NEURAL NETWORK MODELS FOR NEUROPHYSIOLOGY DATA

by

Bryan Jimenez

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Electrical and Computer Engineering

School of Electrical and Computer Engineering

West Lafayette, Indiana

December 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Joseph G. Makin, Chair

School of Electrical and Computer Engineering

Dr. David I. Inouye

School of Electrical and Computer Engineering

Dr. Ronnie B. Wilbur

School of Speech, Language, and Hearing Sciences

Approved by:

Dr. Dimitrios Peroulis

ACKNOWLEDGMENTS

Special thanks to my advisor, Professor Joseph G. Makin, for providing guidance throughout my journey as a Master's student. Also, special thanks to my lab mate, Miss Ganga Meghanath. Many of the results in this thesis were the product of a collaboration with Miss Ganga Meghanath. These include Wave2Vec2 (Section 2.2.1), MarianMT (Section 2.2.1), RNNF (Section 3.2.1), and Neural r-RoBERTa (Section 3.2.1). Where possible, I have tried to distinguish our contributions, but that has not always been possible. Additionally, to compare the performances of models decoding ECoG data into English text with the original work done by Professor Makin, I also include result from RNN model with cross attention which was solely Miss Meghanath's work. Finally I would like to thank the ECE department for providing the necessary resources to be able to perform these projects.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

BERT	Bidirectional Transformers
CER	Character Error Rate
ECoG	Electrocorticography
fMRI	Functional Magnetic Resonance Imaging
GRU	Gated Recurrent Unit
iEEG	intracranial Electroencephalography
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
RNN	Recurrent Neural Network
RoBERTa	Robustly Optimized BERT
WER	Word Error Rate

ABSTRACT

Over the last decade, measurement technology that records neural activity such as ECoG and Utah array has dramatically improved. These advancements have given researchers access to recordings from multiple neurons simultaneously. Efficient computational and statistical methods are required to analyze this data type successfully. The time-series model is one of the most common approaches for analyzing this data type. Unfortunately, even with all the advances made with time-series models, it is not always enough since these models often need massive amounts of data to achieve good results. This is especially true in the field of neuroscience, where the datasets are often limited, therefore imposing constraints on the type and complexity of the models we can use. Not only that, but the Signal-to-noise ratio tends to be lower than in other machine learning datasets. This paper will introduce different architectures and techniques to overcome constraints imposed by these small datasets. There are two major experiments that we will discuss. (1) We will strive to develop models for participants who lost the ability to speak by building upon the previous state-of-the-art model for decoding neural activity (ECoG data) into English text. (2) We will introduce two new models, RNNF and Neural RoBERTa. These new models impute missing neural data from neural recordings (Utah arrays) of monkeys performing kinematic tasks. These new models with the help of novel data augmentation techniques (dynamic masking) outperformed state-of-the-art models such as Neural Data Transformer (NDT) in the Neural Latents Benchmark competition.

1. INTRODUCTION

In Neuroscience, the amount of data available is significantly smaller compared to other machine learning areas such as NLP and Computer Vision. Table 1.1 shows the differences in the ratio of training data per parameter in three different models; RoBERTa (NLP), EfficientNet-B7 (CV), and NDT-2 (Neuroscience).

Table 1.1. Table Showing Ratios of Data per Parameters in Different ML fields

Model	**Parameters**	**Training Data**	**Ratio**
RoBERTa base	125M	496GB (Books+Wiki)	3.9680
EfficientNet-B7	66M	150GB (Imagenet)	2.2727
NDT-2	0.07M	0.016GB (area2_bump)	0.2286

NDT-2 is a state-of-the-art neuroscience model used for accurately capturing dynamical structure from neural population activity. From table 1.1, we can conclude that the amount of data per parameter for NDT-2 is at least 10 times smaller than RoBERTa base and EfficientNet-B7. This poses a challenge when designing models since a lot of these models require large amounts of data to achieve state-of-the-art results. This thesis will introduce different architectures and techniques to overcome constraints imposed by these small datasets. There are two major experiments that we will discuss. (1) We will strive to develop a speech prosthesis for participants with who lost the ability to speak by building upon the previous state-of-the-art model for decoding neural activity (ECoG data) into English text. The first problem we will address is that it cannot generalize well when given ECoG data from unseen sentences. We will be addressing this problem by fine-tuning pretrained models using ECoG data. The second problem we will address is that we cannot perform well for all participants. We theorize that we will be able to solve this problem by utilizing the attention mechanism to keep track of the relationships contained in ECoG data due to the attention mechanism's ability to keep track of long sequences [1]. (2) We will discuss using time-series models to analyze and predict neural behaviors of recordings from monkeys' neural activity. We theorize that by accurately predicting neural behavior, we will be able to develop a deeper understanding of how the brain works.

2. DECODING ECOG DATA INTO ENGLISH TEXT FROM PARTICIPANTS

2.1 Introduction

In recent years, the field of neuroscience has extended its focus from neural correlates of speech processing to decoding speech directly from human neural activity. The methods most commonly used to record human neural activity are fMRI, EEG, Utah Array, and ECoG. Functional magnetic resonance imaging or fMRI is a common non-invasive imaging technique that records the changes in blood flow and oxygen levels in the brain [2]. Even though this technique has previously been used successfully for neural correlates of speech because of its high spatial resolution, it has some drawbacks for the task at hand. One of the significant drawbacks is the sampling rate. Hemodynamic response or how fast the blood flow changes dictates fMRI's sampling rate (up to 2 Hz) [2]. The low sampling rate makes it difficult for fMRI to capture the speech dynamics, which can change in 5-20 ms [2]. Another drawback is that it does not directly record the neuron's activity, which means it is recording blood flow and not the neuron's action potential, which can add to the uncertainty when decoding speech. Another method commonly used is electroencephalogram or EEG, a non-invasive procedure that records the aggregate activity of 10s-100s of thousands of neurons [3]. This method overcomes one of the drawbacks mentioned for fMRI, but it offers a lower spatial resolution and a frequency bandwidth up to 100 Hz [3]. Studies have shown that most data correlated to speech lie within the high-gamma frequency range (70-180 Hz) [2][4]. Therefore, EEG is not ideal since it can only access a fraction of information contained in high-gamma. The last measuring technique is electrocorticography or ECoG. ECoG is similar to EEG, except it is an invasive procedure requiring an electrode array placed on the brain [4]. This procedure solves many shortcomings that make EEG ineffective for speech decoding. Due to its proximity to the area of interest in the brain, ECoG offers high spatial resolution and a high-frequency range (200 Hz or higher). These reasons make ECoG data the ideal choice for real-time speech decoding [2]. Utah array is a four-by-four-millimeter that contains 100 electrodes [5]. In contrast with ECoG, each electrode in the Utah array allows us to measure a single neuron instead of a population of neurons and excels in obtaining stable

and long-term neural recordings [5]. The downside of Utah arrays as compared to ECoG is that participants only are allowed to get an implant if they are paralyzed [6]. Therefore making this type of data incompatible for real-time neural activity to English text.

Some of the work done with ECoG includes decoding it into speech and text. The following paper shows state-of-the-art results for decoding ECoG into text. In [7], the researchers use an RNN-based encoder-decoder model to decode neural activity into English text. They managed to get down to a 3% word error rate (WER) from data consisting of repetitions of 50 sentences. Not only that, but the authors also show that using transfer learning across participants can help overcome some of the limitations imposed by the dataset.

The following shows state-of-the-art results for ECoG to speech. In [8], the authors synthesize speech by taking advantage of human cortical activity with kinematics and sound representation encoded within. This paper also used an RNN-based approach where they would first decode neural activity into the kinematics representation of the mouth and throat. Then use the kinematics representation extracted to synthesize speech.

