THE SOUND WITHIN: LEARNING AUDIO FEATURES FROM ELECTROENCEPHALOGRAM RECORDINGS OF MUSIC LISTENING

A Dissertation
Presented to
The Academic Faculty

By

Ashvala Vinay

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Music

Georgia Institute of Technology

May 2020

THE SOUND WITHIN: LEARNING AUDIO FEATURES FROM ELECTROENCEPHALOGRAM RECORDINGS OF MUSIC LISTENING

Approved by:

Dr. Grace Leslie
School of Music
Georgia Institute of Technology

Dr. Alexander Lerch
School of Music
Georgia Institute of Technology

Dr. Justin Romberg
School of Electrical Engineering
Georgia Institute of Technology

Date Approved: April 24, 2020

If I have seen further than others, it is by standing upon the shoulders of giants.

Issac Newton

To my grandmothers, Kamala and Vinoda

ACKNOWLEDGEMENTS

To my parents, Subhashini and Vinay who have been a pillar of support throughout this journey. Without your unyielding support and love, it's unlikely that I would have been able to explore and discover new worlds.

To my partner, Shalini, thanks for being there for me. Your presence, support, encouragement and love mean the world to me and none of this would be doable without you.

To Grace Leslie, my advisor. Thank you for giving me the freedom, encouragement, mentorship and resources to pursue ideas to grow as a researcher and person.

To Alexander Lerch. Thank you for the many productive meetings, insights, advice and making me a part of the Music Informatics Lab.

To Sidd and Ashis, thanks for indulging me in my ideas and thought processes. Thanks for all the conversations and I look forward to many more.

To my housemates - Jason, Lisa, Sandeep, Antoine for being some of the best people I know and have the pleasure of spending time with. I cannot wait to play more video games, watch TV shows with you all again.

To Vlad Jimenez at CSUN, for being one of my best friends through the years and being a cool parallel. Thanks for being patient with my poor response times and listening to my rants about PyTorch failing. I believe multiple *Denny's* meals are at stake here.

To the faculty and staff at GTCMT, who have been incredibly supportive both in terms of academic support and infrastructure.

And to my friends Shauna, Kaushal, Raghav, Tejas and Snehesh. Thanks for being generous with your time and friendship. I cannot wait to meet all of you again soon.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

We look at the intersection of music, machine Learning and neuroscience. Specifically, we are interested in understanding how we can predict audio onset events by using the electroencephalogram response of subjects listening to the same music segment. We present models and approaches to this problem using approaches derived by deep learning. We worked with a highly imbalanced dataset and present methods to solve it - tolerance windows and aggregations.

Our presented methods are a feed-forward network, a convolutional neural network (CNN), a recurrent neural network (RNN) and a RNN with a custom unrolling method. Our results find that at a tolerance window of 40 ms, a feed-forward network performed well. We also found that an aggregation of 200 ms suggested promising results, with aggregations being a simple way to reduce model complexity.

CHAPTER 1

INTRODUCTION AND BACKGROUND

1.1 Introduction

We will seek to answer the following question:

> Given an electroencephalogram (EEG) signal of a person listening to music, how well can an Artificial Neural Network (ANN) extract features of audio from the EEG signal?

Specifically, we are interested in using ANNs to predict onsets in music using the EEG signal recorded during a person's listening to the same music signal. A visual description of the problem is outlined in Fig 1.1.

Onsets mark the beginnings of transient events [1]. Detecting onsets is a fundamental task in the both the signal processing and music information retrieval domains [2]. In recent years, the problem of detecting onsets in audio has been approached using ANNs [2, 3]. We will discuss neural networks in more detail in Chapter 2. However, there are no present methods that use EEGs to estimate the presence of onsets in a music signal.

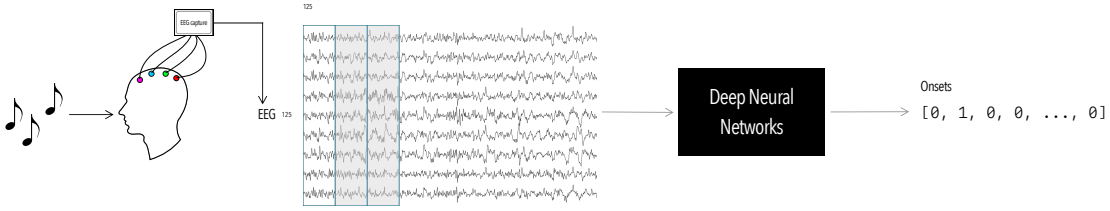

Figure 1.1: What are we trying to do? We have EEG data of people listening to music. Can we feed that into a deep neural network and figure out where the onsets are in the music?

1.2 Motivation

Given the growing ubiquity of machine learning in our everyday lives where it is used to perform translations between a sequence in one language to another [4] and learn good representations of data [5, 6], we are naturally driven to wonder - *Can machine learning models learn audio features from EEG recorded during music?*.

Prior works exists at the intersection of machine learning of music, machine learning and physiological data [7, 8, 9]. However, it is worth noting that this is a new area of research and therefore, no immediately tractable solution esists.

The end-goal [10] for research within the aforementioned area is to reconstruct the full audio signal from the EEG of someone listening to the music. The most recent paper on this topic by Ofner and Stober [7] reconstructs the mel spectrum of audio data from EEG, but, does not have any numerical benchmarks. We would like to look at a smaller sub-class of problems within the idea of reconstruction, i.e, feature extraction.

We foresee audio onsets extracted from EEG of music listening being used two ways - First, a recommendation system where the audio features that can be extracted from the EEG signal could be used to produce recommendations or identify songs similar to this set in a database. Second, we also see it being useful in a generative-music oriented system, detected onsets could be used to modulate rhythmic structure of performed or generated music.

Our hope with this thesis is to make progress towards the goal of reconstructing a full audio signal, while providing good benchmarks and methods that future research can build upon.

1.3 Related Works

For almost a century, neuroscientists have studied and remained fascinated with the ability of electroencephalograms to capture activity in a person's brain [11]. One of the earliest

works of literature we could find on the use of music and its effects, as observed by EEG was in 1949 by Dr. H.B. Stubbe Teglbjærg in his study on *Musicogenic Epilepsy*, a form of epilepsy that is triggered upon listening to music [12].

Since 1949, there have been countless studies and literature that assess how well humans perform at music recognition [13, 14], how music affects emotions [14, 15, 16] and how broader studies on how music is processed by the brain [17, 18].

A seminal result in this area is Fujioka et al's paper on neural beat processing [19]. The authors use magnetoencephalography (MEG), a non-invasive method similar to EEG, in order to measure neural activations at the scalp level. This study used alternating loud and soft tones at a fixed 390 ms inter onset interval (time between two onsets) as stimulus. Their results found that there are spikes in activity in the Beta (10-25 Hz) and Gamma (28-48 Hz) bands of the MEG signal consistent with the 390 ms inter onset interval. They used wavelet transforms to decompose the data into a time-frequency representation to better analyze synchronicity between onsets in audio and onsets in the MEG data.

Extracting musical features from physiological responses to music is a challenging task [7]. There have been successes with the use of electrocorticography (ECoG) data [20, 18]. ECoG is an invasive procedure that allows us to probe the areas of auditory processing such as the auditory cortex and Heschl's gyrus [21] and identify music-related activations more precisely. Traditional EEG paradigms [19, 10, 22] get scalp-level activations non-invasively and additionally, relies on a standardized pipeline of processing to ensure that the data's artifacts are removed, epoched around event time codes in order to understand the physiological responses to musical events.

In recent years, there has been a significant shift in trends at conference venues such as ISMIR towards machine learning and deep learning based approaches towards feature extraction and information retrieval from music [23, 24]. Simultaneously, this has piqued the interest of researchers who have worked on related works that are researching questions combining music and EEG [9, 25, 7, 10, 26, 27].

