UNIVERSITY OF SUSSEX

Reservoir Computing in Phone Recognition

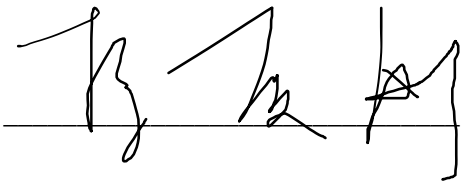
Candidate Number: 291139
Supervisor: Dr. Jeff Mitchell
MSc, Artificial Intelligence and Adaptive Systems
Department of Informatics
University of Sussex
Summer 2025
Word Count: 11526

Statement of Originality and Intellectual Property Rights

This dissertation is submitted in fulfilment of the requirements for the MSc in Artificial Intelligence and Adaptive Systems at the University of Sussex, and it has not been submitted, either in whole or in part, for other degree or diploma at this or any other institution of higher learning. Except where explicitly stated within the text, this dissertation is the result of my own independent work and investigation. All sources of information, quotations, and contributions from others have been properly referenced in accordance with academic standards.

I confirm that I have exercised reasonable care to ensure that the work is original, and does not, to the best of my knowledge and belief, infringe upon any copyright, intellectual property rights, or other rights of any third party.

The intellectual property arising from this dissertation remains the property of the author, subject to the regulations of the University of Sussex regarding the ownership and use of student research. Any use, reproduction, copy, or distribution of this dissertation and the code, in whole or in part, must be carried out in accordance with the University's policies and with proper acknowledgement of the source.

Signed:  **Date: 18 August 2025**

ACKNOWLEDGEMENTS

This project is completed with the help and support of my supervisor Dr. Jeff Mitchell. I would like to sincerely thank him for guiding me through this dissertation with his endless patience, consistent help, and valuable feedback. I would also like to thank Dr. Christopher Johnson who inspired me to work on Reservoir Computing application and provided me resources and literature that help establishing my knowledge on this topic. This project would not have been possible without the generosity and professionality of these people.

ABSTRACT

This study explores the potential of Reservoir Computing models (RCs) as large-vocabulary, continuous speech, speaker-dependent acoustic model in phone recognition. RCs emerged in the early 2000s as a unifying framework for Recurrent Neural Network models (RNNs) framework. It offers faster learning convergence with compact model sizes and fast training efficiency due to its structure of a fixed, dynamic reservoir and a light, trainable readout. In this paper, with a comparison to Long-Short-Term-Memory models (LSTMs), we examine RCs performance and efficiency in terms of training data size and parameters size. The results align with our central hypothesis that RCs are more compact and efficient than LSTMs, giving the model a boast in catering to scenarios where practitioners face constrains in both data availability and computational resources. We also explore Hierarchal Reservoir Computing models (HRCs), a multilayer-RCs, in phone recognition, and, contrary to our hypothesis, increasing the layer number does not induce a significant improvement than the increase in the reservoir size. Lastly, we examine how far RCs are from state-of-the-art of acoustic model, Wav2Vec 2.0 in terms of performances, and RCs' advantages in training data sizes and parameter sizes. This research provides a solid example for future work related to RCs and HRCs in speech recognition, particularly the architecture variation and hyperparameter optimisation in HRCs and Physical Reservoir Computing (PRC), which has been found yielding comparable results to Transformers.

TABLE OF CONTENT

ACRONYMS

Abbreviation	Definition
ASR	Automated Speech Recognition
BPTT	Backpropagation Through Time
BD-ESNs	Bi-Directional Echo State Network models
bi-LSTMs	Bi-Directional Long Short-Term Memory Models
CWR	Connected Words Speech Recognition
CTC	Connectionist Temporal Classification
CNNs	Convolutional Neural Network Models
DNNs	Deep Neural Networks
DCT	Discrete Cosine Transform
ESNs	Echo State Network Models
FFT	Fourier Transform
FACC	Frame Accuracy
HMMs	Hidden Markov Models
HRCs	Hierarchal Reservoir Computing Models
HTR	Hierarchal-Task Reservoir
IWR	Isolated Word Speech Recognition
LLMs	Large Language Models
LVCSR	Large-Vocabulary Continuous Speech Recognition
LPC	Linear Predictive Coding
LSMs	Liquid State Machines
LSTMs	Long Short-Term Memory Models
MFCCs	Mel-Frequency Cepstrum Co-Efficients
OOV	Out-Of-Vocabulary
PER	Phone Error Rate
PRC	Physical Reservoir Computing
PCA	Principal Component Analysis
RNNs	Recurrent Neural Network Models
RCs	Reservoir Computing Models
STFT	Short-Time Fourier-Transform
SSR	Spontaneous Speech Recognition
VAD	Voice Activity Detector
Wav2Vec2	Wav2Vec 2.0
WER	Word Error Rate

1 – INSTRODUCTION

Automated Speech Recognition (ASR) has received increasing attention due to the rise in the popularity of smart devices that utilise it. This increasing popularity and the increasing demand for its use has made ASR a popular research topic in machine learning. ASR is a process of identifying an acoustic input sequence $X = \{x_1, \cdots, x_T\}$ of length T as a label sequence $L = \{l_1, \cdots, l_N\}$ of length N (Wang et al., 2019). The labels are generally words, sub-word units, phones, or characters. Each input in the sequence $x_t \in R^D$, which is a D-dimensional speech input vector corresponding to the t^{th} speech frame. Each label l_n belongs to a set $\mathcal{V}$, giving us $l_n \in \mathcal{V}$. The task of ASR is to find the most likely label sequence $\hat{L}$ given X. Formally, it can be written as:

$$\hat{L} = \text{argmax}_{L \in V} \, p(L|X) \qquad (1)$$

Therefore, the essential work of ASR is to establish a model that can accurately calculate the posterior distribution $p(L|X)$. As a result, employing large machine learning models with hidden Markov models (HMMs), a statistic model, became the mainstream technology (Liao et al., 2013, Li et al., 2016). Besides HMM, researchers also explore a selection of classification methods (Zaman et al., 2023) and training technique in model parameter optimisation as well as different deep neural networks (DNNs) such as Recurrent Neural Network models (RNNs) (Medsker & Jain, 1999), Long Short-Term Memory models (LSTMs) (Cheng et al., 2016), and Convolution Neural Network models (CNNs) (O'Shea & Nash, 2015). Of all DNNs, LSTMs jump out due to their strengths in working on sequential data as well as fixing the vanishing gradient problem RNNs have (Hochreiter, 1998, Hu et al., 2018). In 2017, transformers were introduced in Vaswani et al. (2017) and they outperform most if not all existing DNNs. However, all these models rely on extensive training data with huge memory demands to ensure a satisfactory performance.

