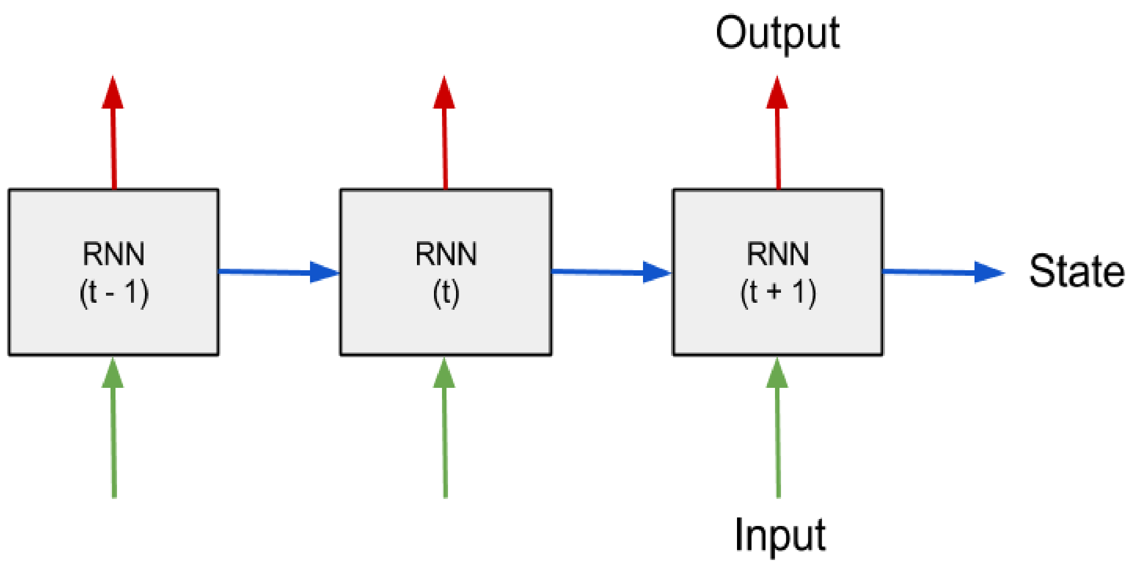
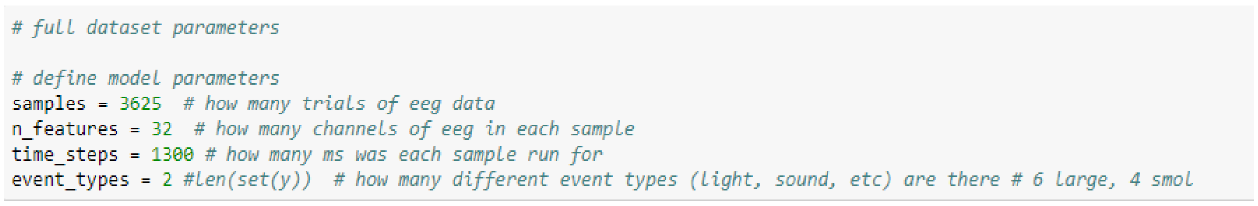


Figure 3: [6/7/10] EEG Signals are sequential and continuous. Instead of “Neurons” LTSM networks use blocks that have memory of previous states and timesteps which it uses to makes predictions on. LTSMs are designed to avoid long term dependency problems, and can learn information for long periods of time.

Recurrent Neural Network

For this project, we created a Long short-term memory (LTSM) network. LTSM networks were created by Sepp Hochreiter and Jurgen Schmidhuber in 1997, and has a higher rate of success with pattern recognition compared to other architectures. LTSM networks are also capable of handling discrete time steps, long delays between events, and signals that mix low and high frequency components which makes it a logical approach for EEG classification. **Figure 3** gives a basic overview of the architecture of recurrant neural networks, but our network uses a different amount of blocks, which is 100.

For reproductivity of the code, we set the seed = 42, and made sure to initialize the random seed generator. Prior to running the Model, we first had to load and reformat the data from the two csv files. The 32 channels are seperated and matched with the Event time stamps. Since brain activity occurs in specific regions of the mind, not all 32 channels will pick up the activity produced by the stimuli, so all inactive channels were zerod out based on the events.