At the convergence of EEG, music and machine learning, a majority of the work has primarily been in asking and solving classification-related questions. Within this sub-topic, the questions can be broken down as follows:

- Can we identify what song the subject was listening to?
- Can we identify what emotion the subject was feeling?

The identification process in all the papers we discuss use EEG as their primary modality of input and a class label such as the rhythm type or emotion as their output.

Schaefer et al [17] in 2011 published a paper in which they identify the song a person was listening to by using event-related potentials (ERP), a time-series representation typically computed around event timecodes. They use audio snippets of roughly 3.7 seconds long. For classification, they use a logistic regression classifier to effectively solve a *One vs Rest* problem where each class represents a song. They use data from ten subjects, a hundred and forty overall trials. By computing ERPs, you are effectively removing a broad section of your data, since epoching is performed over small time frames around a specific event. The results they cite include a 100% accurate classification across multiple trials. Given the small scale of data, it is likely that the classifier overfit.

An important paper on the topic of classifying EEG data of music listening is by Stober, Cameron and Grahn [8]. They discriminate between rhythmic patterns found in East African music and Western Classical music. For this task, they collect the EEG recordings of participants listening to these rhythmic patterns and train a convolutional neural network (CNN) to discriminate between these two classes of rhythms. This classification task gives the authors a classification accuracy of 8.7% across *all subjects and all rhythmic stimuli*. However, when evaluated as a binary classification problem, they have a 55.4% accuracy across the same criteria of subjects and rhythmic stimuli.

Ofner and Stober's work [7] is amongst the most recent work at this intersection. They propose the usage of Wang's VCCA [28], which uses a multi-head variational auto-encoder

(i.e, multiple encoders and decoders)[29] in order to use a one-second window of EEG to generate one second of mel spectrum. They use the private variant, which shares a common view between multiple encoders. This paper uses the NMED-T [25] dataset for their task. While they are the most recent paper, a primary issue is that they do not provide numerical benchmarks or numbers for their results, which means that we cannot compare our feature output to theirs or conduct mel spectrum based studies. We are therefore required to pick our own set of features and establish baselines and benchmarks for the same.

For predicting the emotions of people listening to music with EEG, the most recent and relevant paper is by Tripathi et al. [30], who use a deep learning approach to do this classification task. They use a convolutional neural network with 2D convolutions on the input EEG signal to predict the arousal and valance values. They find that their CNNs are capable of predicting valance with an accuracy of 81% and arousal with an accuracy of 73.36%.

CHAPTER 2

ONSET DETECTION IN A MUSICAL SEQUENCE

Knowing where onsets occur in a signal play a significant role in EEG studies. Traditionally, to compute an evoked potential of a channel, a participant's EEG signal is captured for multiple trials of stimulus presentation. For an individual stimulis, one computes an "epoch" around a small interval surrounding the onset of a trial, usually one second before and two seconds after the trial. Across trials, we compute evoked potential by computing the average signal per channel across each epoch. An example of this paradigm was discussed in [19], where the authors create an experiment with onsets that exist at a 390 ms inter-onset-interval, which when decomposed to a time-frequency representation across all MEG channels, showed a periodic synchrony between the onsets and activity in the MEG signal at the β and γ bands.

Event related potentials (ERPs) have been the standard paradigm in understanding neural processing in visual, motor and auditory tasks. We use the NMED-T dataset [25], which we will discuss in the next section, is largely single trials. Given an EEG signal of a subject listening to music, we would like to know whether there is a possibility of an onset that is found in the music.

Contemporary methods for onset processing in music use deep neural networks such as convolutional networks (CNN) and recurrent neural networks (RNN). A RNN is a neural network that forms a directed acyclic graph temporally. An RNN is derived by using a series of feed-forward networks, where the hidden weights for the network propagate across time. Mathematically, an RNN at a given time step t can be described as:

$$h(t) = f(h(t-1), x(t)) \tag{2.1}$$

Where $h(t)$ is the hidden weight of the network at a time step t and $x(t)$ is the input.

Convolutional neural networks are a feed-forward network that learn a feature map by convolving the output of a kernel over the input. The kernel slides across the size of the input to construct a feature or activation map.

Eyben et al. [2] use bi-directional recurrent neural networks (Bi-RNN) to detect onsets in musical signals. These networks are trained on the power spectrogram of the audio. They minimize cross-entropy between the output of their network and the ground truth label sequence. Schülter and Böck [3] subsequently used convolutional neural networks for onset detection. They use rectangular filters which are "wide in time, narrow in frequency".

Both networks are evaluated using F-scores, which we use in our evaluation method. To evaluate accuracy of prediction, they use a tolerance threshold of 50 ms around an onset, i.e, if a predicted onset is within $\pm$ 50 ms of the ground truth, it is counted as an accurate prediction.

MADMOM [31], a Music Informational Retrieval focused package has both the aforementioned methods built-in for onset detection.

There are no known works that predict onsets in music using the EEG signal from the perception of music. In this chapter we present some methods, some inspired by the works in the field Music Informatics, and establish a baseline for this task.

2.1 Dataset

We used the NMED-T dataset [25]. The dataset captures the EEG of people who are listening to full length recordings of popular music. In contrast to more traditional multimodal EEG and music datasets, such as OpenMIIR[10] which use short samples of the auditory stimuli presented across multiple trials, NMED-T uses a single trial of a full length natural music excerpt, usually between four and five minutes long. A benefit of the dataset is the preprocessed 125-component EEG version available for every song. There are 20 participants who listen to 10 songs in a session. The preprocessed EEG files are sampled

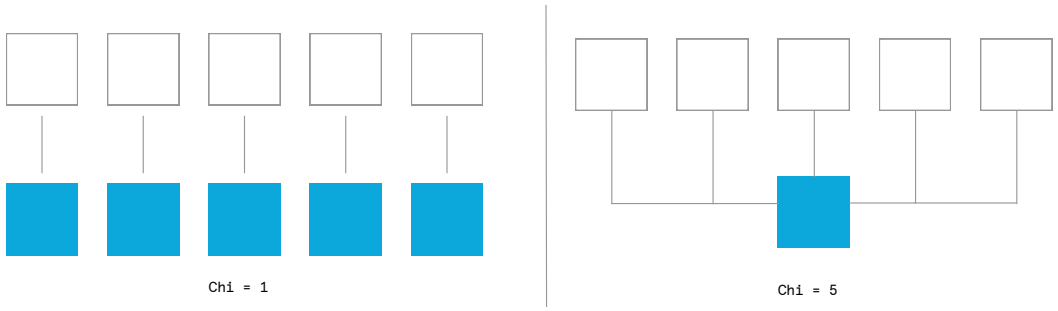

Figure 2.1: How χ behaves. The blank boxes represent ground-truth and the blue boxes represent the reduced variant. $\chi = 1$ is essentially a one to one mapping and shown on the left. As an example, $\chi = 5$ is shown on the right, where five elements in a list get reduced to one.

at 125 Hz and the audio links are provided in the cited paper above. A table containing the song names, amazon ID and duration are in the Appendix.

We split the dataset along a 60-20-20 ratio, where 60% of the data represents the training set and the remainder represent the validation and testing set.

2.1.1 Audio Processing

Since our dataset does not provide ground truth audio onset data, we use the aforementioned MADMOM library [31], which has the onset detection methods as discussed above. MADMOM outputs the onsets in an audio file in seconds. This is converted to the sampling rate of the EEG signal by multiplying the aforementioned output by 125. In effect, we generate a binary sequence across a set of time-steps where we use a "1" to indicate the presence of an onset and a "0" to indicate the lack thereof. Our dataset has a severe imbalance with the onset labels, skewing 99.5%-0.05% for non-onset to onset labels.

For *RNN-context* experiments (which we will discuss in the upcoming section), we introduce a hyperparameter χ, which controls an aggregation length. We define an aggregation as condensing a sub-sequence of length 125 (or one second of data) by scanning window length χ for onsets. If the window contains an onset, the aggregate value for that window is "1", otherwise, it is "0"s.

