BACHELOR THESIS
ARTIFICIAL INTELLIGENCE

Radboud University

Decoding of spoken words in a spiking neural network

Author:

Suzanne Snoek
s1035431
suzanne.snoek@ru.nl

First supervisor:

Dr. H. Fitz
Dpt. Of Artificial Intelligence, Donders Institute
h.fits@donders.ru.nl

A. Quaresima
Max Planck Institute
alessio.Quaresima@mpi.nl

Second supervisor:

F.E. Olalere
Dpt. Of Artificial Intelligence, Donders Institute
feyisayo.olalere@ru.nl

August 18th 2022

Abstract

Speech is the most common mode of human communication and thus this is studied. Word recognition is a start to speech recognition in the brain. A spiking neural network is used in this project to conduct a word application task. The project's goal was to gain a better understanding of an optimal decoding pipeline for maximizing the accuracy of a word recognition task that receives audio input. The presented decoding pipeline has multiple components. First, the received network features are various variations of spike and state feature. State features consist of the membrane potential and adaptive current. Principal Component Analysis (PCA) was used for dimensionality reduction. Following the application of four classifiers: Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine, the classification for both words and phones can be evaluated. This resulted in a maximum k-score of about 85% for words and 60% for phones. The computation time varied from a few seconds to about an hour, depending on the classifier, and was considered when deciding the best classifier. Overall, the best pipeline for a word recognition task to maximize accuracy uses a 'non-averaged feature space' with PCA. Afterwards, Random Forest classification is applied. The study showed that state features outperformed spike features. This is interesting since even though state features relate to spiking features, spiking neurons are the foundation of a spiking neural network. Overall, the findings of this project, when combined with research on other components of the spiking neural network, contribute to an overall improved spiking neural network for spoken words.

Keywords: spiking neural network, word recognition, decoding, classification, computation time

Table of Contents

1. Introduction

Language comprehension is a concept that most people use on a regular basis in their daily lives. Understanding the domains of both written and spoken communication requires the decoding of projected acoustic or printed features to create mental representations of meaning, events, and situations (Steen & Stine-Morrow, 2016). In the concept of language comprehension, the human brain maps a spectro-temporal speech signal onto categorical features such as phonemes. These features are used to construct larger and more meaningful units in the brain, such as words, syllables, and phrases.

Models for speech recognition applications are consistently applied in daily life. In speech recognition, spoken sentences are recognised and translated to text of a different data representation (Dominguez-Morales et al., 2018). Examples include voice-activated security systems and virtual assistants like Siri or Alexa. Most current conventional speech recognition algorithms are based on ANN applications, despite having limited biological similarity to the brain. Currently, it is unclear how the brain extracts information and can combine these features across place and time to aid in speech recognition. With speech being the most common mode of human communication, this implies relevance for research on this topic (Pan et al., 2020). My research is part of a larger project that is investigating and developing a model for spoken word recognition. This is accomplished by creating a spiking neural network, which is a simulated neurobiologically motivated network.

Similarly, to spiking neurons, a spiking neural network (SNN) fires spikes when a certain threshold is reached. It is inspired by event-based computation, can operate in real-time, and is made up of realistic synapse models (Wu et al., 2020). Spiking models, due to their stronger biological evidence, provide powerful tools for analysing certain brain processes such as neural information processing and learning (Ponulak & Kasiski, 2011). SSNs have also been shown to be a promising research direction for a word recognition task (Pan et al., 2020).

To optimize a model, the entire pipeline in a network must be investigated. This includes encoding and feature representation, as well as plasticity and decoding. Classification can be used as an inspection tool in this pipeline for human word recognition. It is used to determine whether the network's learned representation can generalize to new input, such as other dialects or gender, or whether certain decoding methods perform better than others. The focus of my project is on optimizing the decoding of human word recognition. The decoding process includes several steps, beginning with network features retrieved from the network. The possibility of dimensionality reduction is investigated based on these features. In addition, various classifiers and evaluation metrics will be used. My thesis project is centred on these various steps and potential experimental manipulations. Given that classification can be used as an evaluation tool, this research focuses on improving the evaluation tool for human word recognition and determining the best decoding or classification pipeline, with the purpose to provide insight into what the network is doing. This leads to the following research question:

Given the features that can be extracted from a running spiking neural network, what decoding pipeline maximizes the accuracy of a word recognition task?

There is no defined distinguishing hypothesis because this research is exploratory in nature. To elaborate on the necessary theory for the topic, I will first go into the Preliminaries, after which the Methodology provides an overview of the methods and materials that were employed. The findings of the paper will be presented along graphs and tables the Results section. In the Discussion I will go into the implications and challenges of my findings, as well as elaborate of possible directions for further research. Lastly, the Conclusion will provide an answer to the research question.

2. Preliminaries

A basic implementation of a spiking neural network was provided at the start of my project. The project continued on a provided network, which is adapted from the work of Litwin-Kumar & Doiron (2014), with an AdEx neural activity model (Brette & Gerstner, 2005). The fundamentals needed to understand this project are explained. In addition, some key terms related to word recognition and spiking neurons are defined.

2.1 Neurons

Neurons are the basic building blocks of the brain and nervous system. They use electrical and chemical signals to transmit information between different regions of the brain or the nervous system. A neuron is made up of three parts: a cell body, an axon, and a dendrite (Gerstner et al., 2014). The cell body controls the activities and genetic material of a cell. The axon can send information from the cell, whereas dendrites can receive information from the cell (National Institute of Neurological Disorders and Stroke, z.d.). Neurons conduct a variety of functions, including the ability to integrate incoming signals and communicate them to other neurons. Furthermore, they can receive sensory input from the outside world and control our muscles via brain signals. The synapse is the location where an axon of a presynaptic neuron gets to contact the dendrite of a postsynaptic cell, such that an electrical or chemical signal can be transmitted from one neuron to another. (Gerstner et al., 2014; National Institute of Neurological Disorders and Stroke, z.d.). Neurotransmitters are the chemical messengers secreted by a neuron to affect another cell across a synapse. The three main classes of neurotransmitters are excitatory, inhibitory and neuromodulators, such as Dopamine, Endorphins and Serotonin (Hyman, 2005). These neurotransmitters are the basis for the distinction between inhibitory and excitatory neurons. Excitatory neurons are neurons with excitatory currents that prompt one neuron to share information with the next through an action potential (Ippolito, 2020). While inhibitory neurons reduce the likelihood of such a transfer occurring. The balance of inhibitory and excitatory currents is essential for a stable brain function (Ippolito, 2020).

As previously stated, information processing in cells can take place via chemical and electrical signalling. The membrane potential is the electrical potential between the interior and exterior of a cell. Neuronal activity on the cell causes the potential to rise (depolarisation) or decrease (repolarisation) (Gerstner et al., 2014). When the membrane potential exceeds a certain threshold, which is greater than the resting potential, the neuron produces an action potential, as shown in Figure 1. (Gerstner et al., 2014). This action potential is also referred to as a spike.

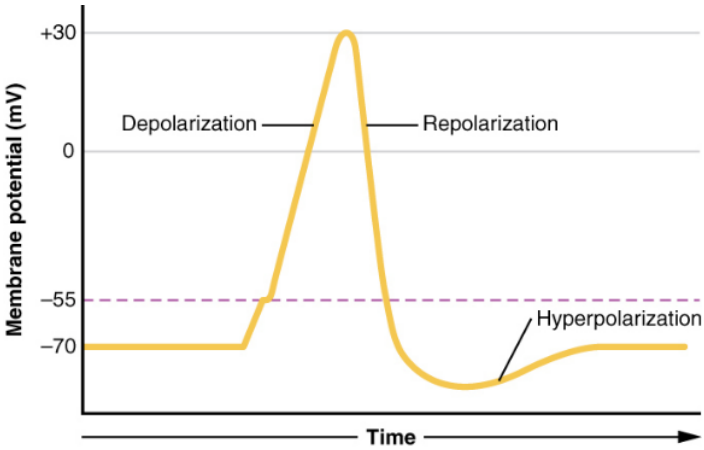


Figure 1 Representation of action potential (spike) after reaching threshold of -55 mV (Chen & Lui, 2011)

2.1 AdEx model

The model that is used to describe the neural activity in this network is an Adaptive Exponential Integrate-and-Fire model, shorthand the AdEx model (Brette & Gerstner, 2005). This model is an enhanced version of the traditional Leaky Integrate-and-Fire (LIF) model. A traditional Leaky Integrate-and-Fire model combines a linear differential equation for describing a membrane potential, with a strict voltage threshold for spike firing (Gerstner et al., 2014). The AdEx model differs from the LIF model in that it fits a non-linear term in the membrane differential equation. As previously stated, the membrane potential is the electrical potential between the interior and exterior of a cell. Moreover, the AdEx model is an adaptive model. As a result, a second differential equation is created that accounts for continuous firing and keeps a short-term memory of previous activity (Brette & Gerstner, 2005). This is known as the adaptive current. The adaptive current has a longer timescale than the membrane potential and can store data for a longer time interval. Figure 2 illustrates an example of membrane potential and adaptive current representation over time. The slower decreasing peaks in the adaptive current account for the 'short term memory' that is being kept.

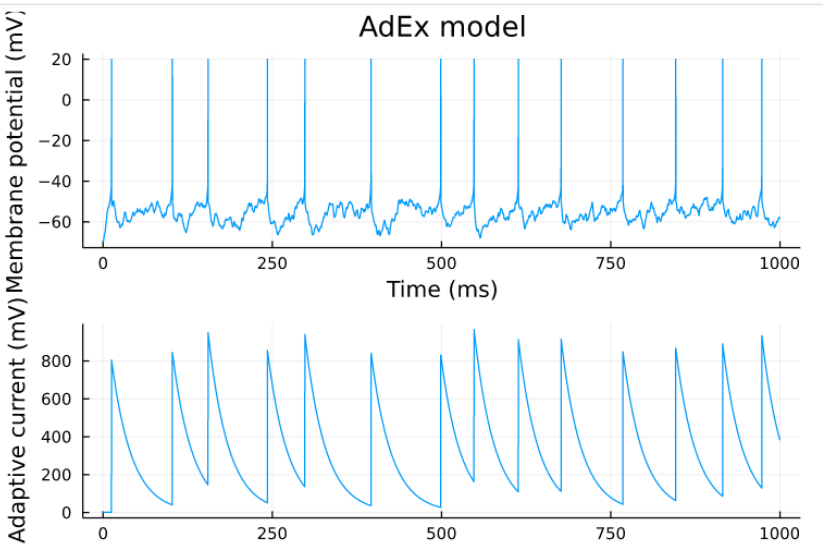


Figure 2 Membrane potential and Adaptive current (Quaresima, 2022)

2.2 Word recognition task

To understand an utterance such as "Good morning, how are you?" a person must be able to recognize the individual words in the utterance. Before the meaning of sentences and speech recognition as a whole can be understood, this recognition of words is inevitable. A word recognition task is the focus of this study. The emphasis is on mapping the input words and phones, with no regard for the meaning of words or phones. Furthermore, several factors complicate mapping spoken word language. To begin, certain words are similar to one another (Weber & Scharenborg, 2012). Some words will inevitably be similar because languages build large vocabularies from a limited set of phonemes. Furthermore, speech is highly variable for a variety of reasons. Each speaker has a distinct speaking rate, style, and word sound. Finally, speech is time-distributed and continuous. Individual words in an acoustic speech signal have no distinct boundaries, and some words may be embedded in others, if a speaker barely pauses between speaking words (Weber & Scharenborg, 2012). These factors may be the source of difficulties in optimizing a word recognition task in a network. As words are made up of a set of phones, my thesis focuses on both word and phone recognition. To clarify, phones and phonemes have different meaning. A phoneme is a sound that can support lexical contrasts in a particular language (Li et al., 2020). While their corresponding phones are the actual spoken sounds, which are language independent (Li et al., 2020).

3. Methodology

3.1 Schematic network structure

Figure 3 depicts a schematic representation of the entire spiking neural network pipeline. Each element in the network has a purpose which altogether results in the auditory input being outputted for classification evaluation. To provide a baseline understanding of the network, a brief overview of each component will be provided. Following that, each element will be discussed in greater detail.