Some drawback of translating ECoG into speech in real-time is a phenomenon called delayed auditory feedback (DAF). DAF can seriously affect the speech patterns of people. Some side effects of experiencing DAF can include articulated changes such as increase length and intensity of utterance [9]. This can negatively impact the performance of these models. For ECoG to text, even if there is a small delay during the decoding process it will not affect the participant's ability to articulate. Because we want to build a speech prosthesis that can work on anyone regardless of the degree of speech impediment and that can work in real-time, in this chapter, we will be translating from ECoG to text. Here are the problems that we hope to address from the previously presented state-of-the-art experiment for ECoG to text. The first problem we will address is the fact that it is not able to generalize well. We will be addressing this problem by fine-tuning pre-trained models using ECoG data.

The second problem we will try to address is that it does not work perfectly for all subjects. We theorize that we will be able to solve this problem by utilizing the attention mechanism to keep track of the relationships contained in ECoG data due to the attention mechanism's ability to keep track of long sequences [1]. This feature is handy for the task since ECoG data has a high spatial and temporal resolution.

We will test on the dataset used in [7] because it is publicly available and will allow us to make precise performance comparisons.

2.2 Methods

In this section, we will discuss all the different methods that we tried. This includes models, training methods, optimization methods, and evaluation methods.

2.2.1 Dataset

The dataset we will use to test the performance of our models is the dataset from the study [7]. We will use this dataset because it is publicly available and will allow us to make precise performance comparisons. This dataset contains ECoG recordings of participants undergoing monitoring for epilepsy reading aloud sentences from picture descriptions and MOCHA-TIMIT; Picture descriptions contain 30 sentences and 125 unique words, while MOCHA-TIMIT contains 460 sentences and 1800 unique words, but researchers split MOCHA-TIMIT into blocks of 50 sentences due to the large number of sentences it contained (MOCHA-1, MOCHA-2, Etc.). The recordings typically take place in "blocks" of 5-10 minutes to prevent patient fatigue. Participants are in the hospital for approximately one week, which limits data collection.

The study [7] considers data from blocks that contain at least three recordings. Consequently, Makin et al. made the major claims of their paper on the basis only of the MOCHA-1 blocks. More importantly, this means that the same set of sentences (albeit with different neural data) was used in the training and testing sets during their experiments. We instead took a different approach. We found that when only using training and testing sets the models suffer from high variance. Therefore, the approach we took was to split the blocks into training, evaluation, and testing sets. This allows us to utilize the evaluation set for early-stopping.

In our second set of experiments, we attempt to show generalization beyond MOCHA-1. We use all the data available for the task of generalizing; this includes all the blocks associated with MOCHA-TIMIT and picture descriptions. We hold out all MOCHA-1 blocks

to use to evaluate the model performance and the remaining blocks (MOCHA-TIMIT, picture descriptions) for fine-tuning the pre-trained models to achieve generalization. The study [7] does not cover or include results related to this generalization task.

2.2.2 Models

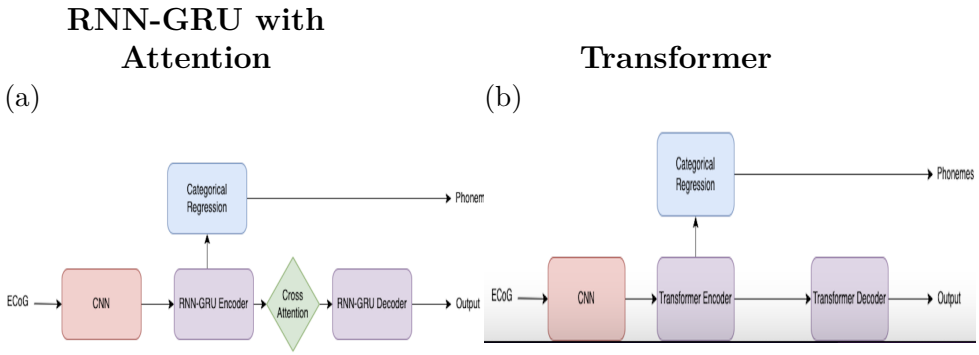

Figure 2.1. Figure 2.1.a shows a high-level representation of the attention RNN-GRU model used to decode text from ECoG data. Figure 2.1.b shows a high-level representation of the transformer model used to decode text from ECoG data.

GRU

Despite its name, long short-term memory (LSTM) struggles with sequences longer than 50 samples due to information decay [10]. The ECoG sequences in our study likely contain information up to 200 Hz [2] and are on the order of one second in length. Thus, a typical ECoG sequence contains about 200 samples. [7] circumvented this difficulty by downsampling the sequences by a factor of 12 with their convolutional input layer. We hypothesize that some of their errors in decoding are due to this severe downsampling. Therefore, we propose the following. We propose adding an attention module between the encoder and the decoder. In this case, instead of relying on the last hidden state, which compresses all the necessary information of an input into a fixed-length vector, we can extract the hidden states at each time step and feed it into the attention mechanism [11]. The attention mechanism will

preserve the information from all the hidden states and use the information for making predictions.

Auxiliary classifiers are a common technique used in deep networks such as neural architectural search (NAS) which are used to automate the process of developing artificial neural network (ANN) architectures. By adding auxiliary classifiers, we can increase the gradient flow, increase stability during training, and encourage discrimination during earlier stages of models [12]. In [7], the auxiliary classifier predicts MFCCs. Due to medical privacy policies, MFCCs from participants are unavailable since audio files are considered personally identifiable information. Therefore, we propose to replace MFCC with phonemes for our auxiliary classifier.

Finally, due to the current dataset for evaluating this model (MOCHA-1) having as few as three blocks available per participant, we propose swapping RNN-LSTM with RNN-GRU. The critical difference between RNN-GRU and RNN-LSTM is the complexity of the two models. Because the RNN-GRU has fewer gates than the LSTM, it has a lower computational complexity [13]. Therefore, if the dataset is small, then GRU is preferred over LSTM.

Transformer

Despite modifying the original architecture from [7], both [7] and our implementation of RNN with attention get their best results after downsampling the sequences by a factor of 12 with their convolutional input layer. As mentioned in the last section, we hypothesize that some decoding errors are due to this severe downsampling. Therefore, we propose to replace the RNN-LSTM in their network with a fully attention-based network from [1]. In attention networks, O(1) operations separate any two elements of the input sequence instead of the O(n) operations between two elements n steps apart in a recurrent neural network. Not only that, but due to this, parallelization is possible. More importantly, the attention mechanism does not have the same limitations as the RNN-LSTM model, where the model relies on only the last hidden state, which compresses all the necessary information of an input into a fixed-length vector risking the loss of data in the process. Attention allows models to keep information despite the input size without risking information decay.

In machine translation tasks, embedding layers embed indices associated with different words or pieces. Because ECoG measures neurons' voltage (action potential), the embedding layer no longer applies and is removed from the model.

As mentioned in the previous section, we use an auxiliary classifier that predicts phonemes. By adding auxiliary classifiers, we can increase the gradient flow, increase stability during training, and encourage discrimination during earlier stages of models [12].

Wave2Vec2

In [7], just 50 sentences were used to learn the structure of English, which probably helped with decoding those 50 sentences, but it is fatal for generalization. We hypothesize that generalization would be improved if the (implicit) language model were learned from generic English corpora many orders of magnitude larger than the ECoG and text data which motivates using a pre-trained model for the decoder. We thought that Wave2Vec2 was an obvious choice for this due to its similarities when comparing it with the state-of-the-art model for ECoG to text. The Wave2Vec2 model contains convolutions in front, but swaps out the RNN encoder and decoder with a transformer-based encoder and decoder [7] [14]. For this task, we fine-tuned the model by optimizing the encoder portion of wave2vec using all the data available except for MOCHA-1, which was the held-out and used for evaluating how well the model could generalize.