Table 2.1: Values of χ

χ	Sequence Length
5	25
10	12
25	5
50	2

Values of $\chi > 1$ tend to behave as an aggregation across time. Given that our sampling rate is at 125 Hz, $\chi = 5$ behaves as an aggregation of activity over 40 ms and $\chi = 25$ behaves as an aggregation of activity over 200 ms. By default, $\chi = 1$ does no aggregation.

2.1.2 EEG Processing

Given that the EEG is already pre-processed, our additional processing is limited to a few conveniences. We zero-pad all of our signals to a uniform length of five minutes, or 37,500 samples. We also filter our EEG data using an FIR bandpass filter to get frequencies between 10 and 60 Hz. Filter frequencies were chosen in order to restrict the search space to more clearly focus on β and γ bands.

2.2 Methods

We created models to predict onsets in audio using the EEG data of a person listening to the same. We started by implementing a simple feed-forward network, which we use as a baseline for all other networks we built. We additionally implemented approaches using CNNs and RNNs.

All experiments were conducted under similar conditions:

- We use the ADAM optimizer,
- We use a learning rate of 10^{-3}

- The batch size is 64
- Binary Cross-Entropy logits loss with class weights as our loss function
- ReLU non-linearity between all layers unless specified otherwise.
- Hidden state size for the RNNs is set to 64.

The learning rate and batch size were chosen by running a grid-search on a set of learning rates and batch-sizes. We applied ReLU to the outputs of our multi-layer outputs in order to remove all negative values/outputs of the network. Additionally, our network uses the *Binary Cross Entropy Loss with Logits*. The definition for BCE-Logits loss is provided below.

The input sequence lengths were kept constant at 125 samples (i.e one second) and models were expected to predict a sequence of length 125. The RNN-context model was an exception to this where the model outputs a smaller subsequence, dependent on the hyperparameter χ. Unless specified otherwise, this means that the input sizes to the networks are 125×125, i.e, one second of data. This idea was derived from [7].

For evaluation, we use F-scores, element-wise accuracy and sequence accuracy. We will discuss these in the upcoming evaluation subsection. In order to choose an appropriate number of epochs, we use *early stopping* to stop training when the network does not improve over a span of epochs. Improvement is measured by tracking the improvements in the *cosine similarity* score of the validation set.

We built the networks with PyTorch [32] and implemented hyper parameter searches with RayTune [33].

2.2.1 Feed-forward network

To start, we build a feed-forward network. We convert the 125 channels $\times$ 125 time-step signal into a 1×15625 dimensional vector that goes through two hidden layers with 256

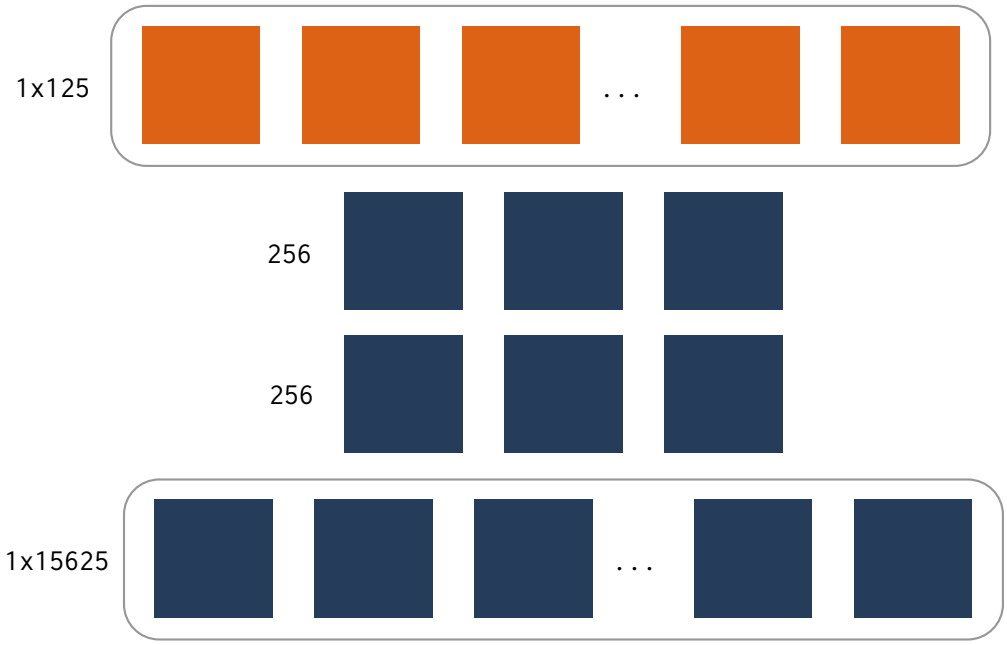

Figure 2.2: Linear Model architecture with an input layer of size 15625, two hidden layers of size 256 and an output layer of size 125

units. The output is a 1×125 sequence that represents the equivalent onset outputs in the audio example. The model architecture is shown in Fig 2.2

Each layer in the feed-forward network takes in an input vector x and applies the transformation $y = W^T x + b$, where y is the output, W is the learnable weights for the layer and b is a bias or affine that is learnable. The weights for each layer are of the size $\text{output\_dim} \times \text{input\_dim}$. For instance, the first layer's weights in Fig 2.2 are of shape 256×15625.

2.2.2 Convolutional Model

This model uses a series of one-dimensional convolution layers followed by a series of transposed convolution layers that produce a 1×125 sequence containing the probabilities for detected onsets. The architecture is shown in Fig 2.3.

We had to choose between one-dimensional and two-dimensional convolutions. Traditionally, for image classification tasks, two dimensional convolutions are used with kernels

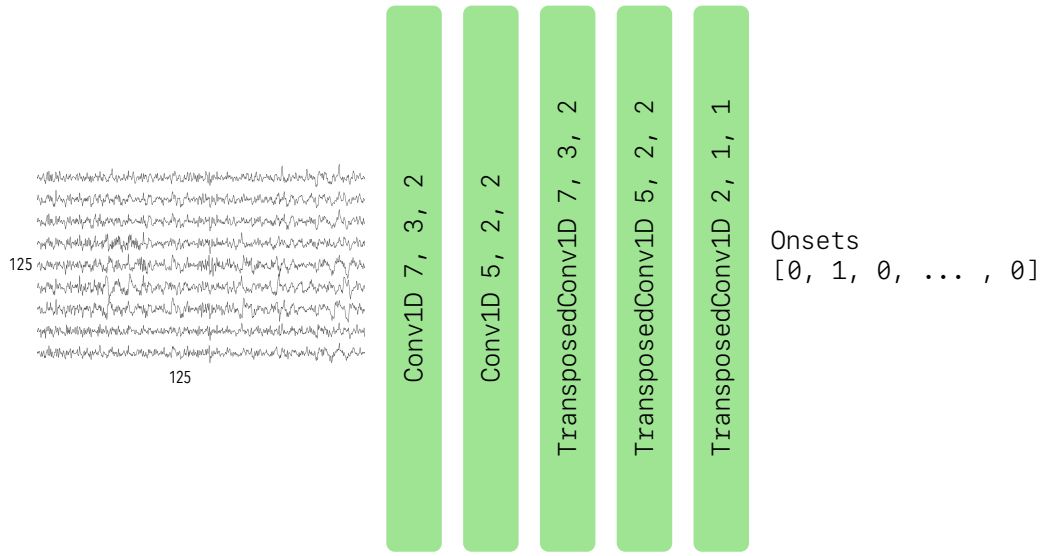

Figure 2.3: The Convolutional Model architecture: It takes a 125 $\times$ 125 input through a series of convolutions and transposed convolutions to create a 1 $\times$ 125 sequence outlining the onsets. Numbers are specified in the format: (kernel size, stride, padding)

that scan across both the width W and height H of the image.

