EPFL

Thèse n° 10 659

Biologically Inspired Spiking Neural Networks for Speech Recognition

Présentée le 29 novembre 2024

Faculté des sciences et techniques de l'ingénieur
Laboratoire de l'IDIAP
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Alexandre Samuel Philippe BITTAR

Acceptée sur proposition du jury

Dr H. Lissek, président du jury
Dr J.-M. Odobez, Dr Ph. N. Garner, directeurs de thèse
Prof. A.-L. Giraud, rapporteuse
Prof. F. Zenke, rapporteur
Dr A. Popescu-Belis, rapporteur

École polytechnique fédérale de Lausanne

2024

Acknowledgements

I would first like to thank my supervisor and co-director, Phil Garner, for his support and guidance throughout this thesis. His constant availability allowed for frequent, fruitful, and insightful discussions, which significantly helped orient our research in some unique directions. I would also like to thank my thesis director, Jean-Marc Odobez, for his administrative support and oversight, notably during the annual reports, to ensure the smooth progress of my research. Additionally, I extend my gratitude to the Idiap institute's staff. Both administrative and technical support were always friendly and helpful.

I would also like to express my sincere gratitude to the Swiss National Science Foundation for their financial support, which made this thesis possible as part of the Neural Architectures for Speech Technology (NAST) project.

I finally would like to thank my thesis committee Prof. Anne-Lise Giraud, Prof. Friedemann Zenke, Prof. Andrei Popescu-Belis and Prof. Hervé Lissek for their time and effort in reviewing my thesis and participating in the oral exam.

Clarens, October 2024 Alexandre Bittar

Abstract

Biological neural networks, driving cognitive processes in the human brain, have long been a source of inspiration for computational models. Drawing from the physiology of neural dynamics, spiking neural networks stand out as prominent candidates for replicating and understanding the brain's functionality through efficient information processing.

In this thesis, we investigate spiking neural networks during sequential processing by leveraging deep learning frameworks to train and evaluate them on speech recognition tasks. Focusing on the acoustic model, our approach captures the temporal patterns and phonetic features inherent to speech signals, providing insights into speech processing throughout the human auditory pathway.

A first part is dedicated to conventional artificial neural networks and their utilisation of recurrence to address context dependencies and develop a form of working memory. Building upon a recent probabilistic derivation of recurrent neural networks, our research extends the approach and yields novel interpretable deep learning modules. While the main contributions remain theoretical, the resulting lightweight Bayesian recurrent units are shown to improve speech recognition performance compared to standard recurrent neural networks.

In a second part, we shift to the main focus of spiking neural networks that encode and transmit information via sparse and binary spike sequences. Using the surrogate gradient method, we formulate physiologically inspired architectures as a special case of recurrent neural networks. This enables us to bootstrap a study of spiking neural networks from existing deep learning frameworks. Here we also explore the role of recurrence and memory in the form of different feedback mechanisms including layer-wise recurrent connections and unit-wise spike frequency adaptation in the neuron model. While the main aim is to develop the understanding of physiological processes, our results on speech recognition tasks also contribute to the field of energy-efficient neuromorphic technology.

Lastly, an analysis of our trained spiking architectures reveals the replication of key features observed in biological networks, offering a novel and scalable approach for their study. In particular, we explore the phenomenon of neural oscillations, characteristic of cognitive processes in the brain. Our analysis confirms the presence of cross-frequency couplings in the trained networks during speech processing, notably between theta and gamma frequency bands. This synchronisation of the spiking activity is shown to arise

naturally, simply through gradient descent training, and is enhanced by the incorporation of recurrent mechanisms.
In summary, this thesis contributes to the field of neural network research by offering insights into the concepts of recurrence and spiking dynamics during sequential processing tasks, particularly in the context of speech recognition. Through a combination of theoretical analysis and practical experimentation, we develop novel methods, deep learning modules, and physiologically inspired architectures that advance our understanding of neural computation and its applications. By replicating key features observed in biological networks, our research contributes to future developments in neuromorphic computing and cognitive science.

Keywords: spiking neural networks, speech recognition, surrogate gradient, Bayesian recurrent units, neural oscillations, neuromorphic technology, recurrent neural networks

Résumé

Les réseaux de neurones biologiques, dirigeant les processus cognitifs du cerveau humain, ont longtemps été une source d'inspiration pour les modèles informatiques. S'appuyant sur la physiologie de la dynamique neuronale, les réseaux de neurones à impulsions se distinguent comme candidats de premier plan pour reproduire et comprendre les fonctionalités cérébrales à travers un traitement efficace de l'information.
Dans cette thèse, nous étudions les réseaux de neurones à impulsions appliqués à du traitement séquentiel, en utilisant des environnements d'apprentissage profond pour les entraîner et les évaluer sur des tâches de reconnaissance vocale. En se concentrant sur le modèle acoustique, notre approche capture les motifs temporels et les caractéristiques phonétiques inhérentes aux signaux vocaux, fournissant des informations sur le traitement vocal ayant lieu dans le système auditif humain.
Une première partie est consacrée aux réseaux de neurones artificiels conventionnels et à leur utilisation de la récurrence pour gérer les dépendances contextuelles et développer une forme de mémoire de travail. En s'appuyant sur une récente dérivation probabiliste des réseaux de neurones récurrents, notre recherche étend l'approche et produit de nouveaux modules d'apprentissage profond interprétables. Bien que les principales contributions restent théoriques, les unités récurrentes bayésiennes qui en résultent ont une faible charge de paramètres et améliorent les performances de reconnaissance vocale par rapport aux réseaux de neurones récurrents standard.
Dans une deuxième partie, nous passons au principal focus de notre recherche sur les réseaux de neurones à impulsions qui encodent et transmettent l'information à travers des séquences d'impulsions éparses et binaires. En utilisant la méthode du gradient de substitution, nous formulons des architectures inspirées de la physiologie comme un cas particulier des réseaux de neurones récurrents. Cela nous permet d'amorcer une étude des réseaux de neurones à impulsions à partir d'environnements d'apprentissage profond existants. Ici, nous explorons également le rôle de la récurrence et de la mémoire à travers différents mécanismes de rétroaction, comprenant les connexions récurrentes au sein d'une couche de neurones ainsi que les mécanismes d'adaptation aux fréquences d'impulsions présentes au sein du modèle de neurone. Bien que l'objectif principal soit de développer la compréhension des processus physiologiques, nos résultats sur des tâches de reconnaissance vocale contribuent également au domaine des technologies neuromorphiques économes en énergie.

Enfin, une analyse de nos architectures à impulsions entraînées met en évidence la réplication de caractéristiques clés observées dans les réseaux biologiques, offrant une approche nouvelle pour leur étude, adaptée à des architectures de plus grande envergure. En particulier, nous explorons le phénomène des oscillations neuronales, caractéristique de processus cognitifs cérébraux. Notre analyse confirme la présence de couplages entre différentes bandes de fréquences d'activité dans les réseaux entraînés lors du traitement vocal, notamment entre les bandes de fréquences thêta et gamma. Nous montrons que cette synchronisation de l'activité des impulsions apparaît naturellement, simplement grâce à l'entraînement par descente de gradient, et est renforcée par l'incorporation de mécanismes récurrents.
En résumé, cette thèse contribue au domaine de la recherche sur les réseaux de neurones en offrant des éclaircissements sur les concepts de récurrence et de dynamique des impulsions pendant les tâches de traitement séquentiel, en particulier dans le contexte de la reconnaissance vocale. Grâce à une combinaison d'analyses théoriques et d'expérimentations pratiques, nous développons des méthodes, des modules d'apprentissage profond et des architectures inspirées de la physiologie qui font progresser notre compréhension de la computation neuronale et de ses applications. En reproduisant des caractéristiques clés observées dans les réseaux biologiques, notre recherche contribue aux développements futurs en informatique neuromorphique et en sciences cognitives.

Mots-clés : réseaux de neurones à impulsions, reconaissance vocale, méthode du gradient de substitution, unités récurrentes bayésiennes, oscillations neuronales, technologie neuromorphique, réseaux de neurones récurrents

Contents

List of Figures

List of Tables

Acronyms

AC	Accumulate operation.
AdLIF	Adaptive Leaky Integrate-and-Fire.
ANN	Artificial Neural Network.
ASR	Automatic Speech Recognition.
BPTT	Backpropagation Through Time.
BRU	Bayesian Recurrent Unit.
CE	Cross-Entropy.
CFC	Cross-Frequency Coupling.
CNN	Convolutional Neural Network.
CTC	Connectionist Temporal Classification.
DNN	Deep Neural Network.
EEG	Electroencephalography.
FC	Fully-Connected.
fMLLR	Feature-space Maximum Likelihood Linear Regression.
GPU	Graphics Processing Unit.
GRU	Gated Recurrent Unit.
HD	Heidelberg Digits.
HMM	Hidden Markov Model.
KWS	Keyword Spotting.
Li-BRU	Light Bayesian Recurrent Unit.
Li-GRU	Light Gated Recurrent Unit.
LIF	Leaky Integrate-and-Fire.

LSTM	Long Short-Term Memory.
LVCSR	Large Vocabulary Continuous Speech Recognition.
MAC	Multiply-and-accumulate operation.
MFCC	Mel-Frequency Cepstral Coefficient.
MLP	Multilayer Perceptron.
PAC	Phase-Amplitude Coupling.
PER	Phoneme Error Rate.
RAdLIF	Recurrent Adaptive Leaky Integrate-and-Fire.
ReLU	Rectified Linear Unit.
RLIF	Recurrent Leaky Integrate-and-Fire.
RNN	Recurrent Neural Network.
SC	Google Speech Commands.
SFA	Spike Frequency Adaptation.
SGD	Stochastic Gradient Descent.
SHD	Spiking Heidelberg Digits.
SNN	Spiking Neural Network.
SRM	Spike Response Model.
SSC	Spiking Speech Commands.
SSM	State Space Model.
STDP	Spike Timing Dependent Plasticity.
UBRU	Unit-wise Bayesian Recurrent Unit.
WER	Word Error Rate.

1 Introduction

1.1 Motivation

Biological neurons encode and transmit information in the form of sparse sequences of binary events called spike trains. To replicate these dynamics, researchers have developed mathematical models of single neurons that can produce realistic responses to stimuli (Gerstner and Kistler, 2002; Izhikevich, 2003; Brette and Gerstner, 2005). Despite these advancements, understanding the mechanisms through which large populations of neurons, especially across different brain regions, process information remains a key area of research. One prominent focus is the study of neural oscillations, a macro-scale phenomenon that plays a critical role in driving cognitive processes in the brain. Neuroscientists explore these oscillations by analysing brain data directly or by applying physiologically inspired models to various tasks. On top of understanding fundamental properties of physiological processes, this field also enables practical applications in the biomedical domain.

In contrast, the field of machine learning has demonstrated the impressive capabilities of Artificial Neural Networks (ANNs), often approaching or even surpassing human performance on a wide range of tasks. The single neuron model in ANNs can be seen as a simplified, rate-based approximation of biological neurons. However, the main strength of the resulting networks lies in their scalability. By combining Stochastic Gradient Descent (SGD) training, vast amounts of data, and fast matrix multiplications on Graphics Processing Units (GPUs), ANNs have achieved remarkable success in a multitude of technological applications, including image, natural language, and speech processing. Moving forward, the development of architectures that are both energy and parameter-efficient is crucial for enabling these powerful models to run effectively on low-powered devices.

At the intersection of neuroscience and machine learning, Spiking Neural Networks (SNNs) mirror biological processes more closely than conventional ANNs. By capturing

the temporal dynamics and precise spike-timing patterns seen in the brain, integrating SNNs into modern machine learning frameworks offers a promising pathway to enhance our understanding of brain function and improve the efficiency and capabilities of machine learning models.

Speech represents the primary mode of human communication. The sequential nature of speech signals is characterised by clear temporal dependencies, where the sound produced at any given time step is influenced by preceding sounds in terms of intonation, phonetics and other linguistic factors. These temporal dependencies emerge from a complex interplay of physiological processes involved in speech production and perception. At the core of these processes are neural network dynamics.

During speech production, these dynamics orchestrate motor control by coordinating precise sequences of articulatory movements from the tongue, lips, jaw, and vocal folds to produce speech sounds. Conversely, in speech perception, neural dynamics facilitate the temporal integration and parsing of continuous acoustic signals into meaningful linguistic components.

Our interest in the neural network dynamics driving speech processing is twofold.

1. Understanding the underlying physiology at a more fundamental level.
2. Exploring its natural energy-efficiency for technological applications.

1.2 Approach

In this thesis, we focus on speech encoding through Automatic Speech Recognition (ASR) tasks for the following reasons. Firstly, ASR offers objective evaluation metrics that accurately reflect performance and facilitate model comparisons. Secondly, ASR benefits from the availability of extensive datasets for training. More generally, developing biologically inspired speech encoders provides a unique opportunity to compare computational approaches with the human auditory pathway, enhancing our understanding of both biological and artificial intelligence systems. Finally, the architectures we develop as general speech encoders show potential for broader use in diverse speech processing applications.

Our general approach is to cast techniques from Bayesian statistics and physiological processes to develop more efficient and interpretable neural architectures for speech encoding. On top of the relevance to low-powered, neuromorphic technology, we also hope that our models can serve as novel tools to test neuroscience hypotheses and infer some properties about the physiology.

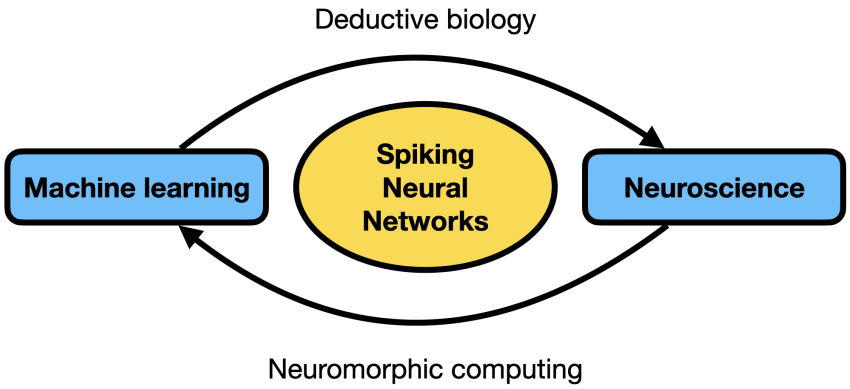

Figure 1.1: General approach using SNNs at the intersection of machine learning and neuroscience.

1.3 Outline and contributions

This thesis is organised into seven chapters, with the main contributions presented in Chapters 3-6. The current chapter introduces the overall motivation, approach and outline for the thesis.

Chapter 2 provides relevant background material for the thesis. It begins with an overview of the architectural components of a typical ASR system, including its evaluation metrics and frameworks. The chapter then covers ANNs, with a particular focus on Recurrent Neural Networks (RNNs), which will be the central type of architecture throughout the thesis. Finally, the databases used in this thesis are presented at the end of the chapter.

In Chapter 3, we start from a recent Bayesian derivation of recurrence in conventional ANNs (Garner and Tong, 2021) and derive a layer-wise feedback without the approximation of previous work. We show that the natural feedback domain is log-probability, leading to the softplus activation function. Adding an update gate yields a unit very close to the Light Gated Recurrent Unit (Li-GRU) (Ravanelli, Bordes, and Bengio, 2018) but with a valid probabilistic formulation. In a second step, we come back to the special case where hidden features are independent. We show that the corresponding unit-wise feedback can be described as a first-order two-state Hidden Markov Model (HMM) and derive a backward recursion that enables to consider future context with no additional parameters. While the contributions are mostly theoretical, the resulting lightweight models give competitive results on ASR tasks.

In Chapter 4, we shift to the main focus of biologically inspired SNNs, which constitute a special case of RNNs. We derive our approach from continuous-time single neuron dynamics (Gerstner and Kistler, 2002) using both differential equations and kernel-based formulations. Instead of a moving threshold adaptation mechanism as defined by Bellec et al. (2018) and typically used in SNNs (Yin, Corradi, and Bohté, 2020; Yin, Corradi, and

Bohté, 2021; Salaj et al., 2021; Shaban, Bezugam, and Suri, 2021), our implementation uses a recovery current coupled to the membrane potential, for which we also derive stability conditions. With the idea of integrating SNNs into modern deep learning frameworks, we review relevant training approaches and conclude that the surrogate gradient method is most appropriate. Overall this chapter serves as a theoretical baseline for our SNN approach, that we apply to several tasks in the subsequent chapters.

In Chapter 5, the SNN approach defined in Chapter 4 is integrated into deep learning frameworks to serve as a physiologically inspired encoder for speech recognition tasks. We begin with speech command recognition where our approach convincingly improves upon previous efforts (Yin, Corradi, and Bohté, 2020; Yin, Corradi, and Bohté, 2021; Salaj et al., 2021; Shaban, Bezugam, and Suri, 2021), achieving new state-of-the-art results with SNNs. We then tackle the more challenging task of Large Vocabulary Continuous Speech Recognition (LVCSR), where, to the best of our knowledge, our work is the first to successfully integrate and train a surrogate gradient spiking encoder. On both tasks, comparisons with conventional RNNs reveal that our SNNs achieve competitive performance while significantly reducing computational cost. We also confirm the effectiveness and computational efficiency of using adaptation in the neuron model compared to layer-wise recurrent connections.

In Chapter 6, we define a new ASR architecture to favour physiology over performance. By including auditory nerve fibers with plausible frequency sensitivity and by minimising the size of ANN modules, almost all processing power is given to the central SNN. After a preliminary architectural analysis, we concentrate on analysing the spiking activity throughout the trained encoder. Our study reveals the natural emergence of neural oscillations in the form of cross-frequency couplings, consistent with neuroscience observations. We also find that these oscillations occur exclusively during speech processing and not when processing other types of noise. Our results suggest that the SNN has learned similar processes to those of the biological auditory mechanism.

A final chapter is dedicated to an overall conclusion with directions for future work.

2 Background

2.1 Automatic speech recognition

The field of ASR seeks to enable computer systems to understand and process human speech, transforming spoken language into written text.

Early ASR systems employed template matching techniques, where spoken words were matched against pre-recorded templates. The introduction of statistical models, particularly HMMs, marked a significant milestone in the development of ASR. In recent years, the advent of deep learning, enabled by scalable networks and extensive datasets, has revolutionised ASR, allowing ANNs to outperform traditional HMM-based systems.

Despite significant advancements, substantial challenges remain, particularly for low-powered, on-device applications. The high computational demand and energy consumption of traditional ASR systems limits their feasibility on resource-constrained or battery-operated devices such as smartphones, home pods and wearable technology. The need for efficient, low-power models is further emphasised by the growing demand for privacy-preserving solutions, where on-device processing is preferred over cloud-based alternatives. This thesis addresses these challenges by developing parameter and energy-efficient alternatives to traditional ANN speech encoders.

2.1.1 Architecture

Starting from the input audio signal, ASR systems typically apply feature extraction and acoustic modelling, which produces an encoded representation that is then decoded into output text sequences. In this thesis, our focus lies on the acoustic model, which corresponds to the encoder component in modern sequence-to-sequence, encoder-decoder architectures. Nonetheless, to provide a comprehensive understanding of ASR architectures, we present all relevant components below.

Feature extraction

Feature extraction involves transforming raw audio signals into a more compact and informative representation, making the subsequent stages of acoustic modeling and decoding more effective. Common practice involves splitting the waveform into overlapping frames with a length of 25 ms and a shift of 10 ms. Each frame is usually multiplied by a bell-shaped windowing function, such as the Hann window, before applying a Fourier transform to convert it into a vector of frequency components. The resulting power spectrum can then be processed using triangular filters spaced according to the Mel scale to emulate the human ear's perception of sound frequencies. Finally, to compress the dynamic range, a logarithm is applied to the Mel-filtered spectrum.

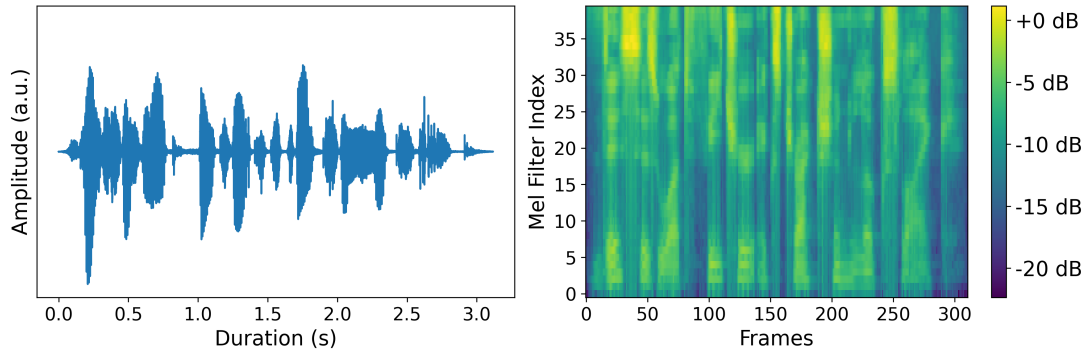

Figure 2.1: Raw audio signal (left) and its Mel filterbank feature representation (right).

To illustrate, let us consider a single-channel audio signal with a sampling rate of 16 kHz and a duration of about three seconds. As presented in Figure 2.1, the above procedure using 40 Mel filters transforms the audio signal with length 49,853 into a two-dimensional filterbank representation with 311 frames and 40 frequency bins, where the number of frames is calculated using the formula

$$\text{Number of frames} = \left\lceil \frac{\text{Signal length} - \text{Frame length}}{\text{Frame shift}} \right\rceil + 1\,. \tag{2.1}$$

While the Mel filterbank-based feature extraction described above is a common procedure used in popular ASR baselines like the Conformer (Gulati et al., 2020) or Whisper (Radford et al., 2023), alternative approaches can use different types of filters or additional processing techniques to extract features such as Mel-Frequency Cepstral Coefficients (MFCCs) or Feature-space Maximum Likelihood Linear Regression (fMLLR). Additionally, some approaches bypass traditional feature extraction altogether and directly apply Convolutional Neural Networks (CNNs) on the waveform or use trainable filters (Ravanelli and Bengio, 2018; Baevski et al., 2020; Zeghidour et al., 2021).

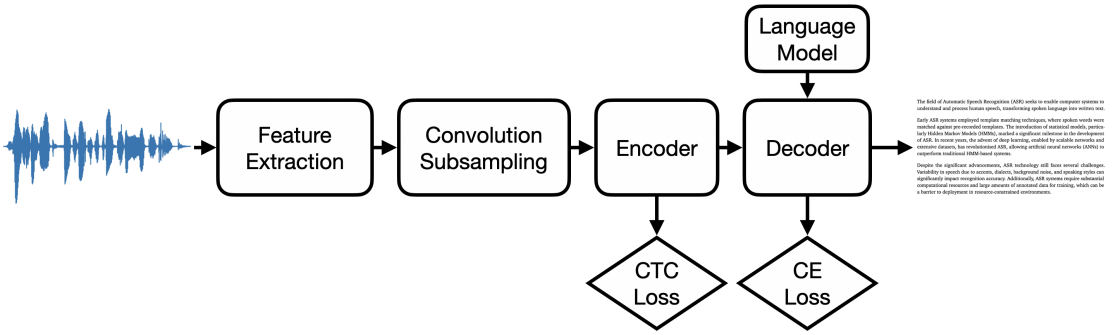

Figure 2.2: ASR architecture to transform input speech waveform into output text.

Subsampling convolutional module

After feature extraction, further reduction of the sequence length can optionally be carried out using convolutional subsampling before entering the encoder (Karita et al., 2019; Gulati et al., 2020; Radford et al., 2023).

Acoustic model or encoder

After the initial stages of feature extraction and convolutional subsampling have condensed the raw audio signal into a more compact and informative representation, the acoustic model, or encoder, further processes this representation to extract higher-level features that capture phonemic or subword information. The output of the encoder typically consists of a two-dimensional array, omitting the batch dimension, representing a sequence of probabilities over time across phoneme or subword classes.

The processing within the acoustic model must therefore map auditory features to phonemic features before classifying them into token classes (phonemes, subwords, words). To achieve this, the ASR encoder must effectively handle context dependencies, such as how the pronunciation of a particular phoneme may vary depending on its neighboring phonemes in a spoken word or sentence. These context dependencies are typically shorter-range compared to those addressed during the decoding stage. Therefore, the encoder mainly focuses on capturing local contextual information relevant for recognising individual phonetic or linguistic units within the audio signal.

While traditional approaches relied on HMMs and Gaussian Mixture Models, RNNs, and more recently, Transformers have been adopted as common choices for the encoder component in modern ASR systems.

Decoder

Upon identifying phonetic units from the audio input, the encoder passes them to the decoder, which converts the probability distributions over these units into the most likely sequence of words.

Traditional HMM-based ASR systems often used a pronunciation dictionary, called a lexicon, to provide linguistic constraints and guide the selection of words in the output sequence. By directly mapping input audio features to subword units, modern systems often bypass the need for a phoneme-level lexicon.

ASR decoding involves handling variable-length input and output sequences, accommodating the inherent flexibility of spoken language. One challenge arises from the fact that the encoder output sequences are typically longer than their corresponding ground truth token sequences. Two prevalent methods for addressing this are Connectionist Temporal Classification (CTC) decoding and RNN or Transformer-based decoders. We provide an overview of both approaches below.

RNN or Transformer based decoders

In RNN or Transformer-based ASR decoders, the output sequence $\boldsymbol{y} = [y_1, y_2, \ldots, y_{T_y}]$ of length T_y represents discrete class labels and is generated token by token. At step t, the generation process is conditioned on the entire input sequence $\boldsymbol{x} = [x_1, x_2, \ldots, x_{T_x}]$ of length T_x, provided by the encoder, as well as on the previously generated tokens $\boldsymbol{y_{<t}} = [y_1, y_2, \ldots, y_{t-1}]$. This conditioning yields a refined probability distribution over the token classes, expressed as follows

$$P\Big(\boldsymbol{y} \,\Big|\, \boldsymbol{x}\Big) = \prod_{t=1}^{T_y} P\Big(y_t \,\Big|\, \boldsymbol{y_{<t}}, \boldsymbol{x}\Big)\,. \tag{2.2}$$

Beam search is commonly used to find the most likely transcription. For the first step, it starts the output sequence with the initial start-of-sequence token. At each subsequent decoding step, all possible one-token continuations of the current hypotheses are scored based on the following equation,

$$P\Big(\boldsymbol{y_{<t+1}} \,\Big|\, \boldsymbol{x}\Big) = P\Big(\boldsymbol{y_{<t}} \,\Big|\, \boldsymbol{x}\Big) \cdot P\Big(y_t \,\Big|\, \boldsymbol{y_{<t}}, \boldsymbol{x}\Big)\,, \tag{2.3}$$

where $\boldsymbol{y_{<t}}$ is the current partial sequence and y_t is one of the next possible tokens. After scoring, the algorithm selects the top-k hypotheses based on their scores, where k is the beam width. A larger beam width allows for more exploration but increases computational complexity. The process repeats until either the end-of-sequence token is generated or a maximum sequence length is reached. The hypothesis with the highest overall score is then selected as the final output sequence.

To improve transcription accuracy, ASR decoders often incorporate a language model to rerank the final top-k output sequences at the end of the beam search procedure. Language models can assess the likelihood of the next token based on the preceding ones, denoted as $P_{\text{LM}}(y_t \Big| \boldsymbol{y_{<t}})$, so that the rescoring is computed as,

$$P\Big(\boldsymbol{y} \,\Big|\, \boldsymbol{x}\Big) = \prod_{t=1}^{T_y} P\Big(y_t \,\Big|\, \boldsymbol{y_{<t}}, \boldsymbol{x}\Big) \cdot P_{\text{LM}}\Big(y_t \,\Big|\, \boldsymbol{y_{<t}}\Big)\,. \tag{2.4}$$

During training with RNN or Transformer-based ASR decoders, the scores $P\Big(y_t \,\Big|\, \boldsymbol{y_{<t}}, \boldsymbol{x}\Big)$ are compared against the ground truth labels for each token y_t in the sequence using Cross-Entropy (CE) loss. Both predicted and ground truth sequences are often padded to a maximum length, so that all sequences in a mini-batch have the same length. During the loss computation, a mask is applied to ignore padded elements, comparing only the valid tokens up to the length of the shorter sequence.

CTC decoding

As introduced by Graves, Fernández, et al. (2006), a CTC decoder receives probabilities from the encoder's softmax output layer, with one more unit as there are token classes, typically representing phonemes or subwords. This additional unit is referred to as the blank token, and corresponds the probability of observing no particular class at any given time step. The encoder's output can then be interpreted as the alignment probabilities of all possible token sequences with the encoder's input sequence.

The mapping $\mathcal{B}$ applied to a sequence is defined as removing all blanks and condensing repeated tokens into single occurrences. The probability of a particular token sequence $\boldsymbol{y} = [y_1, y_2, \ldots, y_{T_y}]$ is then computed by summing over all possible alignments $\boldsymbol{y'}$ that include blanks and repeated tokens, which can be reduced to the same final sequence, i.e., $\mathcal{B}(\boldsymbol{y'}) = \boldsymbol{y}$.

$$P(\boldsymbol{y} \,|\, \boldsymbol{x}) = \sum_{\mathcal{B}(\boldsymbol{y'})=\boldsymbol{y}} P(\boldsymbol{y'}|\boldsymbol{x})\,. \tag{2.5}$$

The CTC decoded token sequence is therefore conditioned to be at most as long as the encoder's output sequence, $T_y \leq T_x$. A differentiable forward-backward algorithm is used to efficiently compute $P(\boldsymbol{y} \,|\, \boldsymbol{x})$ from the encoder's outputs without any additional trainable parameter. These probabilities can then be used to define a loss function that maximises the likelihood of the ground truth sequence given the input.

At inference time, beam search can be applied to decode the CTC probabilities $P(\boldsymbol{y} \,|\, \boldsymbol{x})$ into the most likely token sequences.

2.1.2 Keyword spotting

Keyword Spotting (KWS) is a specialised area of speech recognition technology focused on detecting specific, predefined words or phrases within an audio stream. Unlike general ASR systems, which aim to transcribe entire spoken passages, KWS systems are inherently simpler as they only listen for and identify a finite set of target words or phrases. Similar to ASR systems, a typical KWS architecture also involves feature extraction and encoding. However, instead of using a sequence-to-sequence decoder, the encoder outputs in KWS are directly mapped to keyword probabilities.

2.1.3 Evaluation metric

The performance of an ASR system is most commonly evaluated using the Word Error Rate (WER) metric, which is calculated by comparing the ASR system's output with reference transcripts from the testing split of the dataset. It is defined as follows:

$$\text{WER} = \frac{\text{Substitutions} + \text{Deletions} + \text{Insertions}}{\text{Number of words in reference transcription}} \tag{2.6}$$

WER provides a clear, quantitative measure to compare different ASR systems or models. Note that for KWS, it is more common to report accuracy rather than error rate. Accuracy is defined as,

$$\text{Accuracy} = \frac{\text{Correctly predicted keywords}}{\text{Total number of keyword predictions}}. \tag{2.7}$$

Credible intervals on error rates

When testing an ASR or KWS model, the error rate corresponds to the fraction of words or phonemes that were incorrectly predicted by the model. The number of successes in such an experiment with binary outcomes can be modelled as a binomial distribution. For trivial priors, the posterior of the binomial distribution is beta distributed. Throughout the thesis, we will use the equal-tailed 95% credible intervals on the posterior distribution to compute error bars for the reported error rates.

2.1.4 Frameworks

In this thesis, we trained and evaluated ASR models using the pytorch-kaldi toolkit (Ravanelli, Parcollet, and Bengio, 2019), before updating to SpeechBrain (Ravanelli, Parcollet, Plantinga, et al., 2021). Both are open-source PyTorch-based (Paszke, Gross, Massa, et al., 2019) toolkits. For keyword spotting, we wrote our own framework, released at https://github.com/idiap/sparch.

2.2 Hidden Markov models

2.2.1 Discrete Markov chains

A discrete Markov chain is a stochastic model that defines the evolution of a system over discrete time steps $\{h_t \,|t = 1, 2, \ldots, T\}$. At each step, the system occupies one of H distinct states, i.e., $h_t \in \{\phi_1, \phi_2, \ldots, \phi_H\}$. For a n-th order Markov chain, the Markov property states that the current state of the system, h_t, depends only on a fixed number of previous states, $h_{t-1}, h_{t-2}, \ldots, h_{t-n}$.

In a first-order Markov chain, the current state h_t only depends upon the preceding state h_{t-1}, so that the joint probability of a particular sequence of T random variables $\mathcal{H} = [h_1, h_2, \ldots, h_T]$ can be computed as,

$$P(\mathcal{H}) = P(h_1) \prod_{t=2}^{T} P(h_t|h_{t-1})\,. \tag{2.8}$$

The evolution of the system is then fully described by the initial state distribution vector $a \in [0, 1]^H$ representing the initial state probability of the Markov chain $P(h_1)$,

$$a = \Big[P(h_1 = \phi_1), P(h_1 = \phi_2), \ldots, P(h_1 = \phi_H)\Big] \quad \text{with} \quad \sum_{i=1}^{H} a_i = 1\,, \tag{2.9}$$

and the transition probabilities matrix $A \in [0, 1]^{H\times H}$ to evaluate $P(h_t|h_{t-1})$,

$$A_{ij} = P\Big(h_t = \phi_j \Big| h_{t-1} = \phi_i\Big) \quad \text{with} \quad \sum_{j=1}^{H} A_{ij} = 1 \quad \forall i \in \{1, 2, \ldots, H\}\,. \tag{2.10}$$

2.2.2 Application to speech recognition

HMMs are typically based on first-order Markov chains and have long been employed on speech processing tasks (Rabiner, 1989). By defining an autoregressive process, HMMs are effective at capturing the sequential dependencies inherent in speech data, where each observation is influenced by its preceding context.

In these models, speech is represented as a series of observations $[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T] \in \mathbb{R}^{T\times F}$, where $\boldsymbol{x}_t$ denotes a speech feature vector of size F at time t. An HMM assumes that these observations are generated by a sequence of underlying hidden states $\{h_t \,|t = 1, 2, \ldots, T\}$ that follow a first-order Markov chain but are not directly observable.

Each hidden state $h_t = \phi_i$ must therefore be associated with an additional emission probability distribution $b_i(\boldsymbol{x}_t) = P(\boldsymbol{x}_t|h_t = \phi_i)$ which determines the likelihood of emitting a particular observation given the current hidden state.

An HMM is then fully parameterised by the initial state distribution vector a, the transition probabilities matrix A and the set of emission probability distributions $B(\boldsymbol{x}_t)$,

$$B(\boldsymbol{x}_t) = \Big[b_1(\boldsymbol{x}_t), b_2(\boldsymbol{x}_t), \ldots, b_H(\boldsymbol{x}_t)\Big]\,. \tag{2.11}$$

HMM-based acoustic model

In HMM-based acoustic models, each phonetic unit is typically represented by a three-state HMM. This structure captures the statistical properties of the acoustic features corresponding to a particular phoneme. The three states represent different phases of the phoneme: the beginning (transition from the previous phoneme), the middle (stable, central portion of the phoneme), and the end (transition to the next phoneme). This model is trained to output $P(\text{phonetic units}|\text{acoustic observations})$ using labeled acoustic data where the phonetic units are known.

The training process involves estimating the HMM parameters to best represent the observed acoustic features for each phoneme. To estimate the parameters of an HMM (i.e., a, A and B), the Baum-Welch algorithm (Baum and Petrie, 1966; Baum, Petrie, et al., 1970; Baum et al., 1972; Bahl, Jelinek, and Mercer, 1983) is typically employed. This algorithm is a special application of the expectation-maximisation algorithm (Dempster, Laird, and Rubin, 1977) for HMMs. During the expectation step, the forward-backward algorithm (Chang and Hancock, 1966; Rabiner, 1989) is used to compute the expected values of the hidden state sequences given the current parameter estimates. In the subsequent maximisation step, the HMM parameters are updated to maximise the expected likelihood obtained in the expectation step. The algorithm then iteratively alternates between these two steps until either convergence or a maximum number of iterations is reached.

HMM-based decoding

HMM-based decoding aims to find the most likely sequence of words from a given sequence of acoustic features. The decoding starts by using the HMM-based acoustic model to identify the top-k most likely phoneme sequences based on the acoustic inputs. These phoneme sequences are then mapped to sequences of linguistic units – such as words, triphones or syllables – using a fixed pronunciation dictionary. Finally, a language model evaluates the likelihood of these potential sequences of linguistic units. Throughout this overall decoding process, the Viterbi algorithm (Viterbi, 1967; Rabiner, 1989) plays a crucial role in efficiently searching through the possible phoneme sequences to determine the one that maximises the likelihood of the resulting sequence of linguistic units.

2.3 Artificial neural networks

The vast majority of neural networks used in modern machine learning tasks are organised in layers of artificial neurons from the second generation as defined by Maass (1997), and trained via SGD. Here we are interested in networks that can process sequential inputs, in particular speech. Starting from some input tensor $x \in \mathbb{R}^{B\times T\times N^0}$ with batch size B, length T and number of features N^0, an ANN processes the information layer by layer, so that the l-th layer outputs a tensor $y^l \in \mathbb{R}^{B\times T\times N^l}$, where N^l is the number of neurons in the layer. The outputs y^L from the last layer represent predictions that are evaluated using a loss function,

$$\mathcal{L} = D\Big(y^L - y_{\text{desired}}\Big) \tag{2.12}$$

where $D(\cdot)$ evaluates some cost associated with prediction errors from the current mini-batch that we intend to minimise. All trainable parameters θ inside the ANN architecture can be updated according to the gradient of the loss with respect to them,

$$\theta \to \theta - \lambda\, \nabla_\theta \mathcal{L}\,, \tag{2.13}$$

where $\lambda > 0$ is some learning rate. The model iteratively learns to make better predictions by looping over mini-batches of training data. In each iteration, the loss is computed and the parameters updated accordingly. After a complete pass over the training data, the model is typically tested on some validation split of the data which it never trains on. The procedure repeats until sufficient convergence is reached, and the model is finally evaluated on some remaining testing split of the data.

To compute gradients efficiently, the backpropagation algorithm (Rumelhart, Hinton, and Williams, 1986) traverses the network in reverse order, leveraging the chain rule of derivatives to reuse relevant partial derivatives from earlier layers. In the PyTorch deep learning framework (Paszke, Gross, Massa, et al., 2019) employed throughout this thesis, a computational graph is dynamically constructed during each forward pass, enabling automatic differentiation (Paszke, Gross, Chintala, et al., 2017) without the need to explicitly implement the backward pass.

2.3.1 Multilayer perceptrons

A Multilayer Perceptron (MLP) refers to a feedforward ANN where each neuron in a layer is fully connected to all neurons in the subsequent layer. Using feedforward weights $W^l \in \mathbb{R}^{N^{l-1}\times N^l}$ and bias $b^l \in \mathbb{R}^{N^l}$, the l-th layer performs a linear combination of inputs y^{l-1} from the previous layer,

$$I^l = W^l\, y^{l-1} + b^l \tag{2.14}$$

before applying a nonlinear activation function $g(\cdot)$,

$$y^l = g\Big(I^l\Big)\,. \tag{2.15}$$

Using a sigmoid activation function $g(x) = \Big(1 + e^{-x}\Big)^{-1}$ for instance, the neuron output $y \in [0, 1]$ can be interpreted as the firing rate of a biological spiking neuron over some arbitrary period of time.

This form of ANN processes all time steps at once without maintaining any memory of previous inputs. This means that MLPs are incapable of considering temporal dependencies or contextual information across different time steps. MLPs are therefore often combined with other modules that can deal with context dependencies, or reserved for tasks that require processing independent data points with no temporal dependency.

2.3.2 Convolutional neural networks

In the wider deep-learning field, modelling context is crucial. Analogous to finite impulse response filters, CNNs apply fixed length kernels that slide along the temporal dimension of the input to capture local patterns.

Convolution and cross-correlation operations

Convolution is a mathematical operation on two functions f, g that produces a third function $(f * g)$ expressing how the shape of one is modified by the other. It is defined as,

$$(f * g)(t) := \int_0^\infty f(t')\, g(t - t')dt' \tag{2.16}$$

for a continuous variable $t \in [0, \infty)$ and as,

$$(f * g)[n] := \sum_{m=0}^{N-1} f[m]\, g[n - m] \tag{2.17}$$

for a discrete variable $n \in \{0, 1, \ldots, N-1\}$. The cross-correlation operation $\star$ is similar to the convolution operation $*$ for real-valued functions of continuous or discrete variables,

$$f(t) \star g(t) = f(-t) * g(t) = \int_0^\infty f(t')\, g(t + t')dt'\,, \tag{2.18}$$

except in contrast to convolution, $f(t)$ and $g(t)$ are compared without reversing the order of one of them.

Single channel, one-dimensional convolution

In the context of neural networks, a single trainable filter or kernel $W_K \in \mathbb{R}^{K_t}$ with size K_t can directly be applied to some discrete input sequence $x \in \mathbb{R}^T$ by sliding the filter over the input data and computing the dot product between the filter and the overlapping region of the input,

$$(W_K \star x)[t] = \sum_{k=0}^{K_t-1} W_K[k] \cdot x[t+k]\,, \tag{2.19}$$

where $t \in \{0, 1, \ldots, T-1\}$. Cross-correlation is typically used instead of convolution to simplify the implementation by not requiring to flip the trainable filter.

Single channel, two-dimensional convolution

During speech processing, two-dimensional convolution is often preferred, so that for a two-dimensional input $x \in \mathbb{R}^{T \times N}$, the trainable filter $W_K \in \mathbb{R}^{K_t \times K_f}$ slides over both temporal and feature dimensions,

$$(W_K \star x)[t, j] = \sum_{k=0}^{K_t-1} \sum_{m=0}^{K_f-1} W_K[k, m] \cdot x[t+k, j+m]\,. \tag{2.20}$$

Multi-channel, two-dimensional convolution

Now if input $x \in \mathbb{R}^{T \times N \times C_{\text{in}}}$ has an additional channel dimension of size C_{in}, we can have a trainable filter W_K with size $K_t \times K_f \times C_{\text{in}} \times C_{\text{out}}$. For time step t, feature n and output channel $c \in \{0, 1, \ldots, C_{\text{out}} - 1\}$, we now have,

$$(W_K \star x)[t, j, c] = \sum_{k=0}^{K_t-1} \sum_{m=0}^{K_f-1} \sum_{c'=0}^{C_{\text{in}}-1} W_K[k, m, c', c] \cdot x[t+k, j+m, c']\,, \tag{2.21}$$

where an output channel is produced by summing the convolved results across all input channels. These operations can be efficiently computed on GPUs by leveraging highly optimised operations for matrix multiplication and convolution.

Convolutional layer

In a CNN, the l-th convolutional layer applies a trainable kernel or filter W_K^l to the input y^{l-1} from the previous layer,

$$I^l = W_K^l \star y^{l-1} + b^l\,, \tag{2.22}$$

where b^l is the trainable bias. This operation reduces the temporal and feature dimensions from T^{l-1} to T^l and from N^{l-1} to N^l respectively. The new dimensions depend on

the kernel sizes $\{K_t, K_f\}$ along the temporal and feature dimensions, as well as the corresponding amounts of padding $\{P_t, P_f\}$ and stride $\{S_t, S_f\}$,

$$T^l = \left\lfloor \frac{T^{l-1} + 2\,P_t - K_t}{S_t} + 1 \right\rfloor \quad \text{and} \quad N^l = \left\lfloor \frac{N^{l-1} + 2\,P_f - K_f}{S_f} + 1 \right\rfloor \tag{2.23}$$

where $\lfloor \cdot \rfloor$ is the floor function. Similar to the MLP presented in Section 2.3.1, a nonlinear activation such as the Rectified Linear Unit (ReLU) is then applied to I^l as in Eq. (2.15). The main difference between MLPs and CNNs therefore lies in the matrix operation they apply to the input. In a convolutional layer, the same trainable filter is applied across different parts of the input. This form of weight sharing allows convolutional layers to require fewer parameters compared to fully connected layers, especially for large inputs like images. Additionally, the resulting translation invariance enables the network to detect features regardless of their position in the input.

2.3.3 Recurrent neural networks

While CNNs excel at capturing local dependencies through the use of finite impulse response filters, signal processing often favours recurrence for processes that are understood to be autoregressive, the analogy being to infinite impulse response filters. A pertinent example is in ASR, where RNNs have played a crucial and foundational role, remaining key in the development and understanding of ASR systems.

Feedback mechanism

The distinctive feature of RNNs lies in their utilisation of feedback connections. In addition to a linear combination of outputs $y_t^{l-1} \in \mathbb{R}^{N^{l-1}}$ from the previous layer $l-1$ at the current time step t, an RNN layer receives a linear combination of its own outputs $y_{t-1}^l \in \mathbb{R}^{N^l}$ from the previous time step $t-1$. Unlike MLPs which process time steps independently, recurrent connections enable RNNs to capture temporal dependencies in sequential data.

Standard non-gated RNNs

Mathematically, the overall stimulus of neurons in the l-th layer can be expressed in vectorised form as,

$$I_t^l = W^l\, y_t^{l-1} + V^l\, y_{t-1}^l + b^l\,, \tag{2.24}$$

where $W^l \in \mathbb{R}^{N^{l-1} \times N^l}$ and $V^l \in \mathbb{R}^{N^l \times N^l}$ represent the trainable feedforward and recurrent weight matrices respectively, while $b^l \in \mathbb{R}^{N^l}$ is a trainable bias vector. Due to the feedback on outputs from the previous time step y_{t-1}^l, a recurrent layer cannot directly parallelise its computations over the time dimension, which typically leads to longer

training time compared to MLPs and CNNs.