MarianMT

As the previous section stated, we hypothesize that generalization would be improved if the (implicit) language model were learned from generic English corpora many orders of magnitude larger than the ECoG and text data, encouraging the use of a pre-trained model for the decoder such as MarianMT [15]. We put the transformer encoder from the previous section with the MarianMT decoder trained on the University of Helsinki multilingual dataset trained on translating from French to English.

We want to test generalization, but we recognize that this task may be too difficult given how little data we have. Therefore, we will constrain the possible outputs using a beam search

with bi-grams. Based on our current dataset, bi-grams kept track of all possible transitions a word piece could make. We would use this information to mask invalid transitions, making it unavailable when extracting the top-k results.

2.2.3 Optimization

To optimize these models, we did semi-manually tuning. By semi-manually tuning, we combine two popular techniques: manual hyperparameter tuning and grid-search. The intuition is that manual tuning should give us a rough idea of the hyper-parameter boundaries that we might want to search within, as well as help us prune hyper-parameters that have little to no effect on the actual performance of the model. Then we can use a grid search within these boundaries to find what we presume to be the optimal parameter set and also saves us time in the long run since we can reduce the search space that the grid search uses for each hyper-parameter, reducing the total amount of time to get the final results. Of course, this approach is even more attractive because the ECoG datasets are small. For example, the mocha-1 dataset for subject EFC403 only contains 150 sentences, with 50 being unique, which allows us to train the model within 10 minutes.

2.2.4 Evaluation

The metrics chosen to evaluate the models' performance are word error rate (WER) and character error rate (CER). We use CER to evaluate generalization results because the models predict in terms of word pieces and not in terms of words. Instead, when predicting repeated sentences, we predict in terms of words and not word pieces; therefore, we use WER for repeated sentences result. These two evaluation methods are a type of edit distance that calculates the score by measuring the minimum number of insertions, deletions, and substitutions required to make both sentences equivalent. To calculate the number of deletions and insertions, we use the Wagner-Fischer algorithm.

Formula (WER and CER)

The following formula shows how WER and CER are calculated:

$$\text{WER} = \frac{S + D + I}{N}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions and, N is the number of words in the reference.

Similarly, CER is calculated with the same exact formula except we use characters instead of words as a means of comparison.

$$\text{CER} = \frac{S + D + I}{N}$$

2.3 Experiments

In this section, we will review some of the experiments we designed to test the models we mentioned in the previous section.

2.3.1 Repeated Sentences

We took the same approach as in [7] to train and evaluate results on datasets containing repeated sentences. We only consider data from blocks that contain at least three recordings since, at the very least, we need one block for training, one block for testing, and one block for evaluation. The only dataset that fits that criteria for all participants (400, 401, 403) is MOCHA-1. Therefore we only use recordings associated with MOCHA-1 to evaluate our performance in decoding text from ECoG data. For this task, we will use the Transformer, and Attention-RNN-GRU described in the methods section.

2.3.2 Generalization

For generalization, we use all the data available for generalizing to sentences outside the repeated dataset; this includes all the blocks associated with MOCHA-TIMIT and picture

descriptions. We use MOCHA-1 as the held-out block and the remaining blocks for fine-tuning the pre-trained models to achieve generalization, which means that the models do not see any of the sentences in MOCHA-1 during the fine-tuning phase. The models that we will be using are the Wave2Vec2 and the MarianMT model described in the methods section. The paper [7] does not cover or include results related to this generalization task.

2.4 Results

2.4.1 Repeated Sentences

We evaluated three different models on the MOCHA-1 dataset. Table 2.1 displays the WER and CER for each participant.

Table 2.1. Mocha-1 Performance for all Models

Model	Participant	Avg. WER	Best WER
Transformer	400	0.58006	0.5507
GRU	400	0.53908	0.5194
LSTM [7]	400	0.5	N/A
Transformer	401	0.19124	0.1633
GRU	401	0.11403	0.0
LSTM [7]	401	0.03	N/A
Transformer	403	0.76254	0.7238
GRU	403	0.67916	0.6229
LSTM [7]	403	0.8	N/A

Unfortunately, the attention-based models could not outperform the LSTM used in [7]. We disproved the original hypothesis that decoding errors are due to this severe downsampling. Despite modifying the original architecture from [7], both [7] and our implementation of RNN with attention get their best results after downsampling the sequences by a factor of 12 with their convolutional input layer. These networks outperform the transformer even though we reduced the downsampling by half. A possible explanation is that there was not enough data. Transformer-based models tend to be data-driven. The best results recorded for these methods had to train on 100s, sometimes 1000s of gigabytes of data. In our case,

we had a small amount of data available for this task. A quick example of this problem is for participant 403; we only had three data blocks. Each block consists of fifty sentences. In the mocha-1 dataset, there are fifty unique sentences; therefore, each recording corresponded to one of the unique sentences in mocha-1. Because of this, for participant 403, we use one block for training, validation, and testing, which causes these models trouble, especially during the evaluation process.

2.4.2 Generalization

Unfortunately, generalizing using pre-trained models failed when using both MarianMT and Wave2Vec2. Even with equipping the beam-search, the generalized model would predict the same sentence repeatedly, making the results invalid. The average word error rate and the character error rate is around one. Once again, a possible problem is the lack of data. It can be challenging to fine-tune these large, complex models with such small data. This experimentation on creating generalized models for ECoG to text needs a deeper look.

2.5 Future Work

There are different directions this study can take. One of the obstacles that need to be addressed is the limited amount of neural data for generalization. This can be addressed in two different ways. First, collect more data from participants, or second, create a model to splice together data to create new variations of sentences.

2.6 Conclusion

In conclusion, the results from the repeated sentences experiment were satisfactory. On the MOCHA-1 set, the transformer managed to outperform the current state-of-the-art model on participant 403, match the performance of participant 400, and underperformed for participant 401. These results contradicted our initial hypothesis that we could increase prediction accuracy by reducing the downsampling of ECoG performed by the state-of-the-art model. The other unexpected result is that we outperformed the RNN-LSTM on the participant with the least amount of data recorded for MOCHA-1 while simultaneously not

being able to perform on par with the LSTM model on participant 401, which contained the most amount of data for MOCHA-1. We expected the opposite result since the transformer tends to be more data-driven than the smaller, simpler RNN models. Finally, even by adding the attention mechanism to our RNN model, we get our best results by downsampling by the same amount as the state-of-the-art model. The main limiting factor for the performance of these models is the amount of data available. Finally, the generalization experiment did not yield acceptable results. We could not obtain significant results without the use of transfer learning. We hypothesize that the amount of data was insufficient to fine-tune the pre-trained models (MarianMT and Wave2Vec2) correctly, which is an area that requires further investigation and experimentation.

3. AUTOREGRESSIVE MODELS OF NEURAL POPULATION ACTIVITY

3.1 Introduction

The standard paradigm for investigating the neural basis of complex motor tasks is based on recording from the motor areas of the brains of monkeys while they perform reaching tasks. The Utah array is one of the most popular methods of recording neural activity from monkeys. This array is a four-by-four-millimeter that contains 100 electrodes [5]. In contrast with ECoG, each electrode in the Utah array allows us to measure a single neuron instead of a population of neurons and excels in obtaining stable and long-term neural recordings [5].

One of the drawbacks of using a Utah array is that the neurons picked up from individual electrodes can vary from day to day. Because of this, we can not assume the recordings from one session are equivalent to another. This is problematic since neural recording sessions for monkeys usually only last 1-2 hours. Therefore, even if we record multiple sessions of monkeys performing the same task, we can only train models on a single session of neural activity at a time.

This lack of data poses a challenge when designing models since many of these models require large amounts of data to achieve state-of-the-art results. In this chapter, we will discuss using semi-supervised models to analyze and predict neural behaviors of recordings from monkeys' neural activity. We theorize that by accurately predicting neural behavior, we will be able to develop a deeper understanding of how the brain works. This chapter explicitly covers designing systems that can accurately predict neural behavior using time-series models. Not only that, but we will also go over how to overcome some limitations imposed by these small datasets.