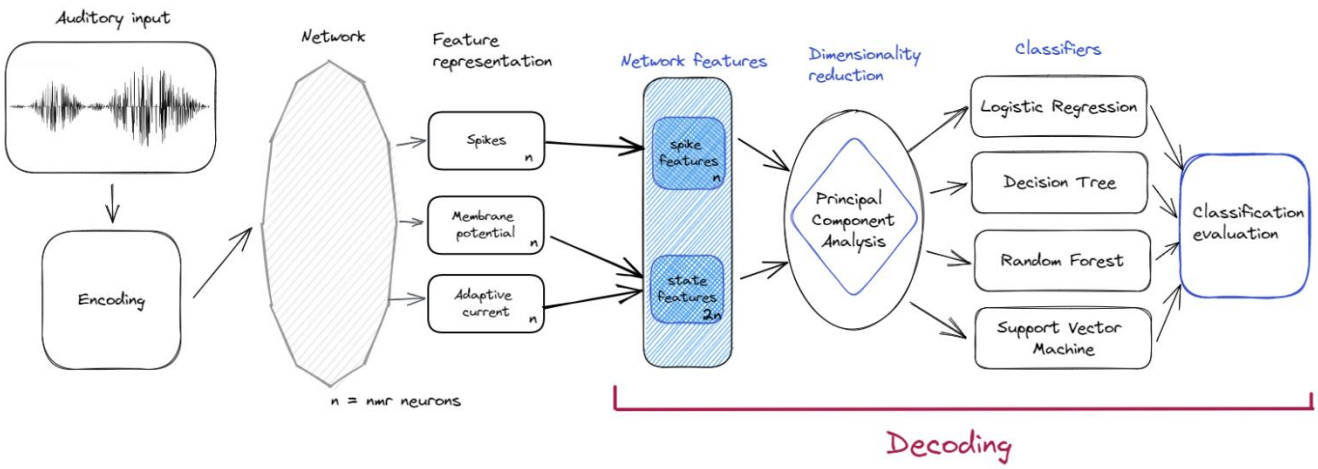


Figure 3 Schematic structure of spiking neural network

The model's auditory input is made up of data from the Spike-TIMIT dataset (Pan et al, 2020). This is a dataset which consists of phoneme-rich sentences, such that certain words can be taken from this dataset. In the encoding phase, the input words are transformed into spikes, corresponding to each distinct word. Different features of the network can be measured. Features can be thought of as specific characteristics with their corresponding values. It is critical for me to determine which features are important to the research. In my project that would be the spikes, membrane potential and adaptive current, topics of which are elaborated on in the Preliminaries chapter. The feature representation is made up of these features. After features have been distinguished from the network, these features are referred to as network features.

My project begins with the development of network features. The 'decoding pipeline' is made up of network features, dimensionality reduction, classifier application, and classification evaluation. My thesis is focused on this decoding pipeline. The network features are the ‘spike features’ and the ‘state features’. The spike features consist of spikes of the original input words, while the state features are a combination of the membrane potential and the adaptive current of the input. Various variations in network features are explored. Dimensionality reduction reduces the dimensionality of the columns of network features. This can help with both computation time and accuracy. Principal Component Analysis (PCA) is the chosen dimensionality reduction method. Following that, various classifiers are applied to the dimensionality-reduced features. The following classifiers have been chosen: Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine. Eventually, after classification, performance evaluation is applied. This focuses on both accuracy and computation time. In addition, other analyses and visualization methods have been used.

3.2 Data: auditory input

3.2.1 TIMIT

The database used for this study is an audio dataset from the TIMIT dataset, which is the benchmark database for auditory data (Garofolo et al, 1993). The TIMIT corpus for speech is a dataset created specifically to provide speech data for acoustic-phonetic studies as well as the development and testing of automatic speech recognition systems. It comes with both word and phone-level transcriptions of speech and a 16-bit, 16kHz speech waveform file for each utterance. The dataset includes broadband recordings from 630 speakers, 438 men and 182 women. Each speaker spoke ten phoneme-rich sentences. The corpus contains a vocabulary of 6224 words. The readers represent eight major American English dialects:

- Southern
- New York City
- New England
- Western
- South Midland
- Northern
- North Midland
- Moved around

3.2.2 Spike-TIMIT

A database that can work with spikes is required for studies with spiking neural networks. The TIMIT data set was encoded into spike trains, resulting in the spike-TIMIT dataset (Pan et al, 2020). The spike-TIMIT database is the event-based counterpart to the TIMIT database. In this research, the spike-TIMIT database is used as the database of choice due to the encoded spiking trains.

3.2.3 Data selection

A selection of the data from the Spike-TIMIT dataset was used for the application of this research. Working on the entire dataset is computationally very expensive, so with the exploratory design of this project, a sample of the entire dataset was chosen. The decision was made to take the 17 most frequently occurring words, ignoring short words like 'a' or 'the'. This resulted in the following input words:

“that”, “had”, “she”, “me”, “your”, “all”, “like”, “don't”, “year”, “water”, “dark”, “rag”, “oily”, “wash”, “ask”, “carry”, “suit”.

The corresponding phones of the specified words are:

“aa”, “ae”, “ao”, “ax”, “ax-h”, “axr”, “ay”, “ch”, “d”, “dcl”, “dh”, “dx”, “eh”, “er”, “g” “gcl”, “hv”, “ih”, “ix”, “iy”, “jh”, “k”, “kcl”, “l”, “m”, “n”, “nx”, “ow”, “oy”, “q”, “r”, “s”, “sh”, “t”, “tcl”, “u”, “ux”, “w”, “y”.

3.3 Encoding

The network's encoding focuses on converting auditory input into spikes. Various encoding methods are available. BAE encoding was chosen as the encoding method for my project. BAE encoding is an abbreviation for Biologically-plausible Auditory Encoding (Pan et al., 2020). It is intended to be more biologically plausible, mimicking the human auditory system, including the inner hair cells and cochlear filter bank. Furthermore, it is a spike neural encoding that produces more sparse output while being more energy efficient.

3.4 Network

The original network provided at the start of my project includes an implementation based on Litwin-Kumar and Doiron's work from 2014. An unsupervised learning rule is applied such that it has spike-timing-dependent plasticity for excitatory and inhibitory cells. For the sake of simplicity, learning for excitatory cells is disabled in this project.

The network is made up of 5000 neurons, with 4000 inhibitory neurons and 1000 excitatory neurons. The weights of the network are randomly initialized. Furthermore, the inputs given to the network are the 17 words described in the data. The network was trained on the male gender in combination with the New England dialect. Ten samples were taken for each word, with 20 repetitions. The network simulation is run twice after the various network parameters have been initialized. First, the simulation is run with learning enabled. The learning is then stopped, and the simulation is rerun, resulting in an output of various feature spaces. The network, as well as various features, are saved for future use.

3.5 Feature representation

The network measures different features. This includes the weights, spikes, rates, and states. As not all feature representations are of importance to the classification pipeline, it is important to determine which features need to be focussed on. These would be the spikes, membrane potential and adaptive current. Each inhibitory neuron in the model gets its own feature, for a total of 4000 features each for spikes, membrane potential, and adaptive current.

3.6 Network features

The classification pipeline begins with the creation of the network features. There are two types of features which can be distinguished on, the so-called ‘spike features’ and ‘state features’. Different variations of both feature spaces have been investigated.

3.6.1 Spike features

The spike features are the spiking rate of a determined point in time for each word/phone. As the spike rate is continuous over time, it is necessary to sample at specific time points as the network cannot work with continuous output.

3.6.1.1 Synchronized sampling

The sampling of the spiking rate has been synchronized with the word/phone as input. With this synchronization a sample has been taken at the beginning, middle and end of a word/phone. To ensure that the speaker is speaking, the sample is taken 1ms after the beginning and 1ms before the end of the audio. This method of sampling leads to the opportunity to investigate differences in accuracies over when a sample is taken.

3.6.1.2 Unsynchronized sampling

In unsynchronized sampling, a sample of the spiking rate is taken every 100ms. This is a static way of sampling and does not take synchronization into account when words/phones are spoken. This leads to the possibility that silence can be sampled, as between the spoken words 0.1 seconds of silence is implemented.

3.6.2 State features

The state features consist of both the membrane potential and adaptive current. The first half of the features correspond to membrane potential, while the second half contains the adaptive current features. The state features altogether are analysed on their performance, as well as for the membrane potential and adaptive current separately. As it may be that either of these features is of higher influence on classification. Each state is saved ten times per word or phone call, with equal time intervals between them.

3.6.2.1 Average

The states can be averaged per word/phone, resulting in one datapoint per feature. This translates to 4000 features for the membrane potential and 4000 features for the adaptive current. This leads to 8000 state features in total. Averaging the states can help reduce the size of a data matrix. Furthermore, it removes noise from the data, but it also results in a loss of information.

3.6.2.2 Non-average

Another option is to not average the state features and leave them as is. Since 10 states are recorded per word/phone, for each feature 10 data points are recorded instead of one. This results in a matrix with 80.000 data points per word/phone.

3.7 Dimensionality reduction

As indicated in the network features, the network can output matrices with thousands of data points per word/phone. Classification with many dimensions is computationally expensive. Also, having matrices with a large difference between the number of data points to the number of input values can lead to a poor network output. Principal Component Analysis (PCA) is the method used to reduce dimensionality.

Principal Component Analysis is a dimensionality reduction method that I chose for this project. This method takes an input matrix and returns an output matrix that is sorted by decreasing variance. The first column has the most variance, and the last column has the least. Some parameters can be altered. The principal ratio and the maximum output dimensions of the matrix were the main parameters that I tuned. A matrix contains values, which together hold all the information of the matrix. The primary ratio is the amount of information preserved from the original features (Julia Programming Language, 2022). With a principal ratio of 1.0, all the information from the original matrix is maintained after PCA, only in fewer dimensions. With a principal ratio of 0.0, 0% of the information of the original matrix is kept. As a result, the output matrix no longer says anything about the input matrix. In this project, I determined the principal ratio value to be 0.8. This implying implies that 80% of the information from the original feature spaces is kept after PCA. Furthermore, another important parameter restricts the maximum number of columns that can be included in the output matrix (Julia Programming Language, 2022).

As mentioned, limiting the dimensions for classification can help with the reduction of computation time. Additionally, it may also help with possibly increasing accuracy. Since words and phones, as well as different feature representations, have a varying number of network features, the individual differences should be considered. Trial-and-error has been applied to vary over different values for this parameter to determine the optimal size of the output matrix to take for classification. For words, as well as phones, this resulted in a value of 60 columns per word/phone. This means that I took the first 60 dimensions, with the highest level of variance in the data.

3.8 Classifiers

A choice was made to integrate different supervised classifiers. As the input data that were used have labelled data, it seemed useful to start with different supervised classification methods.

3.8.1 Multinomial Logistic Regression

Binary logistic regression is a two-class supervised classification method that estimates the likelihood of an event occurring. The result is a probability ranging between 0 and 1. The class assignment of the input is determined by a specified threshold, such as 0.7. All probabilities below 0.7 would be classified into one class, while all output above the threshold is assigned to the other class (LaValley, 2008). Multinomial Logistic Regression is an extension of binary

logistic regression that allows classification of more than two classes. This classification method was chosen since it is a straightforward method and easy to implement. The Julia Package MLJLinearModels was used to implement this. The function implementation allows for parameter tuning in the classifier (Julia Programming Language, 2019). I experimented with parameters to see how they affected accuracy or computation time. Only the intercept was not fitted in my code, so this parameter was set to 'false.' Furthermore, the threshold for classification is set at 0.5, so at a 50% probability.

3.8.2 Decision Tree

The next classifier that I chose is the Decision Tree. A Decision Tree classifier classifies an input into segments that construct an inverted tree, with a root node, internal nodes, and leaf nodes (Sharma & Kumar, 2016). It constructs the inverted tree based on certain rules and uses it to classify the data. The packages DecisionTree and SciKitLearn fit! and predict functions were added. The package has multiple parameters to possibly tune. You can, for example, specify the maximum depth of the tree or the minimum number of samples required for each tree leaf. Parameter tuning for this classifier did not result in significant differences in accuracy, such that all parameter values were kept at their default values.

3.8.3 Random Forest

The Random Forest classifier is an extension of the Decision Tree classifier. I have chosen this classifier because it typically generalizes better over data than Decision Trees and thus has the potential for better performance. Random Forest will run a predetermined number of Decision Trees and determine the classification output through majority voting. Random Forest required the DecisionTree package in Julia to be implemented. Parameter tuning lead to the change of only one parameter, which showed to have a significant influence on the performance of Random Forest. The number of trees that were created has a default value of 10 trees. The increase of this parameter with more trees consistently increases the accuracy. I ended with a value of 200 trees, as this maximized the classifier's performance.

