8-2022

Towards Real-Time Classification of Human Imagined Language

Joseph Zonghi
jaz8207@rit.edu

Towards Real-Time Classification of Human Imagined Language

Joseph Zonghi

Towards Real-Time Classification of Human Imagined Language

Joseph Zonghi

August 2022

A Thesis Submitted
in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in
Computer Engineering

R·I·T | Kate Gleason
College of ENGINEERING

Department of Computer Engineering

Towards Real-Time Classification of Human Imagined Language

Joseph Zonghi

Committee Approval:

Dr. Cory Merkel *Advisor* Date
Professor

Dr. Minoru Nakazawa *Advisor* Date
Professor

Dr. Andres Kwasinski Date
Professor

Abstract

The primary goal of this research was to develop a system capable of predicting a given user's imagined language in real time using their brainwave data. This research analyzed both FPGA-based and software based approaches to real-time classification of imagined language. The classification was binary between English and Japanese, and a dataset containing imagined speech from both languages was also created. Another goal of this research was to consider the effects of quantization of the network weights in order to examine the resulting utilization of the FPGA to allow for other applications to run in conjunction with our proposed system. With test accuracies over 95% but real-time accuracies only barely approaching 60%, it can be considered partly successful. Real-time approaches to predicting imagined EEG words are rare, and attempts at predicting imagined language are even rarer. Such a system could be beneficial in helping multi-lingual environments that standard natural language processing systems have difficulty in noticing changes in language, especially those that occur in real time. Further iterations on this proposed system could also assist those who have difficulty articulating speech and would benefit from having a brainwave-based system that is portable and works in real-time. It is hopeful that this work can lead to future iterations and advancements in the realm of real-time imagined speech classification both through their own attempts or perhaps with the help of the English/Japanese imagined speech dataset created through this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

A common problem in the realm of neuroscience is with regards to patients who have anarthria, or the loss of the ability to articulate speech. There are many ways to assist those with anarthria in order to communicate, but properly classifying electroencephalography (EEG) data obtained from the user's brain wave signals was thought to be an eventual possibility at best. However, as time and technology progress, whole word and phrase decoding using EEG data has been done to varying degrees of success by researchers in multiple studies [1] [2]. They discovered, through the use of invasive measures implanted inside the user's head to record their EEG signals, that classification of imagined words was indeed possible. However, one issue encountered was that post-hoc accuracy was higher than real time accuracy. Furthermore, an invasive device used to perform these measurements incurs surgery and maintenance costs that might not be easily attainable for the average person. Fortunately, non-invasive EEG measuring technologies exist in the form of a headset the user can place on their head. Were these to become cheaper and more commonplace, it stands to reason that many people would consider it a more agreeable solution. However, the major drawback with non-invasive methods is their generally lower accuracy rates compared to that of intra-cranial options typically in the range of 20 to 100 times

worse signal quality when using signal to noise ratio (SNR) as a metric [3].

When analyzing EEG data, a common problem encountered is its heavy dependence on multiple time points scattered across multiple channels. This high level of dimensionality results in standard linear regression systems lacking sufficient accuracy in properly classifying or predicting said data. Therefore, more advanced data classification techniques such as an artificial neural network (ANN) or perhaps one with multiple layers, a deep neural network (DNN), could be beneficial in this case. An ANN's ability to better classify a wide range of input data than that of standard linear regression algorithms could be a suitable choice for this type of problem. An important consideration regarding EEG data though is its time dependence, as data from two nearby time points do have some level of correlation with each other. As such, it is important to consider that a new input data point does have relation with previous or future data points.

To somewhat rectify this issue, a windowed approached could be applied to better represent multiple data points from the EEG signal over a given time frame. This maintains the temporal properties of the data by acting over a span of time rather than a single point, and it would reduce noise present at any given point that may not be present at following or preceding points. However, EEG data has a low signal-to-noise ratio, so taking data points as they are might not lead to any clear results. Through the use of a moving window, the data can have multiple features extracted from it in order to better represent the given EEG input.

Chapter 2

Related Work

2.1 Related Work

As a result of all of these reasons, analyzing and classifying EEG data is not necessarily a simple task. Through this we propose a system wherein a neural network is loaded onto and trained on an FPGA. This FPGA is also able to handle input data, save it across a given window, perform feature extraction on said window, and utilize the aforementioned neural network to classify the input data as either English or Japanese. The target of this being English and Japanese is to have a dataset with its two states having comparatively high variance between each other. Japanese and English have very different syntax and sentence structures [4], potentially leading to similarly different EEG signals due to the inherent temporal property sentences have (with a beginning, middle, and end most of the time). Furthermore, there is a measured difference in a native Japanese speaker's ability to speak English [5], and it might be the case that these difficulties also carry over into imagined speech. Therefore, such a dataset should theoretically be a good fit for having two distinctly differentiable classes to classify. However, the ability to classify these on their own may not provide much tangible benefit. Instead, it might be beneficial to use this EEG data to assist with speech recognition systems that have difficulty differentiating between a change in language in real time. Some solutions to such issues are using

video in conjunction with audio in busy environments, but the accuracies from such pursuits were still below 30% [6]. Perhaps a low cost EEG-based system could assist in environments, particularly learning environments, where multiple languages are present but hard to differentiate through audio alone. Of course, this would incur costs with regards to supplying headsets to users as long distance EEG is not known to be feasible currently.

Decoding speech from EEG has been showing steady improvements in accuracy in the past few years [7][2] both in terms of audible and imagined speech. One attempt [8] with k-nearest neighbor approaches at classification resulted in 58% accuracy for binary classification, with a big takeaway being that syllable classification is easier than word-based classification. A similar approach [9] using Naive Bayes, Support Vector Machines, and Random Forests tried to expand beyond binary and reached accuracies marginally above random chance for Support Vector Machines (20-35% for 5 classes) but over 40% accuracies when using Random Forests-based approach. Echo state network (ESN) based methods have found success in extracting features from EEG data even through unsupervised methods [10], so it might be useful to incorporate similar feature extraction methods if the current setup is found to be unsatisfactory in terms of accuracy. An ESN is a recurrent neural network that specializes in recognizing relationships or extracting features from temporal data, which while they could be done using functions like mean or median, they might not provide enough of a difference between points to allow the network to successfully converge on a solution. When it comes to bilingual classification, there is one previous study that had attempted to do so [11]. They managed to have high accuracies (about 92%) for language classification, but their dataset only contained responses to yes or no questions.

The primary takeaway from these previous studies and how this research can add to them is as follows. First, imagined language classification seems to have only

been performed on a single word pair scale [11], but this work hopes to expand that to classification of language where whole phrases are involved in order to have a more generalized model. Ideally, the phrases would, even within the same language, feature a wide range of muscle/mouth movements if spoken aloud to allow for the most variation in data. More varied input data should theoretically allow for more accurate classification results on real, unscripted imagined speech data. Furthermore, the vast majority of attempts at classification used post-hoc methods where the dataset is static, and nothing is done or calculated in real time with a real user connected.

2.2 Goals

Though many attempts get results slightly above random chance, it stands to reason that such a dataset should be theoretically differentiable. Not to mention studies [2] wherein relatively high (about 90%) accuracies when classifying between consonants and vowels were found or when determining the presence of specific sounds (such as /uw/). It should be noted that [2] did include facial data from a Microsoft Kinect, so a wholly EEG-based approach likely would not reach similar accuracies. However, a particularly successful attempt [12] using a convolutional neural network with careful consideration for headcap channel relevance found accuracies above 90% for binary word classification using the dataset from [2]. Specifically the difference between the two languages physically exists in sentence structure [4], but it is also argued to exist generally speaking for bilingual persons [13] in terms of the process of mentally representing the meaning of a word regardless of language. Based on this, it can be hypothesized that bilingual EEG data would also be differentiable enough to classify between either language.

Through this research, our goal was to find a means of bridging the many various gaps involved. If imagined language is theoretically differentiable and even classifiable on a word-to-word basis [2], we wanted to provide the first steps in doing so at a

bilingual level. Furthermore, real-time approaches to doing so are hard to find, so a generally low-cost, real-time solution could be beneficial as well. Finally, such a system could aid in fields that have strengths in recognizing speech for a target language but need assistance in recognizing changes in language happening in real-time. The specific, tangible goals for this research were as follows.

1. Create a physical FPGA-based system capable of receiving incoming Bluetooth data and generates an output based on features extracted from said data in real time.

2. Based on accuracies obtained from similar work and assumed complexity of the proposed dataset, a successful classification accuracy in real time can be considered to be a minimum of 60%.

3. A thorough examination of the impact of quantization of the network (both in terms of during training and the final network size itself) on the resulting accuracies to show to what extent space on the FPGA can be conserved.

4. Creation of the imagined English/Japanese speech dataset wherein users are instructed to formulate imagined speech in the specified language while reading specific prompts given to them in order to obtain their EEG signals.

Chapter 3

Early Network Tuning and Kara One Dataset Findings

The neural network was primarily trained and tested in Tensorflow for the Japanese and English dataset due to its relative ease of use and ability to quickly change parameters of the data/network/etc as well as ability to train with quantization awareness, but the initial findings regarding the use of the Kara One dataset [2] were done using MATLAB's DeepNetworkDesigner. The Kara One dataset is cleanly marked to delineate changes in the user's state (speaking, resting, stimulus, preparing to speak). This makes the set useful for training purposes and checking the viability of neural networks for a similar approach. However, an important distinction to be made between this dataset and that of the goals for this project is that this dataset was designed with one language in mind. A dataset needed to do training for this project was not known to be publicly available. Thus, this dataset was created in conjunction with Kanazawa Institute of Technology's students because of the number of bilingual students present and the professors specialized in EEG-related fields and will be discussed heavily in Chapter 5. Despite this temporary shortcoming, the Kara One dataset follows a similarly desired structure and served for a decent introduction into working with and analyzing EEG data. It uses a headset with 62 channels instead of 32, so it can be assumed that a 32-channel headset may result in a loss of accuracy due to a loss of the input dimensionality. For the purposes of aligning with the target of this research, all training methods mentioned moving forward will ref-

erence classification between the "resting" and "speaking" mental states marked by [2]. Speaking states are states where the user is physically speaking a given prompt, and resting states are states where the user was instructed to clear their mind. Both states use the same timing, window sizes, EEG sampling rate (1000 Hz), and features. An example of the raw EEG data can be visualized in Figure 3.1 over a range of 30 seconds, but timing labels for each state could not be overlaid on EEGLAB's plotting function. The user begins in a resting state, is introduced to visual stimuli by receiving a prompt, then they prepare to say the prompt by readying the relevant muscles and mouth positions before finally saying the prompt out loud and returning to the resting state. There are approximately 160 individual word state cycles per person. With binary classification, the input data will be pruned to represent two states, bringing the total to approximately 320 data points per person, notwithstanding the extra dimension added when incorporating channels and the features across said channels. Ultimately, the input data becomes 930 (15 features per each of the 62 channels) by 320 per person. There are 12 people present in the study, so there are approximately 3840 data points available for testing and training.

