Bachelor thesis

Artificial Intelligence

Radboud University 

FORCE training a Neurobiologically-Motivated Network for Speech Recognition

Author:
Timon Crouzen s1034392

First supervisor:
dr. H. Fitz
Donders Institute for Brain, Cognition and Behaviour
hartmut.fitz@mpi.nl

Second supervisor:
A. Quaresima
MPI for Psycholinguistics
alessio.quaresima@mpi.nl

August 20, 2022

Abstract

How language is processed in the brain is currently not well understood. With the use of neurobiologically-motivated models, brain functions like these can be better studied. Supervised learning methods provide one way to shape the network dynamics of such models. One such method is FORCE training. Based on concepts from reservoir computing, FORCE has been used to reproduce various complex behaviours. In this project, a spiking neural network with realistic neural dynamics is FORCE-trained to classify spike-encoded speech input. It was found that FORCE training does not enhance word recognition in the network. Based on the idea that FORCE training works best when replicating repetitious input, further tests were done with a different input structure, as well as tests with FORCE-training a High Dimensional Temporal Signal. The utility of FORCE training appeared to improve when reproducing a repetitious input, although these results are too limited to be definitive.

Contents

Chapter 1

Introduction

1.1 Realistic models of speech recognition

Language comprehension, like many other brain functions, is currently not well understood. It is unclear how the human brain processes variable speech input into discrete categories like phonemes and words. In recent years, advancements in cognitive computational neuroscience have led to more neurobiologically-motivated (NBM) models of neural systems. Recurrent neural network (RNN) models in particular have been proven capable of modelling various cognitive processes well. The recurrent connections in the network allow for internal memories that can capture the complex temporal dependencies of real-time input. This enables RNNs to model the processing of ongoing natural stimuli in the brain better than other network structures[1]. One type of recurrent network, the spiking neural network (SNN), functions most similarly to neural networks in the brain, which consist of discrete action potentials or "spikes".

These models have allowed for a new approach to studying various cognitive processes. From the introduction of a spike-based algorithm for speech encoding & decoding based on real-world data of the human auditory system[2], to new evidence supporting the theory that working memory in language processing is the result of intrinsic plasticity in single neurons[3], constructing and using NBM models can lead to a greater understanding of speech recognition.

1.2 Supervised learning

One approach to constructing methods that shape network dynamics is through supervised learning methods. These methods, commonly used in machine learning, use labelled training data to train networks to map input to output. One of the most widely used supervised methods is backpropagation, where the network weights are updated by gradient methods to

minimize a loss function. However, implementing these methods in SNNs is problematic, as they are usually designed for feedforward (non-recurrent) networks of neurons with continuous (non-spiking) output.

It has been shown that supervisory signals play a role in many hierarchical cognitive processes[4]. The signals are based upon the discrepancy between real and predicted stimuli, e.g. sensory feedback of actual movements vs desired movements. In a recent review, various NBM networks with realistic plasticity rules were shown to be able to approximate the backpropagation algorithm[5]. This shows how supervised learning methods, if modified for NBM networks, could be used to model learning in the brain.

The value of using supervised learning methods goes beyond modelling real learning, however. They can be used to enforce target dynamics based on real-world data unto a NBM network. So, while the way that these dynamics are constructed isn't necessarily neurobiologically authentic, the resulting network dynamics can still be used for study.

1.3 FORCE training

A unique approach to supervised learning is FORCE training. First introduced in Sussillo and Abbott[6], FORCE-trained networks were shown to be able to produce various complex output patterns, as well as reproduce neural activity of the motor and premotor cortex. FORCE training stands for First-Order, Reduced and Controlled Error learning. This is in reference to how the method has an initially low ("First-Order") error signal, which is used to train the network to reduce the training modifications required to keep the error low ("Reduced and Controlled Error").

FORCE training is based on concepts from reservoir computing. Reservoir computing is a computation method where an RNN with randomly initialized connections is used as a "reservoir" of high dimensional dynamical activity. Input is fed to the reservoir, which induces a non-linear modification of the network dynamics. In this way, complex information can be stored in patterns of neural dynamics. A decoder matrix then maps the spiking rates of the neurons to an output. During training, the decoder weights are tuned to output an approximant of a supervisor signal.

What sets FORCE training apart from standard reservoir computing methods is that, during simulation, the approximant z is encoded as a current that feeds back into the network. In other words, the network dynamics are used to calculate the approximant, and simultaneously the approximant influences the network dynamics. The method is graphically showcased in Fig. 1.1.

In Nicola and Clopath[7], it is argued that FORCE training has advantages over other supervised learning methods for recurrent networks such as predictive coding[8]. As the only thing that is needed to FORCE-train

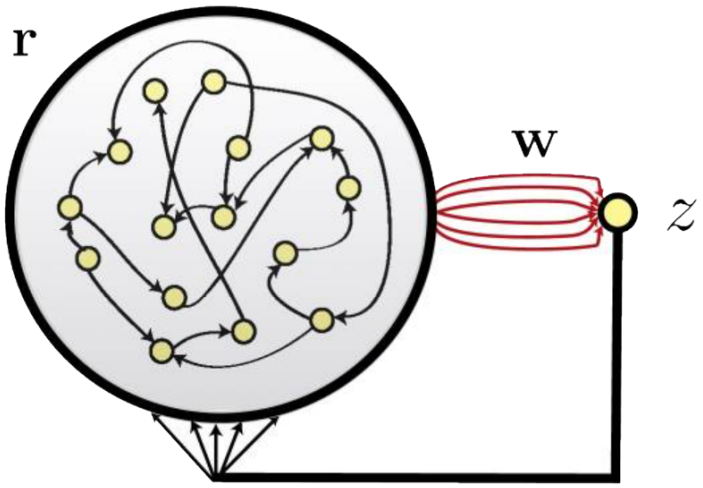

Figure 1.1: Graphical Model of the FORCE Method. The rate-based recurrent network r generates a reservoir of high-dimensional dynamic activity. Every iteration step an approximant z of the supervisor signal is decoded with decoder matrix W. The decoder weights are tuned during training. The approximant z is then encoded as a current and fed back into the network. Adapted from Sussillo and Abbott [6]

any dynamical system is an error signal, the method is highly versatile. It doesn't require any specific information about the network, nor for the task to be specified beforehand. Furthermore, tests can easily be done by directly modifying the decoder weights. This makes FORCE suitable for a wider variety of networks and tasks than other supervised learning methods, and thus a promising method to use in NBM models.

With this versatility in mind, the authors introduced an implementation of FORCE for SNNs. It is shown that FORCE training can be applied to various types of SNNs to reproduce various target behaviours. For example, they modelled a network on a neural circuit in the brain of the zebra finch that is responsible for learned singing behaviour. Then they FORCE-trained it to reproduce the finch's song. The authors also FORCE-trained a network to replay an 8-second 1920 pixel movie scene, which they posit as a model of spontaneous replay in the brain.