When designing these systems, we focused on the task at hand. The goal was to be able to impute data in terms of both time and space. Since we are trying to predict the time, we decided to use time-series models. We decided to start with the most straightforward time-series models (RNNs) and slowly iterate and add complexity until we get acceptable results. Ultimately, we went with an RNN-GRU and transformer-based architecture because RNNs and transformers work well for language modeling and time-series forecasting.

Finally, we developed some approaches to overcome constraints imposed by small datasets. The primary approach developed is dynamic masking the input and output during training. This is a standard method used in computer vision when trying to work with small datasets and choosing not to create synthetic data. This idea also has been applied in NLP. A good example is the RoBERTa model. The RoBERTa model applied dynamic masking to sentence data; this way it sees different versions of the same sentence at every epoch. The paper notes a slight improvement in performance over static masking [16].

3.2 Methods

3.2.1 Datasets

In this section, we will discuss some of the datasets that were used to evaluate the models we created. The datasets were assembled and curated by the Systems Neural Engineering Lab at Georgia Tech as part of a contest for imputing held-out neural data from macaque motor-control tasks. The ultimate goal of the contest, Neural Latent Benchmark, is to compare the performance of various models across the field of neuroscience. For more details, see [17].

Four major datasets are used to evaluate our models. They are mc_maze, mc_rtt, area2_bump, and dmfc_rsg.

The dataset mc_maze contains spiking times and behavioral data recorded from the motor cortex and dorsal premotor cortex of a monkey while it was performing a reaching task. The task involved a center-out task. In this case, the center out task involved the monkey moving its hand from the center of a maze to another marked destination [17].

The dataset mc_rtt contains spiking times and behavioral data recorded from the primary cortex of a monkey while it was performing a reaching task. The reaching task was a set of sequential tasks where the monkeys are expected to touch a target which is randomly generated in an 8x8 matrix [17].

The dataset area2_bump contains spiking times and behavioral data recorded from the somatosensory cortex of a monkey while it was performing a reaching task. This task is also a center-out task where the monkey is bumped before reaching for the target [17].

The dataset dmfc_rsg contains spiking times and behavioral data recorded from the dorsomedial frontal cortex of a monkey while it was performing a ready/set/go task where the monkey is shown two stimuli and must get to the target by either moving its hand or making visual contact with it.[17].

All the datasets will be split into four sections; held-in, impute forward, impute space, impute forward + space. Our task is to use the held-in portion of the data to impute missing neural data.

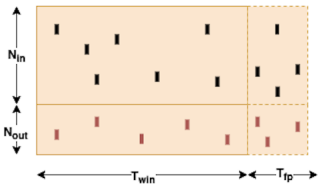

Figure 3.1. Figure 3.1 shows the the overall structure of the Utah array data. N_{in} to T_{win} is the held-in data. N_{out} to T_{win} is impute space. N_{in} to T_{fp} is impute forward. N_{out} to T_{fp} is impute space + forward.

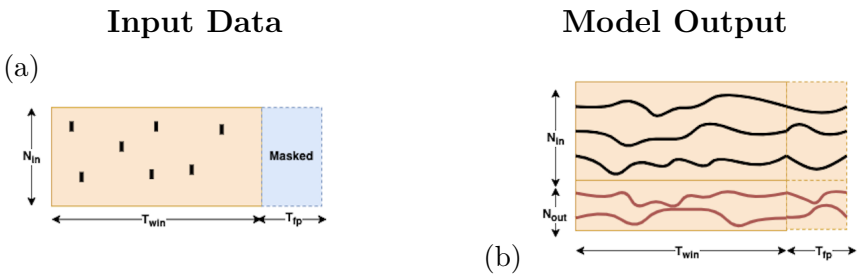

Figure 3.2. Figure 3.1.a shows the initial input that the model takes in. We input the held-in data and the impute forward data, but we mask the impute forward portion and replace that section with zeroes. Figure 3.1.b shows the output of the model. The output averages the amount of neural activity or spikes and predicts in terms of space and time.

3.2.2 Models

In this section we will discuss how we came up with new architectures to predict neural behavior in monkey data.

RNNF

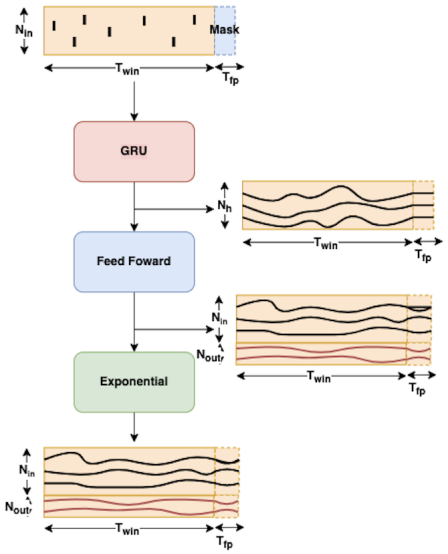

Figure 3.3. Figure 3.3 shows the architecture of RNNF and the output after each operation. The GRU predicts in terms of time. The feed-forward predicts in terms of space. Exponential is used for averaging.

One of the assumptions we made while designing different architectures to model neural data is that the temporal correlation is localized. This makes it different from language modeling where information aggregation from long temporal distances is significant. Given the limited training data, this assumption allows us to reduce the model complexity. Hence, the first architecture that we designed was RNN based model.

We used Gated Recurrent Units (GRU) coupled with a feed-forward layer to maximize our performance. The GRU predicts forward in terms of time while the feed-forward predicts in terms of space.

However, we found that transformer-based models such as the neural data transformer achieve superior cross-entropy loss when imputing data forward in time, but perform more or less the same when calculating the cross-entropy loss across the entire output. In the above model, we're using the RNN to predict forward in time and the feed-forward layer to make the predictions in space (held-in + held-out neurons).

Neural RoBERTa

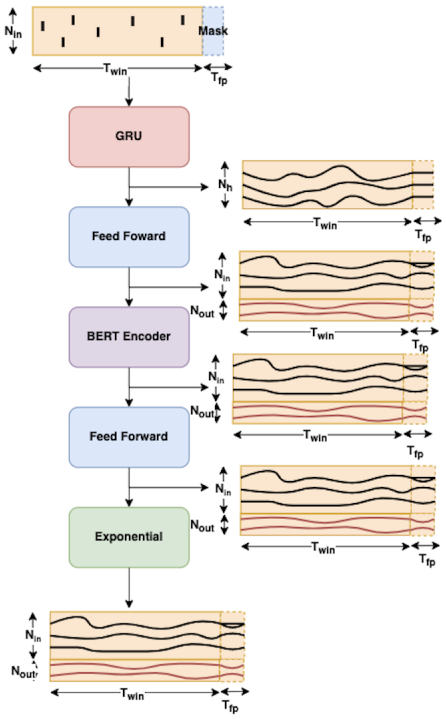

Figure 3.4. Figure 3.4 shows the architecture of Neural RoBERTa and the output after each operation. The GRU predicts in terms of time. The feed-forward predicts in terms of space. BERT encoder and feed-forward clean up the data in terms of both time and space. Exponential is used for averaging.

The Neural RoBERTa is a model designed to make up for some of the shortcomings that the RNNF. One of the drawbacks of using an RNNF model was in terms of predicting forward in time. We saw that even though the RNNF models could compete with other models submitted to the Neural Latent Benchmark in terms of overall cross-entropy, the RNNF's cross-entropy loss of imputed data forward in time was significantly lower than the other transformer-based model. To solve this issue we created the Neural RoBERTa architecture. This architecture couples RNNF with a transformer (BERT) encoder. The RNN part of the network predicts forward in time, the feed-forward expands in space and the transformer makes corrections in space and time to provide better results. Adding a

BERT encoder on top of the RNN layer aggregates information from across time steps and neuron channels to generate relatively more stable predictions.