As the demand for greater computational power increases, researchers have been exploring alternative computational paradigms (Larger et al., 2017). In parallel, researchers have been exploring other solutions outside of Turing–von Neumann architectures. Therefore, Reservoir Computing models (RCs) (Verstraeten et al, 2007) attract much attention thanks to various physical implementations of the hardware with a surprisingly low cost in memory and training resources. Specifically, Echo State Network models (ESNs) in RCs generate complex temporal dynamics through the utilisation of the recurrent connections, and its low-power dissipation has been proven to be an intrinsic property of RC networks (Jaurigue et al., 2022). Furthermore, ESNs offer faster learning convergence and Zhou et al. (2020) found this character useful when applying ASR in wireless transmission systems; the model mitigates unknown non-linear radio frequency distortion. All in all, ESNs offer an efficient and interpretable approach for ASR tasks with fewer meta-parameters and high potential (Gauthier et al.).

Based on these predecessors' research, we raise these research questions:

> *1) How do RCs perform in ASR, and where does its efficiency lie in terms of training data size and parameters size.*
>
> *2) How do RCs perform in ASR using different architecture.*
>
> *3) How far do RCs exceed other models in the RNNs framework, e.g. LSTMs.*
>
> *4) How far away are the RCs from reaching the same performance as the state-of-the-art machine learning model, transformers (Wolf et al., 2020, Hannan et al., 2020).*

This paper is organised as follows: In Section 2, we review the related work of ASR. In Section 3 & 4, we explain the design of study, including the experiments' details and the methodologies. In Section 5 & 6, we present the results of the experiments and discuss the models' advantages and disadvantages. In Section 7, we show the possible future works of RCs in ASR.

2 – BACKGROUND & RELATED WORK

In this section, we discuss the related works of building different ASR systems. We first explain the standard systematic way for ASR systems classification, then we introduce the challenges in building each kind of ASR system. Thirdly, we describe the models used for developing ASR systems, including RNNs, LSTMs, and Transformers. Lastly, we introduce the related work of RCs in ASR tasks.

2.1 – Classification of ASR System

ASR system is established based on what kind of speech and output text we are expecting. This can vary depending upon the ASR application. For example, some ASR systems required to be capable of recognising speech in different accents and speaking styles if they have users from different regions. The type of application indicates which dataset, model, and training methods we choose for the development. Researchers provide a systematic way for classifying ASR systems, that they can be classified by the type of speech, the type of speaker mode, and the vocabulary size (J.Arora & Pal Singh, 2012; Ghai & Singh, 2012; Ibrahim & Varol, 2020). In this section, we describe related works of developing ASR systems in each category.

2.1.1 – Type of Speech

The type of speech utterances includes three categories: 1) isolated words, 2) connected words (interchangeable with continuous speech), and 3) spontaneous speech.

2.1.1.1 – Isolated Word Speech Recognition

An isolated word speech recognition (IWR) system deals with discrete utterance and accepts a single word/phrase. By definition in Ghai & Singh (2012), IWR system recognises a single word/phrase as a complete entity with clearly defined beginning and ending point. Kumar et al. (2012) develop an IWR system using HTK, a Hidden Markov Model tool, on a Linux platform. Other IWR System applications include phone repertory dialler where the system responds to isolated word inputs (Rabiner & Levinson, 1981), mental health consultation where the system retrieves information from databases based on keywords mentioned in speaker's voice information (Fu, 2020), and voice control application that gets triggered only when wake words like "Hi Alexa" or "Hi Siri" is detected.

2.1.1.2 – Connected Word Recognition

Connected words speech recognition (CWR) system, interchangeable with continuous speech recognition, accepts a sequence of words as input, and the words are separated by pauses that normally occur in speech. While IWR is a mature technology that has been developed for commercial use, CWR, as shown in the survey by Zweig & Picheny (2004) is still in progress and face multiple issues in particular recognising large sets of vocabulary. CWR can be applied to recognise various speech types, but it is trained on scripted speech data and therefore works best in formal settings. It started with commercial organisations like AT&T, BBN, IBM and academic institutions such as Carnegie Mellon University who work on English conversational telephone speech (Chen et al., 2006; G. Evermann et al., 2004) transcription and broadcast news transcription (Spyros Matsoukas et al., 2006). With the emergence of smart phone and more computationally powerful machine learning models, recent applications span from meeting transcriptions

to environmental sound identifier (Sharrab et al., 2025; Oh et al., 2021).

2.1.1.3 – Spontaneous Speech Recognition

Similar to CWR, spontaneous speech recognition (SSR) system accepts a sequence of words as input; however, the former is trained on scripted data, and the latter is trained on spontaneous (i.e. unscripted) speech. This means an SSR system handles a variety of natural speech feature such as exclamations, repetitions, and interjections, i.e. words or utterances for expressing spontaneous reactions, better then CWR. In Raza et al. (2018), they trained a model with audio data recorded by one single speaker with a mixture of script reading and spontaneous speech data. The suggested ratio of this mixture is 5:9 for words and 5:9 for phoneme.

2.1.2 – Type of Speaker model

The type of speaker model is defined by the specifications of the speaker's voice which can depend upon their physical body, personality, cadence, accent, and dialect. The ASR system can be either speaker model dependent or speaker model independent. Speaker model dependent ASR means that it is designed for a specific speaker type but can much less accurate for other speaker types. Speaker model independent ASR means that it can be generalised to a wide range of speaker types, e.g. it can recognise two sounds that come from the same word but spoken in different way of pronunciation.

2.1.3 – Vocabulary Size and Elementary Units

The complexity and applications of ASR decides the vocabulary size the system requires. The vocabulary size can span from small-size (tens of words), medium-size (hundreds of words), to large-size (thousands of words or more) (Gowda & May, 2020). In large-vocabulary continuous speech recognition (LVCSR) systems, the elementary units for speech sound modelling can be word-based, character-based, phone-based, syllable-based, and, for certain languages, sub-syllable based. Samy Bengio & Georg Heigold (2014) introduced word embedding that gives a representation for each of the words in the vocabulary, including out-of-vocabulary (OOV). However, in the survey of S. Joshua Johnson et al. (2023), they point out that word embedding's disadvantage is that the speech recognition system's dimensionality grows by the amount of vocabulary size. Yu et al. (2012) also agree that such massive memory requirement makes LVCSR not practically applicable as word-based units can lead to sparsity issue in training data. A phone-based model can overcome the memory demand issue as the total amount of phones are far less than full size of vocabulary. Furthermore, word-based models require more training data than shorter-length acoustic units such as phoneme-based units to reach to the same accuracy (Bhatt et al., 2020). However, phone-based model can suffer from contextual effects. Though using triphone as elementary units can overcome this problem, it can lead to data scarcity (Patlar, 2009). Using syllable system is also an alternative for reducing the contextual effects, but it does not apply to all languages as some languages are low-resource languages and are morpheme loaded (A. Sethy & Narayanan, 2003). In short, it is preferable to use universal phone sets as elementary units for ASR, especially multilingual speech recognition and low-resources languages (Burget et al., 2010).