Conversely, in a one-dimensional convolution, the input signal is one-dimensional and is analogous to using a $1 \times N$ filter on a $1 \times W$ image. In effect, one dimensional convolutions on a signal can learn filter maps for a time series. A critical difference is that the two dimensional convolutions assume that the input "image" is spatially related. Since we do not have montage information indicating the positions of the 125 channels, while we can certainly argue that the data is spatially related, we can equally argue that without prior knowledge of the montage we do not know that the relationship is spatially describable. Therefore, we chose to take the safer route and pick the one-dimensional convolutional approach.

We start by using 125 channels in the first layer, which produces 256 filters. The transposed convolutional layers produce 128 filters in the first layer, 64 filters in the second and 1 filter output in the third, representing the 1×125 onset sequence.

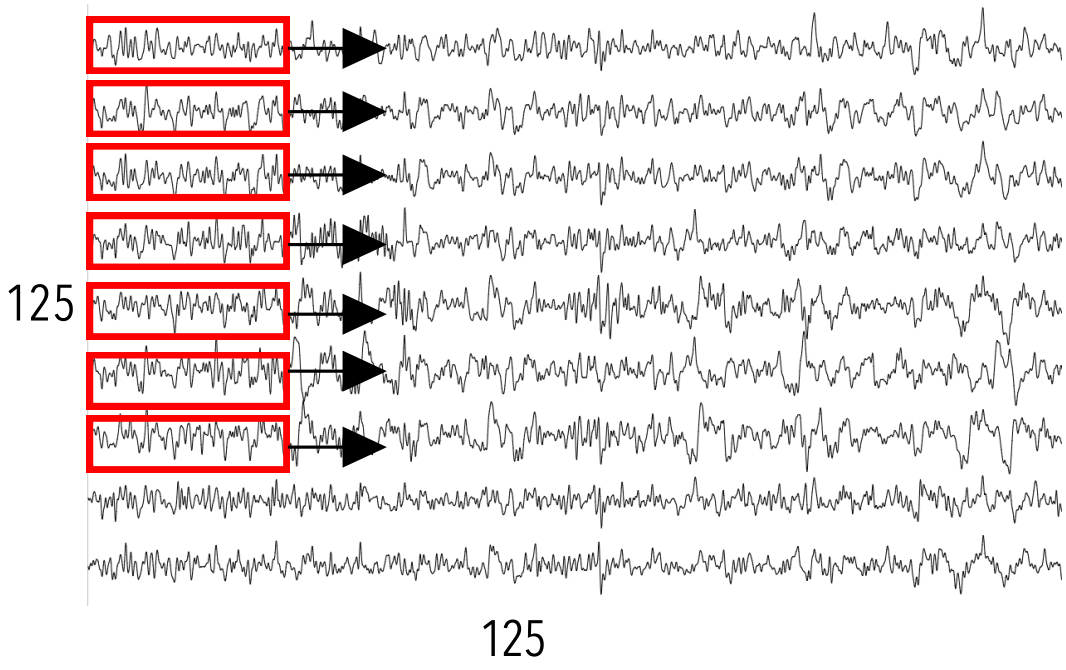

Figure 2.4: What happens in a one-dimensional convolution: The input signal is treated as 125 independent time series and kernel stride across 125 channels of a 1x125 input

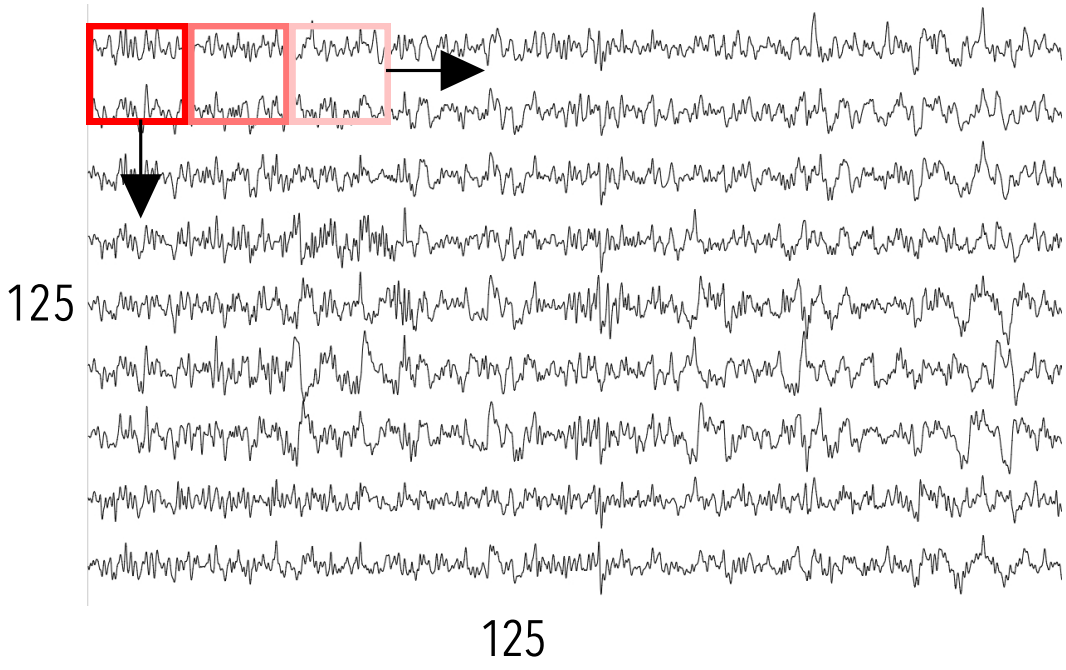

Figure 2.5: What happens in a two-dimensional convolution: The input signal is treated as a 125×125 image and the kernel strides across it

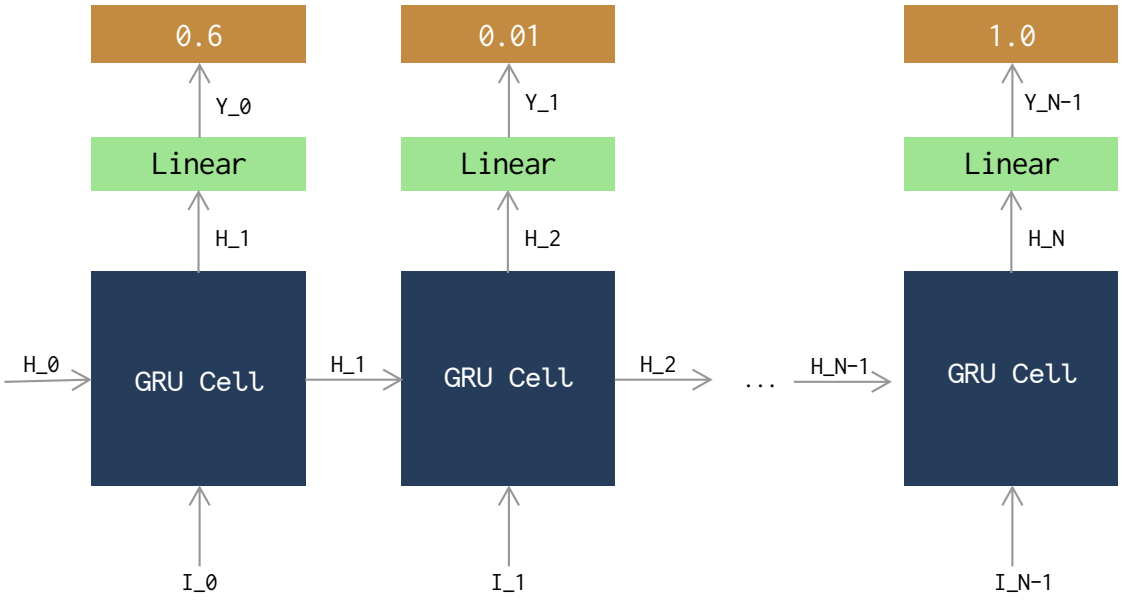

Figure 2.6: Recurrent Neural Network architecture with a many to many mapping. Each step of the network predicts an output for a given input.