1.4 Research question

In both Sussillo and Abbott [6] Nicola and Clopath [7], the utility of FORCE training for cognitive computational neuroscience is conceptualized as a model for supervised learning in the brain, or at least as a tool to repro-

duce neural dynamics for further study. The extent of the utility of FORCE training as either a model or a tool requires further investigation. In this project, I will implement the FORCE method to train a NBM network to classify speech input to try to answer the question:

> *RQ: Can FORCE training enhance speech recognition in a neurobiologically-motivated network?*

Chapter 2

Methods

2.1 Implementing FORCE training

For this project I and my fellow group members were provided with base code to work with, developed by our second supervisor A. Quaresima. The code consisted of a NBM network of spiking neurons, along with methods to project spike-encoded speech data unto the network during simulation. We were also provided with spike-encoded speech data, part of which was also provided by A. Quaresima. Further explanation of the network and input will be given in the following subchapters. For this project I implement FORCE training in the network to classify the encoded speech input. The implementation of FORCE training in this project is based on how it was implemented in Nicola and Clopath [7].

2.1.1 The approximant and FORCE feedback

FORCE training requires a supervisor signal $\hat{z}(t)$. In this implementation, $\hat{z}(t)$ is a list of word vectors which have as many components as the number of unique words being presented in the simulation. The list is as long as the number of iteration steps during simulation. So, for example, a network is trained to classify the words ["your", "that", "she", "me", "had"]. For every unique word, 4 are sampled from the spike-encoded speech dataset. The supervisor signal is plotted in Figure 2.1.

The simulation starts with a 2 second delay to avoid pathological network behaviour caused by initial chaotic spiking. Then a sequence of spike-encoded speech input starts being projected unto the network. The network is trained to output an approximant of the supervisor word vector at the current iteration step. For example, when the word "she" is being presented, then the supervisor word vector at that iteration step will be [0,0,1,0,0]. Subsequently, the goal of training is to have the approximant vector z match the word vector of whichever word the network is being presented with at that moment during simulation.

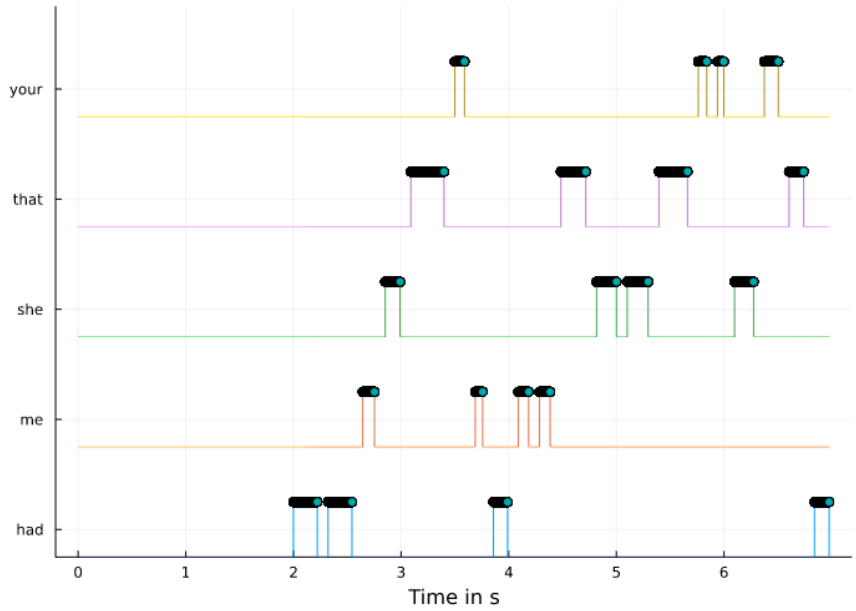

Figure 2.1: The supervisor signal. This is an example of a 7-second supervisor signal when training the network on 4 samples of 5 different words. The word presentations are plotted as black stripes. The word vector component values are plotted as coloured lines with either a value of 0 or 1. The value is at one (the line moves up) during word presentation. There is a 2 second delay before word presentation starts. Word presentations are randomly distributed.

The recurrent network consists of excitatory and inhibitory spiking neurons that have static, randomly initialized synaptic weight connections. During simulation, the spike-encoded input neurons I project unto the excitatory neurons in the network. The resulting spiking dynamics of the network serve as a reservoir of activity ω^0 that can be decoded. To avoid pathological behaviour in the network caused by initial chaotic spiking in the network, both FORCE-learning and the speech input have a 2 second delay at the start of a simulation. The decoder matrix φ requires continuous rate values of the neurons to compute the approximant. Therefore, the spiking times have to be converted to a vector of spiking rates r. To this end, a double exponential filter is applied. Then z is calculated as follows:

$$z(t) = \varphi^\top(t) \cdot r(t) \tag{2.1}$$

where:

$z(t)$ = approximated word vector at iteration step t

$\varphi^\top(t)$ = transpose of the decoder matrix at iteration step t

$r(t)$ = vector of filtered neuron spiking rates at iteration step t

The approximant is then fed back into the network as a feedback current $feedback(t)$, which is a vector with a component for every cell in the network. A static matrix η with variables drawn from a random distribution over [-1,1] and scaled by parameter Q is used to encode this feedback current:

$$feedback(t) = Q \cdot \eta \cdot z(t) \tag{2.2}$$

where:

$feedback(t)$ = vector of encoded feedback current values at iteration step t
Q = feedback current scalar
η = encoder matrix
$z(t)$ = approximated word vector at iteration step t

2.1.2 FORCE-training for word recognition

A graphical model for the network during training on spike-encoded word input is shown in Figure 2.2. While FORCE training is active, training happens through the modification of the decoder weights. These are the values of φ. Training is based on the error signal. This signal is the vector of the difference between the approximant and the supervisor:

$$err(t) = z(t) - \hat{z}(t) \tag{2.3}$$

where:

$err(t)$ = error vector at iteration step t
$z(t)$ = approximated word vector at iteration step t
$\hat{z}(t)$ = supervisor word vector at iteration step t

While training is active, two matrices are updated every two milliseconds. The Recursive Least Squares algorithm is used to update φ:

$$\varphi(t) = \varphi(t - \Delta t) - err(t)P(t)r(t) \tag{2.4}$$

where:

$\varphi(t)$ = updated decoder matrix at iteration step t
$err(t)$ = error vector at iteration step t
$P(t)$ = inverse of the correlation matrix of the neurons
$r(t)$ = vector of filtered neuron spiking rates at iteration step t

Alongside φ the matrix P is updated, which is the network estimate of the inverse of the correlation matrix of the neurons.

$$P(t) = P(t - \Delta t) - \frac{P(t - \Delta t)r(t)r(t)^\top P(t - \Delta t)}{1 + r(t)^\top P(t - \Delta t)r(t)} \tag{2.5}$$

where:

$P(t)$ = inverse of the correlation matrix of the neurons

$r(t)$ = vector of filtered neuron spiking rates at iteration step t

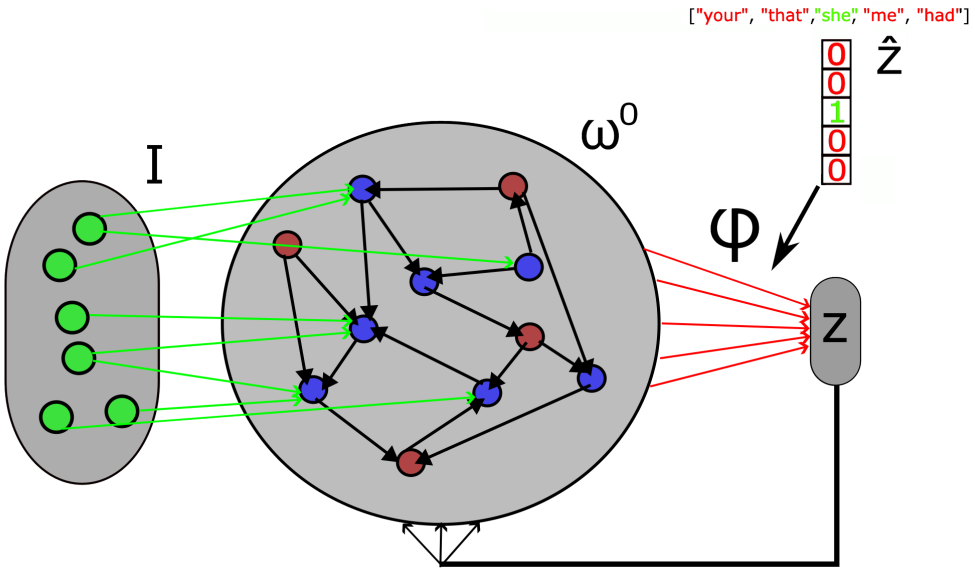

Figure 2.2: Model of the network during training. The SNN ω^0 consists of excitatory (blue) and inhibitory(red) AdEx neurons. The NBM input neurons I project the spike time-encoded input of the currently presented word "she" unto the excitatory neurons. φ computes the approximant word vector z with the network activity. With supervisor signal $\hat{z}$ the error is computed to train φ. z is fed back into the network as an encoded feedback current.

After training is over, the final weight values of φ are recorded. It can then be used for test runs, during which it computes approximants with static weights. The approximants are fed back into the network just like during the training simulation. The approximants are also recorded during the test runs, so that they can be used for offline classification.

To classify, first the average approximant component values during each word presentation are computed. Secondly, the approximated words are determined, which are the words that correspond to the component with the highest average value during the word presentations. Finally, these approximated words can be compared to the actual words to calculate the classification accuracy of the FORCE-trained network during the test run.

2.1.3 The spiking neural network

The provided code for the recurrent network is based off of the model used in Litwin-Kumar and Doiron [9]. In this paper, a NBM network with realistic

voltage-based and spike timing-dependent plasticity rules is used to study how these forms of plasticity affect learning and memory. These plasticity rules aren't relevant to this project, so they were disabled.

The network is made up of Adaptive Exponential Integrate-and-Fire (AdEx) neurons. This model has an exponential voltage dependence and an adaptive current [10]. Note that in Nicola and Clopath [7], from the various neuron models tested, the Izhikevich neurons [11] were found to perform the best, and this was attributed to the model's adaptive current. AdEx is a conductance-based model, which are highly NBM. It has been shown that conductance-based networks can carry more information in their dynamics than current-based networks [12].

The network neurons are defined as either excitatory or inhibitory, and as such they obey Dale's principle, which states that all pre-synaptic cells send either excitatory or inhibitory signals to their post-synaptic cells. The synaptic connections of the network are randomly initialized, but the weight values are set. These values are realistic, although they are static due to the disabled plasticity. But since FORCE training works with any dynamical system, the static weights will still work to generate network activity for computations.

The network is sparse, with a low rate of chaotic excitation, which hinders the encoded input's impact on the network dynamics, and means that there isn't enough chaotic activity in the network to make computations with for FORCE training. To solve this issue, the resting potential of excitatory neurons was increased from $-70mV$ to $-55mV$, so that there is more excitatory activity in the network.

2.1.4 Spike-encoded speech input

Speech input has to be projected unto the recurrent network for the speech recognition task. The benchmark dataset TIMIT [13] consists of thousands of high-quality English-language recordings of phonetically rich sentences from both male and female speakers, and with various American dialects. The raw audio data of TIMIT can't be projected unto the network, however. It first needs to be converted into spiking data with a NBM encoding.

One such encoding scheme is the Cochlea encoding model [14]. This encoding is modelled on audio processing in the human cochlea. A cochlea-encoded version of TIMIT was provided alongside the base network and input processing code, with encodings for both 35 and 70 dimensions. A raster plot of the spike times of the cochlea encoding with 70 dimensions is shown in Figure 2.3a.

Another NBM encoding, plotted in Figure 2.3b, is the Biologically plausible Auditory Encoding(BAE) [2]. This encoding modles various components of the human auditory system (e.g. the cochlear filter bank and auditory masking effects) that together encode speech data to spiking times of 620

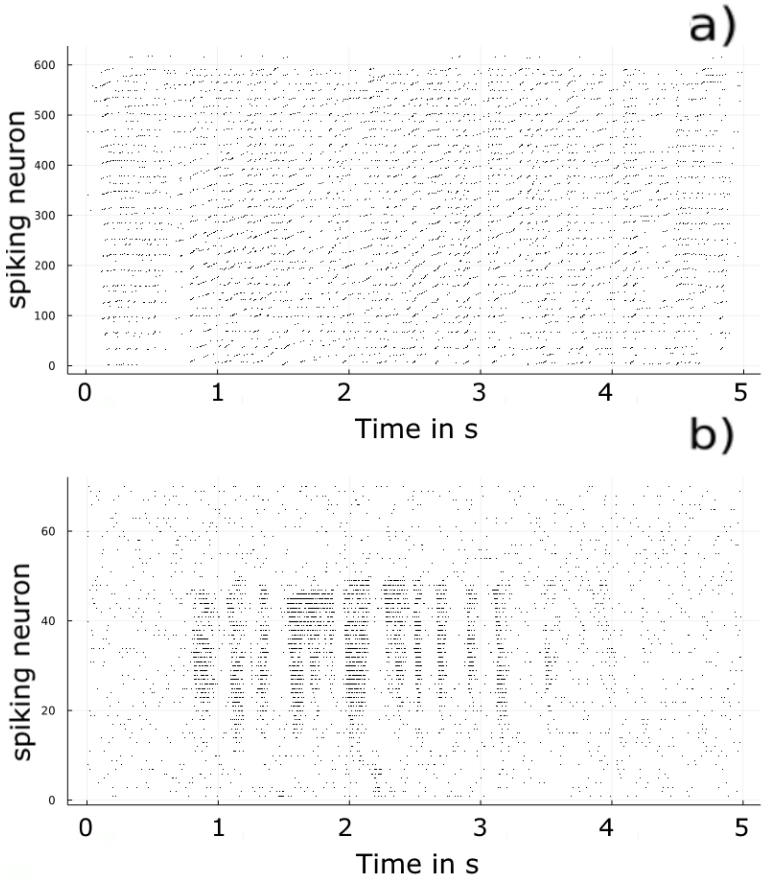

Figure 2.3: Raster plots of the spike times of 2 different speech encodings. 2 different spike time encodings of the following example sentence: "*Both have excellent integration of their fiscal tax collection year calendars.*"**a)** BAE-encoded speech input. Note that that this is before the resampling of the dimensions. **b)** Cochlea-encoded speech input.