3.8.4 Support Vector Machine

A Support Vector Machine (SVM) is another supervised classification method that I chose. This method determines a hyperplane between two classes, such that the classes can be best distinguished from each other. For multi-class classification a “one-versus-one” approach is taken, in which multiple classifiers are created each training data from two classes. SVM is relatively simple and flexible and therefore can work with a range of classification problems (Pisner & Schnyer, 2020). The LIBSVM and ScikitLearn package allowed for the implantation of a Support Vector Machine classifier. With parameter tuning, no clear distinctions in performance were found, so the default values of all parameters were kept.

3.9 Standardization

Standardization of data is useful when there are varying ranges of initial features. It brings features to a common scale and can improve accuracy. The accuracy differences between applying standardization or not are significant. In my thesis, standardization is applied to the training data. In my project, Z-score standardization is applied with the implementation of the StatsBase package. This standardization results in the mean of the data becoming 0, and the standard deviation becoming 1 (Mohamad & Usman, 2013). This is done through the following formula:

$$Z = \frac{value - mean}{standard\ deviation}$$

3.10 Classification evaluation

The data for the learning and recognition phase is divided 70% for training and 30% for testing. These training and testing phases are referred to as the learning phase and the recognition phase. A different split such as 90/10 is not chosen, since the input size is not very large. This results in a test set that is undesirably small.

Each classification method is run ten times, and the k-scores from each run are calculated and saved. To account for possible variance in different runs, the mean and standard deviation were calculated per classifier based on this output. In addition, the total computation time for each classifier method is calculated and saved. Furthermore, various performance metrics can be used to assess network performance. The k-score and computation time are the main metrics I used in my project. To get insight into principal ratio and its corresponding dimensions, this is plotted for certain feature spaces. Lastly, from the most optimal feature spaces a visualisation through PCA and confusion matrices are plotted.

3.10.1 K-score

The k-score is a modified accuracy metric. It entails calculating the raw accuracy, which is the accuracy of the input and predicted labels. Furthermore, the input words or phone numbers are shuffled at random, and the accuracy of the shuffled data and predicted data is calculated, which is referred to as the random accuracy. The k-score considers the level of random classification. When the k-score is close to zero, the classifier performs similarly to chance, and when the k-score is one, the classifier correctly classified each of the words or phones in the recognition phase. The k-score is determined as follows:

$$\text{k-score} = \frac{raw_{accuracy} - random_{accuracy}}{1 - random_{accuracy}}$$

3.10.2 Computation time

The computation time is the amount of time it takes for a function to run. The computation time of the four different classifiers was measured in this project so that it could be evaluated. The total computation time of ten runs, including training and prediction, has been calculated. The computation time is given in seconds, with two decimal places. The desired computation time is as small as possible. The computation point is measured using the @elapse method.

3.10.3 Visualisation after PCA

PCA can be used to visualize data in a 2D, or 3D plane. A graph of the first two dimensions after the PCA application is plotted against each other. These two dimensions account for the two dimensions with the greatest variance. The mean and standard deviation of the words and phones have been plotted. When words or phones are distinctly separated by the standard deviation, it may be concluded that the first two dimensions of PCA lead to enough information for a correct classification between words. However, the first two dimensions may not be sufficient for accurate classification. This plot will help to illustrate this.

3.10.4 PCA: principal ratio against dimensions

The number of dimensions in a feature vector will vary depending on the network feature. As a result, the maximum number of allowed dimensions in a PCA matrix can be varied, resulting in different principal ratios. This is calculated and plotted in a graph to provide insight into how dimensions relate to the principal ratio for different feature spaces.

3.10.5 Confusion matrix

A confusion matrix compares the predicted classes to the true classes. This can reveal whether certain words or phones are explicitly misclassified as another word or phone. The Lighthouse package was used to implement confusion matrices. Only for the most optimal performing

feature space, a confusion matrix has been configured for all four classifiers. The confusion matrix normalizes the columns so that the total of the actual labels equals one.

3.11 Materials

The programming language that was used for this research is Julia (The Julia Programming Language, 2021). Julia is an open-source language developed in 2014 (Bezanson et al., 2017). It was designed to have high performance, so it is very fast. It is a high-level and dynamic programming language. Therefore, it is a well-suited choice for this type of project. The IDE of choice was Visual Studio Code, and all code was run on the GPU of a laptop.

4. Results

This section provides an overview of the results of the experimental manipulations that I have done throughout my thesis. Figures and tables are used with the understanding of the results. As words and phones in my thesis were approached separately, that is also how they are separated in the results. As explained in the Methodology, there are different evaluation metrics that I chose. The main two metrics are the k-score and the computation time. These metrics together are of high value, in determining the optimal decoding pipeline. Only a high k-score or low computation time is not sufficient, the balance of these two metrics is valuable to my research. Additionally, I visualised three supplementary metrics. These metrics are only presented for the optimal network feature space, as determined by the k-score and computation time together. I made the choice only to take the most optimal feature space. Since a low-performing feature space does not lead to the possibility to draw conclusions and thus the metrics will lack significance. Also, by only taking the best performing feature space, I wanted to prevent the number of figures for these metrics to become too overwhelming, with as mentioned, possible unnecessary graphs due to lack of significance. The main graphs are presented in the Results section itself, while the tables with the k-scores corresponding to the graphs and the confusion matrices are presented in Appendix A. As some graphs may entail large amounts of information, I chose to only highlight the most important findings.

4.1 Words

As mentioned in the Methodology, 17 words have been used in getting these results. There are various experimental manipulations that I experimented on for words. The experimental manipulations are of different network feature spaces. As mentioned in the Methodology, there are topics I call state features and spike features. Within state features, there are the averaged and non-averaged state features. As state features consist of membrane potential and adaptation current, I additionally chose to also calculate the k-scores for these aspects separately. This leads to four more feature spaces: Averaged state features and non-averaged state features for the adaptation current, abbreviated to v and the same for membrane potential (w). This created six different feature spaces for state features. Next, for spike features I created four different feature spaces: unsynchronized sampling, begin sampling, middle sampling, and end sampling. As these ways of sampling are all very different, I experimented on each of them Next to experimental manipulations in the feature space, I also looked at the different chosen classifiers. As this research is exploratory in nature, the classifiers may lead to varying performance and thus I wanted to look at all combinations of feature spaces and classifiers. This leads to being able to draw conclusions later about the most optimal feature space in combination with a specific classifier.

4.1.1 K-score

The k-score shows the performance of the word classification though accuracy, based on the input and the entire network. With the k-score, I experimented on the above-described manipulations of varying feature spaces as well as classifiers. As I ended up with ten different feature spaces to visualise and each running on four classifiers, I chose to present these in multiple graphs and tables. Figures 4-6 show the k-scores for the different feature spaces. The graphs are bar plots which show the k-score on the y-axis, with the standard deviation presented by a black line. For state features and spike features. I will go into the specifics of the graphs separately.

4.1.1.1 States

Figure 4 directly shows differences between the k-scores for different manipulations. The averaged and non-averaged k-scores are shown. Averaged state features in combination with Random Forest or Support Vector Machine result in the highest k-score for this feature space.

This k-score is about 0,6, so about a 60% correct classification. Non-averaged state features result in higher best-performing k-scores, with about 85% correct classification. The Logistic Regression, Random Forest and Support Vector Machine classify similarly, while also considering the standard deviation. For both feature spaces, the Decision Tree Classifier behaves significantly more poorly.

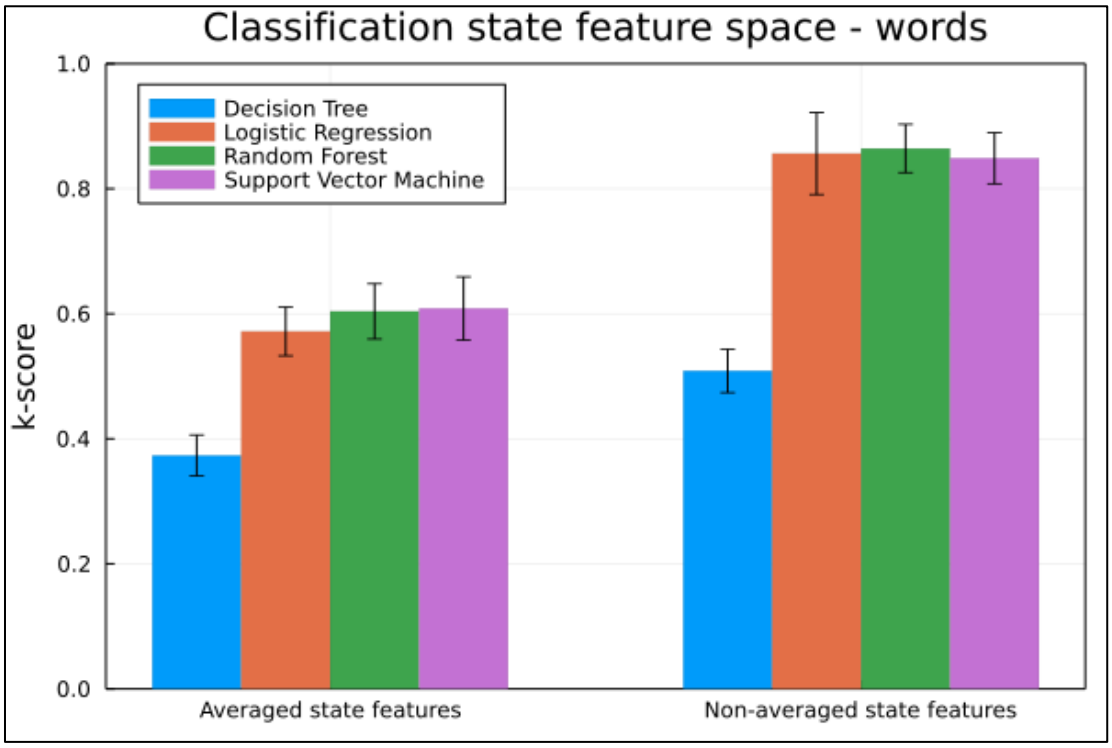


Figure 4 Classification accuracy of state features on words

To continue with the state features, Figure 5 shows the k-scores for experimental manipulations with the membrane potential and adaptation current separately. This again leads to varying outcomes. I found that the adaptation current for non-averaged state features gives the highest k-scores. A k-score of about 85% is gotten by classifying with a Support Vector Machine. The averaged state features with a k-score range of 30 to 60 percent may not be as valuable. The non-averaged state features for the membrane potential are highest at about 75%. This is higher than the averaged state features but does not measure up against the 85% range of the adaptation current on the same feature space.

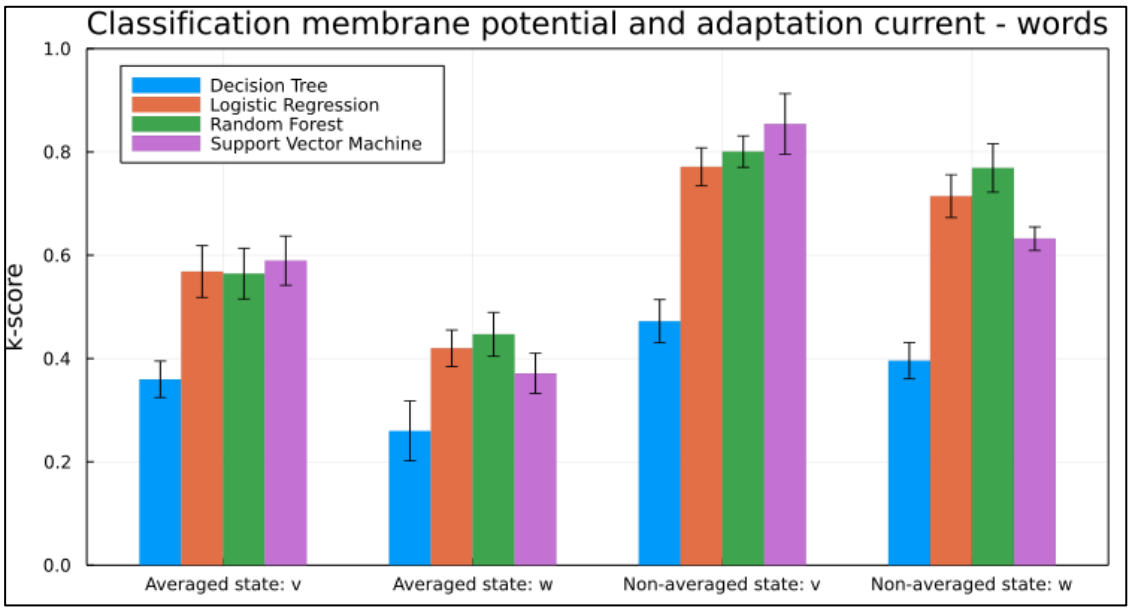


Figure 5 Classification accuracy of adaptation current: v and membrane potential: w, on words

4.1.1.2 Spikes

The k-scores of the spike features are visualised in Figure 6. As earlier mentioned, four experimental manipulations of the feature spaces are presented. It is directly visible that overall, these feature spaces do not perform that well, in comparison to the state features. The best performing feature space is the sampling at the end of a word. This has a k-score of about 33% with classification through Logistic Regression. Another interesting result is the classification of begin sampling. Begin sampling of a word leads to k-scores around zero for all classifiers and seems to perform around chance level.