Given these options, we chose to work with a one dimensional convolution architecture because we can treat each of the 125 channels as a time series input and learn a series of filters for each channel. The differences in the different convolution models are outlined in Figures 2.4 and 2.5

2.2.3 Recurrent Neural Network

We built a RNN because it is traditionally used for sequence and time-series modeling [34]. While recurrent networks can be configured to predict values across time in multiple ways, we chose to work with a *one-to-one* mapping and a *many-to-one* mapping. A *one-to-one* mapping in an RNN produces an output at every time-step. This is similar to Equation 2.2. Using Equation 2.2, given some hidden state $H(n)$, Input $I(n)$ at time step n, we can write a one to one mapping as:

$$y(n) = W^T H(n+1) + b \tag{2.2}$$

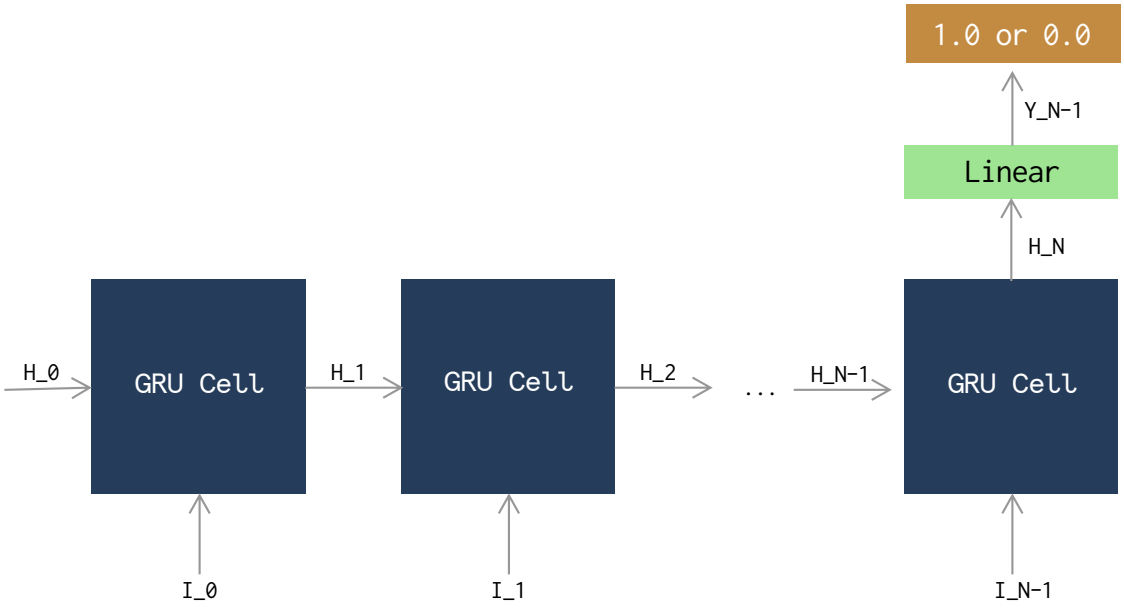

Figure 2.7: Recurrent Network with many to one mapping. One step of the network predicts an output for many input time-steps.

where b is a bias or affine term. This is applied across all time-steps. This architecture is shown in Fig 2.6. This is essentially the same transformation that is described in 2.3.1, but, the dimensions of the hidden state features, H, do not change.

In our network designs we use *Gated Recurrent Units* (GRU). GRU are a variation of a long short term memory unit (LSTM), which is classically described as a cell comprised of the gates - input, output and forget. The implementation in PyTorch for a GRU is a fully-gated network, which is an LSTM without an output and forget gate.

In the many-to-one mapping case, instead of outputting a prediction for every time-step, the network will produce *one* output for *many* time-steps. Typically, this is used for predicting an output at the end of a sequence by applying Equation 2.3 to the last hidden state of the recurrent network. This architecture is shown in 2.7.

In order to predict with an aggregation, we output a value every χ time-steps by using a subsequence reduction method described in the dataset section. For the purposes of this problem the RNN predicts an output every χ time-steps by using the hidden state at $H(\chi)$.

But, the RNN still looks at every time step in the input EEG sequence and hidden states are still propagated across time. We call this model the *RNN-context* network. We initialize the hidden state with zeros. The unrolling method can be described as follows:

The values of χ that we use and the corresponding new sequence lengths are in Table 2.1.

Algorithm 1 RNN Unrolling method

```
outputs = []
for all timesteps do
    current_time_step = 0
    for every timestep within χ do
        Get the input for current time step
        Pass input into RNN with hidden state
        Update hidden state
        Update current timestep
        if last time step of χ then
            Apply Equation 2.3 to the last hidden state to get output
        Append to the outputs array
```

2.2.4 Loss Function

All of our models above minimize the binary cross entropy between the output sequences in a batch and the ground-truth sequences.

Cross Entropy

Cross entropy is a classification-oriented logarithmic likelihood loss that penalizes values that are far away from a given label. In the binary case, given predictions p and labels l, cross entropy is mathematically defined as:

$$l \cdot \log p + (1 - l) \cdot \log(1 - p) \tag{2.3}$$

This can then further be multiplied by a weighting term w:

$$w(l \cdot \log p + (1 - l) \cdot \log(1 - p)) \tag{2.4}$$

PyTorch provides a *BCEWithLogitsLoss* function, that implements the above equation and passes the output labels through a sigmoid σ. This allows us to pass the raw outputs from a network and fit them between zero and one. Applying a sigmoid to Equation 2.4 gives:

$$w(l \cdot \log \sigma(p) + (1 - l) \cdot \log(1 - \sigma(p))) \tag{2.5}$$

Choosing an appropriate weighting term w is crucial and plays a role in how the network learns. Applying a heavy weight on onsets, i.e, the "1" class, tends to cause the network to produce a lot more of the "1" class. This is since the losses for producing a "1" with a heavy weighting for the same is low. Conversely, producing a "0" is penalized heavily. In cases of severe imbalances, it is better to use the skew in the dataset to inform the weighting terms value. In our case, the dataset is highly skewed towards the absence of an onset at a 99.5%:0.05% ratio.

However, when using a thresholded sequence derived above in Algorithm 1, we note that the skew in our data is different. Using a value for $\chi > 1$ creates a smaller ground-truth sequence and a smaller output sequence - at $\chi = 10$, we find that 53% of our data is a "1". In our results, we will discuss various values of χ and how they play a role in the learning process.

2.3 Evaluation Metrics

We use F-scores, cosine similarity and accuracy to evaluate how well our networks learn and how well they can predict values.

2.3.1 Cosine Similarity

We use Cosine Similarity to compute how similar two sequences are to each other. This is done by computing the dot product of the two sequences and normalizing it over the L2-norm of the same. We use the *scikit-learn* implementation of cosine distances [35], where, given two sequence vectors v_1 and v_2, the cosine similarity is given as:

$$f(v_1, v_2) = \frac{v_1 v_2^T}{||v_1||_2^2 ||v_2||_2^2} \tag{2.6}$$

If two sequences are entirely dissimilar, they will have a cosine similarity score of zero. Conversely, exactly similar (or identical) sequences will have a cosine similarity score of one. This is commonly used in natural language processing tasks for identifying similarities between two documents [36].