Neural r-RoBERTa (Ensemble)

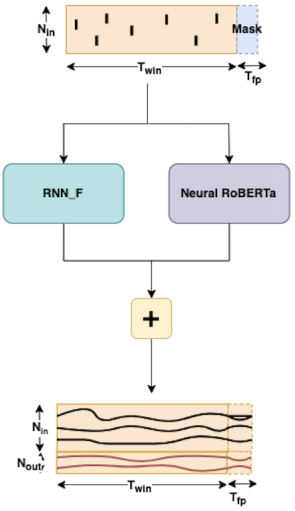

Figure 3.5. Figure 3.5 shows the architecture of Neural r-RoBERTa and the output after each operation. In this model we average the output of both the RNNF and Neural RoBERTa.

To maximize the results from both RNNF and Neural RoBERTa, we decided to build Neural r-RoBERTA. Neural r-RoBERTA is an ensemble model comprising a combination of the RNNF and Neural RoBERTa architectures. We decided to go with a simple ensemble technique. We took the top-performing model of each type (RNNF and Neural RoBERTa), then we load these models onto the Neural r-RoBERTa architecture. During training and testing, the Neural r-RoBERTa model feeds the input data into both the RNNF and the Neural RoBERTa models and averages the two outputs together. To make sure that we get the best performance possible, we train the model for another 50-100 epochs before evaluating the model.

3.2.3 Masking

During preliminary investigations, we observed the models were not stable. A small change in any hyper-parameter would cause a big change in performance. Not only that, but we observed that the model was not able to perform as well as the other top models submitted to the Neural Latent Benchmark. Therefore we decided to apply dynamic masking. This is crucial because of the size of the dataset. We hypothesized performance can be improved by dynamically masking the data, thereby generating 'extra' training data and making the model robust [16]. We propose two different kinds of masking: 'dot' masking, in which random, isolated samples are masked; and 'strip' masking, in which random pairs of consecutive samples are masked. The latter forces to model to learn more dynamic structures.

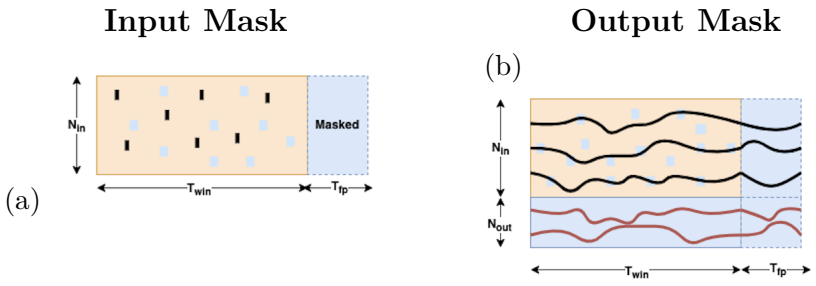

Figure 3.6. This figure is an example of the application of the 'dot' mask. The same procedure applies to the 'strip' mask. The 'dot' mask technique is only applied to the held-in portion while the other portions we want to predict get completely masked. Figure 3.6.a is an example of the application of the 'dot' mask on the input data. All the values of the areas that are masked get replaced by zeroes. Figure 3.6.b is an example of the application of masking on the output data. For the output data, we introduce the concept of inverse masking. In inverse masking, as opposed to regular masking where the values selected by the masking procedure get replaced by zeroes, we extract those values selected by the masking procedure and compute the loss by comparing the expected values at the same coordinate points.

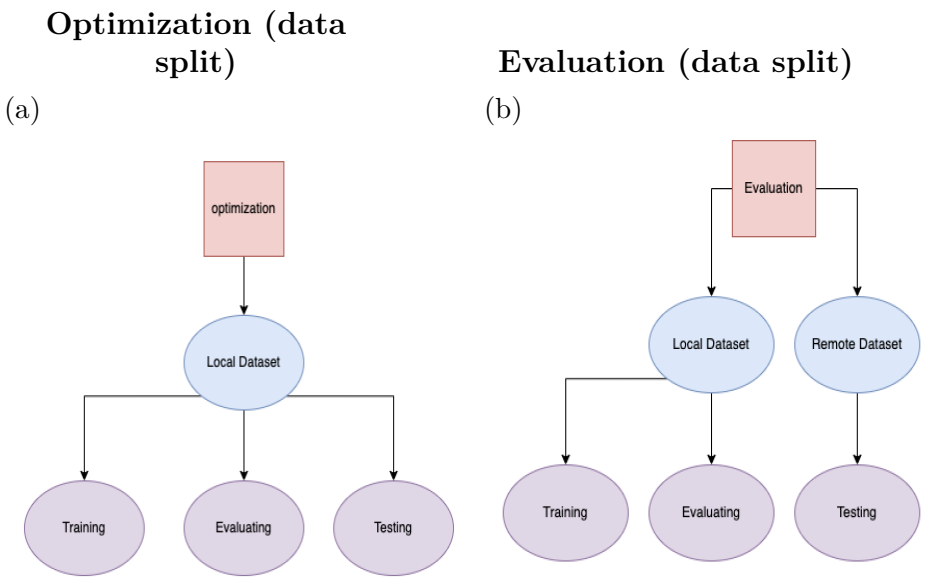

Figure 3.7. Figure 3.7.a shows a high-level representation of the data distribution during hyper-parameter optimization. We split the local dataset into training, evaluating, and testing data. Figure 3.7.b high-level representation of the data distribution during the hyper-parameter evaluation. We use the same local dataset from the optimization step except for this time we split it to make the training and evaluating data. For the testing data, we extract it from the remote dataset. The remote dataset is a private dataset that only includes the input data specially designed for the Neural Latent Benchmark.

3.2.4 Training

The training process contains two distinct steps; hyper-parameter optimization and evaluation. We run hyperparameter optimization by splitting the neural data from the 'local dataset' into training, testing, and evaluating data. Once split, we use the training data to optimize the model while using the evaluation data to save the best-performing model and perform early stopping. We iterate through this training cycle until we complete all the assigned epochs or the early stopping mechanism triggers. Once done training, the top model is loaded, and the test data is fed into the model. We then analyze the results by computing the 'co-bps' by computing the cross-entropy of the predicted and the expected output. We repeat this process with models of differing hyper-parameter values for 100 trials.

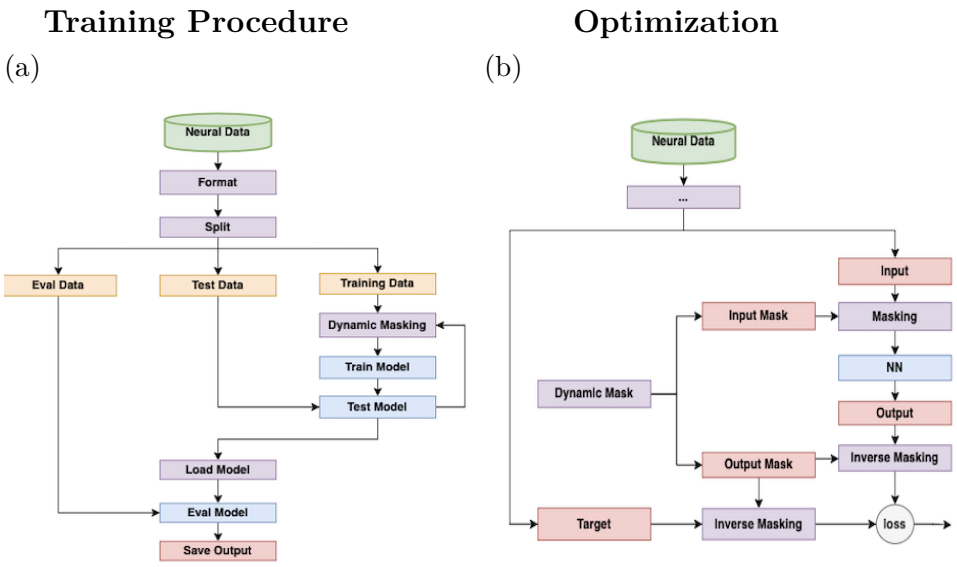

Figure 3.8. Figure 3.8.a shows a high-level representation of the training procedure used to train Neural RoBERTa and RNNF. Figure 3.8.b shows a more detailed representation of how the model computes the loss during the training procedure.