input neurons. The paper was published alongside SpikeTIMIT, a BAE-encoded version of the TIMIT dataset that is used in this project. The BAE encoding has a lot of dimensions with sparse spikes. This means that it can't provide enough excitation during word presentation to meaningfully influence the networks dynamics. To solve this, the spike times of 620 BAE-encoded neurons are resampled to 80 dimensions. Then the resampled spike times are transformed into bursts, with a burst being a quick succession of spikes. These bursts do lead to network dynamics usable for computations.

2.2 Testing with a repetitious input structure

2.2.1 The high dimensional temporal signal

In Nicola and Clopath[7], FORCE training is first trained to approximate one-dimensional oscillatory signals, such as a sine wave or a Van der Pol oscillator. Secondly, they apply it to replicate an 8-second sequence of Ode to Joy consisting of 5 notes. Finally, they train to replicate high-dimensional signals, such as a movie sequence with a vector component for every pixel value, as well as the complex bird song spectrogram. To help with replicating these high-dimensional signals, the authors add an extra element to their FORCE implementation. They refer to this as a high-dimensional temporal signal (HDTS). The HDTS consists of a vector of components that each "pulse" at one specific part of the supervisor that the network is being presented with. For example, the 8-second movie clip has 8 pulses per second, so there are 64 HDTS components for the 8-second signal. The pulses are constructed as the positive portion of a sinusoidal wave that are all of equal length and don't overlap.

HDTS is implemented either as a seperate input vector current that is generated beforehand (externally-generated HDTS), or as additional components in the supervisor signal which are then approximated alongside the word vector values (internally-generated HDTS). In the latter case, the decoder matrix has additional columns for each HDTS component, and these weights are tuned to approximate the HDTS pulses. In other words, the network is FORCE-trained to approximate one vector, which consists of both the "standard" target components and the additional HDTS components. In the movie clip example (with externally-generated HDTS), the supervisor and approximant vectors have length 1920+64, for the 1920 pixels and the 64 HDTS components. The HDTS pulses affect the network dynamics, either as an externally generated input current, or as the approximant vector encoded within the feedback current (internally-generated).

2.2.2 The neurobiological basis for repetitious signals

The authors originally implemented the HDTS to model the neural circuit in the zebra finch's brain responsible for its song. In this circuit, one group of neurons (HVC) project a chain of spiking activity to another group of neurons (Robust nucleus of the Arcopallium, RA), where the projecting neurons in HVC fire only at a specific time during the song. The RA neurons project to the lower motor nuclei to active the bird's vocal muscles. The HDTS was based on the "repetitiously" pulsing HVC neurons. However, there are many examples of neural dynamics with this type of repetitious signal. Neural circuits containing so-called "pacemaker neurons" have been found to influence rhythmic motor behaviours such as the heart beat, and

also appear to play a role in other, non-repetitious activities such as memory and cognition[15].

The authors used FORCE training to replicate specific sequences (like the 8-second movie clip) of input. The input stream therefore consisted of this same sequence looped over and over again, with every HDTS component pulsing once per loop. For the birdsong example, they conceptualized the HDTS as a model for the HVC, which repetitiously pulses to impose network dynamics that output a memorised birdsong. FORCE trainig was then posited as similar to how young birds learn their song by internalizing their parent's song, which then functions as a target song to emulate. For the movie example, the authors conceptualized HDTS as a model for "time cells", which are hippocampal neurons that fire successively during recall of an episodic memory[16]. They found that applying HDTS increased the speed of learning and accuracy for both the birdsong and movie clip tasks, as well as a more complex version of the Ode to Joy task.

2.2.3 HDTS with encoded speech input

Given the improved task performance conferred by the HDTS in the paper, I wanted to implement HDTS in my network as well. However, the task that I had set out with did not complement a HDTS. For my task, the spike-encoded input and the supervisor signal were constructed for word presentations in a random order (non-repetitious). Furthermore, the input consists of structured spiking activity corresponding to specific sounds. The words "that" and "rag", for example, are classified as different unique words in the supervisor signal, but the "a" sound in both words will be encoded as similar spiking activity.

Despite these differences, I was interested to see if an HDTS could improve task performance in my network. First I needed to structure the speech input so that my supervisor signal consisted of loops of identical input sequences. I made a function that takes a sequence of encoded input (which was generated like normal), and then constructs a long chain of this input sequence looped many times. Then I implemented the externally-generated HDTS method as it was done in the paper. The HDTS components are added to the supervisor word vector components (Fig. 2.4). These HDTS components are approximated based on network spiking rates, just like the word vector components. The HDTS influences the network dynamics as part of the encoded feedback current. Off-line performance evaluation happens as standard, with classification only based upon the word vector component values. The HDTS influences task performance indirectly, by altering the network dynamics as part of the feedback current.

However, while e.g. the Ode to Joy song sequence has a quarter-second rhythm built into the supervisor, the TIMIT word durations vary a lot. Therefore, I implemented 2 versions of internally-generated HDTS. One is