2.2 – Challenges in ASR

Followings are the difficulties in establishing an ASR system:

2.2.1 – Ambiguity in Word Boundaries

One of the challenges of developing CWR and SSR systems is that it is difficult to infer from audio signal alone where the word boundaries are (C & Radha, 2012). It can add to the

difficulties when there is lack of pauses in the speech or when there is co-articulate affect, i.e. an effect describing how acoustic properties of phonetic units can be influenced by the adjacent sounds from the neighbouring phonemes. In Braga et al. (2005), they outline a set of linguistic rules for improving word-boundary segmentation in voice recognition systems. They suggest that a usage of statistic model like HMM complemented with the linguistic rules and phonologic tendencies and can help identify words in speech that lack pause boundaries.

2.2.2 – Homophones

Furthermore, when vocabulary size is large, CWR and SSR systems can face challenges when trying to distinguish homophones. Srivastava & Sitaram (2018) found that the effect of homophones includes giving misspellings due to the same sound, such as "benifit" and "benefit", and such spelling errors can increase the word error rate (WER). They also found that there are words between Hindi and English that sounds similar, such as "टफ" and "tough" and there are mis-identified instances due to the similarity of the sound, such as "colour's" and "colours". They hence propose a WX-based common pronunciation scheme, a code-switching technique for speech containing multiple languages or homophones. They list out 442 pairs of homophones from the corpus resulting from the above reasons and consider the pair as the same token in the training. This approach smoothens out the irregularities by identifying not only homophones but also spelling variants and errors. In Zheng et al. (2020), they use homophone-based label smoothing technique and reduce WER by 0.4%. The smoothing approach includes giving an over confidence penalty by KL divergence and a homophone unigram based prior distribution.

2.2.3 – Noises

Not all audio is recorded in studios, so the audio may contain unwanted sound in the background. This means that ASR system should be able to distinguish these noises from the speech signal. Another type of noises is the echo effect which stems from speech signals bouncing off surrounding objects and re-appearing in the recording. Li et al. (2014) give a survey of noise-robust techniques in the past 3 decades and summarise the solutions into five different criteria: 1) feature-do-main vs. model-domain processing, 2) the use of prior knowledge about the acoustic environment distortion, 3) the use of explicit environment-distortion models, 4) deterministic vs. uncertainty processing, and 5) the use of acoustic models trained jointly with the same feature enhancement or model adaptation process used in the testing stage.

2.2.4 – Realization

The same word can produce different acoustic waveforms each time it is spoken. To tackle this issue, Grivel et al. (2002) propose a Kalman filter-based speech enhancement approach based on Chen & Mehra (2008) study. It uses the estimation of the steady Kalman gain and the method successfully avoids using a voice activity detector (VAD), which is found producing silent frames in the estimation of the variance of the white noise.

2.2.5 – Speaker Variation

The same scripts can sound different when having different speakers. It can vary by the characteristics including speaker's sex, dialects, and speaking style. Females have a shorter vocal tract than males, therefore they typically have a pitch almost twice as high as male's. Furthermore, each person can have their own distinctive way of pronouncing words, and within the same individual emotion can affect their speech creating a large variation in pitch, tone, and volume. Lastly, speech can be

different due to regional or social dialect. Different social group of speakers can have distinctive accents. This can be crucial when deciding whether the ASR system should be speaker model dependent or independent. If the ASR system should be speaker model independent, developers should consider taking speaker variation into account when choosing their dataset for the training data.

Furthermore, M. Benzeguiba et al. (2006) point out that such speaker variation distinguishes the available audio data by its characteristics; and for the characteristic that have smaller population, there will be limited data resources available. Therefore, ASR systems, both speaker model dependent and speaker model independent, can struggle to analyse speakers from those populations.

2.2.6 – Text generation

One of the challenges in developing an ASR system is that it deals with not only audio input but also text output. For example, an ASR system for transcription services might prefer the text output to be a readable script that is case-sensitive and includes punctuations and spaces.

2.3 – Modules of ASR and Techniques

Modules identified for ASR systems are (Ghai & Singh, 2012; Bhatt et al., 2020; Ibrahim & Varol, 2020)

i. Speech Signal Acquisition
ii. Feature Extraction
iii. Acoustic Modelling
iv. Language & Lexical Modelling
v. Recognition

2.3.1 – Speech Signal Acquisition and Feature Extraction

First, ASR system acquires speech signal in raw waveform, then it selects useful features from the raw waveform and converting them into a sequence of fixed-size vectors (Lee et al., 2003).

Feature extraction methods include Principal Component Analysis (PCA), Linear Predictive Coding (LPC), Cepstral Analysis, Mel - Frequency Scale Analysis, Mel-Frequency Cepstrum Co-efficients (MFCCs) (Shrawankar & M, 2025; Kesarkar & Rao, 2003). The most common is the MFCCs feature set, which are seen in works including but not limited Trouvain & Hinaut (2021) and Loubser et al. (2024). The process is as following:

- Pre-emphasis: A pre-emphasis filter is applied to amplify the high frequencies in the signal. Since high frequencies usually have smaller magnitudes compared to lower frequencies, this filter can balance the frequency spectrum. The filter also helps the Fourier transform (FFT) operation later on and can improve the Signal-to-Noise Ratio (SNR) (Zhou, Odedeyi, et al., 2020).

- Framing: Since frequencies in a signal change over time, applying FFT across the entire signal can result losing the frequency contours of signal over time. Therefore, we do framing first in convenience for applying the FFT on each frame later (Figure 1). Studies show that typical frame size in speech processing range from 20ms to 40ms with 50% overlap between consecutive frames (Ghoraani & Krishnan, 2011). Popular settings are 25ms for the frame size, and 10ms frame stride (i.e. length of overlap).

- Window: Before FFT, a window function such as the Hamming window (Sulistyaningsih et al., 2018) is applied to each frame to reduce spectral leakage that could be made by FFT. The Hamming window follows the equation:

$$w[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$$

Where $0 \leq n \leq N-1$, N is the window length.

- Fourier Transform: This step applies a Short-Time Fourier-Transform (STFT) on each frame to get the power frequency spectrum. The equation is as follow:

$$P = \frac{|FFT(x_i)|^2}{N}$$

Where x_i is the i^{th} frame of signal x and N is typically 256 or 512.

- Filter Banks: This step computes, typically, 40 filters on a Mel-scale to the power spectrum. This step aims to mimic human's perception of sound by being more discriminative at lower frequencies than higher frequencies. This gives us a mel-scaled power spectrogram which is often referred as mel-spectrogram.

- Mel-frequency Cepstral Coefficients (MFCCs): This final step applies a Discrete Cosine Transform (DCT) to decorrelate the filter bank coefficients. Typically, in ASR, a resulting cepstral coefficient 2-13 are retained and the rest are ignored. One may apply sinusoidal liftering to the MFCCs to de-emphasise higher MFCCs which has been evidently shown to improve speech recognition in noisy signals (Miranda et al., 2019; Fernandes et al., 2018; Al-Imam, 2020).