2.3.2 Accuracy

Accuracy is a simple metric often used to indicate how correct a model is in its predictions. This is done by computing the percentage of correct predictions (true predictions, i.e, the sum of true positives and true negatives) over the number of samples.

$$\text{acc} = \frac{\text{True-Positives} + \text{True-Negatives}}{\text{N}} \tag{2.7}$$

The value of N can be written as the sum:

$$\text{True-Positives} + \text{True-Negatives} + \text{False-Negatives} + \text{False-Positives} \tag{2.8}$$

In the case of highly-skewed datasets, evaluating models by accuracy alone can be misleading. As an example, we can use a binary-class dataset that has a 90%:10% skew and we train a network to predict between the two classes. A network that predicts all zeros will get a 90% accuracy score. While a 90% accuracy is good on paper, it misleads us into

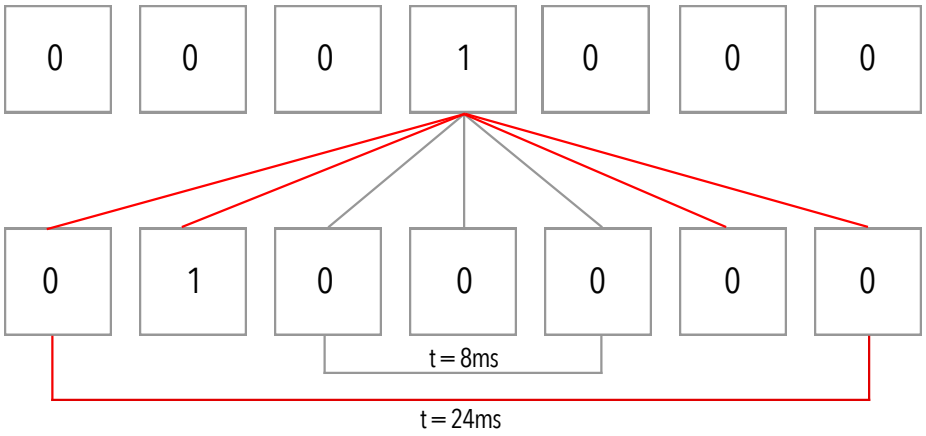

Figure 2.8: How a tolerance window works. At 8 ms, the tolerance looks at either side of the onset. 24 ms allows us to look at 3 samples behind and ahead of an onset.

thinking that the network performs well. We use this to verify how well our network does.

For our model evaluation, we report element-wise accuracy (the two sequences are required to be exactly the same) and implement a tolerance accuracy metric where, given an onset in the ground truth, we look at the predictions around a threshold value and adjust our predictions accordingly to reflect this tolerance.

In order to compute the tolerance accuracy metric, we look at the ground truth sequence and pick the onsets. As an example, let's say that in our 125 length sequence, our network is expected to predict an onset at sample 12. With a tolerance of 8 ms, we are looking at either side of 12, i.e 11 and 13 in our predictions to see if our network predicted an onset in those locations. If there is an accurate prediction within the span of 11, 12, 13, we count that as an accurate prediction. A visual example of this is shown in Fig 2.8.

2.3.3 F-score, precision and recall

F-score is a standard metric to better measure the accuracy of a classification model. F-scores are a harmonic mean of the precision and recall measures for a classifier. Precision

(p) is defined as the number of correct predictions over the total amount of data that is present.

$$p = \frac{\text{True-Positives}}{\text{True-Positives} + \text{False-Positives}} \tag{2.9}$$

Recall (r) is defined as the number of correct predictions over all of the predictions that were supposed to be correct.

$$r = \frac{\text{True-Positives}}{\text{True-Positives} + \text{False-Negatives}} \tag{2.10}$$

The F-measure of a model is the harmonic mean of the precision and recall is defined as:

$$F = \frac{2\,\mathrm{p} \cdot \mathrm{r}}{\mathrm{p} + \mathrm{r}} \tag{2.11}$$

A score of one indicates perfect precision and recall. A score of zero, conversely denotes a poor performance.

In addition to computing the precision and recall by sequences, we also compute the precision and recall performance by class, i.e, “0” and “1”. This allows us to understand the performance of the network individually for both classes and evaluate if the network accurately learns the objective.

We use the macro variant of the F-score, which uses the F-score of each class and averages them.

CHAPTER 3

RESULTS

In this chapter, we present the results for each network and present an overall outline of results at the end. The outline of results can be seen in Table 3.1.All of these metrics are computed on the test set. It must be noted that sequences with all-zeros were not used for computing these results.

We are using w as a short hand for class weights and t for tolerance window.

We report three different tables - Table 3.2 reports the F-scores, cosine similarity and accuracy scores for each network. Table 3.3 reports the F-scores for the onset and non-onset events in our sequence. Table 3.4 breaks the F-scores for onset and non-onset events down further into its precision and recall values.

3.1 Class Weights

With the exception of the *RNN-context* networks, we train our networks with weighting for each class. At class weight 1 (i.e, there is no priority given to either class), across the board, networks tended to output a zero. This gives them a fairly high element-wise accuracy scores throughout, but, poor cosine score performance. While using the likelihood of zero in a sequence length of 125, i.e 124.3, we found that the loss term tended to prioritize the production of "1", or an onset. This lowered the accuracy scores significantly and lowered the F-scores significantly. This result is also visible in the precision and recall breakdown.

When increasing the values of χ, we kept the class weights at 1, because the class imbalance eases up as we increase the window size.

Table 3.1: Table outlining all results

Network	**Class Weights** (w)	**Tolerance Window** (t)	**Sequence Length**	**Aggregation**
Linear	1	0	125	8ms
Linear	1	8	125	8ms
Linear	1	24	125	8ms
Linear	1	40	125	8ms
Linear	124.3	0	125	8ms
Conv	1	0	125	8ms
Conv	1	8	125	8ms
Conv	1	24	125	8ms
Conv	1	40	125	8ms
Conv	124.3	0	125	8ms
RNN	1	0	125	8ms
RNN	1	8	125	8ms
RNN	1	24	125	8ms
RNN	1	40	125	8ms
RNN	124.3	0	125	8ms
RNN-context	1	0	25 ($\chi = 5$)	40ms
RNN-context	1	0	12 ($\chi = 10$)	96ms
RNN-context	1	0	5 ($\chi = 25$)	200ms

Table 3.2: F-score, Cosine Similarity and Accuracy Metrics

Network	**Seq-Length**	**F-score**	**Cosine Similarity**	**Accuracy**
Linear (w=1, t=0ms, $\chi = 1$)	125	0.529443	0.007126	0.950250
Linear (w=1, t=8ms, $\chi = 1$)	125	0.527218	0.002477	0.950135
Linear (w=1, t=24ms, $\chi = 1$)	125	0.530756	0.0.00993	0.950747
Linear (w=1, t=40ms, $\chi = 1$)	125	0.534335	0.017376	0.951344
Linear (w=124.3, t=0ms, $\chi = 1$)	125	0.263365	0.230234	0.279490
Conv (w=1, t=0ms, $\chi = 1$)	125	0.527655	0	0.951515
Conv (w=1, t=8ms, $\chi = 1$)	125	0.527655	0	0.951515
Conv (w=1, t=24ms, $\chi = 1$)	125	0.527655	0	0.951515
Conv (w=1, t=40ms, $\chi = 1$)	125	0.527655	0	0.951515
Conv (w=124.3, t=0ms,$\chi = 1$)	125	0.130816	0.2	0.08
RNN (w=1, t=0ms, $\chi = 1$)	125	0.527655	0	0.951515
RNN (w=1, t=8ms, $\chi = 1$)	125	0.527655	0	0.951515
RNN (w=1, t=24ms, $\chi = 1$)	125	0.527655	0	0.951515
RNN (w=1, t=40ms, $\chi = 1$)	125	0.527655	0	0.951515
RNN (w=124.3, t=0ms, $\chi = 1$)	125	0.091147	0.20	0.094336
RNN-context (w=1, t=0ms, $\chi = 5$)	25	0.469146	0.016046	0.757832
RNN-context (w=1, t=0ms, $\chi = 10$)	12	0.547260	0.410340	0.606812
RNN-context (w=1, t=0ms, $\chi = 25$)	5	0.762023	0.831227	0.858378