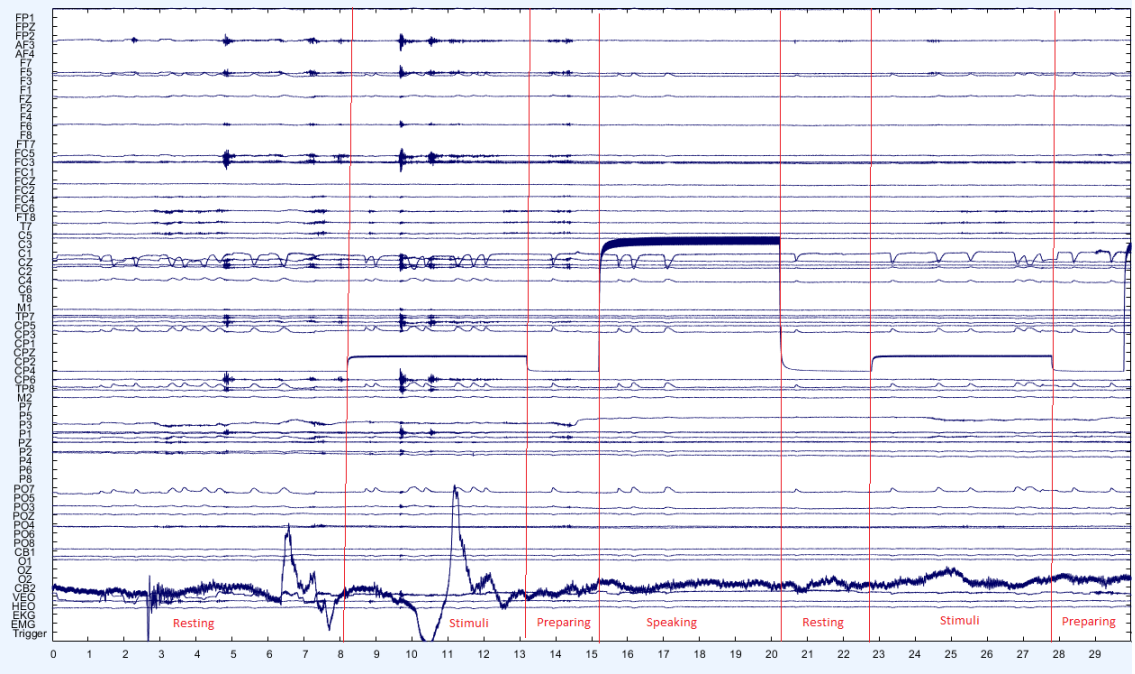

Figure 3.1: Small sample of the raw EEG output data used for training and testing (1000 Hz, 30 seconds). Red denotes labels of the user's mental state.

3.1 Pre-training

The first tests performed in MATLAB were designed to check whether or not a pre-trained model was necessary in obtaining a more accurate output. If pre-training a model on separate user data did in fact result in higher accuracy, it stands to reason that the FPGA need also accommodate initializing itself with a pre-trained model rather than a default or uninitialized one. Figures 3.2 and 3.3 show the results of testing the output of two different types of training methods on a single layer neural network with 62x15 (930) inputs. The first method involved a randomly initialized model without any transfer learning used, and the second method involved using a pre-trained model with transfer learning. Both methods performed multiple training runs, and the results shown are the average accuracy across 50 different training runs as well as the total number of class predictions as seen in the confusion matrices. The difference between the plots within each figure is solely the quantization levels of the

weights of each plot. Quantization in these cases was done by manually changing the value of each weight using MATLAB's fi function. The weights are then saved into a new network with the same structure as the original and used to compute the output.

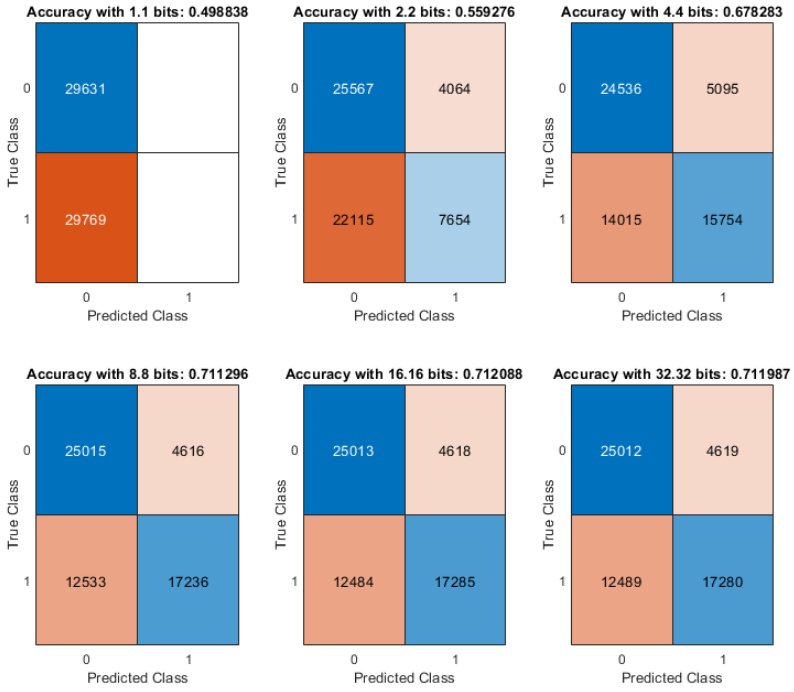

Figure 3.2: Testing results when using a freshly trained model.

When at high quantization levels, the network certainly shows some promise. However, a 71% accuracy for binary classification could be improved upon in this case. The confusion matrices further reinforce this, as the model has about 85% accuracy when guessing 0 (resting state) correctly but 58% accuracy for correctly guessing 1 (thinking state) despite the data set having a balanced set of labels. This in particular will be touched upon later, but it starts to set up the notion that stimuli heavily affect a user's variance in EEG output as opposed to less stimuli-based states like resting. Fortunately, this can be rectified through the use of the aforementioned transfer learning operations.

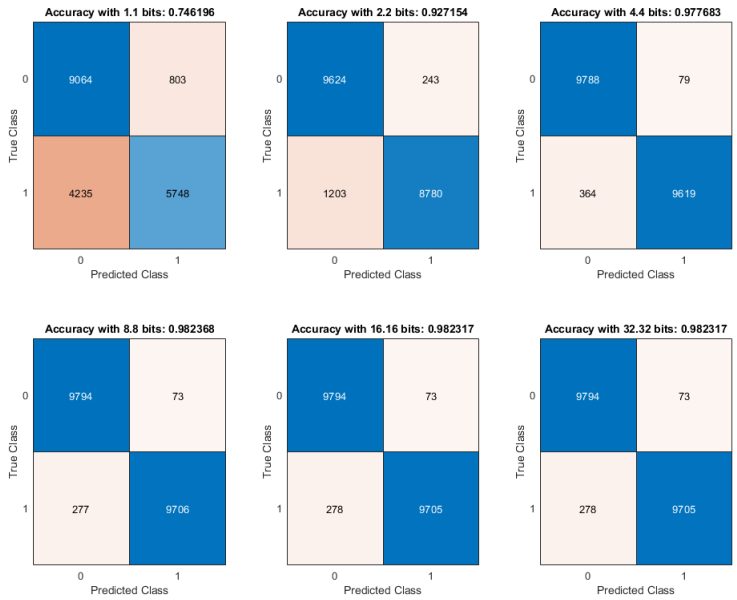

Figure 3.3: Testing results when using a pre-trained model re-trained via transfer learning.

Both examples used the same training and testing data to create these confusion matrices with the notable exception that Figure 3.3 used a model pre-trained on separate data. In this case, even with only a single layer neural network, the accuracy improves by over 30% to see results in the 98% range. However, as seen previously, the presence of stimuli-based data (class 1) incurs a higher level of misclassification.

It should be noted that the testing and (re)training data for both of these examples belonged to specific people P10-P14 (of course with no overlap of the specific training and testing data points). The model that was pre-trained was trained on people P1-P9. The point of this is to show that both a pre-trained model and a model tailored to new users work in conjunction to improve the overall accuracy.

3.2 Inter-Personal Uniqueness

As found in [14] [8], there is a noticeable difference in EEG signals between users who are receiving the same stimuli. Based on this, it can be assumed that EEG

is inherently unique to different individuals but still acts in somewhat predictable manners depending on the action performed as seen in Figure 3.3. The reason for using a pre-trained model is primarily to allow for the general aspects of EEG data to be trained upon, but if a new person's EEG data were to be introduced and used directly as input data, the resulting accuracy would not be as high as it could be. As such, a method wherein the FPGA can retrain on any new user is critical in reaching accuracies closer to optimal. For better visualization, Figure 3.4 shows the results of using the same testing data as the previous two examples without training on the people said testing data belongs to.

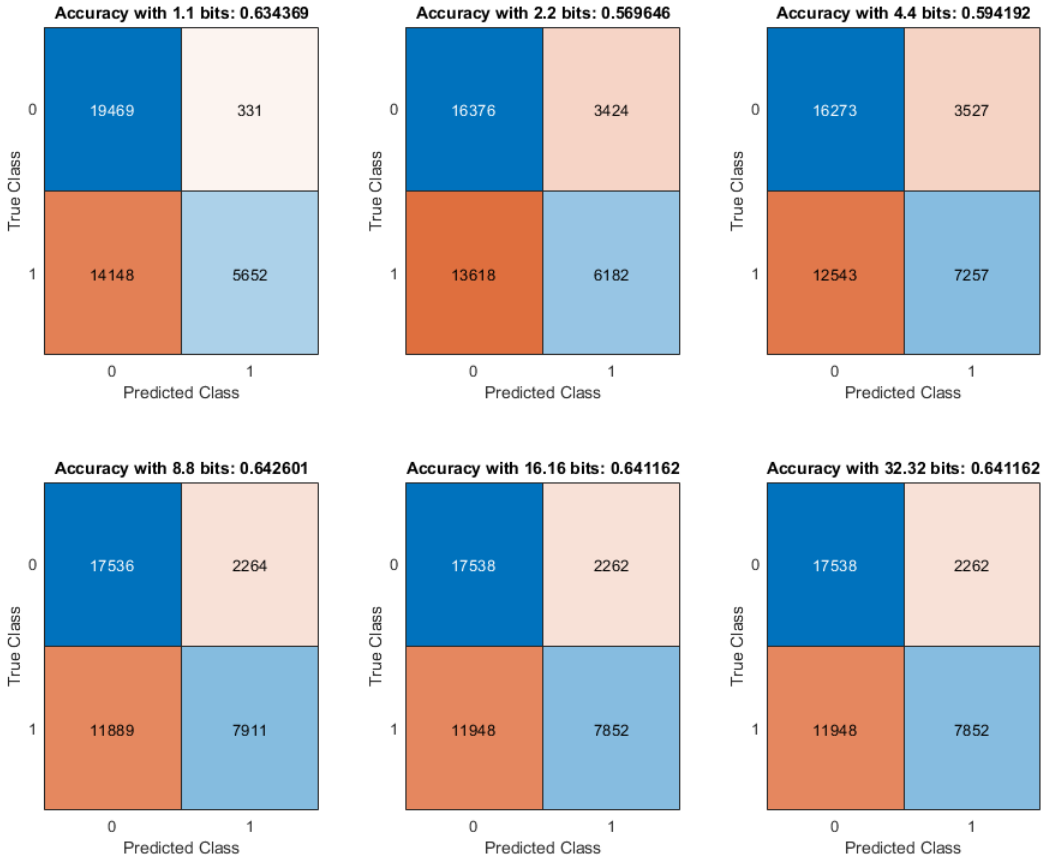

Figure 3.4: Testing results when using a pre-trained model without re-training on a new user.

This case appears to better demonstrate the high levels of personality or unique-

ness an individual user's EEG data has. To give a crude ranking of the three types of testing done here, transfer learning on new users with a pre-trained model has the best accuracies. Training with a brand new model follows behind in second, and testing on new users with a pre-trained model trained on separate users results in the lowest accuracies. Moreover, similar results were found by [12] using the same dataset [2] wherein a 10% drop in accuracy occurred when performing leave one subject out cross validation rather than including all subjects as training candidates. They also grouped specific channels together to help produce a more accurate output, as specific regions of the brain correlate to specific activities and as such specific, spatially significant EEG signals [15].

It is also important to consider the models' comparatively high accuracy at correctly classifying outputs as 0 (resting) rather than 1 (speaking) despite using a completely balanced training dataset. This further reinforces [14]'s point that stimuli-based EEG has a higher level of variance between users. As such, an unbalanced training set that more heavily weighs stimuli-based states could provide higher accuracies overall, but when considering the target dataset, English vs. Japanese imagined speech, both states have stimuli present. Perhaps users imagining their native language have less variance than those who are not. It would be prudent to consider these potential differences in users when it comes to determining how long they should train using the FPGA. Perhaps having users spend a longer time training with their non-native language results in higher accuracies, or vice-versa.

3.3 Quantization Results

Another goal of this research is to examine the effects that quantization of the network through multiple facets. The first of which would be converting the network's values to various quantized levels. Ideally, in order to conserve the most amount of utilization of the FPGA, a network consisting of single bit weights would be ideal.