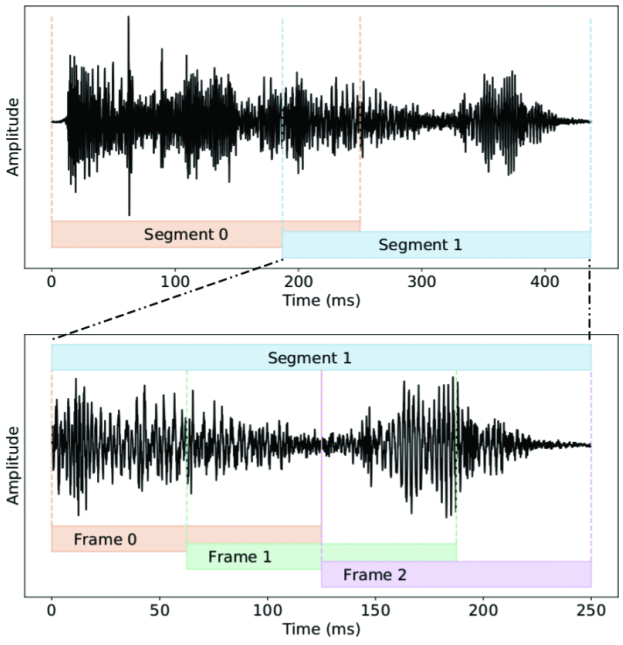


Figure 1. This figure is cited from Miranda et al. (2019), demonstrating how framing and FFT is done. The figure shows how each frame is extracted and how they overlap each other. In this paper, we set the frame size to 25 ms, which is ≈1024 samples for a sampling rate of 44 kHz audio data. The frame stride, i.e. the length of the overlapping, is 10 ms.

2.3.2 – RNNs and LSTMs as Acoustic Model

Acoustic model is one of the most crucial parts in ASR for processing acoustic features and recognising elementary units (usually phonetic based). HMM was one of the most commonly used statistical models until the advent of RNNs and LSTMs. The recurrent connections in RNNs captures the long-range dependencies between input data, making it a competitive model in nonlinear sequential processing tasks (Liu et al., 2016), including speech recognition (Deng et al., 2013). For example, Li, Zhou, et al. (2014) implement a two-layer RNNs on the Microsoft Research Sentence Completion (MRSC) Challenge, achieving an accuracy of 47%.

LSTMs fixes the vanishing gradient problem RNNs have while preserving the effectiveness in handling long-range dependencies in sequential data. LSTMs are comprised of memory cell, i.e. LSTM cell (Figure 2), using "cell state" and "hidden state" to carry information over time. The information is recognised and memorised by using the component called gates, and they are categorised as input gate, forget gate, and output gate. Each LSTM cell uses the following equations:

Input gate:

$$i_t = \sigma(W_i x_t + R_i h_{t-1} + P_i c_{t-1} + b_i) \quad (2)$$

Forget gate:

$$f_t = \sigma(W_f x_t + R_f h_{t-1} + P_f c_{t-1} + b_f) \quad (3)$$

Cell state:

$$c_t = f_t \odot c_{t-1} + i_t \odot \varphi(W_c x_t + R_c h_{t-1} + b_c) \quad (4)$$

Output gate:

$$o_t = \sigma(W_o x_t + R_o h_{t-1} + P_o c_t + b_o) \quad (5)$$

LSTM Cell output:

$$h_t = o_t \odot \varphi(c_t) \quad (6)$$

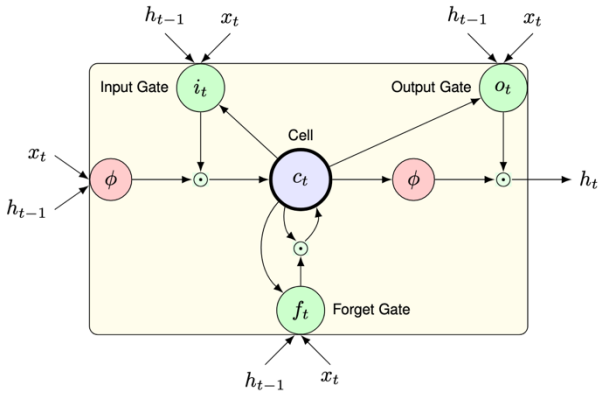


Figure 2. This figure (cited from Billa, 2017) demonstrates how each LSTM memory cell works. As show in the equations, each of the LSTM cell has an input gate, an output gate, and a forget gate. Each of the gate takes the input data and previous hidden state into calculation. The LSTM cell produce the final output h_t with a use of tanh and sigmoid function.

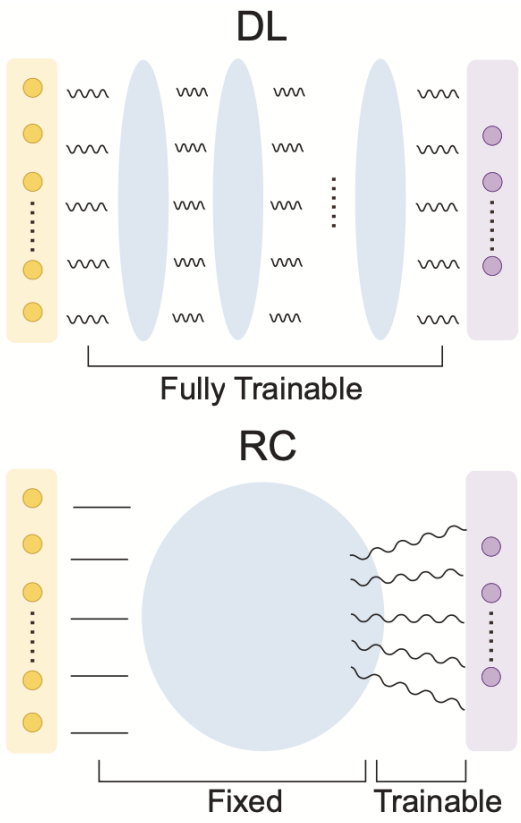


Figure 3. This figure (cited from Yan et al., 2024) demonstrates a comparison between a Deep Learning (DL) model and a Reservoir Computing (RC) model. Both models take inputs, learn from the features, and transform them (nonlinearly) to match desired outputs. However, in DL, all parameters are fully trainable, whereas in RC, only the readout is being trained. This structure indicates that RC usually has fewer parameters than DL. Less parameters allow RC to have the flexibility to apply on many devices. In terms of training, many methods, such as backpropagation (BP), stochastic gradient descent (SGD), are used in DL; however, in RC, the training uses only a simple regression, e.g., linear regression, Lasso regression and ridge regression.