For the evaluation step, we set the hyper-parameters equal to the top performing model during the optimization step. After, neural data is split from the 'local dataset' into training and evaluating set and then we use the 'remote dataset' as the testing set. Once split, we use the training data to optimize the model while using the evaluation data to save the best-performing model and perform early stopping. We iterate through this training cycle until we complete all the assigned epochs or the early stopping mechanism triggers. Once done training, the top model is loaded, and the test data is fed into the model. We then save the output and submit it to the Neural Latent Benchmark where they analyze and compute the 'co-bps' and 'fp-bps'.

Figure 3.7.b focuses on how we compute the training loss. The training data contains two different data. The input data contains the held-in data, while the target data expects the expected output, which is the held-in data and the held-out data. Held-out data is just the data that the model has to predict. As a reminder, we are trying to predict in terms of time and space. Once the data is ready, we use our dynamic masking function to generate

an input and output mask. Once we create the masks, we apply the input mask to the input data before feeding it into our neural network. Afterward, we gather the neural network's output and use the output mask as an inverse mask (See Sec. 3.2.3) on both the target and the neural network's output, using the Poisson cross entropy to compute the loss.

3.2.5 Optimization

To perform hyperparameter optimization on Neural RoBERTa and RNNF we used two different approaches. The first approach we took was to semi-manually tune models. By semi-manually tuning, we are referring to a combination between two popular techniques which are manual hyperparameter tuning and grid-search. The intuition is that manual tuning should give us a rough idea of the hyper-parameter boundaries that we might want to search within as well as help us prune hyper-parameters that have little to no effect on the actual performance of the model. Then we can use a grid search to search within these boundaries to find what we presume to be the optimal parameter set. This also saves us time in the long run since we can reduce the search space that the grid search uses for each hyper-parameter, therefore, reducing the total amount of time to get the final results. This approach was used for the Neural Latent Benchmark competition.

The second strategy we took was an automatic approach to optimize the hyper-parameters. The automatic approach we use is Bayesian optimization. Bayesian optimization assumes that the results from running models with different hyper-parameters are sampled from a Gaussian process and maintain a posterior distribution [18]. Therefore, the posterior can be used to optimize and narrow down hyper-parameters. Specifically, we use the Optuna framework because of its efficient implementation of both searching and pruning strategies [19]. Since we already had some data from optimizing using the semi-manual approach we were able to determine what the appropriate boundaries were for each hyper-parameter, but I ended up slightly expanding the range of the boundaries since the Bayesian optimization can narrow down the search space all by itself. Not only that, but it can do a better job at optimizing than our previous approach due to having more flexibility than a grid search when searching through different combinations of hyperparameter values.

3.2.6 Evaluation

Five main metrics were used to analyze the performance of our models on the monkey neural data. They are as follows; co-bps, fp-bps, Vel R2, tp-corr, and psth R2 (See Paper [17]). In this section, we will only go over a brief overview of how the two main metrics used to evaluate the models work.

The main metric used to evaluate models the cross-entropy between our model output and the expected output. Following the terminology of the Neural Latent Benchmark, we refer to this as 'co-bps'. This is applied to all datasets. This is done with the following formula:

$$\text{bits/spike} = \frac{1}{n_{sp} \log 2}(\mathcal{L}(\lambda; \hat{y}) - \mathcal{L}(\hat{\lambda}_n; \hat{y}_{n,t}))$$

Where n_{sp} is the total number of spikes, $\hat{y}_{n,t}$ is the observed spiking activity of neuron n at time t, $\lambda_{n,t}$ is the submitted rate predictions of neuron n at time t, $\hat{\lambda}_n$ is the mean firing rate of neuron n, and $\mathcal{L}(\lambda; y)$ is the Poisson log-likelihood of observed.

The secondary metric used to evaluate models is calculating the cross-entropy between our model output and the expected output for the imputed data that was predicted forward in time. Following the terminology of the Neural Latent Benchmark, we refer to this as 'fp-bps'.

3.3 Experiments

3.3.1 Effects of Masking

We hypothesized performance can be improved by dynamically masking the data, thereby generating 'extra' training data. We investigated two different kinds of masking: 'dot' masking, in which random, isolated samples are masked; and 'strip' masking, in which random pairs of consecutive samples are masked. The latter forces the model to learn more dynamical structure. We compare these results to training without masking that is otherwise identical.

3.3.2 Statistical Tests

To show the superiority of the Neural RoBERTa network over the RNNF on our tasks, we compared the performance of ten instances of each model under a one-sided Wilcoxon rank-sum test. More precisely, we ran Bayesian optimization for one hundred trials. Once the one hundred trials were complete, we took the top ten trials of each model based solely on co-bps. Once we extracted the values from the top ten trials of each model we computed the p-values by using a one-sided Wilcoxon rank-sum test.

We choose not to use paired t-test because we extracted the top ten trials instead of randomly sampling from the available trials. This means that the distribution would no longer be normal. Therefore we choose to use the one-sided Wilcoxon rank-sum test to compute the p-values which impose fewer assumptions than the paired t-test. For this experiment, we assumed that if the p-value is less than 0.05 then the results are statistically significant. This procedure is done for all the different datasets.

3.4 Results

3.4.1 Overall Performance

In this section, we will compare how the models compare against each other and compare to some of the other top models that are used to extract neural behavior from monkey data. Table 3.1 shows the overall results from the analysis.

3.4.2 Masking

In this section, we will compare how different types of masking affects the performance of the RNNF models. We performed experiments with three different masking alternatives; 'dot' mask, 'strip' mask and no mask. Table 3.2 shows the results of the masking experiment.

Despite performing parameter optimization during each experiment and only taking the top result, the no mask approach performs the worse. As we hypothesize, by utilizing dynamic masking we can get a slight boost in performance. We believe this is because by

Table 3.1. Overall Performance of all models

Dataset	model	co-bps	fp-bps/tp corr	Vel R2	psth R2
mc_maze_small	Neural RoBERTa	0.3518	0.1445	0.8453	0.5051
mc_maze_small	RNNF	0.3413	0.1246	0.8185	0.4339
mc_maze_small	Neural r-RoBERTa	0.3541	0.1470	0.8254	0.5041
mc_maze_medium	Neural RoBERTa	0.3385	0.1663	0.9041	0.6456
mc_maze_medium	RNNF	0.3034	0.1532	0.8271	0.5950
mc_maze_medium	Neural r-RoBERTa	0.3298	0.1754	0.8762	0.6547
mc_maze_large	Neural RoBERTa	0.3956	0.2123	0.9181	0.7193
mc_maze_large	RNNF	0.3852	0.1967	0.9139	0.7124
mc_maze_large	Neural r-RoBERTa	0.3981	0.2105	0.9185	0.7560
mc_maze	Neural RoBERTa	0.3551	0.2282	0.8731	0.6780
mc_maze	RNNF	0.3382	0.2117	0.9083	0.6361
mc_maze	Neural r-RoBERTa	N/A	N/A	N/A	N/A
area2_bump	Neural RoBERTa	0.2888	0.1486	0.8724	0.6586
area2_bump	RNNF	0.2823	0.1353	0.8651	0.6259
area2_bump	Neural r-RoBERTa	0.2931	0.1464	0.8761	0.6613
mc_rtt	Neural RoBERTa	0.2074	0.1279	0.6410	N/A
mc_rtt	RNNF	0.2119	0.1148	0.6133	N/A
mc_rtt	Neural r-RoBERTa	0.2168	0.1341	0.6489	N/A
dmfc	Neural RoBERTa	0.1746	0.1497/-0.7195	N/A	0.5323
dmfc	RNNF	0.1781	0.1538/-0.7049	N/A	0.5290
dmfc	Neural r-RoBERTa	0.1794	0.1596/-0.7063	N/A	0.5316

utilizing dynamic masking during training, the models can see different versions of the same neural sequence. Therefore, getting a slight performance boost.