However, in practice this could lead to lower accuracies due to the comparatively lower precision. Figures 3.2, 3.3, and 3.4 already show examples of the effects of using a "finalized" network and simply changing their quantization levels through post-training rounding. Another example with this framework can be seen in Figure 3.5 which uses the same training parameters as the previous examples but with a 3-layer neural network, with each layer fully connected.

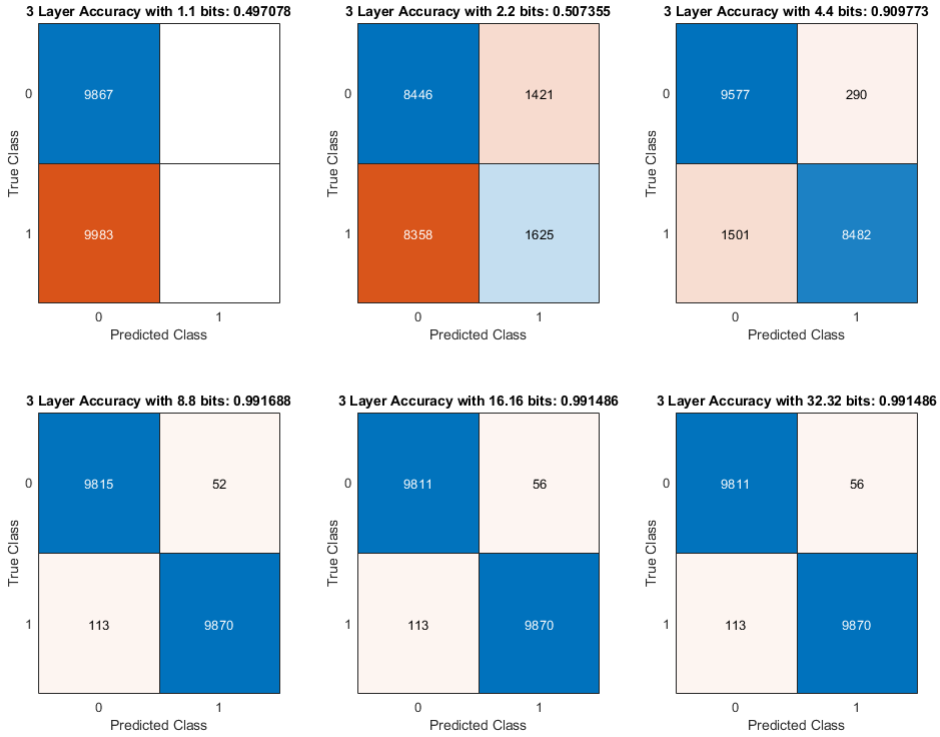

Figure 3.5: Testing results when using a pre-trained 3-layer model re-trained via transfer learning.

Based on these results, it might be assumed that 16 bit fixed-point weights (8 bits for the integer portion and 8 bits for the fraction) are "optimal" for this specific model if solely aiming for the highest accuracy without adding unnecessary utilization. To better test what levels were optimal, a sweeping run of all possible combinations of integer and fractional fixed point sizes from 1 to 32 was performed on a 1-layer neural network. These tests used the same training and testing parameters as those in Figure

3.3 and can be visualized in Figure 3.6.

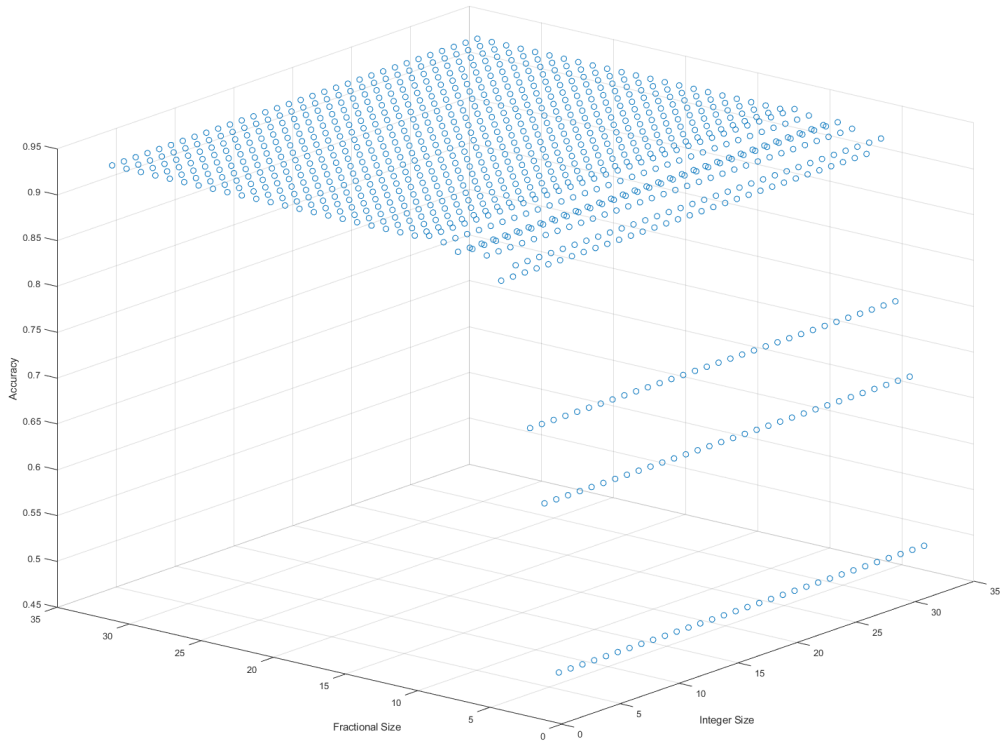

Figure 3.6: 3D scatter Plot of accuracies as a function of quantization for a 1-layer network.

Since the weights are all between 0 and 1, the integer portion of the weight becomes irrelevant when it comes to quantization. Despite that, it is also clear that fractional sizes exponentially drop off in improvements in accuracy as soon as 5 bits or more are used. This behavior becomes even more profound when using more complex networks, such as the 3-layer network shown in Figure 3.7.

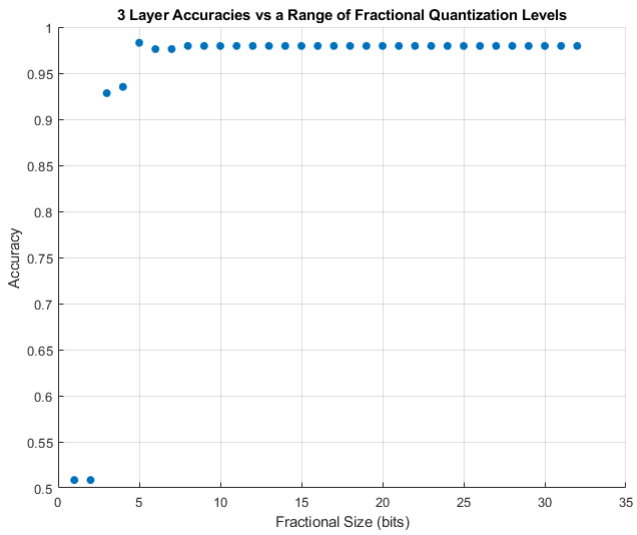

Figure 3.7: 2D scatter plot of accuracies as a function of quantization for a 3-layer network.

This plot follows a very similar pattern to that of Figure 3.6 with the main difference being the scale of the accuracies involved. The 1-layer network appeared to have peak accuracies around 93%, whereas this one sees accuracies in the 97% range. It can be assumed a more complex model would have even higher accuracies. Since the difference in accuracy per plot changes quickly in a short span of bits, it is interesting to note that a 3-layer network sees greater than 90% accuracies at only 3-bit precision, whereas the 1-layer model needs 4 bits to do so.

3.4 Quantization-Aware Training

When it came to designing the network itself, the decision was made to switch from MATLAB's Deep Learning Toolbox to Tensorflow with Python for ease of use with quantization-aware training. However, MATLAB was still used to prepare any training data for its ease of use for quickly adjusting various features, window sizes, and other general meta-level components of the training data. This would then be saved to a MATLAB .mat file for the Python program to import. Many tests were performed in Python to determine the optimal fixed point size with regards to assumed utiliza-

tion. There were three methodologies performed, standard model quantization, i-bit quantization, and 1-bit quantization. Standard model quantization means using Tensorflow's standard training methods and then quantizing the network's weights after training is complete. The other two, named i-bit and 1-bit quantization, use Tensorflow's quantization-aware training functionality to train the network. Quantization-aware training is the practice of training a model to be able to better handle incoming data that might not be a standard floating point number. For example, 8-bit quantization aware training trains the model to handle incoming data that is 8 bits in precision. i-bit training, in this case, would be training the model to handle data of any given i-bit resolution and comparing it against the 1-bit quantization aware and standard training methods. Figure 3.8 shows the results of training on prompt-based speech confined to an approximate network size at various fixed point sizes. The accuracy values shown are also the average of multiple runs.

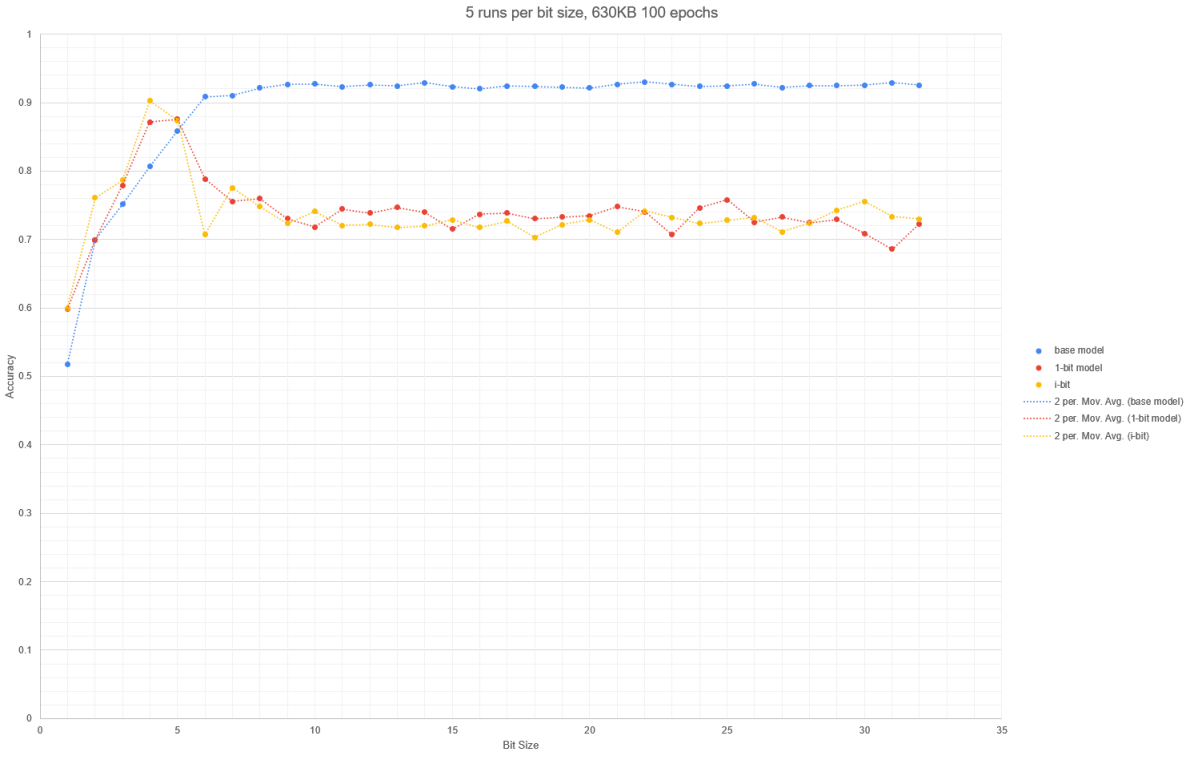

Figure 3.8: Results of the 3 training methodologies per post-training bit rounding.