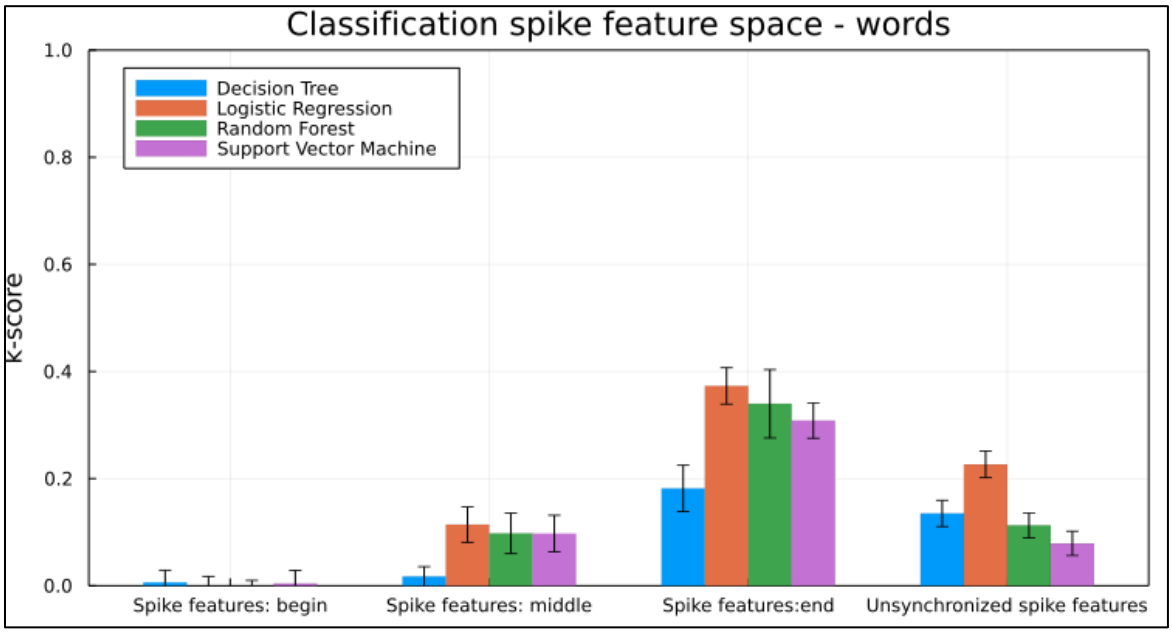


Figure 6 Classification accuracy of spike features on words

The presented result of the k-scores shows that the performance varies for different feature spaces and classifiers. The highest classification scores are from the non-averaged state features. Out of the four classifiers the Support Vector Machine, Random Forest, and Logistic Regression each performed similarly to each other with a k-score of about 85%. While the Decision Tree method showed to overall perform worse. Additionally, the non-averaged state features only on the adaptation current also received a k-score of 85%, with a Support Vector Machine.

4.1.2 Computation time

The next metric for word classification I will elaborate on is the computation time of the training and recognising of the 10 runs for each classifier. The computation time for the experimental manipulations of the feature spaces in combination with the classifiers is measured and shown in Tables 1 and 2. The computation time is measured in seconds.

4.1.2.1 States

Table 1 shows the computed computation time to run the classification on all manipulations of the averaged and non-averaged state features. From these manipulations, I found out that the computation time for all these feature spaces is in a similar range for each classifier. Between the different classifiers, the amount of time to run gives varying outcomes. Logistic Regression took the most time to run with about 500 to 550 seconds for the averaged and non-averaged state features. The manipulations on membrane potential and adaptation current create a time range that is higher from 550 to 600 seconds. For all feature spaces, the Decision Tree classifier is the faster with about 0,37 to 0,43 seconds. The Random Forest ranges around 0,7 seconds. Random Forest runs 200 decision trees instead of 1, while it only doubles in time. Lastly, the

Support Vector Machine takes about 1,2 to 1,4 seconds to complete. To emphasize this difference between the fastest and slowest classifier, the Decision Tree classifier

is about 1300 to 1500 faster than Logistic Regression.

	Averaged states	Non-averaged states	Averaged state: v	Averaged state: w	Non-averaged state: v	Non-averaged state: w
Logistic Regression	513,95	550,56	565,11	592,63	587,79	592,63
Decision Tree	0.37	0.37	0,40	0,43	0,37	0,43
Random Forest	0.73	0,68	0,75	0,72	0,68	0,72
Support Vector Machine	1.21	1,32	1,29	1,45	1,34	1,44

Table 1 Computation time for 10 classification runs averaged and non-averaged feature spaces, also with distinction on membrane potential: w and adaptation current: v, for each classifier and word classification

4.1.2.2 Spikes

Next, are the computation times for the spike variations. In Table 2 it can be seen that sampling at the beginning, middle or end again similar computation times per classifier. Where Logistic Regression by far takes more time than the other three classifiers. Interesting to mention is that the unsynchronized spike features run distinctly slower.

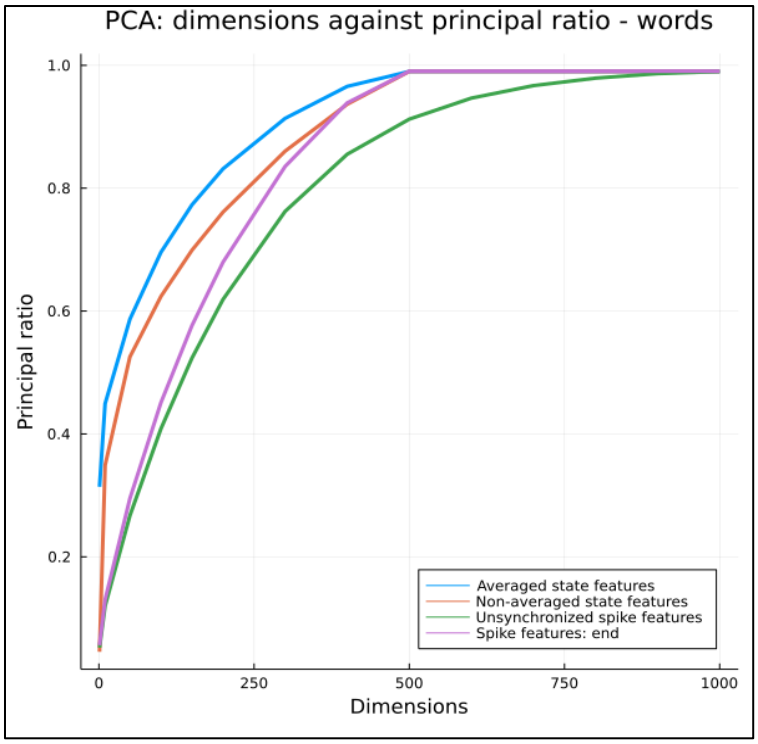
	Spike features: Begin	Spike features: Middle	Spike features: End	Spike features: Unsynchronized
Logistic Regression	356,1	355,1	343,2	625,5
Decision Tree	1,2	0,5	0,4	2,3
Random Forest	0,8	0,8	0,8	3,7
Support Vector Machine	1,3	1,3	1,3	8,7

Table 2 Computation time for 10 classification runs on spike feature space, for each classifier and word classification

Now that the k-scores and the computation time of the word classification results have been presented, it is important to look at the combined results. As mentioned, from the k-scores it can be determined that two feature spaces result in the highest k-scores. The non-averaged state in combination specifically with the Support Vector Machine, Random Forest, or Logistic Regression each performed comparable to each other with a k-score of about 85%. Furthermore, non-averaged state features of the adaptation current also resulted in about 85% for a Support Vector Machine. For state features Logistic Regression is about 800 times slower compared to a Support Vector Machine or Random Forest. To optimize the classification pipeline for words, a non-averaged feature space with a Random Forest or Support Vector machine results in the most optimal trade-off between accuracy and computation time.

4.1.3 PCA: principal ratio against dimensions

Now I focussed on the supplementary metrics, starting with the Principal Component Analysis. In the Methodology, I explained that I wanted to plot cut off in the number of dimensions against its principal ratio, which is shown in Figure 7. I made the choice not to take all ten feature spaces, as this would likely become unclear and give too much information. Therefore, I chose to focus on the averaged and non-averaged state features. For the spike features, I took the unsynchronized and end sampled spike features. As these two feature spaces differed in k-score and computation time. The end sampling performed the best out of the synchronized sampling methods, so therefore I took this feature space. Figure 7 shows that all plotted feature



spaces result in a principal ratio above 99% after reaching 1000 dimensions. Even though the input dimensions for all these feature spaces differ greatly. The averaged state features, non-averaged state features and unsynchronized spike features all go towards a 100% principal ratio after 500 dimensions. For unsynchronized sampling, about 1000 dimensions are necessary to keep almost all information in the PCA output. This is quite a decrease and can help optimize computation speed and accuracy for future purposes.

Figure 7 Principal ratio against number of dimensions of various feature spaces on words

4.1.4 PCA visualisation

Additionally, Figure 8 shows a visualisation of the first two dimensions of PCA for the non-average state feature space, as this resulted in being the most optimal feature space. The purpose of this visualisation was to see to what extent certain words can be distinguished only by projecting the first two columns of PCA, so the two columns that hold the most variance. The large dots in the graph represent the mean of the data points for the input words, with the corresponding standard deviations given by the equally coloured lines. After carefully reviewing this graph, I determined that only a few words are slightly distinguishable. The words 'me' and 'your' show to be placed more left in the graph, than any of the other words. With that, these words seem to differ in placement enough, to possibly be able to distinguish them from some of the other words. Although no words are distinctly placed enough, mostly due to the large standard deviations, to be distinguishable from all other words.

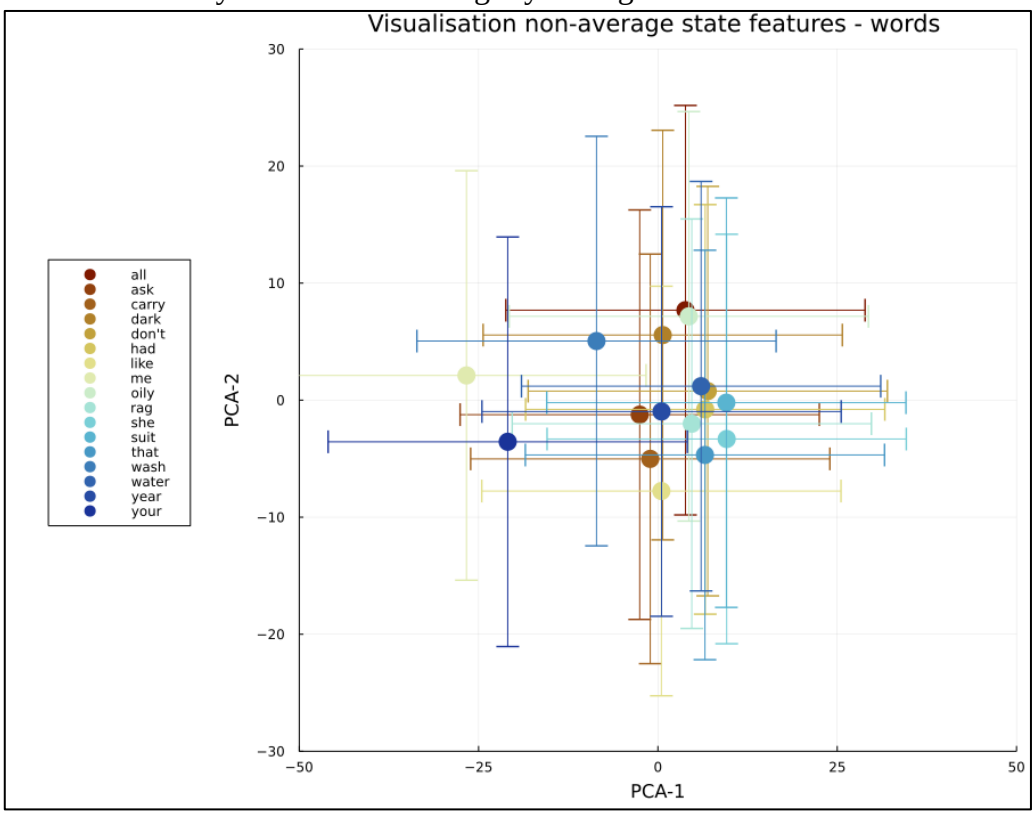


Figure 8 PCA visualisation of first two dimensions, on non-average state feature space of words