3.4.3 Statistical Tests

In this section, we discuss some of the results of running the statistical tests. For this experiment, we compare across models to prove Neural RoBERTa is statistically superior to RNNF in terms of both co-bps and fp-bps where the p-value must be below 0.05 to be considered significant. Table 3.3 shows the results from computing running statistical tests comparing RNNF and Neural RoBERTa.

Table 3.2. RNNF Masking Experiment Results on mc_rtt dataset

Mask Type	dataset	co-bps	fp-bps	vel R2
No Mask	mc_rtt	0.1962	0.1099	0.5267
dot mask	mc_rtt	0.2032	0.1038	0.5544
strip Mask	mc_rtt	0.2058	0.1170	0.5731

Table 3.3. Statistical Test Results (p-values) Comparing RNNF and Neural RoBERTa in terms of co-bps and fp-bps at a p-value threshold of 0.05

Dataset	Metric	Unpaired t-test	Wilcoxon rank-sum	Status
mc_maze_small	co-bps	7.2202e-05	0.0050	reject
mc_maze_small	fp-bps	0.0406	0.0744	accept
mc_maze_medium	co-bps	1.1100e-10	0.0050	reject
mc_maze_medium	fp-bps	0.0005	0.0069	reject
mc_maze_large	co-bps	1.8984e-09	0.0050	reject
mc_maze_large	fp-bps	2.4604e-06	0.0050	reject
area2_bump	co-bps	8.6066e-10	0.0050	reject
area2_bump	fp-bps	0.0018	0.0125	reject
mc_rtt	co-bps	2.8365e-08	0.0050	reject
mc_rtt	fp-bps	0.0035	0.0468	reject
dmfc	co-bps	0.7172	0.5076	accept
dmfc	fp-bps	0.3094	0.2845	accept

As you can see, Neural RoBERTa proves to be statistically superior to RNNF in terms of co-bps for almost all the datasets. The only two exceptions are mc_maze_small and dmfc. This was expected. For the mc_maze_small dataset, the RNNF was statistically better due to the size of the dataset. Due to the BERT encoder, the Neural RoBERTa thrives with a larger amount of data compared to the RNNF model. Since mc_maze_small only contains about 75 samples, the Neural RoBERTa struggles to perform therefore getting outperformed by the RNNF architecture. The second dataset where the Neural RoBERTa is not superior to the RNNF architecture is dmfc. This is because both models struggle with this dataset as mentioned in the previous section. We believe that this is caused by the fact that the data is not collected straight from the motor cortex like in the other datasets. Therefore, our

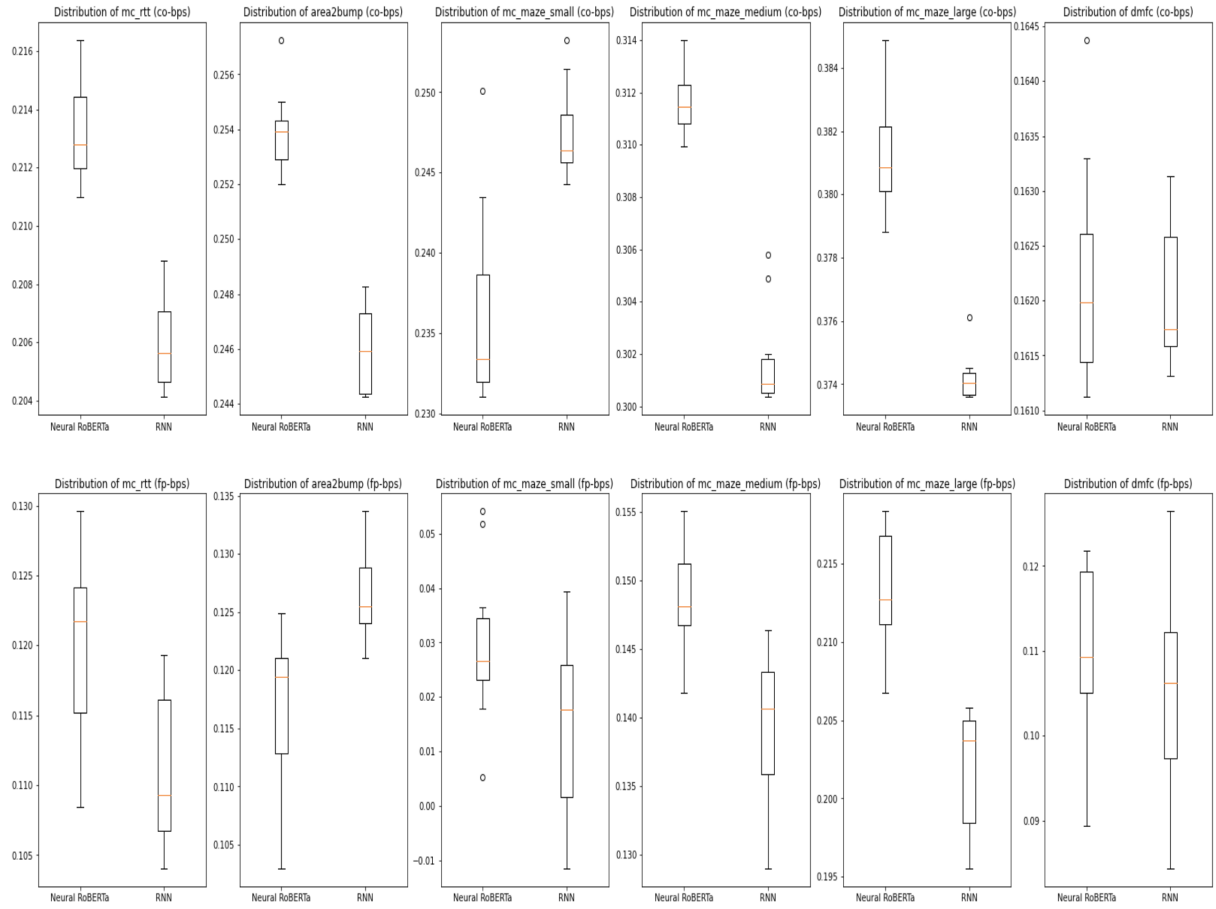

Figure 3.9. Figure 3.9 shows the distribution of RNNF and Neural RoBERTa based on co-bps and fp-bps. These values belong to the models used to compute the p-value.

models find it hard to find correlation when predicting neural behavior. This is why both models performed similarly based on our metrics.

Similarly, for fp-bps, we can see that the Neural RoBERTa is superior to RNNF in almost all the datasets except dmfc. This is once again due to the fact mentioned in the last paragraph.

3.5 Future Work

There are many different paths we can take with these models. Because of the outstanding performance achieved by the RNNF and Neural RoBERTa, the following logical path is to run predictions of kinematic tasks on the monkey datasets.

Another path we can take in the future is the application of these models to impute missing ECoG data. Often, a small percentage of electrodes can go bad, and these channels get removed during pre-processing. Depending on how many channels are removed, we can lose important data, which can negatively impact the performance of models decoding ECoG data. Therefore, we can use these models to save these bad channels by predicting (interpolating) the missing values.