The x axis refers to the resolution of the weights after training by manually changing them to their respective quantized values. This is true for all three methodologies, where the only difference between them is the level of awareness of quantization. The base model takes no quantization awareness into account when training, the 1-bit model always trains under the awareness it will function with 1-bit resolutions regardless of the final rounding resolution, and i-bit quantization trains under the awareness that it is training on bit resolutions equal to the resolution that its weights will also be rounded to after training. From Figure 3.8, it can be seen that there are few differences between 1-bit and i-bit quantization aware training. They tend to have the same average accuracy regardless of post-training rounding, and this also holds true for a network of half the size as seen in Figure 3.9.

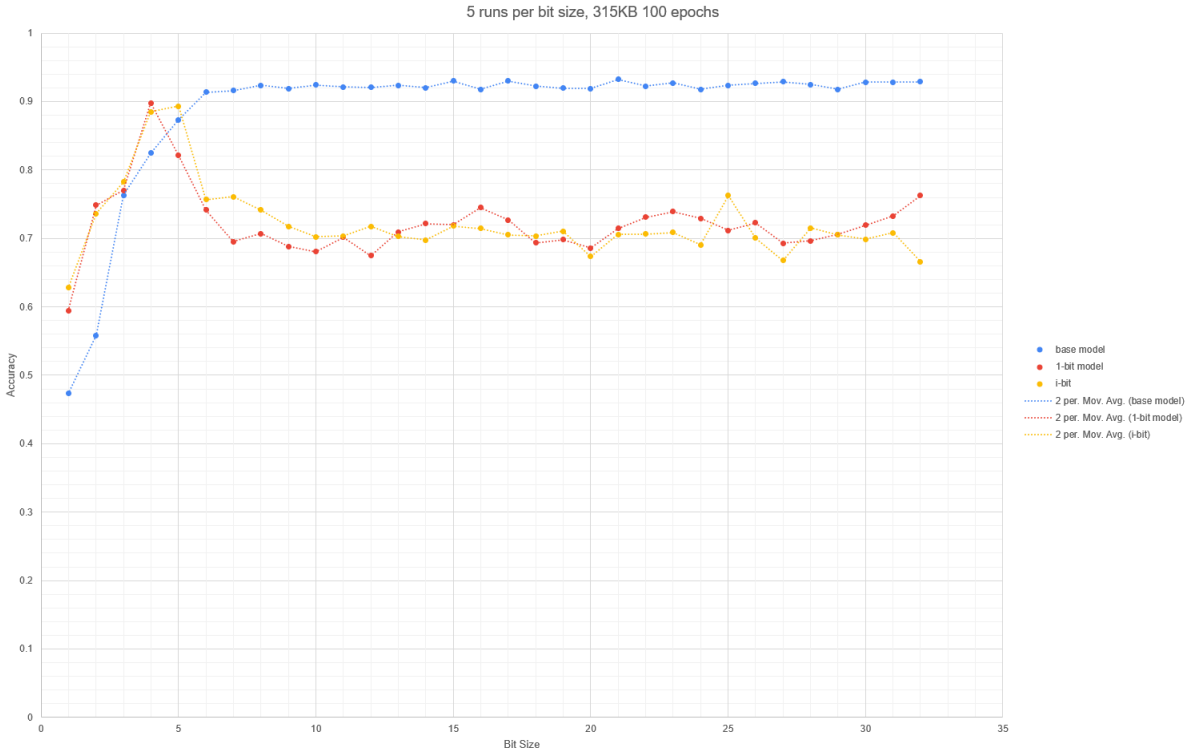

Figure 3.9: Results of the 3 training methodologies per post-training bit rounding.

Another takeaway from these tests is that resolution does not necessarily have a noticeable effect on the accuracy of the network after a certain point. Moreover, a

sufficiently large network can have relatively low numbers of hidden neurons without compromising accuracy. As per Tables 5.5 and 5.7, network sizes above 100 do not necessarily result in noticeably large increases in accuracy, and small networks with only 20 neurons are capable of adapting to training data belonging to a single user.

With regards to quantization, it depends on the network with regards to what training methodology and bit resolution should be used. For a full 32 bit fixed point implementation, using Tensorflow's standard training procedure is recommended. For models that need to use smaller number sizes, such as those below 6 bits, using either single bit or i-bit training would be recommended. For the purposes of this research, speed is important, and as such minimizing the number of weights is also important. i-bit and 1-bit trained models use higher numbers of weights to achieve the same results as a normal model. With more weights present, the network has to spend more time computing the addition of each weight and input combination. Thus, Tensorflow's standard training was used.

Chapter 4

FPGA and Utilization Results

4.1 FPGA Design

By using a field-programmable gate array (FPGA), the final design would be able to perform multiple different tasks simultaneously in real time. An FPGA was chosen due to the ability to easily adjust specific parameters of the design in order to accommodate various rapid prototyping configurations. For example, the neural network would likely go through various iterations, so it would be beneficial to have a generically scalable design to assist with rapid prototyping. The FPGA initially used a three-stage approach to handling the input data as seen in Figure 4.1.

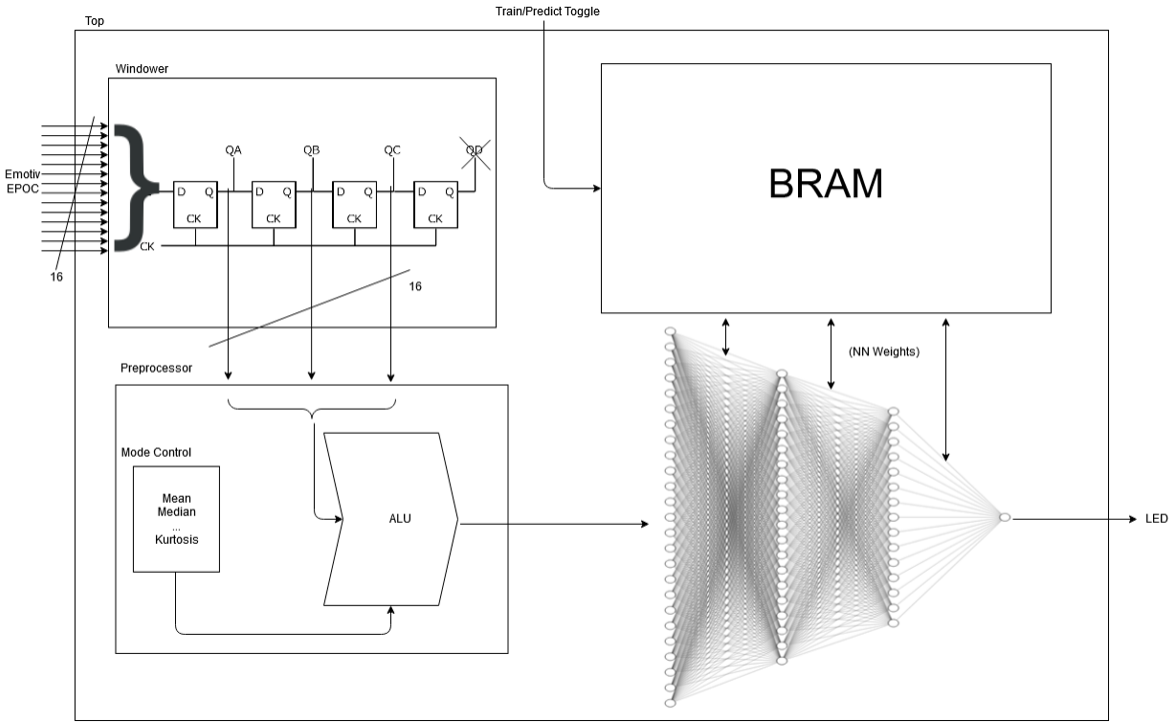

Figure 4.1: RTL diagram of the proposed FPGA layout.

Since one of the goals of this research was to examine the trade-offs between accuracy, number quantization, and FPGA utilization, relatively small-size FPGAs are the target for this. Specifically, the Zybo Z7 Zynq-7020 SoC board was used [16]. This board was chosen for its relatively lower price in conjunction with the Pmod capability to allow Bluetooth connections. Bluetooth was going to be the communication medium between the EEG headset, the Emotiv EPOC Flex, and the FPGA itself. However, many portions of this design had to be changed to allow for a more software-based approach. The Emotiv series of devices require the use of their proprietary software in order to obtain raw EEG data from a network data stream, so the data can not start as being directly sent to the FPGA itself. Figure 4.2 shows the updated design for the FPGA with these changes in mind.

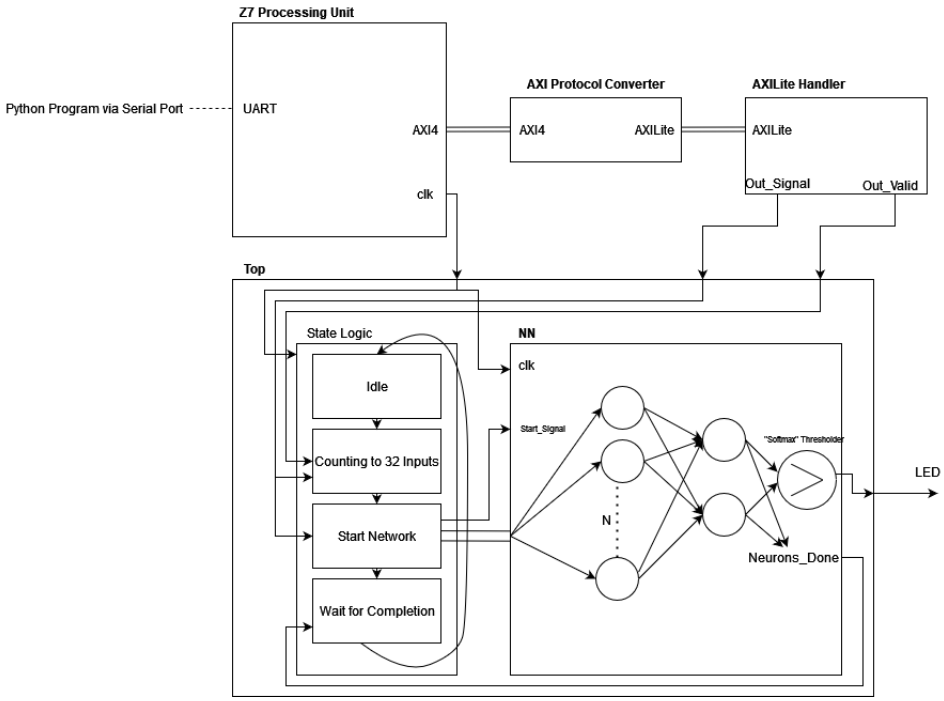

Figure 4.2: Updated RTL Diagram of the final FPGA design.

The FPGA design was ultimately performed using RTL simulations to estimate and observe the effects of changing the network size and quantization levels and the resulting utilizations of the device. Due to only using the mean as input as well as using the UART serial port for communication with the Emotiv EPOC Flex (with a Python program acting as an intermediary), much of the design was moved off of the digital logic. It was initially hoped to be the case that the Emotiv EPOC X or Flex could connect to any Bluetooth compatible device, but they have to go between their proprietary software, the Emotiv Launcher and EmotivPRO. Fortunately, EmotivPRO has a Lab Streaming Layer wherein the raw EEG data can be obtained from the headset and accessed from multiple devices (as well as other data such as motion though not used in this research). A Python program is used to communicate with and interact with this data stream, and it saves the data to a single 1x32 array. For training, each newly obtained array (at a rate of 128 Hz) is saved as a new line in a text file including the current language's corresponding label, 0 for English and 1 for

Japanese. The labels are determined by the Python program itself when it displays the language to the user over command prompt. For real-time purposes, the data is not saved to a file but instead sent over the serial port to the FPGA itself. First, the Python program must convert each number to a two's complement string of binary and then separate the string into separate bytes and send as a packet. This alone currently takes more than 1/128 of a second to process, meaning the input is delayed and the Python program must clear the queue from the data stream to get the next real value, taking even more time. For this reason, the RTL simulation-based approach will be discussed instead with the hardware approach saved for future work.