4.1.5 Confusion matrix

The last evaluation metric that I will present is a confusion matrix. The k-scores give an overall accuracy and do not distinguish between accuracies of various words. To get some insight into these possible patterns of correctly or incorrectly classifying certain words or phones, I plotted confusion matrices. As mentioned, I only chose the best performing feature space, which is the non-averaged state features. In Appendix C, Figures 14 through 17, a confusion matrix is plotted for different classifiers. The Decision Tree classifier performed significantly worse regarding its k-score than the three other classifiers, with about 50% k-score against 85%. This means that half the words are misclassified. I did choose to present this graph, only to see whether there are patterns in specific words often being misclassified as other words. In the confusion matrices, the accuracy instead of the k-score is given. This means that the random classification aspect of the k-score is not considered and generally results in slightly higher scores. This is the case since a package from the Julia library is used, and these use accuracies instead of k-scores. The graphs can be found in Appendix C

The confusion matrix of the Random Forest classifier in Figure 14, shows that certain words are correctly classified each time, 'that', 'all', 'like', 'dark' and 'rag'. The other words have some level of misclassification, which makes sense as the accuracy is not 100%. To specify the harder to classify words, I would get to the words 'your', 'don't, and 'suit'. The other words are somewhere in between. In the misclassified words, there does not seem to be a pattern. In Figure 15, the Support Vector Machine confusion matrix is presented. The word 'suit' is misclassified the most, about half the time. After 'suit', the words 'your', 'all' and 'don't' are misclassified the most. The misclassification is spread over multiple other words, not one distinctive word. The words 'that', 'had', 'she', 'year', 'water', 'ask' and 'carry' are correctly classified. Next, the Logistic Regression confusion matrix is shown in Figure 16. Here the words 'me', 'year', 'water', 'dark', 'rag', 'wash', 'carry' are classified correctly. On the other hand, the word 'all' has the lowest accuracy with about 35%. This word is classified as often to the correct class 'all', as it is misclassified to 'had'. Furthermore, the words 'that', 'she', 'your' and 'suit' have accuracy on the lower side. Lastly, the confusion matrix for a Decision Tree classifier can be found in Figure 17. This shows that no word is classified correctly for all words and only the words 'dark', 'rag', 'wash', and 'carry' have high accuracy. Most other words are incorrectly classified with a spread over various other words. Overall, the words 'suit', 'your' and 'all' seem harder to classify by all classifiers.

4.2 Phones

For phone classification very similar experimental manipulations have been performed compared to word classification. The k-scores and computation time both focus on the same 10 feature spaces, in combination with the same four classifiers as word classification did. There are more phones in the input than there are words, with 39 different phones. This may influence outcomes, which is something in which I was interested. Once again, first I start with the k-scores and computation time for getting towards an optimal decoding pipeline. After that, I will elaborate on the different evaluation metrics that I employed. In the results of word classification, I explained why I made certain choices. To prevent repetition, I will not repeat those explanations in the phone classification, as the argumentation is the same.

4.2.1 K-score

In Figures 9 to 11, the k-scores for phone classification have been presented. I experimented on the 10 different feature spaces, each running on four different classifiers. Once again, I made the split between the presentation of the graphs from state features and spike features.

Figure 4 directly shows differences between the k-scores for different manipulations. The averaged and non-averaged k-scores are shown. Averaged state features in combination with Random Forest or Support Vector Machine result in the highest k-score for this feature space. This k-score is about 0,6, so about a 60% correct classification. Non-averaged state features result in higher best-performing k-scores, with about 85% correct classification. The Logistic Regression, Random Forest and Support Vector Machine classify similarly, while also considering the standard deviation. For both feature spaces, the Decision Tree Classifier behaves significantly more poorly.

4.2.1.1 States

Figure 9 shows the k-scores for the averaged and non-averaged feature spaces. This graph shows that the non-averaged state features generally seem to lead to higher k-scores. More specifically, the non-averaged state features in combination with Random Forest or Support Vector Machine outperform the other two classifiers. The maximal k-score ranges at about 58%. While the best performing classifier for averaged state features only reaches 45% for Random Forest. For both feature spaces, the Decision Tree performs worse with only a k-score of about 30%.

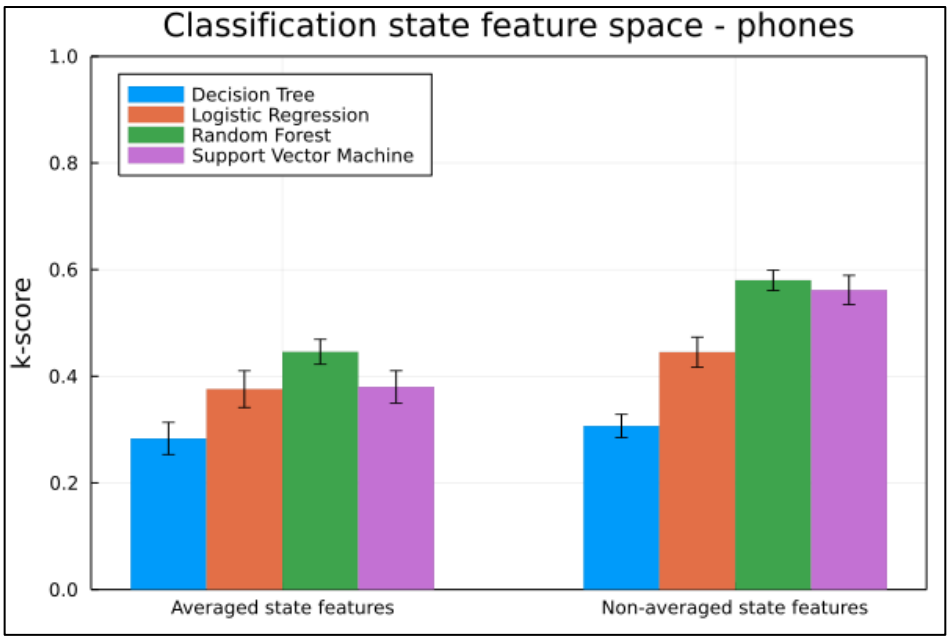


Figure 9 Classification accuracy of state features on phones

Furthermore, in Figure 10 I presented the experimental manipulations with the membrane potential and adaptation current. Once again, this results in varying outcomes. By looking at the Figure as a whole, the membrane potential for averaged and non-averaged features gives higher k-scores than the adaption current does. The highest k-score from this Figure would be the non-averaged state features for the membrane potential at a little under 60%, for the Random Forest Classifier.

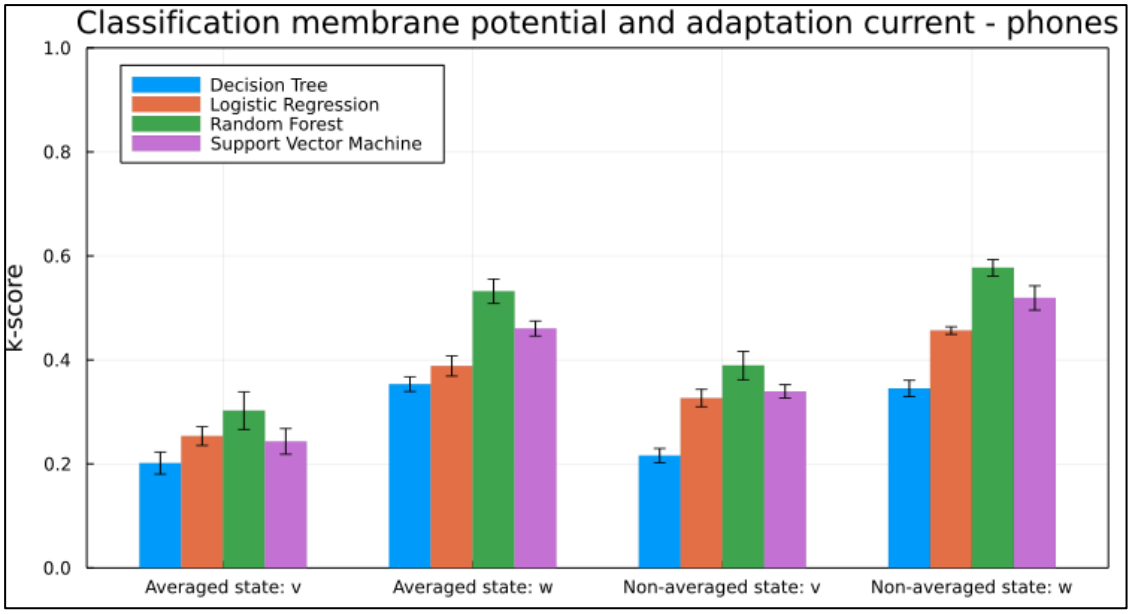


Figure 10 Classification accuracy of adaptation current: v and membrane potential: w, on phones

4.2.1.2 Spikes

Figure 11 displays the classification accuracy for the various spike features. The results of this graph directly stand out, as it shows that barely any feature space has a k-score above zero. Only for end-of-phone sampling and unsynchronized sampling there are slightly higher k-score compared to the rest. With Logistic Regression the k-scores of these two feature spaces are around 5%. As a classification method, these low k-scores are not effective for usage.

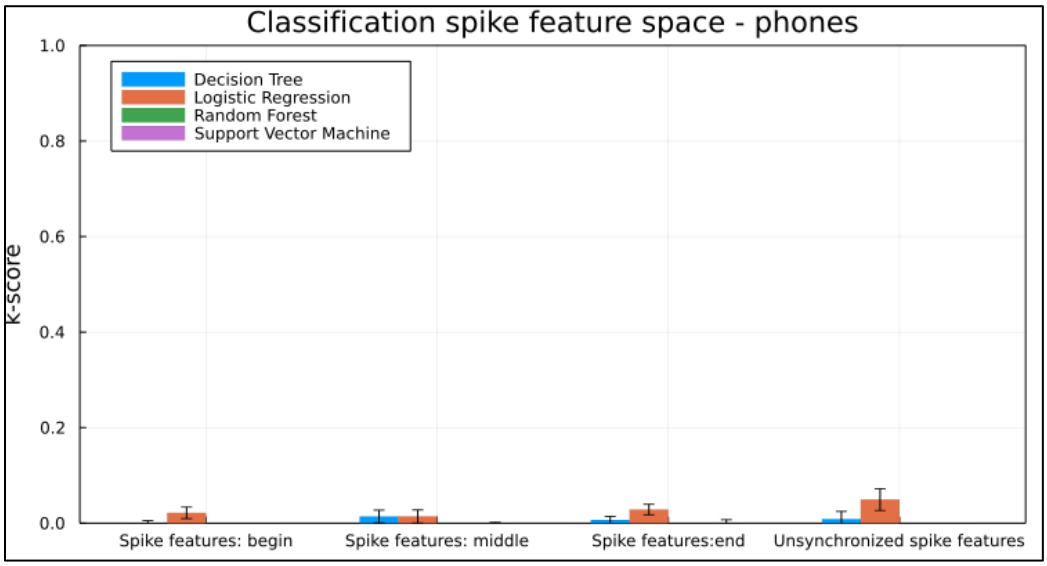


Figure 11 Classification accuracy of spike features on phones

The performance of the k-scores in phone classification differs, from zero to about 60%. The highest k-scores are gotten from non-averaged state features entirely or the non-averaged state features of the membrane potential with 60%. The Random Forest classifier in both cases performs slightly better than some of the other options. On the other hand, the Decision Tree overall resulted in the lowest scores out of the used classifiers.

4.2.2 Computation time

Now I will continue with the results of the computation time. The same experimental manipulations have been performed as with the k-score. The results are presented in Tables 3 and 4 and measured in seconds.

4.2.2.1 States

Table 3 shows the computation times for all the state features. I found again that the computation time per classifier is again similar. The Logistic Regression runs from about 3400 to 3900 seconds, while the Decision tree is finished in the range of 2,3 to 2,9 seconds. Logistic Regression compared to the other classifiers is about 300 to 1300 times slower.