3.6 Conclusion

In conclusion, the RNNF and Neural RoBERTa outperform other state-of-the-art models in the neural latent benchmark. We got the top results for both mc_rtt and area2_bump. Not only that, but in this chapter, we prove the positive effect of dynamic masking. We improve the performance of our model by almost 0.01 compared to the non-masking method. Finally, we proved that the Neural RoBERTa architecture is significantly superior to the RNNF in terms of both fp-bps and co-bps. The only exception is the dmfc dataset. We hypothesize that this is due to the area of the brain where this data is recorded from. Unlike the other datasets, dmfc is recorded from the Dorsomedial Prefrontal Cortex which is a cognitive region of the brain. This makes it complicated for the models to be able to learn the correlation between movement and neural activity therefore both RNN and Neural RoBERTa perform poorly. In the future, we hope to expand these algorithms to ECoG data especially for solving the task of imputing missing neural data.

4. CONCLUSION

In conclusion, the results from the repeated sentences experiment were satisfactory. On the MOCHA-1 set, the transformer managed to outperform the current state-of-the-art model on participant 403, match the performance of participant 400, and underperformed for participant 401. These results contradicted our initial hypothesis that we could increase prediction accuracy by reducing the downsampling of ECoG performed by the state-of-the-art model. The other unexpected result is that we outperformed the RNN-LSTM on the participant with the least amount of data recorded for MOCHA-1 while simultaneously not being able to perform on par with the LSTM model on participant 401, which contained the most amount of data for MOCHA-1. We expected the opposite result since the transformer tends to be more data-driven than the smaller, simpler RNN models. Finally, even by adding the attention mechanism to our RNN model, we get our best results by downsampling by the same amount as the state-of-the-art model. The main limiting factor for the performance of these models is the amount of data available. Finally, the generalization experiment did not yield acceptable results. We could not obtain significant results without the use of transfer learning. We hypothesize that the amount of data was insufficient to fine-tune the pre-trained models (MarianMT and Wave2Vec2) correctly, which is an area that requires further investigation and experimentation. In the future, we hope to overcome limitations imposed by dataset size creating more data through data splicing.

The RNNF and Neural RoBERTa outperform other state-of-the-art models in the neural latent benchmark. We got the top results for both mc_rtt and area2_bump. Not only that, but we prove the positive effect of dynamic masking. We improve the performance of our model by almost 0.01 compared to the non-masking method. Finally, we proved that the Neural RoBERTa architecture is significantly superior to the RNNF in terms of both fp-bps and co-bps. The only exception is the dmfc dataset. We hypothesize that this is due to the area of the brain where this data is recorded from. Unlike the other datasets, dmfc is recorded from the Dorsomedial Prefrontal Cortex which is a cognitive region of the brain. This makes it complicated for the models to be able to learn the correlation between movement and neural activity therefore both RNN and Neural RoBERTa perform poorly.

In the future, we hope to expand these algorithms to ECoG data especially for solving the task of imputing missing neural data.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[2] S. Chakrabarti, H. Sandberg, J. Brumberg, and D. Krusienski, "Progress in speech decoding from the electrocorticogram," *Biomedical Engineering Letters*, vol. 5, pp. 10–21, Mar. 2015. DOI: 10.1007/s13534-015-0175-1.

[3] J. N. Saby and P. J. Marshall, "The utility of EEG band power analysis in the study of infancy and early childhood," en, *Dev. Neuropsychol.*, vol. 37, no. 3, pp. 253–273, 2012.

[4] B. N. Pasley and R. T. Knight, "Decoding speech for understanding and treating aphasia," en, *Prog. Brain Res.*, vol. 207, pp. 435–456, 2013.

[5] E. M. Maynard, C. T. Nordhausen, and R. A. Normann, "The utah intracortical electrode array: A recording structure for potential brain-computer interfaces," *Electroencephalography and Clinical Neurophysiology*, vol. 102, no. 3, pp. 228–239, 1997, ISSN: 0013-4694. DOI: https://doi.org/10.1016/S0013-4694(96)95176-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0013469496951760.

[6] K. Woeppel, C. Hughes, A. J. Herrera, J. Eles, E. C. Tyler-Kabara, R. A. Gaunt, J. L. Collinger, and X. T. Cui, "Explant analysis of utah electrode arrays implanted in human cortex for brain-computer-interfaces," *medRxiv*, 2021. DOI: 10.1101/2021.08.28.21262765. eprint: https://www.medrxiv.org/content/early/2021/08/30/2021.08.28.21262765.full.pdf. [Online]. Available: https://www.medrxiv.org/content/early/2021/08/30/2021.08.28.21262765.

[7] J. G. Makin, D. A. Moses, and E. F. Chang, "Machine translation of cortical activity to text with an encoder-decoder framework," *bioRxiv*, 2019. DOI: 10.1101/708206. eprint: https://www.biorxiv.org/content/early/2019/07/22/708206.full.pdf. [Online]. Available: https://www.biorxiv.org/content/early/2019/07/22/708206.

[8] G. K. Anumanchipalli, J. Chartier, and E. F. Chang, "Speech synthesis from neural decoding of spoken sentences," *Nature*, vol. 568, no. 7753, pp. 493–498, Apr. 2019, ISSN: 1476-4687. DOI: 10.1038/s41586-019-1119-1. [Online]. Available: https://doi.org/10.1038/s41586-019-1119-1.

[9] A. J. Yates, "Delayed auditory feedback.," *Psychological Bulletin*, vol. 60, no. 3, pp. 213–232, 1963. DOI: 10.1037/h0044155. [Online]. Available: https://doi.org/10.1037/h0044155.

[10] H.-Y. S. Chien, J. S. Turek, N. Beckage, V. A. Vo, C. J. Honey, and T. L. Willke, *Slower is better: Revisiting the forgetting mechanism in lstm for slower information decay*, 2021. DOI: 10.48550/ARXIV.2105.05944. [Online]. Available: https://arxiv.org/abs/2105.05944.

[11] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2014. DOI: 10.48550/ARXIV.1409.0473. [Online]. Available: https://arxiv.org/abs/1409.0473.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Going deeper with convolutions*, 2014. DOI: 10.48550/ARXIV.1409.4842. [Online]. Available: https://arxiv.org/abs/1409.4842.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, 2014. DOI: 10.48550/ARXIV.1412.3555. [Online]. Available: https://arxiv.org/abs/1412.3555.

[14] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "Wav2vec 2.0: A framework for self-supervised learning of speech representations," *CoRR*, vol. abs/2006.11477, 2020. arXiv: 2006.11477. [Online]. Available: https://arxiv.org/abs/2006.11477.

[15] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. Fikri Aji, N. Bogoychev, A. F. T. Martins, and A. Birch, "Marian: Fast neural machine translation in C++," in *Proceedings of ACL 2018, System Demonstrations*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 116–121. [Online]. Available: http://www.aclweb.org/anthology/P18-4020.

[16] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. arXiv: 1907.11692. [Online]. Available: http://arxiv.org/abs/1907.11692.

[17] F. Pei, J. Ye, D. M. Zoltowski, A. Wu, R. H. Chowdhury, H. Sohn, J. E. O'Doherty, K. V. Shenoy, M. T. Kaufman, M. Churchland, M. Jazayeri, L. E. Miller, J. Pillow, I. M. Park, E. L. Dyer, and C. Pandarinath, "Neural latents benchmark '21: Evaluating latent variable models of neural population activity," in *Advances in Neural Information Processing Systems (NeurIPS), Track on Datasets and Benchmarks*, 2021. [Online]. Available: https://arxiv.org/abs/2109.04463.

[18] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," *CoRR*, vol. abs/1907.10902, 2019. arXiv: 1907.10902. [Online]. Available: http://arxiv.org/abs/1907.10902.