4.2 Input Handling

The first stage of the FPGA handles receiving the input from the Emotiv EPOC Flex device. The device can transmit the EEG data over Bluetooth using 14-bit ADC signals per channel of a total of 14 channels. Each bit of the ADC corresponds to 0.51μV, so the resolution is limited to steps of 0.51μV. All outputs from the ADC are unsigned, and the designers report an average DC offset of 4200μV [17], meaning negative values can be obtained by subtracting the DC offset from the output values. This can be visualized in Figure 4.3 which shows a theoretical output from the device as if it were a sine wave.

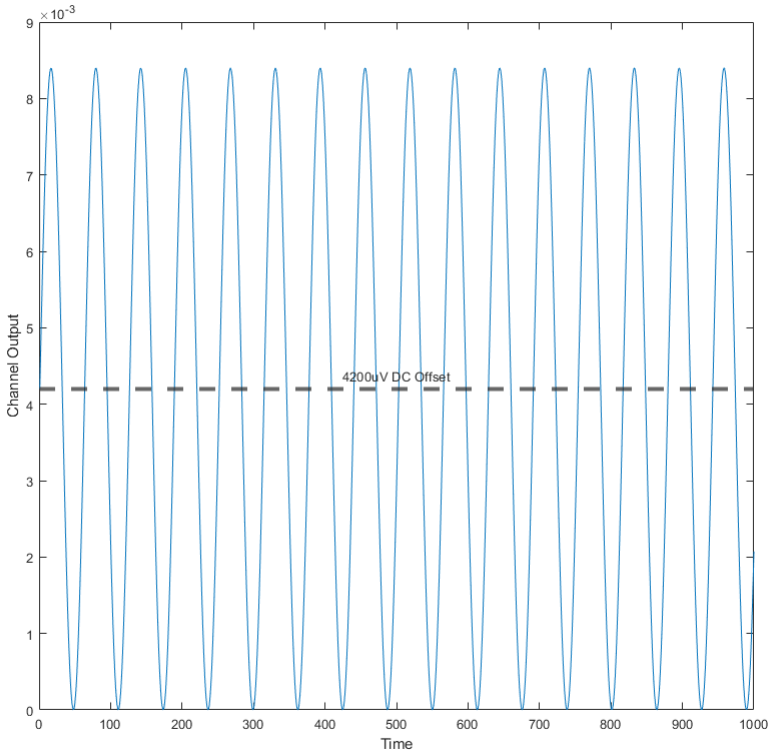

Figure 4.3: Fake sine output of the Emotiv EPOC to show the DC offset.

In order to preserve some of the temporal features of the input data, a windowing system was used. This system allows the FPGA to save data points up to a specified number of time points. For example, if the Emotiv EPOC Flex outputs data at a rate of 128 samples per second, this windower segment could use 128 groups of shift registers all in a row to store a second's worth of data. However, due to the reliance on a Python program first, only the theoretical properties of the windower will be discussed as it was not used in the final design, thought it is still recommended for designs that are able to directly connect to the EEG-recording device of choice.

By taking the output of each register all at once, this windowed segment of the data could then be used to compute the various features. For example, a window of size 128 would allow for 128 different data points, and by taking the average of those 128 signals, you would essentially have a moving average where the signal at

the 128th register gets overwritten by the 127th when a new signal reaches the first register. The FPGA can essentially assume the given window is full of valid signals at any time. Furthermore, at startup, the values would be initialized to 0, so the output would almost certainly be incorrect until the window fills. While window size is one component, another component is what speed the window fills at. A window that shifts values in every second would likely not produce any meaningful output, but one that works in smaller ranges should work better. The estimated time spent for processing in this region is technically just the delay for a single clock period when accessing the output of the registers, but the accuracy of adding one data point to any given window of time is likely low. Therefore, a small amount of time for the window to be filled at startup is necessary. For our method, the FPGA's CPU needs to receive 32 inputs times the window size in order to begin sending the averages of said values. Assuming a 128 Hz sample rate from the headcap, a window of 50 would fill in (1/128) * 50 seconds, or 0.39 seconds.

4.3 Pre-processing Region

Since the data can be considered to be always available from the previous windower region, the pre-processing region could operate on its own, getting the output from the previous stage automatically and assuming it is valid to use to compute the features. Based on the success found in [2] with their choice of features, a similar feature set was used in this case. Those features were computed over the given window and then treated as one data point per feature per channel. Specifically, the features are: sum, mean, absolute mean, median, sixth power mean, standard deviation, variance, maximum, absolute maximum, minimum, absolute minimum, delta of maximum and minimum, sum of maximum and minimum, skewness, and kurtosis seen in Table 4.1. N is the length of the window, and x is the input signal.

While these features were considered for implementation in the FPGA, software

Table 4.1: Table of features computed across the window

Feature	Equation
sum	$\sum_{n=1}^{N} x_n$
mean (μ)	$\frac{1}{N}\sum_{n=1}^{N} x_n$
sixth power mean	μ^6
standard deviation (σ)	$\sqrt{\frac{\sum_{n=1}^{N}(x_n-\mu)^2}{n-1}}$
variance (σ^2)	$\frac{\sum_{n=1}^{N}(x_n-\mu)^2}{n-1}$
skewness	$\frac{E(x-\mu^3)}{\sigma^3}$
kurtosis	$\frac{E(x-\mu^4)}{\sigma^4}$

attempts at training on the data found that the presence of more features resulted in decreases in accuracy as can be seen in Table 5.2 similar to the findings in [2] when using simpler SVM methods with variable numbers of features present.

4.4 Neural Network Design

The neural network region works in conjunction with the FPGA's ROM instantiation. There were two main states that affect the neural network region. The first state is a simple output compute state. The network takes input from the previous feature extraction region and computes the output as is. In this case, upon boot, the FPGA just receives the neural network weights from ROM and computes the output by summing the products of the inputs and the weights as a standard neural network would. The estimated delay for this region will be a direct function of the total amount of weights used. For example, if the network takes all 32 inputs to directly compute two separate output nodes, the FPGA requires what can be considered to be $32t$, where t is the time taken to calculate the product of an input and weight added to the running sum of products. Equation 4.1 shows this relationship for a neural network with L layers and W weights per layer (including the bias weight), and Figure 4.4 shows an RTL mock up of how a single layer network functions.

$$T = \sum_{l}^{L} \sum_{\omega}^{W_l} t \tag{4.1}$$

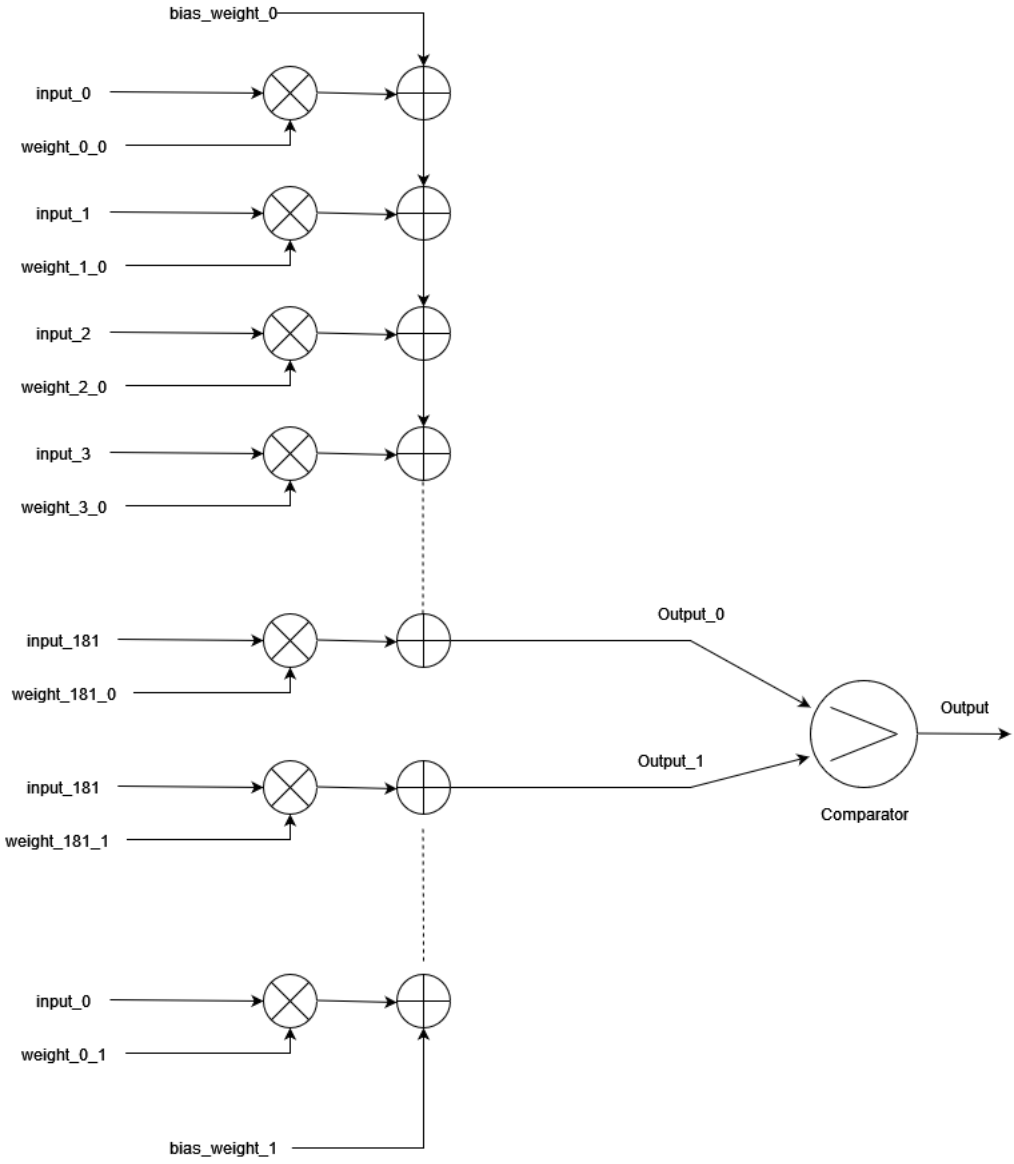

Figure 4.4: RTL mock up of a single layer neural network.

The weight signals correspond to the weights saved in memory ahead of time. They will be preloaded and not influenced or driven by anything during runtime. The input signals are retrieved from the output of the AXI region that gets the values from the processing unit that communicate with the Python script over UART. Since

the summation needs to be done sequentially, the estimated time to get an output is the total time taken for each of an output's summation to complete. The output signals would use a thresholded softmax activation function (Equation 4.2) wherein the signal with the higher value of the two will be chosen as the output.

$$Output = \begin{cases} output\_0, & \text{if } output\_0 \geq output\_1 \\ output\_1, & \text{otherwise} \end{cases} \tag{4.2}$$

4.5 Utilization

Assuming the data coming from Python is raw EEG data, the processing unit on the FPGA uses a C program to compute the moving mean. It uses two arrays to do so. The first array (1x32) contains the sums of the incoming raw data per channel while the second array is a 2D array (Nx32) containing the raw EEG values to subtract from the corresponding channel sum with N being the desired window size. The subtractions do not occur until the first N number of values come in. At this point, the program adds the next incoming values to their corresponding sums while subtracting the first values in the subtraction 2D array. The memory usage of this program can be estimated to be approximately the size of an integer (4 bytes) multiplied by 32 and then by N, the size of the desired window. Those values are then replaced with the newly received values and the pointer moves to the second values to subtract up until N where they loop back to the first values (time points with a multiple of N). At this point the program also sends a start signal to the FPGA to begin accepting incoming data over AXILite. The FPGA then uses a simple state machine to accept the next 32 signals as input to the neural network portion. The neural network region generates a generic number of neurons for the input layer and 2 for the output layer. Weights for the network are created ahead of time using the Tensorflow model's weights, converting them to the desired fixed point value,

Table 4.2: Utilizations of the FPGA with respect to the hidden layer size and fixed point size

Neurons	Bits	LUTs (%)	Registers (%)	Muxes (%)	Slices (%)
10	32	7774 (14%)	13914 (13%)	1280 (5%)	3849 (33%)
20	32	14618 (27%)	26159 (25%)	2624 (10%)	6984 (60%)
40	32	FAILURE	FAILURE	FAILURE	12673 (110%)
10	16	3868 (7%)	7124 (7%)	658 (2%)	2592 (22%)
20	16	7393 (14%)	13656 (13%)	1312 (5%)	4646 (40%)
40	16	14141 (27%)	26236 (25%)	2616 (10%)	8404 (73%)
20	4	3124 (6%)	4280 (4%)	184 (1%)	1423 (12%)
40	4	5793 (11%)	7940 (7%)	256 (1%)	2539 (22%)
100	4	13228 (25%)	18919 (18%)	1324 (5%)	7021 (61%)

and writing them out as special arrays of sfixed_bus_array (named WeightsL1 and WeightsL2). The general size in bits of this synthesizable ROM is given by Equation 4.3 where L is the size of the hidden layer, I is the number of inputs to the network, X is the length of the integer portion of the fixed point number, and Y is the fractional length of the fixed point number. The additions of 1 are to account for the bias weight.

$$S = ((X + Y) * (I + 1) * L) + ((X + Y) * (L + 1) * 2) \tag{4.3}$$

It should be noted, however, that the FPGA may not necessarily synthesize this as ROM with an S amount of bits. When using Xilinx Vivado, the utilization is reported as LUTs, registers, multiplexers, and logic slices. Table 4.2 shows the utilization reported by Xilinx Vivado for different network hidden layer sizes and fixed point shapes, with Figure 4.5 showing a 3D bar graph of the utilization percentages.