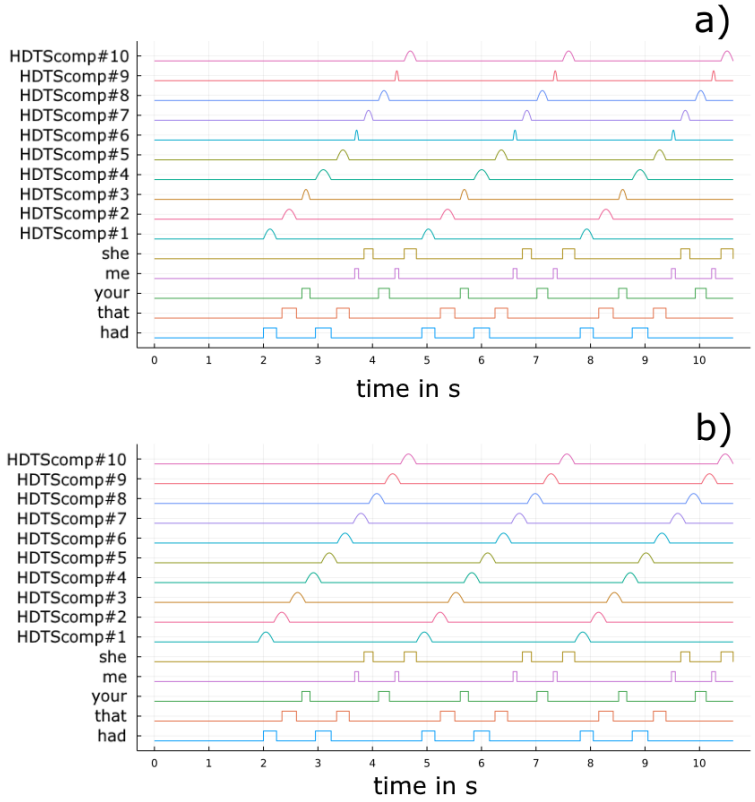

Figure 2.4: Supervisor signal with an internally-generated HDTS. The input sequence consists of 2 samples of 5 unique words, and each one of the 10 word presentations has its own HDTS component in the supervisor signal. This sequence is looped three times. During training, the decoder matrix is tuned to approximate the HDTS components alongside the 5 standard word vector components. **a)** Supervisor for HDTS-hard. **b)** Supervisor for HDTS-soft.

"HDTS-hard" (Fig. 2.4a) , where the pulses follow the word presentations exactly, and so vary in duration. The other is "HDTS-soft" (Fig. 2.4b), which has pulses that are all as long as the average length of the looped sequence divided by the number of word presentations.

There were issues with the interaction between my input sequence looping function, and the transformation of spike times into spike bursts. This pathological behaviour persisted even in absence of a feedback current. As the BAE-encoded input needs to be transformed into bursts to properly function, I could only use the Cochlea encoding when testing with repetitious input.

Chapter 3

Results

3.1 FORCE-training the network

3.1.1 Reproducing a sinus wave

To test if the network could be FORCE-trained to reproduce a simple output, the network was trained with a 1-dimensional sinus wave supervisor signal, without any speech input. The network output during training followed the sinus closely (Fig. 3.1a). The output after training is noisy and a little unstable, but it is also recognisably an oscillating wave (Fig. 3.1b).

3.1.2 Speech perception

In Figure 3.2a, the approximant values of each word are showcased during training. The values follow the supervisor very closely, which corresponds to FORCE very quickly getting a low classification error, and then the training happens so that the least decoder weight modifications have to be made to keep the error low. In Figure 3.2b it is shown how a sample row of decoder weights are modified during training and start to become more stable towards the end, before the training simulation ends. To showcase how the network classifies words, an example test simulation was run on the training data. In Fig. 3.3 the approximant values during the simulation are shown alongside an accuracy heatmap. The approximant values are noisier than during training, but they still follow the supervisor signal well. This is reflected in the heat map, which showcases a higher accuracy. The performance is so strong because the network wasn't tested on novel input.

3.2 Generalization performance of the FORCE-trained network

To test the efficacy of FORCE training for word classification, the following testing scheme was used: a decoder matrix is trained on BAE-encoded

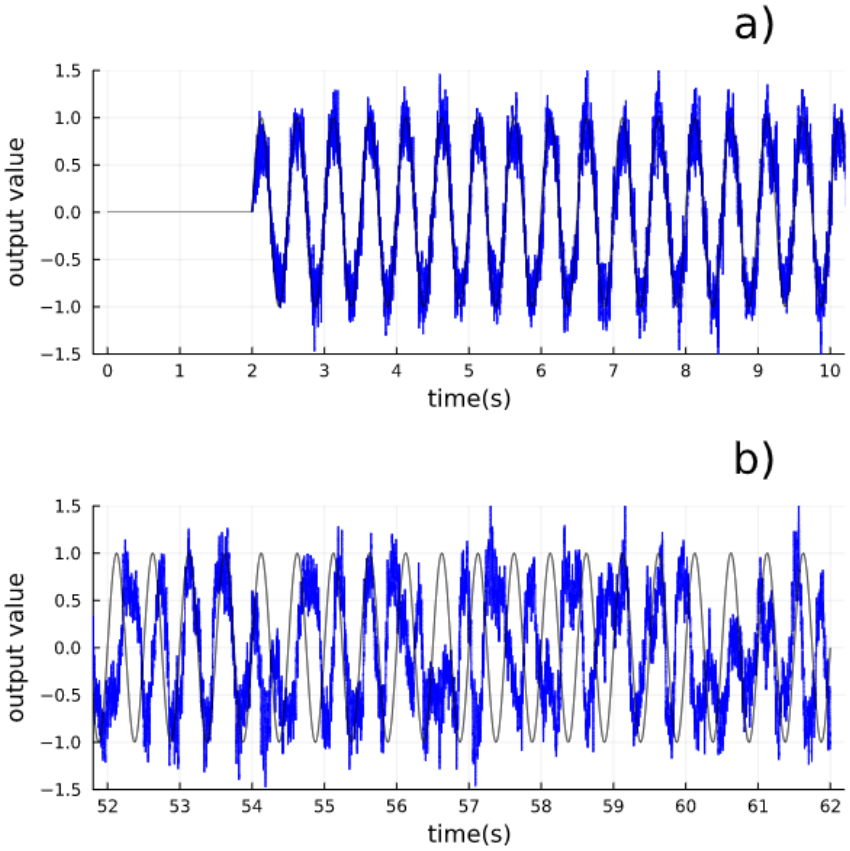

Figure 3.1: FORCE-training the network to reproduce a sinus wave. a) Network output during the first 10 seconds of training. **b)** Network output after training ends.

speech input from males with a New England dialect. The trained decoder matrix is then used for 20 test trials on novel speakers. The resulting 20 accuracy scores are used to construct a mean performance with a 95% confidence interval. This confidence interval is computed with a Student-t distribution.