The input vector xt at time t is passed to input gate (Eq. 2), forget gate (Eq. 3), and output gate (Eq. 5). Each gate has three different weight matrices in the computation: W, R, and P. W are the weight matrices that connects the input, and R are the recurrent weight matrices connecting to the previous hidden state; and P are diagonal peephole weight matrices connecting to the previous cell state. The sum of these values is added with a bias vector b and passed through a sigmoid function σ. The final output of the LSTM cell uses a φ hyperbolic tangent function to add the nonlinearities. Operator $\odot$ represents the point-wise multiplication of two vectors. If there is a second layer of LSTM cells processing the sequential data in a reverse way, we called the two-layered LSTMs bi-directional LSTMs (bi-LSTMs) (Siami-Namini et al., 2019). The output of the model would be the concatenation of the output from the forward and the backward direction layers:

$$yt = [\overrightarrow{ht}, \overleftarrow{ht}] \quad (6)$$

Many ASR studies have adopted bi-LSTMs in ASR tasks. Billa (2017) performed ASR using a deep bi-LSTM, which a bi-LSTM stacked with multiple bi-directional layers. An empirical ASR study of Connectionist Temporal Classification (CTC) acoustic models of Miao et al. (2016) uses bi-LSTMs with and additional trained CTC, resulting a 3.9 – 4.6% reduction in WER comparing to hybrid LSTM/HMM. Li et al. (2015) presents a new deep LSTM model called F-T-LSTM for ASR. Instead of having the layers process input data in an opposite manner, some layers perform recurrence along the frequency axis of the speech waveform while the others perform along the time axis of the speech waveform. F-T-LSTM obtained a 3.6% in WER reduction from a feedforward neural network.

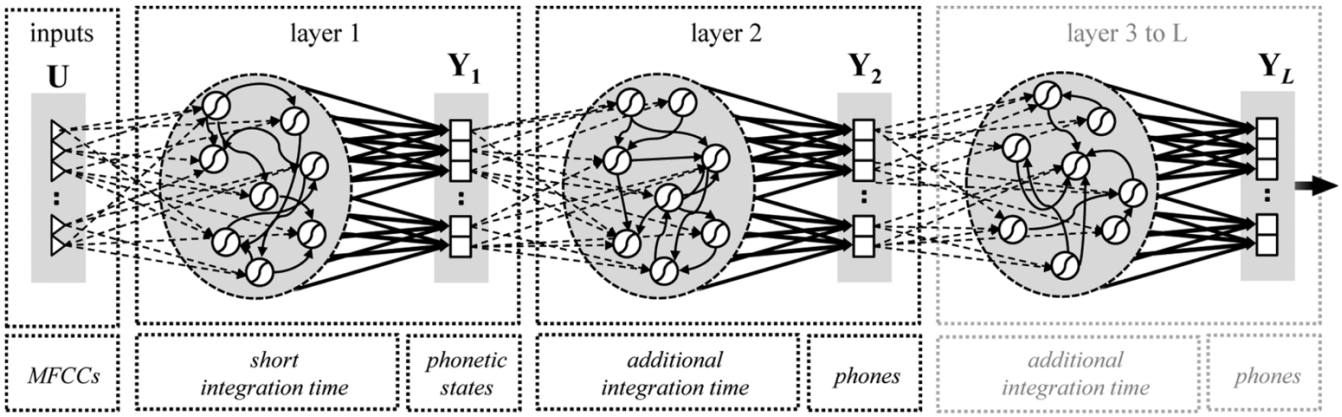


Figure 4. This figure (cited from Triefenbach et al., 2013) illustrates a multilayer RC using hierarchal architecture. The pre-processed audio input is transformed to MFCCs and fed to the first layer of the RC. The second layer takes the output of the first layer and pass on to the next layer. Each layer is an ESN consisting of a reservoir and a readout layer.

2.3.3 – RCs as Acoustic Model

Reservoir computing models (RCs) (Nakajima, 2020) emerged in the early 2000s as a unifying framework for RNNs, with the introduction of Echo State Networks (ESNs) by Jaeger (2006) and Liquid State Machines (LSMs) by Maass et al. (2002). ESNs preserve the character of RNNs in their reservoir; through fixed, randomly initialised recurrent connections, the reservoir generates diverse temporal dynamics (Figure 3). The reservoir is connected to a trainable output layer, mapping the reservoir states to the target output. As the DNNs such as transformers like BERT (Koroteev, 2021) emerged with good performance in ASR, Pedrelli & Hinaut (2022) argue that comparing to backpropagation through time (BPTT) used in DNNs, the dynamic connection in reservoir is similar to the brain's mechanism in speech processing (Kröger et al., 2009). Furthermore, feedforward and feedback connections between reservoir and readout layer simulate the connections between the V4 and V2 areas in the brain (Markov & Kennedy, 2013). RNNs internal architecture models how brain processes time-series information and develops temporal dependencies of the input (Lin et al., 1998; Güçlü & van Gerven, 2017). Recently, RCs have achieved results as good as, or even better than, n-gram models, HMM models, and LSTMs. Thrun et al. (2004) used ESNs in nonlinear signal processing applications, and Mikolov et al. (2011) shows that an RC architecture is competitive with the standard feedforward architecture with a 3% lower WER than a backoff 5-gram model. Triefenbach et al. (2025) use a multilayer RC architecture (Figure 4), consisting of 3 reservoirs with 20,000 nodes each in phone recognition, and achieve a phone error rate (PER) of 2% lower than an LSTM+CTC model (Graves & Schmidhuber, 2005). Trouvain & Hinaut (2021) use ESNs for canary song decoding and achieve 22% lower WER than a single layer LSTM.

Different reservoir-based models have been used as an acoustic model in phone recognition task. The standard architecture is a single-layer reservoir network consisting of one reservoir and one readout (a.k.a single ESNs). Multi-reservoir architecture has been used as well. For example, in Triefenbach et al. (2013), they use a bi-directional ESNs (BD-ESNs) that contains two reservoirs which one processes the input in a chronological order and the other processes them in a reverse order. Furthermore, in an empirical analysis by Gallicchio et al. (2017), they found that a deep-layered RCs, which has multiple reservoirs connect to input and output layer in parallel manner, allows an effective diversification of temporal representations in the computation, and amplify the richness of the activation dynamics between the recurrent units. Instead of having

reservoirs placed in parallel inside one layer, Triefenbach et al. (2025) performs acoustic modelling using a hierarchal architecture by cascading multiple ESNs (HRC) (Figure 4). They argue that the hierarchal architecture is a more efficient framework in signal processing when compared to DNNs using BPTT. The hierarchal architecture models the hierarchical structures in brain, for instance, a visual stimulus will go through the primary visual areas one after another through LGN, V1, V2, V3, V4, and so on. A similar hierarchy is found when the brain process audio signals, which starts from primary auditory areas (Kaas & Hackett, 2000).

2.3.4 – Transformers as Acoustic Model

Attention based models such as transformers from Vawswani et al. (2017) has become the state-of-the-art of ASR models. Transformers was first developed as a language model for machine translation and text generation, and they quickly surpass RNNs and CNNs. Therefore, transformers receive high attention in ASR applications. In Min & Wang (2023), they investigate the potential of using transformers-based large language models (LLMs) for ASR and improve WER by leveraging transcription accuracy. Despite initial experiments resulting in higher WER, the study sheds light on utilising LLMs for ASR applications. In Kubo et al. (2022), they use pretrained transformers LLMs for developing an end-to-end ASR system, which integrates acoustic model and language model into one ASR module. Through this integration plus transfer learning, they also attempt to address the data hunger issue.