Standard non-gated RNNs simply follow Eq. (2.24) with an activation function similar to Eq. (2.15). Nevertheless, this basic form of RNNs has been observed to encounter difficulties with long-term dependencies due to problems like vanishing and exploding gradients during training (Bengio, Simard, and Frasconi, 1994). To overcome these challenges, more advanced RNN architectures were defined to incorporate gating mechanisms, designed to regulate the flow of information through the network. On top of Eq. (2.24), each added gate employs its own unique feedforward and recurrent weights, thereby increasing the total number of trainable parameters.

Gated RNNs

The most successful recurrent architectures are based on the Long Short-Term Memory (LSTM), defined by Hochreither and Schmidhuber (1997), with a memory cell and input and output gates to filter out irrelevant information and tackle vanishing and exploding gradient problems. An additional forget gate and peephole connections were subsequently added by Gers, Schmidhuber, and Cummins (2000) and Gers, Schraudolph, and Schmidhuber (2002). A simplification of the unit then resulted into the Gated Recurrent Unit (GRU) by Cho et al. (2014), where the input and forget gate were combined into a single update gate, and the output gate was changed to a reset gate that acts on the feedback inside the cell state. Further efforts to reduce the size of recurrent units were pursued by Zhou et al. (2016) with the minimally gated unit, where a single gate is used twice as the update and reset gates of the GRU. In the same spirit of getting rid of redundancies, Ravanelli, Brakel, et al. (2017) and Ravanelli, Bordes, and Bengio (2018) proposed an alternative simplification of the GRU called Li-GRU by removing the reset gate altogether. The Li-GRU notably outperformed the GRU and LSTM on several ASR tasks (Ravanelli, Bordes, and Bengio, 2018), where it represents a parameter-efficient alternative to LSTMs and GRUs.

A Li-GRU layer with input x and output y can be implemented as,

$$\tilde{y}_t = \text{ReLU}\Big(W\, x_t + V\, y_{t-1} + b\Big) \tag{2.25a}$$

$$z_t = \text{sigmoid}\Big(W_z\, x_t + V_z\, y_{t-1} + b_z\Big) \tag{2.25b}$$

$$y_t = z_t \odot y_{t-1} + \Big(1 - z_t\Big) \odot \tilde{y}_t\,. \tag{2.25c}$$

Here $\tilde{y}_t \in [0, \infty)$ represents a candidate output state and is computed by applying a ReLU to the standard RNN update equation from Eq. (2.24). On the other hand, z_t is a gate with values in $(0, 1)$ which defines the weighted combination of the previous output y_{t-1} with the current candidate output state $\tilde{y}_t$ using the element-wise product $\odot$.

Bidirectional RNNs

As defined by Schuster and Paliwal (1997) and similar to Graves, Jaitly, and Mohamed (2013), PyTorch implements bidirectionality in all its RNN types (RNN, GRU and LSTM) by essentially running two separate RNNs for each layer: one to process the input sequence in the forward direction, and the other to process it in the backward direction. The outputs of these two RNNs are then concatenated at each time step to form the final output.

As notably used in pytorch-kaldi (Ravanelli, Parcollet, and Bengio, 2019) for the Li-GRU implementation, bidirectionality can also be made more parameter-efficient by sharing weights when processing forward and backward directions. In this approach, the input to an RNN layer is duplicated, flipped and concatenated on the batch dimension so that the forward pass processes both forward and backward sequences in parallel using the same weights. The layer outputs for the forward and backward directions are then split, the backward part flipped, and concatenated on the feature dimension, doubling the number of features sent to the next layer. With this method, instead of doubling the number of parameters, the first layer does not require additional parameters and subsequent layers only double their feedforward matrix and not the recurrent one.

2.3.4 Attention-based networks

Instead of processing sequences in a step-by-step fashion like RNNs, the attention mechanism enables networks to perform parallelisable matrix multiplications over the sequence dimension, thereby fully exploiting GPU capabilities. This approach gave rise to the transformer architecture, which quickly became state-of-the-art across a wide range of tasks including speech recognition. Here, we provide a brief overview of core concepts and their relation to RNNs.

Self Attention

For self-attention, three feedforward weight matrices W_Q, W_K and $W_V \in \mathbb{R}^{N\times N}$ are applied to the input tensor $x \in \mathbb{R}^{B\times T\times N}$ over its feature dimension of size N,

$$y_Q = W_Q\, x\,, \quad y_K = W_K\, x \quad \text{and} \quad y_V = W_V\, x\,. \tag{2.26}$$

The feature dimension of the resulting queries y_Q, keys y_K and values y_V can be split into two dimensions, $N = N_H \cdot H$, where N_H represents the number of attention heads and H the number of features per head. We reshape these tensors to $B \times N_H \times T \times H$. The computations in the attention mechanism are parallelised over the batch and head dimensions, similar to broadcasting in Python, where operations are applied element-wise across specified dimensions without explicit loops. The dimensions of interest in the

attention computations are therefore the sequence and feature dimensions.

For each query time step $t = 1, 2, \ldots, T$, the query vector of features $y_Q[b, h, t, :] \in \mathbb{R}^H$ is compared with all $t' = 1, 2, \ldots, T$ key vectors $y_K[b, h, t', :] \in \mathbb{R}^H$ in the sequence. The t-th query vector is compared with the t'-th key vector via a dot-product over the feature dimension,

$$y_Q[b, h, t, :] \cdot y_K[b, h, t', :] = \sum_{n=1}^{H} y_Q[b, h, t, n]\, y_K[b, h, t', n]\,, \tag{2.27}$$

which produces a single value for each comparison. The attention scores then correspond to the scaled dot-product between $y_Q \in \mathbb{R}^{B \times N_H \times T \times H}$ and $y_K^\intercal \in \mathbb{R}^{B \times N_H \times H \times T}$,

$$\frac{y_Q \cdot y_K^\intercal}{\sqrt{H}} \in \mathbb{R}^{B \times N_H \times T \times T}\,, \tag{2.28}$$

so that for each mini-batch element b and attention head h, a $T \times T$ score matrix represents the relationships between time steps. Because that matrix does not need to be symmetric, how time step t is related to t' can be different to how t' is related to t. Taking the softmax of the score matrix produces the attention weights $A \in \mathbb{R}^{B \times N_H \times T \times T}$,

$$A = \text{softmax}\left(\frac{y_Q \cdot y_K^\intercal}{\sqrt{H}}\right), \tag{2.29}$$

which sum to one over the last sequence dimension. These attention weights are finally applied to the value tensor y_V across its sequence dimension, which outputs a tensor with shape $B \times N_H \times T \times H$ that can be reshaped back to $B \times T \times N$ to match the input tensor x. Overall the self-attention operation,

$$\text{SelfAttention}(x) = \text{softmax}\left(\frac{(W_Q\, x) \cdot (W_K\, x)^\intercal}{\sqrt{H}}\right) \cdot (W_V\, x)\,, \tag{2.30}$$

does not affect the shape of the input tensor x, enabling the use of residual connections (He et al., 2016). A transformer (Vaswani et al., 2017) layer $T^l(x)$ then combines self-attention with a feedforward module, typically a two-layered MLP. Using layer normalisation (Ba, Kiros, and Hinton, 2016) and residual connections (He et al., 2016), it can be defined as,

$$y = \text{LayerNorm}\Big(\text{SelfAttention}(x) + x\Big) \tag{2.31a}$$

$$T^l(x) = \text{LayerNorm}\Big(\text{MLP}(y) + y\Big)\,. \tag{2.31b}$$

Reducing the quadratic complexity

Despite its remarkable performance, self-attention has computational and memory requirements that scale quadratically $\mathcal{O}(T^2)$ with the sequence length T, limiting its

applicability to long sequences. Several approaches have managed to reduce the complexity of transformers to $\mathcal{O}(T)$, reducing memory usage and enabling faster inference on autoregressive tasks. The Performer (Choromanski et al., 2020) approximates the attention mechanism via positive orthogonal random features. The Linformer (Wang, Li, et al., 2020) projects the n-dimensional sequence into a lower k-dimensional space where $k \ll n$. The Longformer (Beltagy, Peters, and Cohan, 2020) combines local windowed attention with global attention, restricting most computations to a local context. Finally, Katharopoulos et al. (2020) replaced the traditional softmax attention with a feature map based dot product attention.

Relation to RNNs

The work of Katharopoulos et al. (2020) is particularly relevant as it directly links transformers to RNNs. In particular, they show that a transformer with causal masking can be formulated as an RNN with two hidden states s_t and z_t,

$$s_t = s_{t-1} + \phi(W_K\, x_t)\phi(W_V\, x_t)^\intercal \tag{2.32a}$$

$$z_t = z_{t-1} + \phi(W_K\, x_t) \tag{2.32b}$$

$$y_t = \text{LayerNorm}\left(\frac{\phi(W_Q\, x_t)^\intercal\, s_t}{\phi(W_Q\, x_t)^\intercal\, z_t} + x_t\right) \tag{2.32c}$$

$$T^l(x_t) = \text{LayerNorm}\Big(\text{MLP}(y_t) + y_t\Big)\,. \tag{2.32d}$$

Here $\phi(\cdot)$ is a feature map that results in a positive similarity function $\text{sim}(x, y) = \phi(x)\,\phi(y)^\intercal$. In their paper, they use

$$\phi(x) = \text{elu}(x) + 1\,, \tag{2.33}$$

where $\text{elu}(x)$ is the exponential linear unit (Clevert, Unterthiner, and Hochreiter, 2015).

2.3.5 State space models

In the same spirit of leveraging parallelisable computations during training while enabling fast inference on autoregressive tasks, several State Space Model (SSM) approaches (Gu, Goel, and Ré, 2021; Gupta, Gu, and Berant, 2022; Gu, Goel, Gupta, et al., 2022; Smith, Warrington, and Linderman, 2022) have proven to be viable alternatives to transformers for handling long-range dependencies. These models have also been successfully applied to speech recognition and synthesis tasks (Miyazaki, Murata, and Koriyama, 2023; Saon, Gupta, and Cui, 2023; Shan et al., 2024). We give here a brief overview of their mechanism as they can be seen as a subclass of RNNs.

A continuous-time SSM is described by the following differential equation,

$$\frac{dh}{dt}(t) = A\,h(t) + B\,x(t)\,, \quad y(t) = C\,h(t)\,. \tag{2.34}$$

Here, an input signal $x(t) \in \mathbb{R}$ gets mapped to a hidden state representation $h(t) \in \mathbb{R}^{N\times 1}$ via matrix $A \in \mathbb{R}^{N\times N}$ and vector $B \in \mathbb{R}^{N\times 1}$, before being projected to an output signal $y(t) \in \mathbb{R}$ with $C \in \mathbb{R}^{1\times N}$. Assuming that the value of $x(t)$ remains constant over the duration of a single time step Δt, we obtain a discrete time representation where the mapping between input $x = [x_0, x_1, \ldots, x_{T-1}] \in \mathbb{R}^T$ and output $y = [y_0, y_1, \ldots, y_{T-1}] \in \mathbb{R}^T$ occurs via recurrence,

$$h_t = \bar{A}\,h_{t-1} + \bar{B}\,x_t\,, \quad y_t = C\,h_t\,. \tag{2.35}$$

Here $\bar{A} = \exp(A\Delta t)$ and $\bar{B} = (\bar{A} - \mathbb{I})\,A^{-1}\,B$ are the discrete time versions of the continuous-time parameters A and B. Assuming $x_{-1} = 0$, unrolling the recurrence over time yields,

$$y_t = \sum_{n=0}^{t} C\bar{A}^n\bar{B}\cdot x_{t-n} = \sum_{n=0}^{t} \bar{K}_n \cdot x_{t-n}\,, \tag{2.36}$$

where we define the SSM kernel $\bar{K} \in \mathbb{R}^T$ as,

$$\bar{K} = (C\bar{B}, C\bar{A}\bar{B}, \ldots, C\bar{A}^{T-1}\bar{B})\,. \tag{2.37}$$

The kernel can also be written using the continuous-time parameters as,

$$\bar{K} = \Big(C\,e^{A\cdot t\cdot\Delta t}(e^{A\cdot\Delta t} - \mathbb{I})A^{-1}B\Big)_{0\le t<T}\,. \tag{2.38}$$

Once the SSM kernel $\bar{K}$ is known, then y can be computed in parallel via discrete convolution,

$$y = \bar{K} * x\,. \tag{2.39}$$

To achieve this, different approaches (Gu, Goel, and Ré, 2021; Gupta, Gu, and Berant, 2022) assume particular structures of the underlying state space, leading to parametrisations where $\bar{K}$ is easier to compute. The SSM can then use the convolution formulation of Eq. (2.39) to speed up training and switch to the recurrent formulation of Eq. (2.36) during inference to enable autoregressive decoding.

2.4 Databases

2.4.1 TIMIT

The TIMIT dataset (Garofolo et al., 1993) provides a comprehensive and widely utilised collection of phonetically balanced American English speech recordings from 630 speakers with detailed phonetic transcriptions and word alignments. It represents a standardised benchmark for evaluating ASR model performance. The training, validation and test sets contain 3,696, 400 and 192 sentences respectively. Utterance durations vary between 0.9 to 7.8 seconds. The TIMIT dataset, along with relevant access information, can be found on the Linguistic Data Consortium website at https://catalog.ldc.upenn.edu/LDC93S1.

2.4.2 AMI

The AMI (Carletta et al., 2005) corpus consists of 100 hours of multi-modal meeting recordings. The meetings, primarily involving non-native speakers, were recorded in English across three rooms with varying acoustic properties. Because of its conversational style and the frequent overlap and interruptions between speakers, the AMI corpus presents a significant challenge for ASR tasks. We use the training, validation and testing splits of the *Full-corpus-ASR* defined in http://groups.inf.ed.ac.uk/ami/corpus/datasets.shtml where training data represents about 81 hours. The dataset is available open-access at https://groups.inf.ed.ac.uk/ami/corpus/.

2.4.3 LibriSpeech

The LibriSpeech (Panayotov et al., 2015) corpus contains about 1,000 hours of English speech audiobook data read by over 2,400 speakers with utterance durations between 0.8 and 35 seconds. There are two testing splits of the dataset: the *test-clean* set which represents 2,620 sentences for a total of 52,576 words, and the *test-other* set which corresponds to more challenging data and includes 2,939 sentences for a total of 52,343 words. The dataset is freely available at https://www.openslr.org/12.

2.4.4 Google Speech Commands

The Google Speech Commands dataset (Warden, 2018) contains one-second audio recordings of 35 spoken commands such as "yes," "no," "stop," "go," "left," "right" "up". The training, validation and testing splits contain approximately 85k, 10k and 5k examples respectively. It is available open-access at https://www.tensorflow.org/datasets/catalog/speech_commands.

2.4.5 Spiking Speech Commands Datasets

In order to rectify the absence of free spike-based benchmark datasets, Cramer et al. (2020) recently released two spiking datasets for speech command recognition using LAUSCHER, available at https://compneuro.net/datasets/.

Spiking Heidelberg Digits

The Spiking Heidelberg Digits (SHD) dataset contains spoken digits from 0 to 9 in both English and German (20 classes). The recordings are from twelve different speakers, two of which are only present in the test set. The train set contains 8,332 examples and the test set 2,088 (there is no validation set).

Spiking Speech Commands

The Spiking Speech Commands (SSC) dataset is based on the Google Speech Commands v0.2 dataset and contains 35 classes from a larger number of speakers. The number of examples in the train, validation and test splits are 75,466, 9,981 and 20,382 respectively.

3 Bayesian Recurrent Units

In machine learning, recurrent neural networks represent a fundamental type of architecture to address sequential data processing tasks such as speech recognition. They are characterised by feedback loops analogous to infinite impulse response filters, which distinguishes them from traditional feedforward neural networks, and enables them to retain and utilise information from previous steps. In this chapter we adopt a statistically rigorous approach to describe recurrence in neural networks. Our Bayesian derivation yields two novel and interpretable types of recurrent units. While the contributions are mostly theoretical, our resulting lightweight models give competitive results on speech recognition tasks.

Publication Note

The material presented in this chapter is adapted from the following publications.

- A. Bittar and P. N. Garner (2021). "A Bayesian Interpretation of the Light Gated Recurrent Unit". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2965–2969. DOI: 10.1109/ICASSP39728.2021.9414259
- A. Bittar and P. N. Garner (2022b). "Bayesian Recurrent Units and the Forward-Backward Algorithm". In: *Proc. Interspeech 2022*, pp. 4137–4141. DOI: 10.21437/Interspeech.2022-11035

The underlying code has been released and is openly accessible on GitHub at https://github.com/idiap/bayesian-recurrence

3.1 Introduction

Recurrent models have widely been employed in signal processing and statistical pattern recognition, notably in the form of Kalman's state space filter (Kalman, 1960; Scharf and Demeure, 1991) and HMMs (Baum and Petrie, 1966; Baum and Eagon, 1967; Baum, Petrie, et al., 1970; Bahl, Jelinek, and Mercer, 1983). Both approaches use a forward-backward training procedure to make a statistical estimation of the model parameters.

With the current success of machine learning techniques, speech recognition architectures can be trained in an end-to-end fashion, by exploiting auto-differentiation inside deep learning frameworks like PyTorch (Paszke, Gross, Chintala, et al., 2017). Here, recurrence is also an important concept and RNNs are commonly trained via the Backpropagation Through Time (BPTT) algorithm (Rumelhart and McClelland, 1986; Williams and Zipser, 1989), which is a generalisation of gradient descent to process sequential data. During the forward pass, a mini-batch of input examples is passed through the network, and a loss function is applied to the final outputs. During the subsequent backward computation of derivatives, the network trainable parameters are updated to minimise the loss.

Similarities between HMMs and RNNs have long been observed. Bourlard and Wellekens (1990) have shown that the outputs of RNNs approximate the maximum *a posteriori* output probabilities of HMMs trained via the Viterbi algorithm (Forney, 1973). Bridle (1990a) also showed that the *alpha* part of the forward-backward algorithm can be simulated by a recurrent network, and that the *beta* part bears similarities with the backward computation of derivatives in the training of neural networks. Bidirectional RNNs were subsequently defined by Schuster and Paliwal (1997) to explicitly allow networks to take into account future observations. This approach was later applied to gated RNNs (Graves and Schmidhuber, 2005) and is now the standard approach for LSTMs (Hochreither and Schmidhuber, 1997), GRUs (Cho et al., 2014) and Li-GRUs (Ravanelli, Bordes, and Bengio, 2018). As detailed in Section 2.3.3, bidirectional recurrent layers significantly increase the amount of trainable parameters compared to the unidirectional case.

MLPs have long been shown to have probabilistic interpretations (Bridle, 1990b; Neal, 2012). Recently, Garner and Tong (2021) have used a Bayesian interpretation of gated RNNs to derive a recurrent unit architecture similar to the GRU. By including a backward recursion through the input sequences, their probabilistic approach, analogous to a Kalman smoother, allows the consideration of future observations without requiring any additional trainable parameters. Their work also shed some light on the seemingly *ad-hoc* concepts of gates and memory cells inside commonly used recurrent units. In this chapter, we adopt a similar Bayesian approach, treating the input as a sequence of observations and interpreting the unit outputs as the probabilities of hidden features being present at each time step. Recurrence emerges naturally from Bayes's theorem which updates a

prior probability into a posterior given new observational data.

In a first part, we summarise previous work (Garner and Tong, 2021) showing that the basic sigmoid activation function arises as an instance of Bayes's theorem, and that recurrence follows from the Bayesian prior. Assuming interdependent hidden features, we derive a layer-wise recurrence without the approximations of previous work, and show that it leads to a standard non-gated recurrent unit with modest modifications to reflect the use of log-probabilities. After adding an update gate, the resulting architecture closely resembles the Li-GRU, a lightweight alternative to LSTMs and GRUs that we presented in Section 2.3.3. Although the contribution is mainly theoretical, we show that our derived unit outperforms these standard recurrent units on the TIMIT (Garofolo et al., 1993) and AMI (Carletta et al., 2005) ASR datasets.

In a second part, we come back to the simpler case of unit-wise recurrence without a gate. By assuming a latent space of independent hidden features, our derivation using transition probabilities yields a Bayesian recurrent unit equivalent to a first-order two-state HMM. Similar to the Kalman smoother (Kalman, 1960) and forward-backward algorithm (Baum et al., 1972), we derive two backward recursions and prove their equivalence. The resulting lightweight recurrent unit can be trained via gradient descent like an RNN, and the backward recursion enables the consideration of future observations without additional parameters. While this contribution is again mostly theoretical, we show that the derived unit has practical value for ASR, as it can replace significantly larger bidirectional gated RNNs without any loss of performance.

More generally, by aiming to develop the growing toolkit of Bayesian techniques applicable to deep learning, the work presented in this chapter contributes novel theoretical insights into the understanding of recurrent processes in neural networks. Additionally, by demonstrating the effectiveness of the resulting parameter-efficient recurrent units in ASR tasks, we highlight the practical advantages of a probabilistic derivation.

3.2 A Bayesian Interpretation of the Light Gated Recurrent Unit

This first part stems from our attempts to remove some of the approximations by Garner and Tong, 2021, in particular regarding the layerwise feedback. We show that

1. A probabilistic layerwise feedback can be introduced via a sigmoid unit. Without a forget mechanism, it reduces to the common fully-connected approach.

2. The natural feedback domain is log-probability, leading naturally to the softplus activation function.

The resulting architecture is very close to the Li-GRU described in Section 2.3.3, but with a valid probabilistic formulation. We hence add an update gate, yielding a Light Bayesian Recurrent Unit (Li-BRU) which forms a basis for evaluation in terms of architecture and number of trainable parameters. While we intend the main contribution to be theoretical, augmenting the evolving toolkit of Bayesian components in deep learning, a modest evaluation shows that the resulting collection of modifications outperforms Li-GRUs, GRUs and LSTMs on ASR tasks.

3.2.1 Bayesian interpretation of recurrence

Consider an input sequence $\boldsymbol{X}_T = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_T] \in \mathbb{R}^{F \times T}$ of length T, where each observation $\boldsymbol{x}_t$ is a vector with F input dimensions. We assume that there are H hidden features $\{\phi_i \,|i = 1, \dots, H\}$ that we wish to detect along the sequence. At each time step t, a feature has two possible states: present or absent, that we write as $\phi_{t,i}$ and $\neg\phi_{t,i}$ respectively. Using a Bayesian approach, we want to build a layer of H recurrent units that will output the stacked probabilities $\boldsymbol{h}_t := P(\boldsymbol{\phi}_t|\boldsymbol{X}_t) \in [0,1]^H$ of the different features being present at each time step $t = 1, \dots, T$. Let us start with the Bayesian update formula

$$P(\phi_{t,i}|\boldsymbol{X}_t) = \frac{p(\boldsymbol{x}_t|\phi_{t,i})P(\phi_{t,i}|\boldsymbol{X}_{t-1})}{\sum_{\phi'_{t,i}} p(\boldsymbol{x}_t|\phi'_{t,i})P(\phi'_{t,i}|\boldsymbol{X}_{t-1})}\,, \tag{3.1}$$

that we can rewrite as,

$$h_{t,i} = \frac{1}{1 + \dfrac{p(\boldsymbol{x}_t|\neg\phi_{t,i})}{p(\boldsymbol{x}_t|\phi_{t,i})} \cdot \dfrac{P(\neg\phi_{t,i}|\boldsymbol{X}_{t-1})}{P(\phi_{t,i}|\boldsymbol{X}_{t-1})}} \tag{3.2}$$

for the two-class case, where the posterior representing the desired unit output $h_{t,i}$ is expressed as a function of the ratio of likelihood $r_{t,i}$ and prior $p_{t,i}$, that we define in

vectorised form for the whole layer as

$$\boldsymbol{r}_t := \frac{p(\boldsymbol{x}_t|\boldsymbol{\phi}_t)}{p(\boldsymbol{x}_t|\neg\boldsymbol{\phi}_t)} \quad \text{and} \quad \boldsymbol{p}_t := P(\boldsymbol{\phi}_t|\boldsymbol{X}_{t-1})\,. \tag{3.3}$$

As pointed out by Bridle (1990a) and more recently reiterated by Garner and Tong (2021), Bayes's theorem has an explicit relationship with the sigmoid function $\sigma(x) = 1/(1+e^{-x})$, so that equation (3.2) can be rewritten as,

$$\boldsymbol{h}_t = \sigma\Big[\,\log(\boldsymbol{r}_t) + \text{logit}(\boldsymbol{p}_t)\Big]\,, \tag{3.4}$$

where $\text{logit}(x) = \log\Big[x/(1-x)\Big]$. As demonstrated by Garner and Tong (2021), if we assume that the likelihood of observing $\boldsymbol{x}_t$ given the current state of the features $\boldsymbol{\phi}_t$ can be represented with multivariate normal distributions that share the same covariance matrix $\boldsymbol{\Sigma}$, i.e. $p(\boldsymbol{x}_t|\boldsymbol{\phi}_t) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $p(\boldsymbol{x}_t|\,\neg\boldsymbol{\phi}_t\,) \sim \mathcal{N}(\boldsymbol{\nu}, \boldsymbol{\Sigma})$, then the ratio of likelihood can be expressed as

$$\boldsymbol{r}_t = \exp\Big[\boldsymbol{W}_r^T\,\boldsymbol{x}_t + \boldsymbol{b}_r\Big]\,. \tag{3.5}$$

where $\boldsymbol{W}_r \in \mathbb{R}^{F\times H}$ and $\boldsymbol{b}_r \in \mathbb{R}^H$ are defined as,

$$\boldsymbol{W}_r = \Big(\boldsymbol{\nu}^T - \boldsymbol{\mu}^T\Big)\,\boldsymbol{\Sigma}^{-1} \tag{3.6a}$$

$$\boldsymbol{b}_r = -\frac{1}{2}\Big(\boldsymbol{\nu}^T\,\boldsymbol{\Sigma}^{-1}\,\boldsymbol{\nu} + \boldsymbol{\mu}^T\,\boldsymbol{\Sigma}^{-1}\,\boldsymbol{\mu}\Big)\,, \tag{3.6b}$$

In the next subsection, we will derive a novel way of estimating the prior $\boldsymbol{p}_t$, leading to a layer-wise recurrence without any approximation.

Prior

In the simplest case, where the features are time-independent, i.e. a feature is either present in the entirety of the sequence or not there at all, the prior probability is simply given by the one from the previous time step $P(\phi_{t,i}|\boldsymbol{X}_{t-1}) = P(\phi_{t-1,i}|\boldsymbol{X}_{t-1}) = h_{t-1,i}$.

Now let us assume that the features can occur and vanish arbitrarily throughout the sequence. We can first assume they are independent, which results in a unit-wise feedback, where the prior probability $P(\phi_{t,i}|\boldsymbol{X}_{t-1})$ only depends on $P(\phi_{t-1,i}|\boldsymbol{X}_{t-1}) = h_{t-1,i}$.

In the more realistic scenario, where we further assume that there is an interdependence between the different features, i.e. that some will more naturally occur together than others, the prior now needs to depend on all $P(\phi_{t-1,j}|\boldsymbol{X}_{t-1})$ with $j = 1,\ldots,H$. Here we need to combine all H feature probabilities $h_{t-1,j}$ of the layer into a single prior

probability. Dropping the $t-1$ in the index for simplicity, the h_i are between 0 and 1; it is reasonable to assume that they are each independently beta distributed:

$$\begin{aligned} p(h_i|\alpha_i,\beta_i) &= \frac{1}{B(\alpha_i,\beta_i)} h_i^{\alpha_i-1}(1-h_i)^{\beta_i-1} \\ &= \exp\Big(-\log B + (\alpha_i-1)\log(h_i) + (\beta_i-1)\log(1-h_i)\Big)\,. \end{aligned} \tag{3.7}$$

The joint distribution is then,

$$p(\boldsymbol{h}|\boldsymbol{\alpha}) = \prod_{i=1}^{H} p(h_i|\alpha_i,\beta_i)\,, \tag{3.8}$$

where $\boldsymbol{\alpha} = \Big[\alpha_1,\ldots,\alpha_H,\beta_1,\ldots,\beta_H\Big]^T$.

We define one set of parameters $\boldsymbol{\alpha}_1$ that represents the beta-distribution when the features are present at the next time step: $p(\boldsymbol{h}_{t-1}|\boldsymbol{\alpha}_1) = p(\boldsymbol{h}_{t-1}|\boldsymbol{\phi}_t)$, and a second one, $\boldsymbol{\alpha}_2$ that corresponds to the distribution when they are absent: $p(\boldsymbol{h}_{t-1}|\boldsymbol{\alpha}_2) = p(\boldsymbol{h}_{t-1}|\neg\boldsymbol{\phi}_t)$. We then write

$$\begin{aligned} P(\boldsymbol{\phi}_t|\boldsymbol{h}_{t-1}) &= \frac{p(\boldsymbol{h}_{t-1}|\boldsymbol{\phi}_t)P(\boldsymbol{\phi}_t)}{P(\boldsymbol{h}_{t-1})} \\ &= \frac{1}{1+\dfrac{p(\boldsymbol{h}_{t-1}|\neg\boldsymbol{\phi}_t)}{p(\boldsymbol{h}_{t-1}|\boldsymbol{\phi}_t)}\,\dfrac{1-P(\boldsymbol{\phi}_t)}{P(\boldsymbol{\phi}_t)}} \end{aligned} \tag{3.9}$$

Using equations (3.7) and (3.8), the ratio of likelihood in the denominator of equation (3.9) can be computed as,

$$\frac{p(\boldsymbol{h}_{t-1}|\boldsymbol{\alpha}_2)}{p(\boldsymbol{h}_{t-1}|\boldsymbol{\alpha}_1)} = \exp\Big[-\boldsymbol{V}_p \log(\boldsymbol{h}_{t-1}) - \boldsymbol{b}_p\Big] \tag{3.10}$$

where $\boldsymbol{V}_p := \boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2$ and $\boldsymbol{b}_p := \log B(\boldsymbol{\alpha}_2) - \log B(\boldsymbol{\alpha}_1)$. Notice that we have ignored the $\log(1-h_i)$ terms. To do this, we need to make the assumption that all the β_i are equal to 1, or that they are the same for each class (feature presence). The second of these is similar to the identical covariance in the Gaussian assumption case for the ratio of likelihood.

The prior $P(\boldsymbol{\phi}_t)$ in equation (3.9) represents the probability of having the feature present at time t even before seeing the previous observation $\boldsymbol{x}_{t-1}$ or knowing $\boldsymbol{h}_{t-1}$. This can be assumed to be some unconditional prior $P(\boldsymbol{\phi}_0)$. By putting equation (3.9) in sigmoid form like we did with equation (3.4), the constant prior term $\text{logit}\Big[P(\boldsymbol{\phi}_0)\Big]$ can be integrated

inside $\boldsymbol{b}_p$, and we get this final expression for the layer-wise prior

$$\boldsymbol{p}_t = \sigma\Big[\boldsymbol{V}_p \log(\boldsymbol{h}_{t-1}) + \boldsymbol{b}_p\Big], \tag{3.11}$$

where we assumed that $P(\boldsymbol{\phi}_t|\boldsymbol{X}_{t-1}) = P(\boldsymbol{\phi}_t|\boldsymbol{h}_{t-1})$.

Resulting Bayesian recurrent unit

With equations (3.5) and (3.11), we now have a probabilistically plausible way of computing the ratio of likelihood $\boldsymbol{r}_t$ and prior $\boldsymbol{p}_t$. By plugging them into equation (3.4), we can use the relationship between the sigmoid and logit functions,

$$\text{logit}\Big[\sigma(x)\Big] = x \tag{3.12}$$

to get the following update equation,

$$\boldsymbol{h}_t = \sigma\Big[\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{V}_h \log(\boldsymbol{h}_{t-1}) + \boldsymbol{b}_h\Big], \tag{3.13}$$

where we redefined the parameters as $\boldsymbol{W}_h = \boldsymbol{W}_r$, $\boldsymbol{V}_h = \boldsymbol{V}_p$ and $\boldsymbol{b}_h = \boldsymbol{b}_r + \boldsymbol{b}_p$. These parameters are representative of the distributions of $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$ when the features are present or absent, and can be treated as trainable parameters of the model.

If we instead choose our units to output log-probabilities, i.e. $\boldsymbol{h}_t := \log\Big[P(\boldsymbol{\phi}_t|\boldsymbol{X}_t)\Big]$, we can then write,

$$\boldsymbol{h}_t = \log\sigma\Big[\boldsymbol{W}_h\, \boldsymbol{x}_t + \boldsymbol{V}_h\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_h\Big]. \tag{3.14}$$

This actually corresponds to a softplus activation function, $\text{softplus}(x) = -\log\Big[\sigma(-x)\Big]$, as described by Dugas et al. (2001), where the sign differences can be integrated inside the trainable parameters and we write the resulting forward pass as,

$$\boldsymbol{h}_t = \text{softplus}\Big[\boldsymbol{W}_h\, \boldsymbol{x}_t + \boldsymbol{V}_h\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_h\Big]. \tag{3.15}$$

3.2.2 Comparison with RNN equation

The equation (3.13) resulting from this Bayesian approach resembles the following forward pass of a standard RNN unit,

$$\boldsymbol{h}_t = \tanh\Big[\boldsymbol{W}_h\, \boldsymbol{x}_t + \boldsymbol{V}_h\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_h\Big], \tag{3.16}$$

which is also the basis of LSTMs and GRUs. In these standard recurrent units, the feedback is not taken on the logarithm of probabilities but on $\boldsymbol{h}_{t-1}$ directly. The activation function is a hyperbolic tangent, which is a rescaled version of a sigmoid with $\tanh(x) = 2\sigma(2x)-1$. If $\sigma(x)$ describes a probability in [0,1], then $\tanh(x)$ is simply a rescaled representation of that probability in the [-1,1] range. The main difference is therefore on the absence of the log in the feedback, but since here $\boldsymbol{h}_{t-1} \in [-1, 1]^H$ it does not make sense to take its logarithm, as $\log(x \leq 0)$ is not defined.

Coming back to equation (3.15) with the softplus activation, we notice that there is no log in the feedback as $\boldsymbol{h}_t$ already describes log-probabilities. As shown by Glorot, Bordes, and Bengio (2011), the ReLU function is a linear approximation of the softplus. Using the ReLU activation turns out to be exactly the forward pass of a Li-GRU (Ravanelli, Bordes, and Bengio, 2018) without the update gate,

$$\boldsymbol{h}_t \approx \text{ReLU}\Big[\boldsymbol{W}_h\, \boldsymbol{x}_t + \boldsymbol{V}_h\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_h\Big]\,. \tag{3.17}$$

In the following section, we derive an alternative to the Li-GRU by adding an update gate through a Bayesian approach.

3.2.3 Defining the Li-BRU

Let us start with the Bayesian Recurrent Unit (BRU) described by equation (3.13). In a slight simplification of the approach by Garner and Tong (2021), let us define a binary state variable $\rho_{t,i}$ that is indicative of the relevance of the current observation $\boldsymbol{x}_t$ for the occurrence of the i-th hidden feature. The associated probabilities $z_{t,i} = P(\rho_{t,i}|\boldsymbol{X}_t)$ can be computed as a layer of BRUs with equation (3.13),

$$\boldsymbol{z}_t = \sigma\Big[\boldsymbol{W}_z\, \boldsymbol{x}_t + \boldsymbol{V}_z\, \log(\boldsymbol{h}_{t-1}) + \boldsymbol{b}_z\Big]\,. \tag{3.18}$$

We chose the recurrence to be on $\boldsymbol{h}_{t-1}$ instead of $\boldsymbol{z}_{t-1}$, simply because we observed better performance. Both are valid choices as they represent probabilities and can be considered to be beta-distributed (see subsection 3.2.1).

The idea is to apply $\boldsymbol{z}_t$ as a gate on probabilities, which is why we choose equation (3.13) and not (3.15), that describes log-probabilities instead. The desired output probability $h_{t,i}$ can then be expressed by marginalizing as,

$$\begin{aligned} h_{t,i} &= P(\phi_{t,i}|\boldsymbol{X}_t) \\ &= \textstyle\sum_{\rho'_{t,i}} P(\phi_{t,i}|\boldsymbol{X}_t, \rho'_{t,i})\, p(\rho'_{t,i}|\boldsymbol{X}_t) \\ &= (1 - z_{t,i}) P(\phi_{t,i}|\boldsymbol{X}_{t-1}) + z_{t,i}\, P(\phi_{t,i}|\boldsymbol{X}_t) \\ &= (1 - z_{t,i})\, h_{t-1,i} + z_{t,i}\, h_{t,i}\,, \end{aligned} \tag{3.19}$$

which represents taking an arithmetic weighted mean of two probabilities p_1, p_2 as $p = z_t \cdot p_1 + (1 - z_t) \cdot p_2$. When context is not relevant, we write $P(\phi_{t,i}|\boldsymbol{X}_t, \neg\rho_{t,i}) = P(\phi_{t,i}|\boldsymbol{X}_{t-1})$.

In a Li-GRU, due to the ReLU activation, the update gate acts on log-probabilities and thus corresponds to taking a geometric weighted mean of two probabilities: $p = p_1^{z_t} \cdot p_2^{1-z_t}$, since $\log(p) = z_t \cdot \log(p_1) + (1 - z_t) \cdot \log(p_2)$. The proper approach would be to first exponentiate the log-probabilities before applying the gate. In practice, we found no significant difference in doing so, suggesting that taking the geometric mean of the probabilities is an appropriate approximation.

We can now define the Bayesian equivalent of a Li-GRU, that we call Li-BRU, where the z-gate acts on probabilities,

$$\boldsymbol{z}_t = \sigma\Big[\boldsymbol{W}_z\,\boldsymbol{x}_t + \boldsymbol{V}_z\,\log(\boldsymbol{h}_{t-1}) + \boldsymbol{b}_z\Big] \tag{3.20a}$$

$$\tilde{\boldsymbol{h}}_t = \sigma\Big[\boldsymbol{W}_h\,\boldsymbol{x}_t + \boldsymbol{V}_h\,\log(\boldsymbol{h}_{t-1}) + \boldsymbol{b}_h\Big] \tag{3.20b}$$

$$\boldsymbol{h}_t = \boldsymbol{z}_t * \tilde{\boldsymbol{h}}_t + (1 - \boldsymbol{z}_t) * \boldsymbol{h}_{t-1}\,. \tag{3.20c}$$

Additionally, the input of the i-th layer corresponds to the log-probabilities of the previous layer $\boldsymbol{x}_t^{[i]} = \log\left(\boldsymbol{h}_t^{[i-1]}\right)$.

An alternative way of defining the same unit is with

$$\boldsymbol{z}_t = \sigma\Big[\boldsymbol{W}_z\,\boldsymbol{x}_t + \boldsymbol{V}_z\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_z\Big] \tag{3.21a}$$

$$\tilde{\boldsymbol{h}}_t = \text{softplus}\Big[\boldsymbol{W}_h\,\boldsymbol{x}_t + \boldsymbol{V}_h\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_h\Big] \tag{3.21b}$$

$$\boldsymbol{h}_t = \log\left(\boldsymbol{z}_t * e^{\tilde{\boldsymbol{h}}_t} + (1 - \boldsymbol{z}_t) * e^{\boldsymbol{h}_{t-1}}\right), \tag{3.21c}$$

where $\boldsymbol{x}_t^{[i]} = \boldsymbol{h}_t^{[i-1]}$ as $\boldsymbol{h}_t$ already describes log-probabilities. The first version of the Li-BRU as defined by equations (3.20) actually gave marginally better results than the second one from equations (3.21) and will be the one used in the experiments described in the next section.

3.2.4 Experiments

Following the pytorch-kaldi implementation of Ravanelli, Bordes, and Bengio (2018) and Ravanelli, Parcollet, and Bengio (2019), all presented experiments use a recurrent architecture with four layers of H=550 bidirectional units. The F=50 fMLLR input features are extracted via the Kaldi (Povey et al., 2011) recipe. Experiments on the TIMIT (Garofolo et al., 1993) and AMI (Carletta et al., 2005) corpora are performed with Adam (Kingma

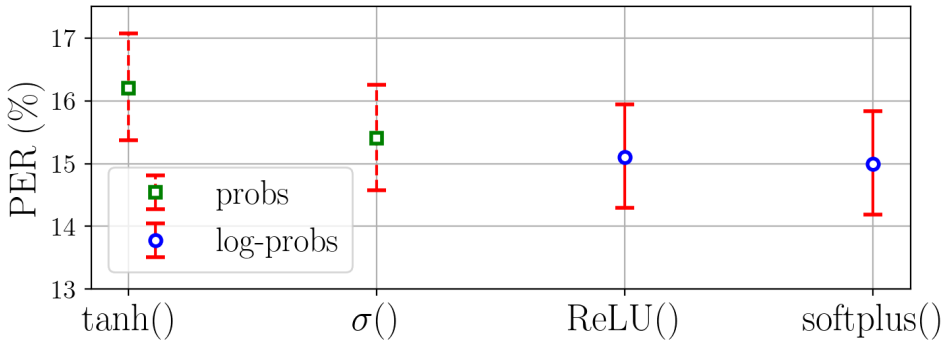

Figure 3.1: PERs on TIMIT test-set for various RNN architectures.

and Ba, 2015) and RMSprop (Tieleman and Hinton, 2012) optimisers respectively, both during 24 epochs with drop-out regularisation (p=0.2). Batch-normalisation (Ioffe and Szegedy, 2015) is used on feed-forward connections, as suggested by Ravanelli, Bordes, and Bengio (2018). Our aim is not to surpass the best reported Phoneme Error Rate (PER) on TIMIT or AMI, but to perform a self-consistent comparison between recurrent units.

Without the update gate

We first test the simple BRU resulting from section 3.2.1 on the TIMIT dataset and compare it to standard non-gated RNN units (see Figure 3.1). We make a distinction between the units that have a feedback on (rescaled) probabilities, like the standard RNN of equation (3.16), and the ones that use log-probabilities (as justified in subsection 3.2.1). Note that all units have the same number of trainable parameters (i.e. F+H+1 per unit). We make the hypothesis that the units with a feedback on log-probabilities perform better than the ones with probabilities. The error-bars on Figure 3.1 show the 95% equal-tailed confidence interval for a beta-assumption for the error-rate. As they are relatively large on TIMIT due to the small size of the test set (7,215 utterances versus 90,002 for AMI), we perform a matched-pairs test, as described by Gillick and Cox (1989), on the different speakers and obtain a p-value of $4.96 \cdot 10^{-5}$. Alternatively using a Wilcoxon signed-rank test (Wilcoxon, 1945) gives us a p-value of $5.09 \cdot 10^{-4}$. We consider both results to be small enough to validate our hypothesis.

Testing the complete Li-BRU

Let us now add the update gate and compare the resulting Li-BRU from equations (3.20) to state of the art recurrent units. As illustrated in Figure 3.2, the Li-BRU outperforms all other state of the art recurrent architectures on both TIMIT and AMI datasets with error-rates of 14.4 %, and 26.4% respectively. We also tested the Li-GRU architecture with a softplus activation instead of a ReLU. As mentioned at the end of section 3.2.3,

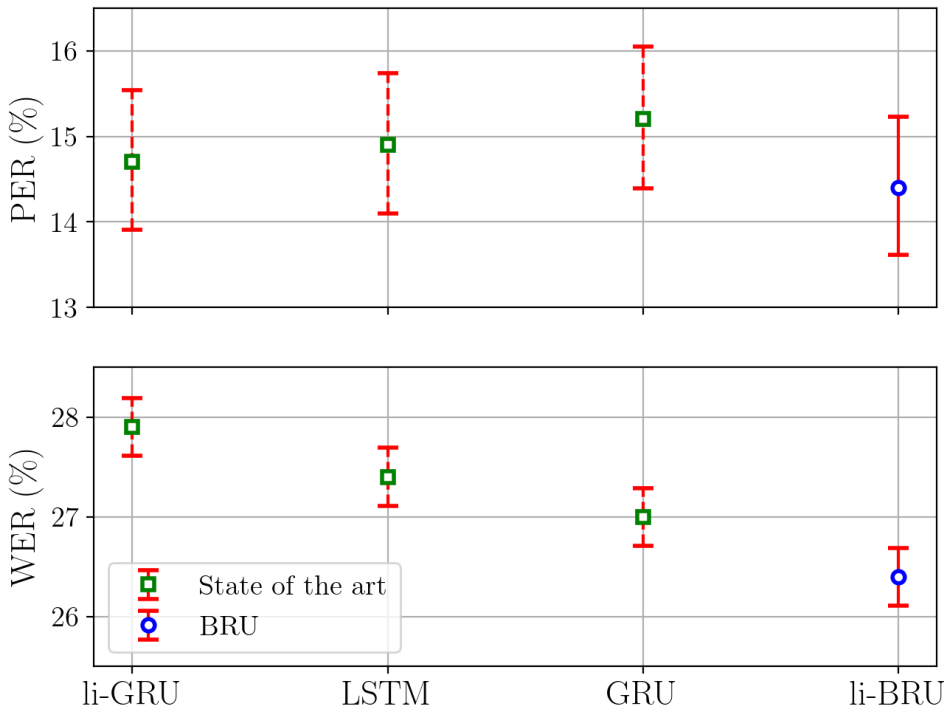

Figure 3.2: PERs on TIMIT (top) and WERs on AMI (bottom) for various RNN architectures.

the unit gave the same results as the Li-BRU, both on TIMIT and AMI, signifying that applying the gate to probabilities or log-probabilities is practically equivalent. The root of the improvement therefore seems to lie in the choice of the activation function. The importance of using the softplus function instead of its approximation by the ReLU is especially visible on AMI.