By testing on novel speakers, it can be shown if the classification system generalizes to new input or not. If a testing speaker type has e.g. a different gender or dialect to the training type, it will be harder for a classifier to generalize to that testing data. The degree of generalization can be tested by comparing performances on samples from speaker types that are more and less similar to the training data. The two testing speaker types used are male speakers with a New England Dialect and male speakers with a Western dialect. The first speaker type is similar to the training type, as the gender and dialect of the novel test speakers are the same. The second speaker type has a the same gender but a different dialect. The Western

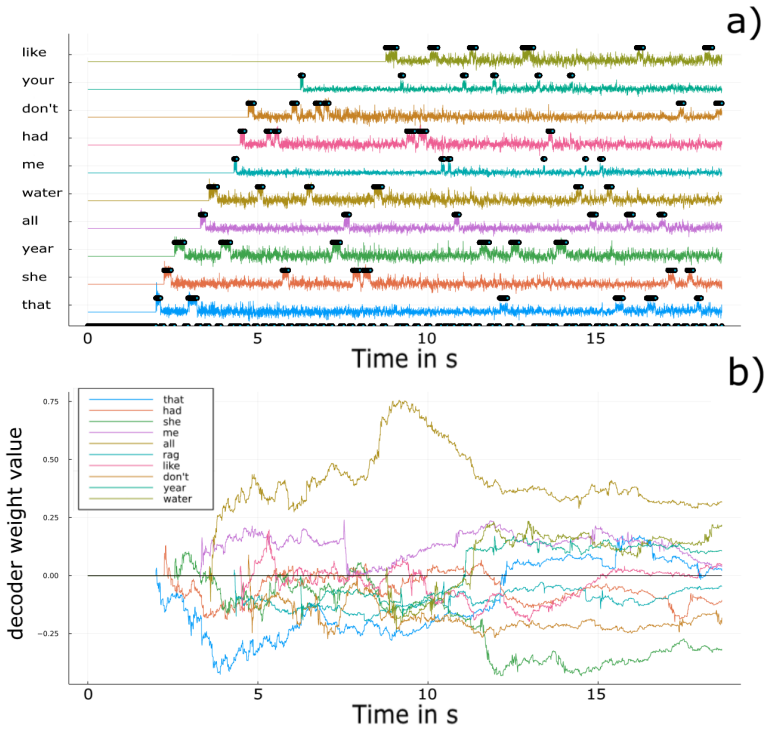

Figure 3.2: Training the readout matrix on 10 words. a) Approximant word vector values during training, with the supervisory input words in black. **b)** Values of a randomly selected row of decoder weights during train.

dialect was chosen, as it is from a part of the USA that is geographically far from New England. This means that the second speaker type will have less similar input, and so generalisation will be harder for those tests.

The mean accuracy isn't enough by itself to show if FORCE-training is effective for word recognition. To see how FORCE training affects task performance, a baseline needs to be established for evaluation. This baseline is the performance when the FORCE feedback current weights are zero (when scalar Q=0). In this case, the model no longer applies FORCE training. Instead, the network simply trains a decoder matrix based on the activity of the network with spike-encoded speech input, without influence from the approximant signal. In other words, the network becomes a more standard liquid state machine (LSM), which is a reservoir computer that uses a spiking neural network. So, if enabling the feedback current (Q>0) reliably leads to increased word classification accuracy compared to an LSM, then FORCE training can be said to enhance speech recognition.

To this end, the testing scheme was run with the 2 testing data types for 9 different feedback current weight scalar values, in a range of 0 to 200 (Fig. 3.4). The plots on the left show the classification accuracy means, and the

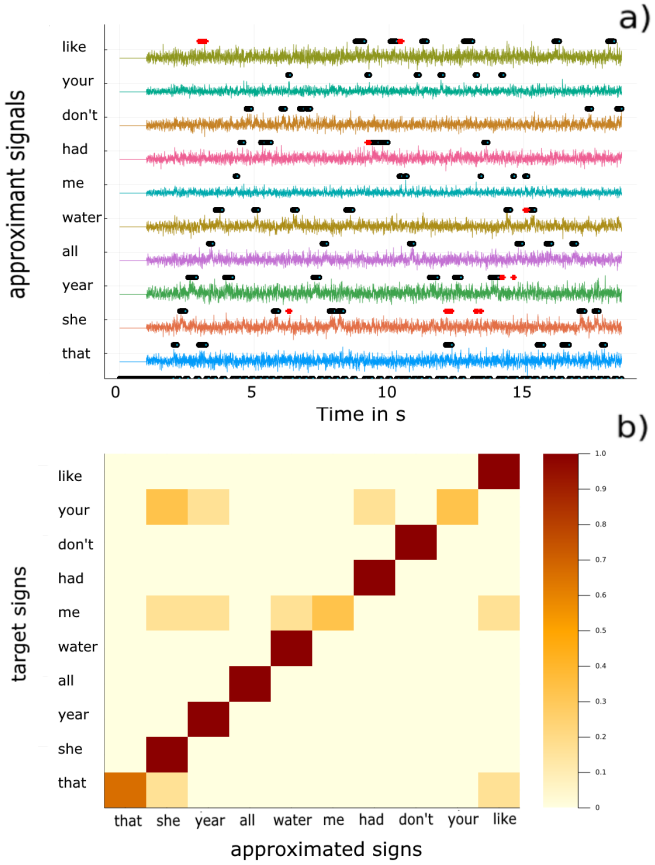

Figure 3.3: Testing with the trained readout matrix on the training data. **a)** Approximant word vector values during testing with false positives in red. **b)** Classification heatmap.

plots on the right show the difference in performance with FORCE enabled (Q>0) compared to the baseline (Q=0). Note that each trial with FORCE enabled has a seperate baseline to measure the difference in performance. Afterward, the mean of these differences is taken and a 95% confidence interval is computed.

The overall performance (on the left side) appears to be about equal for the two speaker types. The accuracy score remains mostly stable until the FORCE feedback scalar values get too high, and then there is a drop in performance. FORCE training appears to marginally enhance recognition performance for tests with the same speaker dialect as for training. The best attained mean improvement is 3%. Enabling FORCE training appears to have a negligible affect on classification performance for the tests on the Western dialect.