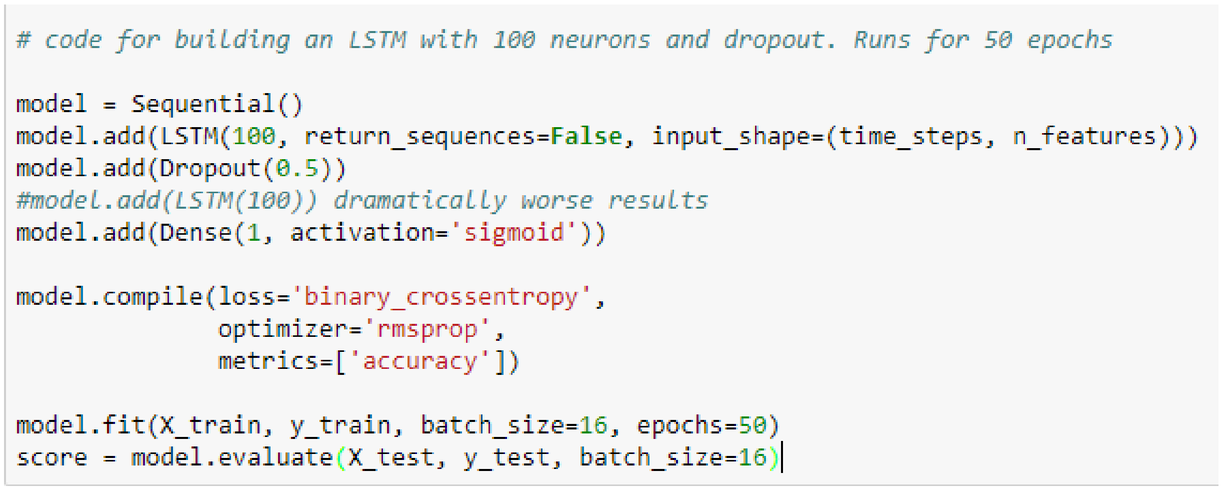


```
# full dataset parameters

# define model parameters
samples = 3625  # how many trials of eeg data
n_features = 32  # how many channels of eeg in each sample
time_steps = 1300 # how many ms was each sample run for
event_types = 2 #len(set(y))  # how many different event types (light, sound, etc) are there # 6 large, 4 smol
```

Figure 4: Screenshot from code showing the parameters and values used for the model. There were 3625 trials that was sampled, per each sample there 32 channels and the duration of the trials were 1300 milliseconds per run. There were 2 event types, Sound or flashes, used to classify the data.

Figure 4 provides a code snippet of the parameters used for the model and provides explanation on where the values came from. Stratification is used to rearrange the data to ensure each fold is an accurate representation of the whole. We used binary classification, so ensuring that each fold has half the instances was crucial for testing the accuracy of the model. **Figure 5** shows



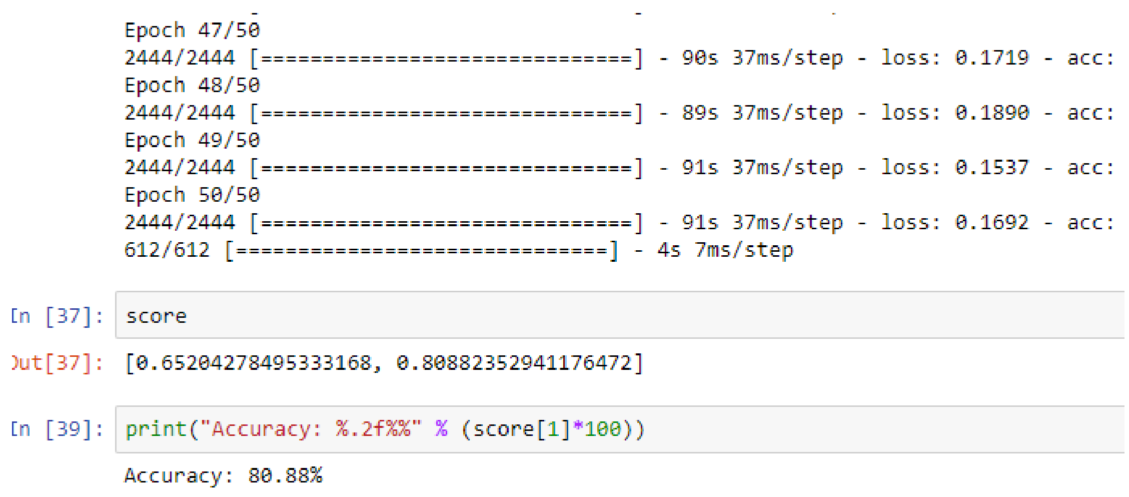
```
# code for building an LSTM with 100 neurons and dropout. Runs for 50 epochs

model = Sequential()
model.add(LSTM(100, return_sequences=False, input_shape=(time_steps, n_features)))
model.add(Dropout(0.5))
#model.add(LSTM(100)) dramatically worse results
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

model.fit(X_train, y_train, batch_size=16, epochs=50)
score = model.evaluate(X_test, y_test, batch_size=16)
```