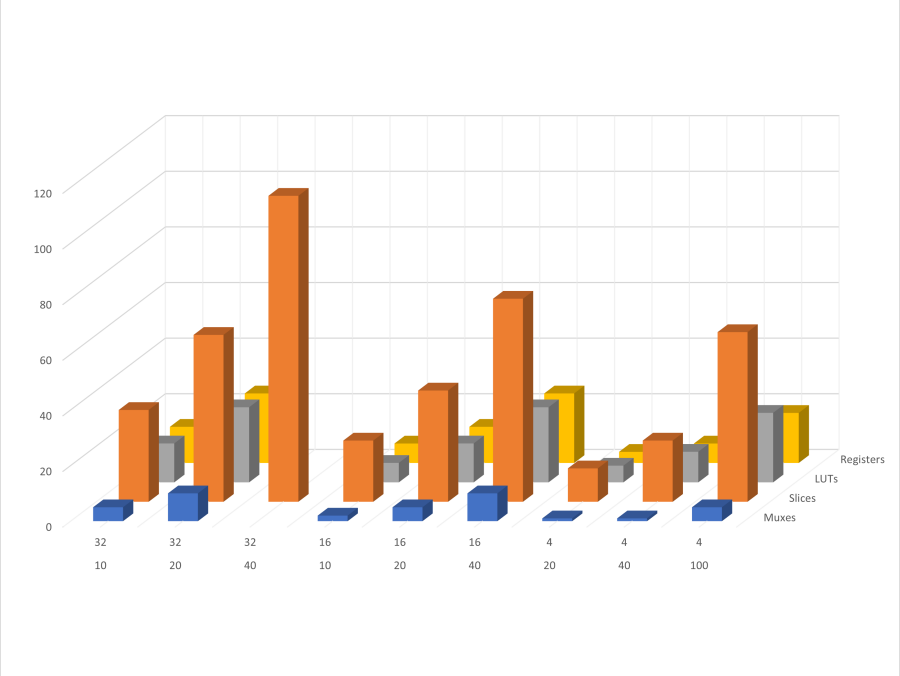

Figure 4.5: Percentage of utilization for each component type with the X-axis showing total bit usage per number above and total number of neurons below.

Interestingly, the utilization appears to follow a linear trend with regards to the size of the network, with only a variation of 1-3%, likely due to larger networks using the same amount of control lines as a previous size with the only difference then being more components used to handle the extra weights present. Utilization appears to be consistent with total number of bits in the system, as the utilizations are very similar for a 20 neuron network with 32 bit weights compared to a 40 neuron network with 16 bit weights. However, as bit sizes become low, it seems to be the case that the utilization increases at a larger rate. In this case, there may be too many weights present to properly partition the logic slices, the consistently largest part of all of the configurations. Though it is likely to differ to some extent between FPGAs and synthesization software, we believe this methodology to be effective in predicting the utilization of any desired network size due to its comparatively linearly comparable nature.

Chapter 5

Dataset Creation and Results

5.1 Prompt-Based Imagined Speech

The dataset contains 5 subjects, ages 20-25, with 4 male subjects and 1 female subject. 4 of the subjects are native Japanese speakers while one of the male subjects is a native English speaker. Each user was instructed to read random combinations of phrases for a total of approximately 8 seconds per combination. The prompts were displayed via Windows command prompt without any other words presented on the screen. The prompts presented would be given by switching between English and Japanese and having the user read one language at a time. For example, a prompt could be "Today is very hot, but it seems that it will rain next week." combined with "The supermarket sells bananas, but they do not have blueberries." with the Japanese version being a similarly translated version: "kyou ha totemo atsui kedo, raishuu ha ame ga furisou" and "suupaa ha banana wo utteiru kedo, buruuberii ga nai". Prompts contained an array of topics in order to more closely mimic natural speech. By using random combinations of prompts, the user would be more likely to focus if the given prompts were somewhat randomized. Each user would be given 60 combinations in total, split up as 30 English and 30 Japanese. With a 128 Hz sampling rate from the Emotiv Flex, this results in approximately 61,440 data points per user. The decision was made to keep the dataset balanced between English and

Japanese as they are both considered to be stimuli-based and inherently do not have balance-related differences between them in the way that the previous dataset did with regards to the resting and speaking states, with speaking being more varied between users than resting. The raw EEG data obtained from the each of the 32 channels from all of the 5 subjects can be seen in the MATLAB plot in Figure 5.1. The vertical lines demarcate changes in subject, and these transitions did not happen in real time; the plot was created after the experiments were complete. Furthermore, the downward trend in amplitude is just coincidence, as the 5th subject, the English native subject, was actually recorded first. Due to them being a potential outlier due to the language difference, they were chosen to be number 5.

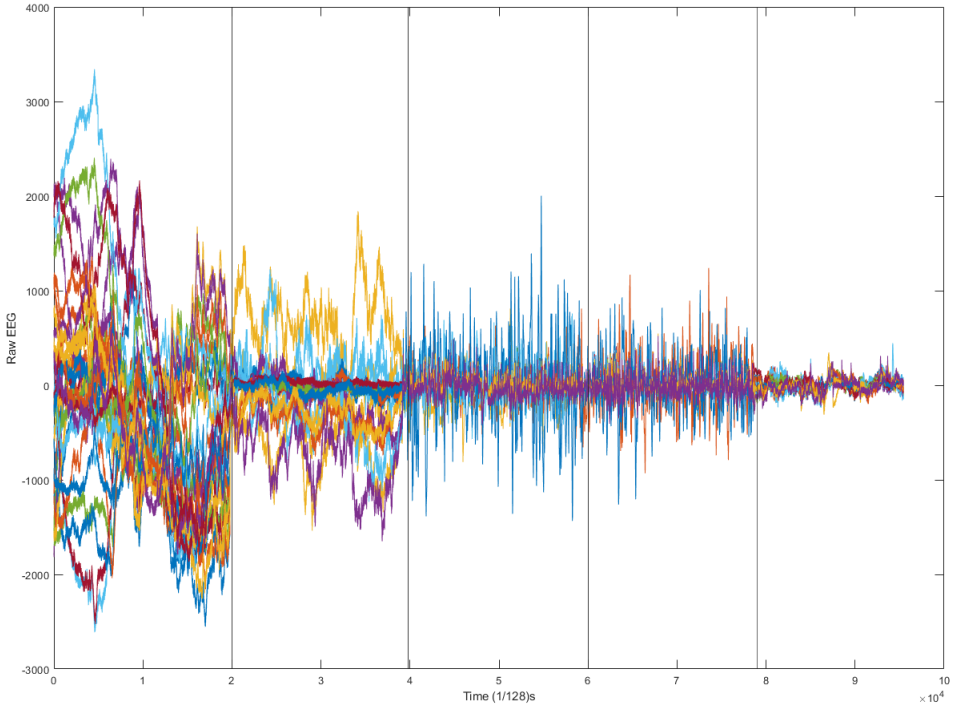

Figure 5.1: EEG Data of all users using the 32 channel Emotiv Flex.

Initially the Emotiv EPOC X was used on the native English speaker subject for its lightweight practicality and comparatively low setup time. However, when

Table 5.1: Table comparing the average test accuracies of imagined speech between EPOC X (15 features) and EPOC Flex (1 feature) for user 5.

	EPOC X	EPOC Flex
random + no regularization	0.7094 ±0.033	0.9538 ±0.021
random + L1 & L2 regularization of 0.0001	0.7583 ±0.054	0.9962±0.001
prompts + L1 & L2 regularization of 0.0001	0.5940 ±0.051	0.8750±0.043

attempting on the same user with the Emotiv Flex, a 32 channel head cap, the increase in accuracy ranges from 10-25% depending on methodology over that of the EPOC X. No other parameters of the training, pre-processing, or data collection methods were changed, just the device used. Taking this into consideration, it was decided that the sacrifice in ease of use with the EPOC X was worth it in exchange for the large increase in accuracy provided by the EPOC Flex. Use of the EPOC Flex requires more time calibrating and adjusting saline levels of the individual sensors as well as a higher price for purchasing. Despite this, the increased accuracy was ultimately considered to be worth more, so subsequent tests were done using the EPOC Flex. Table 5.1 shows the preliminary results when comparing the two for a single user.

Random refers to data obtained from having the user imagined completely random speech whereas prompts refer to speech generated by reading prompts. Based on the results in Table 5.1, the remaining users would have their data taken using the Flex for the markedly higher accuracy. Another important result from these early findings was the feature usage. In both cases, it appears that reducing the amount of features available generally increases the accuracy incrementally. This is likely due to the other features introducing or amplifying any noise present in the data whereas the mean should work to better neutralize it by nature of acting as a low pass filter. A various assortment of feature combinations and their resulting accuracies for the EPOC X and EPOC Flex can be seen in Table 5.2 for random sentences for one subject.

The results of Table 5.2 further reinforce the decision to use the EPOC Flex for

Table 5.2: Table showing the resulting accuracies of various feature combinations.

	EPOC X	EPOC Flex
Mean Only	0.7137 ±0.039	0.9846 ±0.010
Above + Max + Min + Max/Min Related	0.6368 ±0.020	0.9538 ±0.018
Above + Standard Deviation + Variance	0.5897 ±0.022	0.9077 ±0.014
Skewness & Kurtosis	0.5214 ±0.027	0.3692 ±0.076
All 14	0.5940 ±0.019	0.8923 ±0.034
Raw EEG	0.5024 ±0.041	0.9940 ±0.003

the real-time calculations. Interestingly raw EEG data does not work for the EPOC X, but it does work considerably well for the EPOC Flex. Fewer features naturally results in a lower input size and less time spent calculating the pre-processing of the data. This becomes particularly important when considering the sample rate of the Epoc Flex, 128 Hz. This only leaves about 7ms to process the input for the network before the next one is received. With only the mean giving the highest accuracy, there would be less of a delay going from raw EEG data to a full feature set for the neural network input. With this information and the decision to use the EPOC Flex, the dataset was then constructed using the remaining 4 subjects. Henceforth, unless stated otherwise, neural network training and testing results will be discussed with regards to the data of all 5 users as opposed to the single user 5 up until now.

Since the mean becomes such an integral part in having an effect on the classification compared to any other single feature, some further analysis was performed. Figure 5.2 shows the moving averages for each channel over time for three users, and Figure 5.3 shows the same time frame but using the combined average of all channels instead of per channel with the black vertical lines representing changes from one subject to another. Do note that the English and Japanese plots do no occur simultaneously, but they were recorded separately from each other and comprise the same amount of time as each other, so it would be reasonable to show them as if they occurred simultaneously for the sake of visual comparison.