3.2.5 Summary of contributions

In previous work (Garner and Tong, 2021), it was shown that a Bayesian analysis of a sigmoid activation function led naturally to a unit-wise recurrence and, with approximations, to a layer-wise recurrence. In this section, in a mainly theoretical contribution, we have shown that beta-distributed sigmoid outputs feeding into another sigmoid unit constitute a layer-wise recurrence without approximations. Without a forget gate, this reduces to a standard fully-connected recurrence, but with a softplus activation. Given that the update gate of a GRU can also be derived probabilistically, the approach led naturally to comparison with a Li-GRU. In an experimental evaluation, we confirmed that the resulting Li-BRU can modestly but significantly outperform the state of the art on two ASR tasks (TIMIT and AMI datasets), demonstrating the importance of the probabilistic derivation. More generally, the new techniques contribute to a growing toolkit of Bayesian approaches for neural architectures.

3.3 Unit-wise Bayesian Recurrent Unit

In our work on the Li-BRU presented in Section 3.2, hidden features were assumed to be interdependent, which led to a layer-wise recurrence for the computation of prior probabilities. In this section, in a mainly theoretical contribution, we come back to the simpler case of an RNN with unit-wise recurrence and no gate. Here, by assuming a latent space of independent features, a Bayesian analysis demonstrates that the trainable parameters of the network directly correspond to standard parameters of a first-order two-state HMM.

In a second step, similarly to the Kalman smoother (Kalman, 1960) and to the forward-backward algorithm (Baum et al., 1972), we derive two different backward recursions that allow the consideration of future observations without relying on any additional parameters. We also prove by induction that the two are equivalent. In contrast with the approach of Garner and Tong (2021), the unit-wise recurrence is here derived using transition probabilities instead of a context relevance gate.

The derived Unit-wise Bayesian Recurrent Units (UBRUs) can be trained like standard RNNs inside modern Deep Neural Networks (DNNs). Even though UBRUs have much less representational power than state-of-the-art RNNs, they are appropriate when the features are decorrelated. We confirm this by showing that, when placed on the phoneme (rather than acoustic) region of a DNN for ASR, they are able to replace larger standard bidirectional gated RNNs without any loss of performance, highlighting their practical relevance and efficiency in real-world applications.

3.3.1 A hidden Markov model approach

Consider an input sequence $\boldsymbol{X}_T = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_T] \in \mathbb{R}^{F \times T}$ of length T, where each observation $\boldsymbol{x}_t$ is a vector with F input dimensions. We assume that there are H hidden features $\{\phi_i \,|i = 1, \dots, H\}$ that we wish to detect along the sequence. At each time step t, a feature has two possible states: present or absent, that we write as $\phi_{t,i}$ and $\neg\phi_{t,i}$ respectively. Each hidden feature can be represented as a first-order two-state Markov process, such that its probability of occurring at time step t only depends on its state at the previous time step $t-1$. For a single hidden feature ϕ, an initial state probability $a \in [0,1]^{2\times 1}$,

$$a = \begin{bmatrix} P(\phi_0), P(\neg\phi_0) \end{bmatrix}, \tag{3.22}$$

and a transition matrix $A \in [0,1]^{2\times 2}$,

$$A = \begin{bmatrix} P(\phi_t|\phi_{t-1}) & P(\neg\phi_t|\phi_{t-1}) \\ P(\phi_t|\neg\phi_{t-1}) & P(\neg\phi_t|\neg\phi_{t-1}) \end{bmatrix}, \tag{3.23}$$

can be defined to describe the evolution of the state through discrete time. Then, for any binary sequence of hidden states, the probability of the sequence being generated by the Markov chain is fully defined in terms of a and A as a product of initial and transition probabilities. Since the hidden sequence is not directly observable, let us additionally define a set of distributions,

$$B(\boldsymbol{x}_t) = \Big[b_1(\boldsymbol{x}_t), b_2(\boldsymbol{x}_t)\Big] = \Big[p(\boldsymbol{x}_t|\phi_t),\, p(\boldsymbol{x}_t|\neg\phi_t)\Big]\,, \tag{3.24}$$

representing the likelihood of seeing observation $\boldsymbol{x}_t$ at time step t given the two possible feature states. As explained by Juang and Rabiner (1991), the stochastic process represented by $\boldsymbol{X}_T$ can then be fully characterised by the HMM parameters a, A and $B(\boldsymbol{x}_t)$, without requiring the knowledge of the sequence of hidden states.

3.3.2 Neural network formulation

Let us start by using a more machine learning oriented formulation of the HMM parameters a and A. We define trainable scalars $\rho_{0,i}$, $\tau_{11,i}$ and $\tau_{01,i} \in [0, 1]$ that describe the initial and transition probabilities of the i-th hidden feature,

$$a_i = \Big[\rho_{0,i}, 1-\rho_{0,i}\Big] \quad \text{and} \quad A_i = \begin{bmatrix} \tau_{11,i} & 1-\tau_{11,i} \\ \tau_{01,i} & 1-\tau_{01,i} \end{bmatrix}\,, \tag{3.25}$$

where we used the notation $\tau_{kl} = P(\phi_t = l|\phi_{t-1} = k)$, $k, l \in \{0, 1\}$. These can then be vectorised for the whole layer as $\boldsymbol{\rho}_0$, $\boldsymbol{\tau}_{11}$ and $\boldsymbol{\tau}_{01} \in [0, 1]^H$. In order to express the remaining HMM parameters related to the set of distributions $\boldsymbol{B}(\boldsymbol{x}_t)$, we can assume that the likelihood of observing $\boldsymbol{x}_t$ given the current state of the hidden features $\boldsymbol{\phi}_t$, can be described using a distribution from the exponential family. As we will see in Section 3.3.3, only the ratio of these distributions will be necessary to compute. As demonstrated by Garner and Tong (2021) drawing from Bridle (1990b), this ratio of likelihood $\boldsymbol{r}_t$ can then be expressed as

$$\boldsymbol{r}_t := \frac{p(\boldsymbol{x}_t|\,\neg\boldsymbol{\phi}_t\,)}{p(\boldsymbol{x}_t|\,\boldsymbol{\phi}_t\,)} = \exp\Big[-\boldsymbol{W}^T\boldsymbol{x}_t - \boldsymbol{b}\Big]\,. \tag{3.26}$$

As we saw in Section 3.2.1, for the case of multivariate normal distributions that share the same covariance matrix $\boldsymbol{\Sigma}$, i.e., $p(\boldsymbol{x}_t|\boldsymbol{\phi}_t) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $p(\boldsymbol{x}_t|\,\neg\boldsymbol{\phi}_t\,) \sim \mathcal{N}(\boldsymbol{\nu}, \boldsymbol{\Sigma})$, the parameters $\boldsymbol{W} \in \mathbb{R}^{F\times H}$ and $\boldsymbol{b} \in \mathbb{R}^H$ can be expressed as,

$$\boldsymbol{W} = \Big(\boldsymbol{\nu}^T - \boldsymbol{\mu}^T\Big)\,\boldsymbol{\Sigma}^{-1} \tag{3.27a}$$

$$\boldsymbol{b} = -\frac{1}{2}\Big(\boldsymbol{\nu}^T\,\boldsymbol{\Sigma}^{-1}\,\boldsymbol{\nu} + \boldsymbol{\mu}^T\,\boldsymbol{\Sigma}^{-1}\,\boldsymbol{\mu}\Big)\,. \tag{3.27b}$$

Overall, we have shown that the Markov processes corresponding to a layer of H independent hidden features can be fully described by a set of trainable tensors (or parameters) $\boldsymbol{\rho}_0$, $\boldsymbol{\tau}_{11}$, $\boldsymbol{\tau}_{01} \in [0,1]^H$, $\boldsymbol{W} \in \mathbb{R}^{F\times H}$ and $\boldsymbol{b} \in \mathbb{R}^H$. In the next section, we will derive a forward-backward formulation that is similar to that of RNNs. This will allow them to be trained inside a deep learning framework, while retaining a probabilistic interpretation as they correspond to standard HMM parameters.

3.3.3 Forward-backward procedure

In order to make inference about the state of the hidden features throughout the sequence, we use a Bayesian approach and design a layer of recurrent units that will evaluate the stacked conditional probabilities $\boldsymbol{\gamma}_t := P(\boldsymbol{\phi}_t|\boldsymbol{X}_T) \in [0,1]^H$ of the different features being present at each time step $t = 1,\dots,T$, given the information of the complete input sequence $\boldsymbol{X}_T$. In the first *alpha* or forward part of the procedure, the probabilities $\boldsymbol{\alpha}_t := P(\boldsymbol{\phi}_t|\boldsymbol{X}_t) \in [0,1]^H$ are computed. In the subsequent *beta* or backward part, these probabilities are smoothed by taking into account future observations and produce the desired outputs $\boldsymbol{\gamma} \in [0,1]^{T\times H}$, that are fed into the next layer.

Derivation of the forward pass

The quantity $\boldsymbol{\alpha}_t$ is defined as $\boldsymbol{\alpha}_t := P(\boldsymbol{\phi}_t|\boldsymbol{X}_t) \in [0,1]^H$. Using Bayes's formula, we can write it as,

$$\boldsymbol{\alpha}_t = \frac{p(\boldsymbol{x}_t|\boldsymbol{\phi}_t)\, P(\boldsymbol{\phi}_t|\boldsymbol{X}_{t-1})}{\sum_{\boldsymbol{\phi}_t'} p(\boldsymbol{x}_t|\boldsymbol{\phi}_t')\, P(\boldsymbol{\phi}_t'|\boldsymbol{X}_{t-1})}\,. \tag{3.28}$$

Dividing both numerator and denominator by $p(\boldsymbol{x}_t|\boldsymbol{\phi}_t)$ gives

$$\boldsymbol{\alpha}_t = \frac{\boldsymbol{p}_t}{\boldsymbol{p}_t + \boldsymbol{r}_t\,(1-\boldsymbol{p}_t)}\,, \tag{3.29}$$

where $\boldsymbol{r}_t$, $\boldsymbol{p}_t$ and $\boldsymbol{\alpha}_t$ correspond to the ratio of likelihood, prior and posterior probabilities of the Bayesian update respectively. One can also reformulate Eq. (3.29) by dividing the numerator and denominator by the prior. This gives rise to the well known sigmoid activation function $\sigma(x) = 1/(1+e^{-x})$,

$$\boldsymbol{\alpha}_t = \sigma\Big[\boldsymbol{W}^T\boldsymbol{x}_t + \boldsymbol{b} + \text{logit}(\boldsymbol{p}_t)\Big]\,, \tag{3.30}$$

where the logit function, $\text{logit}(x) = \log\Big[x/(1-x)\Big]$, is the inverse of the sigmoid. The prior $\boldsymbol{p}_t := P(\boldsymbol{\phi}_t|\boldsymbol{X}_{t-1})$ represents the probability of having the features present at time t before seeing the current observation $\boldsymbol{x}_t$. For a time independent prior $\boldsymbol{p}_t = \text{const.}$, the quantity $\text{logit}(\boldsymbol{p}_t)$ is also constant and can be integrated into the trainable bias $\boldsymbol{b}$, so that the forward pass corresponds to a hidden layer of a standard feed-forward neural

network. With this probabilistic interpretation, it is therefore the time dependence of the Bayesian prior that leads to recurrence in neural networks. By assuming independent hidden features, the prior can be expanded as a function of the transition probabilities,

$$\begin{aligned} \boldsymbol{p}_t &:= P(\boldsymbol{\phi}_t|\boldsymbol{X}_{t-1}) \\ &= P(\boldsymbol{\phi}_t|\boldsymbol{\phi}_{t-1})P(\boldsymbol{\phi}_{t-1}|\boldsymbol{X}_{t-1}) + P(\boldsymbol{\phi}_t|\neg\boldsymbol{\phi}_{t-1})P(\neg\boldsymbol{\phi}_{t-1}|\boldsymbol{X}_{t-1}) \\ &= \boldsymbol{\tau}_{11}\,\boldsymbol{\alpha}_{t-1} + \boldsymbol{\tau}_{01}\,(1-\boldsymbol{\alpha}_{t-1})\,. \end{aligned} \tag{3.31}$$

Using Bayes's theorem, we have thus derived a forward pass from $t = 1$ to $t = T$ through the sequence, that allows the computation of $\boldsymbol{\alpha}_t = P(\boldsymbol{\phi}_t|\boldsymbol{X}_t)$ via a unit-wise first-order recurrence on $\boldsymbol{\alpha}_{t-1}$. So far, the inference on the state of the hidden features at time step t only takes the previous observations $\boldsymbol{X}_t = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t]$ into account. In order to include the future observations $\boldsymbol{X}_{>t} = [\boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_T]$, we define a backward recursion that will smooth out the probabilities.

Derivation of HMM backward recursion

Using the relationship between joint and conditional probabilities as well as the independence of observations, we can express the desired quantity $\boldsymbol{\gamma}_t$ as,

$$\begin{aligned} P(\boldsymbol{\phi}_t|\boldsymbol{X}_T) = \frac{P(\boldsymbol{\phi}_t, \boldsymbol{X}_T)}{P(\boldsymbol{X}_T)} &= \frac{P(\boldsymbol{X}_t, \boldsymbol{\phi}_t)\,P(\boldsymbol{X}_{>t}|\boldsymbol{X}_t, \boldsymbol{\phi}_t)}{P(\boldsymbol{X}_{>t}|\boldsymbol{X}_t)\,P(\boldsymbol{X}_t)} \\ &= P(\boldsymbol{\phi}_t|\boldsymbol{X}_t)\,\frac{P(\boldsymbol{X}_{>t}|\boldsymbol{\phi}_t)}{P(\boldsymbol{X}_{>t}|\boldsymbol{X}_t)} = \boldsymbol{\alpha}_t\;\boldsymbol{\beta}_t\,, \end{aligned} \tag{3.32}$$

where we use the notation $\boldsymbol{X}_{>t} := \boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_T$ and define $\boldsymbol{\beta}_t$ and its counterpart $\overline{\boldsymbol{\beta}_t}$ as,

$$\boldsymbol{\beta}_t := \frac{P(\boldsymbol{X}_{>t}|\boldsymbol{\phi}_t)}{P(\boldsymbol{X}_{>t}|\boldsymbol{X}_t)} \quad \text{and} \quad \overline{\boldsymbol{\beta}_t} := \frac{P(\boldsymbol{X}_{>t}|\neg\boldsymbol{\phi}_t)}{P(\boldsymbol{X}_{>t}|\boldsymbol{X}_t)}\,. \tag{3.33}$$

Let us start by expanding the numerator of $\boldsymbol{\beta}_t$ and use the independence of observations,

$$\begin{aligned} &P(\boldsymbol{X}_{>t}|\boldsymbol{\phi}_{t+1})\,P(\boldsymbol{\phi}_{t+1}|\boldsymbol{\phi}_t) + P(\boldsymbol{X}_{>t}|\neg\boldsymbol{\phi}_{t+1})\,P(\neg\boldsymbol{\phi}_{t+1}|\boldsymbol{\phi}_t) \\ &= P(\boldsymbol{x}_{t+1}|\boldsymbol{\phi}_{t+1})\,P(\boldsymbol{X}_{>t+1}|\boldsymbol{\phi}_{t+1})\,P(\boldsymbol{\phi}_{t+1}|\boldsymbol{\phi}_t) \\ &\quad + P(\boldsymbol{x}_{t+1}|\neg\boldsymbol{\phi}_{t+1})\,P(\boldsymbol{X}_{>t+1}|\neg\boldsymbol{\phi}_{t+1})\,P(\neg\boldsymbol{\phi}_{t+1}|\boldsymbol{\phi}_t)\,. \end{aligned} \tag{3.34}$$

The denominator of Eq. (3.33) can similarly be decomposed as,

$$P(\boldsymbol{X}_{>t}|\boldsymbol{X}_t) = P(\boldsymbol{x}_{t+1}|\boldsymbol{X}_t)\,P(\boldsymbol{X}_{>t+1}|\boldsymbol{X}_{t+1})\,, \tag{3.35}$$

so that combining Eqs. (3.34) and (3.35) gives,

$$\boldsymbol{\beta}_t = \frac{\boldsymbol{b}_1(\boldsymbol{x}_{t+1})\,\boldsymbol{\beta}_{t+1}\,\boldsymbol{\tau}_{11} + \boldsymbol{b}_2(\boldsymbol{x}_{t+1})\,\overline{\boldsymbol{\beta}_{t+1}}\,(1-\boldsymbol{\tau}_{11})}{P(\boldsymbol{x}_{t+1}|\boldsymbol{X}_t)}\,. \tag{3.36}$$

We finally need to deal with the remaining denominator of Eq. (3.36),

$$\begin{aligned} P(\boldsymbol{x}_{t+1}|\boldsymbol{X}_t) &= P(\boldsymbol{x}_{t+1}|\boldsymbol{\phi}_{t+1})\,P(\boldsymbol{\phi}_{t+1}|\boldsymbol{X}_t) + P(\boldsymbol{x}_{t+1}|\neg\boldsymbol{\phi}_{t+1})\,P(\neg\boldsymbol{\phi}_{t+1}|\boldsymbol{X}_t) \\ &= \boldsymbol{b}_1(\boldsymbol{x}_{t+1})\boldsymbol{p}_{t+1} + \boldsymbol{b}_2(\boldsymbol{x}_{t+1})(1-\boldsymbol{p}_{t+1})\,. \end{aligned} \tag{3.37}$$

By dividing the numerator and denominator by $\boldsymbol{b}_1(\boldsymbol{x}_{t+1})$, we then get the following final expression for $\boldsymbol{\beta}_t$,

$$\boldsymbol{\beta}_t = \frac{\boldsymbol{\tau}_{11}\,\boldsymbol{\beta}_{t+1} + \boldsymbol{r}_{t+1}(1-\boldsymbol{\tau}_{11})\,\overline{\boldsymbol{\beta}_{t+1}}}{\boldsymbol{p}_{t+1} + \boldsymbol{r}_{t+1}\,(1-\boldsymbol{p}_{t+1})}\,. \tag{3.38}$$

Similarly, one can derive that,

$$\overline{\boldsymbol{\beta}_t} = \frac{\boldsymbol{\tau}_{01}\,\boldsymbol{\beta}_{t+1} + \boldsymbol{r}_{t+1}\,(1-\boldsymbol{\tau}_{01})\,\overline{\boldsymbol{\beta}_{t+1}}}{\boldsymbol{p}_{t+1} + \boldsymbol{r}_{t+1}(1-\boldsymbol{p}_{t+1})}\,. \tag{3.39}$$

Derivation of Kalman backward recursion

Following the approach of a Kalman smoother, a simpler backward pass can be derived by expanding $\boldsymbol{\gamma}_t$ on possible future states,

$$P(\boldsymbol{\phi}_t|\boldsymbol{X}_T) = P(\boldsymbol{\phi}_t|\boldsymbol{\phi}_{t+1})P(\boldsymbol{\phi}_{t+1}|\boldsymbol{X}_T) + P(\boldsymbol{\phi}_t|\neg\boldsymbol{\phi}_{t+1})P(\neg\boldsymbol{\phi}_{t+1}|\boldsymbol{X}_T)\,. \tag{3.40}$$

The transition probabilities need to be flipped using Bayes theorem, which gives

$$\begin{aligned} P(\boldsymbol{\phi}_t|\boldsymbol{\phi}_{t+1}) &= \frac{P(\boldsymbol{\phi}_{t+1}|\boldsymbol{\phi}_t)P(\boldsymbol{\phi}_t|\boldsymbol{X}_t)}{\sum_{\boldsymbol{\phi}_t'} P(\boldsymbol{\phi}_{t+1}|\boldsymbol{\phi}_t')P(\boldsymbol{\phi}_t'|\boldsymbol{X}_t)} \\ &= \frac{\boldsymbol{\tau}_{11}\,\boldsymbol{\alpha}_t}{\boldsymbol{\tau}_{11}\,\boldsymbol{\alpha}_t + \boldsymbol{\tau}_{01}\,(1-\boldsymbol{\alpha}_t)} = \boldsymbol{\tau}_{11}\frac{\boldsymbol{\alpha}_t}{\boldsymbol{p}_{t+1}} \end{aligned} \tag{3.41}$$

for the first one, using the definition of the prior given in Eq. (3.31). Applying the same treatment to the second one then gives the following backward recursion,

$$\boldsymbol{\gamma}_t = \boldsymbol{\alpha}_t\left(\boldsymbol{\tau}_{11}\frac{\boldsymbol{\gamma}_{t+1}}{\boldsymbol{p}_{t+1}} + (1-\boldsymbol{\tau}_{11})\frac{1-\boldsymbol{\gamma}_{t+1}}{1-\boldsymbol{p}_{t+1}}\right)\,. \tag{3.42}$$

Equivalence of the two backward recursions

Let us start with the HMM formulation of $\boldsymbol{\gamma}_t = \boldsymbol{\alpha}_t \, \boldsymbol{\beta}_t$. We can use Eq. (3.29) of the forward pass to rewrite $\boldsymbol{\beta}_t$ as,

$$\begin{aligned} \boldsymbol{\beta}_t &= \frac{\boldsymbol{\alpha}_{t+1}}{\boldsymbol{p}_{t+1}} \left(\boldsymbol{\tau}_{11} \, \boldsymbol{\beta}_{t+1} + (1 - \boldsymbol{\tau}_{11}) \, \boldsymbol{r}_{t+1} \, \overline{\boldsymbol{\beta}_{t+1}} \right) \\ &= \boldsymbol{\tau}_{11} \frac{\boldsymbol{\gamma}_{t+1}}{\boldsymbol{p}_{t+1}} + (1 - \boldsymbol{\tau}_{11}) \frac{(1 - \boldsymbol{\alpha}_{t+1}) \overline{\boldsymbol{\beta}_{t+1}}}{1 - \boldsymbol{p}_{t+1}} \, . \end{aligned} \tag{3.43}$$

By comparing with Eq. (3.42), we see that in order to prove that the HMM and Kalman recursions are equivalent, the following equality,

$$1 - \boldsymbol{\gamma}_{t+1} = (1 - \boldsymbol{\alpha}_{t+1}) \overline{\boldsymbol{\beta}_{t+1}} \, , \tag{3.44}$$

must be satisfied $\forall t \in \{T-1, T-2, \ldots, 0\}$. This can be demonstrated by induction as follows.

Proof. We start by considering the base case $t = T-1$. Here $1 - \boldsymbol{\gamma}_T = (1 - \boldsymbol{\alpha}_T) \overline{\boldsymbol{\beta}_T}$ follows trivially from $\overline{\boldsymbol{\beta}_T} = 1$ and $\boldsymbol{\gamma}_T = \boldsymbol{\alpha}_T$. By assuming that Eq. (3.44) is correct for the case $n = t+1$, we must now prove that it holds for the next case $n = t$. Let us start with the left hand side $1 - \boldsymbol{\gamma}_t$ and use the assumption for $n = t+1$ to express it as,

$$1 - \boldsymbol{\gamma}_t = 1 - \boldsymbol{\alpha}_t \left(\boldsymbol{\tau}_{11} \frac{\boldsymbol{\gamma}_{t+1}}{\boldsymbol{p}_{t+1}} + (1 - \boldsymbol{\tau}_{11}) \frac{1 - \boldsymbol{\gamma}_{t+1}}{1 - \boldsymbol{p}_{t+1}} \right) . \tag{3.45}$$

The transition probabilities $\boldsymbol{\tau}_{11}$ can be expressed as a function of $\boldsymbol{\tau}_{01}$ using Eq. (3.31),

$$\boldsymbol{\tau}_{11} = \frac{\boldsymbol{p}_{t+1} - \boldsymbol{\tau}_{01}(1 - \boldsymbol{\alpha}_t)}{\boldsymbol{\alpha}_t} \, . \tag{3.46}$$

By plugging Eq. (3.46) into (3.45), we get that

$$\begin{aligned} 1 - \boldsymbol{\gamma}_t &= (1 - \boldsymbol{\alpha}_t) \left(\boldsymbol{\tau}_{01} \frac{\boldsymbol{\gamma}_{t+1}}{\boldsymbol{p}_{t+1}} + (1 - \boldsymbol{\tau}_{01}) \frac{1 - \boldsymbol{\gamma}_{t+1}}{1 - \boldsymbol{p}_{t+1}} \right) \\ &= (1 - \boldsymbol{\alpha}_t) \, \overline{\boldsymbol{\beta}_t} \, . \end{aligned} \tag{3.47}$$

□

Implementation of the method

The ratio of the distributions $\boldsymbol{b}_2(\boldsymbol{x}_t)$ and $\boldsymbol{b}_1(\boldsymbol{x}_t)$, can be computed in advance for all time steps and hidden features using Eq. (3.26). At $t = 0$, $\boldsymbol{\alpha}_0$ is initialized with the trainable

unconditional prior probability $\boldsymbol{\rho}_0 = P(\boldsymbol{\phi}_0)$. A forward pass from $t = 1$ to $t = T$ is then performed to compute and store the Bayesian prior $\boldsymbol{p}_t = P(\boldsymbol{\phi}_t|\boldsymbol{X}_{t-1})$ and posterior $\boldsymbol{\alpha}_t = P(\boldsymbol{\phi}_t|\boldsymbol{X}_t)$ using Eq. (3.31) and (3.29) respectively. Since the two backward procedures are equivalent, we use the Kalman recursion, as it is computationally simpler. At $t = T$, $\boldsymbol{\gamma}_T$ is initialized with $\boldsymbol{\alpha}_T$, and from $t = T - 1$ to $t = 1$, $\boldsymbol{\gamma}_t = P(\boldsymbol{\phi}_t|\boldsymbol{X}_T)$ is computed using Eq. (3.42).

Overall a UBRU layer of size N^l requires $N^l\,(N^{l-1} + 4)$ trainable parameters, which include the weight matrix $\boldsymbol{W}$, bias $\boldsymbol{b}$, and initial and transition probability vectors $\boldsymbol{\rho}_0$, $\boldsymbol{\tau}_{11}$ and $\boldsymbol{\tau}_{01}$. This parameter count is comparable to that of an MLP, emphasising the parameter-efficiency of the unit-wise recurrence.

3.3.4 Experiments

Speech recognition experiments are performed on the TIMIT corpus (Garofolo et al., 1993), using the SpeechBrain (Ravanelli, Parcollet, Plantinga, et al., 2021) framework. Mel filterbank features are extracted from the waveforms and fed into two convolutional layers, followed by recurrent layers of H=512 hidden units. After two additional linear layers and a final log-softmax activation, the network outputs log-probabilities of phoneme classes. The training is done using the CTC loss (Graves, Fernández, et al., 2006) and the Adadelta optimiser (Zeiler, 2012) for 50 epochs. Batch-normalisation (Ioffe and Szegedy, 2015) is also used on feed-forward connections, as suggested by Ravanelli, Bordes, and Bengio (2018).

Speech features entering the architecture are highly correlated. This suggests that the layer-wise recurrence of standard RNNs, which assumes interdependent hidden features, is best suited for processing them. Nevertheless, once the speech information has been processed and decorrelated, the classification of phoneme or subword representations does not require to assume the same level of correlation. As one expects a phoneme to stay in a state before transitioning to the next one, HMMs have been widely employed in ASR frameworks to process this form of information. While the general aim of the experiments is to implement the derived UBRUs with the backward recursion and demonstrate that the mathematical predictions can be reflected practically, we also make the following hypotheses,

1. As they assume a latent space of independent hidden features, UBRUs should be best placed after layers of standard gated RNNs that can first decorrelate the highly interdependent speech features.

2. Since future observations can already be taken into account with the analytically derived backward recursion, we expect that this method can compete with the standard bidirectional approach.

We start by evaluating UBRUs on their own. We consider unidirectional and bidirectional units, with or without the backward recursion, which leads to four different models. The results are presented in Table 3.1. As expected, the error-rates are relatively high due to the low representational capacities of the units. Nevertheless, we observe that the derived backward recursion improves the error-rate without requiring more trainable parameters, whereas making the units bidirectional does not.

Table 3.1: PERs on TIMIT with only two layers of UBRUs.

Model type	**Parameters**	**PER**
Unidirectional	3.2M	23.62%
Udir. + backward	3.2M	**22.67**%
Bidirectional	3.7M	24.08%
Bidir. + backward	3.7M	23.27%

We then test UBRUs by placing them after layers of state-of-the-art bidirectional Li-GRUs. We again find that unidirectional UBRUs with the backward recursion perform the best, as shown in Table 3.2, which corroborates our second hypothesis and highlights the importance of our probabilistic derivation.

Table 3.2: PERs on TIMIT with four Li-GRU and one UBRU layers.

Model type	**Parameters**	**PER**
Unidirectional	10.0M	14.36%
Udir. + backward	10.0M	**13.96**%
Bidirectional	10.3M	14.75%
Bidir. + backward	10.3M	14.19%

By comparing with the Li-GRU baseline in Table 3.3, we find that adding a single unidirectional UBRU layer with the backward recursion brings the same improvement as adding another Li-GRU layer, even though the latter contains seven times more trainable parameters. In contrast, placing the UBRUs before the Li-GRUs in initial *ad-hoc* experiments suggested that the units were not effective at the acoustic level. This adheres to our first hypothesis that if features at that level represent phonemes and not acoustics, then the HMM-like derived UBRUs are appropriate for the classification task.

Table 3.3: PERs on TIMIT only with Li-GRU layers.

Model type	**Parameters**	**PER**
Li-GRU 4x512	9.8M	14.83%
Li-GRU 5x512	11.3M	13.99%

For reference, we made the same experiments with cross-entropy loss inside the pytorch-kaldi (Ravanelli, Parcollet, and Bengio, 2019; Povey et al., 2011) framework. Here again,

a layer of unidirectional UBRUs with backward recursion is able to compete with a fifth Li-GRU layer, both scoring an accuracy of 14.4% compared to 14.8% for four Li-GRU layers. Here fMLLR input features are extracted via the Kaldi recipe (Povey et al., 2011), and fed into recurrent layers, followed by a final linear layer. The networks are trained for 24 epochs using the RMSprop (Tieleman and Hinton, 2012) optimiser.

In summary, due to their correspondence with HMMs, the analytically derived unidirectional unit-wise recurrent units with a backward recursion are capable of replacing considerably larger, state-of-the-art, bidirectional, layer-wise units on the phoneme end of an ASR architecture, at an extremely low cost in terms of trainable parameters.

3.3.5 Summary of contributions

Using a probabilistic formulation of neural network components, we have analytically derived a new type of recurrent unit with a unit-wise feedback and a backward recursion. The similarity with Kalman smoothers and the forward-backward algorithm of HMMs is made explicit, and the equivalence of both approaches is proven by induction. Evaluating on a standard speech recognition task shows that the derived backward recursion gives better results compared to the conventional bidirectional approach. Moreover, adding the derived unit-wise Bayesian recurrent units after layers of larger gated RNNs is capable of considerably improving upon their performance, while only relying on a limited amount of trainable parameters, showing the importance of a probabilistic derivation.

3.4 Conclusion

In this chapter, we derived novel interpretable recurrent units using a Bayesian approach. On top of the theoretical contributions, we demonstrated their practical application in achieving more parameter-efficient speech recognition architectures.

Owing to time limitation, we did not combine the approaches from the two sections, i.e., a layer-wise HMM. We instead shifted our focus to the main theme of this thesis: biologically-inspired SNNs. As the latter can be implemented as a special case of RNNs, this initial work on Bayesian recurrence provided a solid foundation in understanding conventional RNNs and paved the way for the exploration of SNNs in the following chapter. With their potential to mimic biological neural processes more closely than traditional RNNs, SNNs offer promising advancements in computational neuroscience and energy-efficient technology.

4 Spiking Neural Networks

Spiking neural networks are a physiologically plausible alternative to conventional artificial neural networks. In this chapter, we derive the spiking dynamics from single neuron model to multi-layered architectures. Using the surrogate gradient method, the resulting spiking neural networks can be trained like recurrent neural networks. The derived approach serves as a theoretical baseline for the subsequent chapters.

Publication Note

While this chapter serves as background and methods for the subsequent chapters, it consolidates material from the following publications,

- A. Bittar and P. N. Garner (2022a). "A surrogate gradient spiking baseline for speech command recognition". In: *Frontiers in Neuroscience* 16, p. 865897. DOI: 10.3389/fnins.2022.865897
- A. Bittar and P. N. Garner (2022c). *Surrogate gradient spiking neural networks as encoders for large vocabulary continuous speech recognition*. arXiv: 2212.01187 [cs.CL]
- A. Bittar and P. N. Garner (2024). "Exploring neural oscillations during speech perception via surrogate gradient spiking neural networks". In: *Frontiers in Neuroscience* 18. DOI: 10.3389/fnins.2024.1449181

The underlying code has been released and is openly accessible on GitHub at https://github.com/idiap/sparch and https://github.com/idiap/sparse.

4.1 Introduction

So far, our focus has been on conventional ANNs, which lie at the core of recent artificial intelligence advancements, notably in speech processing. Within these widely used networks, artificial neurons typically operate by receiving real numbers as inputs and producing real-valued outputs. In contrast, biological neurons encode and transmit information through binary sequences of events known as spike trains. The neurons in ANNs can be seen as an instantaneous firing rate approximation of biological spiking neurons, so that the information about the individual timings of the spikes is neglected. Several neuroscience studies suggest that precise spike timings are important in transmitting information, especially in the visual cortex and in auditory neurons (Mainen and Sejnowski, 1995; Van Rullen and Thorpe, 2001; Butts et al., 2007; Gollisch and Meister, 2008). With the idea of simulating brain-like networks to process information, this firing rate interpretation of spikes can be improved to spiking neuron models.

Physiologically plausible mathematical models have been developed to describe the neuronal dynamics (Gerstner and Kistler, 2002; Izhikevich, 2007). The resulting spiking neurons constitute the building blocks of SNNs, and have been called the third generation of neural network models (Maass, 1997). Due to the temporal dimension of spike trains, SNNs are naturally adapted to sequential input data, such as speech, which in principle grants them a higher representation capability compared to traditional ANNs (Kasabov, 2019). In current practice however, only rarely do SNNs outperform ANNs (Leng et al., 2018; Moraitis, Sebastian, and Eleftheriou, 2020). Nevertheless, interest in SNNs is growing due to their potential for energy-efficient hardware implementations (Davies et al., 2018; Roy, Jaiswal, and Panda, 2019; Panda, Aketi, and Roy, 2020; Dellaferrera, Martinelli, and Cernak, 2020). The sparsity of spikes over time allows for such implementations, especially when coupled with event-based sensors, leading to the development of portable, low-powered devices. This practical motivation for SNNs is particularly relevant given the energy-consuming nature of conventional ANNs, which rely on high-end GPUs (Pfeiffer and Pfeil, 2018). Recent studies (Jeffares et al., 2022) have shown that spike-based techniques can not only improve the ANN efficiency but also its task performance, further highlighting the potential of SNNs in advancing neural network technology.

ANNs are most commonly trained using SGD, which relies on the chain rule of derivatives. During the forward pass, a mini-batch of input examples is passed through the network, and a loss function is applied to the final outputs. During the subsequent backward pass, the network trainable parameters are updated to minimise the loss. The network gradually adapts to the task at hand by repeating this operation over a data set of prepared examples. SNNs are not directly compatible with gradient descent owing to their non-differentiable threshold behaviour. Different methods have been developed to alleviate the problem. In particular, the surrogate gradient approach allows SNNs to be trained like RNNs using the BPTT algorithm, which is a generalisation of gradient

descent to process sequential data. As demonstrated in Chapter 3, RNNs have proven to be efficient on speech recognition tasks. In general, the reported performance of SNNs is inferior to that of the best ANNs (Wu, Chua, Zhang, Li, and Tan, 2018; Wu, Yılmaz, et al., 2020; Cramer et al., 2020; Yin, Corradi, and Bohté, 2020; Yin, Corradi, and Bohté, 2021; Yao et al., 2021; Shaban, Bezugam, and Suri, 2021), even if the gap is gradually closing. There is in fact theoretical (Maass, 1997; Moraitis, Sebastian, and Eleftheriou, 2020; Perez-Nieves et al., 2021) and recent experimental (Moraitis, Sebastian, and Eleftheriou, 2020) evidence that SNNs can outperform ANNs.

With this work, we aim to make SNNs available to a speech processing audience, the frameworks of which are currently dominated by ANNs. In this chapter, we review relevant spiking neuron concepts and derive an appropriate framework, compatible with ANNs and gradient descent training, to serve as a theoretical baseline for subsequent experiments on speech recognition tasks that will be presented in the next chapters. Our first goal is therefore,

1. To identify which SNN neuron models and training techniques are compatible with successful modern ANN frameworks, and establish a method that can compete with the ANN performance while retaining the advantages of energy efficiency.

More generally, we aim

2. To assess the general capability of SNNs on speech recognition tasks, and how they might represent an attractive alternative to standard ANNs.
3. To use a physiologically plausible approach to provide some insights on how the corresponding biological mechanisms in humans might be functioning.

4.2 Single Neuron Model

In our approach, the single neuron model serves as a fundamental building block for constructing deep SNNs. To ensure compatibility with the modern machine learning frameworks, we prioritise the use of computationally efficient single neuron models that rely on a limited number of parameters.

While physiologically grounded conductance-based neuron models such as the well-known Hodgkin and Huxley model (Hodgkin and Huxley, 1952) offer detailed insights into the dynamics of ion channels, their computationally intensive formulation limits their applicability to our approach. Nonetheless, efforts to simplify these complex models have notably resulted in the reduction of the Hodgkin and Huxley model to just two variables in certain contexts (FitzHugh, 1961; Morris and Lecar, 1981). Building upon these simplifications, more contemporary models, such as the Izhikevich (Izhikevich, 2003) and adaptive exponential integrate-and-fire (Brette and Gerstner, 2005) models, have similarly demonstrated the capacity to accurately replicate voltage traces observed in biological neurons using just membrane potential and adaptation current as essential variables (Badel, Lefort, et al., 2008).

The above mentioned spiking neuron models all use differential equations to describe the neuron's internal dynamics. Alternatively, one can use the Spike Response Model (SRM) approach (Jolivet, Timothy, and Gerstner, 2003), where two kernel functions describe the neuron's responses to incoming and outgoing spikes. Instead of solving differential equations, the evolution of the membrane potential in the SRM is computed by convolving input and output spike trains with their corresponding kernel functions.

The linear Leaky Integrate-and-Fire (LIF) model and its adaptive variant, the Adaptive Leaky Integrate-and-Fire (AdLIF) model possess equivalent formulations in both SRM and differential equation representations (Gerstner and Kistler, 2002). These models are particularly relevant for our approach as they enable flexible and efficient implementations in computational simulations. We therefore concentrate on the LIF and AdLIF neuron models through this chapter.

4.2.1 Leaky integrate and fire

The simplest and most widely used single neuron model is the LIF model, the origin of which dates back to the beginning of the twentieth century with the work of Lapicque (1907). The dynamics of a single neuron are described by the membrane potential $u(t)$, which evolves over time as a function of some input current $I(t)$. In the absence of stimuli, i.e., when $I(t) = 0$, the membrane potential $u(t)$ decays exponentially to some resting value u_{rest} with a time constant $\tau_u \approx 10$ ms. When $I(t) \neq 0$, the membrane potential $u(t)$ integrates the incoming stimuli and increases or decreases accordingly.

As presented by Gerstner and Kistler (2002), the dynamics in continuous time follow the differential equation,

$$\tau_u \, \dot{u}(t) = -\Big(u(t) - u_{\text{rest}}\Big) + R\, I(t)\,, \tag{4.1}$$

where R is the membrane resistance. In order to have spikes, a threshold value ϑ must be added to the model, so that when the potential reaches the critical value, a spike is emitted and the potential is reset to a new value $u_r < \vartheta$.

$$\text{if} \quad u(t = t^f) \geq \vartheta \quad \text{then} \quad s(t^f) = 1 \quad \text{and} \quad \lim_{\delta \to 0; \delta > 0} u(t^f + \delta) = u_r\,. \tag{4.2}$$

While the real-valued membrane potential $u(t) \in \mathbb{R}$ characterises the internal state of the neuron, the only information transmitted to other neurons is the binary spike train $s(t) \in \{0, 1\}$ that can be expressed as a sum of delta-functions,

$$s(t) = \sum_f \delta(t - t^f) = \begin{cases} 1 & \text{if } u(t) \geq \vartheta \\ 0 & \text{otherwise} \end{cases}\,. \tag{4.3}$$

In summary, the dynamics of a single neuron are characterised by four phases, (i) integration of stimuli, (ii) decay back to rest in the absence of stimuli, (iii) emission of a short pulse when a critical threshold value is attained, and (iv) recovery period after a spike is emitted. The stimuli $I(t)$ received by the neuron can be approximated as a weighted combination of spikes emitted by pre-synaptic neurons. As illustrated in Figure 4.1, although the membrane potential is constantly evolving, the neuron emits spikes sparingly and remains silent most of the time.

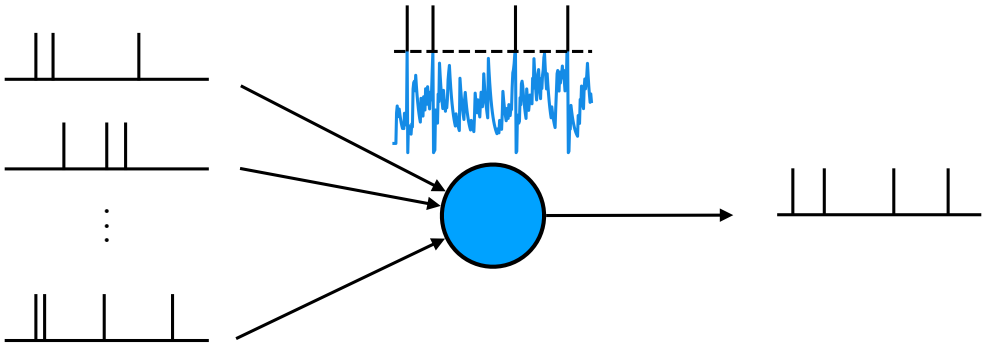

Figure 4.1: Behaviour of a single spiking neuron that receives spike trains as inputs and produces an output spike train. The evolution of the membrane potential is shown in blue.

4.2.2 Adding an adaptation variable

Although widely utilised, the LIF model is not sufficient to reproduce many of the various firing patterns observed in biological neurons, such as adaptive, bursting, transient and

delayed (Gerstner and Kistler, 2002). The idea of a second equation to describe an adaptation (or accommodation) variable between threshold and subthreshold voltage can be traced back to Hill (1936). The work of Treves (1993), Izhikevich (2001) and Brunel, Hakim, and Richardson (2003) have notably led to its modern formulation, in which a recovery variable $w(t)$ is linearly coupled to the membrane potential $u(t)$ in the subthreshold regime, and a mechanism is used for spike-triggered adaptation.

In this formulation, Spike Frequency Adaptation (SFA) is achieved through the adaptation variable $w(t)$ without explicitly including a moving or dynamic threshold (Fuortes and Mantegazzini, 1962; Chacron, Pakdaman, and Longtin, 2003; Jolivet, Rauch, et al., 2006; Badel, Gerstner, and Richardson, 2006). By evolving more slowly than the potential, the adaptation variable modulates the neuron's firing patterns, contributing to a richer repertoire of dynamical behaviours. While SFA typically refers to a form of attenuation in a neuron's activity when stimulated by a prolonged input, we will specifically use this term throughout this thesis to denote the adaptation mechanism in the AdLIF neuron model presented below.

With the objective of incorporating realistic neuronal dynamics into large-scale neural network simulations with gradient descent training, the linear AdLIF neuron model stands out as an adequate compromise between physiological plausibility and computational efficiency. It can be described in continuous time by the following differential equations,

$$\tau_u \, \dot{u}(t) = -\Big(u(t) - u_{\text{rest}}\Big) - R\, w(t) + R\, I(t) - \tau_u \, (\vartheta - u_r) \sum_f \delta(t - t^f) \tag{4.4}$$

$$\tau_w \, \dot{w}(t) = -w(t) + a\Big(u(t) - u_{\text{rest}}\Big) + \tau_w \, b \sum_f \delta(t - t^f)\,. \tag{4.5}$$

Similarly to the LIF model, the neuron's internal state is characterised by the membrane potential $u(t)$ which linearly integrates stimuli $I(t)$ and gradually decays back to a resting value u_{rest} with time constant $\tau_u \in [3, 25]$ ms. A spike is emitted when the threshold value ϑ is attained, $u(t) \geq \vartheta$, denoting the firing time $t = t^f$, after which the potential is decreased by a fixed amount $\vartheta - u_r$. In the following, we will use $u_r = u_{\text{rest}}$ for simplicity. The particularity of the AdLIF model is that a second variable $w(t)$ is coupled to the potential with strength a and decay constant $\tau_w \in [30, 350]$ ms, characterising sub-threshold adaptation. Additionally, $w(t)$ experiences an increase of b after a spike is emitted, which defines spike-triggered adaptation. The spike-induced shifts of the membrane potential and adaptation variable are directly incorporated into Eqs. (4.4) and (4.5) using delta functions. These differential equations can be simplified as,

$$\tau_u \, \dot{u}(t) = -u(t) - w(t) + I(t) - \tau_u \sum_f \delta(t - t^f) \tag{4.6}$$

$$\tau_w \, \dot{w}(t) = -w(t) + a\, u(t) + \tau_w \, b \sum_f \delta(t - t^f)\,. \tag{4.7}$$

by making all time-dependent quantities dimensionless with changes of variables,

$$u \to \frac{u - u_{\text{rest}}}{\vartheta - u_{\text{rest}}}\,, \quad w \to \frac{Rw}{\vartheta - u_{\text{rest}}} \quad \text{and} \quad I \to \frac{RI}{\vartheta - u_{\text{rest}}}\,,$$

and redefining neuron parameters as,

$$a \to Ra\,, \quad b \to \frac{R\,b}{\vartheta - u_{\text{rest}}}\,, \quad \vartheta \to \frac{\vartheta - u_{\text{rest}}}{\vartheta - u_{\text{rest}}} = 1 \quad \text{and} \quad u_{\text{rest}} \to \frac{u_{\text{rest}} - u_{\text{rest}}}{\vartheta - u_{\text{rest}}} = 0\,.$$

By getting rid of R, u_{rest}, u_r and ϑ, this procedure halves the number of parameters, so that a neuron ends up being fully characterised by four parameters: τ_u, τ_w, a and b.