	Averaged states	Non-averaged states	Averaged state: v	Averaged state: w	Non-averaged state: v	Non-averaged state: w
Logistic Regression	3962,9	3728,4	3446,3	3433,0	3486,6	3600,9
Decision Tree	2,9	2,7	2,7	2,3	2,7	2,4
Random Forest	4,3	3,9	3,7	3,7	3,7	3,7
Support Vector Machine	12,4	11,9	10,9	10,9	10,9	10,6

Table 3 Computation time for ten classification runs averaged and non-averaged feature spaces, also with distinction on membrane potential: w and adaptation current: v, for each classifier and phone classification

4.2.2.2 Spikes

Furthermore, the computation times for spike features are presented in Table 4. The computation times for begin, middle and end sampling are very similar to each other. Only the computation time of unsynchronized spike features is slightly higher for each classifier. The significant difference in computation time for Logistic Regression among the other classifiers is yet again present.

	Spike features: Begin	Spike features: Middle	Spike features: End	Spike features: Unsynchronized
Logistic Regression	869,0	835,8	835,7	933,9
Decision Tree	3,6	3,1	3,1	3,5
Random Forest	4,1	4,1	4,0	4,5
Support Vector Machine	11,5	10,5	10,5	12,6

Table 4 Computation time for ten classification runs on spike feature space, for each classifier and phone classification

Since I have presented both the k-scores and computation time, I want to focus on the combined results. As mentioned earlier, the highest k-scores can be determined by two feature spaces. These are the non-averaged features entirely or only the membrane potential of the non-averaged feature space. Both classifiers give the highest outcome with Random Forest classification. This classification method is the second fasted in computation time. Decision Tree classification is faster, although giving worse k-scores. The most balanced combination of k-score and computation time shows to be the mentioned feature spaces with Random Forest.

4.2.3 PCA: principal ratio against dimensions

Continuing with the other evaluation metrics, I will now go into specifics of the Principal Component Analysis. Figure 12 shows that averaged and non-averaged state features have a similar line, as well as end sampling and unsynchronized sampling to each other. Within 1500 dimensions, each of the feature spaces reaches a value over 99% principal ratio. Interestingly, the state features create a steeper increase in principal ratio until about 300 dimensions, over the spike features. After that, the spike features take over such that they have a higher principal ratio with the same number of dimensions in their PCA matrix.

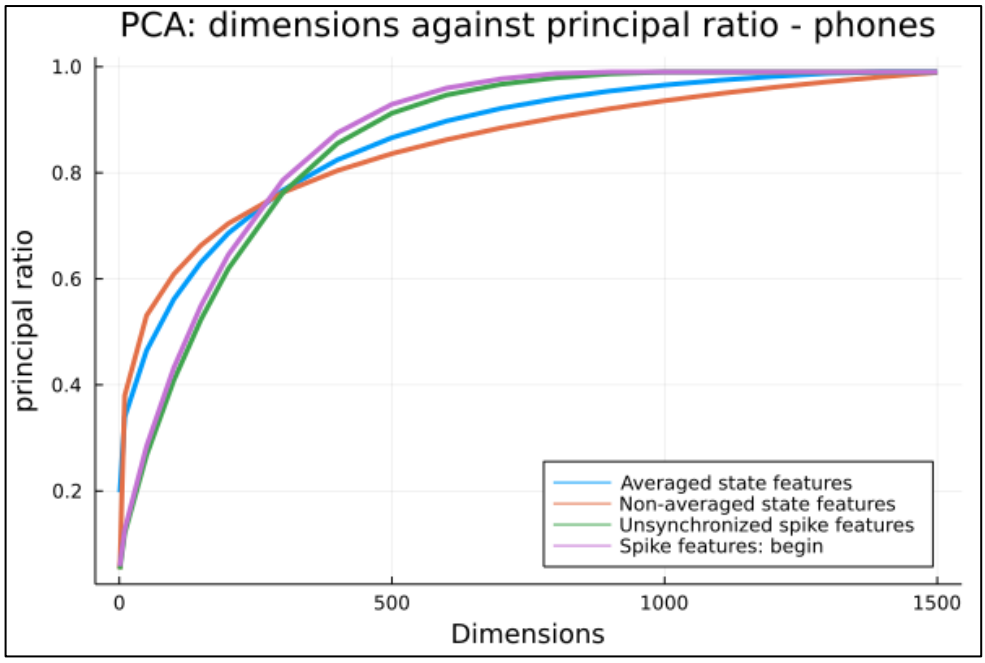


Figure 12 Principal ratio against number of dimensions of various feature spaces on phones

4.2.4 PCA visualisation

Furthermore, I will visualise the first two matrices of the PCA matrix for non-averaged state features in Figure 13. With 39 phones this graph contains a lot of information and I think is somewhat overwhelming. I will only mention two aspects. Most of the phones have a lot of overlap with other phones in their standard deviations and thus are not distinguishable in this graph. Only the phone ’d’ is placed more towards the left on the x-axis. Therefore, this may be distinguished from a part of the other phones.

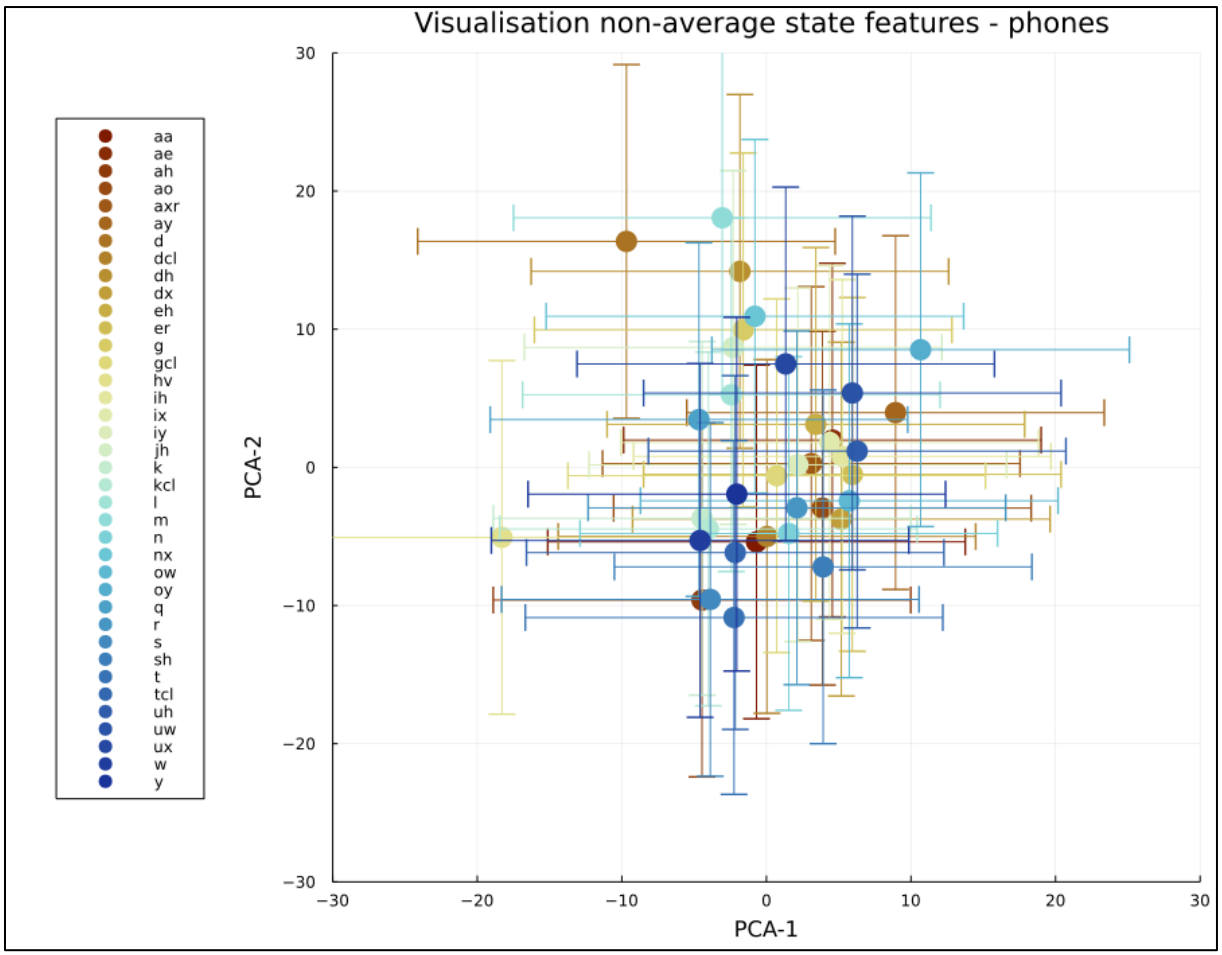


Figure 13 PCA visualisation of first two dimensions, on non-average state feature space of phones

4.2.5 Confusion matrix

At last, I experimented with confusion matrices for phone classification. I have run many confusion matrices for each classifier and through this, I figured out that each run leads to vastly different results for correctly classified and misclassified phones. This was not as much the case with word classification, which could relate to the higher number of classes with phones. Due to these variances, I determined that I would not elaborate on confusion matrices of phone classification. As the results of one presented confusion matrix graph are not informative, as no valid conclusions can be drawn from them at this point.

5. Discussion

In this project, there have been numerous significant discussion points that will be explained. Furthermore, directions for future research will be elaborated on.

Julia Lang is an open-source and relatively new programming language. As a result, certain methods may not yet be as straightforward to implement as in other languages, or there may be outdated or unfinished packages. This has led to some decisions in the decoding pipeline being made in part based on the convenience of using easy availability of methods in the Julia Language.

The k-scores for word classification have often been higher than those for phone classification. There are 39 phone classes compared to 17-word classes, which may help to explain this difference. Additionally, phones have reduced lengths and certain phones are similar in structure. These factors may make it more difficult to classify them effectively, which could have an impact on the result.

Within the k-score graphs, some interesting distinctions between word classification and phone classification and their different features are shown. The adaptation current alone produced the highest k-score for non-averaged state features in word classification, however, the membrane potential surpassed the adaptation current for phone classification. Additionally, the k-scores for phone classification demonstrated that taking non-averaged state features as a whole resulted in similar accuracies for the Random Forest classifier that was comparable to that of using only the membrane potential, so half of the non-averaged state features. The reason for this distinction is not clear. Next, for non-averaged state features in word classification, the adaptation current on its own resulted in the highest k-score, while for phone classification the membrane potential outperformed the adaptation current. Lastly, in phone classification, the k-scores revealed that using all non-averaged state features resulted in similar accuracies for Random Forest classification as using only the membrane potential. It is unclear why these distinctions exist.

In transforming the spikes to features, for begin, middle and end sampling, the resulting matrices lead to some interesting values. There were many zeroes produced in the first and last few columns. When using PCA, this could occasionally lead to a linear algebra error, making it impossible to reduce the dimensionality of certain inputs. PCA was correctly performed for several simulation runs, although the classification of words or phones was hardly better than the chance level. The errors vanished and the accuracy increased once the first and last five columns were removed. I am still unsure of how and why these zero values were initially established, as well as how they have such an impact. Further study on this is something that should be done. Additionally, word classification on begin sampling features results in low accuracy, compared to end sampling. It is possible that the beginning sampling still "classifies" based on the prior word for some unknown reason. Future research must examine whether this is happening or is simply a hypothesis.

Phones have a longer computation time than words. There are more samples for phones, resulting in a larger matrix. The computation time for unsynchronized spikes is also longer for word classification than it is for synchronized sampling. This relates to unsynchronized sampling having more samples than synchronized sampling and thus again resulting in a larger matrix which is computationally more expensive. Between feature spaces the computation time is comparable. This is since after PCA is applied, the number of dimensions of the matrices of each feature space becomes equal. If this had not occurred, the difference in computation times would have probably been greater.

The Logistic Regression fairly often is performing similar to Random Forest and Support Vector Machine in k-score. The only difference is that its computation time is hundreds of times slower. It might be interesting to investigate why this classifier takes so much longer. Next, there are a lot more classification methods than the four classifier I focused on in my thesis. It may be interesting to investigate the performance of other classifiers.

The PCA visualisation graphs for both words and phones did not yield many clear distinctions between certain words or phones. This would imply that only the first two dimensions are insufficient to provide a good classification distinction between the various classes. This was also demonstrated during the project, as only classifying on the first two dimensions of PCA did not result in high accuracy scores. The final optimal number of dimensions is more in the range of 60. As a result, no conclusions can be drawn from this graph.