Table 3.3: F-score breakdown for onset and no onset

Network	**No Onset**	**Onset**
Linear (w=1, t=0ms, $\chi = 1$)	0.974337	0.006507
Linear (w=1, t=8ms, $\chi = 1$)	0.974283	0.002112
Linear (w=1, t=24ms, $\chi = 1$)	0.974605	0.008866
Linear (w=1, t=40ms, $\chi = 1$)	0.974918	0.015710
Linear(w=124.3, $\chi = 1$)	0.370733	0.111947
Conv (w=1, t=0ms, $\chi = 1$)	0.894739	0.000000
Conv (w=1, t=8ms, $\chi = 1$)	0.894739	0.000000
Conv (w=1, t=24ms, $\chi = 1$)	0.894739	0.000000
Conv (w=1, t=40ms, $\chi = 1$)	0.894739	0.000000
Conv (w=124.3, $\chi = 1$)	0.046632	0.091613
RNN (w=1, t=0ms $\chi = 1$)	0.975024	0.000000
RNN (w=1, t=8ms $\chi = 1$)	0.975024	0.000000
RNN (w=1, t=24ms $\chi = 1$)	0.975024	0.000000
RNN (w=1, t=40ms $\chi = 1$)	0.975024	0.000000
RNN (w=124.3, $\chi = 1$)	0.09	0.097
RNN-context (w=1, $\chi = 5$)	0.857544	0.010996
RNN-context (w=1, $\chi = 10$)	0.659836	0.382555
RNN-context (w=1, $\chi = 25$)	0.199903	0.824662

3.2 Tolerance windows

We found in our tests that the networks that were evaluated with a tolerance window yielded improvements only on the feed-forward network, where each metric consistently improved through and through. However, we found that in the case of our RNN and Convolution networks, they did not yield better results. Upon closer inspection of these networks, we noted that the network produced all zeros on our testing set.

3.3 Final results

Overall, we found that the RNN-context network with χ set to 25 yielded highest metrics across the board.

Table 3.4: Precision(P) and Recall(R) breakdown for onset and no onset

Network	**Onset-P**	**Onset-R**	**No-Onset-P**	**No-Onset-R**
Linear (w=1, t=0ms, $\chi = 1$)	0.010318	0.005535	0.951736	0.998349
Linear (w=1, t=8ms, $\chi = 1$)	0.004291	0.001712	0.951535	0.998452
Linear (w=1, t=24ms, $\chi = 1$)	0.014942	0.007829	0.951855	0.998776
Linear (w=1, t=40ms, $\chi = 1$)	0.025266	0.013972	0.952177	0.999079
Linear(w=124.3, $\chi = 1$)	0.060037	0.915617	0.998846	0.241322
Conv (w=1, t=0ms, $\chi = 1$)	0.0	0.0	0.951515	1.000000
Conv (w=1, t=8ms, $\chi = 1$)	0.0	0.0	0.951515	1.000000
Conv (w=1, t=24ms, $\chi = 1$)	0.0	0.0	0.951515	1.00000
Conv (w=1, t=40ms, $\chi = 1$)	0.0	0.0	0.951515	1.000000
Conv (w=124.3, $\chi = 1$)	0.052	0.9986	0.9991	0.08
RNN (w=1, t=0ms, $\chi = 1$)	0.0	0.0	0.951515	1.000000
RNN (w=1, t=8ms, $\chi = 1$)	0.0	0.0	0.951515	1.000000
RNN (w=1, t=24ms, $\chi = 1$)	0.0	0.0	0.951515	1.000000
RNN (w=1, t=40ms, $\chi = 1$)	0.0	0.0	0.951515	1.000000
RNN (w=124.3, $\chi = 1$)	0.000000	0.000000	0.9991	0.08
RNN-context (w=1, $\chi = 5$)	0.041099	0.006536	0.758507	0.997949
RNN-context (w=1, $\chi = 10$)	0.580874	0.818284	0.572859	0.315125
RNN-context (w=1, $\chi = 25$)	0.842279	0.834656	0.214334	0.205012

CHAPTER 4
DISCUSSION

We looked at methods to do audio onset detection using EEG. The problem is not simple, since we are looking at a fairly noisy [8] input signal and trying to extract a feature in the audio signal. Here, we would like to discuss a few of the results and things we would like to implement going forward.

A primary issue we notice is with class weighting. At a class weighting of 1, most non-aggregate networks performed poorly and 124.3 did not provide a better result. This is generally a problem that one would face if the datasets are heavily imbalanced. Finding a more appropriate class weighting is difficult and could potentially be done by using a searching algorithm. However, this prominently highlights the issues with dealing with heavily imbalanced datasets.

In our analysis, we found that the feed-forward network outperformed the other networks (barring the context network) at parity weighting (or a class weighting of 1). This is further shown at various tolerance windows, where the performance of the feed-forward models improve, but, the other networks remain stagnant. By and large, at parity weightings, surprisingly, our feed-forward networks was able to predict more onsets and benefited from using tolerance window based analysis.

However, we could not use the same tolerance window method to analyze the networks trained on a class weighting of 124.3, because we found that the networks often predicted an onset, even when the safest prediction at a given time step would be to predict an non-onset. This resulted in a network that predicted a whole series of ones across 125 samples. This made analyzing with a tolerance window a lot more difficult, since we could not often tell if a network was indeed predicting an onset because it believed that there was on onset in a tolerance window, or if there was nothing.

We worked on using aggregations as a thresholding strategy. We built a RNN capable of predicting across aggregations of time-steps defined by a hyperparameter χ. We reported results of aggregations at 5 samples, 10 samples and 25 samples.

Across the board, we found that the networks which used values of $\chi > 1$ often reported significantly lower "no-onset" precision and recall performances, while increasing the scores on onsets and F-scores. We believe that this is because of how aggregations work. Given that we are aggregating across a window of size χ, we note that the dataset accordingly increases in the number of predictable onsets.

An easy experimental improvement would involve using a hop size that overlaps with the current window at some percentage of the window size. By sliding, we are guaranteed that all values can be used in aggregation. Using an overlapping window should allow for values of χ that are not perfect divisors to stride the sequences and not miss the onsets, as might have been the case at $\chi = 10$.

We found that $\chi = 25$ yielded impressive metrics for accuracy and F-scores. We found that only 19.1% of our data had no onset. Given that there are more onsets to predict in an aggregation of 25 samples, it is logical that it would present results with better onset precision and recall and worse non-onset precision and recall.

Crucially, this highlights the difference between aggregating and tolerances. While tolerances do not rely on predicting a single value for a collection of data and can be evaluated at any precision, aggregations tend to behave as a consolidated dataset. Aggregations allow for the problem complexity to decrease for the networks, which have to eventually predict shorter sequences. However, tolerance windows are a better way to evaluate performance, because they tend to give some leeway for the network to make errors.

CHAPTER 5
CONCLUSION

We present models that attempt to extract onsets in audio using the physiological response of a person listening to the audio. While the networks do not perform perfectly, we are presenting these networks as a starting point for research on this class of problems at the intersection of music, machine learning and physiological data. Our experiments find that networks that predict a value across an aggregation of time steps tend to perform better than ones that predict at a sample to sample level. Additionally, our key conclusions are as follows:

- Our convolutional and recurrent networks need some work in order to be good. Having networks that predict all zeros could be addressed with better architectural choices.
- Aggregations solve some imbalance issues, but, sliding windows may help attain a more balanced dataset.
- Aggregating networks (RNN-context) perform better than our feed-forward network. However, they likely work better because of lowered complexity of the model since they have to predict shorter sequences.

We are equally cognizant of the fact that the network was trained on a relatively small dataset, compared to works in the image or natural language domain. We are hoping that in order to address that, in the future, we can combine heterogenous datasets and compose a larger dataset to address this specific issue and hopefully compile a benchmarking test for networks across datasets and features.

Going forward, we would like to try the following:

- Longer input sequences - 250 time steps or more.
- Spectral representations for each input EEG signal.
- Amplitude Envelope prediction - as was used in the Ofner and Stober paper [7],
- Attempt to learn other musical features, such as beat, melody and harmony.

Our hope is that this work will motivate and provide benchmarks for future research in this area.