Adaptation and refractoriness

In the AdLIF model, refractoriness is not directly implemented as in some conductance-based models but emerges from the interplay between the adaptation current $w(t)$ and the membrane potential $u(t)$. The immediate refractoriness is produced by the delta-function terms in Eqs. (4.6) and (4.7). After emitting a spike, $u(t)$ is decreased by -1 and $w(t)$ is increased by $+b$, which lowers the membrane potential and results in a longer refractory period during which the neuron is less responsive to stimuli. Additionally, the adaptation time constant τ_w controls how quickly $w(t)$ returns to equilibrium. A longer τ_w means that $w(t)$ stays elevated for a longer period, prolonging the refractory period. Over time, as $w(t)$ increases after each emitted spike and decays more slowly than the potential, the neuron becomes less responsive to sustained inputs, leading to the phenomenon of SFA. The concept of refractoriness is therefore subsumed into the more general adaptation mechanism.

4.2.3 Spike response model equivalent formulation

The AdLIF neuron has an SRM representation that is equivalent to the differential equations representation in Eqs 4.6 and 4.7. In this subsection, we derive the kernel-based formulation step-by-step.

Eigenvalues of AdLIF free equations

The free equations of the AdLIF neuron model are obtained by considering Eqs. (4.6) and (4.7) in the special case where there is no input, $I(t) = 0$, and no emitted spikes, $s(t) = 0$. They can be rewritten in matrix form as,

$$\frac{d}{dt}\begin{bmatrix} u \\ w \end{bmatrix} = \begin{bmatrix} -1/\tau_u & -1/\tau_u \\ a/\tau_w & -1/\tau_w \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix} = A \begin{bmatrix} u \\ w \end{bmatrix}\,. \tag{4.8}$$

The eigenvalues can be found by setting the determinant of $A - \lambda \mathbb{I}$ to zero,

$$\begin{vmatrix} -1/\tau_u - \lambda & -1/\tau_u \\ a/\tau_w & -1/\tau_w - \lambda \end{vmatrix} = 0\,, \tag{4.9}$$

yielding the characteristic polynomial,

$$\lambda^2 + \lambda\left(\frac{1}{\tau_u} + \frac{1}{\tau_w}\right) + \frac{1+a}{\tau_u \tau_w} = 0\,, \tag{4.10}$$

whose roots correspond to the two eigenvalues of the system,

$$\lambda_{1,2} = -\frac{1}{2}\left(\frac{1}{\tau_u} + \frac{1}{\tau_w}\right) \pm \frac{1}{2}\sqrt{\left(\frac{1}{\tau_u} + \frac{1}{\tau_w}\right)^2 - \frac{4(1+a)}{\tau_u \tau_w}}\,. \tag{4.11}$$

Stability conditions

In order to prevent the occurrence of exponentially growing solutions and ensure stability, both eigenvalues need to have a strictly negative real part, which can be realised by imposing a lower bound $a > -1$ on the coupling strength. Moreover, allowing eigenvalues to have a nonzero imaginary part introduces the potential for oscillatory modes that may amplify perturbations. This could cause some challenges in terms of numerical stability, convergence and interpretability, especially in the context of deep neural networks trained with gradient descent. We therefore impose an additional upper bound on the values of a leading to the overall stability condition,

$$-1 < a \leq \frac{\left(\tau_w - \tau_u\right)^2}{4\tau_u \tau_w}\,. \tag{4.12}$$

Eigenvectors and projection matrix

By solving the eigenvalue equations $A\vec{v} = \lambda\vec{v}$, i.e.,

$$\begin{bmatrix} -1/\tau_u & -1/\tau_u \\ a/\tau_w & -1/\tau_w \end{bmatrix} \begin{bmatrix} v_{i,u} \\ v_{i,w} \end{bmatrix} = \lambda_i \begin{bmatrix} v_{i,u} \\ v_{i,w} \end{bmatrix}\,, \tag{4.13}$$

we find that the eigenvectors are

$$\vec{v}_i = \begin{bmatrix} v_{i,u} \\ v_{i,w} \end{bmatrix} = \begin{bmatrix} 1 \\ -1 - \tau_u\,\lambda_i \end{bmatrix} \tag{4.14}$$

for $i = 1, 2$, which defines the projection matrix P and its inverse P^{-1},

$$P = \begin{bmatrix} 1 & 1 \\ -1 - \tau_u \lambda_1 & -1 - \tau_u \lambda_2 \end{bmatrix} \quad \text{and} \quad P^{-1} = \frac{1}{\tau_u(\lambda_1 - \lambda_2)} \begin{bmatrix} -1 - \tau_u\lambda_2 & -1 \\ 1 + \tau_u\lambda_1 & 1 \end{bmatrix} . \tag{4.15}$$

Solving the uncoupled system

The projection matrix defined by the eigenvectors allows us to transform the system of differential equations into a diagonal form,

$$\frac{d}{dt} \begin{bmatrix} u \\ w \end{bmatrix} = P \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} P^{-1} \begin{bmatrix} u \\ w \end{bmatrix} , \tag{4.16}$$

which can be rewritten as,

$$\frac{d}{dt} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} , \tag{4.17}$$

in terms of variables x, y, defined as,

$$\begin{bmatrix} x \\ y \end{bmatrix} = P^{-1} \begin{bmatrix} u \\ w \end{bmatrix} . \tag{4.18}$$

As the equations are now uncoupled, they have simple solutions of the form,

$$x(t) = x_0\, e^{\lambda_1 t} \quad \text{and} \quad y(t) = y_0\, e^{\lambda_2 t} . \tag{4.19}$$

Impulse responses

Starting from some arbitrary initial conditions u_0 and w_0, we now have an analytical solution that describes how the unforced system evolves and decays back to equilibrium.

$$\begin{bmatrix} u(t) \\ w(t) \end{bmatrix} = e^{A\, t} \begin{bmatrix} u_0 \\ w_0 \end{bmatrix} = P \begin{bmatrix} e^{\lambda_1 t} & \\ & e^{\lambda_2 t} \end{bmatrix} P^{-1} \begin{bmatrix} u_0 \\ w_0 \end{bmatrix} \tag{4.20}$$

Inserting P and P^{-1} gives us,

$$\begin{bmatrix} u(t) \\ w(t) \end{bmatrix} = \alpha_1\, e^{\lambda_1 t} \begin{bmatrix} 1 \\ -1 - \tau_u\lambda_1 \end{bmatrix} + \alpha_2\, e^{\lambda_2 t} \begin{bmatrix} 1 \\ -1 - \tau_u\lambda_2 \end{bmatrix} = \alpha_1\, e^{\lambda_1 t}\, \vec{v_1} + \alpha_2\, e^{\lambda_2 t}\, \vec{v_2} \tag{4.21}$$

where the coefficients are defined as,

$$\alpha_1 = -\frac{u_0(1 + \tau_u\lambda_2) + w_0}{\tau_u(\lambda_1 - \lambda_2)} , \quad \alpha_2 = \frac{u_0(1 + \tau_u\lambda_1) + w_0}{\tau_u(\lambda_1 - \lambda_2)} , \tag{4.22}$$

and

$$\alpha_1 + \alpha_2 = u_0 \,. \tag{4.23}$$

Response to an input pulse

Let us now inject a single input pulse $I(t) = \delta(t)$ into the system at equilibrium. We can assume that it will instantly bring the system into a state $u = 1$, $w = 0$. The response of the system to such an input pulse of magnitude 1 at $t = 0$ is then described by equation (4.21) with initial conditions $u_0 = 1$ and $w_0 = 0$, which represents a linear combination of the eigenvectors,

$$\begin{bmatrix} u_0 \\ w_0 \end{bmatrix} = \beta_1 \, \vec{v}_1 + \beta_2 \, \vec{v}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{4.24}$$

with coefficients,

$$\beta_1 = -\frac{1 + \tau_u \lambda_2}{\tau_u (\lambda_1 - \lambda_2)} \quad \text{and} \quad \beta_2 = 1 - \beta_1 \,. \tag{4.25}$$

Response to an afterspike pulse

Similarly, in response to an afterspike pulse, $s(t) = \delta(t)$, let us assume that the membrane potential instantly decreases by $\Delta u = -1$ and that the adaptation current instantly increases by $\Delta w = b$. Again, the response of the system is a linear combination of the eigenvectors,

$$\begin{bmatrix} u_0 \\ w_0 \end{bmatrix} = \gamma_1 \, \vec{v}_1 + \gamma_2 \, \vec{v}_2 = \begin{bmatrix} -1 \\ b \end{bmatrix} \tag{4.26}$$

with coefficients,

$$\gamma_1 = \frac{(1 + \tau_u \lambda_2) - b}{\tau_u (\lambda_1 - \lambda_2)} \quad \text{and} \quad \gamma_2 = -1 - \gamma_1 \,. \tag{4.27}$$

SRM formulation of AdLIF neuron

Using the SRM formulation, the membrane potential $u(t)$ is defined as the convolution of input stimuli $I(t)$ and output spike trains $s(t)$ with their corresponding kernel functions $\kappa(t)$ and $\eta(t)$,

$$u(t) = \kappa(t) * I(t) + \eta(t) * s(t) \,. \tag{4.28}$$

In continuous time, it can then be described in terms of integrals instead of differential equations as,

$$u(t) = \int_0^\infty \kappa(t') \, I(t - t') dt' + \int_0^\infty \eta(t') \, S(t - t') dt' \,, \tag{4.29}$$

where the two kernels $\kappa(t)$ and $\eta(t)$ describe the response to an input pulse and the response to an afterspike reset pulse respectively,

$$\kappa(t) = \left(\beta_1\, e^{\lambda_1 t} + (1-\beta_1)\, e^{\lambda_2 t}\right)\Theta(t) \tag{4.30a}$$

$$\eta(t) = \left(\gamma_1\, e^{\lambda_1 t} - (1+\gamma_1)\, e^{\lambda_2 t}\right)\Theta(t)\,. \tag{4.30b}$$

Here λ_1, λ_2 are the eigenvalues of the system given in Eq. (4.11), β_1 and γ_1 are defined in Eqs. (4.25) and (4.27) and $\Theta(t)$ is the Heaviside step function. The response of the input kernel is such that the membrane potential increases by $\Delta u = 1$, without any effect on the recovery current, i.e., $\Delta w = 0$. The afterspike reset kernel is such that the membrane potential decreases by $\Delta u = -1$ and the recovery current jumps by an amount $\Delta w = b$, The kernel functions $\kappa(t)$ and $\eta(t)$ are illustrated in Fig. 4.2.

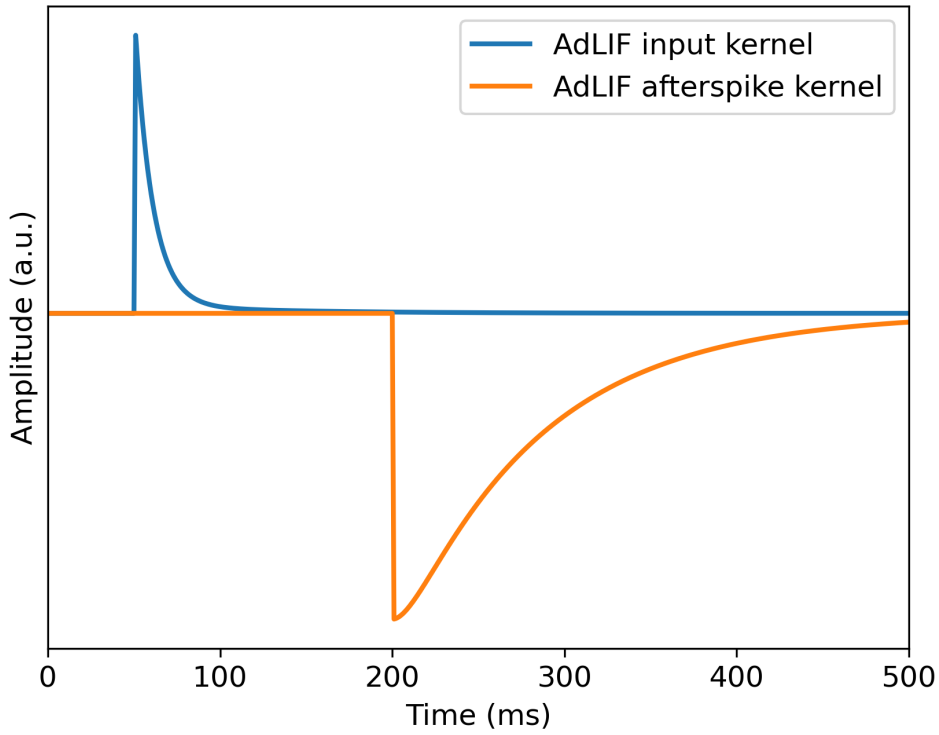

Figure 4.2: Membrane potential response of an AdLIF neuron to an input pulse at $t = 50$ ms (in blue) and to an emitted spike at $t = 200$ ms (in orange). The neuron parameters are $\tau_u = 10$ ms, $\tau_w = 100$ ms, $a = 0.2$ and $b = 1$.

4.2.4 Reduction to LIF neuron

With $a = b = 0$, the AdLIF neuron reduces to the standard LIF model with kernel functions,

$$\kappa(t) = e^{-t/\tau_u}\,\Theta(t) \quad \text{and} \quad \eta(t) = -e^{-t/\tau_u}\,\Theta(t)\,. \tag{4.31}$$

As illustrated when comparing Figure 4.3 with Figure 4.2, the main effect of the adaptation variable lies in the afterspike response. While the LIF model uses the same relatively short time constant τ_u for both input and afterspike responses, the AdLIF afterspike response can inhibit the neuron on a longer scale using dedicated neuron parameters τ_w, a and b. These additional parameters also yield a higher heterogeneity of responses and firing patterns among neurons.

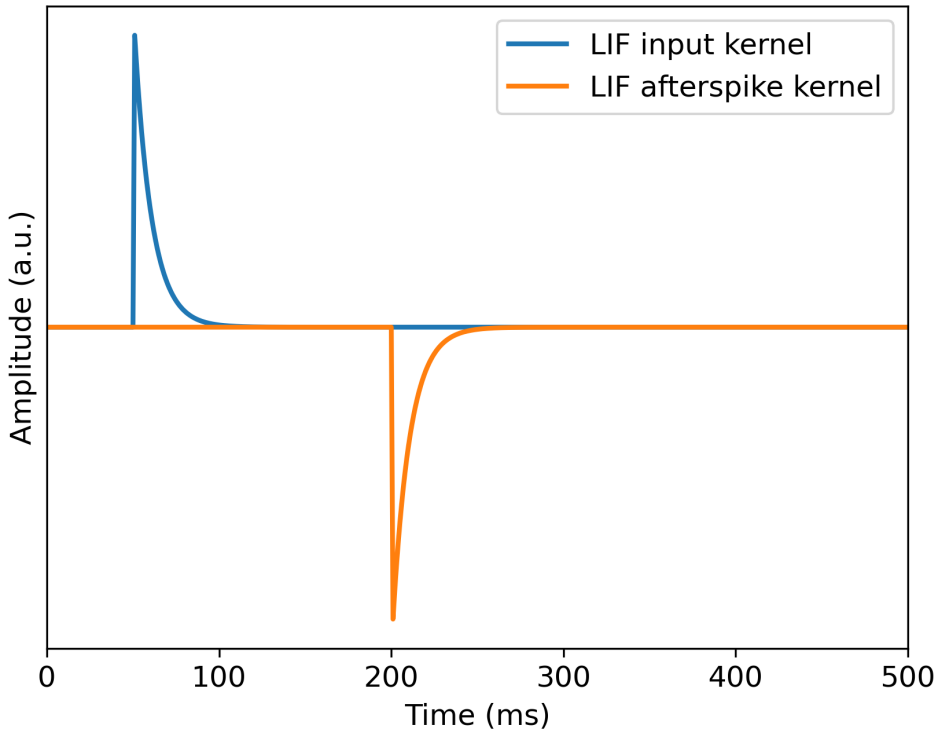

Figure 4.3: Membrane potential response of a LIF neuron to an input pulse at $t = 50$ ms (in blue) and to an emitted spike at $t = 200$ ms (in orange). Compared to the AdLIF, here the only neuron parameter is $\tau_u = 10$ ms, so that both responses have the same shape.

4.2.5 Discrete time formulation

Here we use a finite difference scheme to approximate the derivatives in the continuous-time equations of the AdLIF neuron to differences in discrete time.

Forward-Euler first-order exponential integrator method

The continuous-time differential equations of the AdLIF neuron from Eqs. (4.6) and (4.7) are of the form,

$$\tau\,\dot{y} = -y + \mathcal{N}(y) + \tau\,\gamma\,s(t)\,, \tag{4.32}$$

where $\mathcal{N}(y)$ is a nonlinear function and $s(t)$ is a spike train defined as

$$s(t) = \sum_f \delta(t - t^f)\,. \tag{4.33}$$

Multiplying both sides by $\frac{1}{\tau}\exp(\frac{t}{\tau})$ and then integrating over $[t_n, t_{n+1}]$, where $t_n = n\,\Delta t$, yields

$$\int_{t_n}^{t_{n+1}} \left(\dot{y}\exp\frac{t}{\tau} + y\,\frac{1}{\tau}\,\exp\frac{t}{\tau}\right)dt = \frac{1}{\tau}\int_{t_n}^{t_{n+1}} \mathcal{N}(y)\exp\frac{t}{\tau}\,dt + \gamma\sum_f \int_{t_n}^{t_{n+1}} \exp\frac{t}{\tau}\,\delta(t - t^f)\,. \tag{4.34}$$

The left hand side has an exact solution

$$y(t)\,\exp\frac{t}{\tau}\bigg|_{t=t_n}^{t_{n+1}} = \left(y_{n+1}\exp\frac{\Delta t}{\tau} - y_n\right)\exp\frac{t_n}{\tau}\,. \tag{4.35}$$

For the first term of the right hand side, the nonlinearity $\mathcal{N}(y)$ can be approximated as constant over $[t_n, t_{n+1}]$ for sufficiently small Δt, so that $\mathcal{N}(y) \approx \mathcal{N}(y_n)$ and we can solve it as

$$\frac{1}{\tau}\int_{t_n}^{t_{n+1}} \mathcal{N}(y)\exp\frac{t}{\tau}\,dt \approx \mathcal{N}(y_n)\,\exp\frac{t}{\tau}\bigg|_{t=t_n}^{t_{n+1}} = \mathcal{N}(y_n)\exp\frac{t_n}{\tau}\left(\exp\frac{\Delta t}{\tau} - 1\right). \tag{4.36}$$

Finally for the last term, the width Δt of the interval $[t_n, t_{n+1}]$ can be set sufficiently small to include at most a single spike. The exact firing time $t^f \in [t_n, t_{n+1}]$ can then be discretised as $t^f = t_n$ so that $s_n = \sum_f \delta(t_n - t^f)$ and

$$\gamma\sum_f \exp\frac{t^f}{\tau}\bigg|_{t^f\in[t_n,t_{n+1}]} = \gamma\exp\frac{t^n}{\tau}\,s_n \tag{4.37}$$

Putting everything together, we get the following update equation for y in discrete time,

$$y_{n+1} = \exp\frac{-\Delta t}{\tau}\Big(y_n + \gamma\, s_n\Big) + \Big(1 - \exp\frac{-\Delta t}{\tau}\Big)\,\mathcal{N}(y_n)\,. \tag{4.38}$$

AdLIF discrete time formulation

Using Eq. (4.38) derived from the forward-Euler first-order exponential integrator method, the differential equations of the AdLIF neuron can now be solved in discrete time. After initialising $u_0 = w_0 = s_0 = 0$, and defining $\alpha := \exp\frac{-\Delta t}{\tau_u}$ and $\beta := \exp\frac{-\Delta t}{\tau_w}$, the neuronal dynamics can be solved by looping over time steps $t = 1, 2, \ldots, T$ as,

$$u_t = \alpha\Big(u_{t-1} - s_{t-1}\Big) + (1-\alpha)\Big(I_t - w_{t-1}\Big) \tag{4.39}$$

$$w_t = \beta\Big(w_{t-1} + b\, s_{t-1}\Big) + (1-\beta)\, a\, u_{t-1} \tag{4.40}$$

$$s_t = \Big(u_t \geq 1\Big)\,. \tag{4.41}$$

We apply stability conditions from Eq. 4.12 for the value of the coupling strength a. Additionally, we constrain the values of the neuron parameters to biologically plausible ranges (Gerstner and Kistler, 2002; Augustin, Ladenbauer, and Obermayer, 2013),

$$\tau_u \in [3, 25]\,\text{ms}, \quad \tau_w \in [30, 350]\,\text{ms}, \quad a \in [-0.5, 5], \quad b \in [0, 2]\,. \tag{4.42}$$

The four neuron parameters τ_u, τ_w, a and b all characterise the shape of the membrane potential response to input and output spikes illustrated in 4.2.

4.3 Networks

4.3.1 Relationship between spiking and recurrent neural networks

Similar to an MLP with Eq. (2.14), the stimulus of a layer of N^l spiking neurons can be implemented as a linear combination of spike trains $s^{l-1} \in \{0,1\}^{T\times N^{l-1}}$ from the N^{l-1} neurons in the previous layer

$$I^l = W^l\, s^{l-1} + b^l\,. \tag{4.43}$$

If recurrent connections are enabled, the stimulus of the l-th layer also includes a feedback term from its own spike trains $s^l \in \{0,1\}^{T\times N^l}$, resulting in an equation similar to Eq. (2.24) for RNNs,

$$I^l_t = W^l\, s^{l-1}_t + V^l\, s^l_{t-1} + b^l\,. \tag{4.44}$$

Here the weight matrices $W^l \in \mathbb{R}^{N^{l-1}\times N^l}$ and $V^l \in \mathbb{R}^{N^l\times N^l}$ correspond to the strength of the synaptic connections, and the bias b^l to heterogeneous resting values of the membrane potential among neurons. The excitatory and inhibitory connections between physiological neurons are here represented by positive and negative weights respectively.

Enabling layer-wise recurrence is biologically plausible, as neurons frequently form recurrent connections with other neurons in their local circuits. While longer-range recurrent connections are also common in the brain, we do not consider them here for compatibility with deep learning frameworks, though they represent an interesting direction for future work. In terms of unit-wise recurrent connections, diagonal elements of V^l are set to zero as afterspike self inhibition is already accounted for in Eq. (4.39). While this choice excludes autapses, which are rare but do exist in the brain, it simplifies the model for our study.

Additionally, a binary mask can be applied to matrices W^l and V^l to limit the number of nonzero connections. Similarly, a portion of neurons in a layer can be reduced to LIF dynamics without any SFA by applying another binary mask to the neuron adaptation parameters a and b. Indeed, as illustrated in Section 4.2.4, if $a = b = 0$, the adaptation current vanishes $w_t = 0\ \forall t \in \{1, 2, \dots, T\}$ and has no more impact on the membrane potential.

The forward pass through an SNN is then defined by vectorising Eq. (4.44) and Eqs. (4.39)-(4.41), and looping them over layers and time. The main difference with ANNs is therefore that the dynamics of the membrane potential, combined with the threshold behaviour replace the simple activation function of Eq. (2.15) and produce binary signals $s^l \in \{0,1\}^{T\times N^l}$ instead of real-valued ones $y^l \in \mathbb{R}^{T\times N^l}$. As pointed out by Neftci, Mostafa, and Zenke, 2019, such dynamics can be viewed as a nonlinear activation function which makes SNNs a special case of RNNs. Nevertheless, throughout this thesis the term RNN refers to purely non-spiking recurrent network, as defined in Section 2.3.3.

4.3.2 Neural heterogeneity

In reaction to a given stimulus I_t, different sets of values for the neuron parameters τ_u, τ_w, a and b will lead to different firing patterns, which is an important particularity of SNNs. In ANNs, since the same activation function is typically applied to all neurons, as defined in Eq. (2.15), two neurons receiving the same stimulus I_t will always produce the same output.

As demonstrated by Perez-Nieves et al. (2021), the introduction of heterogeneity in the spiking neuron parameters can considerably improve the network performance, especially for tasks that have a rich temporal structure. This form of neural heterogeneity therefore seems to represent a theoretical advantage of SNNs over standard ANNs on such tasks, as it may allow superior representations of the temporal information.

As reviewed by Apicella et al. (2021), trainable or adaptable activation functions have also been used inside ANNs, and are known to improve their accuracy. In this thesis however, we do not attempt to additionally cover this large field. The above review points out that the enabled improvements can usually be replicated using a more conventional non-trainable homogeneous activation function, and simply more neurons or layers.

In line with the current standard practices in the field, the ANNs we implement in Chapter 5 for comparison with our ANNs all use homogeneous activation functions, as defined in Equation (2.15). As our own ANN implementations sometimes outperform those of the literature (Cramer et al., 2020; De Andrade et al., 2018), we believe that they form an adequate ANN baseline for comparing with SNNs.

4.3.3 Relationship between SNNs and state-space models

With our approach, a recurrent layer of spiking neurons can be represented as a threshold-based non-linear SSM with feedback. We can write the differential equations of the system as those of a continuous time SSM,

$$\frac{d}{dt}\begin{bmatrix} u(t) \\ w(t) \end{bmatrix} = \begin{bmatrix} W \\ 0 \end{bmatrix} s_{\text{in}}(t) + A \begin{bmatrix} u(t) \\ w(t) \end{bmatrix} + \begin{bmatrix} V - \mathbb{I} \\ b\,\mathbb{I} \end{bmatrix} s_{\text{out}}(t) \tag{4.45}$$

$$s_{\text{out}}(t) = \Theta\Big(u(t) - \vartheta\Big)\,, \tag{4.46}$$

where Eq. (4.45) is the state equation and Eq. (4.46) the observation equation.

For a layer of N neurons, the state of the complete system is characterised by a $2N$-dimensional vector $\tilde{u} := [u_1, u_2, \ldots, u_N, w_1, w_2, \ldots, w_N](t)$. The state equation from Eq.

(4.45) can then be vectorised as,

$$\frac{d}{dt}\tilde{u}(t) = \tilde{W}\, s_{\text{in}}(t) + \tilde{A}\, \tilde{u}(t) + \tilde{V}\, s_{\text{out}}(t)\,. \tag{4.47}$$

As described in Eq. (4.8), the state transition matrix $\tilde{A} \in \mathbb{R}^{2N\times 2N}$ depends on neuron parameters τ_u, τ_w and $a \in \mathbb{R}^N$. It represents a unit-wise feedback from the system's dynamics, characterising leakiness and subthreshold adaptation,

$$\tilde{A} = \begin{bmatrix} -1/\tau_{u,1} & & & & -1/\tau_{u,1} & & & \\ & -1/\tau_{u,2} & & & & -1/\tau_{u,2} & & \\ & & \ddots & & & & \ddots & \\ & & & -1/\tau_{u,N} & & & & -1/\tau_{u,N} \\ a_1/\tau_{w,1} & & & & -1/\tau_{w,1} & & & \\ & a_2/\tau_{w,2} & & & & -1/\tau_{w,2} & & \\ & & \ddots & & & & \ddots & \\ & & & a_N/\tau_{w,N} & & & & -1/\tau_{w,N} \end{bmatrix}. \tag{4.48}$$

Matrix $\tilde{W} \in \mathbb{R}^{2N\times M}$ has zeros in its bottom half as the input spike trains $s_{\text{in}}(t) \in \{0,1\}^M$ stimulate the membrane potential but not the adaptation currents. Matrix $\tilde{V} \in \mathbb{R}^{2N\times N}$ depends on neuron parameters $b \in \mathbb{R}^N$ and recurrent connections $V \in \mathbb{R}^{N\times N}$. The upper and bottom diagonals correspond to unit-wise feedbacks describing spike-triggered refractoriness, whereas non-diagonal elements in the upper half correspond to synaptic weights between different neurons in the same layer.

$$\tilde{W} = \begin{bmatrix} W_{11} & W_{12} & \dots & W_{1M} \\ W_{21} & W_{22} & \dots & W_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ W_{N1} & W_{N2} & \dots & W_{NM} \\ & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} \quad \text{and} \quad \tilde{V} = \begin{bmatrix} -1 & V_{12} & V_{13} & \dots & V_{1N} \\ V_{21} & -1 & V_{23} & \dots & V_{2N} \\ V_{31} & V_{32} & -1 & \dots & V_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ V_{N1} & V_{N2} & V_{N3} & \dots & -1 \\ b_1 & & & & \\ & b_2 & & & \\ & & b_3 & & \\ & & & \ddots & \\ & & & & b_N \end{bmatrix} \tag{4.49}$$

We illustrate Eq. (4.47) in Figure 4.4 using a block diagram representation.

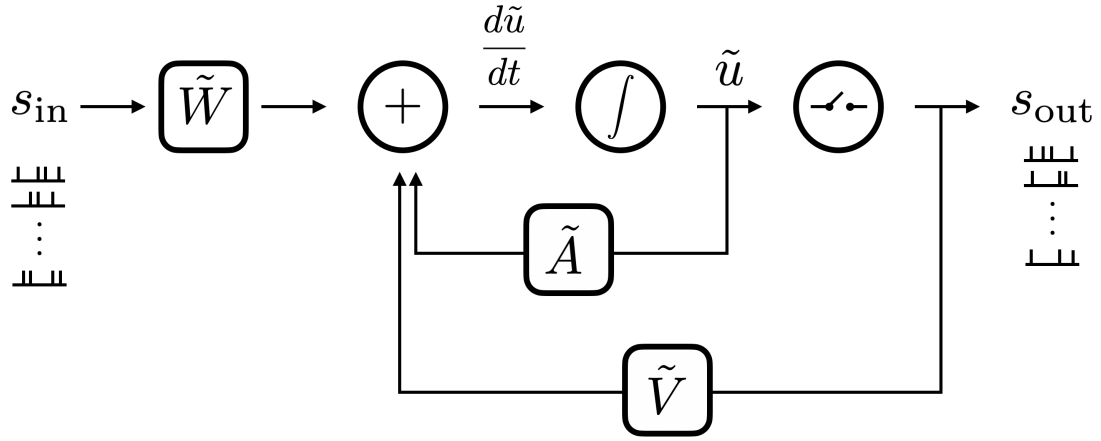

Figure 4.4: Block diagram representation of a layer of spiking neurons with input and output spike trains s_{in} and s_{out}. $\tilde{W}$, $\tilde{V}$ and $\tilde{A}$ represent vectorised versions of input, feedback and state matrices respectively. The gate symbol represents the nonlinear threshold operation.

4.4 Training methods for SNNs

Biological neurons exhibit activity-dependent synaptic plasticity characterised by long term potentiation and depression. This behaviour can be modelled using a form of Spike Timing Dependent Plasticity (STDP), as defined by Dan and Poo (2006). Without the need of labeled examples, this form of unsupervised Hebbian learning is sufficient to detect correlations in the input stimuli and learn encodings of real-world data. However, in order to perform motor tasks, STDP must be combined with a form of global reward-based learning that involves neuromodulators in the brain (Schultz, Dayan, and Montague, 1997; Schultz, 2007; Schultz, 2010; Frémaux and Gerstner, 2016).

Artificial neurons, on the other hand, are most commonly trained using SGD. As detailed in Section 2.3, this technique involves comparing the model predictions to desired outputs for a mini-batch of examples via a loss function. The error of the whole mini-batch is then backpropagated through the network using the chain-rule of derivatives, and the trainable parameters of the entire network are updated accordingly. This form of supervised, global and offline learning is however highly biologically implausible (O'Reilly and Munakata, 2000). In comparison, the weight adjustment with STDP happens *online*, i.e., each time a spike is emitted, and has only a local dependence on the pre- and post-synaptic neurons. Nevertheless, SGD represents the most successful training algorithm used in ANNs. With the aim of evaluating the compatibility of SNNs within ANN frameworks, we will focus on training SNNs with gradient descent.

Using SGD with SNNs is challenging because the derivative of the spike function in equation (4.41) with respect to the membrane potential is zero in the subthreshold regime (when no spikes are emitted, i.e., almost everywhere) and undefined when threshold is reached and a spike is produced. Moreover, small perturbations of the synaptic weights can either lead to considerably different output spike trains, or produce no change at all. The numerous discontinuities caused by the threshold mechanism make the search of a global optimum particularly difficult, especially in multi-layered architectures. Nevertheless, training SNNs with SGD can still be achieved through a variety of approaches that can be grouped into the following three general categories:

1. Sharing weights with ANNs
2. Using a differentiable neuron model
3. Using surrogate gradients.

4.4.1 Weight-sharing with ANNs

The first category, which has been reviewed by Abbott, DePasquale, and Memmesheimer (2016), circumvents the problem of training SNNs by using conventional rate-based ANNs instead. The general approach is to first train an ANN before converting it to an

equivalent SNN (Diehl and Cook, 2015). The resulting architectures do exhibit a spiking behaviour during the forward pass, but the spike timings are ignored in the learning rule. Recently, Wu, Chua, Zhang, Li, Li, et al. (2021) have managed to ensure an efficient gradient based back-propagation by coupling an SNN with an ANN through layer-wise weight sharing. During the forward pass, the SNN computes the exact spiking neural representations, and the ANN the corresponding approximate spike counts (or firing rates). During the backward pass, the error is backpropagated through the ANN via SGD and the weight updates are transferred to the SNN. This tandem learning technique allows fast and efficient learning with multi-layered architectures and has notably proven to be successful on speech recognition tasks (Wu, Yılmaz, et al., 2020). Nevertheless, one could argue that the information about the timings of the individual spikes is still reduced to a rate-based approximation during the backward pass. Moreover, this approach so far neither includes adaptive neuron models nor recurrent connections.

4.4.2 Differentiable neuron models

The second category, which has been reviewed by Neftci, Mostafa, and Zenke (2019), involves spiking neuron models that are differentiable. These include soft-threshold models (Hodgkin and Huxley, 1952; FitzHugh, 1961; Morris and Lecar, 1981; Huh and Sejnowski, 2018), probabilistic models (Jang et al., 2019), spike train convolution models (Lin, Wang, and Hao, 2017; Lin and Shi, 2018; Wang, Lin, and Dang, 2019) as well as single-spike timing-based models (Bodyanskiy and Dolotov, 2013; Mostafa, 2017; Comsa et al., 2020). Although interesting, these models are beyond the scope of our focus on the non-differentiable neuron models presented in Section 4.2.

4.4.3 Surrogate gradient method

Also presented by Neftci, Mostafa, and Zenke (2019), the problem of the non-differentiable threshold behaviour can be solved using surrogate gradients. During the backward pass, the Heaviside step function of the spike generation is smoothed into a suitable differentiable function. With this approach, the threshold operation is only approximated during the backward pass, and remains a step function inside the forward computations. An SNN can then be considered as a special case of RNNs and the BPTT algorithm becomes applicable for training. Nevertheless, the sparsity in time of non-zero gradients, combined with the problems of exploding and vanishing gradients remain.

As illustrated in Figure 4.5, the derivative of the step-function has notably been approximated using the sigmoid derivative (Schrauwen and Van Campenhout, 2006; Zenke and Ganguli, 2018), exponential functions (Shrestha and Orchard, 2018), piecewise linear functions (Bohte, Kok, and La Poutre, 2002; Bellec et al., 2018; Panda, Aketi, and Roy, 2020), a Gaussian (Yin, Corradi, and Bohté, 2020), a multi-Gaussian (Yin, Corradi, and

Bohté, 2021) and a boxcar function (Kaiser, Mostafa, and Neftci, 2020). These typically involve a width hyperparameter controlling how close neurons must be to their threshold value for their gradients to be backpropagated.

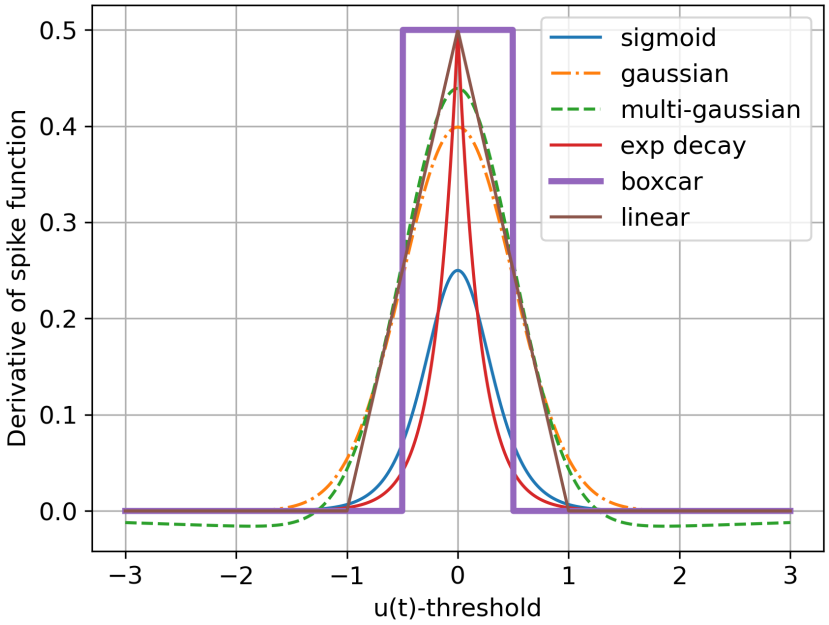

Figure 4.5: Different surrogate gradient functions to approximate the derivative of the step-function responsible for spike generation.

This third and last category of SGD-based training methods is rather versatile compared to the first two as it is not limited to a specific neuron model and allows the use of the different spiking neuron models described in Section 4.2. We will therefore concentrate our analysis on the surrogate gradient approach, but still include a comparison with the tandem method of Wu, Chua, Zhang, Li, Li, et al. (2021).

In an SNN as defined in Section 4.3, by exploiting auto-differentiation inside the deep learning framework PyTorch (Paszke, Gross, Chintala, et al., 2017), one can manually replace the undefined gradient of the step function in Eq. (4.41) with a surrogate, and make the backward pass and therefore gradient descent possible for the whole network. After conducting a series of *ad-hoc* experiments to compare the different surrogate gradient functions illustrated in Figure 4.5, we selected the boxcar function, as it performed the best overall and is computationally efficient. Consequently, we used the boxcar method, previously employed by Kaiser, Mostafa, and Neftci (2020), in all our reported results. It is defined as,

$$\frac{\partial s_t}{\partial u_t} = \begin{cases} 0.5 & \text{if} \quad |u_t - 1| \leq 0.5 \\ 0 & \text{otherwise}\,. \end{cases} \tag{4.50}$$

4.5 Implementation of an AdLIF spiking layer

In order to implement this model of spiking neurons inside a trainable network, we define here a method relying on the surrogate gradient approach that will be compatible with standard deep learning frameworks.

4.5.1 Differential equations formulation

Algorithm 1 Forward pass through AdLIF layer using differential equations formulation

Input: $x \in \mathbb{R}^{B \times T \times N^{l-1}}$

Trainable parameters: matrices W, V and vectors α, β, a, b
Optionally apply masks to W and V to reduce the connectivity
Optionally apply masks to a and b to also include LIF neurons with no SFA
Clamp values of α, β, a and b to specified ranges

Apply feedforward matrix:
$I \leftarrow \text{BatchNormalisation}(W\,x)$

Pre-loop initializations:
$u_t \leftarrow 0$, $w_t \leftarrow 0$, $s_t \leftarrow 0$
$s \leftarrow [\,]$ as an empty list

for $t = 1, 2, \ldots, T$ **do**

  if $V \neq 0$ **then**

    $I_t \leftarrow I_t + V \cdot s_t$

  end if

  $u_t \leftarrow \alpha(u_t - s_t) + (1 - \alpha)(I_t - w_t)$

  $w_t \leftarrow \beta(w_t + b\,s_t) + (1 - \beta)\,a\,u_t$

  $s_t \leftarrow (u_t \geq 1)$

  $s.\text{append}(s_t)$

end for

Stack list elements of s over time dimension

Return $s \in \{0, 1\}^{B \times T \times N^l}$

4.5.2 Spike response formulation

Algorithm 2 Forward pass through AdLIF layer using SRM formulation

Input: $x \in \mathbb{R}^{B \times T \times N^{l-1}}$

Trainable parameters: matrix W and vectors α, β, a, b
Optionally apply mask to W to reduce the connectivity
Optionally apply masks to a and b to also include LIF neurons with no SFA
Clamp values of α, β, a and b to specified ranges
Compute SRM parameters λ_1, λ_2, β_1, γ_1 from α, β, a and b

Apply feedforward matrix:
$I \leftarrow \text{BatchNormalisation}(W\,x)$

Convolve with input kernel:
Define time vector: $\vec{t} = [0, 1, \ldots, T-1]\Delta t$
Compute input kernel: $\kappa = \beta_1 \exp(\lambda_1 \vec{t}) + (1 - \beta_1) \exp(\lambda_2 \vec{t})$
Compute membrane potential: $u \leftarrow I * \kappa$

Precompute afterspike kernel:
$\eta = \gamma_1 \exp(\lambda_1 \vec{t}) - (1 + \gamma_1) \exp(\lambda_2 \vec{t})$

Pre-loop initializations:
$s \leftarrow [\,]$ as an empty list

for $t = 1, 2, \ldots, T$ **do**

  Check which neurons fire:
  $s_t \leftarrow (u_t \geq 1)$
  $s.\text{append}(s_t)$

  Apply afterspike resets to neurons that fired:
  $\eta_{\text{shifted}} \leftarrow [0, 0, \ldots, 0, \eta_1, \eta_2, \ldots, \eta_{T-t}]$
  $u \leftarrow u + s_t \cdot \eta_{\text{shifted}}$

end for

Stack list elements of s **over time dimension**

Return $s \in \{0, 1\}^{B \times T \times N^l}$

5 Applications of SNNs to Speech Recognition Tasks

After defining our SNN approach in the previous chapter, we first apply it to speech command recognition tasks and obtain competitive performance compared to conventional non-spiking ANNs. Building on this initial success, we then extend the approach to tackle the more challenging task of large vocabulary continuous speech recognition. Overall, we demonstrate that surrogate gradient SNNs can be effectively trained within end-to-end deep learning frameworks and are applicable to various speech recognition tasks. In addition to the potential for low-power hardware implementations due to the inherent energy efficiency of spiking neurons, this chapter establishes a foundation for exploring how trained networks encode speech information in the next chapter, offering insights relevant to neuroscience.

Publication Note

This chapter is based upon the following publications.

- A. Bittar and P. N. Garner (2022a). "A surrogate gradient spiking baseline for speech command recognition". In: *Frontiers in Neuroscience* 16, p. 865897. DOI: 10.3389/fnins.2022.865897
- A. Bittar and P. N. Garner (2022c). *Surrogate gradient spiking neural networks as encoders for large vocabulary continuous speech recognition*. arXiv: 2212.01187 [cs.CL]

The underlying code has been released and is openly accessible on GitHub at https://github.com/idiap/sparch

5.1 A baseline for speech command recognition

In the previous chapter, we defined our SNN approach that combines the AdLIF neuron model with the surrogate gradient method, and allows the resulting SNNs to be trained just like RNNs using the BPTT algorithm. In this section, we first apply this approach to speech command recognition tasks and compare the performance with ANNs. We also discuss potential energy advantages for integration into low-powered devices.

Recently, spiking versions of speech command recognition datasets have been released, using the physiological cochlea model LAUSCHER to convert input audio into spikes (Cramer et al., 2020). We use these as well as their respective non-spiking, traditional versions to conduct experiments with both SNNs and ANNs. Using a recovery current instead of the more conventional moving threshold formulation of Bellec et al. (2018), our implementation of adaptive neurons seems to convincingly improve upon previous efforts on similar tasks (Yin, Corradi, and Bohté, 2020; Yin, Corradi, and Bohté, 2021; Shaban, Bezugam, and Suri, 2021; Salaj et al., 2021), as we achieve new state-of-the-art results with SNNs. Furthermore, a comparison with traditional gated RNNs shows that our spiking baseline is capable of achieving competitive results, even without resorting to recurrent connections, showing the effectiveness of a physiologically inspired approach.

5.1.1 Architecture

The overall speech command recognition pipeline is presented in Figure 5.1.

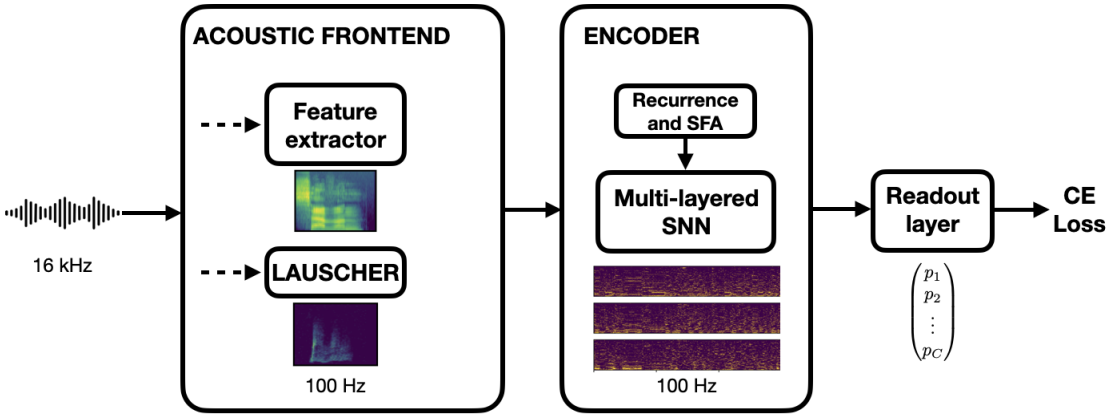

Figure 5.1: Speech command recognition pipeline. For non-spiking datasets, Mel filterbank features are extracted from the input waveform and directly sent to the SNN. For the spiking datasets, spike train representations of the input waveforms have been pre-computed using LAUSCHER. The multi-layered SNN processes the information in the form of spike trains before the final readout layer transforms these into a vector of probabilities over the C classes (with no time dimension).