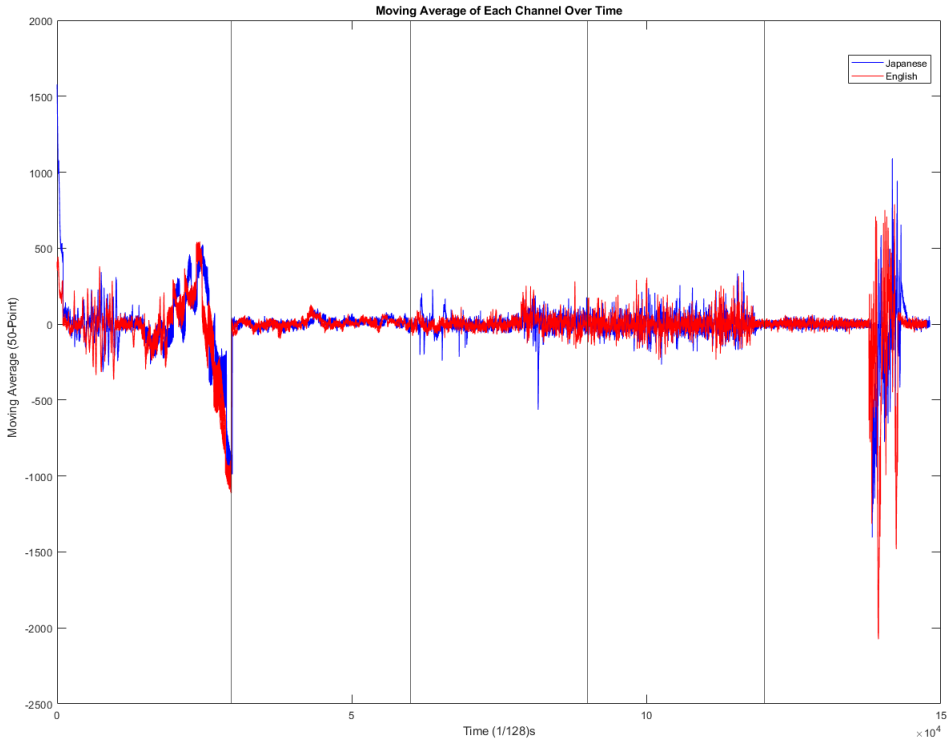

Figure 5.2: Comparison of moving average per channel.

Table 5.3: Table showing the variance per user for the duration of their experiment.

User 1	User 2	User 3	User 4	User 5
1.0426e5	1.3016e4	1.8492e3	3.3944e3	1.9413e4

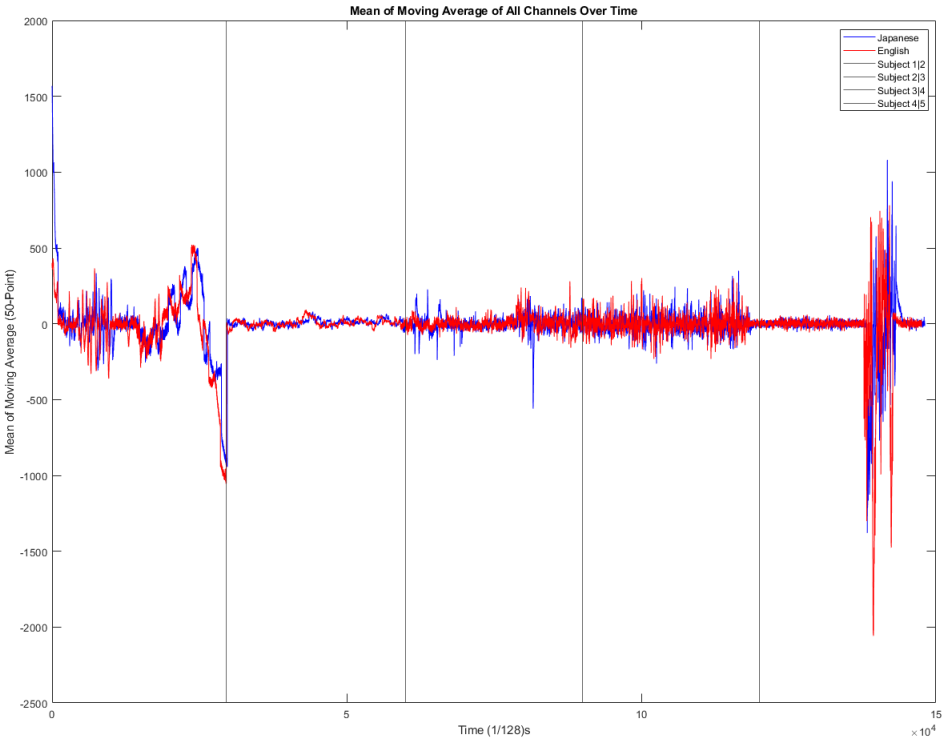

Figure 5.3: Combined average of moving average for all channels.

The first visually distinct part of the plots is the large variation in structure for each subject. Despite each subject having wildly different patterns, their average value is still as expected, 0. Furthermore, this unique variation per subject is also expected based on the initial findings from Section 3.2 that found that different users have different reactions to the same stimuli. While the same exact combinations of prompts were not given to each user, the experiments were done under the same conditions with combinations from a small set of prompts. Essentially, this further reinforces the notion that giving users similar stimuli incurs different reactions between them. Table 5.3 shows the variance per user.

Despite the differing variances between users, the model can successfully converge towards "successful" accuracies (those over 60%) with multiple numbers of users present in the training data and variable sizes for the neural network. However, with higher numbers of users, the training accuracy does not converge towards 100% in the way that it does for training sets containing a single or two users. This may just speak to the comparatively large number of training data samples available when taking a moving average across 60 different 8 second 128Hz samples per person (close to 300,000 per channel in total). It is possible that such a large number of points incurs a level of noise too large for the network to adequately adapt to.

When only using the moving mean as a feature, the input to the network then becomes simply just a moving average over a desired window size per each channel. Table 5.4 shows the test accuracies across various window sizes and their results. The EPOC Flex uses a 128 Hz sampling rate, and the EPOC X uses a 256 Hz sampling rate. The moving average is computed via two different methodologies, named normal moving mean and stepwise moving mean. The difference and rationale for them is as follows. Normal moving mean uses MATLAB's built in movmean() function which truncates the window at edges and returns the same sized matrix used as the input to the function. Stepwise moving mean is calculating the mean over a given window size, but then incrementing the sampling indexes by the window size, rather than by 1. The latter method was considered to encourage more unique data points for the training data. While normal moving mean technically generates a full set of unique, discrete data points, two individual points close to each other become very similar. This could potentially cause the network to train on data too similar to the test data even after being shuffled around for training purposes and may lead to overfitting.

The test data is typically taken as a random splice of individual indices for the whole dataset, but the test data used for the moving mean was taken as contiguous chunks of the dataset to potentially offset issues with test samples being temporally

Table 5.4: Table showing the effects of various moving window sizes for the Emotiv EPOC X and Emotiv EPOC Flex for prompt-based speech from user 5.

Window Size	Moving Mean	Stepwise
10	0.9942 ±0.005	0.8995 ±0.026
20	0.9978 ±0.002	0.8454 ±0.030
50	0.9957 ±0.003	0.6538 ±0.059
100	0.9964 ±0.002	0.6410 ±0.124
200	0.9982 ±0.001	0.4500 ±0.265

Table 5.5: Combinations of subjects and the resulting accuracies amongst themselves and compared to a new subject.

100 Hidden Neuron Network	Test Accuracy	Accuracy for a New Subject
Subjects 1-4	0.7842 ±0.010	0.5111 ±0.047
Subjects 1-3	0.7984 ±0.011	0.5130 ±0.032
Subjects 1-2	0.8570 ±0.017	0.4983 ±0.045
Subjects 2-3	0.8443 ±0.013	0.5083 ±0.032
Subject 2	0.8646 ±0.013	0.5101 ±0.039
Subject 2 (20 hidden neurons)	0.7467 ±0.006	0.5264 ±0.028
Subject 2 (1000 hidden neurons)	0.8919 ±0.005	0.4969 ±0.030

close to training samples. Despite this, it appears to be the case that the moving mean methodology works for any window size below 200 with minimal changes in test accuracy. On the other hand, very small window sizes work best for the stepwise method. With sizes this low, it could be possible that using only the raw EEG data is more effective for training.

5.2 Inter-Personal Uniqueness

As discussed in Section 3.2, the variance between subjects adds a considerably large hindrance towards reaching successful classification accuracies. In fact, this variance appears to be so large that the network is unable to properly classify any data from a user not included in the training set. Table 5.5 shows the differences between various user combinations and the resulting test accuracies among said users and the accuracies of the same model tried on a completely separate user.

Based on these results, it appears to be the case that an individual user presents a new level of variance that the network is not capable of classifying in any reasonable manner given that the accuracies all stay very close to random guessing, 0.5. Though this does not align with the goal of this research, that is not to say it is an entirely unexpected outcome. As the number of users present in the training set increases, the accuracy seems to decrease, but it decreases at a rate that also declines with respect to the number of users present. Perhaps the way to rectify this issue is the addition of more subjects to the dataset as a whole in order to potentially smooth out the overall variance through the introduction of more data points. In Table 5.5, the most potentially promising combinations were those of subjects 2 and 3 tested on subject 4. They reached a reasonably high test (and train) accuracy of 0.8443, but the accuracy when testing on subject 4 resulted in 0.5083. The subject 1 and 2 combination was done to see if combining two separate subjects with highly contrasting levels of variance from Table 5.3 would be more effective when introducing a new subject, but that was not the case as the final accuracy for said subject was 0.4983. Based on these results alone, especially with how low the accuracies are for a new subject, the current methodology and dataset are not sufficient in creating a system capable of working immediately for new users. It should also be noted that the test accuracies for all of these tests were very close to the training accuracies. This could potentially mean the data itself can not effectively be classified by any manner, and accuracies above 90% for prompt-based methodologies are out of reach currently. Despite this, accuracies above 60% are certainly possible when using training and testing data belonging to all users. In this regard, the dataset can be considered successful, as the introduction of new subjects to the training set does not necessarily preclude the network from accurately classifying the output from any of the members in the dataset, regardless of native language.

5.3 Random Imagined Speech

Prompts were used in order to demonstrate the potential to reproduce the findings. However, interesting results were also obtained with regards to having the user randomly create their own imagined sentences. They were instructed to imagine anything as long as it was done using the language instructed to them. For this case, the users were instructed to only imagine sentences 10 times for each language. As a result, these experiments can never be closely replicated, but they can act as a relatively closer comparison to real-time unscripted usage of the system. The first and potentially most surprising result from these experiments is that random imagined speech results in generally higher test accuracies than that of prompt-based imagined speech. Figure 5.4 shows the moving average EEG data for each channel over time.

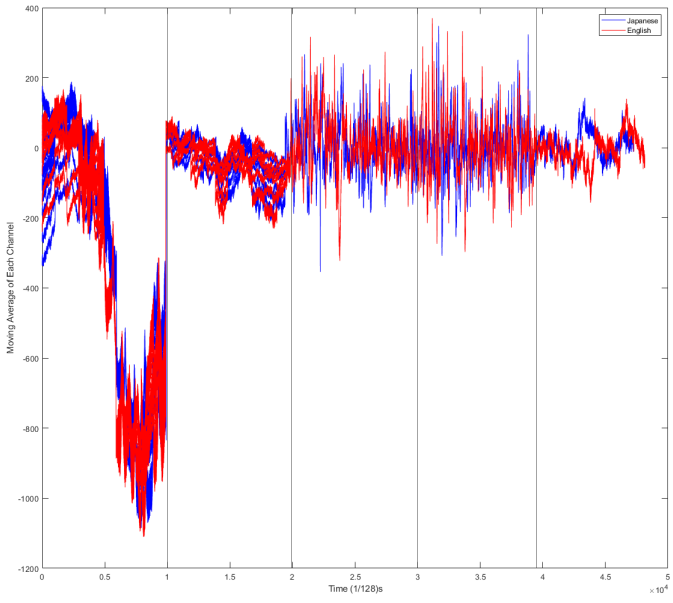

Figure 5.4: Moving average per channel for random imagined speech over time.