Transformers uses an encoder-decoder architecture with a self-attention mechanism. The encoder finds the representations for the input and the decoder generates a sequence based on the input along with those representations. The encoder itself can be used with a CTC for sequence classification tasks, and the encoder-decoder together can be used for sequence-to-sequence (Seq2Seq) mapping or text generation. One of the downsides of CTC head is that it outputs individual characters from the hidden states, and therefore can have spelling mistakes in the transcriptions. One way to improve this audio transcriptions quality issue is to use an external language model as a spellchecker on top of the CTC output.

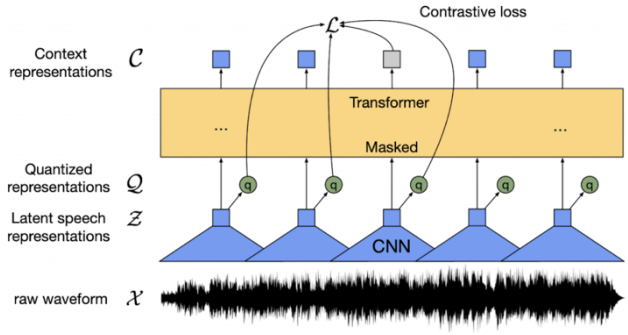


Figure 5. This figure (cited from Nasersharif & Namvarpour, 2024) shows the architecture of Wav2Vec2. Wav2Vec2 is pretrained with more than 50,000hours of unlabelled speech data. Similar to BERT, it learns contextualised speech representations by randomly masking feature vectors before passing them to a transformer network.

Popular encoder-only models in the pre-training/fine-tuning paradigm for speech include Wav2Vec 2.0 (Wav2Vec2) (Figure 5), HuBERT, and XLSR. Basevski et al. (2020) introduces Wav2Vec2 and achieves a WER of 1.8% on LibreSpeech test-clean data. In Baevski et al. (2021), a 3.8% of WER is achieved using a Wav2Vec-Unsupervised and a 4-gram language model trained on the same LibreSpeech dataset. Wav2Vec-Unsupervised contains 24 transformer blocks, 16 attention heads, and 317 million parameters. Later, Conneau et al. (2021) introduces XLSR, a cross-lingual speech representation model created from Wav2Vec2 and finetuned with Mozilla Common Voice Data (Hintz & Siegert, 2024). XLSR largely improved the computational power and training data hunger as well as reducing WER drastically. With an additional 4-gram language model, XLSR produces an average WER of English common voice testing data of 7.6% over 53 languages. Babu et al. (2022) later

introduces a new transformer model XLS-R, containing 2 billion parameters and pretrained on 436k hours of data from over 128 languages. For English, however, it does not surpass Wav2Vec2. It achieves a 5.6% WER on LibriSpeech test-clean dataset, which is worse than Wav2Vec2 model's results.

3 – RESEARCH AIM & EXPERIMENTS

In this project we aim to develop an ESN-based ASR system that uses large-vocabulary, continuous speech and speaker-dependent model. This means we do not take different dialects and speaker styles into account. We do not consider the robustness of the model in noises, nor the model ability in recognising interjections. The model is English language focused, and we solely focus on the performance in predicting the correct English labels from the clean, scripted audio data. Therefore, we use LibreSpeech dataset which are audiobooks recordings done in studios by one female speaker with an American accent.

The project aims to complete the first three modules in a standard ASR system, which is Speech Signal Acquisition, Feature Extraction, and Acoustic Modelling. The acoustic model predicts the phoneme from the audio signal, and we use PER (which is calculated in the same way as WER but treating phones as words) and confusion metrics as evaluation metrics.

To answer the four research questions (listed in Section 1), the models we choose to compare in the experiments are single ESNs, HRC, LSTMs, bi-LSTMs, and Wav2Vec2. One approach with HRC would be to work with longer sub-word units in the higher layers (e.g., phones in the 1st layer and syllables in the 2nd layer), but in this work, we stick to working solely on phones.

There are 4 experiments in this project:

1. In the first experiment, we compare the performance of single ESNs and single LSTMs to a function of training data size. We examine how ESNs perform in terms of data size efficiency. We also examine how the two models perform differently when both are models in RNNs framework.

2. In the second experiment, we compare the performance of single ESNs, LSTMs, and bi-LSTMs in phone recognition to a function of parameter size. We examine how ESNs perform in terms of parameter sizes (tuned by reservoir sizes), and whether they exceed LSTMs when two model types have similar parameter sizes.

3. In the third experiment, we explore multi-reservoir ESNs using a hierarchal architecture, the HRC. We explore their performance in phone recognition when multiple reservoirs in different sizes are used.

4. In the fourth experiment, from the ESNs used in the first three experiments we pick those that had the best results and compare them with an existing work using Wav2Vec2 in phone recognition. This gives us an understanding of how ESNs perform comparing to the state-of-the-art of ASR models. In the Discussion section, we delve deep into other RC-based models and explore its potential of achieving the same results as the state-of-the-art of ASR models.

4 – METHODS

In this section, we explain the algorithms and models used in this project. Firstly, we introduce the dataset used for model training and evaluation, and the reason we picked this dataset among all ASR datasets. We also give an understanding of what the dataset includes, and how we pre-process the audio signal and the annotations. We also present the models used in this project along with the architecture design and the hyperparameters. Lastly, we explain the evaluation metrics we use.

4.1 – About Dataset

This project uses LibriSpeech dataset (Panayotov et al., 2015) for model training and testing. LibriSpeech is an ASR corpus containing 1000 hours of read English speech derived from audiobooks made in Project Gutenberg (Gerlach & Font-Clos, 2020). The 1000 hours of speech is sampled at 16 kHz, including three train subsets: 100 hours (train-clean-100), 360 hours (train-clean-360) and 500 hours (train-other-500). We use the train-clean-360 data with a 4:1 of train/test split. Audio data is provided in .flac format. Each audio file is approximately 15 seconds long.

Every audio file has a corresponding .TextGrid file that provides the annotations of the audio. The annotation includes the absolute timestamps and phonetic labels, generated from Montreal Forced Aligner. The alignments suggest how each phonetic label delimits the audio. The annotation contains data fields that include, firstly, "text" which is the type of phone spoken, secondly, "xmin" which is the start time of the phone spoken in seconds, and lastly, "xmax" which is the end time of the phone spoken in seconds (Figure 6).

Typically, the conventional agreement is that the phonological system of the English language is composed of 44 phonemes, of which 24 are consonants and 20 are vowels (Bizzocchi, 2017). However, employing different definitions of phoneme, phone, and allophone can result 39 to 44 phonemes in English phoneme inventories. The LibriSpeech annotations includes 71 phonemes deriving from 39 English phonemes with stress marking (e.g. there are *AE0*, *AE1*, *AE2* which all suggest the sound of phone *AE* with different stress).