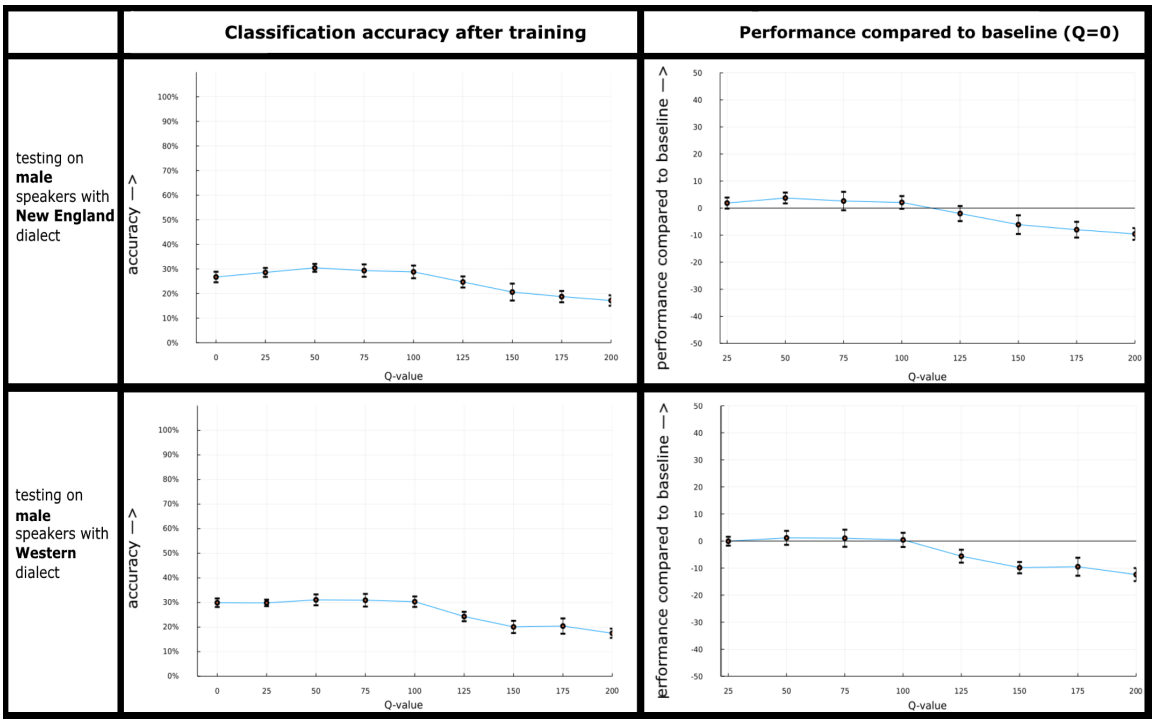

Figure 3.4: Generalisation performance plots. After training a decoder matrix on male speakers with a New England dialect, 20 test trials are run to get a mean classification accuracy with a 95% confidence interval. The plots on the left show the mean classification accuracies of these tests for 9 different feedback current scalar values Q. The plots on the right show the mean difference in performance when FORCE is enabled (Q>0) compared to the baseline (Q=0). The input consisted of samples of 17 unique words from the BAE-encoded SpikeTIMIT dataset. Training was done on 8 samples of each word repeated 3 times. Testing was done on 4 novel samples of each word.

3.3 FORCE training with repetitious input structures

During preliminary testing, I observed that my implementation of FORCE training wasn't meaningfully enhancing task performance. Therefore, I decided to test if altering the input and supervisor in a way that is more in line with Nicola and Clopath [7] would make FORCE more useful. The input was constructed so that the same sequence of the spike-encoded input and supervisor signal are played in a loop. This makes the task similar to the Ode to Joy test in the paper, where the network is FORCE trained to reproduce a small sequence of a "song" (the supervisor word vectors) with several "notes" (the unique words). Because of the new repetitious input structure, an internally-generated HDTS could be FORCE-trained as well. I wanted to see if the HDTS would increase training speed and improve performance,

like in the paper.

The performances for various input structures with or without HDTS are plotted in Fig. 3.5. The same input sequence was used for training and testing, as the goal of this task is to reproduce a specific input sequence. The sequence consisted of 3 samples of 10 words. This sequence was looped 5, 15 30 times and input during training for networks with HDTS (blue), HDTS-hard (red) and HDTS-soft (orange). As the effect of the repetitious input structure (with or without HDTS) is being evaluated, a baseline had to be established. This baseline uses the same train and test words (3 samples of 10 words repeated 5/15/30 times), but the words are instead presented in a random, non-repetitious order (green). All tests were run once for 3 FORCE feedback scalar values Q.

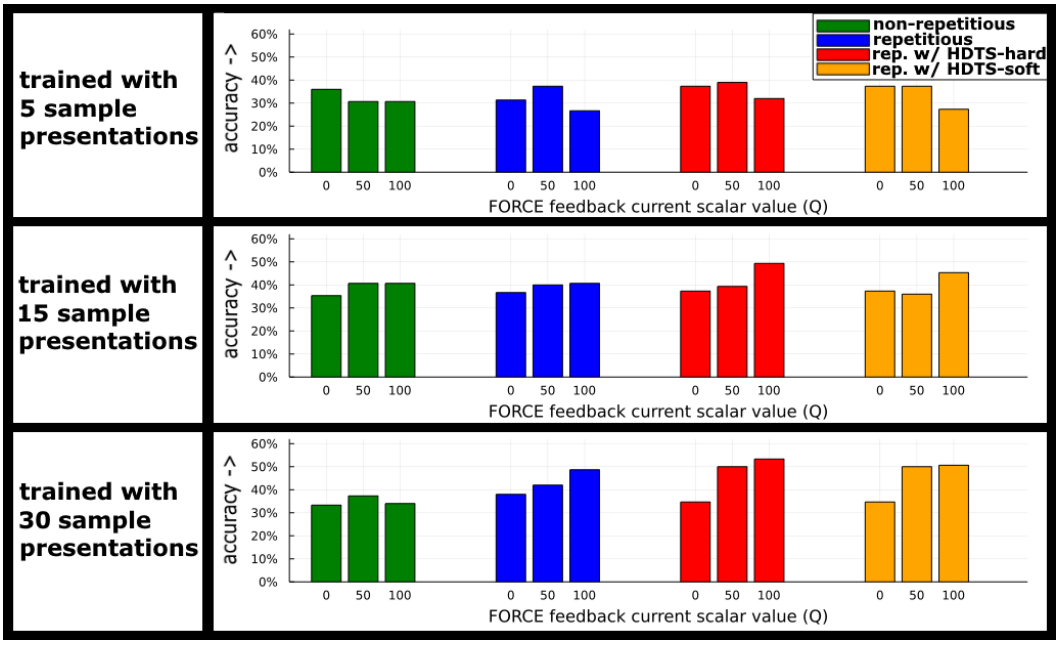

Figure 3.5: Repetitious speech input and HDTS testing. Bar plots of testing accuracy for non-repetitious input (green), repetitious input (blue), repetitious input with HDTS-hard (red), and repetitious input with HDTS-soft (orange). The input sequence consisted of 3 samples of 10 words. For training of the repetitious input, the sequence was looped 5/15/30 times, and then tested on 5 loops of the sequence. For non-repetitious input (green), the samples were presented 5/15/30 times and distributed randomly. For graphical reasons, the window for the accuracy is set between 0% and 60%

As the length of the training input increases, there appears to be a trend of increase in the performance with the tests for repetitious input. When training on 5 presentations of the sequence, the performances are about equal for all 4 plots. At 15 presentations, the runs with repetitious input HDTS enabled (red and orange) show the highest accuracies, both with

the feedback scalar set at the highest (Q=100). They both outperform the baseline performance (green) as well as for the repetitious input run without HDTS. At 30 presentations, however, all runs with repetitious input perform about the same, and they all outperform the baseline at higher feedback scalar values.

Due to limitations in computing power and time, there is only one simulation run for every accuracy score. This limits the reliability of these results.

Furthermore, because of an issue in the code that I couldn't resolve, the BAE-encoded input leads to pathological behaviour in the network when I use my sequence looping function. Therefore, I used the 70-dimensional Cochlea spike encoded speech input for the simulations of Fig. 3.5.