Acoustic frontend

On the spiking datasets, the inputs $s^{l=0}$ already represent spike trains encodings of the speech signals. Selecting a 100 Hz sampling rate, we directly process them with a multi-layered SNN without requiring any feature extraction. On the non-spiking datasets, we extract Mel filterbank features from the audio waveforms using 25 ms windows and a shift of 10 ms. In this case, the first SNN layer receives real-valued features and outputs the first spike trains.

Encoder

Based on *ad-hoc* experiments, we use two hidden layers. Each hidden unit integrates the incoming stimuli from feedforward and recurrent connections at time step t with Eq. (4.44). The membrane potential is updated using Eqs. (4.39)-(4.40) and a spike is emitted with Eq. (4.41) if threshold is reached.

Readout layer

In both spiking and non-spiking datasets, the spike trains of the last hidden layer need to be converted to class probabilities using a readout layer.

Instead of a sequence of spikes, this readout layer must output one value o_i per neuron $i = 1, \dots, N^L$ that indicates its level of activity over time. During inference, the neuron with the highest activity will be chosen. We considered four different methods for the readout layer,

1. a spiking layer using the spike count,

$$o_i = \sum_{t=1}^{T} s_{t,i}^L \tag{5.1}$$

2. a non-spiking layer using the last potential value over time,

$$o_i = u_{T,i}^L \tag{5.2}$$

3. a non-spiking layer using the maximal potential value over time,

$$o_i = \max_{t=1,\dots,T} u_{t,i}^L \tag{5.3}$$

4. a non-spiking layer using a sum of the softmax of the potential across neurons over time,

$$o_i = \sum_{t=1}^{T} \text{softmax}(u_{t,i}^L)\,. \tag{5.4}$$

In *ad-hoc* experiments presented in Table 5.1, the last technique gave the best performance and is what is used in all presented results. We define readout units as non-spiking LIF neurons that integrate purely feedforward stimuli and update their membrane potential without spiking. We then use Eq. (5.4) to compute the outputs $[o_1, o_2, \dots, o_{N^L}]$, where o_i corresponds to the sum of the i-th unit potential over time, normalised by the softmax function across units at each time step.

Table 5.1: Results on the SHD dataset for SNNs with different types of readout layer.

Network	Potential softmax sum	Spike count	Last potential	Max potential
LIF 3x128	87.27%	**87.45**%	62.45%	77.99%
LIF 3x512	**89.94**%	87.27%	67.74%	79.46%
AdLIF 3x128	**93.06**%	90.35%	90.07%	88.56%
AdLIF 3x512	**93.93**%	93.52%	90.53%	89.61%

Cross-entropy loss function

All models take inputs of size (B, T, N^0) and return outputs o of size (B, N^L), where B is the batch size (i.e., the number of examples in one mini-batch), T the number of time steps, N^0 the number of input features/channels and N^L the number of command classes. The ground truths are given as a vector y of size (B) containing the label indexes. The CE loss is then computed as,

$$\mathcal{L}_{\text{CE}} = -\frac{1}{B}\sum_{b=1}^{B} \log \frac{\exp\Big(o[b, y[b]]\Big)}{\sum_{c=1}^{C}\exp\Big(o[b,c]\Big)}\,. \tag{5.5}$$

ANN baseline for comparison

We implement MLPs, non-gated RNNs, Li-BRUs and GRUs to serve as an ANN-baseline to compare with our SNNs.

With the objective of assessing the capabilities of spiking neurons compared to standard artificial ones, we must define equivalent architectures, that only differ in the type of neurons that they employ. As we will see in Section 5.1.3, even though the spiking neuron parameters τ_u, τ_b, a and b can be made trainable, the total number of trainable parameters in a network remains largely dominated by the amount of connecting weights. This means that in terms of the total number of trainable parameters, SNNs with and without recurrent connections are comparable to non-gated RNNs and MLPs respectively. On the other hand, state-of-the-art gated RNNs remain considerably larger and do not have any direct spiking equivalent in this study.

It is also worth mentioning that even though purely feedforward SNNs do not have recurrent connections, they still include a form of unit-wise recurrence from Eqs. (4.39) and (4.40), where a dependence to the previous time step is present in the dynamics of the membrane potential. This implies that non-recurrent SNNs are theoretically capable of developing a form of memory, without the need of recurrent connections, which is not the case for MLPs.

The readout layer of RNNs, Li-BRUs and GRUs was first defined as a recurrent layer, and the same cumulative sum used for SNNs was applied to its output sequence. Even though this might appear as the best choice for comparison with the SNN technique, we found that applying the cumulative sum in the penultimate ($L-1$) layer instead, followed by a final linear layer gave better results, which is what is used for all reported RNN, Li-BRU and GRU results in the manuscript. For MLPs, the cumulative sum is simply applied to the final output sequences.

Optimiser and learning rate scheduler

The Adam optimizer (Kingma and Ba, 2015) is used for all experiments with initial learning rates of 0.01 and 0.001 on the spiking and non-spiking datasets respectively. A scheduler is also defined to reduce the learning rate by a factor 0.7 if there is no improvement on the validation set accuracy during 10 epochs in a row. This approach proved to be suitable for both SNNs and ANNs and is employed in all presented networks.

5.1.2 Spiking and non-spiking datasets

This research focuses on the bio-inspired processing of auditory information, leading to the formation of appropriate representations and the extraction of relevant features that can then be used for different tasks. The long-term objective is to perform ASR using a physiologically plausible approach that includes waveform to spike conversion followed by processing of the information via spiking neural networks.

However, ASR is a complex task; modern approaches involve end-to-end deep networks, whereas previous techniques needed to solve a series of subtasks, typically feature extraction, phoneme recognition and decoding. In the field of SNNs, it appears that one could benefit from first focusing on the simpler task of speech command recognition to better understand spiking networks. While retaining the processing of auditory information, this more elementary task neither involves too many components in the pipeline, nor requires very deep networks and therefore constitutes a first necessary step in the direction of efficient ASR with SNNs.

We will start by giving a short summary of the biological processes involved in speech perception. We then review LAUSCHER, a bio-inspired model to convert audio waveforms

into spike trains, and some resulting, newly available spiking datasets.

From waveform to spikes

A speech utterance arrives at the ear in the form of air vibrations. From the eardrum it travels via the ossicles to the cochlea, hence the basilar membrane and the organ of Corti, ultimately stimulating hair cells that convert the physical movement into electrical signals. The signals take the form of spike trains on the auditory nerve. Many conventional ASR "filterbank" front-ends are rough analogues of this process, notably modelling the logarithmic response to frequency and to amplitude.

Cramer et al. (2020) have developed LAUSCHER, a biologically plausible model to convert audio waveforms into spike trains. A cochlear model, based on the models developed by Sieroka, Dosch, and Rupp (2006) is used to calculate the hydrodynamic shallow water basilar membrane response to the input waveform. The output of the cochlea then goes into a transmitter pool-based hair cell model, derived from the work of Meddis (1986) and Meddis (1988). Finally, a layer of auditory neurons called bushy cells convert the signal to spike trains using LIF dynamics.

Such a framework allows a direct conversion from audio waveforms into spike trains, while solely relying on physiological processes. In order to train SNNs on speech data, the general and most commonly used alternative is to extract acoustic features from the waveform and interpret them as firing rates to produce spike trains via Poisson processes. Even though the latter approach still shows some physiological plausibility, a single firing rate value is used to produce spikes during the length of a frame (typically 25 ms). This concession comes from the need of using datasets that were originally designed for ANNs, i.e., rate-based approximations of SNNs.

Spiking datasets

In order to rectify the absence of free spike-based benchmark datasets, Cramer et al. (2020) recently released two spiking datasets using LAUSCHER:

- The SHD dataset contains spoken digits from 0 to 9 in both English and German (20 classes). The recordings are from twelve different speakers, two of which are only present in the test set. The train set contains 8332 examples and the test set 2088 (there is no validation set).

- The SSC dataset is based on the Google Speech Commands v0.2 dataset and contains 35 classes from a larger number of speakers. The number of examples in the train, validation and test splits are 75466, 9981 and 20382 respectively.

In both datasets, the original waveforms have been converted to spike trains over 700 input channels. These spiking datasets form an adapted benchmark and allow the investigation of SNNs as well as the comparison of different techniques.

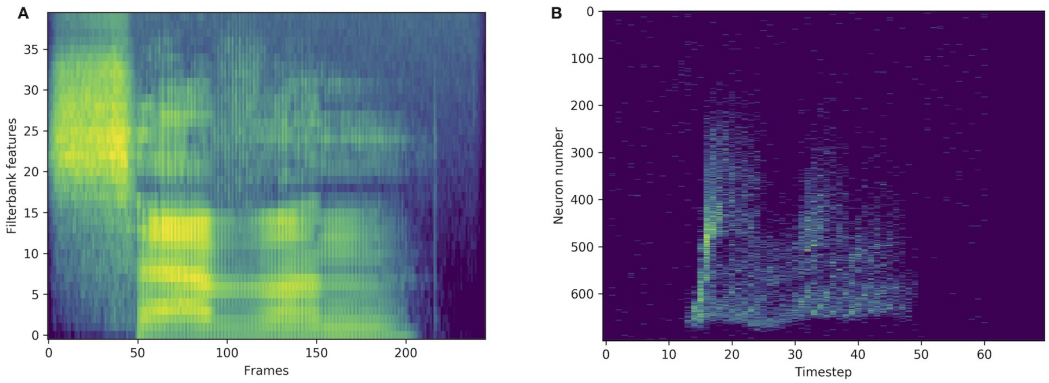

Figure 5.2: Standard representation via filterbank features **(A)** and spike train representation via LAUSCHER **(B)** of the same spoken digit (seven in English) from the SHD dataset.

The current state-of-the-art on the SHD and SSC datasets is summarized in Tables 5.2 and 5.3. The SNN methods are presented in the upper section of the tables, and in the lower sections, the non-spiking CNN and LSTM serve as a point of comparison with the ANN performance.

Table 5.2: State-of-the-art on the SHD dataset.

Method	**Test acc.**
Attention (Yao et al., 2021)	**91.1**%
Recurrent + adaptation (Yin, Corradi, and Bohté, 2021)	90.4%
Recurrent + adaptation (Yin, Corradi, and Bohté, 2020)	84.4%
Recurrent + data augm. (Cramer et al., 2020)	83.2%
Recurrent + heter. time const. (Perez-Nieves et al., 2021)	82.7%
Recurrent (Cramer et al., 2020)	71.4%
Non-recurrent (Cramer et al., 2020)	47.5%
CNN (Cramer et al., 2020)	**92.4**%
LSTM (Cramer et al., 2020)	89%

Non-spiking datasets

The original, non-spiking versions of the SHD and SSC datasets are available and will also be considered in this work. For the Heidelberg Digits (HD) and Google Speech Commands (SC) datasets, acoustic features are extracted from the waveforms and fed

Table 5.3: State-of-the-art on the SSC dataset.

Method	Test acc.
Recurrent + adaptation (Yin, Corradi, and Bohté, 2021)	**74.2**%
Recurrent + heter. time const. (Perez-Nieves et al., 2021)	57.3%
Recurrent (Cramer et al., 2020)	50.9%
Non-recurrent (Cramer et al., 2020)	41.0%
CNN (Cramer et al., 2020)	**77.7**%
LSTM (Cramer et al., 2020)	73%

into neural networks. An input example is illustrated in Figure 5.2, where the filterbank and spiking approaches are compared. The second version of the original SC dataset introduced by Warden (2018) has the same number of examples as its spiking version (SSC), but different training, validation and testing splits of 84843, 9981 and 11005 examples respectively. The SSC has a 70% / 10% / 20% partition instead of 80% / 10% / 10% for the SC. This makes a direct comparison impossible between the accuracies on the two tasks, as the SC has considerably more training data. For the HD and SHD datasets however, the splits are the same. We were not able to find state-of-the-art results on the non-spiking HD dataset, however, for the SC dataset, the state-of-the-art is presented in Table 5.4. Note that certain approaches, such as that of Pellegrini, Zimmer, and Masquelier, 2021, use the first version of the SC dataset. Others, like Zhang et al. (2017) or Rybakov et al. (2020) do use the second version of the dataset, but only 12 labels instead of 35, by using an "unknown" category that includes some of the remaining words. This explains the absence of some of the literature inside the state-of-the-art table, as their results unfortunately cannot directly be compared with ours. In all state-of-the-art tables, the best test accuracies by SNNs and ANNs are written in bold.

Table 5.4: State-of-the-art on the SC dataset (version 2 with 35 labels).

Method	Test acc.
Recurrent + adaptation (Salaj et al., 2021)	**91.21**%
Recurrent + adaptation (Shaban, Bezugam, and Suri, 2021)	91%
Transformers (Gong, Chung, and Glass, 2021)	**98.11**%
Attention RNN (De Andrade et al., 2018)	93.9%

5.1.3 Results

We present results for the LIF and AdLIF neuron models defined in Section 4.2. We distinguish between models with and without recurrent connections, in the form of a

weight matrix V applied to a layer-wise feedback as defined in Section 4.3, so that SNNs without recurrent connections are considerably lighter in terms of number of trainable parameters. In the following, Recurrent Leaky Integrate-and-Fire (RLIF) and Recurrent Adaptive Leaky Integrate-and-Fire (RAdLIF) refer to the recurrent networks using LIF and AdLIF neurons respectively.

The hidden size of a network corresponds to the number of neurons in each of the hidden layers. Although different hidden layers can have different sizes, we focused on hidden layers of the same size in this study. The number of layers is the number of hidden layers plus one (the readout layer). Networks of increasing size and depth were investigated by varying the hidden size from 128 to 1024 neurons per layer, and the number of layers from two to five. Overall, three-layered architectures appeared as the best compromise between size and performance and are used in all presented results with both SNNs and ANNs.

Nevertheless, increasing the number of layers showed that the training of the networks was robust to considerable depth. The chosen approach was therefore able to discard scalability limitations of SNNs on these four tasks, which is very encouraging for the compatibility of SNNs with modern deep learning frameworks.

In the following sections dedicated to each of the four tasks, the results with SNNs will be presented in the upper region of the tables. We distinguish between the following types of network:

1. **tandem**: non-recurrent network of non-adaptive integrate-and-fire neurons (no leak) trained with tandem learning rule.
2. **LIF:** non-recurrent network of non-adaptive LIF neurons trained with a surrogate gradient.
3. **AdLIF:** non-recurrent network of AdLIF neurons trained with a surrogate gradient.
4. **RLIF:** recurrent network of non-adaptive LIF neurons trained with a surrogate gradient.
5. **RAdLIF:** recurrent network of AdLIF neurons trained with a surrogate gradient.

Based on ad-hoc experiments, all presented surrogate gradient SNNs use (i) trainable neuron parameters within fixed ranges of values (α for LIF and α, β, a and b for AdLIF neurons as defined in Section 4.2), (ii) a surrogate gradient with the boxcar function, and (iii) a non-spiking readout layer with a cumulative sum over time as defined in Section 5.1.1.

On the other hand, for the ANN baseline, the following types of network will be presented in the lower section of the tables of results:

1. **MLP:** a simple feed-forward network without recurrence
2. **RNN:** a standard recurrent network
3. **Li-BRU:** a network of light Bayesian recurrent units
4. **GRU:** a network of gated recurrent units

The Li-BRU is the probabilistic version of the Li-GRU presented in Section3.2 with a Softplus activation function instead of a ReLU. We show results with the Li-BRU instead of the Li-GRU, as they were slightly better on all four tasks.

In terms of number of trainable parameters, on the one hand, tandem,LIF and AdLIF networks are roughly equivalent to MLPs, and on the other hand, RLIF and RAdLIF networks are comparable to standard RNNs. By contrast, gated non-spiking networks are considerably larger than all SNNs, since each gate includes weight matrices of its own. With respectively one and two gates, Li-BRUs and GRUs contain approximately two and three times as many parameters as RNNs of the same size. LSTMs were also tested, but since their performance was observed to be slightly lower than that of Li-BRUs and GRUs, they are not mentioned in our experiments.

Note that although surrogate gradient SNNs have trainable parameters inside their activation function, the implemented ANNs only use a non-trainable homogeneous activation function, as it is the case in current common practice. The implemented SNNs and ANNs therefore differ with respect to the trainability of their respective activation functions.

All presented ANNs and SNNs use dropout with $p = 0.1$ as well as batch normalisation (Ioffe and Szegedy, 2015). The only hyperparameters in both artificial and spiking networks are the dropout rate, the learning rate, and the patience and decay factor of its scheduler. Given that we were not initially aiming for state-of-the-art performance, a simple ad-hoc search was carried out to tune them.

Spiking Heidelberg digits

Owing to its small size, the SHD data set allows a thorough investigation of the best choice of architecture. On this specific task, we show for the first time that SNNs can surpass ANNs. Our results are illustrated in Table 5.5. First notice that our best SNN results are better than the attention based SNN state of the art of 91.1% by Yao et al. (2021) (see Table 5.2). More importantly, our approach also improves upon the best reported ANN-performance of 92.4% by Cramer et al. (2020), which used a CNN. Our own attempts with recurrent ANNs only reached 90.40% with GRUs. Even using non-recurrent connections and a relatively small network (3x128), we obtained an accuracy of 93.06% with adaptive spiking neurons. This shows a remarkable ability of SNNs to

compete with much larger standard networks. With recurrence and a higher number of neurons, our best performing SNN even reached a test accuracy of 94.62%, which is extremely promising for the future of spiking networks with surrogate gradients.

We also tested the tandem approach of Wu, Chua, Zhang, Li, Li, et al. (2021) presented in Section 4.4.1, which is an alternative to surrogate gradients. This method so far does not allow recurrent connections. Even if the results are slightly higher (62.64%) than those with a MLP (61.63%), they are significantly lower than what we get with the surrogate gradient approach for a network of the same size (87.04% and 93.06% for LIF and AdLIF neurons respectively). This can be seen as evidence of the importance of using the precise spike timings inside the training mechanism.

Table 5.5: Results on the SHD dataset. All models use three layers, i.e., two hidden and one readout layer. Using the method described in Section 2.1.3 to compute credible intervals, we get error bars between $\pm 2.1\%$ and $\pm 0.9\%$ for test set accuracies between 61.63% and 94.62%. Note that larger ANNs were also tested but only obtained slight improvements and remained under the performance of SNNs of the same size.

Network type	**Recurrent connections**	**Hidden size**	**Test accuracy**
tandem	no	128	62.64%
		1024	68.01%
LIF	no	128	87.04%
		1024	89.29%
AdLIF	no	128	93.06%
		1024	93.57%
RLIF	yes	128	89.75%
		1024	92.51%
RAdLIF	yes	128	92.88%
		1024	**94.62%**
MLP	no	128	61.63%
RNN	yes	128	73.48%
Li-BRU	yes	128	89.61%
GRU	yes	128	**90.40%**
SNN SOTA (Yao et al., 2021)			91.1%
ANN SOTA (Cramer et al., 2020)			92.4%

Non-spiking Heidelberg digits

In order to compare with standard methods for speech recognition, some experiments were made on the original, non-spiking HD dataset. Filterbank features were extracted from the waveforms, and directly fed into various networks. As illustrated in Figure 5.2, compared to a spiking input generated with LAUSCHER, which is a 700 neurons $\times$ 100 timesteps sparse binary tensor, here a non-spiking input typically takes the form of a 40 features $\times$ 250 frames real-valued tensor. Even though the first hidden layer receives real-valued sequences instead of spike trains, spiking networks remain compatible with this approach. They even outperform their non-spiking equivalents, as presented in Table 5.6, where the LIF and RLIF networks surpass the MLP and RNN respectively. The Li-BRU was also tested and gave the best overall performance, athough it requires roughly twice as many trainable parameters as a RNN or RLIF network. The accuracies reached with this filterbank approach are considerably higher than the ones on the spiking dataset. Most investigated models were able to reach a test accuracy close to 100%, which is why we only show a few relevant results. This seems to indicate that some information is lost when performing the conversion from waveform to spikes with LAUSCHER, compared to the extraction of acoustic features. Here the conversion from filterbank features to spike trains happens in a trainable fashion inside the neuronal dynamics of the first hidden layer. Moreover, the initial (non-trainable) transformation of the audio waveforms into filterbank features is fast enough to be performed during training, so that our approach with the non-spiking data sets does not require any preliminary processing of the audio, and could be suitable for low-powered hardware implementations.

Table 5.6: Results on the HD dataset. All models use three layers, i.e., two hidden and one readout layer. Using the method described in Section 2.1.3 to compute credible intervals, we get error bars between $\pm 0.5\%$ and $\pm 0.1\%$ for test set accuracies between 96.99% and 99.96%.

Network type	Recurrent connections	Hidden size	Test accuracy
LIF	no	128	98.40%
RLIF	yes	128	**99.35**%
MLP	no	128	96.99%
RNN	yes	128	99.13%
Li-BRU	yes	128	**99.96**%
GRU	yes	128	99.91%

Spiking Google speech commands

The SSC dataset is roughly ten times bigger than the SHD and has 35 labels instead of 20. It already represents a more complicated classification task to solve for a neural network. Our results are presented in Table 5.7. Here, we managed to close the gap between the SNN and ANN performances by reaching a test accuracy of 77.4% with an SNN. Even though this already represents considerable improvements upon the best previously reported SNN result of 74.2% by Yin, Corradi, and Bohté (2021), our results remain slightly lower than the (non-spiking) CNN performance of 77.7% reported by Cramer et al. (2020), and also lower than our best ANN-performance of 79.05% with GRUs. Nevertheless, in terms of number of trainable parameters, if we compare SNNs to ANNs of the same size, the LIF and AdLIF networks score substantially better (66.67% and 71.66%) than the MLP (only 29.27%), and the RLIF and RAdLIF outperform (73.87% and 73.25%) the RNN (70.01%).

Table 5.7: Results on the SSC dataset. Using the method described in Section 2.1.3 to compute credible intervals, we get error bars of approximately $\pm 0.6\%$ on all accuracies due to the large size of the test set.

Network type	Recurrent connections	Hidden size	Test accuracy
LIF	no	128	66.67%
		512	68.14%
AdLIF	no	128	71.66%
		512	73.58%
RLIF	yes	128	73.87%
		512	75.91%
RAdLIF	yes	128	73.25%
		512	76.21%
		1024	**77.40**%
MLP	no	128	29.27%
RNN	yes	128	70.01%
Li-BRU	yes	512	78.70%
GRU	yes	512	**79.05**%
SNN SOTA (Yin, Corradi, and Bohté, 2021)			74.2%
ANN SOTA (Cramer et al., 2020)			77.7%

Non-spiking Google speech commands

Our results on the non-spiking SC dataset are presented in Table 5.8. We find that our approach is able to reach even better accuracies than the current SNN state-of-the-art of 91.21% by Salaj et al. (2021), which also uses recurrent SNNs, but with a different model of adaptation. We also find that the chosen SNN approach surpasses the performance of almost all implemented ANNs. With a similar number of trainable parameters, the non-recurrent LIF and AdLIF networks give much better results (82.12% and 90.46%) than the MLP which only scores 48.80% on this task. Similarly, the recurrent RLIF and RAdLIF networks achieve accuracies of 90.71% and 92.48% respectively, compared to 90.61% for a non-spiking equivalent RNN. For larger units, we even observe that a non-recurrent, adaptive SNN (AdLIF) is able to outperform a conventional RNN with 93.12% against 92.09%. This illustrates the advantage of physiologically plausible spiking neuron models as the former is significantly lighter than the latter in terms of trainable parameters. By adding recurrence and a larger number of hidden units, we find that our best performing SNN (94.51%) even surpasses the Attention RNN approach of De Andrade et al. (2018) (93.9%), which remained as the ANN state-of-the-art on this task for a long time. With roughly twice as many trainable parameters, the Li-BRU is the only ANN in our baseline that is able to modestly exceed the RAdLIF SNN performance. More generally, the SNN approach appears able to compete with state-of-the-art gated recurrent networks, while retaining a definitive advantage of parameter and energy efficiency; this is extremely encouraging for further work in this direction.

Reducing the network size

So far in this study, we have only considered fully connected layers as it was the most general case and allowed a direct comparison with standard ANNs. The number of trainable parameters can be an important limitation in energy efficient implementations of neural networks. The contributions of the different trainable components used in our spiking architectures are listed here below for a layer l with N^l hidden units.

- Feedforward weights: $N^{l-1} \cdot N^l$
- Recurrent weights: $N^l \cdot N^l$
- Biases: N^l
- LIF neuron parameters (α): N^l
- AdLIF neuron parameters (α, β, a, b): $4 \cdot N^l$

We see that the main contribution to the total number of trainable parameters comes from the weights. Imposing a lower connectivity can therefore greatly reduce the network size. This is especially effective in the first layer on the spiking datasets due to the very high number of input neurons ($N^0 = 700$). Further experiments with a sparser connectivity in the first layer have been carried out on the SHD dataset, and are presented in Table 5.9.

We observe that gradually reducing the connectivity in the first layer only hinders the accuracy by about 1.5%, even when randomly removing up to 99% of the connections. Other experiments presented in Table 5.10 were made with large RAdLIF networks. Here a portion of both feedforward and recurrent connections was randomly removed in all hidden layers. We see that the sparser networks are still able to achieve state-of-the-art accuracies, and that even reducing the number of weights by a factor of 20 only decreases the accuracy by about 1.4%.

Another way of reducing the size of the network is through the parameters of the spiking neuron model. We distinguished four cases in *ad-hoc* experiments: (i) fixed and homogeneous, i.e., the same fixed value for all neurons in the layer, (ii) fixed and heterogeneous, i.e., distributed but fixed values for all neurons in a layer, (iii) trainable and homogeneous, i.e., having a single trainable parameter shared by all neurons in the same layer, and finally (iv) trainable and heterogeneous, i.e., each neuron has its own trainable parameters. We found that the fourth case gave significantly better results and was therefore chosen throughout the whole thesis. As demonstrated by Perez-Nieves et al. (2021), the heterogeneous nature of a spiking layer allows each neuron to develop its own "activation function", defined by the values of its parameters. This appears as one core advantage over conventional ANN models in which the same activation function is shared by all units. We believe that this neural heterogeneity contributes to the superior representational capacities of spiking neurons when applied to auditory sequences.

5.1.4 Towards physiological plausibility

Spike-frequency adaptation

As presented in Section 5.1.3, adding adaptation with the AdLIF model consistently improved the performance compared to the LIF. In this work, adaptation is described by the discrete time Eqs. (4.39) to (4.41), that are not based on a moving threshold, but on subthreshold coupling and spike-triggered currents. These directly stem from the continuous time formulation of Eqs. (4.4) and (4.5), as defined by Gerstner and Kistler (2002), which represent a linear version of the quadratic or Izhikevich neuron model (Izhikevich, 2007). Moreover, Mensi et al. (2012) have shown that depending on the type of cortical neurons, SFA was dominantly mediated by spike-triggered currents or moving threshold. Although both produce SFA, they inherently represent different mechanisms.

A fair amount of the reported approaches have used adaptive neurons on the studied datasets (Yin, Corradi, and Bohté, 2020; Yin, Corradi, and Bohté, 2021; Salaj et al., 2021; Shaban, Bezugam, and Suri, 2021). However, they all employ a moving threshold formulation of adaptation, that is similar to that of Bellec et al. (2018), in which the dynamical threshold is specific to each neuron and increases by a fixed amount after firing, before decaying back to some rest value. Note that Shaban, Bezugam, and Suri

(2021) actually use a more complex version of the adaptive threshold that includes a second time constant. Nevertheless, these are all based on a moving threshold adaptation model. In order to test whether the improvements came from our specific implementation of adaptation, several experiments were made using the moving threshold formulation. A comparison between the two approaches is presented in Table 5.11. We observe that our formulation of adaptation significantly outperforms the moving threshold alternative, especially on the SHD dataset.

Coming back to Table 5.8, on the SC dataset, adding adaptation had the same impact as adding recurrent connections, even though the former requires remarkably fewer trainable parameters than the latter. On the SHD dataset, the effect of adding adaptation is even more pronounced as the considerably lighter AdLIF networks scored better than the RLIFs (see Table 5.5). This shows the importance of the neuron model and, more generally, of a physiologically plausible approach. As pointed out by Perez-Nieves et al. (2021), the heterogeneity of the spiking neurons is a metabolically and computationally efficient strategy. In ANNs, as defined in Eq. (2.15), all neurons have the same activation function. The resulting homogeneity in the behaviour of standard artificial neurons implies that the only source of heterogeneity lies in the synaptic connections, that can be different for each neuron. However, adding neurons to the network increases the computational cost by an order of $\mathcal{O}(N^2)$ for fully-connected layers. With spiking neurons, the more complex neuronal dynamics enable heterogeneous behaviours among neurons by depending on trainable parameters that only scale with $\mathcal{O}(N)$, hence the more efficient computational strategy.

Learning rule

In terms of the learning rule, the compatibility with deep learning methods was favoured over the biological plausibility of the approach. Nevertheless, the surrogate gradient technique can actually be a good candidate towards more physiologically plausible learning algorithms. Kaiser, Mostafa, and Neftci (2020) have recently defined a deep continuous local learning rule (DECOLLE) using random readouts at each layer. Their method still uses surrogate gradients to allow SGD, but is closer to a form of bio-inspired plasticity. This seems to indicate that the evaluated compatibility of training SNNs within ANN frameworks could lead to further improvements of the training methods, and allow more physiologically plausible learning rules by retaining the advantages of well developed ANN techniques.

Finally, compared to the SNN-ANN tandem method of Wu, Chua, Zhang, Li, Li, et al. (2021), the chosen surrogate gradient approach does not ignore the spike timings during the backward pass. In addition to being more flexible to easily include recurrence and different neuron models, the latter gave considerably better results than the former, which suggests that precise spike timings are of importance in processing temporal information.

5.1.5 Towards energy efficient hardware

In other papers that used the spiking datasets (see Tables 5.2 and 5.3), non-recurrent SNNs were always reported to perform substantially less well compared to their recurrent counterparts. In this study, we managed to raise the performance of lighter, non-recurrent SNNs. Our results with non-recurrent AdLIF models on the SHD and SC data sets were even able to surpass those of the best previously reported recurrent SNNs on the same tasks. This allows competitive networks with much fewer trainable parameters, and could lead to hardware implementations that require less space, power and memory.

The average firing rate $\bar{\nu}$ of the implemented spiking networks (over all neurons and all time steps) was observed to consistently converge around $\bar{\nu} \approx 0.1$, which corresponds to 10 Hz. To compare the energy consumption of SNNs with ANNs, similarly to Panda, Aketi, and Roy (2020), one can count the number of Accumulate operations (ACs) and Multiply-and-accumulate operations (MACs) that are required at each time step. Here, we consider ANNs that process sequential inputs and focus on the case with recurrent connections, i.e., RNNs. In contrast to Eq. (2.24) where the matrix multiplications involve non-zero real numbers and results in $N^l(N^{l-1} + N^l + 1)$ MACs for RNNs, Eq. (4.44) only requires $\bar{\nu}N^l(N^{l-1} + N^l + 1)$ ACs for SNNs. The first energy gain therefore comes from the sparsity of the spike trains in Eq. (4.44), which gets rid of $1 - \bar{\nu} \approx 90\%$ of the required operations as most neurons are not activated. Even if the internal neuronal dynamics of SNNs described by Eqs. (4.39−4.41) require additional operations compared to the ANN activation of Eq. (2.15), these only scale with the number of hidden units N^l in the current layer l, whereas the benefits of sparsity scale with $(N^l)^2$.

Moreover, SNNs replace the MACs by ACs in the dot-product computations as a consequence of the binary nature of spike trains, which constitutes the second gain of energy. As presented by Han et al. (2015) with a 45 nm complementary metal-oxide-semiconductor process, a single 32-bit integer AC only requires 0.1pJ compared to 3.2pJ for a MAC. This reduces the energy consumption by another factor of 32. Even if they usually lead to slightly worse accuracies, regularisers can additionally be used to obtain even sparser spike trains and reduce the value of $\bar{\nu}$. By combining the advantages of the sparse and binary nature of the information, a recurrent SNN without regularisers already requires roughly 320 times less energy than a non-spiking RNN of the same size.

On the ANN side, this energy gap can nevertheless be reduced. Low footprint KWS techniques involve network quantisation, (Zhang et al., 2017), a max-pooling based loss function (Sun, Raju, et al., 2016), cascaded executions (Sun, Hoffmeister, et al., 2017; Giraldo, O'Connor, and Verhelst, 2019), compute-in-memory architectures (Schaefer et al., 2021) and various specialized hardware (Conti et al., 2018; Giraldo, Lauwereins, et al., 2020; Kadetotad et al., 2020) designed to reduce the energy consumption of RNNs. Very recently, Jeffares et al. (2022) have notably taken inspiration from SNNs to define a threshold based rank coding approach with considerable speed and efficiency

benefits. Even though the efficiency superiority of SNNs can be nuanced by taking into consideration the above ANN methods, they inevitably play a central role in developing lower powered neuromorphic hardware.

The main motivation for using ANNs is their task performance, which is typically superior to that of SNNs, especially when using sophisticated architectures such as gates or attention. In this work however, we have seen that SNNs consistently outperformed non-gated RNNs of the same size. This apparent superiority can be explained by the heterogeneous and trainable activation of spiking neurons, which represents an advantage over the implemented ANNs, that similarly to common practice, use homogeneous activation functions. The higher representational capabilities of using heterogeneous neural activations in SNNs is well illustrated by the fact that on all four tasks, even the much lighter non-recurrent adaptive SNNs managed to surpass the performance of standard RNNs. Only gated RNNs were able to compete with SNNs and marginally surpass them in most cases. However, layers of Li-BRUs and GRUs require two and three times more operations respectively compared to standard RNN layers, thus expanding the energy gap even more drastically. The sparse event-driven processing of the information in SNNs, combined with their assessed capabilities therefore make them extremely attractive for reaching lower powered hardware implementations dedicated to real-world applications.

5.1.6 Conclusion

In Section 4.1, we set out three goals for our work with SNNs. In concluding, by carefully selecting appropriate techniques, we have established an SNN method that, on top of being compatible with standard deep learning frameworks, is capable of competing with ANNs on the same speech processing tasks, while conserving the advantage of energy efficiency. This represents the main contribution of this section, which in fact fulfils the first goal. The chosen surrogate gradient approach allows SNNs to be trained with gradient descent like conventional ANNs. The resulting compatibility with modern ANN frameworks, combined with the observed scalability of our spiking networks to relatively deep architectures, point towards further applications of this method to more advanced tasks, which will be the focus of the next section. In terms of energy consumption, the implemented SNNs are drastically more efficient compared to standard ANNs of the same size, showing promising pathways for low-powered hardware implementations of neural networks.

At the same time, we have also achieved the second goal of assessing the more general capability of SNNs in comparison to conventional ANNs. We have shown that the particular combination of adaptive spiking neurons, surrogate gradients and automatic differentiation can actually compete with strong ANN baselines on speech recognition tasks. Our implementation of adaptation in the neuron model was able to improve

upon the standard moving threshold formulation, and replace recurrent connections at a considerably lower cost in terms of number of trainable parameters. Such lighter non-recurrent SNNs were even capable of competing with much larger, standard gated recurrent units. While the neurons inside such conventional ANNs all share the same activation function, firing behaviours among spiking neurons can become heterogeneous by making the neuron parameters trainable, which appears to allow more complex representations of the temporal information with fewer neurons inside the network. The implemented SNNs were indeed consistently superior to *equivalent* non-spiking architectures, which further corroborates the hypothesis of greater representational capabilities. This also points towards further investigations of heterogeneous activation functions inside ANNs, as they are not commonly used in current practice.

The success of this physiologically plausible approach to modelling neural networks indicates that our more general third and last goal is still valid. So far, our experiments do not attempt to say anything about biological function. However, they show that a representational capability, that is available to biological entities, is capable of solving the same problems as artificial networks that are known to be capable of exceeding human performance on many tasks. This provides a strong hypothesis for future understanding of the biological mechanisms of the brain, which we will focus on in Chapter 6.

Table 5.8: Results on the SC dataset. All models use three layers, i.e., two hidden and one readout layer. Using the method described in Section 2.1.3 to compute credible intervals, we get error bars between $\pm 0.9\%$ and $\pm 0.4\%$ for test set accuracies between 48.80% and 95.06%.

Network type	Recurrent connections	Hidden size	Test accuracy
LIF	no	128	82.12%
		512	83.03%
AdLIF	no	128	90.46%
		512	93.12%
RLIF	yes	128	90.71%
		512	93.58%
RAdLIF	yes	128	92.48%
		512	**94.51**%
MLP	no	128	48.80%
		512	53.16%
RNN	yes	128	90.61%
		512	92.09%
Li-BRU	yes	128	94.55%
		512	**95.06**%
GRU	yes	128	93.65%
		512	94.32%
SNN SOTA (Salaj et al., 2021)			91.21%
ANN SOTA (Gong, Chung, and Glass, 2021)			98.11%

Table 5.9: Results on the SHD dataset for a non-recurrent AdLIF network of size 3x128, with a sparser connectivity in the first layer. Here a sparsity proportion p corresponds to randomly removing $p\%$ of the connecting weights in the first layer. The second column gives the corresponding number of trainable parameters in the complete architecture.

Sparsity proportion	Parameters	Test accuracy
0%	109'864	93.06%
10%	100'904	92.83%
50%	65'064	92.14%
90%	29'224	92.33%
95%	24'744	92.14%
99%	21'160	91.59%

Table 5.10: Results on the SHD dataset for a recurrent RAdLIF network of size 3x1024, with a sparser connectivity in all hidden layers. Here, a sparsity proportion p corresponds to randomly removing $p\%$ of the connecting weights (both feedforward and recurrent) in all hidden layers. The second column gives the corresponding number of trainable parameters in the complete architecture.

Sparsity proportion	Parameters	Test accuracy
0%	3'893'288	94.62%
10%	3'507'035	93.80%
50%	1'962'024	93.57%
90%	417'013	92.51%
95%	223'886	93.20%
99%	69'385	90.95%

Table 5.11: Comparison between our adaptation scheme and an alternative moving threshold on the SHD and SC datasets.

Dataset	Network	Moving threshold	Spike-triggered currents
SHD	RAdLIF 3x128	88.79%	**92.87**%
SHD	RAdLIF 3x512	90.21%	**93.75**%
SHD	RAdLIF 3x1024	89.25%	**94.62**%
SC	RAdLIF 3x128	90.95%	**92.48**%
SC	RAdLIF 3x512	93.61%	**94.51**%

5.2 Extending to large vocabulary continuous speech recognition

In Section 5.1, we demonstrated the effectiveness of surrogate gradient SNNs on speech command recognition, showing competitive performance with larger RNNs. This recent progress is primarily due to the surrogate gradient method introduced in Section 4.4.3, which enables SNNs to leverage modern deep learning frameworks. Despite this compatibility, their usage has so far been restricted to relatively simple tasks and small networks, when compared to current end-to-end ANN architectures. In particular, this method has not yet been thoroughly tested in advanced sequence-to-sequence learning scenarios.

In this section, we extend the successful application of our method from speech command recognition to the more challenging task of LVCSR. Even though the scalability of surrogate gradient SNNs to multi-layered architectures was assessed in Section 5.1.3, this assessment was within the constraints of simpler tasks compared to LVCSR. Speech command recognition involves very short audio samples, each labeled with a single class, whereas LVCSR must handle long speech utterances and predict corresponding token arrangements using encoder-decoder architectures that typically involve more components and parameters.

LVCSR presents unique challenges for SNNs, particularly concerning the potential for vanishing gradients due to the longer sequences and its sequence-to-sequence mapping. In surrogate gradient SNNs, the length of the temporal unrolling during backpropagation is reduced by the sparse and binary properties of the variables to which recurrence is applied, resulting in sparse gradients. This can theoretically lead to vanishing gradients and inefficiencies in learning, especially in such sequence-to-sequence scenarios with long-range temporal dependencies. On the other hand, compared to continuous signal propagation in standard RNNs, the inherent sparsity of spike-based communication could mitigate the risk of exploding gradients. Without having to resort to gates as in LSTMs or GRUs, energy-efficient and lightweight SNNs could then constitute an attractive alternative for such tasks.

Wu, Yılmaz, et al. (2020) previously used SNNs on LVCSR tasks, however, instead of the surrogate gradient approach, they use the tandem learning rule, presented in Section 4.4.1. Through weight sharing, their approach relies on ANNs to approximate the spike counts, thereby discarding meaningful information about spike timings in the learning mechanism. The only research involving surrogate gradient SNNs on LVCSR that we were able to find is the recent work of Ponghiran and Roy (2022), which uses a custom LSTM-inspired version of SNNs that combines a forget gate with multi-bit outputs instead of binary spikes. While these studies represent significant advancements, applying a more standard and physiologically plausible implementation with surrogate gradients on LVCSR remains unexplored.

In this section, we therefore come back to the standard and most commonly used LIF neuron model without gates or multi-bit outputs. By training the resulting physiologically inspired speech encoder on LVCSR tasks, we aim to,

1. Assess the compatibility and scalability of surrogate gradient SNNs within end-to-end sequence-to-sequence architectures.
2. Evaluate the representational capabilities of SNNs as lightweight, energy-efficient speech encoders.
3. Investigate the robustness of SNNs to exploding gradient issues in the context of LVCSR.

More generally, having used a physiological neuron model on connected speech, our work lays the foundation for spike train analysis and comparison with neuroscience, which will be the main focus of the next chapter.

5.2.1 ANN baseline architecture

Using the SpeechBrain framework (Ravanelli, Parcollet, Plantinga, et al., 2021), we selected our ANN baseline based on its default RNN-based ASR recipes for TIMIT (Garofolo et al., 1993) and LibriSpeech (Panayotov et al., 2015). As our surrogate gradient SNNs constitute a special case of RNNs, we assume that substituting them into an RNN baseline will require fewer hyperparameter adjustments such as learning rate and batch size compared to starting from a Transformer-based recipe. The overall ASR pipeline, illustrated in Figure 5.3 contains the following modules.

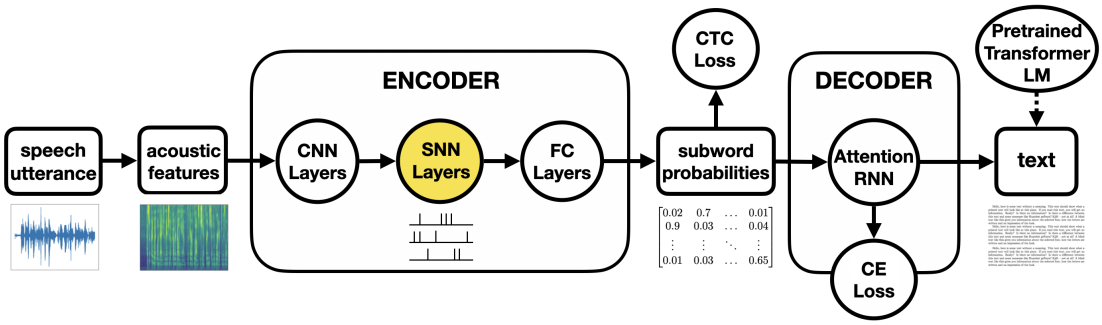

Figure 5.3: Sequence-to-sequence LVCSR pipeline on LibriSpeech, where the LSTM layers from the SpeechBrain baseline have been replaced by an SNN inside the encoder.

Feature extraction

On all tasks, the inputs consist of 40 Mel-filterbank features that are extracted from the 16 kHz utterance waveforms. These features are obtained using 25 ms windows with a 10 ms shift, aligning with standard ASR practice.

Encoder

The 40 filterbank features are passed through two CNN layers with 64 and 128 channels respectively, kernel sizes of (3, 3) and strides of (2, 2). These CNN layers divide both sequence and feature dimensions by a factor of four. Merging the feature and channel dimensions results in tensors with 1280 features.

These tensors are then passed through four bidirectional recurrent LSTM layers of 512 hidden units each, which represent the main components of the encoder and contain the majority of trainable parameters.

The LSTM outputs are projected to 1024 features and finally to the desired number of phoneme or subword classes via Fully-Connected (FC) layers. We use a dropout probability of 0.15 and the Leaky ReLU activation function throughout the whole encoder, except for the last layer which uses a log-softmax activation.

Decoder

On TIMIT, the output log-probabilities from the encoder, which correspond to 40 different phoneme labels (including a blank class), directly go into a CTC loss (Graves, Fernández, et al., 2006) without any additional decoder.

On LibriSpeech (both 100 and 960 hours), these log-probabilities correspond to 5000 byte-pair encoding subword units (Sennrich, Haddow, and Birch, 2015). Here a hybrid CTC/attention architecture (Watanabe et al., 2017) is used. In this approach, an attention-based RNN decoder is employed alongside the CTC loss, with CE loss applied to the decoder predicted sequence. Finally, at inference time, a pretrained Transformer-based language model is used to refine the predictions from the RNN decoder.