The confusion matrices for words on Random Forest, Logistic Regression, and Support Vector Machine produced decent results overall. This makes sense given that the overall classification for used feature space is in the 85% accuracy range. Some words appeared to be more susceptible to misclassification. This includes the words "suit," "your," "don't," and "all." There does not appear to be a single word to which the misclassified word is then classified; instead, it is different across the different classifiers and runs. These confusion matrices only show one classifier run, rather than the ten runs that are run to obtain the k-score. As a result, if another confusion matrix is created, the misclassified words for each classifier may differ slightly from the now-produced matrices. Additionally, I did not plot confusion matrices for phones. As already mentioned in the results, the variation between confusion matrices was so large that I decided not to plot them in the results as no conclusions could be drawn from one confusion matrix. The variation in phones was a lot higher per run than it was for words. This may relate to the lower k-score of about 25% compared to words and the fact that is has more than double the number of classes. For k-score calculation I averaged ten runs, to help with a problem like this. Although, the package I used for confusion matrices did not have an option like this and I did not write code for such a solution myself. The package also had very limited options in colouring and sizing. This led to the accuracies for highly classified words being hard to read with a dark blue and black test. I did not manage to get these changes for readability unfortunately.

There is always a time constraint in a project like a thesis. This results in many future research directions. Due to time constraints, the results of this project were generated from a single simulation run. Running various simulations, where the feature representations and entire decoding pipeline are all run and analysed, is a good choice for improving validation of the performance. This project's results have only been trained and recognized on one dialect and gender. It is both interesting and necessary to broaden this. On the one hand, it is intriguing to consider the level of generalisation that the best decoding pipeline may have while training on one dialect and testing on another. The same can be done for gender, although generalisation over gender may be more complicated. On the other hand, training can be expanded to include more dialects or genders, allowing testing to cover a larger portion of the population. Lastly, as the final goal is more broadly focused on speech recognition, it is critical to broaden the amount of input words in the future. Vocabulary should be expanded and later shifted to sentences, if in time word recognition should be used for speech recognition application based on a spiking neural network.

6. Conclusion

My thesis experimented with a decoding pipeline to maximize the accuracy of a word recognition task while employing a spiking neural network. Because this is essential for learning more about optimizing classification in a word recognition task. As a starting point, a basic implementation of a spiking neural network was provided. Because my thesis was exploratory, I did not have specific hypotheses, so experimentation was vital. This study gained more insight into an optimal decoding pipeline by evaluating many combinations of experimental manipulations of ten different feature spaces, with dimensionality reductions and different classifiers, which I evaluated using classification metrics. My experiments revealed that Principal Component Analysis significantly improved accuracy. Furthermore, non-averaged state features perform best on word classification, with a k-score of 85%. For words, non-averaged state features for the adaptation current yield comparable results. Non-averaged state features or non-averaged state features of the membrane potential result in the highest k-score of 60% for phone classification. Random Forest classification is the best performing method for both words and phones because it has the lowest computation time coupled with high k-score values. Spike features may appear to be a more logical choice based on the theory, as a spiking neural network is based on the concept of spiking neurons. States do relate to spikes, as they are made up of the membrane potential and adaptation current, but it is a different representation. According to my findings, state features perform noticeably better than spike features, indicating that a possible better performance of spike features was not true. As a result, the best decoding pipeline would use non-averaged state features with PCA to reduce dimensionality and Random Forest as a classification method.

My results must be contextualized because my thesis focuses on a specific component of a word recognition task. Aside from classification, there are numerous other aspects of the used spiking neural network and word application task that must be researched. My findings can be combined with other aspects of research once more research is completed. Overall, this could contribute to the overall strengthening and improvement of a spiking neural network for a word recognition task.

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Brette, R., & Gerstner, W. (2005). Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity. *Journal of Neurophysiology*, *94*(5), 3637–3642. https://doi.org/10.1152/jn.00686.2005

Chen, I., & Lui, F. (2011). *Neuroanatomy, Neuron Action Potential*. Treasure Island (FL): StatPearls Publishing;

DecisionTree.jl. (2021). Julia Programming Language. https://julialang.org/

Dominguez-Morales, J. P., Liu, Q., James, R., Gutierrez-Galan, D., Jimenez-Fernandez, A., Davidson, S., & Furber, S. (2018). Deep Spiking Neural Network model for time-variant signals classification: a real-time speech recognition approach. *2018 International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/ijcnn.2018.8489381

Garofolo, J. S., Lori, L. F., Fisher, W. M., Fiscus, J. G., Pallett, D. S., Dahlgren, N. L., & Zue, V. (1993). TIMIT Acoustic-phonetic Continuous Speech Corpus. Linguistic Data Consortium. *Philadelphia: Linguistic Data Consortium*. https://doi.org/10.35111/17gk-bn40

Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014). Neuronal dynamics: From single neurons to networks and models of cognition. *Cambridge University Press*. https://doi.org/10.1017/cbo9781107447615

Hyman, S. E. (2005). Neurotransmitters. *Current Biology*, *15*(5), R154–R158. https://doi.org/10.1016/j.cub.2005.02.037

Ippolito, K. (2020, 14 juli). *Balancing Act in the Brain: Excitatory and Inhibitory Activity*. Max Planck Florida Institute for Neuroscience. https://mpfi.org/balancing-act-in-the-brain-excitatory-and-inhibitory-activity/

Julia Programming Language. (2019). *MLJLinearModels.jl*. GitHub. https://github.com/JuliaAI/MLJLinearModels.jl

Julia Programming Language. (2020). *DecisionTree.jl*. https://docs.juliahub.com/DecisionTree/pEDeB/0.10.10/

Julia Programming Language. (2022). *Principal Component Analysis*. Juliastats. https://juliastats.org/MultivariateStats.jl/dev/pca/

LaValley, M. P. (2008). Logistic Regression. *Circulation*, *117*(18), 2395–2399. https://doi.org/10.1161/circulationaha.106.682658

Li, X., Dalmia, S., Li, J., Lee, M., Littell, P., Yao, J., Anastasopoulos, A., Mortensen, D. R., Neubig, G., Black, A. W., & Metze, F. (2020). Universal Phone Recognition with a Multilingual Allophone System. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. https://doi.org/10.1109/icassp40776.2020.9054362

Litwin-Kumar, A., & Doiron, B. (2014). Formation and maintenance of neuronal assemblies through synaptic plasticity. *Nature Communications*, *5*(1). https://doi.org/10.1038/ncomms6319

Mohamad, I. B., & Usman, D. (2013). Standardization and Its Effects on K-Means Clustering Algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, *6*(17), 3299–3303. https://doi.org/10.19026/rjaset.6.3638

National Institute of Neurological Disorders and Stroke. (z.d.). *Brain Basics: The Life and Death of a Neuron*. https://www.ninds.nih.gov/health-information/patient-caregiver-education/brain-basics-life-and-death-neuron

Pan, Z., Chua, Y., Wu, J., Zhang, M., Li, H., & Ambikairajah, E. (2020). An Efficient and Perceptually Motivated Auditory Neural Encoding and Decoding Algorithm for Spiking Neural Networks. *Frontiers in Neuroscience*, *13*. https://doi.org/10.3389/fnins.2019.01420

Pisner, D. A., & Schnyer, D. M. (2020). Support vector machine. *Machine Learning*, 101–121. https://doi.org/10.1016/b978-0-12-815739-8.00006-7

Ponulak, F., & Kasiński, A. (2011). Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologiae experimentalis*, *71*, 409–433.

Quaresima, A. (2022). *Tutorial for spiking neurons* [Graph].

scikit learn. (2022). *sklearn.svm.SVC*. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Sharma, H., & Kumar, S. (2016). A Survey on Decision Tree Algorithms of Classification in Data Mining. *International Journal of Science and Research (IJSR)*, *5*(4), 2094–2097. https://doi.org/10.21275/v5i4.nov162954

Steen, A. A., & Stine-Morrow, E. A. L. (2016). Language: Comprehension. *Encyclopedia of Geropsychology*, 1–9. https://doi.org/10.1007/978-981-287-080-3_220-1

Weber, A., & Scharenborg, O. (2012). Models of spoken-word recognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, *3*(3), 387–401. https://doi.org/10.1002/wcs.1178

Wu, J., Yılmaz, E., Zhang, M., Li, H., & Tan, K. C. (2020). Deep Spiking Neural Networks for Large Vocabulary Automatic Speech Recognition. *Frontiers in Neuroscience*, *14*. https://doi.org/10.3389/fnins.2020.00199

Appendix

A. Words – k-scores

States

	Averaged states	Non-averaged states
Logistic Regression	57,16 (3,2)	85,62 (3,5)
Decision Tree	37,33 (3,8)	50,86 (6,6)
Random Forest	60,37 (4,4)	86,43 (3,9)
Support Vector Machine	60,84 (5,0)	84,89 (4,0)

Table 5 k-scores for state features on word classification

	Averaged state: v	Averaged state: w	Non-averaged state: v	Non-averaged state: w
Logistic Regression	56,8 (5,7)	42,0 (3,5)	77,1 (3,5)	71,4 (4,2)
Decision Tree	36,0 (3,5)	26,0 (5,0)	47,2 (4,2)	39,6 (3,7)
Random Forest	56,4 (4,2)	39,5 (4,9)	80,0 (4,7)	76,9 (3,0)
Support Vector Machine	58,9 (3,5)	47,2 (4,7)	85,4 (2,3)	63,2 (2,3)

Table 6 k-scores for adaptation current: v and membrane potential: w, on word classification

Spikes

	Spike features: Begin	Spike features: Middle	Spike features: End	Spike features: Unsynchronized
Logistic Regression	-0.10 (2,3)	11,3 (1,8)	37,3 (4,3)	22,7 (2,5)
Decision Tree	0,07 (2,7)	17,2 (3,3)	18,1 (3,4)	13,5 (2,5)
Random Forest	-0,00 (1,6)	9,8 (3,8)	33,4 (6,4)	11,3 (2,3)
Support Vector Machine	0,00 (2,4)	9,8 (3,4)	30,8 (3,3)	7,8 (2,3)

Table 7 k-scores for spike features on word classification

B. Phones – k-scores

States

	Averaged states	Non-averaged states
Logistic regression	37,6 (3,0)	44,6 (2,2)
Decision Tree	28,3 (3,4)	30,7 (2.8)
Random Forest	44,6 (2,3)	58,0 (1,9)
Support Vector Machine	38,0 (3,0)	56,2 (2,7)

Table 8 k-scores for state features on phone classification

	Averaged state: v	Averaged state: w	Non-averaged state: v	Non-averaged state: w
Logistic regression	25,3 (1,4)	38,9 (2,1)	32,7 (1,5)	45,6 (1,4)
Decision Tree	20,2 (1,9)	35,3 (1,8)	21,6 (0,7)	34,5 (1,7)
Random Forest	30,2 (2,4)	53,2 (3,6)	38,9 (1,6)	57,7 (2,7)
Support Vector Machine	24,3 (1,4)	46,0 (2,5)	33,9 (1,3)	51,9 (1,3)

Table 91 k-scores for adaptation current: v and membrane potential: w, on word classification

Spikes

	Spike features: Begin	Spike features: Middle	Spike features: End	Spike features: Unsynchronized
Logistic regression	2,1 (1,2)	1,4 (1,3)	2,8 (0,7)	4,9 (1,6)
Decision Tree	-0,1 (1,2)	1,4 (1,4)	0,7 (1,0)	0,9 (2,3)
Random Forest	-2,5 (1,1)	-2,2 (0,9)	-2,7 (1,1)	5,1 (1,6)
Support Vector Machine	-1,1 (0,9)	-0,9 (1,1)	-0,3 (1,1)	1,1(0,6)