Figure 5: Snippet of code showing the 100 blocks the LTSM used. The input shape was created by the time steps (1300 milliseconds) and the features (32 channels). A dropout rate of 0.5 was used for this model, and we calculated loss by using Binary cross-entropy.



```
Epoch 47/50
2444/2444 [==============================] - 90s 37ms/step - loss: 0.1719 - acc:
Epoch 48/50
2444/2444 [==============================] - 89s 37ms/step - loss: 0.1890 - acc:
Epoch 49/50
2444/2444 [==============================] - 91s 37ms/step - loss: 0.1537 - acc:
Epoch 50/50
2444/2444 [==============================] - 91s 37ms/step - loss: 0.1692 - acc:
612/612 [==============================] - 4s 7ms/step

In [37]: score

Out[37]: [0.65204278495333168, 0.80882352941176472]

In [39]: print("Accuracy: %.2f%%" % (score[1]*100))

         Accuracy: 80.88%
```

Figure 6: Screen shot showing Binary Cross-entropy of 0.65 and an accuracy of 80.88%. The best run for this model had a binary cross-entropy of 0.41 and an accuracy of 85%. The model results always fell in between 80-86% with each run.

Results/Discussion

Figure 6 shows the improvement of the model due to the decrease in loss per each epoch. The results of the model have a maintained accuracy between 80-86%. This adds to previous work, which shows RNNs are proven to be better when analyzing data in a time series. The key to validating the success of the model lies in the data and experiment. The data was filtered and labeled so that the timestamp and type of event was indicated, as well as the signal value at the time. The model proved to be accurate in the classification of EEG signals, and the next steps for this project are:

1. Change the input from an excel worksheet to a MATLAB file that contains filtered data
2. Use the current model to predict which stimuli (Visual or Audio) that the subject received using filtered data
3. Use MATLAB file containing raw data as an input to see if the model can classify and predict which stimuli the subject received in real-time

As mentioned in the introductory paragraph, the classification and prediction accuracy of the EEG Signals will help advance Brain-Computer Interfaces by allowing them to have a stronger presence in its application to real life.

References

[1] Hyvärinen & Erkki. "Independent component analysis: algorithms and applications." *Neural networks* 13.4 (2000): 411-430.
[2] Delorme, Sejnowski & Makeig (2007). Enhanced detection of artifacts in EEG data using higher-order statistics and independent component analysis. *Neuroimage*, *34*(4), 1443-1449.
[3] Delorme & Makeig, S. (2004). EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of neuroscience methods*, *134*(1), 9-21.
[4] Tabar, Rezaei & Halici. "A novel deep learning approach for classification of EEG motor imagery signals." *Journal of neural engineering* 14.1 (2016): 016003.
[5] Electroencephalography (EEG) Scan Definition & Test Results. (n.d.). Retrieved November 19, 2017, from https://www.emedicinehealth.com/electroencephalography_eeg/article_em.htm#electroencephalography_eeg_introduction.
[6] Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.
[7] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
[8] Luck, S. J. (2014). *An introduction to the event-related potential technique*. MIT press.
[9] Vourvopoulos, A., i Badia, S. B., & Liarokapis, F. (2017). EEG correlates of video game experience and user profile in motor-imagery-based brain–computer interaction. *The Visual Computer*, *33*(4), 533-546.
[10] Posted by Emmanuelle Rieuf on October 3, 2017 at 3:00pmView Blog. (n.d.). A simple neural network with Python and Keras. Retrieved December 11, 2017, from https://www.datasciencecentral.com/profiles/blogs/a-simple-neural-network-with-python-and-keras