Training

On TIMIT, the architectures are trained end-to-end for 50 epochs with the Adadelta optimiser, an initial learning rate of 1 and a batch size of 8. A scheduler reduces the learning rate by a factor of 0.8 if the validation PER does not improve after each epoch. During training, noise is added to the audio samples with a signal-to-noise ratio ranging between 0 and 15 dB to simulate various noise conditions and improve model robustness. Additionally, the speed of the audio is varied between 95% and 105%.

On LibriSpeech 100h, we use the same configuration except the number of epochs is reduced to 25. For our experiments using all 960h of training data, we further reduce the number of epochs to 10. While TIMIT only used CTC, the total loss on LibriSpeech is half CTC from the encoder and half CE from the decoder.

5.2.2 Hybrid ANN-SNN architecture

We focus on the encoder and gradually replace LSTM layers with SNNs. The rest of the pipeline, which corresponds to the default SpeechBrain baseline, remains unchanged. Replacing all layers yields the architecture illustrated in Figure 5.3. In placing the spiking neurons in the encoder we are focusing on the low level auditory processing, as opposed to the higher level semantic processing.

LSTMs have proven representational capabilities for speech. We have no hypothesis that SNNs can replace all of their capabilities. Nevertheless we would expect simpler recurrence to suffice for at least some of the layers. A single bidirectional LSTM layer corresponds to eight times more trainable parameters, compared to a bidirectional RLIF layer with the same number of units. This comes from the absence of gates, as well as a lighter way of tackling the bidirectionality in the SNN compared to RNNs (see Section 2.3.3). In order to compare with equivalent non-spiking layers, we also investigate replacing the LSTM layers with standard non-gated RNNs.

5.2.3 TIMIT experiments

We start with TIMIT to first assess that that our approach can successfully be integrated and trained within an end-to-end ASR architecture. The use of this rather limited data set is here justified because (i) our experiments with SNNs are considerably slower compared to non-spiking RNNs that benefit from a CUDA implementation, and (ii) it allows comparison with non-gated RNNs that were found to diverge on LibriSpeech.

As presented in Table 5.12, we obtain satisfying results when gradually replacing LSTMs in the encoder with SNNs. While requiring eight times fewer parameters, replacing all four LSTM layers with RLIF layers only increases the PER by less than two percent. To compare architectures of more similar sizes, we also replaced the LSTM layers with standard non-gated RNNs. Here the performance is the same between RLIFs and RNNs up to three layers, even though RNNs still employ twice as many parameters due to their implementation of bidirectionality.

These initial experiments show that surrogate gradient SNNs are capable of being trained as part of an end-to-end hybrid architecture. Moreover, they perform similarly to their non-spiking RNN counterpart, which suggests that sparse, binary information is sufficient to encode speech information. It is worth noting that even if LSTMs remain marginally better, they appear to be over-parameterised when used on such a small data set.

Table 5.12: PERs on TIMIT test set via CTC decoding. Using the method described in Section 2.1.3 to compute credible intervals, we get error bars between $\pm 0.8\%$ and $\pm 0.9\%$

Encoder	**Layers**			**Parameters**			**PER [%]**	
	RLIF	RNN	LSTM	RLIF	RNN	LSTM	testing	validation
with SNN	1	0	3	0.9M	0	18.9M	16.1	14.4
	2	0	2	1.7M	0	12.6M	16.6	14.5
	3	0	1	2.5M	0	6.3M	16.8	14.7
	4	0	0	3.3M	0	0	17.6	15.5
without SNN	0	0	4	0	0	26.2M	15.8	14.0
	0	1	3	0	1.8M	18.9M	16.1	14.0
	0	2	2	0	3.4M	12.6M	17.0	14.9
	0	3	1	0	5.0M	6.3M	16.8	15.2
	0	4	0	0	6.6M	0	16.3	14.7

5.2.4 LibriSpeech experiments

After this first success on TIMIT, we now train similar architectures on a larger and more challenging dataset with longer sequences. We use two versions of LibriSpeech, one with 100h and the other with 960h of training data, both utilising the same validation and testing splits. Our results, presented in Table 5.13, improve upon previous efforts with SNNs on the same data sets (Wu, Yılmaz, et al., 2020; Ponghiran and Roy, 2022).

Table 5.13: WERs on LibriSpeech test sets. Using the method described in Section 2.1.3 to compute credible intervals, we get error bars between $\pm 0.2\%$ and $\pm 0.3\%$ for the reported WERs on *test-clean* and *test-other* respectively.

Layers		**Parameters**		**WER [%] 100h**		**WER [%] 960h**	
RLIF	LSTM	RLIF	LSTM	clean	other	clean	other
0	4	0	26.2M	5.9	19.5	3.5	10.0
1	3	0.9M	18.9M	6.3	20.9	3.5	10.7
2	2	1.7M	12.6M	6.7	23.0	4.2	12.7
3	1	2.5M	6.3M	7.3	24.2	4.7	14.6
4	0	3.3M	0	11.5	32.0	9.9	25.1

We first notice that while benefiting from a considerable decrease in the number of trainable parameters, gradually replacing the first three LSTM layers with spiking RLIF layers only marginally degrades the performance (1% absolute difference on *test-clean* and 4% on *test-other*). This corroborates our initial results on TIMIT, and assesses that surrogate gradient SNNs are compatible and effective on LVCSR tasks.

When also replacing the last LSTM layer, the difference in error rates becomes more significant (6% absolute difference on *test-clean* and 15% on *test-other*). Although the number of parameters is reduced by a factor of eight, this result suggests that the gating

mechanism present in LSTMs remains important for an optimal processing of speech information, but a single layer may suffice.

We also compared with fewer LSTM layers and ensured that adding SNNs was indeed beneficial. For instance, two RLIF layers followed by two LSTM layers did outperform two LSTM layers on their own. The trade-off between performance and network size was further investigated using four LSTM layers with fewer hidden units. With a comparable number of trainable parameters, we found that hybrid RLIF-LSTM encoders performed similarly to purely LSTM encoders. These results were left out of the tables for clarity.

Finally, as presented in Table 5.14, we also evaluated the impact of recurrent connections and SFA in the spiking neuron model. We observe that adding either SFA or recurrent connections to a LIF baseline produces comparable improvements. This is in line with our speech command recognition experiments presented in Section 5.1.3, where SFA constituted a parameter-efficient alternative to layer-wise recurrent connections.

Table 5.14: Results on LibriSpeech 100h to evaluate the impact of recurrent connections and SFA in our SNNs. The last two columns contain the WERs [%] on the two test sets. Here we use three SNN layers and one LSTM layer in the encoder.

Model	**SNN parameters**	**clean**	**other**
LIF	1.7M	8.2	26.7
AdLIF	1.7M	7.5	24.8
RLIF	2.5M	7.3	24.2
RAdLIF	2.5M	7.1	23.7

5.2.5 Robustness to exploding gradients

Non-gated RNNs have been known to suffer from exploding and vanishing gradient problems (Bengio, Simard, and Frasconi, 1994; Hochreiter et al., 2001). In our experiments, we found that non-spiking RNNs were particularly susceptible to such issues on the LibriSpeech corpus, which notably involves longer utterances. Even by gradually reducing the initial learning rate from 1.0 to 0.001, we were unable to use them successfully, at the exception of a single RNN layer followed by three LSTMs on the 100h version which gave a similar error rate to its spiking counterpart.

By contrast, spiking RNNs did not suffer from such divergence problems owing to the sparsity of the spike sequences counteracting excessive gradient accumulation from past time steps. Indeed, the surrogate gradient, as defined in Eq. (4.50), will be zero whenever the membrane potential lies sufficiently far from its threshold value. This inherent robustness to exploding gradients makes SNNs an even more viable alternative to standard non-gated RNNs.

5.2.6 Speech encoding using SNNs

The information transmitted between spiking layers takes a sparse and binary form. This is illustrated in Figure 5.4, where we use the firing rates as a mean to add structure to the y-axis. Despite the simplistic method, it is extremely interesting to observe that a spectrogram structure appears, at least distinguishing speech from silence.

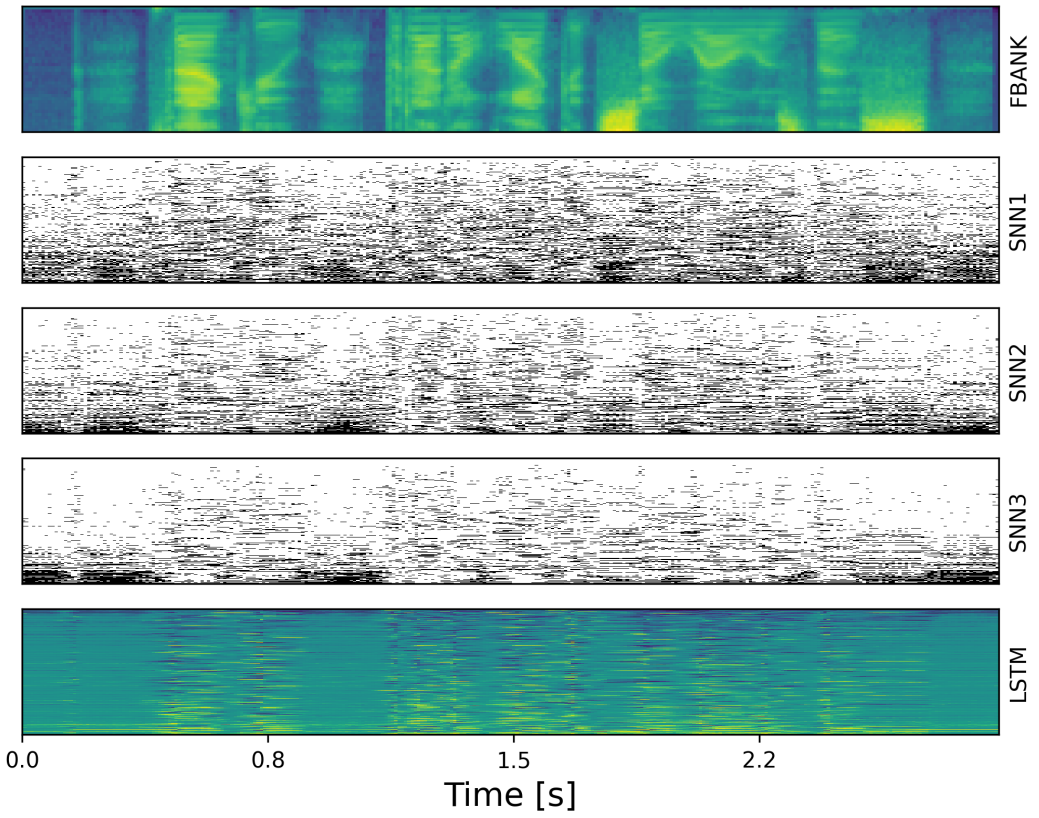

Figure 5.4: Evolution of the representation of speech through the encoder. The utterance in question is 'thank you but I don't like it so well as this'.

As no residual connections are used in our ASR pipeline, the SNN fully transforms speech information into sparse sequences of binary events. Our results confirm that this conversion to spike trains does not result in any loss of valuable information, as a three-layered SNN followed by an LSTM layer improves upon a single LSTM layer.

Our approach demonstrates that SNNs can be successfully trained to encode continuous speech information in the form of spike trains, replicating some of the physiological processes underlying human cognition. On top of indicating that such physiological networks are applicable to LVCSR, our work establishes a tool that could be used towards a better understanding of how the brain represents and processes speech. The analysis of spiking layers trained inside a more physiologically inspired ASR architecture will be the focus of Chapter 6.

5.2.7 Energy efficiency

From Figure 5.4, it is also quite apparent that what gets transmitted from one spiking layer to the next (rows 2-4) is remarkably sparse compared to an LSTM (last row). On average, we measured that 90% of the neurons stay silent, which reduces the number of required operations by a factor of 10. As we detailed in Section 5.1.5 the binary nature of the information also implies that MACs reduce to ACs, which can lower the energy consumption by an additional factor of 32.

As our hybrid ANN-SNN architectures produced competitive results compared to purely ANN baselines with similar number of parameters, our work demonstrates that SNNs are effective as continuous speech encoders and suggests their potential for neuromorphic hardware development, which has so far mostly concentrated on keyword spotting tasks.

5.2.8 Summary of findings

In the introduction, we set three goals for our work on SNNs. After successfully training them in the context of LVCSR, we can conclude that surrogate gradient SNNs are compatible with modern end-to-end, sequence-to-sequence architectures. This answers our first and more general goal of assessing their scalability to more advanced tasks and deeper networks. Secondly, on all tasks, spiking layers, which involve eight times fewer trainable parameters, were shown to be capable of replacing LSTMs with only minor losses in performance. Even though keeping a single LSTM layer still significantly helped reducing the error rate, this demonstrates that the information inside neural networks can efficiently be reduced to sparse and binary events using a physiological approach without considerably affecting the network encoding capabilities. Thirdly, recurrent SNNs proved to be robust to exploding gradient problems without requiring gates, where standard RNNs failed. More generally, these findings contribute not only to creating energy-efficient technology, but also to making SNNs viable deep learning components and promising tools for gaining insights into the neural basis of speech processing in the brain.

5.3 Conclusion

In this chapter, we have demonstrated that the surrogate gradient SNN approach defined in Chapter 4 is compatible with modern deep learning-based ASR frameworks.

A key technical innovation was our implementation of SFA with an adaptation current and stability conditions, as opposed to the previously used moving threshold formulation. This advancement enabled us to first achieve new state-of-the-art results for SNNs in speech command recognition tasks. Furthermore, it allowed us to successfully extend surrogate gradient SNNs to more challenging LVCSR tasks, which involve sequence-to-sequence training with longer sequences and larger datasets. Unlike previous SNN approaches on such tasks, which either discarded precise spike timings in the learning rule or did not use a physiologically plausible neuron model, our method integrated both components and demonstrated robustness against exploding and vanishing gradient issues.

Across both simple speech command and complex LVCSR tasks, our lightweight SNNs not only achieved competitive results compared to RNN baselines but also benefited from the inherent energy efficiency of spiking neurons, underscoring their relevance and potential for future applications in neuromorphic technology.

So far, our architectures have been designed for comparison with ANN baselines. Although our experiments do not directly infer about biological function, they demonstrate that a representational capability found in biological systems can solve the same problems as artificial networks, which are known to surpass human performance on many tasks. This offers a promising hypothesis for better understanding physiological processes using our SNN approach, which is the focus of the next chapter. While this comes at the cost of some recognition performance, our goal is to more closely reproduce physiological processes involved in speech perception. Specifically, this allows us to test whether our approach generates neural oscillations and macro-scale synchronisation phenomena similar to those observed in the brain.

6 Bio-Inspired Speech Encoding and Neural Oscillations

In the previous chapter, we assessed the compatibility and technical plausibility of our SNNs with deep learning frameworks for ASR, comparing their performance to that of non-spiking RNNs. We now shift our focus to the physiology and define a more biologically-inspired speech encoder. After training on ASR tasks, we conduct an analysis of spike trains across the architecture. Our trained networks naturally reproduce neural oscillation phenomena commonly associated with various cognitive processes in the brain. During speech processing, we measure significant forms of cross-frequency couplings of the neural activity both within and across layers. When processing noise, the activity is greatly reduced and no coupling occurs, indicating a natural efficiency of SNNs in adapting their processing power to the input. We also observe the key role of spike-frequency adaptation, recurrent connections and the excitatory-inhibitory properties of neurons in driving these oscillations and regulating the overall activity.

Publication Note

This chapter includes content from the following publication.

- A. Bittar and P. N. Garner (2024). "Exploring neural oscillations during speech perception via surrogate gradient spiking neural networks". In: *Frontiers in Neuroscience* 18. DOI: 10.3389/fnins.2024.1449181

The underlying code is available open-access on GitHub at https://github.com/idiap/sparse

6.1 Introduction

In the field of speech processing technologies, the effectiveness of training deep ANNs with gradient descent has led to the emergence of many successful encoder-decoder architectures for ASR, typically trained in an end-to-end fashion over vast amounts of data (Gulati et al., 2020; Baevski et al., 2020; Li, Pang, et al., 2021; Radford et al., 2023). Despite recent efforts (Brodbeck, Hannagan, and Magnuson, 2024; Millet, Caucheteux, et al., 2022; Millet and King, 2021; Magnuson et al., 2020) towards understanding how these ANN representations can compare with speech processing in the human brain, the cohesive integration of the fields of deep learning and neuroscience remains a challenge.

Nonetheless, biologically inspired SNNs present an interesting convergence point of the two disciplines. Although slightly behind in terms of performance compared to ANNs, we have seen in Chapter 5 that surrogate gradient SNNs were capable of achieving competitive results on speech command recognition and LVCSR tasks. Their successful inclusion into contemporary deep learning ASR frameworks offers a promising path to bridge the existing gap between deep learning and neuroscience in the context of speech processing. This integration not only equips deep learning tools with the capacity to engage in speech neuroscience but also offers a scalable approach to simulate spiking neural dynamics, which supports the exploration and testing of hypotheses concerning the neural mechanisms and cognitive processes related to speech. This investigation of complex brain functions via physiologically inspired networks aligns with the work of Pulvermüller et al. (2021), Henningsen-Schomers and Pulvermüller (2022), and Pulvermüller (2023), who applied biological constraints to large-scale simulations of language learning in SNNs. We complement their approach by training on more realistic speech data, albeit at the cost of some simplifications.

In neuroscience, various neuroimaging techniques such as Electroencephalography (EEG) can detect rhythmic and synchronised postsynaptic potentials that arise from activated neuronal assemblies. These give rise to observable neural oscillations, commonly categorised into distinct frequency bands: delta (0.5-4 Hz), theta (4-8 Hz), alpha (8-13 Hz), beta (13-30 Hz), low-gamma (30-80 Hz), and high-gamma (80-150 Hz) (Buzsaki, 2006). It is worth noting that while these frequency bands provide a useful framework, their boundaries are not rigidly defined and can vary across studies. Nevertheless, neural oscillations play a crucial role in coordinating brain activity and are implicated in cognitive processes such as attention (Fries et al., 2001; Jensen and Colgin, 2007; Womelsdorf and Fries, 2007; Vinck et al., 2013), memory (Kucewicz et al., 2017), sensory perception (Başar et al., 2000; Senkowski et al., 2007) and motor function (MacKay, 1997; Ramos-Murguialday and Birbaumer, 2015). Of particular interest is the phenomenon of Cross-Frequency Coupling (CFC) which reflects the interaction between oscillations occurring in different frequency bands (Jensen and Colgin, 2007; Jirsa and Müller, 2013). As reviewed by Abubaker, Al Qasem, and Kvašňák (2021), many studies have demonstrated a relationship between CFC and working memory performance (Tort, Komorowski, et al.,

2009; Axmacher et al., 2010). In particular Phase-Amplitude Coupling (PAC) between theta and gamma rhythms appears to support memory integration (Buzsáki and Moser, 2013; Backus et al., 2016; Hummos and Nair, 2017), preservation of sequential order (Reddy, Self, et al., 2021; Colgin, 2013; Itskov et al., 2008) and information retrieval (Mizuseki et al., 2009). In contrast, alpha-gamma coupling commonly manifests itself as a sensory suppression mechanism during selective attention (Foxe and Snyder, 2011; Banerjee et al., 2011), inhibiting task-irrelevant brain regions (Jensen and Mazaheri, 2010) and ensuring controlled access to stored knowledge (Klimesch, 2012). Finally, beta oscillations are commonly associated with cognitive control and top-down processing (Engel, Fries, and Singer, 2001).

In the context of speech perception, numerous investigations have revealed a similar oscillatory hierarchy, where the temporal organisation of high-frequency signal amplitudes in the gamma range is orchestrated by low-frequency neural phase dynamics, specifically in the delta and theta ranges (Canolty et al., 2006; Ghitza, 2011; Giraud and Poeppel, 2012; Hyafil et al., 2015; Attaheri et al., 2022). These three temporal scales – delta, theta and gamma – naturally manifest in speech and represent specific perceptual units. In particular, delta-range modulation (1-2 Hz) corresponds to perceptual groupings formed by lexical and phrasal units, encapsulating features such as the intonation contour of an utterance. Modulation within the theta-range aligns with the syllabic rate (4 Hz) around which the acoustic envelope consistently oscillates. Finally, (sub)phonemic attributes, including formant transitions that define the fine structure of speech signals, correlate with higher modulation frequencies (30-50 Hz) within the low-gamma range. The close correspondence between the perception of (sub)phonemic, syllabic and phrasal attributes on one hand, and the manifestation of gamma, theta and delta neural oscillations on the other, was notably emphasised by Giraud and Poeppel (2012). These different levels of temporal granularity inherent to speech signals therefore appear to be processed in a hierarchical fashion, with the intonation and syllabic contour encoded by earlier neurons guiding the encoding of phonemic features by later neurons. Some insights about how phoneme features end up being encoded in the temporal gyrus were given by Mesgarani et al. (2014). Drawing from recent research (Bonhage et al., 2017) on the neural oscillatory patterns associated with the sentence superiority effect, it is suggested that such low-frequency modulation may facilitate automatic linguistic chunking by grouping higher-order features into packets over time, thereby contributing to enhanced sentence retention. The engagement of working memory in manipulating phonological information enables the sequential retention and processing of speech sounds for coherent word and sentence representations. Additionally, alpha modulation has also been shown to play a role in improving auditory selective attention (Strauß, Wöstmann, and Obleser, 2014; Strauß, Kotz, et al., 2014; Wöstmann, Lim, and Obleser, 2017), reflecting the listener's sensitivity to acoustic features and their ability to comprehend speech (Obleser and Weisz, 2012).

Computational models (Hyafil et al., 2015; Hovsepyan, Olasagasti, and Giraud, 2020)

have shown that theta oscillations can indeed parse speech into syllables and provide a reliable reference time frame to improve gamma-based decoding of continuous speech. These approaches (Hyafil et al., 2015; Hovsepyan, Olasagasti, and Giraud, 2020) implement specific models for theta and gamma neurons along with a distinction between inhibitory and excitatory neurons. The resulting networks are then optimised to detect and classify syllables with very limited numbers of trainable parameters (10-20). In contrast, this work proposes to utilise significantly larger end-to-end trainable multi-layered architectures (400k-20M trainable parameters) where all neuron parameters and synaptic connections are optimised to predict sequences of phoneme/subword probabilities, that can subsequently be decoded into words. By avoiding constraints on theta or gamma activity, the approach allows us to explore which forms of CFC naturally arise when solely optimising the decoding performance. Even though the learning mechanism is not biologically plausible, we expect that a model with sufficiently realistic neuronal dynamics and satisfying ASR performance should reveal similarities with the human brain. We divide our analysis in two parts,

1. **Architecture**: As a preliminary analysis, we conduct hyperparameter tuning to optimise the model's architectural parameters. On top of assessing the network's capabilities and scalability, we notably evaluate how the incorporation of SFA and recurrent connections impact the speech recognition performance.

2. **Oscillations**: We then explore the central aspect of our analysis concerning the emergence of neural oscillations within our model. Each SNN layer is treated as a distinct neuron population, from which spike trains are aggregated into a population signal similar to EEG data. Through intra- and inter-layer CFC analysis, we investigate the presence of significant delta-gamma, theta-gamma, alpha-gamma and beta-gamma PAC. We also investigate how incorporating Dale's law (which constrains neurons to be either excitatory or inhibitory, but not both), along with SFA and recurrent connections affects the synchronisation of neural activity.

6.2 Methods

6.2.1 Simulated speech recognition pipeline

Our objective is to design a speech recognition architecture that, while sufficiently plausible for meaningful comparisons with neuroscience observations, remains simple and efficient to ensure compatibility with modern deep learning techniques and achieve good ASR performance. We implement the overall waveform-to-phoneme pipeline illustrated in Fig. 6.1 inside the Speechbrain (Ravanelli, Parcollet, Plantinga, et al., 2021) framework. We provide a description of each of its components here below.

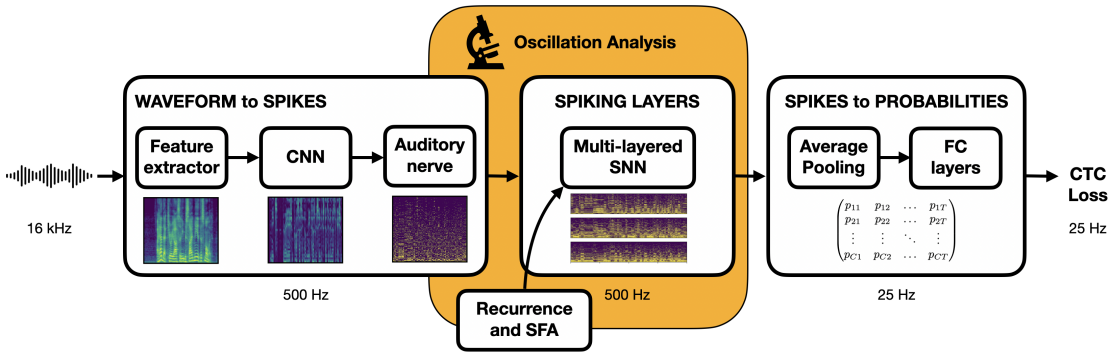

Figure 6.1: End-to-end trainable speech recognition pipeline. Input waveform is converted to a spike train representation to be processed by the central SNN before being transformed into output phoneme probabilities sent to a loss function for training. A sampling rate of 500 Hz is used through the SNN to improve the temporal resolution for our oscillation analysis.

Feature extractor

Mel filterbank features are extracted from the raw waveform using 80 filters and a 25 ms window with a shift of 2 ms. This procedure down samples the 16 kHz input speech signal to a 500 Hz spectrogram representation with 80 frequency bins.

Auditory CNN

A single-layered two-dimensional convolution module is applied to the 80 extracted Mel features using 16 channels, a kernel size of (7, 7), a padding of (7, 0) and a stride of 1, producing $16 \cdot \Big((80-7)+1\Big) = 1184$ output signals with unchanged number of time steps. Layer normalisation, drop out on the channel dimension and a Leaky ReLU activation are then applied. Each produced signal characterises the evolution over time of the spectral energy across a frequency band of 7 consecutive Mel bins.

Auditory nerve fibers

Each 500 Hz output signal from the auditory CNN constitutes the stimulus of a single auditory nerve fiber, which converts the real-valued signal into a spike train. These nerve fibers are modelled as a layer of LIF neurons without recurrent connections and using a single trainable parameter per neuron, $\tau_u \in [3, 25]$ ms, representing the time constant of the membrane potential decay.

Multi-layered SNN

The resulting spike trains are sent to a fully connected multi-layered SNN architecture with 512 neurons in each layer. The proportion of neurons with nonzero adaptation parameters is controlled in each layer so that only a fraction of the neurons are AdLIF and the rest are LIF. Similarly the proportion of nonzero feedforward and recurrent connections is controlled in each layer by applying fixed random binary masks to the weight matrices. Compared to a LIF neuron, an AdLIF neuron has three additional trainable parameters, $\tau_w \in [30, 350]$ ms, $a \in [-0.5, 5]$ and $b \in [0, 2]$, related to the adaptation variable coupled to the membrane potential.

Spikes to probabilities

The spike trains of the last layer are sent to a an average pooling module which down samples their time dimension to 25 Hz. These are then projected to 512 phoneme features using two FC layers with Leaky ReLU activation. A third FC layer with log-softmax activation finally projects them to 40 log-probabilities representing 39 phoneme classes and a blank token as required by CTC.

Training and inference

The log-probabilities are sent to a CTC loss (Graves, Fernández, et al., 2006) so that the parameters of the complete architecture can be updated through back propagation. Additionally, regularisation of the firing rate as defined in Eq. (6.2) is used to prevent neurons from being silent or firing above the Nyquist frequency. At inference, CTC decoding is used to output the most likely phoneme sequence from the predicted log-probabilities, and the PER is computed to evaluate the model's performance. On TIMIT, we use the Adam optimiser (Kingma and Ba, 2015) with an initial learning rate of 0.001 and a batch size of 8 during 50 epochs.

Spike frequency regularisation

The firing rate $f_{b,n}^{l}$ of neuron n in layer l when processing utterance b can be calculated in Hz as,

$$f_{b,n}^{l} = \frac{1}{T_b} \sum_{t=1}^{T} s_{b,t,n}^{l} \,, \tag{6.1}$$

where T is the number of steps and T_b the utterance duration in seconds. We regularise the firing rates of all spiking neurons between $f_{\text{min}} = 0.5$ Hz and $f_{\text{max}} = f_{\text{Nyquist}}$ using the following regularisation loss,

$$\mathcal{L}_{\text{reg}} = \frac{1}{B\,L} \sum_{b=1}^{B} \sum_{l=1}^{L} \frac{1}{N^l} \sum_{n=1}^{N^l} \text{ReLU}\Big(f_{\text{min}} - f_{b,n}^{l}\Big) + \text{ReLU}\Big(f_{b,n}^{l} - f_{\text{max}}\Big) \,, \tag{6.2}$$

to discourage neurons from remaining silent or from firing above the Nyquist frequency.

6.2.2 Physiological plausibility and limitations

After a brief description of the physiology of speech perception throughout the human auditory pathway, we discuss the plausibility and limitations of our model.

Overview of the human auditory pathway

Sound waves are initially received by the outer ear and then transmitted as vibrations to the cochlea in the inner ear, where the basilar membrane allows for a representation of different frequencies along its length (Gundersen, Skarstein, and Sikkeland, 1978). Distinct sound frequencies induce localised membrane vibrations that activate adjacent inner hair cells. These specialised sensory cells, covering the entire basilar membrane, release neurotransmitters when activated, stimulating neighbouring auditory nerve fibers and initiating the production of action potentials. Tonotopy is maintained through the conversion of mechanical motion into electric signals as each inner hair cell, tuned to a specific frequency, only affects nearby auditory nerve fibers (Saenz and Langers, 2014). The resulting spike trains then propagate through a multi-layered neural network, ultimately reaching cortical regions associated with higher-order cognitive functions such as speech recognition. Overall, the auditory system is organised hierarchically, with each level contributing to the progressively more sophisticated processing of auditory information.

Cochlea and inner hair cells

While some of the complex biological processes involved in converting mechanical vibrations to electric neuron stimuli can be abstracted, we assume that the key feature

to retain is the tonotopic encoding of sound information. A commonly used metric in neuroscience is the ratio of characteristic frequency to bandwidth, which defines how sharply tuned a neuron is around the frequency it is most responsive to. As detailed in Table 6.1, from measured Q10 values in normal hearing humans reported by Devi et al. (2022), we evaluate that a single auditory nerve fiber should receive inputs from 5-7 adjacent frequency bins when using 80 Mel filterbank features. The adoption of a Mel filterbank frontend can be justified by its widespread utilisation within deep learning ASR frameworks. Although we do not attempt to directly model cochlear and hair cell processing, we can provide a rough analogue in the form of Mel features passing through a trainable convolution module that yields plausible ranges of frequency sensitivity for our auditory nerve fibers.

Table 6.1: Sensitivity to frequency of auditory nerve fibers. The second column gives the sensitivity ranges measured by Devi et al. (2022) that we compute from their reported mean Q10 values in the normal hearing group. The third column gives the Mel scale centers using 80 filters that surround the corresponding sensitivity ranges.

Characteristic Frequency [Hz]	Sensitivity range [Hz]	Nearby Mel bin centers [Hz]	Overlapping Number of bins
500	416-583	416, 452, 488, 525, 564, 604	5-6
1000	881-1119	872, 921, 973, 1026, 1080, 1136	5-6
2000	1770-2230	1729, 1806, 1886, 1967, 2052, 2139, 2228	6-7
4000	3566-4434	3553, 3688, 3827, 3970, 4117, 4270, 4427	7
6000	5501-6499	5479, 5674, 5875, 6083, 6297, 6519	5-6

Simulation time step

Modern ASR systems (Gulati et al., 2020; Radford et al., 2023) typically use a frame period of $\Delta t = 10$ ms during feature extraction, which is then often sub-sampled to 40 ms using a CNN before entering the encoder-decoder architecture. In the brain, typical minimal inter-spike distances imposed by a neuron's absolute refractory period can vary from 0 to 5 ms (Gerstner and Kistler, 2002). We therefore assume that using a time step greater than 5 ms could result in dynamics that are less representative of biological phenomena. Although using a time step $\Delta t < 1$ ms may yield biologically more realistic simulations, we opt for time steps ranging from 1 to 5 ms to ensure computational efficiency. After the SNN, the spike trains of the last layer are down-sampled to 25 Hz via average pooling on the time dimension. This prevents an excessive number of time steps from entering the CTC loss, which could potentially hinder its decoding efficacy. We use $\Delta t = 5$ ms for most of the hyperparameter tuning to reduce training time, but favour $\Delta t = 2$ ms for the oscillation analysis so that the full gamma range of interest (30-150 Hz) remains below the Nyquist frequency at 250 Hz.

Neuron model

The LIF neuron model is an effective choice for modelling auditory nerve fibers as it accurately represents their primary function of encoding sensory inputs into spike trains. We avoid using SFA and recurrent connections, as they are not prevalent characteristics of nerve fibers.

On the other hand, for the multi-layered SNN, the linear AdLIF neuron model with layer-wise recurrent connections stands out as a good compromise between accurately reproducing biological firing patterns and remaining computationally efficient (Bittar and Garner, 2022b; Deckers et al., 2024). Although less popular than the moving threshold formulation by Bellec et al. (2018), recently reviewed in Ganguly et al. (2024), our implementation of SFA using the AdLIF model combines spike-triggered adaptation with subthreshold coupling. In Section 5.1, we demonstrated that the AdLIF outperforms moving threshold implementations (Yin, Corradi, and Bohté, 2021; Salaj et al., 2021; Shaban, Bezugam, and Suri, 2021; Yin, Corradi, and Bohté, 2020) in speech command recognition tasks. Nevertheless, we will still implement and train an additional model with moving threshold SFA to ensure that our conclusions hold consistently across different SFA models.

Organisation in layers

Similarly to ANNs, our simulation incorporates a layered organisation, which facilitates the progressive extraction and representation of features from low-order to higher-order, without the need of concretely defining and distinguishing neuron populations. This fundamental architectural principle aligns with the general hierarchical processing observed in biological brains. However, it oversimplifies the complexities of auditory processing, which extends beyond a straightforward sequential framework. While there is some sort of sequential processing in sub-cortical structures, the levels of processed features are more intricate than a simple hierarchy. This simplification is made to ensure compatibility with deep learning frameworks.

Layer-wise recurrence

While biological efferent pathways in the brain involve complex and widespread connections that span across layers and regions, modelling such intricate connectivity can introduce computational challenges and complexity, potentially hindering training and scalability. By restricting feedback connections to layer-wise recurrence, we simplify the network architecture and enhance compatibility with deep learning frameworks.

Excitatory and inhibitory

In the neuroscience field, neurons are commonly categorised into two types: excitatory neurons, which stimulate action potentials in postsynaptic neurons, and inhibitory neurons, which reduce the likelihood of spike production in postsynaptic neurons. This bio-inspired distinction is referred to as Dale's law.

In ANNs, weight matrices are commonly initialised with zero mean and a symmetric distribution, so that the initial number of excitatory and inhibitory connections is balanced. During training, synaptic connections are updated across all layers without enforcing a distinction between excitatory and inhibitory neurons. Dale's law can nevertheless be imposed (Li, Cornford, et al., 2024; Cornford et al., 2021) even if it typically results in slightly reduced performance.

In our baseline model, Dale's law is not applied, so that similarly to standard ANNs, weight matrices are trained without constraining values to be positive or negative. Additionally, we train separate SNNs with Dale's law to evaluate its impact on neural oscillations. In this setup, half the neurons are excitatory and half are inhibitory, while the auditory nerve fibers are all excitatory.

Delays

In biological neural networks, the propagation time of spikes between neurons introduces delays, primarily due to axonal transmission. To incorporate and assess the impact of these delays on neural oscillations, we additionally train separate SNNs using dilated convolutions in the temporal dimension instead of fully connected feedforward matrices. This approach, introduced by Hammouamri, Khalfaoui-Hassani, and Masquelier (2023), allows us to introduce controlled delays directly into the network architecture. Based on their configuration for speech command recognition tasks, we use a maximum delay value of 300 ms.

Learning rule

SGD, though biologically implausible due to its global and offline learning framework, allows us to leverage parallelisable and fast computations to optimise larger-scale neural networks. While this approach facilitates effective training and scaling, it diverges from biologically inspired synaptic plasticity mechanisms, such as those mediated by AMPA and NMDA receptors.

Decoding into phoneme sequences

Although lower PERs could be achieved with a more sophisticated decoder, our primary focus is on analysing the spiking layers within the encoder. For simplicity, we therefore opt for straightforward CTC decoding, which more directly reflects the encoder's capabilities.

Hybrid ANN-SNN balance

The CNN module in the ASR frontend as well as the ANN module (average pooling and FC layers) converting spikes to probabilities are intentionally kept simple to give most of the processing and representational power to the central SNN on which focuses our neural oscillations analysis.

6.2.3 Speech processing tasks

The following non-spiking datasets, detailed in Section 2.4, are used in our study.

- The TIMIT (Garofolo et al., 1993) dataset, due to its compact size of approximately five hours of speech data, is well-suited for investigating suitable model architectures and tuning hyperparameters. It is however considered small for ASR hence the use of LibriSpeech presented below.
- The LibriSpeech (Panayotov et al., 2015) corpus contains about 1,000 hours of English audiobook recordings, allowing us to validate our findings on a significantly larger scale compared to TIMIT. Given the extended training duration, we train only a limited number of models to confirm that our analysis holds when applied to larger datasets.
- The Google Speech Commands dataset (Warden, 2018) is used to test whether similar CFCs arise when simply recognising single one-second words instead of phoneme or subword sequences.

Using the method described in Section 2.1.3 to compute credible intervals, we get error bars of $\pm 0.8\%$ for the reported PERs on the TIMIT dataset, about $\pm 0.2\%$ for the reported word error rates on LibriSpeech, and about $\pm 0.4\%$ for the reported accuracy on Google Speech Commands.

6.2.4 Analysis methods

Hyperparameter tuning

Before reporting results on the oscillation analysis, we investigate the optimal architecture by tuning some relevant hyperparameters. All experiments are run using a single NVIDIA GeForce RTX 3090 GPU. On top of assessing their respective impact on the error rate, we

test if more physiologically plausible design choices correlate with better performance. Here is the list of the fixed parameters that we do not modify in our reported experiments:

- number of Mel bins: 80
- Mel window size: 25 ms
- auditory CNN kernel size (7, 7)
- auditory CNN stride: (1, 1)
- auditory CNN padding: (7, 0)
- average pooling size: 40 / Δt
- number of phoneme FC layers: 2
- number of phoneme features: 512
- dropout: 0.15
- activation: Leaky ReLU

where for CNN attributes of the form (n_t, n_f), n_t and n_f correspond to time and feature dimensions respectively. The tunable parameters are the following,

- filter bank window shift controlling the SNN time step in ms: {1, 2, 5}
- number of auditory CNN channels (filters): {8, 16, 32, 64, 128}
- number of SNN layers: {1, 2, 3, 4, 5, 6, 7}
- neurons per SNN layer: {64, 128, 256, 512, 768, 1024, 1536, 2048}
- proportion of neurons with SFA: [0, 1]
- feedforward connectivity: [0, 1]
- recurrent connectivity: [0, 1]

While increasing the number of neurons per layer primarily impacts memory requirements, additional layers mostly extend training time.

Population signal

In the neuroscience field, EEG stands out as a widely employed and versatile method for studying brain activity. By placing electrodes on the scalp, this non-invasive technique measures the aggregate electrical activity resulting from the synchronised firing of neurons within a specific brain region. An EEG signal therefore reflects the summation of postsynaptic potentials from a large number of neurons operating in synchrony. The typical sampling rate for EEG data is commonly in the range of 250 to 1000 Hz which matches our desired simulation time steps. With our SNN, we do not have EEG signals but directly the individual spike trains of all neurons in the architecture. In order to perform similar population-level analyses, we sum the binary spike trains $s_b^l \in \{0,1\}^{T \times N^l}$

emitted by all neurons in a specific layer l for a single utterance b as follows,

$$p_{b,t}^{l} = \sum_{n=1}^{N^{l}} s_{b,t,n}^{l} \,. \tag{6.3}$$

Before performing the PAC analysis, the resulting population activity signal $p_{b,t}^{l}$ is then normalised over the time dimension with a mean of 0 and a standard deviation of 1, yielding the normalised population signal $\hat{p}_{b,t}^{l}$ defined as,

$$\hat{p}_{b,t}^{l} = \frac{p_{b,t}^{l} - \mu_{b}^{l}}{\sigma_{b}^{l}} \,, \tag{6.4}$$

where μ_{b}^{l} is the mean and σ_{b}^{l} is the standard deviation of $p_{b,t}^{l}$ over the time dimension.

Phase-amplitude coupling

Using finite impulse response band-pass filters, the obtained population signals are decomposed into different frequency ranges. We study CFC in the form of PAC both within a single population and across layers. This technique assesses whether a relationship exists between the phase of a low frequency signal and the envelope (amplitude) of a high frequency signal.

As recommended in Hülsemann, Naumann, and Rasch (2019), we implement both the modulation index (Tort, Kramer, et al., 2008) and mean vector length (Canolty et al., 2006) metrics to quantify the observed amount of PAC. For each measure type, the observed coupling value is compared to a distribution of 10,000 surrogates to assess the significance. Surrogate couplings are computed by disrupting the temporal order of the amplitude time series while preserving its overall characteristics. Specifically, the amplitude time series is permuted by cutting it at a random point and reversing the order of the two segments. This method, as discussed by Hülsemann, Naumann, and Rasch (2019), maintains all inherent properties of the original data except for the temporal relationship between phase angle and amplitude magnitude, providing a conservative test of significance. A p-value is then obtained by fitting a Gaussian function on the distribution of surrogate coupling values and calculating the area under the curve for values greater than the observed coupling value.

As pointed out by Jones (2016), it is important to note that observed oscillations can exhibit complexities such as non-sinusoidal features and brief transient events on single trials. Such nuances become aggregated when averaging signals, leading to the widely observed continuous rhythms. We therefore perform all analysis on single utterances.

For intra-layer interactions, a single population signal is used to extract both the low-frequency oscillation phase and the high-frequency oscillation amplitude. In a three-

layered architecture, these interactions include nerve-nerve, first layer-first layer, second layer-second layer, and third layer-third layer couplings.

For inter-layer interactions, we consider couplings between the low-frequency oscillation phase in one layer and the high-frequency oscillation amplitude in all subsequent layers. These interactions include nerve-first layer, nerve-second layer, nerve-third layer, first layer-second layer, first layer-third layer, second layer-third layer couplings.

For all aforementioned intra- and inter-layer combinations, we use delta (0.5-4 Hz), theta (4-8 Hz), alpha (8-13 Hz) and beta (13-30 Hz) ranges as low-frequency modulating bands, and low-gamma (30-80 Hz) and high-gamma (80-150 Hz) ranges as high-frequency modulated bands. For a given model, we iterate through the 64 longest utterances in the TIMIT test set. For each utterance, we consider the 10 aforementioned intra- and inter-layer relations, as well as the 8 possible combinations of low-frequency to high-frequency bands. We conduct PAC testing on each of the 5,120 resulting coupling scenarios, and only consider a coupling to be significant when both modulation index and mean vector length metrics yield p-values below 0.05.

6.3 Results

6.3.1 Architectural analysis

In order to draw a comparison with the human auditory pathway, we have introduced the physiologically inspired ASR pipeline illustrated in Fig. 6.1. The proposed hybrid ANN-SNN architecture is trained in an end-to-end fashion on the TIMIT dataset (Garofolo et al., 1993) to predict phoneme sequences from speech waveforms. In the architectural design, we aimed to minimise the complexity of ANN components and favour the central SNN which will be the focus of the oscillation analysis. Here as a preliminary step, we examine how relevant hyperparameters affect the PER. On top of assessing the scalability of our approach to larger networks, we identify the importance of the interplay between recurrence and SFA.

Table 6.2: Hyperparameter tuning for the number of SNN layers and neurons per layer on the TIMIT dataset. The third column gives both the number of trainable parameters in the multi-layered SNN (left) and in the whole encoder (right). The PERs are reported after 50 training epochs using a 5 ms time step, 16 CNN channels, 50% of AdLIF neurons, 100% feedforward and 50% recurrent connectivity. The performance of the architecture when removing the SNN is also reported (bottom). Bold values indicate the lowest achieved PERs.

Number of layers	Neurons per layer	Number of parameters	Test PER [%]	Validation PER [%]
1	512	740k / 1.3M	23.3	21.8
2	512	1.1M / 1.7M	21.0	19.2
3	512	1.5M / 2.1M	20.5	18.2
4	512	1.9M / 2.5M	20.2	**17.4**
5	512	2.3M / 2.9M	**20.0**	17.6
6	512	2.7M / 3.3M	**20.0**	17.9
7	512	3.1M / 3.7M	20.5	18.0
3	64	91k / 394k	30.9	29.6
3	128	211k / 547k	25.5	24.1
3	256	537k / 938k	22.5	20.9
3	768	3.0M / 3.7M	19.6	17.4
3	1024	4.9M / 5.7M	19.1	**17.1**
3	1536	10.1M / 11.2M	**19.0**	17.3
3	2048	17.1M / 18.5M	19.2	17.2
no nerve, no SNN		0 / 873k	34.2	32.0

Network scalability

As reported in Table 6.2, performance improves with added layers, peaking at 4-6 layers before declining, which suggests a significant contribution to the final representation from all layers within this range. Compared to conventional non-spiking RNN encoders used in ASR, our results support the scalability of surrogate gradient SNNs to relatively deep architectures. Additionally, augmenting the number of neurons until about 1,000 per layer consistently yields lower PERs, beyond which performance saturates.