As observed previously, the difference in variance between users continues to be high, but these variance levels also appear to be relatively consistent to each user despite being part of a new experiment as shown in Table 5.6.

Table 5.6: Table showing the variance per user for the duration of their experiment for random imagined speech.

User 1	User 2	User 3	User 4	User 5
6.3909e3	3.9151e3	2.4194e3	2.2040e3	4.9002e4

Table 5.7: Combinations of subjects and the resulting accuracies amongst themselves and compared to a new subject for random imagined speech.

Combination (Neurons)	Test Accuracy on Set	Accuracy for New Subject (% Included)
Subjects 1-5 (1000)	0.8204 ±0.005	N/A
Subjects 1-5 (5000)	0.8141 ±0.003	N/A
Subjects 1-4 (1000)	0.7834 ±0.012	0.6073 ±0.028 (50%)
Subjects 1-4 (1000)	0.7905 ±0.006	0.4463 ±0.032 (25%)
Subjects 1-4 (100)	0.7849 ±0.014	0.5331 ±0.021 (50%)
Subjects 1-4 (100)	0.8167 ±0.008	0.4888 ±0.044 (25%)
Subjects 1-4 (20)	0.6816 ±0.011	0.5331 ±0.051 (50%)
Subjects 1-4 (20)	0.6860 ±0.011	0.4003 ±0.045 (25%)
Subjects 1-3	0.8288 ±0.005	0.4972 ±0.039
Subject 3	0.8714 ±0.010	0.5307 ±0.048

Except for Subject 5, the variances fall within values proportional to those from Table 5.3, but they are lower for subjects 1, 2, and 4. A few tests were performed to examine random imagined speech's ability to classify a brand new subject, similar to the previous experiment. The results of which can be seen in Table 5.7. A separate methodology of including some of the new user's data as part of the training data via transfer learning after the initial test was performed was also attempted and is included. However, this does not reach successful accuracies on the target user either.

These were performed using the normal moving mean methodology to prepare the training data. The first column represents the combinations of subjects used in the training data as well as the number of neurons in the hidden layer; the second column represents the accuracy on the test data taken from column 1's users; and the third column represents the accuracy on a new user if applicable with the percentage of said user's data used in the training data. It can be observed that the accuracy for a new user can only barely reach above 60% only when taking into account the

fact that their data is already included in the training set. Of course, the test data for said user is never included in the training data set. The primary result from this collection of experiments is that both prompt-based and random speech are inherently differentiable and able to be properly classified theoretically. However, when trying to rectify the amount of variance present between users through the use of a general purpose network, this currently seems to be not feasible.

5.4 Real-Time Results

Ultimately, the current setup does not adequately allow for a real-time implementation in hardware nor software. For hardware, the Python program spends too much time to prepare the packets to send over the serial port (which would be 32*4 byte-sized packets every 7 milliseconds). When taking the headcap output directly from the Python program and feeding it into the network using Tensorflow's predict() function for each sample from the headset, the average time spent per function is 14ms, unfortunately above the time needed to get the next output from the headcap, which is approximately 8ms (1/128s). In order to best replicate real-time performance, a file containing raw EEG data taken at a time separately from when the training data was used and given to the network along with its corresponding labels through the Tensorflow evaluate() function.

Test Accuracy refers to the accuracy obtained from data removed from the training data but still part of the same experiments. Target refers to the number of the user whose data will be used as the evaluation data. The "Real-Time" column refers to accuracies obtained from said user's data. For each iteration, the training data contained data from two separate recording sets, with the third recording set (20 prompts per set) used as the "real-time" evaluation data. Despite being able to successfully adapt to the two sets for training and testing purposes, the model does not currently meet successfully accuracy levels for a third set except for barely passing

Table 5.8: Various results of the "real-time" approach to evaluating the model.

Users	Target	Test Accuracy	"Real-Time" Accuracy	Notes
1-5	1	0.9095 ±0.029	0.5540 ±0.034	Raw EEG
1-5	2	0.8817 ±0.036	0.5502 ±0.019	Raw EEG
1-5	3	0.8792 ±0.031	0.5347 ±0.013	Raw EEG
1-5	4	0.9063 ±0.030	0.5477 ±0.022	Raw EEG
1-5	5	0.9418 ±0.038	0.5049 ±0.027	Raw EEG
1-5	1	0.9136 ±0.024	0.4915 ±0.036	Moving Mean
1-5	2	0.9546 ±0.024	0.5071 ±0.038	Moving Mean
1-5	3	0.9008 ±0.026	0.4998 ±0.048	Moving Mean
1-5	4	0.9126 ±0.028	0.5447 ±0.017	Moving Mean
1-5	5	0.9465 ±0.021	0.4958 ±0.066	Moving Mean
1	1	0.9520 ±0.009	0.5118 ±0.034	Raw EEG
2	2	0.8279 ±0.020	0.5241 ±0.047	Raw EEG
3	3	0.8549 ±0.008	0.5256 ±0.033	Raw EEG
4	4	0.8516 ±0.009	0.5379 ±0.023	Raw EEG
5	5	0.9316 ±0.018	0.4615 ±0.038	Raw EEG
1	1	0.9411 ±0.019	0.4861 ±0.027	Moving Mean
2	2	0.9896 ±0.003	0.6042 ±0.025	Moving Mean
3	3	0.9893 ±0.006	0.5421 ±0.044	Moving Mean
4	4	0.8970 ±0.011	0.5122 ±0.033	Moving Mean
5	5	0.9980 ±0.002	0.4648 ±0.032	Moving Mean

in one case for user 2 by reaching above 60% accuracy. There are multiple reasons as to why this may happen. Perhaps EEG data is truly so unique that it is very difficult to generalize it for a large sized amount of entirely new data, even if similar stimuli are being presented to the user. Furthermore, when conducting the experiments, there is also potential that the users' moods or level of focus fluctuated throughout. For training points close to each other temporally, this may not be an issue, but when considering points temporally far from each other, this evidently becomes a much larger issue. While training used L1 and L2 regularization values of 0.001, it still does not seem to be the case that regularization could assist in this case. Visually looking at Figures 5.2, 5.3, and 5.4 further shows that even when constrained to a single user, there tends to be a large fluctuation in the values as time progresses, especially for User 1.

Chapter 6

Conclusion and Future Work

While the desired real-time classification results were not as hoped, there is still much that can be learned from this research. The first is that EEG data is inherently differentiable and classifiable, but it is not a simple task. Despite having very successful accuracies when training and testing the network on a given dataset, the network is unable to adapt to any new user or data obtained with from an existing user at a different time period (even if only approximately 30 seconds separated between recording sessions). The best way to rectify such an issue would be to record and include more EEG data as part of the training data. While a network has more success in classifying data belonging to the same user used in training data compared to a training dataset containing all users, theoretically a large enough set of users balances out this issue to allow for potentially completely generalized models that do not need to train on a new user. However, unfortunately, this is not yet the case with this research and this dataset. The dataset does include the specific prompts given to each user, so those interested may also attempt discrete imagined sentence classification if desired.

With regards to handling the data in real time, a large change with the device used may be in order as well. The Emotiv series of devices were initially chosen due to their ease of use physically for the user, but the proprietary software and data streaming involved adds extra time needed to either calculate the output in software or prepare

said output to be sent to an FPGA. Perhaps future work could use a different EEG measurement device that allows for direct access to the electrodes for the FPGA to skip the current Python middleman program. Though since a moving mean can function satisfactorily in predicting the imagined language, perhaps a software-based approach that calculates a moving mean but does not give the input to the network every time a new input is received could work as well to potentially mitigate the issue of adding unnecessary delay in computing the output.

We believe this data to be valuable in creating a model capable of guessing a user's imagined language in the future, but there needs to be more data added to it in order to successfully do so. Since temporality appears to be a big component of the dataset, perhaps a completely different type of deep learning model aimed more towards temporal data such as an echo state network or long short-term memory network would be more successful. If the raw values are less important than the trend of the data itself, it would stand to reason that a network capable of learning these patterns would perform better. The current implementation only handles temporality with the use of a moving average to incorporate values outside of the current time period, but it might be the case that these values do not hold enough of the information needed to properly classify said data. Despite this, human imagined language is differentiable and could certainly some day be used with a sophisticated enough setup to classify language or potentially even individual words with enough time.

Bibliography

[1] D. A. Moses, S. L. Metzger, J. R. Liu, G. K. Anumanchipalli, J. G. Makin, P. F. Sun, J. Chartier, M. E. Dougherty, P. M. Liu, G. M. Abrams, A. Tu-Chan, K. Ganguly, and E. F. Chang, "Neuroprosthesis for decoding speech in a paralyzed person with anarthria," *New England Journal of Medicine*, vol. 385, no. 3, pp. 217–227, 2021. [Online]. Available: https://doi.org/10.1056/NEJMoa2027540

[2] S. Zhao and F. Rudzicz, "Classifying phonological categories in imagined and articulated speech," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 992–996.

[3] T. Ball, M. Kern, I. Mutschler, A. Aertsen, and A. Schulze-Bonhage, "Signal quality of simultaneously recorded invasive and non-invasive eeg," *NeuroImage*, vol. 46, no. 3, pp. 708–716, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1053811909001827

[4] D. Smith, "An analysis of similarities between japanese and english," *The Classic Journal*, 2020. [Online]. Available: https://theclassicjournal.uga.edu/index.php/2020/11/11/the-unrealized-universality-of-phrase-structure/

[5] K. Ohata, "Phonological differences between japanese and english: Several potentially problematic areas of pronunciation for japanese esl/efl learners," *Asian EFL Journal*, 2000.

[6] L. S. Tapia, A. Gomez, M. Esparza, V. Jatla, M. Pattichis, S. Celedón-Pattichis, and C. LópezLeiva, "Bilingual speech recognition by estimating speaker geometry from video data," 2021.

[7] G. Krishna, Y. Han, C. Tran, M. Carnahan, and A. H. Tewfik, "State-of-the-art speech recognition using eeg and towards decoding of speech spectrum from eeg," 2020.

[8] N. Hashim, A. Ali, and W.-N. Mohd-Isa, *Word-Based Classification of Imagined Speech Using EEG*. Springer, 02 2018, pp. 195–204.

[9] A. Torres-García, C. A. Reyes-Garcia, and L. Villaseñor-Pineda, "Toward a silent speech interface based on unspoken speech," in *BIOSIGNALS 2012 - Proceedings of the International Conference on Bio-Inspired Systems and Signal Processing*, 02 2012.

[10] L. Sun, B. Jin, H. Yang, J. Tong, C. Liu, and H. Xiong, "Unsupervised eeg feature extraction based on echo state network," *Information Sciences*, vol. 475, 09 2018.

[11] A. Balaji, A. Haldar, K. Patil, T. S. Ruthvik, V. CA, M. Jartarkar, and V. Baths, "Eeg-based classification of bilingual unspoken speech using ann," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2017, pp. 1022–1025.

[12] S. Datta and N. V. Boulgouris, "Recognition of grammatical class of imagined words from eeg signals using convolutional neural network," *Neurocomputing*, vol. 465, pp. 301–309, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221012194

[13] J. M. Correia, B. Jansma, L. Hausfeld, S. Kikkert, and M. Bonte, "Eeg decoding of spoken words in bilingual listeners: from words to language invariant semantic-conceptual representations," *Frontiers in Psychology*, vol. 6, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpsyg.2015.00071

[14] G. Krishna, C. Tran, Y. Han, M. Carnahan, and A. H. Tewfik, "Speech synthesis using eeg," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 1235–1238.

[15] M. A. Gernsbacher and M. P. Kaschak, "Neuroimaging studies of language production and comprehension," *Annual Review of Psychology*, vol. 54, no. 1, pp. 91–114, 2003, pMID: 12359916. [Online]. Available: https://doi.org/10.1146/annurev.psych.54.101601.145128

[16] "Zybo z7 - digilent reference," 2020. [Online]. Available: https://digilent.com/reference/programmable-logic/zybo-z7/start

[17] "Emotivpro gitbook," 2021. [Online]. Available: https://emotiv.gitbook.io/emotivpro-v3/notes-on-the-data/dc-offset