Chapter 4

Discussion

My implementation of FORCE training could be used to train a network to reproduce an oscillator signal, albeit a noisy one.

FORCE training did not appear to work well for the speech recognition task. While the network does generalize, the overall accuracy is low. The overall accuracy is about the same for both novel test input types (same dialect different dialect). The FORCE-trained network marginally improved the classification accuracy over the baseline LSM scheme for only the test input with the same dialect. The best mean increase in performance over baseline found was about 3%. This isn't very meaningful, and less so when considering that the accuracy sits at about 30%. Furthermore, this marginal improvement doesn't appear to generalise well, as it disappears when testing on speakers with different dialects than the training speakers.

One possible reason for this overall lackluster performance could be because of the way that the FORCE feedback current is fed into the network. FORCE is fed into the network as a current, which is encoded using a matrix with variables drawn from a distribution over [-1,1]. This means that the feedback current values can be negative or positive, and this is determined by chance. While this is fine in networks that don't have an explicit separation between excitatory and inhibitory neurons in their model, in my network it causes issues. The feedback current can lead to signals that should be excitatory being inhibitory, and vice versa. This means that Dale's principle is violated. I was not able to implement a version of the method that adhered to Dale's principle. Further work should be done to construct an implementation that does this.

Another reason that FORCE doesn't meaningfully improve performance over the baseline could be that network is also fed spike-encoded speech input. The neural dynamics induced by this input seem to be just as informative, if not more so, as the dynamics induced by the feedback current. This would be in line with Nicola and Clopath[7], witn the birdsong task. When testing with an externally-generated HDTS and the FORCE current

disabled, performance was about as good as when FORCE was enabled. In this case, the HDTS input current on the network functioned as a liquid state machine.

Another reason that FORCE training falls short could be that it is not suitable for this type of input structure. The order of the words in the spike-encoded speech input (and therefore in the supervisor signal) is randomly distributed. This is different than the tasks in the paper, where the network is trained to reproduce a specific, relatively short supervisor sequence. The networks were trained with a repetitious supervisor signal consisting of multiple loops of the exact same target sequence. The reason that this could matter is that the FORCE feedback current is best at imposing a repetitious pattern of activity, and doesn't work as well when the target dynamics aren't as consistent. Furthermore, such an input structure allows for the addition of an HDTS, which was shown in the paper to improve training speed and accuracy.

My implementation was evaluated for FORCE training with both non-rythmic and rhythmic input. A trend of increased performance was shown when using the rhythmic input, especially when the feedback current weights were scaled up. The addition of the HDTS improved training speed. With longer input, however, runs with rhythmic input without HDTS seemed to perform about as well as when using HDTS.

These results are in line with the idea that FORCE training functions best when reproducing a rhythmic signal. However, these results aren't reliable, as only one trial run was done to get the network performances. Therefore, more research is needed into the capacity of FORCE to induce such repetitious patterns of activity of activity in NBM networks—especially in light of recent theories about the role of temporally structured signals for memory construction in the hippocampus (as proposed in Eichenbaum [16]). FORCE training could be useful in modelling such signals. If there is a neural circuit involved with language processing in the brain that makes use of such repetitious patterns of activity, then FORCE training could be used to model that neural circuit.

Chapter 5

Conclusions

RQ: Can FORCE training enhance speech recognition in a neurobiologically-motivated network?

FORCE training was not found to meaningfully enhance speech recognition in the NBM network. FORCE training only marginally improves performance over the baseline LSM, and this doesn't generalise well. The overall classification accuracy is also fairly low. A reason for this could be that FORCE training violates Dale's principle in the network. Further work needs to be done to construct an implementation of FORCE training suitable for networks that have an explicit excitatory/inhibitory split.

When using repetitious input structures, FORCE training seemed to be able to enhance performance. This indicates that FORCE training could be more useful in modelling temporally structured signals. These results aren't conclusive, and more research is needed into the utility of FORCE training in neurobiologically-motivated networks.

Chapter 6

Bibliography

[1] U. Güçlü and M. van Gerven. Modeling the dynamics of human brain activity with recurrent neural networks. *Frontiers in Computational Neuroscience*, 11(7):1–14, 2017. doi: https://doi.org/10.3389/fncom.2017.00007.

[2] Z. Pan, Y. Chua, J. Wu, M. Zhang, H. Li, and E. Ambikairajah. An efficient and perceptually motivated auditory neural encoding and decoding algorithm for spiking neural networks. *Frontiers in Neuroscience*, 13(1420):1–17, 2020.

[3] H. Fitz, M. Uhlmann, D. van den Broek, R. Duarte, P. Hagoort, and K. Petersson. Neuronal spike-rate adaptation supports working memory in language processing. *Frontiers in Neuroscience*, 117(34):20881–20889, 2020. doi: https://doi.org/10.1073/pnas.2000222117.

[4] E. Knudsen. Supervised learning in the brain. *Journal of Neuroscience*, 14(7):3985–3997, 1994. doi: https://doi.org/10.1523/JNEUROSCI.14-07-03985.1994.

[5] J. Whittington and R. Bogacz. Theories of error back-propagation in the brain. *Trends in Cognitive Sciences*, 23(3):235–250, 2019. doi: https://doi.org/10.1016/j.tics.2018.12.005.

[6] D. Sussillo and L. Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009. doi: https://doi.org/10.1016/j.neuron.2009.07.018.

[7] W. Nicola and C. Clopath. Supervised learning in spiking neural networks with force training. *Nature Communications*, 8(2208):1–15, 2017.

[8] M. Schwemmer, A. Fairhall, S. Denéve, and E. Shea-Brown. Constructing precisely computing networks with biophysical spiking neurons. *Journal of Neuroscience*, 35(28):10112–10134, 2015.

[9] A. Litwin-Kumar and B. Doiron. Formation and maintenance of neuronal assemblies through synaptic plasticity. *Nature Communications*, 5(5319):1–12, 2014.

[10] R. Brette W. Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94(5):3637–3642, 2005. doi: https://doi.org/10.1152/jn.00686.2005.

[11] E. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572, 2003. doi: https://doi.org/10.1109/tnn.2003.820440.

[12] S. Cavallari, S. Panzeri, and A. Mazzoni. Comparison of the dynamics of neural interactions between current-based and conductance-based integrate-and-fire recurrent networks. *Frontiers in Neural Circuits*, 8 (12):1–23, 2014.

[13] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and V. Zue. Timit acoustic-phonetic continuous speech corpus, 1993.

[14] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–17, 2019.

[15] J. Ramirez, A. Tryba, and F. Pena. Pacemaker neurons and neuronal networks: an integrative view. *Current Opinion in Neurobiology*, 14(6): 665–674, 2004.

[16] H. Eichenbaum. Time cells in the hippocampus: a new dimension for mapping memories. *Nature Reviews Neurosciences*, 15(11):732–744, 2014.