Recurrent connections and spike-frequency adaptation

The impact of adding SFA in the neuron model as well as using recurrent connections are reported in Table 6.3. Interestingly, we find that without SFA, optimal performance is achieved by limiting the recurrent connectivity to 80%. When additionally using SFA, further limitation of the recurrent connectivity about 50% yields the lowest PER. This differs from conventional non-spiking RNNs, where employing FC recurrent matrices is favoured. These results indicate that while requiring fewer parameters, an architecture with sparser recurrent connectivity and more selective parameter usage can achieve better task performance.

Table 6.3: Ablation experiments for the recurrent connectivity and proportion of neurons with SFA on the TIMIT dataset. PERs are reported after 50 epochs using a 5 ms time step, 16 CNN channels, 3 layers, 512 neurons per layer and 100% feedforward connectivity. Bold values indicate the lowest achieved PERs.

Model type	Recurrent connectivity	Proportion of AdLIF neurons	Number of parameters	Test PER [%]	Validation PER [%]
No recurrence no SFA	0	0	1.1M / 1.7M	26.9	24.8
Recurrence only	0.2	0	1.3M / 1.8M	22.0	20.1
	0.5	0	1.5M / 2.1M	21.0	18.9
	0.8	0	1.8M / 2.3M	20.8	18.7
	1	0	1.9M / 2.5M	21.8	19.3
SFA only	0	0.2	1.1M / 1.7M	24.2	21.7
	0	0.5	1.1M / 1.7M	23.7	21.6
	0	0.8	1.1M / 1.7M	23.3	21.0
	0	1	1.1M / 1.7M	22.9	21.1
Recurrence and SFA	0.2	0.2	1.3M / 1.8M	20.9	19.3
	0.5	0.5	1.5M / 2.1M	**20.5**	**18.2**
	0.8	0.8	1.8M / 2.3M	21.2	18.8
	1	1	1.9M / 2.5M	23.3	21.5

Overall, SFA and recurrent connections individually yield significant error rate reduction, although they respectively grow as $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ with the number of neurons N. In line with previous studies on speech command recognition tasks (Perez-Nieves et al., 2021) and our findings in Section 5.1.3, our results emphasise the metabolic and computational efficiency gained by harnessing the heterogeneity of adaptive spiking neurons. Furthermore, effectively calibrating the interplay between unit-wise feedback from SFA and layer-wise feedback from recurrent connections appears crucial for achieving optimal performance.

Enforcing Dale's law

To align with common ANN practice, the previous results were obtained without restricting neurons to be either strictly excitatory or strictly inhibitory. We now train more physiologically inspired models that satisfy Dale's law, with results presented in Table 6.4. Although ASR performance decreased (1-4% absolute PER increase), this may simply be due to suboptimal weight initialisation, which is known to affect performance (Li, Cornford, et al., 2024). This could likely be mitigated in future work by using the approach of Rossbroich, Gygax, and Zenke (2022), who derived fluctuation-driven initialisation schemes compatible with Dale's law.

Table 6.4: Ablation experiments for the recurrent connectivity and proportion of neurons with SFA on the TIMIT dataset when additionally using Dale's law. PERs are reported after 50 epochs using a 2 ms time step, 16 CNN channels, 3 layers, 512 neurons per layer, 100% feedforward connectivity and an excitatory-inhibitory ratio of 1. Bold values indicate the lowest achieved PERs.

Model configuration	Recurrent connectivity	Proportion of AdLIF neurons	Number of parameters	Test PER [%]	Validation PER [%]
No recurrence no SFA	0	0	1.1M / 1.7M	30.7	28.7
Recurrence only	0.5	0	1.5M / 2.1M	23.6	20.6
SFA only	0	0.5	1.1M / 1.7M	25.1	22.9
Recurrence and SFA	0.5	0.5	1.5M / 2.1M	**21.2**	**19.2**

Supplementary findings

We observe in Table 6.5 that decreasing the simulation time step does not affect the performance. Although making the simulation of spiking dynamics more realistic, one might have anticipated that backpropagating through more time steps could hinder the training and worsen the performance as observed in standard RNNs often suffering from vanishing or exploding gradients. With inputs ranging from 1,000 to over 7,000 steps using 1 ms intervals on TIMIT, our results demonstrate a promising scalability of surrogate gradient SNNs for processing longer sequences.

Table 6.5: Hyperparameter tuning for the simulation time step on the TIMIT dataset. PERs are reported after 50 epochs using 16 CNN channels, 3 layers of 512 neurons each, 50% of AdLIF neurons, 100% feedforward and 50% recurrent connectivity.

Time step [ms]	Epoch duration [min]	Test PER [%]	Validation PER [%]
5	21	20.5	18.2
2	53	20.4	18.7
1	156	20.6	18.2

Secondly, as reported in Table 6.6, we did not observe substantial improvement when increasing the number of auditory nerve fibers past $\sim$5,000, even though there are approximately 30,000 of them in the human auditory system. This could be due to both the absence of a proper model for cochlear and hair cell processing in our pipeline and to the relatively low number of neurons ($<$1,000) in the subsequent layer.

Table 6.6: Hyperparameter tuning for the number of CNN channels on TIMIT. PERs are reported after 50 epochs using a 5 ms time step, 3 layers, 512 neurons per layer, 50% of AdLIF neurons, 100% feedforward and 50% recurrent connectivity. Bold values indicate the lowest achieved PERs. The third column gives the number of trainable parameters in the complete encoder.

CNN channels	Nerve fibers	Parameters	Test PER [%]	Validation PER [%]
8	592	1.8M	20.9	18.9
16	1,184	2.1M	20.5	18.2
32	2,368	2.7M	20.2	18.4
64	4,736	3.9M	**19.8**	**18.0**
128	9,472	6.4M	21.3	18.9

Table 6.7: Hyperparameter tuning for the feedforward connectivity on the TIMIT dataset. PERs are reported after 50 epochs using a 5 ms time step, 16 CNN channels, 3 layers of 512 neurons each, 50% of AdLIF neurons and 50% recurrent connectivity. Bold values indicate the lowest achieved PERs. The second column gives both the number of trainable parameters in the multi-layered SNN (left) and in the whole encoder (right).

Feedforward connectivity	Parameters	Test PER [%]	Validation PER [%]
0.2	629k / 1.2M	22.0	19.6
0.5	968k / 1.5M	21.6	19.7
0.8	1.3M / 1.8M	20.7	18.8
1.0	1.5M / 2.1M	**20.5**	**18.2**

As detailed in Table 6.7, reduced feedforward connectivity in the SNN led to poorer overall performance. This contrasts with our earlier findings on recurrent connectivity, highlighting the distinct functional roles of feedforward and feedback mechanisms in the network.

Additionally, we incorporated trainable delays by replacing the fully connected feedforward matrices with dilated convolutions over the temporal dimension. While including delays resulted in similar overall performance, using the groups parameter to control connectivity led to more parameter-efficient models, as shown in Table 6.8. This method of reducing connectivity proved more effective than our previous approach of randomly masking fully connected matrices.

Finally, Table 6.9 shows our results when using the more popular moving threshold formulation of SFA instead of the AdLIF model. Consistent with our previous findings on speech command recognition detailed in Section 5.1, the AdLIF implementation of SFA outperforms its moving threshold alternative, with the same number of parameters.

Table 6.8: Hyperparameter tuning for trainable delays on the TIMIT dataset. PERs are reported after 50 epochs using a 5 ms time step, 16 CNN channels, 3 layers of 512 neurons. Bold values indicate the lowest achieved PERs.

Model configuration	Conv Groups	Parameters	Test PER [%]	Validation PER [%]
No recurrence no SFA	1	2.3M / 2.8M	26.2	24.6
Recurrence only	1	2.7M / 3.2M	21.6	19.5
SFA only	1	2.3M / 2.8M	23.3	20.7
Recurrence and SFA	1	2.7M / 3.2M	**20.8**	18.6
Recurrence and SFA	4	968k / 1.5M	21.0	**18.4**

Table 6.9: Results with moving threshold SFA on the TIMIT dataset. PERs are reported after 50 epochs using a 2 ms time step, 16 CNN channels, 3 layers of 512 neurons.

Model configuration	Parameters	Test PER [%]	Validation PER [%]
SFA only	1.1 / 1.7M	26.3	23.9
Recurrence and SFA	1.5 / 2.1M	22.1	19.2

6.3.2 Oscillation analysis

Based on our previous architectural results that achieved satisfactory speech recognition performance using a physiologically inspired model, we hypothesise that the spiking dynamics of a trained network should, to some extent, replicate those occurring throughout the auditory pathway. Our investigation aims to discern if synchronisation phenomena resembling brain rhythms manifest within the implemented SNNs as they processes speech utterances to recognise phonemes.

Synchronised gamma activity produces low-frequency rhythms

As illustrated in Fig. 6.2, the spike trains produced by passing a test-set speech utterance through the trained architecture exhibit distinct low-frequency rhythmic features in all layers. By looking at the histogram of single neuron firing rates illustrated in Fig. 6.3, we observe that the distribution predominantly peaks at gamma range, with little to no activity below beta.

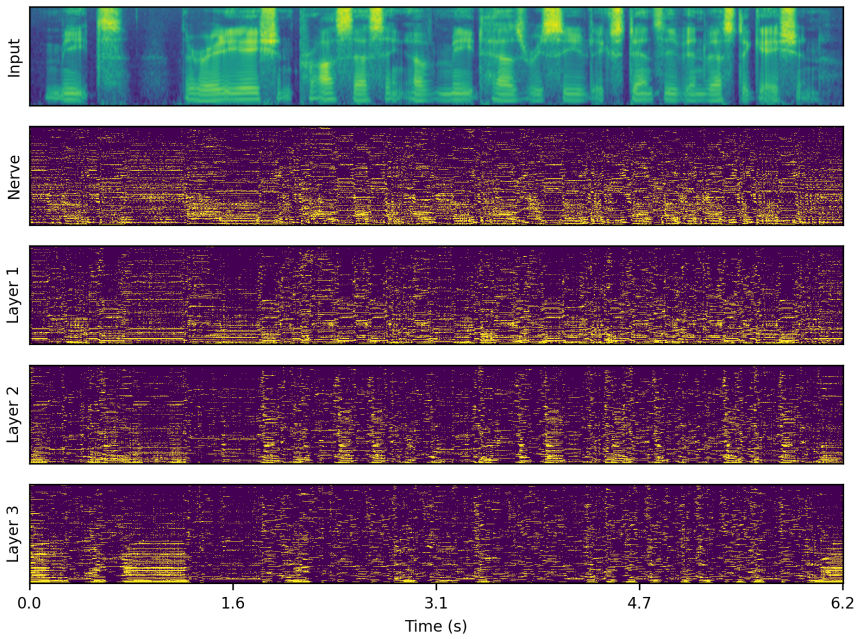

Figure 6.2: Spiking activity in response to speech input. Input filterbank features and resulting spike trains produced across layers. For each layer, the neurons are vertically sorted on the y-axis by increasing average firing rate (top to bottom). The model uses a 2 ms time step, 16 CNN channels, 3 layers of size 512, 50% AdLIF neurons, 100% feedforward and 50% recurrent connectivity with Dale's law.

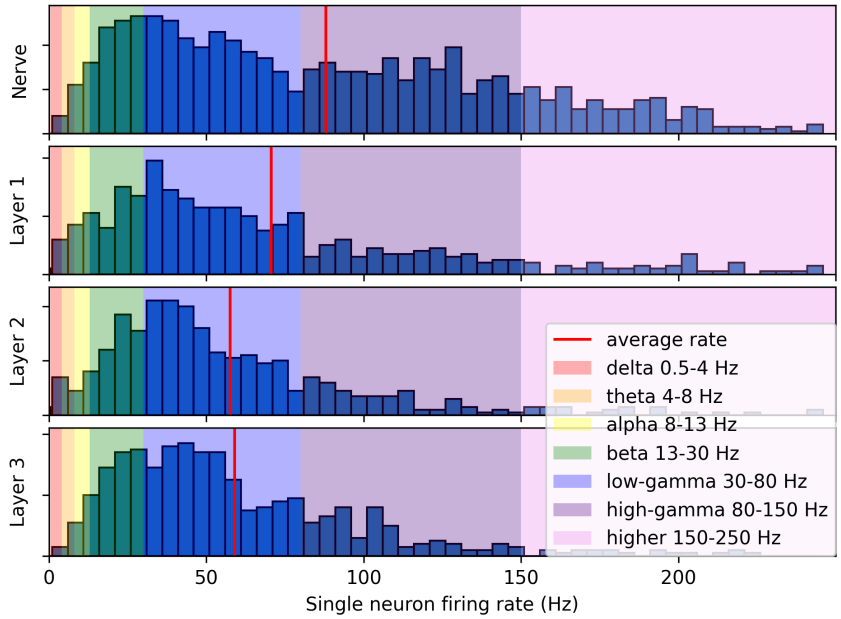

Figure 6.3: Distribution of single neuron firing rates in response to speech input using the same model and utterance as in Fig. 6.2.

This reveals that the low-frequency oscillations visible in Fig. 6.2 actually emerge from the synchronisation of gamma-active neurons. The resulting low-frequency rhythms appear to follow to some degree the intonation and syllabic contours of the input filterbank features and to persist across layers.

Compared to the three subsequent layers, higher activity in the auditory nerve comes from the absence of inhibitory SFA and recurrence mechanisms.

These initial observations suggest that the representation of higher-order features in the last layer is temporally modulated by lower level features already encoded in the auditory nerve fibers, even though each layer is seen to exhibit distinct rhythmic patterns. In the next section, we focus on measuring this modulation more rigorously via PAC analysis.

Phase-amplitude coupling within and across layers

By aggregating over the relevant spike trains as detailed in Section 6.2.4, we compute distinct EEG-like population signals for the auditory nerve fibers and each of the three SNN layers. These are then filtered in the different frequency bands, as illustrated in Fig. 6.4, which allows us to perform CFC analyses.

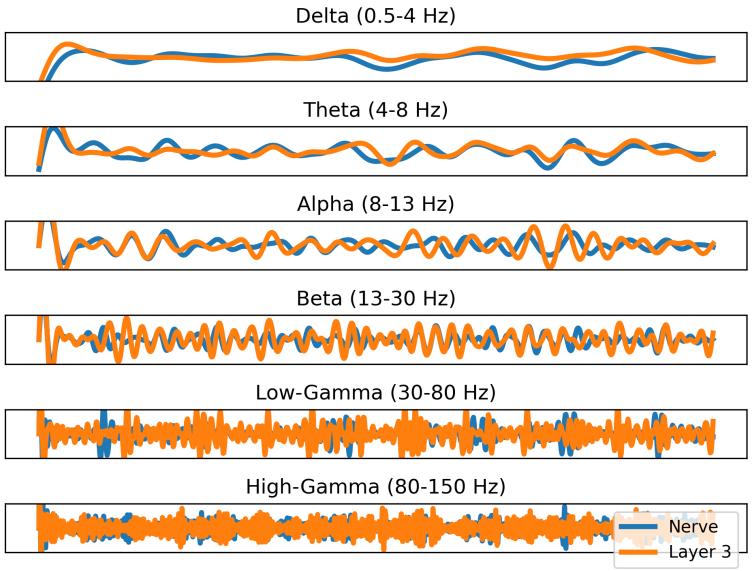

Figure 6.4: Population signals of auditory nerve fibers (blue) and last layer (orange) filtered in distinct frequency bands.

We measure PAC within-layer and across-layers between all possible combinations of frequency bands and find multiple significant forms of coupling for every utterance. An example of significant theta low-gamma coupling between the auditory nerve fibers and the last layer is illustrated in Fig. 6.5. Here input low-frequency modulation is observed to significantly modulate the output gamma activity. This indicates that the network integrates and propagates intonation and syllabic contours across layers through synchronised neural activity along these perceptual cues.

On the majority of utterances, we found significant CFCs between the input waveform and the population signal of the last layer. It is important to note that the synchronisation of neural signals to the auditory envelope emerged without imposing any theta or gamma activity in our network. The optimisation of the PER combined with sufficiently realistic spiking neuronal dynamics therefore represent sufficient conditions to reproduce some broad properties of neural oscillations observed in the brain, suggesting a general functional role of facilitating information processing.

The activity of the final layer of the SNN stands out as the most significantly modulated overall. By architectural construction, modulation in that final layer has the greatest impact on the ASR task, as the spike trains from this layer are converted to phoneme probabilities using a small ANN. A higher number of couplings in the final layer correlates with a decrease in the PER. This suggests that CFCs may be associated with more selective integration of phonetic features, enhanced attentional processes, as well as improved assimilation of contextual information.

Alpha-band oscillations were the most frequently measured, consistent with biological evidence that the alpha rhythm is the most prominent oscillation in spontaneous EEG (Berger, 1929).

More generally, the patterns of coupling between different neural populations and frequency bands were found to differ from one utterance to another. These variations indicate that the neural processing of our network is highly dynamic and depends on the acoustic properties and linguistic content of the input. The observed rich range of intra- and inter-layer couplings suggests that the propagation of low-level input features such as intonation and syllabic rhythm is only one aspect of these synchronised neural dynamics.

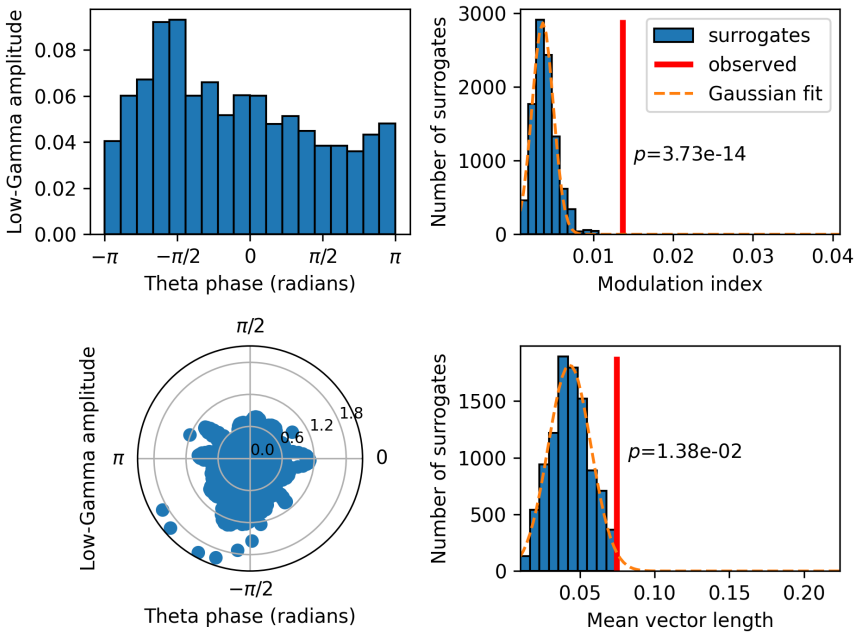

Figure 6.5: Cross-frequency coupling of population aggregated activity. Modulation index and mean vector length metrics as measures of PAC between the theta band of the auditory nerve fibers and the low-gamma band of the last layer. In the bottom-left graph, the amplitude of the low-gamma band is represented by the radius.

Impact of Dale's law on oscillations

As illustrated in Table 6.10, SNNs that satisfy Dale's law display significantly higher numbers of CFCs. In biological neural networks, this principle contributes to a more structured and organised form of network dynamics, as each neuron is either excitatory or inhibitory but not both. When applying Dale's principle to SNNs, it constrains the network with more defined roles for neurons, leading to more coherent oscillatory patterns and more CFCs.

In this study, we use equivalent models for both excitatory and inhibitory neurons, and simply allow them to adapt their parameters within the same fixed ranges during training. We observe that trained values of both membrane and adaptation time constants τ_u and τ_w converge to lower average values across the inhibitory populations compared to the excitatory ones. This suggests that, even with initially equivalent models, the network naturally differentiates the dynamics of excitatory and inhibitory neurons to fulfill their distinct functional roles. Future work could focus on better defining these two types of neurons from the outset, incorporating more biologically plausible initial parameter ranges.

When using Dale's law, the fixed excitatory-inhibitory (E/I) ratio plays a crucial role in shaping the neuronal dynamics inside the SNN. In our study, we observed that increasing the proportion of inhibitory neurons (E/I = 0.33) resulted in a similar ASR performance ($\sim 1\%$ absolute PER difference) compared to the standard ratio (E/I = 1), but with a reduced average firing rate of 56 Hz instead of 68 Hz for equivalent models with 50% of neurons with SFA and 50% of recurrent connectivity (see Table 6.10). This finding suggests that a higher ratio of inhibitory neurons can achieve comparable task performance while maintaining a lower overall level of network activity.

Table 6.10: Effect of Dale's law, SFA, recurrence and delays on oscillatory activity. Comparison of the oscillatory activity resulting from passing the 64 longest TIMIT test-set utterances through different types of trained networks. The last two columns show the total number of significant PACs summed across all 64 utterances and frequency bands, for intra- and inter-layer relations respectively. All networks use a 2 ms time step, 16 CNN channels, 3 layers, 512 neurons per layer and 100% feedforward connectivity (i.e., groups=1 for delays). The E/I ratio is 1 for models with Dale's law, except the last one where it is 0.33.

Model type	**Model configuration**	**Test PER** [%]	**Firing rate** [Hz]	**Number of intra-layer PACs**	**Number of inter-layer PACs**
AdLIF baseline	No recurrence no SFA	26.9	98	133	192
	SFA only	23.7	72	213	390
	Recurrence only	21.0	76	180	132
	Recurrence and SFA	20.5	71	265	177
AdLIF with Dale	No recurrence no SFA	30.7	100	513	643
	SFA only	25.1	77	564	603
	Recurrence only	23.6	67	432	278
	Recurrence and SFA	21.2	68	329	290
	Recurrence and SFA, E/I=0.33	22.3	56	526	496
AdLIF with delays	No recurrence no SFA	26.2	92	116	54
	SFA only	23.3	77	91	53
	Recurrence only	21.6	68	205	66
	Recurrence and SFA	20.8	68	253	60
Moving threshold	SFA only	26.3	56	136	131
	Recurrence and SFA	22.1	62	182	135

Impact of spike-frequency adaptation and recurrent connections on oscillations

Across all model types (AdLIF or moving threshold SFA, with or without delays and Dale's law), both SFA and recurrent connections had an overall inhibitory effect, typically reducing the average network firing rate from around 100 Hz to roughly 60 Hz (see Table 6.10).

This regularisation of the network activity appears to enable more effective parsing and encoding of speech information, as it reliably led to improved ASR performance. While both forms of feedback exhibit an overall inhibitory effect, SFA operates at the individual neuron level whereas recurrent connections act at the layer level.

SFA is known to encourage and stabilise the synchronisation of cortical networks (Crook, Ermentrout, and Bower, 1998) and to promote periodic signal propagation (Augustin, Ladenbauer, and Obermayer, 2013). In our results, this effect is pronounced in SNNs without Dale's law, where the inclusion of SFA was consistently associated with a significant increase in the number of measured CFCs. However, when Dale's law is applied, the overall number of CFCs is significantly higher with no noticeable impact of SFA on CFCs. This suggests that the stricter constraints imposed by Dale's law may lead to more uniform behaviour in CFCs, thereby reducing the observed influence of SFA.

In models with and without Dale's law, incorporating recurrent connections was consistently associated with a decrease in the number of inter-layer couplings, indicating more localised synchronisation.

Finally, using the moving threshold formulation of SFA produces a lower firing rate and fewer significant PACs compared to our AdLIF baseline. The lower number of PACs might mean that the moving threshold formulation is less effective than the AdLIF at coordinating these interactions, which could explain the inferior task performance. Nevertheless, the discrepancy between the two approaches could potentially be narrowed with further hyperparameter optimisation.

Impact of delays on oscillations

As illustrated in Table 6.10, SNNs with delays produced significantly fewer PACs, especially for inter-layer couplings, compared to the baseline with fully connected feedforward matrices. Introducing trainable delays through dilated convolutions allows for temporal dispersion of signals, which may desynchronise neural activity and explain the observed reduction in CFCs.

Effects of training and input type on neuronal activity

In order to further understand the emergence of coupled signals, we consider passing speech through an untrained network, as well as passing different types of noise inputs through a trained network.

Trained architectures exhibit persisting neuronal activity across layers compared to untrained ones, where the activity almost completely decays after the first layer, as illustrated in Figure 6.6. This decay across layers persists even when increasing the input magnitude up to saturating auditory nerve fibers. This phenomenon can be attributed to the random weights in untrained architectures, which transform structured input patterns into uncorrelated noise, leading to vanishing neuronal activity in deeper layers. Our CFC analysis shows no significant coupling, even in layers with sufficient spiking activity, i.e., within the auditory nerve population and the first layer.

In trained networks, noise inputs lead to single neuron firing rate distributions peaking at very low rates and where the activity gradually decreases across layers, as illustrated in Fig. 6.9. This contrasts with the response to speech inputs seen in Fig. 6.3 where the activity was sustained across layers with most of the distribution in the gamma range. We tested uniform noise as well as different noise sources (air conditioner, babble, copy machine and typing) from the the MS-SNSD dataset (Reddy, Beyrami, et al., 2019). Compared to a speech input, all noise types yielded reduced average firing rates (from 60 Hz to about 40 Hz) with most of the neurons remaining silent. This highly dynamic processing of information is naturally efficient at attenuating its activity when processing noise or any input that does not induce sufficient synchronisation. Interestingly, babble noises were found in certain cases to induce some significant PAC patterns, whereas other noise types resulted in no coupling at all. Even though babble noises resemble speech and produced some form of synchronisation, they only triggered a few neurons per layer. Overall, we showed that synchronicity of neural oscillations in the form of PAC results from training and is only triggered when passing an appropriate speech input.

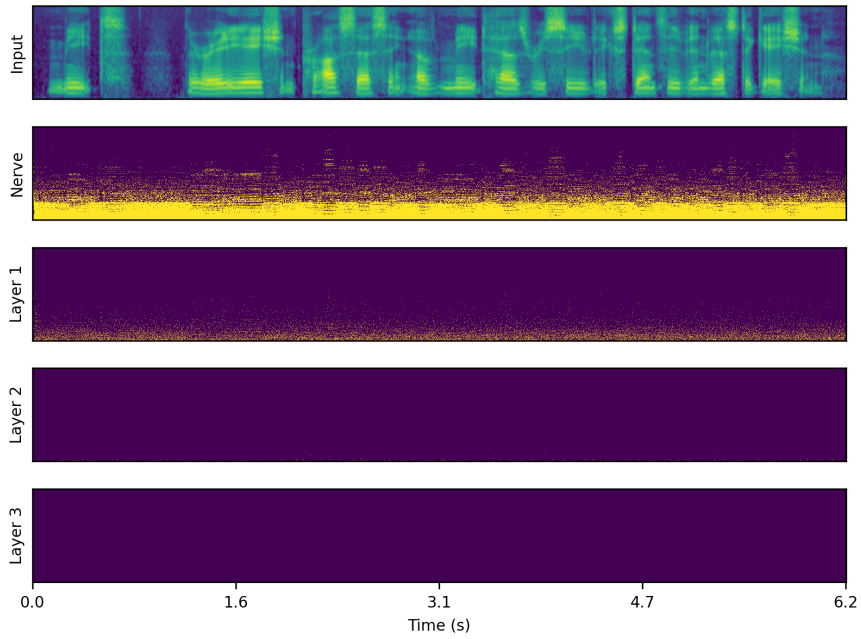

Figure 6.6: Spiking activity of an untrained network in response to speech input. Input filterbank features and resulting spike trains produced across layers. The model uses a 2 ms time step, 16 CNN channels, 3 layers of size 512, 50% AdLIF neurons, 100% feedforward and 50% recurrent connectivity.

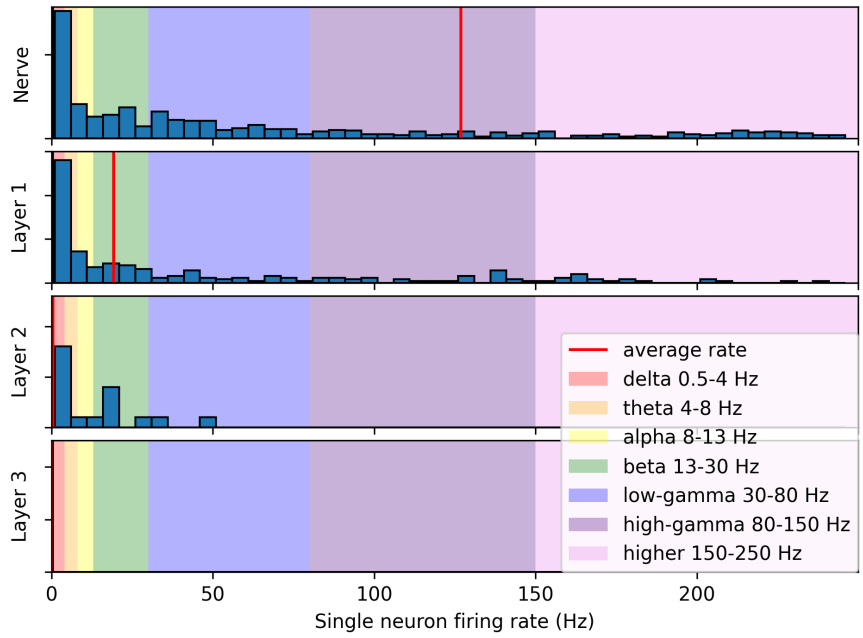

Figure 6.7: Single neuron firing rate distributions of an untrained network in response to speech input. The model and the utterance are the same as in Fig. 6.6

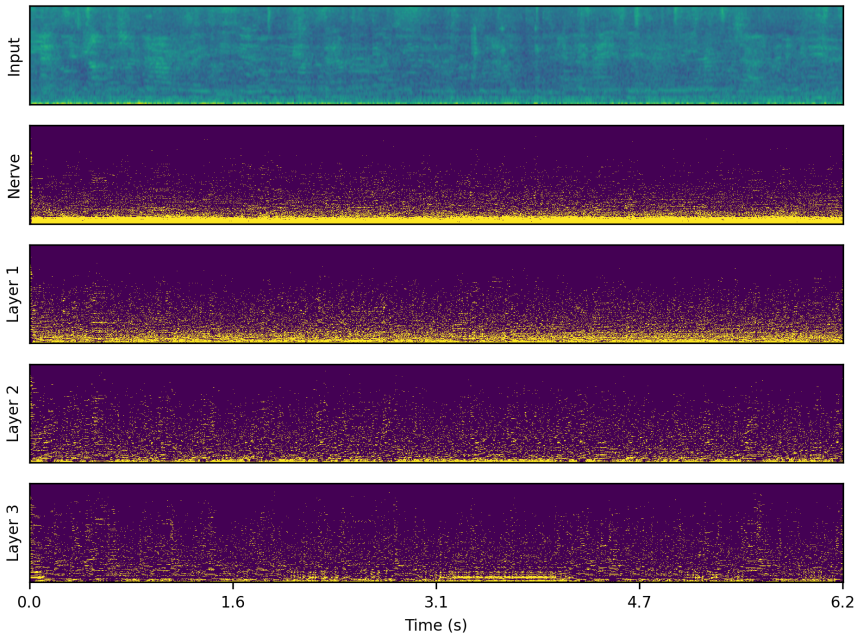

Figure 6.8: Spiking activity of a trained network in response to babble noise input. Input filterbank features and resulting spike trains produced across layers. The model uses a 2 ms time step, 16 CNN channels, 3 layers of size 512, 50% AdLIF neurons, 100% feedforward and 50% recurrent connectivity.

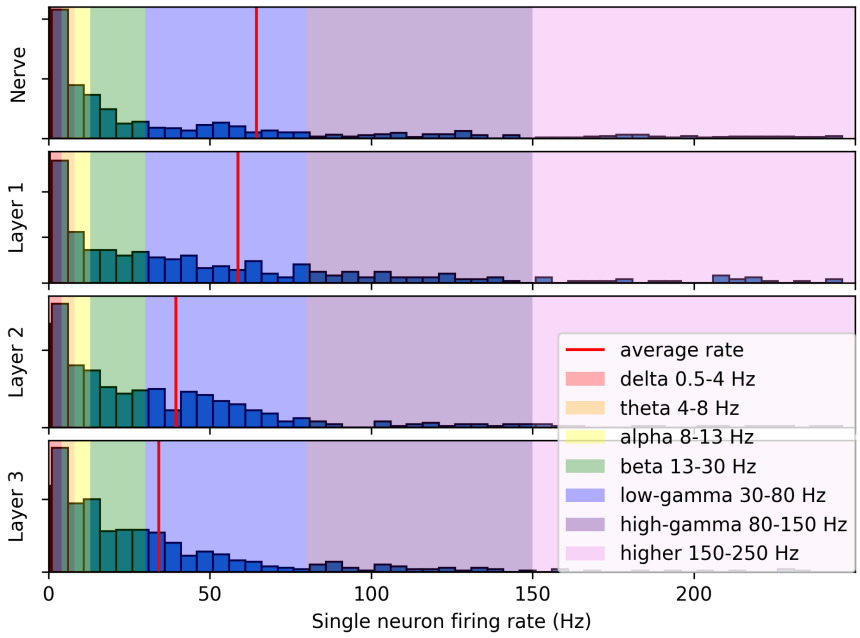

Figure 6.9: Single neuron firing rate distributions of a trained network in response to babble noise input. The model and the utterance are the same as in Fig. 6.8

Scaling to a larger dataset

Our approach was extended to the LibriSpeech dataset (Panayotov et al., 2015) with 960 hours of training data. After 8 epochs, we reached 9.5% WER on the *test-clean* data split. The model uses a 5 ms time step, 16 CNN channels, 3 layers of size 1024, 50% AdLIF neurons, 100% feedforward and 50% recurrent connectivity. During training, dynamic batching was employed with a maximum batch length of 60k steps. As observed on TIMIT, the trained model demonstrated similar CFCs in its spiking activity.

While the computational cost of training on such a large dataset is high – taking approximately two days per epoch with a 5ms time step – we were able to successfully demonstrate the robustness of our method at this scale. However, this extension highlights the need for more efficient tools to speed up training, particularly when finer time resolutions are required for detailed analysis.

Training on speech command recognition task

With our experimental setup, the encoder is directly trained to recognise phonemes on TIMIT and subwords on LibriSpeech. One could therefore assume that the coupled gamma activity emerges from that constraint. In order to test this hypothesis, we run additional experiments on a speech command recognition task where no phoneme or subword recognition is imposed by the training. Instead the model is directly trained to recognise a set of short words. We use the same architecture as on TIMIT, except the average pooling layer is replaced by a readout layer as defined in Section 5.1 which reduces the temporal dimension altogether, as required by the speech command recognition task. Interestingly, using speech command classes as ground truths still produces significant PAC patterns, especially in the last layer. These results indicate that the emergence of the studied rhythms does not require phoneme-based training and may be naturally emerging from speech processing.

Using the second version of the Google Speech Commands data set (Warden, 2018) with 35 classes, we achieve a test set accuracy of 97.05%, which, to the best of our knowledge, improves upon the current state-of-the-art performance using SNNs of 95.35% (Hammouamri, Khalfaoui-Hassani, and Masquelier, 2023). Our model uses a 10 ms time step, 16 CNN channels, 3 layers of size 512, 50% AdLIF neurons, 100% feedforward and 50% recurrent connectivity.

6.4 Conclusions

In this chapter, we introduced a physiologically inspired speech recognition architecture, centred around an SNN, and designed to be compatible with modern deep learning frameworks. As set out in the introduction, we first explored the capabilities and scalability of the proposed speech recognition architecture before analysing neural oscillations.

Our preliminary architectural analysis demonstrated a satisfactory level of scalability to deeper and wider networks, as well as to longer sequences and larger datasets. This scalability was achieved through our approach of utilising the surrogate gradient method to incorporate an SNN into an end-to-end trainable speech recognition pipeline. In addition, our ablation experiments emphasised the importance of including SFA within the neuron model, along with layer-wise recurrent connections, to attain optimal recognition performance. Notably, our implementation of SFA using the AdLIF model outperformed the more popular moving threshold formulation, which corroborates our previous results on speech command recognition in Section 5.1.

The subsequent analysis of the spiking activity across our trained networks in response to speech stimuli revealed that neural oscillations, commonly associated with various cognitive processes in the brain, did emerge from training an architecture to recognise words or phonemes. Through CFC analyses, we measured similar synchronisation phenomena to those observed throughout the human auditory pathway. During speech processing, trained networks exhibited several forms of PAC, including delta-gamma, theta-gamma, alpha-gamma, and beta-gamma, while no such coupling occurred when processing pure background noise. Our networks' ability to synchronise oscillatory activity in the last layer was also associated with improved speech recognition performance, which points to a functional role for neural oscillations in auditory processing. Even though we employ gradient descent training, which does not represent a biologically plausible learning algorithm, our approach was capable of replicating natural phenomena of macro-scale neural coordination. By leveraging the scalability offered by deep learning frameworks, our approach can therefore serve as a valuable tool for studying the emergence and role of brain rhythms.

Building upon the main outcome of replicating neural oscillations, our analysis on SFA and recurrent connections emphasised their key role in actively shaping neural responses and driving synchronisation via inhibition in support of efficient auditory information processing. Our results point towards further work on investigating more realistic feedback mechanisms including efferent pathways across layers. More accurate neuron populations could also be obtained using clustering algorithms.

Further analysis incorporated Dale's law which constrains neurons to be exclusively excitatory or inhibitory. This physiologically inspired principle proved to be a crucial consideration as it significantly increased the number of measured oscillations.

Aside from the fundamental aspect of developing the understanding of biological processes, our research on SNNs also holds significance for the fields of neuromorphic computing and energy efficient technology. Our exploration of the spiking mechanisms that drive dynamic and efficient information processing in the brain is particularly relevant for low-power audio and speech processing applications, such as on-device keyword spotting. In particular, the absence of synchronisation in our architecture when handling background noise results in fewer computations, making our approach well-suited for always-on models.

7 Conclusions and Future Work

7.1 Conclusions

By casting techniques from biological processes, Bayesian statistics and signal processing, this thesis focused on developing the interpretability and physiological plausibility of neural architectures for speech recognition.

In Chapter 3, we built upon a recent Bayesian interpretation of recurrence in conventional ANNs and derived novel parameter-efficient recurrent units. On top of the theoretical contributions, our units were shown to have practical value on ASR tasks, highlighting the importance of a probabilistic derivation.

We then shifted to SNNs in Chapter 4, where we derived biologically-inspired networks from single neuron dynamics. Using the surrogate gradient method, our approach constitutes a special case of RNNs that can be integrated and trained within modern deep learning frameworks while retaining the advantages of energy efficiency. We also showed that the subthreshold dynamics can be solved as a SSM using either a convolution or a recurrent formulation. Nevertheless, threshold-based nonlinear feedbacks still require to loop over time steps.

In Chapter 5, we first applied our SNN approach to speech command recognition tasks and then extended it to LVCSR. Overall, we demonstrated the compatibility and scalability of surrogate gradient SNNs within deep learning frameworks. While requiring considerably fewer parameters and computations, our SNNs achieved competitive results compared to traditional RNNs, demonstrating their applicability to low-powered speech technology.

Finally, in Chapter 6, we transitioned our focus from the technology to the physiology of speech perception. After defining a more biologically inspired architecture, we analysed spike trains across the trained network and measured significant CFCs consistent with neuroscience observations. We showed that gradient descent training on ASR tasks leads to the emergence of neural oscillations in the SNN, but only during speech processing

and not when processing background noise. Our results also highlighted the key role of feedback mechanisms in the form of SFA and layer-wise recurrent connections in regulating the neural activity and improving the recognition performance.

7.2 Future work

Building upon the findings of this thesis, several promising directions for future research are outlined below:

1. **Bayesian interpretation of SNNs**: A first theoretical direction would involve combining the approaches used in this thesis and derive a Bayesian interpretation of SNNs. This could not only enhance our understanding of SNNs, but also help interpret the processing of information in biological networks.

2. **Optimisation of surrogate gradient SNNs**: Further research could also focus on the optimisation of SNNs using the surrogate gradient method. Although we were able to show a certain robustness to vanishing and exploding gradients, our SNNs still employ optimisers originally defined for ANNs. Developing dedicated optimisation techniques specifically designed for SNNs could yield significant performance improvements.

3. **Incorporating more physiological components**: Increasing the number of physiological components, such as including dendrite models and utilising multiple adaptation currents per neuron, could enhance the heterogeneity of spiking neurons in response to stimuli. Additionally, employing more physiologically plausible cochlear and auditory nerve models to convert audio inputs into initial spike trains could improve the biological fidelity of the simulations. Due to the computational intensity of typical biologically-inspired auditory models, pre-computing spike outputs may be necessary. In terms of the learning rule, combining SGD with STDP or other biologically inspired mechanisms could also be explored.

4. **Testing neuroscience hypotheses**: Future work could leverage our approach to reproduce neural dynamics at relatively large scales, facilitating the testing of hypotheses and comparisons with observations in the human brain. On top of developing our fundamental understanding of biological phenomena, this direction could also contribute to biomedical research related to pathological speech or neural disorders, such as the manifestation of auditory hallucinations during psychosis or symptoms of schizophrenia.

5. **SNNs for speech synthesis**: Another avenue for future work is the integration of SNNs into speech synthesis architectures. Due to their physiologically-inspired temporal dynamics, the inclusion of SNNs could enhance the naturalness and expressiveness of synthesised speech.

6. **SNNs for energy-efficient hardware**: The sparse and event-driven nature of biological neural systems can be leveraged to create hardware that operates with significantly lower power consumption compared to traditional ANN-based systems. Research in this area could focus on the design and implementation of neuromorphic chips and architectures that optimise the energy efficiency of SNN operations, potentially revolutionising applications in portable and embedded systems.

In a lot of these directions, writing CUDA code would allow our implementation to run faster on GPUs, reducing training time and enabling more extensive experiments. This could leverage the kernel-based formulation described in Chapter 4.

Bibliography

Abbott, L. F., B. DePasquale, and R.-M. Memmesheimer (2016). "Building functional networks of spiking model neurons". In: *Nature Neuroscience* 19.3, pp. 350–355. DOI: 10.1038/nn.4241.

Abubaker, M., W. Al Qasem, and E. Kvašňák (2021). "Working memory and cross-frequency coupling of neuronal oscillations". In: *Frontiers in Psychology* 12, p. 756661. DOI: 10.3389/fpsyg.2021.756661.

Apicella, A. et al. (2021). "A survey on modern trainable activation functions". In: *Neural Networks* 138, pp. 14–32. DOI: 10.1016/j.neunet.2021.01.026.

Attaheri, A. et al. (2022). "Delta-and theta-band cortical tracking and phase-amplitude coupling to sung speech by infants". In: *NeuroImage* 247, p. 118698. DOI: 10.1016/j.neuroimage.2021.118698.

Augustin, M., J. Ladenbauer, and K. Obermayer (2013). "How adaptation shapes spike rate oscillations in recurrent neuronal networks". In: *Frontiers in Computational Neuroscience* 7, p. 9. DOI: 10.3389/fncom.2013.00009.

Axmacher, N. et al. (2010). "Cross-frequency coupling supports multi-item working memory in the human hippocampus". In: *Proceedings of the National Academy of Sciences* 107.7, pp. 3228–3233. DOI: 10.1073/pnas.0911531107.

Ba, J. L., J. R. Kiros, and G. E. Hinton (2016). "Layer normalization". In: *arXiv preprint*. DOI: 10.48550/arXiv.1607.06450.

Backus, A. R. et al. (2016). "Hippocampal-prefrontal theta oscillations support memory integration". In: *Current Biology* 26.4, pp. 450–457. DOI: 10.1016/j.cub.2015.12.048.

Badel, L., W. Gerstner, and M. J. Richardson (2006). "Dependence of the spike-triggered average voltage on membrane response properties". In: *Neurocomputing* 69.10-12, pp. 1062–1065.

Badel, L., S. Lefort, et al. (2008). "Extracting non-linear integrate-and-fire models from experimental data using dynamic I–V curves". In: *Biological Cybernetics* 99.4, pp. 361–370. DOI: 10.1007/s00422-008-0259-4.

Baevski, A. et al. (2020). “wav2vec 2.0: A framework for self-supervised learning of speech representations”. In: *Advances in Neural Information Processing Systems* 33, pp. 12449–12460.

Bahl, L. R., F. Jelinek, and R. L. Mercer (1983). “A maximum likelihood approach to continuous speech recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2, pp. 179–190. DOI: 10.1109/TPAMI.1983.4767370.

Banerjee, S. et al. (2011). “Oscillatory alpha-band mechanisms and the deployment of spatial attention to anticipated auditory and visual target locations: supramodal or sensory-specific control mechanisms?” In: *Journal of Neuroscience* 31.27, pp. 9923–9932. DOI: 10.1523/JNEUROSCI.4660-10.2011.

Başar, E. et al. (2000). “Brain oscillations in perception and memory”. In: *International Journal of Psychophysiology* 35.2-3, pp. 95–124. DOI: 10.1016/S0167-8760(99)00047-1.

Baum, L. E. et al. (1972). “An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes”. In: *Inequalities* 3.1, pp. 1–8.

Baum, L. E. and J. A. Eagon (1967). “An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology”. In: *Bulletin of the American Mathematical Society* 73.3, pp. 360–363.

Baum, L. E. and T. Petrie (1966). “Statistical inference for probabilistic functions of finite state Markov chains”. In: *The Annals of Mathematical Statistics* 37.6, pp. 1554–1563.

Baum, L. E., T. Petrie, et al. (1970). “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains”. In: *The Annals of Mathematical Statistics* 41.1, pp. 164–171.

Bellec, G. et al. (2018). “Long short-term memory and learning-to-learn in networks of spiking neurons”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., pp. 1412–1421.