The dataset was selected due to its appropriate size, consistent style, and comprehensive annotations. For this project, we are not considering noises or varieties of speaking style, and we solely focus on one language – English; therefore, using an English audiobook recording done by a single female speaker reading from a script in a studio as training data is suitable for this project.

```
intervals [1]:
        xmin = 0.000
        xmax = 0.200
        text = "sil"
intervals [2]:
        xmin = 0.200
        xmax = 0.300
        text = "CH"
intervals [3]:
        xmin = 0.300
        xmax = 0.390
        text = "AE1"
intervals [4]:
        xmin = 0.390
        xmax = 0.480
        text = "P"
intervals [5]:
        xmin = 0.480
        xmax = 0.570
        text = "T"
intervals [6]:
        xmin = 0.570
        xmax = 0.730
        text = "ER0"
```

Figure 6. This figure is a screenshot of one of the .TextGrid files. The columns include "xmin", "xmax", and "text" which represents, respectively, the start of time (in seconds) of the phone spoken, the end of time of the phone spoken, and the type of the phone spoken (the phonetic label).

4.2 – Data Preprocessing

4.2.1 – Audio Data Preprocessing

Firstly, we resample the data from 16 kHz to 44 kHz. Figure 7 shows a log-mel spectrogram of a randomly selected interval from one of the samples in the dataset. This log-mel spectrogram visualises the amplitudes of the individual frequencies in a signal. Notice that the amplitude values are negative. The decibel scale for digital audio signals is re-scaled, and 0 dB is the loudest sound possible. Particularly,

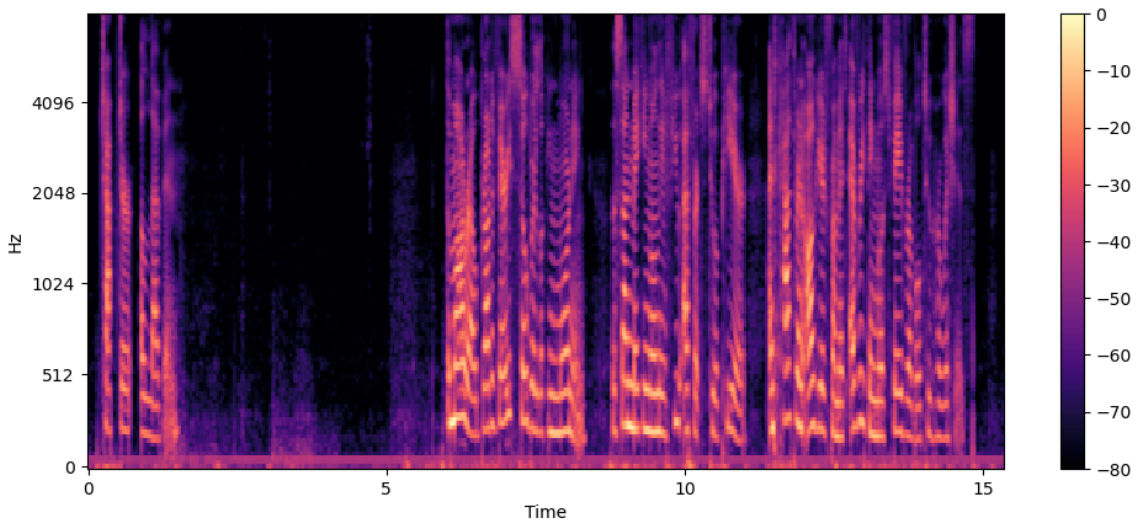


Figure 7. This figure is a log-mel spectrogram of one of the audio sequences. The y-axis is the frequency, and the x-axis is the timeline (in seconds). The colour represents the amplitude where 0 dB is the loudest. The large segment between 2s and 5s is pitched black, which is when pauses between words happened. Anything below -60 dB is generally inaudible to human. The spectrogram shows that below 200 Hz are mostly noises.

every -6 dB is a halving of an amplitude, and anything below -60 dB is generally inaudible to humans. Generally, a sampling rate of f sets an upper frequency limit of $0.5f$. Furthermore, human hearing is theoretically capable of perceiving a 20 Hz to 20,000 Hz frequency range. Therefore, a sampling rate of 16,000 gives an upper frequency limit of 8,000, which can filter a large number of audible signals. This is the reason why we up-sampled the audio data. However, if up-sampling to too high a frequency, it can take up too many resources and slow down the model training. Therefore, an up-sampling to 44,000 Hz is an appropriate compromise.

Secondly, we convert the raw waveform into MFCCs using standard MFCC analysis (Davis & Mermelstein, 1980) with Librosa Python library (McFee, 2015). The analysis involves extracting audio log-mel spectrograms using STFT with a hop length (often called frame stride) of 10 ms, using a hanning window to reduce edge effects. The length of the frame is 25 ms, which is ≈1024 samples for a sampling rate of 44 kHz audio data. Each frame is windowed by a window length of 10 ms where the left- and right-side of the window is padded with zeros. The log-mel spectrogram is filtered by frequency range of $[200, 8000]$ Hz with Mel filterbank set to 128 filters. This frequency range is based in the observation of audio patterns, that occurring below 200 Hz are mostly noises (Figure 7); also, an upper limit lower than 8000 produces empty filters in mel frequency in the STFT.

The filter bank coefficients in the log-mel spectrogram computation are highly correlated, and this can be problematic in machine learning algorithms. Therefore, we apply DCT to decorrelate the filter bank coefficients and yield a cepstral representation of the filter banks. Based on the literature review in Section 2, we found that 13 cepstral coefficients are usually fed to MFCC-based speech recognition models. We also computed the first Δ and second Δ derivatives of the MFCC signal, in order to provide the models with gestures dynamics. Lastly, we apply sinusoidal liftering to the MFCCs to de-emphasise higher MFCCs to improve speech recognition in noisy signals.

All in all, this pre-processing generates a processed data sliced in frames with each frame in length of 25 ms with 10 ms of 39-dimensional feature vector. Each frame represents an interval of pre-processed audio signal along with one corresponding phonetic label (Figure 8).