Table 10 k-scores for spike features on phone classification

True Class

C. Words – Confusion matrices

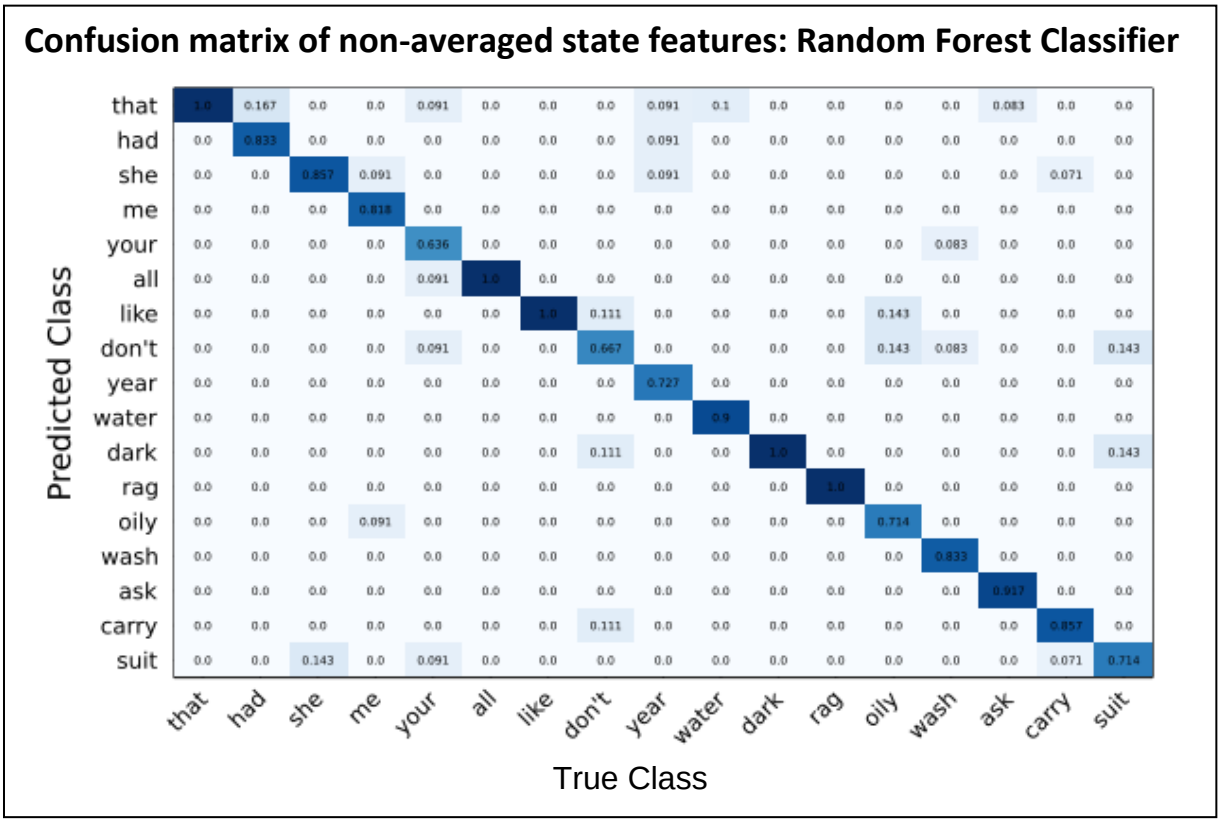


Figure 14 Confusion Matrix on non-averaged state features given word input and Random Forest Classifier

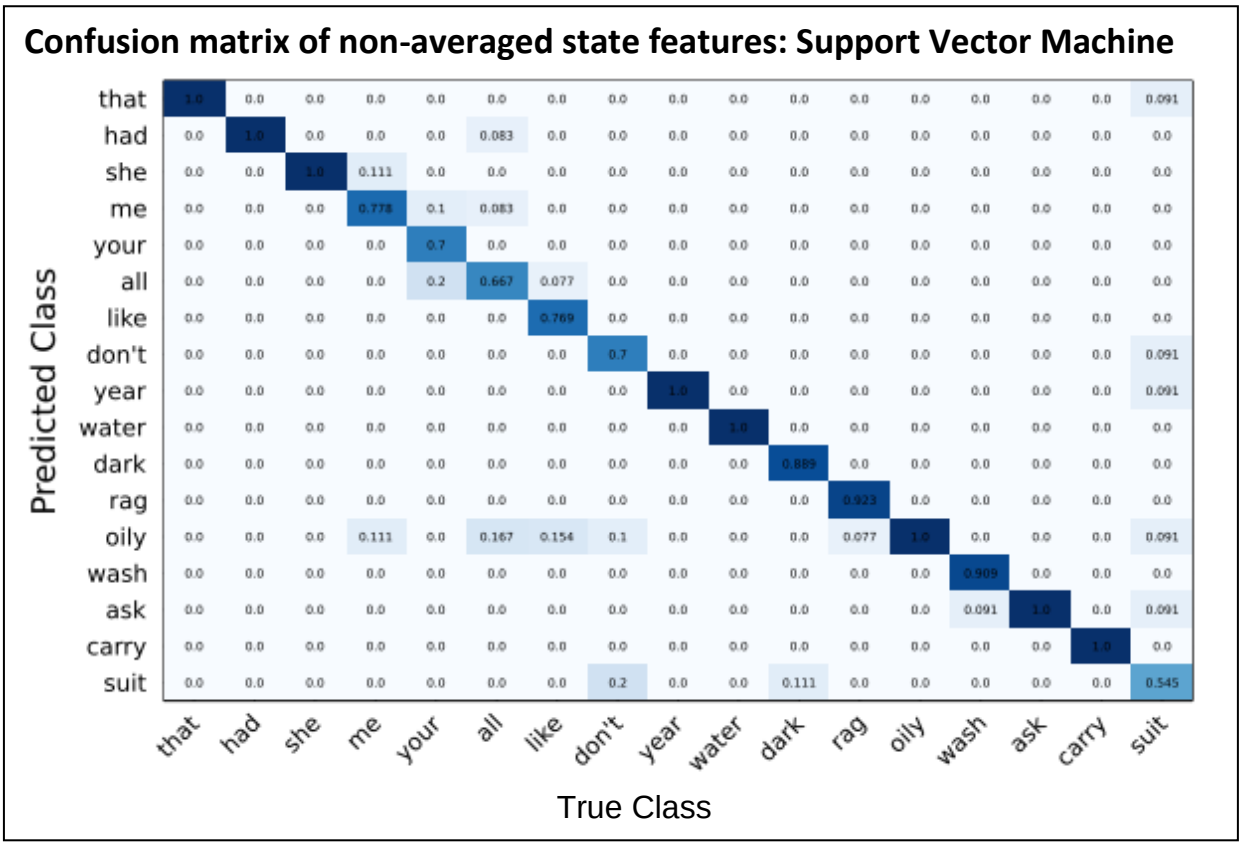


Figure 14 Confusion Matrix on non-averaged state features given word input and Support Vector Machine

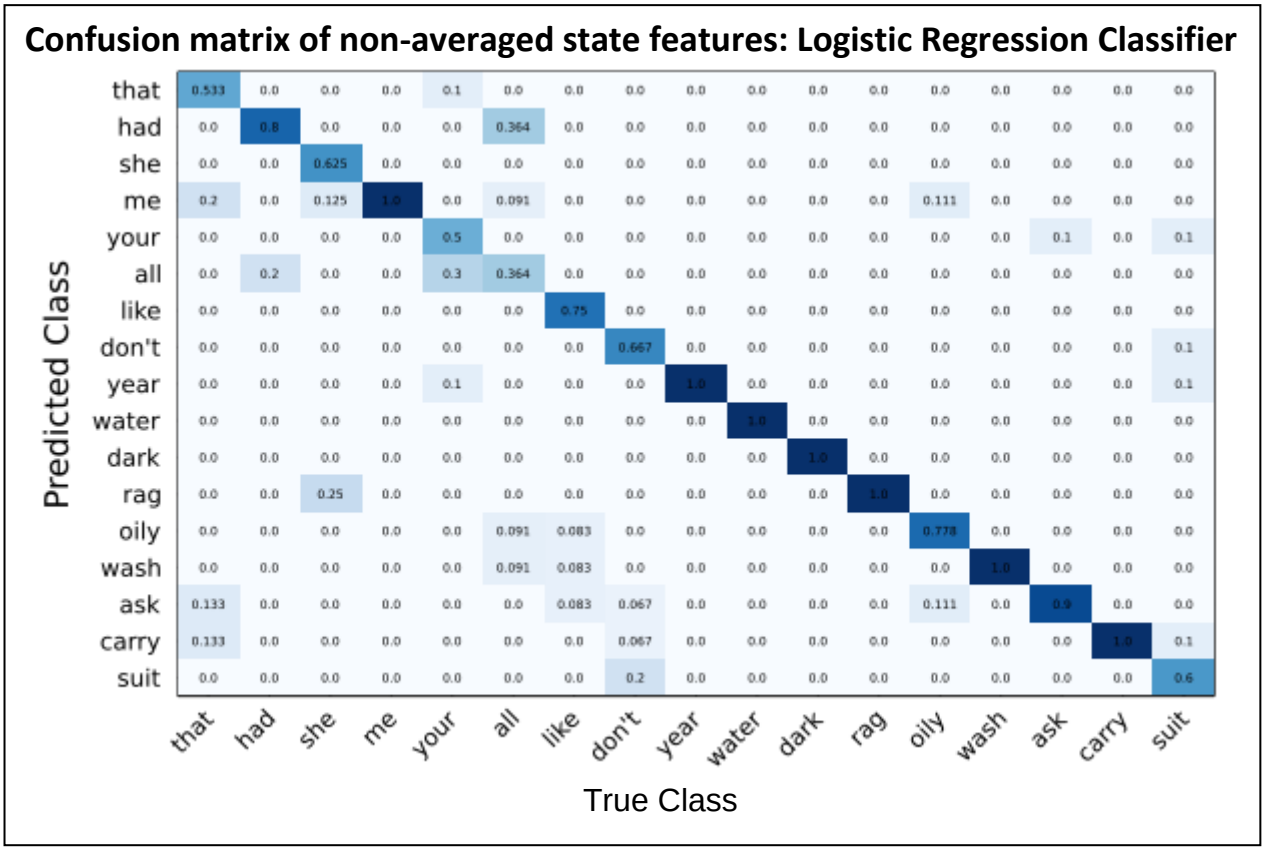


Figure 15 Confusion Matrix on non-averaged state features given word input and Logistic Regression Classifier

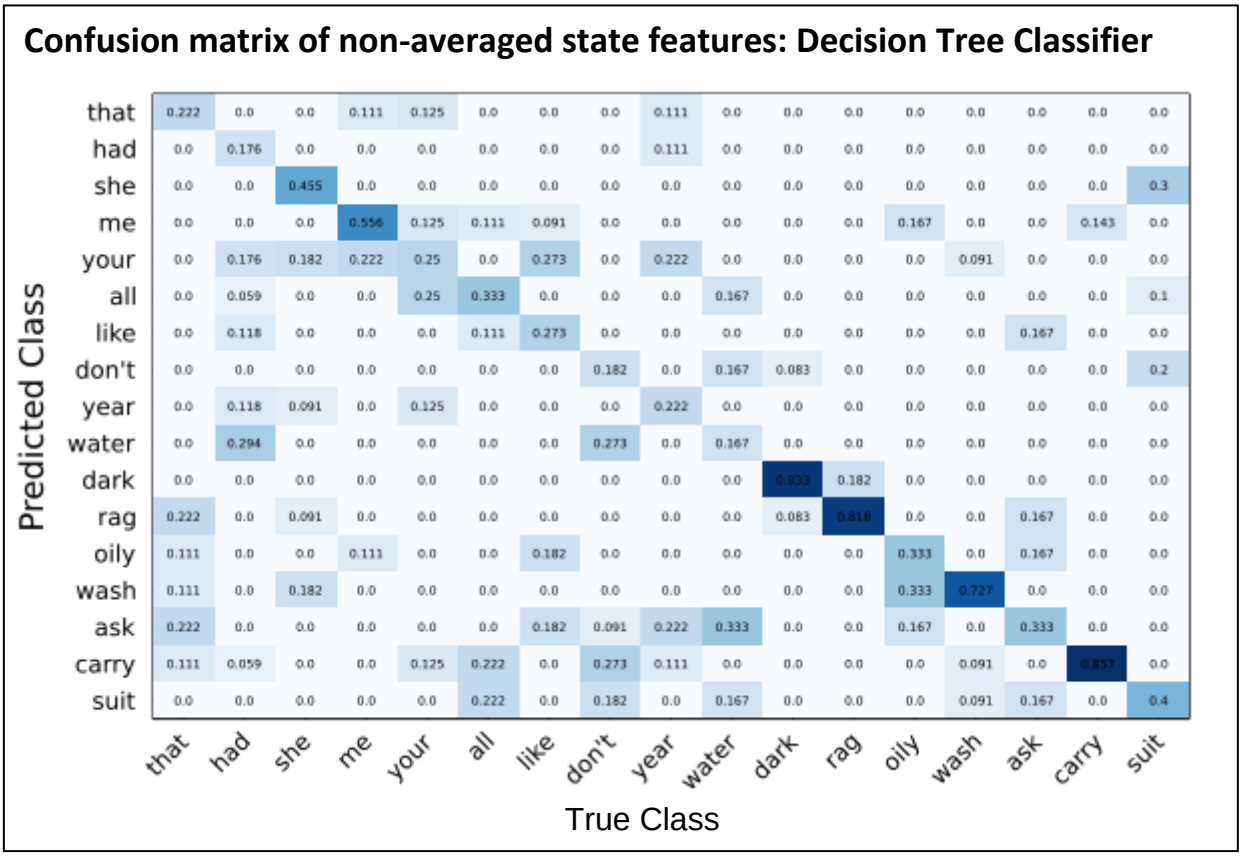


Figure 16 Confusion Matrix on non-averaged state features given word input and Decision Tree Classifier