Beltagy, I., M. E. Peters, and A. Cohan (2020). “Longformer: The long-document transformer”. In: *arXiv preprint*. DOI: 10.48550/arXiv.2004.05150.

Bengio, Y., P. Simard, and P. Frasconi (1994). “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. DOI: 10.1109/72.279181.

Berger, H. (1929). “Über das Elektroenkephalogramm des Menschen”. In: *Archiv für Psychiatrie und Nervenkrankheiten* 87.1, pp. 527–570.

Bittar, A. and P. N. Garner (2021). "A Bayesian Interpretation of the Light Gated Recurrent Unit". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2965–2969. DOI: 10.1109/ICASSP39728.2021.9414259.

— (2022a). "A surrogate gradient spiking baseline for speech command recognition". In: *Frontiers in Neuroscience* 16, p. 865897. DOI: 10.3389/fnins.2022.865897.

— (2022b). "Bayesian Recurrent Units and the Forward-Backward Algorithm". In: *Proc. Interspeech 2022*, pp. 4137–4141. DOI: 10.21437/Interspeech.2022-11035.

— (2022c). *Surrogate gradient spiking neural networks as encoders for large vocabulary continuous speech recognition*. arXiv: 2212.01187 [cs.CL].

— (2024). "Exploring neural oscillations during speech perception via surrogate gradient spiking neural networks". In: *Frontiers in Neuroscience* 18. DOI: 10.3389/fnins.2024.1449181.

Bodyanskiy, Y. and A. Dolotov (2013). "A Spiking Neuron Model based on the Lambert W Function." In: *IJCCI*, pp. 542–546. DOI: 10.5220/0004631605420546.

Bohte, S. M., J. N. Kok, and H. La Poutre (2002). "Error-backpropagation in temporally encoded networks of spiking neurons". In: *Neurocomputing* 48.1-4, pp. 17–37. DOI: 10.1016/S0925-2312(01)00658-0.

Bonhage, C. E. et al. (2017). "Oscillatory EEG dynamics underlying automatic chunking during sentence processing". In: *NeuroImage* 152, pp. 647–657. DOI: 10.1016/j.neuroimage.2017.03.018.

Bourlard, H. and C. J. Wellekens (1990). "Links between Markov models and multilayer perceptrons". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.12, pp. 1167–1178. DOI: 10.1109/34.62605.

Brette, R. and W. Gerstner (2005). "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity". In: *Journal of Neurophysiology* 94.5, pp. 3637–3642. DOI: 10.1152/jn.00686.2005.

Bridle, J. S. (1990a). "Alpha-nets: A recurrent neural network architecture with a hidden Markov model interpretation". In: *Speech Communication* 9.1, pp. 83–92. DOI: 10.1016/0167-6393(90)90049-F.

— (1990b). "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition". In: *Neurocomputing*. Springer, pp. 227–236. DOI: 10.1007/978-3-642-76153-9_28.

Brodbeck, C., T. Hannagan, and J. S. Magnuson (2024). "Recurrent neural networks as neuro-computational models of human speech recognition". In: *bioRxiv*, pp. 2024–02. DOI: 10.1101/2024.02.20.580731.

Brunel, N., V. Hakim, and M. J. Richardson (2003). "Firing-rate resonance in a generalized integrate-and-fire neuron with subthreshold resonance". In: *Physical Review E* 67.5, p. 051916. DOI: 10.1103/PhysRevE.67.051916.

Butts, D. A. et al. (2007). "Temporal precision in the neural code and the timescales of natural vision". In: *Nature* 449.7158, pp. 92–95. DOI: 10.1038/nature06105.

Buzsaki, G. (2006). *Rhythms of the Brain*. Oxford University Press. DOI: 10.1093/acprof:oso/9780195301069.001.0001.

Buzsáki, G. and E. I. Moser (2013). "Memory, navigation and theta rhythm in the hippocampal-entorhinal system". In: *Nature Neuroscience* 16.2, pp. 130–138. DOI: 10.1038/nn.3304.

Canolty, R. T. et al. (2006). "High gamma power is phase-locked to theta oscillations in human neocortex". In: *Science* 313.5793, pp. 1626–1628. DOI: 10.1126/science.1128115.

Carletta, J. et al. (2005). "The AMI meeting corpus". In: *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*. Vol. 88. Edinburgh, p. 100.

Chacron, M. J., K. Pakdaman, and A. Longtin (2003). "Interspike interval correlations, memory, adaptation, and refractoriness in a leaky integrate-and-fire model with threshold fatigue". In: *Neural Computation* 15.2, pp. 253–278.

Chang, R. and J. Hancock (1966). "On receiver structures for channels having memory". In: *IEEE Transactions on Information Theory* 12.4, pp. 463–468. DOI: 10.1109/TIT.1966.1053923.

Cho, K. et al. (Oct. 2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *Proceedings of the 2014 EMNLP Conference*. Association for Computational Linguistics. Doha, Qatar, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

Choromanski, K. et al. (2020). "Rethinking attention with performers". In: *arXiv preprint*. DOI: 10.48550/arXiv.2009.14794.

Clevert, D.-A., T. Unterthiner, and S. Hochreiter (2015). "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint*. DOI: 10.48550/arXiv.1511.07289.

Colgin, L. L. (2013). "Mechanisms and functions of theta rhythms". In: *Annual Review of Neuroscience* 36, pp. 295–312. DOI: 10.1146/annurev-neuro-062012-170330.

Comsa, I. M. et al. (2020). "Temporal coding in spiking neural networks with alpha synaptic function". In: *IEEE International Conference on Acoustics, Speech and Signal*

Processing (ICASSP). IEEE, pp. 8529–8533. DOI: 10.1109/ICASSP40776.2020.9053856.

Conti, F. et al. (2018). "Chipmunk: A systolically scalable 0.9 mm 2, 3.08 Gop/s/mW@ 1.2 mW accelerator for near-sensor recurrent neural network inference". In: *IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, pp. 1–4.

Cornford, J. et al. (2021). "Learning to live with Dale's principle: {ANN}s with separate excitatory and inhibitory units". In: *International Conference on Learning Representations*.

Cramer, B. et al. (2020). "The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14. DOI: 10.1109/TNNLS.2020.3044364".

Crook, S. M., G. B. Ermentrout, and J. M. Bower (1998). "Spike frequency adaptation affects the synchronization properties of networks of cortical oscillators". In: *Neural Computation* 10.4, pp. 837–854. DOI: 10.1162/089976698300017511.

Dan, Y. and M.-M. Poo (2006). "Spike timing-dependent plasticity: from synapse to perception". In: *Physiological Reviews* 86.3, pp. 1033–1048. DOI: 10.1152/physrev.00030.2005.

Davies, M. et al. (2018). "Loihi: A neuromorphic manycore processor with on-chip learning". In: *IEEE Micro* 38.1, pp. 82–99. DOI: 10.1109/MM.2018.112130359.

De Andrade, D. C. et al. (2018). "A neural attention model for speech command recognition". In: *arXiv preprint arXiv:1808.08929*.

Deckers, L. et al. (2024). "Co-learning synaptic delays, weights and adaptation in spiking neural networks". In: *Frontiers in Neuroscience* 18, p. 1360300. DOI: 10.3389/fnins.2024.1360300.

Dellaferrera, G., F. Martinelli, and M. Cernak (2020). "A bin encoding training of a spiking neural network based voice activity detection". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 3207–3211. DOI: 10.1109/ICASSP40776.2020.9054761.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (methodological)* 39.1, pp. 1–22. DOI: 10.1111/j.2517-6161.1977.tb01600.x.

Devi, N. et al. (2022). "Q10 and Tip Frequencies in Individuals with Normal-Hearing Sensitivity and Sensorineural Hearing Loss". In: *Indian Journal of Otology* 28.2, pp. 126–129. DOI: 10.4103/indianjotol.indianjotol_5_22.

Diehl, P. U. and M. Cook (2015). “Unsupervised learning of digit recognition using spike-timing-dependent plasticity”. In: *Frontiers in Computational Neuroscience* 9, p. 99.

Dugas, C. et al. (2001). “Incorporating second-order functional knowledge for better option pricing”. In: *Advances in Neural Information Processing Systems*. Ed. by T. K. Leen, T. G. Dietterich, and V. Tresp. Vol. 13. MIT Press, pp. 472–478. DOI: https://proceedings.neurips.cc/paper/2000/hash/44968aece94f667e4095002d140b5896-Abstract.html.

Engel, A. K., P. Fries, and W. Singer (2001). “Dynamic predictions: oscillations and synchrony in top–down processing”. In: *Nature Reviews Neuroscience* 2.10, pp. 704–716. DOI: 10.1038/35094565.

FitzHugh, R. (1961). “Impulses and physiological states in theoretical models of nerve membrane”. In: *Biophysical Journal* 1.6, p. 445. DOI: 10.1016/s0006-3495(61)86902-6.

Forney, G. D. (1973). “The Viterbi algorithm”. In: *Proceedings of the IEEE* 61.3, pp. 268–278. DOI: 10.1109/PROC.1973.9030.

Foxe, J. J. and A. C. Snyder (2011). “The role of alpha-band brain oscillations as a sensory suppression mechanism during selective attention”. In: *Frontiers in Psychology* 2, p. 10747. DOI: 10.3389/fpsyg.2011.00154.

Frémaux, N. and W. Gerstner (2016). “Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules”. In: *Frontiers in Neural Circuits* 9, p. 85. DOI: 10.3389/fncir.2015.00085.

Fries, P. et al. (2001). “Modulation of oscillatory neuronal synchronization by selective visual attention”. In: *Science* 291.5508, pp. 1560–1563. DOI: 10.1126/science.1055465.

Fuortes, M. and F. Mantegazzini (1962). “Interpretation of the repetitive firing of nerve cells”. In: *The Journal of General Physiology* 45.6, pp. 1163–1179.

Ganguly, C. et al. (2024). “Spike frequency adaptation: bridging neural models and neuromorphic applications”. In: *Communications Engineering* 3.1, p. 22. DOI: 10.1038/s44172-024-00165-9.

Garner, P. N. and S. Tong (2021). “A Bayesian Approach to Recurrence in Neural Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.8, pp. 2527–2537. DOI: 10.1109/TPAMI.2020.2976978.

Garofolo, J. S. et al. (Feb. 1993). *DARPA TIMIT Acoustic-Phonetic Continous Speech Corpus CD-ROM. NIST Speech Disc 1-1.1*. NISTIR 4930. Gaithersburg, MD, USA: NIST. DOI: 10.35111/17gk-bn40.

Gers, F. A., J. Schmidhuber, and F. Cummins (2000). "Learning to forget: Continual prediction with LSTM". In: *Neural Computation* 12, pp. 2451–2471. DOI: 10.1049/cp:19991218.

Gers, F. A., N. N. Schraudolph, and J. Schmidhuber (2002). "Learning precise timing with LSTM recurrent networks". In: *Journal of Machine Learning Research* 3.Aug, pp. 115–143.

Gerstner, W. and W. M. Kistler (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press. DOI: 10.1017/CBO9780511815706.

Ghitza, O. (2011). "Linking speech perception and neurophysiology: speech decoding guided by cascaded oscillators locked to the input rhythm". In: *Frontiers in Psychology* 2, p. 130. DOI: 10.3389/fpsyg.2011.00130.

Gillick, L. and S. J. Cox (May 1989). "Some statistical issues in the comparison of speech recognition algorithms". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vol. 1. Glasgow, UK, pp. 532–535. DOI: 10.1109/ICASSP.1989.266481.

Giraldo, J. S. P., S. Lauwereins, et al. (2020). "Vocell: A 65-nm Speech-Triggered Wake-Up SoC for 10-μ W Keyword Spotting and Speaker Verification". In: *IEEE Journal of Solid-State Circuits* 55.4, pp. 868–878.

Giraldo, J. S. P., C. O'Connor, and M. Verhelst (2019). "Efficient keyword spotting through hardware-aware conditional execution of deep neural networks". In: *IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*. IEEE, pp. 1–8.

Giraud, A.-L. and D. Poeppel (2012). "Cortical oscillations and speech processing: emerging computational principles and operations". In: *Nature Neuroscience* 15.4, pp. 511–517. DOI: 10.1038/nn.3063.

Glorot, X., A. Bordes, and Y. Bengio (2011). "Deep sparse rectifier neural networks". In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. Fort Lauderdale, FL, USA, pp. 315–323.

Gollisch, T. and M. Meister (2008). "Rapid neural coding in the retina with relative spike latencies". In: *Science* 319.5866, pp. 1108–1111. DOI: 10.1126/science.1149639.

Gong, Y., Y.-A. Chung, and J. Glass (2021). "AST: Audio spectrogram transformer". In: *arXiv preprint arXiv:2104.01778*.

Graves, A., S. Fernández, et al. (2006). "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd International Conference on Machine Learning*, pp. 369–376. DOI: 10.1145/1143844.1143891.

Graves, A., N. Jaitly, and A.-r. Mohamed (2013). "Hybrid speech recognition with deep bidirectional LSTM". In: *IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, pp. 273–278. DOI: 10.1007/11550907_126.

Graves, A. and J. Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5-6, pp. 602–610.

Gu, A., K. Goel, A. Gupta, et al. (2022). "On the parameterization and initialization of diagonal state space models". In: *Advances in Neural Information Processing Systems* 35, pp. 35971–35983.

Gu, A., K. Goel, and C. Ré (2021). "Efficiently modeling long sequences with structured state spaces". In: *arXiv preprint*. DOI: 10.48550/arXiv.2111.00396.

Gulati, A. et al. (2020). "Conformer: Convolution-augmented Transformer for Speech Recognition". In: *Proc. Interspeech 2020*, pp. 5036–5040. DOI: 10.21437/Interspeech.2020-3015.

Gundersen, T., ø. Skarstein, and T. Sikkeland (1978). "A study of the vibration of the basilar membrane in human temporal bone preparations by the use of the Mossbauer effect". In: *Acta Oto-laryngologica* 86.1-6, pp. 225–232. DOI: 10.3109/00016487809124740.

Gupta, A., A. Gu, and J. Berant (2022). "Diagonal state spaces are as effective as structured state spaces". In: *Advances in Neural Information Processing Systems* 35, pp. 22982–22994.

Hammouamri, I., I. Khalfaoui-Hassani, and T. Masquelier (2023). *Learning delays in spiking neural networks using dilated convolutions with learnable spacings*. arXiv: 2306.17670 [cs.NE].

Han, S. et al. (2015). "Learning both Weights and Connections for Efficient Neural Network". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., pp. 1135–1143.

He, K. et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

Henningsen-Schomers, M. R. and F. Pulvermüller (2022). "Modelling concrete and abstract concepts using brain-constrained deep neural networks". In: *Psychological Research* 86.8, pp. 2533–2559. DOI: 10.1007/s00426-021-01591-6.

Hill, A. V. (1936). "Excitation and accommodation in nerve". In: *Proceedings of the Royal Society of London. Series B-Biological Sciences* 119.814, pp. 305–355. DOI: 10.1098/rspb.1936.0012.

Hochreiter, S. et al. (2001). *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*.

Hochreither, S. and J. Schmidhuber (Nov. 1997). "Long short-term memory". In: *Neural Computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Hodgkin, A. L. and A. F. Huxley (1952). "A quantitative description of membrane current and its application to conduction and excitation in nerve". In: *The Journal of Physiology* 117.4, p. 500. DOI: 10.1113/jphysiol.1952.sp004764.

Hovsepyan, S., I. Olasagasti, and A.-L. Giraud (2020). "Combining predictive coding and neural oscillations enables online syllable recognition in natural speech". In: *Nature Communications* 11.1, p. 3117. DOI: 10.1038/s41467-020-16956-5.

Huh, D. and T. J. Sejnowski (2018). "Gradient Descent for Spiking Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., pp. 1440–1450.

Hülsemann, M. J., E. Naumann, and B. Rasch (2019). "Quantification of phase-amplitude coupling in neuronal oscillations: comparison of phase-locking value, mean vector length, modulation index, and generalized-linear-modeling-cross-frequency-coupling". In: *Frontiers in Neuroscience* 13, p. 573. DOI: 10.3389/fnins.2019.00573.

Hummos, A. and S. S. Nair (2017). "An integrative model of the intrinsic hippocampal theta rhythm". In: *PloS One* 12.8, e0182648. DOI: 10.1371/journal.pone.0182648.

Hyafil, A. et al. (2015). "Speech encoding by coupled cortical theta and gamma oscillations". In: *Elife* 4, e06213. DOI: 10.7554/eLife.06213.

Ioffe, S. and C. Szegedy (Feb. 2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. arXiv: 1502.03167.

Itskov, V. et al. (2008). "Theta-mediated dynamics of spatial information in hippocampus". In: *Journal of Neuroscience* 28.23, pp. 5959–5964. DOI: 10.1523/JNEUROSCI.5262-07.2008.

Izhikevich, E. M. (2001). "Resonate-and-fire neurons". In: *Neural Networks* 14.6-7, pp. 883–894. DOI: 10.1016/S0893-6080(01)00078-8.

— (2003). "Simple model of spiking neurons". In: *IEEE Transactions on Neural Networks* 14.6, pp. 1569–1572. DOI: 10.1109/TNN.2003.820440.

— (2007). *Dynamical systems in neuroscience*. MIT press. DOI: 10.7551/mitpress/2526.001.0001.

Jang, H. et al. (2019). "An introduction to probabilistic spiking neural networks: Probabilistic models, learning rules, and applications". In: *IEEE Signal Processing Magazine* 36.6, pp. 64–77. DOI: 10.1109/MSP.2019.2935234.

Jeffares, A. et al. (2022). “Spike-inspired rank coding for fast and accurate recurrent neural networks”. In: *International Conference on Learning Representations*.

Jensen, O. and L. L. Colgin (2007). “Cross-frequency coupling between neuronal oscillations”. In: *Trends in Cognitive Sciences* 11.7, pp. 267–269. DOI: 10.1016/j.tics.2007.05.003.

Jensen, O. and A. Mazaheri (2010). “Shaping functional architecture by oscillatory alpha activity: gating by inhibition”. In: *Frontiers in Human Neuroscience* 4, p. 186. DOI: 10.3389/fnhum.2010.00186.

Jirsa, V. and V. Müller (2013). “Cross-frequency coupling in real and virtual brain networks”. In: *Frontiers in Computational Neuroscience* 7, p. 78. DOI: 10.3389/fncom.2013.00078.

Jolivet, R., A. Rauch, et al. (2006). “Predicting spike timing of neocortical pyramidal neurons by simple threshold models”. In: *Journal of Computational Neuroscience* 21.1, pp. 35–49.

Jolivet, R., J. Timothy, and W. Gerstner (2003). “The spike response model: a framework to predict neuronal spike trains”. In: *Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP)*. Springer, pp. 846–853. DOI: 10.1007/3-540-44989-2_101.

Jones, S. R. (2016). “When brain rhythms aren’t ‘rhythmic’: implication for their mechanisms and meaning”. In: *Current Opinion in Neurobiology* 40, pp. 72–80. DOI: 10.1016/j.conb.2016.06.010.

Juang, B. H. and L. R. Rabiner (1991). “Hidden Markov models for speech recognition”. In: *Technometrics* 33.3, pp. 251–272. DOI: 10.1080/00401706.1991.10484833.

Kadetotad, D. et al. (2020). “An 8.93 TOPS/W LSTM recurrent neural network accelerator featuring hierarchical coarse-grain sparsity for on-device speech recognition”. In: *IEEE Journal of Solid-State Circuits* 55.7, pp. 1877–1887.

Kaiser, J., H. Mostafa, and E. Neftci (2020). “Synaptic plasticity dynamics for deep continuous local learning (DECOLLE)”. In: *Frontiers in Neuroscience* 14, p. 424. DOI: 10.3389/fnins.2020.00424.

Kalman, R. E. (Mar. 1960). “A new approach to linear filtering and prediction problems”. In: *ASME Journal of Basic Engineering* 82.1, pp. 35–45.

Karita, S. et al. (2019). “A comparative study on transformer vs rnn in speech applications”. In: *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, pp. 449–456. DOI: 10.1109/ASRU46091.2019.9003750.

Kasabov, N. K. (2019). *Time-space, spiking neural networks and brain-inspired artificial intelligence*. Springer. DOI: 10.1007/978-3-662-57715-8.

Katharopoulos, A. et al. (2020). “Transformers are rnns: Fast autoregressive transformers with linear attention”. In: *International Conference on Machine Learning*. PMLR, pp. 5156–5165.

Kingma, D. and J. Ba (Dec. 2015). “Adam: A method for stochastic optimization”. In: *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA. arXiv: 1412.6980.

Klimesch, W. (2012). “Alpha-band oscillations, attention, and controlled access to stored information”. In: *Trends in Cognitive Sciences* 16.12, pp. 606–617. DOI: 10.1016/j.tics.2012.10.007.

Kucewicz, M. T. et al. (2017). “Dissecting gamma frequency activity during human memory processing”. In: *Brain* 140.5, pp. 1337–1350. DOI: 10.1093/brain/awx043.

Lapicque, L. (1907). “Recherches quantitatives sur l’excitation électrique des nerfs traitée comme une polarisation”. In: *Journal de Physiologie et de Pathologie Générale* 9, pp. 620–635. DOI: 10.1007/s00422-007-0189-6.

Leng, L. et al. (2018). “Spiking neurons with short-term synaptic plasticity form superior generative networks”. In: *Scientific Reports* 8.1, pp. 1–11.

Li, B., R. Pang, et al. (2021). “Scaling end-to-end models for large-scale multilingual ASR”. In: *Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, pp. 1011–1018.

Li, P., J. Cornford, et al. (2024). “Learning better with Dale’s Law: A spectral perspective”. In: *Advances in Neural Information Processing Systems* 36.

Lin, X. and G. Shi (2018). “A supervised multi-spike learning algorithm for recurrent spiking neural networks”. In: *International Conference on Artificial Neural Networks*. Springer, pp. 222–234. DOI: 10.1007/978-3-030-01418-6_22.

Lin, X., X. Wang, and Z. Hao (2017). “Supervised learning in multilayer spiking neural networks with inner products of spike trains”. In: *Neurocomputing* 237, pp. 59–70. DOI: 10.1016/j.neucom.2016.08.087.

Maass, W. (1997). “Networks of spiking neurons: the third generation of neural network models”. In: *Neural Networks* 10.9, pp. 1659–1671. DOI: 10.1016/S0893-6080(97)00011-7.

MacKay, W. A. (1997). “Synchronized neuronal oscillations and their role in motor processes”. In: *Trends in Cognitive Sciences* 1.5, pp. 176–183. DOI: 10.1016/S1364-6613(97)01059-0.

Magnuson, J. S. et al. (2020). “EARSHOT: A minimal neural network model of incremental human speech recognition”. In: *Cognitive Science* 44.4, e12823. DOI: 10.1111/cogs.12823.

Mainen, Z. F. and T. J. Sejnowski (1995). "Reliability of spike timing in neocortical neurons". In: *Science* 268.5216, pp. 1503–1506. DOI: 10.1126/science.7770778.

Meddis, R. (1986). "Simulation of mechanical to neural transduction in the auditory receptor". In: *The Journal of the Acoustical Society of America* 79.3, pp. 702–711. DOI: 10.1121/1.393460.

— (1988). "Simulation of auditory–neural transduction: Further studies". In: *The Journal of the Acoustical Society of America* 83.3, pp. 1056–1063. DOI: 10.1121/1.396050.

Mensi, S. et al. (2012). "Parameter extraction and classification of three cortical neuron types reveals two distinct adaptation mechanisms". In: *Journal of Neurophysiology* 107.6, pp. 1756–1775.

Mesgarani, N. et al. (2014). "Phonetic feature encoding in human superior temporal gyrus". In: *Science* 343.6174, pp. 1006–1010. DOI: 10.1126/science.1245994.

Millet, J., C. Caucheteux, et al. (2022). "Toward a realistic model of speech processing in the brain with self-supervised learning". In: *Advances in Neural Information Processing Systems* 35, pp. 33428–33443.

Millet, J. and J.-R. King (2021). *Inductive biases, pretraining and fine-tuning jointly account for brain responses to speech*. arXiv: 2103.01032 [cs.CL].

Miyazaki, K., M. Murata, and T. Koriyama (2023). "Structured state space decoder for speech recognition and synthesis". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10096135.

Mizuseki, K. et al. (2009). "Theta oscillations provide temporal windows for local circuit computation in the entorhinal-hippocampal loop". In: *Neuron* 64.2, pp. 267–280. DOI: 10.1016/j.neuron.2009.08.037.

Moraitis, T., A. Sebastian, and E. Eleftheriou (2020). *Optimality of short-term synaptic plasticity in modelling certain dynamic environments*. arXiv: 2009.06808.

Morris, C. and H. Lecar (1981). "Voltage oscillations in the barnacle giant muscle fiber". In: *Biophysical Journal* 35.1, pp. 193–213. DOI: 10.1016/S0006-3495(81)84782-0.

Mostafa, H. (2017). "Supervised learning based on temporal coding in spiking neural networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.7, pp. 3227–3235. DOI: 10.1109/TNNLS.2017.2726060.

Neal, R. M. (2012). *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media. DOI: 10.1007/978-1-4612-0745-0.

Neftci, E. O., H. Mostafa, and F. Zenke (2019). "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural

networks". In: *IEEE Signal Processing Magazine* 36.6, pp. 51–63. DOI: 10.1109/MSP.2019.2931595.

O'Reilly, R. C. and Y. Munakata (2000). *Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain*. MIT press. DOI: 10.7551/mitpress/2014.001.0001.

Obleser, J. and N. Weisz (2012). "Suppressed alpha oscillations predict intelligibility of speech and its acoustic details". In: *Cerebral Cortex* 22.11, pp. 2466–2477. DOI: 10.1093/cercor/bhr325.

Panayotov, V. et al. (2015). "Librispeech: an ASR corpus based on public domain audio books". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5206–5210.

Panda, P., S. A. Aketi, and K. Roy (2020). "Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization". In: *Frontiers in Neuroscience* 14, p. 653. DOI: 10.3389/fnins.2020.00653.

Paszke, A., S. Gross, S. Chintala, et al. (2017). "Automatic differentiation in pytorch". In: *NIPS Workshops*.

Paszke, A., S. Gross, F. Massa, et al. (2019). "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems* 32.

Pellegrini, T., R. Zimmer, and T. Masquelier (2021). "Low-activity supervised convolutional spiking neural networks applied to speech commands recognition". In: *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 97–103. DOI: 10.1109/SLT48900.2021.9383587.

Perez-Nieves, N. et al. (2021). "Neural heterogeneity promotes robust learning". In: *Nature Communications* 12.1, p. 5791. DOI: 10.1038/s41467-021-26022-3.

Pfeiffer, M. and T. Pfeil (2018). "Deep learning with spiking neurons: opportunities and challenges". In: *Frontiers in Neuroscience* 12, p. 774. DOI: 10.3389/fnins.2018.00774.

Ponghiran, W. and K. Roy (2022). "Spiking neural networks with improved inherent recurrence dynamics for sequential learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 7, pp. 8001–8008. DOI: 10.1609/aaai.v36i7.20771.

Povey, D. et al. (Dec. 2011). "The kaldi speech recognition toolkit". In: *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*. Hawaii, USA, pp. 1–4.

Pulvermüller, F. (2023). "Neurobiological mechanisms for language, symbols and concepts: clues from brain-constrained deep neural networks". In: *Progress in Neurobiology*, p. 102511. DOI: 10.1016/j.pneurobio.2023.102511.

Pulvermüller, F. et al. (2021). “Biological constraints on neural network models of cognitive function”. In: *Nature Reviews Neuroscience* 22.8, pp. 488–502. DOI: 10.1038/s41583-021-00473-5.

Rabiner, L. R. (1989). “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77.2, pp. 257–286. DOI: 10.1109/5.18626.

Radford, A. et al. (2023). “Robust speech recognition via large-scale weak supervision”. In: *International Conference on Machine Learning*. PMLR, pp. 28492–28518.

Ramos-Murguialday, A. and N. Birbaumer (2015). “Brain oscillatory signatures of motor tasks”. In: *Journal of Neurophysiology* 113.10, pp. 3663–3682. DOI: 10.1152/jn.00467.2013.

Ravanelli, M., A. Bordes, and Y. Bengio (2018). “Light Gated Recurrent Units for Speech Recognition”. In: *Transactions on Emerging Topics in Computational Intelligence* 2.2, pp. 92–102. DOI: 10.1109/TETCI.2017.2762739.

Ravanelli, M., P. Brakel, et al. (Aug. 2017). “Improving speech recognition by revising gated recurrent units”. In: *Proceedings of Interspeech*. Stockholm, Sweden, pp. 1308–1312. DOI: 10.21437/Interspeech.2017-775.

Ravanelli, M., T. Parcollet, and Y. Bengio (May 2019). “The pytorch-kaldi speech recognition toolkit”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, UK, pp. 6465–6469. DOI: 10.1109/ICASSP.2019.8683713.

Ravanelli, M. and Y. Bengio (2018). “Speaker recognition from raw waveform with sincnet”. In: *IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 1021–1028. DOI: 10.1109/SLT.2018.8639585.

Ravanelli, M., T. Parcollet, P. Plantinga, et al. (2021). *SpeechBrain: A General-Purpose Speech Toolkit*. arXiv: 2106.04624.

Reddy, C. K., E. Beyrami, et al. (2019). “A Scalable Noisy Speech Dataset and Online Subjective Test Framework”. In: *Proc. Interspeech*, pp. 1816–1820. DOI: 10.21437/Interspeech.2019-3087.

Reddy, L., M. W. Self, et al. (2021). “Theta-phase dependent neuronal coding during sequence learning in human single neurons”. In: *Nature Communications* 12.1, p. 4839. DOI: 10.1038/s41467-021-25150-0.

Rossbroich, J., J. Gygax, and F. Zenke (2022). “Fluctuation-driven initialization for spiking neural network training”. In: *Neuromorphic Computing and Engineering* 2.4, p. 044016.

Roy, K., A. Jaiswal, and P. Panda (2019). “Towards spike-based machine intelligence with neuromorphic computing”. In: *Nature* 575.7784, pp. 607–617. DOI: 10.1038/s41586-019-1677-2.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). “Learning representations by back-propagating errors”. In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0.

Rumelhart, D. E. and J. L. McClelland (July 1986). *Parallel distributed processing*. Vol. 1. MIT Press. DOI: 10.7551/mitpress/5236.001.0001.

Rybakov, O. et al. (2020). “Streaming keyword spotting on mobile devices”. In: *arXiv preprint arXiv:2005.06720*. arXiv: 2005.06720.

Saenz, M. and D. R. Langers (2014). “Tonotopic mapping of human auditory cortex”. In: *Hearing Research* 307, pp. 42–52. DOI: 10.1016/j.heares.2013.07.016.

Salaj, D. et al. (2021). “Spike frequency adaptation supports network computations on temporally dispersed information”. In: *Elife* 10, e65459. DOI: 10.7554/eLife.65459.

Saon, G., A. Gupta, and X. Cui (2023). “Diagonal state space augmented transformers for speech recognition”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10096271.

Schaefer, C. J. et al. (2021). “LSTMs for Keyword Spotting with ReRAM-based Compute-In-Memory Architectures”. In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, pp. 1–5.

Scharf, L. L. and C. Demeure (1991). *Statistical signal processing: detection, estimation, and time series analysis*. Prentice Hall.

Schrauwen, B. and J. Van Campenhout (2006). “Backpropagation for population-temporal coded spiking neural networks”. In: *IEEE International Joint Conference on Neural Network Proceedings*. IEEE, pp. 1797–1804. DOI: 10.1109/IJCNN.2006.246897.

Schultz, W. (2007). “Behavioral dopamine signals”. In: *Trends in Neurosciences* 30.5, pp. 203–210. DOI: 10.1016/j.tins.2007.03.007.

— (2010). “Dopamine signals for reward value and risk: basic and recent data”. In: *Behavioral and Brain Functions* 6.1, p. 24. DOI: 10.1186/1744-9081-6-24.

Schultz, W., P. Dayan, and P. R. Montague (1997). “A neural substrate of prediction and reward”. In: *Science* 275.5306, pp. 1593–1599. DOI: 10.1126/science.275.5306.1593.

Schuster, M. and K. K. Paliwal (1997). “Bidirectional recurrent neural networks”. In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681. DOI: 10.1109/78.650093.

Senkowski, D. et al. (2007). “Good times for multisensory integration: effects of the precision of temporal synchrony as revealed by gamma-band oscillations”. In: *Neuropsychologia* 45.3, pp. 561–571. DOI: 10.1016/j.neuropsychologia.2006.01.013.

Sennrich, R., B. Haddow, and A. Birch (2015). *Neural machine translation of rare words with subword units*. arXiv:1508.07909. arXiv: 1508.07909 [cs.CL].

Shaban, A., S. S. Bezugam, and M. Suri (2021). “An adaptive threshold neuron for recurrent spiking neural networks with nanodevice hardware implementation”. In: *Nature Communications* 12.1, pp. 1–11. DOI: 10.1038/s41467-021-24427-8.

Shan, H. et al. (2024). “Augmenting Conformers With Structured State-Space Sequence Models For Online Speech Recognition”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 12221–12225. DOI: 10.1109/ICASSP48485.2024.10445950.

Shrestha, S. B. and G. Orchard (2018). “SLAYER: Spike Layer Error Reassignment in Time”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., pp. 1412–1421.

Sieroka, N., H. G. Dosch, and A. Rupp (2006). “Semirealistic models of the cochlea”. In: *The Journal of the Acoustical Society of America* 120.1, pp. 297–304. DOI: 10.1121/1.2204438.

Smith, J. T., A. Warrington, and S. W. Linderman (2022). “Simplified state space layers for sequence modeling”. In: *arXiv preprint*. DOI: 10.48550/arXiv.2208.04933.

Strauß, A., S. A. Kotz, et al. (2014). “Alpha and theta brain oscillations index dissociable processes in spoken word recognition”. In: *NeuroImage* 97, pp. 387–395. DOI: 10.1016/j.neuroimage.2014.04.005.

Strauß, A., M. Wöstmann, and J. Obleser (2014). “Cortical alpha oscillations as a tool for auditory selective inhibition”. In: *Frontiers in Human Neuroscience* 8, p. 350. DOI: 10.3389/fnhum.2014.00350.

Sun, M., B. Hoffmeister, et al. (Mar. 2017). *Model shrinking for embedded keyword spotting*. US Patent 9,600,231.

Sun, M., A. Raju, et al. (2016). “Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting”. In: *IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 474–480.

Tieleman, T. and G. Hinton (2012). “Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural Networks for Machine Learning*.

Tort, A. B., R. W. Komorowski, et al. (2009). “Theta–gamma coupling increases during the learning of item–context associations”. In: *Proceedings of the National Academy of Sciences* 106.49, pp. 20942–20947. DOI: 10.1073/pnas.0911331106.

Tort, A. B., M. A. Kramer, et al. (2008). “Dynamic cross-frequency couplings of local field potential oscillations in rat striatum and hippocampus during performance of a T-maze task”. In: *Proceedings of the National Academy of Sciences* 105.51, pp. 20517–20522. DOI: 10.1073/pnas.0810524105.

Treves, A. (1993). “Mean-field analysis of neuronal spike dynamics”. In: *Network: Computation in Neural Systems* 4.3, p. 259. DOI: 10.1088/0954-898x_4_3_002.

Van Rullen, R. and S. J. Thorpe (2001). “Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex”. In: *Neural Computation* 13.6, pp. 1255–1283. DOI: 10.1162/08997660152002852.

Vaswani, A. et al. (2017). “Attention is all you need”. In: *Advances in Neural Information Processing Systems*.

Vinck, M. et al. (2013). “Attentional modulation of cell-class-specific gamma-band synchronization in awake monkey area v4”. In: *Neuron* 80.4, pp. 1077–1089. DOI: 10.1016/j.neuron.2013.08.019.

Viterbi, A. (1967). “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE Transactions on Information Theory* 13.2, pp. 260–269. DOI: 10.1109/TIT.1967.1054010.

Wang, S., B. Z. Li, et al. (2020). “Linformer: Self-attention with linear complexity”. In: *arXiv preprint*. DOI: 10.48550/arXiv.2006.04768.

Wang, X., X. Lin, and X. Dang (2019). “A delay learning algorithm based on spike train kernels for spiking neurons”. In: *Frontiers in Neuroscience* 13, p. 252. DOI: 10.3389/fnins.2019.00252.

Warden, P. (Apr. 2018). “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition”. In: *arXiv preprint arXiv:1804.03209*. arXiv: 1804.03209 [cs.CL].

Watanabe, S. et al. (2017). “Hybrid CTC/attention architecture for end-to-end speech recognition”. In: *IEEE Journal of Selected Topics in Signal Processing* 11.8, pp. 1240–1253.

Wilcoxon, F. (Dec. 1945). “Individual Comparisons by Ranking Methods”. In: *Biometrics Bulletin*. Vol. 1. 6, pp. 80–83. DOI: 10.1007/978-1-4612-4380-9_16.

Williams, R. J. and D. Zipser (1989). “A learning algorithm for continually running fully recurrent neural networks”. In: *Neural Computation* 1.2, pp. 270–280. DOI: 10.1162/neco.1989.1.2.270.

Womelsdorf, T. and P. Fries (2007). "The role of neuronal synchronization in selective attention". In: *Current Opinion in Neurobiology* 17.2, pp. 154–160. DOI: 10.1016/j.conb.2007.02.002.

Wöstmann, M., S.-J. Lim, and J. Obleser (2017). "The human neural alpha response to speech is a proxy of attentional control". In: *Cerebral Cortex* 27.6, pp. 3307–3317. DOI: 10.1093/cercor/bhx074.

Wu, J., Y. Chua, M. Zhang, G. Li, H. Li, et al. (2021). "A tandem learning rule for effective training and rapid inference of deep spiking neural networks". In: *IEEE Transactions on Neural Networks and Learning Systems*. DOI: 10.1109/TNNLS.2021.3095724.

Wu, J., Y. Chua, M. Zhang, H. Li, and K. C. Tan (2018). "A spiking neural network framework for robust sound classification". In: *Frontiers in Neuroscience* 12, p. 836. DOI: 10.3389/fnins.2018.00836.

Wu, J., E. Yılmaz, et al. (2020). "Deep spiking neural networks for large vocabulary automatic speech recognition". In: *Frontiers in Neuroscience* 14, p. 199. DOI: 10.3389/fnins.2020.00199.

Yao, M. et al. (2021). "Temporal-wise Attention Spiking Neural Networks for Event Streams Classification". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10221–10230.

Yin, B., F. Corradi, and S. M. Bohté (2020). "Effective and efficient computation with multiple-timescale spiking recurrent neural networks". In: *International Conference on Neuromorphic Systems 2020*, pp. 1–8. DOI: 10.1145/3407197.3407225.

— (2021). "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks". In: *Nature Machine Intelligence* 3.10, pp. 905–913. DOI: 10.1038/s42256-021-00397-w.

Zeghidour, N. et al. (2021). *LEAF: A learnable frontend for audio classification*. arXiv: 2101.08596 [cs.SD].

Zeiler, M. D. (2012). *Adadelta: an adaptive learning rate method*. arXiv: 1212.5701.

Zenke, F. and S. Ganguli (2018). "Superspike: Supervised learning in multilayer spiking neural networks". In: *Neural Computation* 30.6, pp. 1514–1541. DOI: 10.1162/neco_a_01086.

Zhang, Y. et al. (2017). "Hello edge: Keyword spotting on microcontrollers". In: *arXiv preprint arXiv:1711.07128*.

Zhou, G.-B. et al. (June 2016). "Minimal gated unit for recurrent neural networks". In: *International Journal of Automation and Computing* 13.3, pp. 226–234. DOI: 10.1007/s11633-016-1006-2.

Alexandre Bittar

+41 78 663 07 77 | Clarens, Switzerland | alex.bittar@yahoo.fr | LinkedIn | Google Scholar | GitHub

Research Interests

Audio Machine Learning, Speech Recognition, Signal Processing, Neuromorphic Computing.

Experience

- **Apple Machine Learning and AI [🌐]** *Apr 2023 - Sep 2023*
  Research Intern Zürich, Switzerland
  - Developed a low-powered keyword spotting architecture with input-dependent dynamic depth.
  - Improved task performance while reducing processing and power costs in always-on streaming scenarios.
  - Dynamic depth allowed the model to skip up to 97% of computation on background noise inputs.
  - Worked in a high-impact research team, contributing innovative solutions for energy-efficient AI models.

- **Idiap Research Intitute [🌐]** *Mar 2020 - Oct 2024*
  Graduate Research Assistant Martigny, Switzerland
  - Researched theoretical foundations of neural networks, including a Bayesian interpretation of recurrence.
  - Specialised in bio-inspired spiking neural networks, advancing their application in speech recognition.
  - Used physiological insights to define innovative energy-efficient neuromorphic systems.
  - Leveraged deep learning to develop new neuroscience tools for inferring physiological mechanisms.
  - Developed parameter-efficient, interpretable, and competitive architectures for speech technology.

- **Digital Music Solutions [🌐]** *Oct 2019 - Mar 2020*
  Research and Development Intern Paris, France
  - Designed deep learning-based source separation methods optimised for a classical music database.
  - Achieved significant improvements in separation performance over existing methods.
  - Adapted rapidly in a dynamic startup environment to deliver impactful solutions.

Education

- **École Polytechnique Fédérale de Lausanne** *Mar 2020 - October 2024*
  Ph.D. in Electrical Engineering Lausanne, Switzerland
  - Successfully defended on October 8th, 2024 (awaiting final grade).

- **University of Edinburgh** *Sep 2018 – Aug 2019*
  M.Sc. in Acoustics and Music Technology Edinburgh, UK
  - With distinction

- **ETH Zürich** *Sep 2013 – Sep 2016*
  B.Sc. in Physics Zürich, Switzerland
  - Grade: 5.15/6

- **Collège Calvin, Genève** *Aug 2009 – Jun 2013*
  German bilingual Swiss Baccalaureate Geneva, Switzerland
  - Grade: 5.5/6

Personal Interests

- **Music:** Play guitar, bass, and piano; enjoy producing music and jam sessions.
- **Hiking:** Regular hikes with my girlfriend and our dog.
- **Tennis:** Former tournament player, now enjoy playing for fun.
- **Chess:** Enthusiastic player with an online Elo rating of about 2000.
- **Mathematics and Physics:** Enjoy exploring new topics and revisiting concepts from previous studies.

PUBLICATIONS

C=CONFERENCE, J=JOURNAL

[J.1] A. Bittar and P.N. Garner (2024). **Exploring neural oscillations during speech perception via surrogate gradient spiking neural networks**. In *Frontiers in Neuroscience*, Vol. 18.

[C.1] A. Bittar, et al. (2024). **Improving vision-inspired keyword spotting using dynamic module skipping in streaming conformer encoder**. In *ICASSP 2024*, pp. 10386-10390. IEEE.

[J.2] A. Bittar and P.N. Garner (2022). **A surrogate gradient spiking baseline for speech command recognition**. *Frontiers in Neuroscience*, Vol. 16. Idiap Paper Award 2022.

[C.2] A. Bittar and P.N. Garner (2022). **Bayesian recurrent units and the forward-backward algorithm**. In *Interspeech 2022*, pp. 4137-4141.

[C.3] A. Bittar and P.N. Garner (2021). **A Bayesian interpretation of the light gated recurrent unit**. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2021*, pp. 2965-2969. IEEE.

SOFTWARE RELEASES

- SPARSE: Spiking Architectures towards Realistic Speech Encoding [] *Related Publication: J.1*
- SpArch: Spiking Architectures for Speech Technology [] *Related Publication: J.2*
- A Bayesian Interpretation of Recurrence in Neural Networks [] *Related Publications: C.2, C.3*

MISCELLANEOUS

- **Teaching Assistant** — *Apr 2020 – Aug 2022*
  Ph.D. student — Idiap Research Institute
  - Helped design the AI-master course on *Foundations in statistics for AI*.
  - Wrote and corrected python-based labs.

- **Physical Modelling of the Fender Rhodes Piano** — *May 2019 – Aug 2019*
  M.Sc. thesis — University of Edinburgh
  - Developed a physics-based computational model of the Fender Rhodes piano mechanism.
  - Applied finite difference schemes and Euler-Bernoulli modes to solve the system's dynamics.

- **Backpack Travel** — *Jan 2018 – Aug 2018*
  Gap period between civil service and M.Sc. — South America and Asia
  - Spent 3 months backpacking through South America to learn Spanish.
  - Traveled 3 months through Asia, including the length of Vietnam by motorbike from North to South.

- **Lattice Dynamics in Condensed Matter Physics** — *Feb 2016 – Aug 2018*
  B.Sc. thesis — ETH Zürich
  - Analysed lattice dynamics in the Shastry-Sutherland quantum magnet $SrCu_2(BO_3)_2$.
  - Used synchrotron inelastic X-ray scattering at the European Synchrotron Radiation Facility in Grenoble.

- **Simulation of Grass Growth and Pasture Yields** — *Sep 2016 – Dec 2017*
  Swiss civil service — Agroscope Changins
  - Worked in agronomy research group focused on production systems and animal health.
  - Modelled grass growth and pasture yields from weather data.
  - Publication: Simulation of grass growth and pasture yields with ModVege, *Recherche Agronomique Suisse, 2018*.

- **Tennis Instructor** — *2010 – 2012*
  Summer camps and weekly sessions — Drizia Miremont, Geneva
  - Supervising and teaching children aged 6 to 12 how to play tennis.

LANGUAGES

Spoken: French (native), English (bilingual), German (proficient), Spanish (elementary)

Programming: Python, PyTorch, Bash, Tensorflow, Keras, Matlab, R, C++