Appendices

APPENDIX A

NMED-T SONG INFORMATION

#	Song Title	Artist	ASIN	min:sec
1	”First Fires”	Bonobo	B00CIE73J6	4:38
2	”Oino”	LA Priest	B00T4NHS2W	4:31
3	”Tiptoes”	Datdeliss	B011SAZRLC	4:36
4	”Careless Love”	Croquet Club	B06X9736NJ	4:54
5	”Lebanese Blonde”	Thievery Corporation	B000SF16MI	4:49
6	”Canopée’	Polo & Pan	B01GOL4IB0	4:36
7	”Doing Yoga”	Kazy Lambist	B01JDDVIQ4	4:52
8	”Until the Sun Needs to Rise”	Rufus du Sol	B01APT6JKA	4:52
9	”Silent Shout”	The Knife	B00IMN40O4	4:54
10	”The Last Thing You Should Do”	David Bowie	B018GS2A46	4:58

REFERENCES

[1] A. Lerch, *An introduction to audio content analysis: Applications in signal processing and music informatics*. Wiley-IEEE Press, 2012.

[2] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long-short term memory neural networks," in *Proc. 11th Intern. Soc. for Music Information Retrieval Conference, ISMIR, Utrecht, The Netherlands*, 2010, pp. 589–594.

[3] J. Schlüter and S. Böck, "Musical onset detection with convolutional neural networks," in *6th international workshop on machine learning and music (MML), Prague, Czech Republic*, 2013.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[5] A. van den Oord, Y. Li, and O. Vinyals, *Representation learning with contrastive predictive coding*, 2018. arXiv: 1807.03748 [cs.LG].

[6] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, *Learning deep representations by mutual information estimation and maximization*, 2018. arXiv: 1808.06670 [stat.ML].

[7] A. Ofner and S. Stober, "Shared Generative Representation of Auditory Concepts and EEG to Reconstruct Perceived and Imagined Music," *19th International Society for Music Information Retrieval Conference – ISMIR 2018*, pp. 392–399, 2018.

[8] S. Stober, D. J. Cameron, and J. A. Grahn, "Using convolutional neural networks to recognize rhythm stimuli from electroencephalography recordings," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014, pp. 1449–1457.

[9] S. Stober, T. Pratzlich, and M. Muller, "Brain Beats: Tempo Extraction from EEG Data," in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, New York, NY, 2016.

[10] S. Stober, A. Sternin, A. M. Owen, and J. A. Grahn, "Towards Music Imagery Information Retrieval: Introducing the OpenMIIR Dataset of EEG Recordings from

Music Perception and Imagination," *16th International Society for Music Information Retrieval Conference (ISMIR'15)*, 2015.

[11] M. Tudor, L. Tudor, and K. I. Tudor, "[hans berger (1873-1941)–the history of electroencephalography]," *Acta medica Croatica : casopis Hravatske akademije medicinskih znanosti*, vol. 59, no. 4, 307—313, 2005.

[12] H. P. S. TEGLBJÆRG, "On musciogenic epilepsy," *Acta Psychiatrica Scandinavica*, vol. 24, no. 3-4, pp. 679–690, 1949. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1600-0447.1949.tb07349.x.

[13] J. L. Walker, "Alpha eeg correlates of performance on a music recognition task," *Physiological Psychology*, vol. 8, no. 3, pp. 417–420, 1a980.

[14] I. Daly, A. Malik, F. Hwang, E. Roesch, J. Weaver, A. Kirke, D. Williams, E. Miranda, and S. J. Nasuto, "Neural correlates of emotional responses to music: An eeg study," *Neuroscience letters*, vol. 573, pp. 52–57, 2014.

[15] T. Field, A. Martinez, T. Nawrocki, J. Pickens, N. A. Fox, and S. Schanberg, "Music shifts frontal eeg in depressed adolescents," *Adolescence*, vol. 33, no. 129, pp. 109–117, 1998.

[16] K. Trochidis and E. Bigand, "Investigation of the effect of mode and tempo on emotional responses to music using eeg power asymmetry," *Journal of Psychophysiology*, 2013.

[17] R. S. Schaefer, R. J. Vlek, and P. Desain, "Music perception and imagery in eeg: Alpha band effects of task and stimulus," *International Journal of Psychophysiology*, vol. 82, no. 3, pp. 254–259, 2011.

[18] S. Martin, C. Mikutta, M. K. Leonard, D. Hungate, S. Koelsch, S. Shamma, E. F. Chang, J. d. R. Millán, R. T. Knight, and B. N. Pasley, "Neural encoding of auditory features during music perception and imagery," *Cerebral Cortex*, vol. 28, no. 12, pp. 4222–4233, 2018.

[19] T. Fujioka, L. J. Trainor, E. W. Large, and B. Ross, "Beta and gamma rhythms in human auditory cortex during musical beat processing," *Annals of the New York Academy of Sciences*, vol. 1169, no. 1, pp. 89–92, 2009. eprint: https://nyaspubs.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1749-6632.2009.04779.x.

[20] H. Akbari, B. Khalighinejad, J. L. Herrero, A. D. Mehta, and N. Mesgarani, "Towards reconstructing intelligible speech from the human auditory cortex," *Scientific Reports*, vol. 9, no. 1, p. 874, 2019.

[21] P. Schneider, M. Scherg, H. G. Dosch, H. J. Specht, A. Gutschalk, and A. Rupp, "Morphology of heschl's gyrus reflects enhanced activation in the auditory cortex of musicians," *Nature neuroscience*, vol. 5, no. 7, pp. 688–694, 2002.

[22] R. S. Schaefer, "Measuring the Mind's Ear: EEG of Music Imagery," PhD Thesis, Radboud University Nijmegen, 2011, ISBN: 9789491027116.

[23] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics.," in *ISMIR*, Citeseer, 2012, pp. 403–408.

[24] J. Ramírez and M. J. Flores, "Machine learning for music genre: Multifaceted review and experimentation with audioset," *Journal of Intelligent Information Systems*, pp. 1–31, 2019.

[25] S. Losorelli, D. T. Nguyen, J. Dmochowski, and B. Kaneshiro, "Nmed-t: A tempo-focused dataset of cortical and behavioral responses to naturalistic music," in *ISMIR*, 2017.

[26] S. Stober, A. Sternin, A. M. Owen, and J. A. Grahn, "Deep feature learning for EEG recordings," *CoRR*, vol. abs/1511.04306, 2015. arXiv: 1511.04306.

[27] B. Kaneshiro and J. P. Dmochowski, "Neuroimaging methods for music information retrieval: Current findings and future prospects.," in *ISMIR*, 2015, pp. 538–544.

[28] W. Wang, X. Yan, H. Lee, and K. Livescu, "Deep Variational Canonical Correlation Analysis," vol. 1, 2016. arXiv: 1610.03454.

[29] C. Doersch, *Tutorial on variational autoencoders*, 2016. arXiv: 1606.05908 [stat.ML].

[30] S. Tripathi, S. Acharya, R. D. Sharma, S. Mittal, and S. Bhattacharya, "Using deep and convolutional neural networks for accurate emotion classification on deap dataset.," in *Twenty-Ninth IAAI Conference*, 2017.

[31] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proceedings of the 24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, Oct. 2016, pp. 1174–1178.

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F.

d Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[33] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

[34] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 240–254, 1994.

[35] *6.8. pairwise metrics, affinities and kernels¶*.

[36] L. Muflikhah and B. Baharudin, "Document clustering using concept space and cosine similarity measurement," in *2009 International Conference on Computer Technology and Development*, IEEE, vol. 1, 2009, pp. 58–62.

[33] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 240–254, 1994.

[34] *6.8. pairwise metrics, affinities and kernels*¶.

[35] L. Muflikhah and B. Baharudin, "Document clustering using concept space and cosine similarity measurement," in *2009 International Conference on Computer Technology and Development*, IEEE, vol. 1, 2009, pp. 58–62.