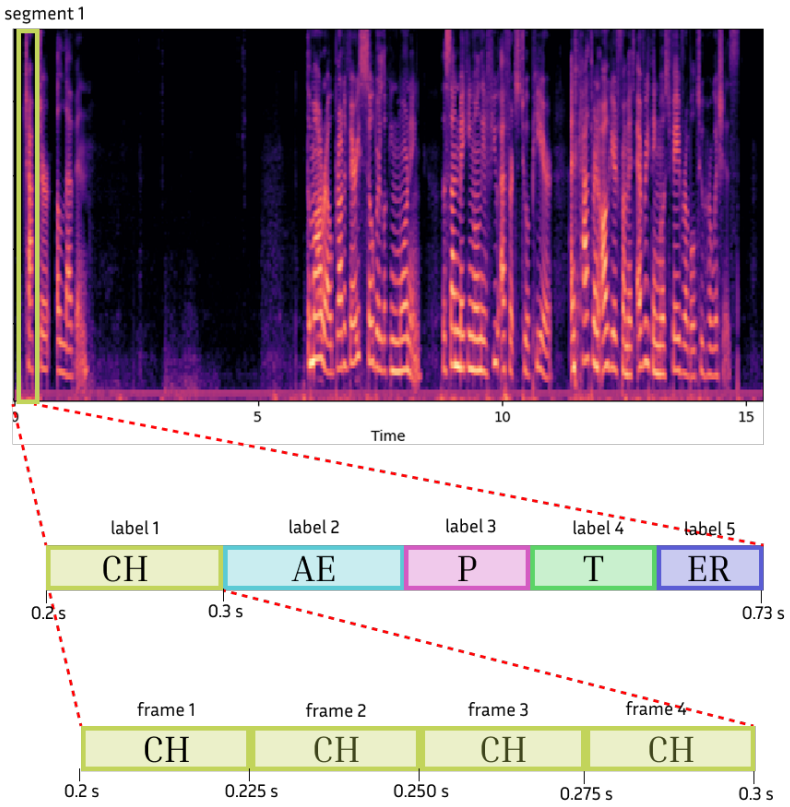


Figure 8. This figure demonstrates how the label annotation is done in alignment to audio data in the log-mel spectrogram. As demonstration, the graph displays how labelling is done to a randomly selected segment (segment 1). Segment 1 starts at 0.2s and ends at 0.73s, which is as long as the word "chapter" is being spoken. The phonetic labels are marked in coloured bars, denoted as "CH", "AE", "P", "T", and "ER". As described in Section 4.2.1, each frame size is 0.025. Therefore, the four frames in time of label "CH" being spoken are all labelled "CH".

4.2.2 – Phonetic Labels

Studies have shown that adding variants to the lexicon does not lead to model improvements in phone recognition, particularly when the pronunciation variation is not apparent in the grapheme (Wester, 2003). We therefore disregard the stressing in phones, resulting a downsize from 71 phonetic labels to 41 phonetic labels, including 39 English phonetic labels, 1 label for silence, and 1 label for OOV.

After the audio signal preprocessing, we have the audio signal sliced into multiple frames. Each frame requires a phonetic label; we therefore annotate the consecutive frames that share the same annotation label (Figure 8). In Figure 8, coloured bars represent the phonetic labels, and their lengths align with the timestamps of the audio signal.

4.3 – Phone Recognition Task and Models

The aim of this project is to evaluate RCs' performance in phone recognition as well as its efficiency in terms of training data size and parameter size in a comparison of other models, including LSTMs, a model in RNNs framework, and Wav2vec2, a transformers-based pretrained model. There are four experiments in this paper: firstly, we explore single ESNs' performance to a function of training data size in a comparison of single LSTMs; secondly, we explore single ESN model's performance in phone recognition to a function of its reservoir size in a comparison of LSTMs and bi-LSTMs; thirdly, we explore multi-layered RC models, the HRC, and their performances; lastly, we compare ESNs with Wav2vec2 in terms of performances in phone recognitions and parameters sizes.

4.3.1 – Reservoir Computing Models

RC is a framework for RNNs training; and ESNs and LSMs are the two most popular models in this framework. In this paper, we use ESNs in all the RC-based systems. ESNs have two components: a reservoir and a readout. A reservoir is a pool of randomly connected nodes. They are similar to a vanilla RNN yet the connections inside the reservoir are not trained; they remain as they are after randomly initialised by several hyperparameters which we will explain in Section 4.3.2.

The random high dimensional activation vector of the reservoir is then fed to a single layer of neurons called readout. The readout layer oversees the decoding of reservoir activations in order to perform the main task. The training of ESNs focuses on training the connections between the readout and the reservoir. In consequence, popular learning algorithms seen in DNNs such as BPTT is replaced by simple linear regression such as a regularized ridge regression.

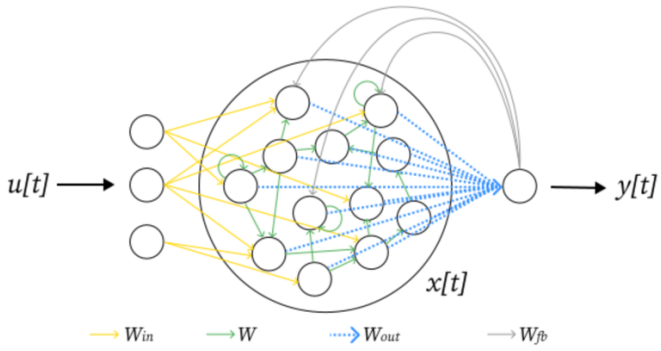


Figure 9. This figure (cited from the user guide of ReserviorPy, 2021) summarizes what an ESN looks like. Different type of connection weights are arrows in different colour. Yellow arrows are the inputs-to-reservoir connection weights (i.e. Input Connection weights). Green arrows are recurrent connection weight inside the reservoir (i.e. Reservoir Connection weights). Blue arrows represent the trained output weights of the readout (Output weights). Grey arrows represent the additional connections feeding the readout outputs back to the reservoir (Feedback weights).

Activation of the readout is fed back to the reservoir. This is called a feedback connection. Feedback connections aim to introduce nonlinearity to the model by integrating past information into current reservoir dynamics (Monomi, 2025). Figure 9 summarizes how neurons inside an ESN works and interacts. Connections between neurons are stored as Numpy arrays or Scipy sparse matrices. The following equation describes the states (i.e. activities) of these neurons:

$$x(t+1) = (1-a)x(t) + af(bW_{in}u(t+1) + Wx(t) + W_{fb}y(t)). \quad (7)$$

where x is the internal state at time step t and f is the neuron activation function (usually a $tanh$ function or a logistic sigmoid); u is the input signal, and y is the output signal; W_{in} and W are weight matrices giving to the connection from inputs and within internal neurons respectively; W_{fb} is the output feedback weight matrix for the connections from output back to the reservoir. a and b are the hyperparameters that control the initial state of the reservoir. a is the leaking rate whereas b is input scaling acting as a coefficient applied on input weight W_{in}.

In this project, we use a single ESN and a hierarchal ESN for the experiments. A single ESN model is a model with solely one reservoir and one readout. For HRC, we stack two identical ESNs which one takes the audio data as input and the other take its output and produce the final output. All RC systems have an output of 41 probability values for each of the phonetic labels, and the prediction is the label that has the highest probability. The number of nodes decide the reservoir size, and they are initialised in accord to each experiment purpose. The other hyperparameters are initialised in accord to the results of the optimisation experiment in the next section.

4.3.2 – Hyperparameters and Reservoir Optimisation

As explained in Section 4.3.1, connections inside a reservoir are not trained but fixed by a set of hyperparameters. The hyperparameters for the reservoir include but are not limited to

units, leak rate lr, spectral radius sr, reservoir connectivity RCo, input connectivity ICo, and feedback connectivity FCo. Units define the number of nodes in a reservoir. lr controls the memory flow of the reservoir between its internal states, and lr must be $[0,1]$. A low lr allows previous states to be remembered more, while $lr = 1$ prevents all in the previous states to "leak" into the current states. In other words, lr has a positive correlation to the recall of previous states. sr defines the maximum absolute eigenvalue of the reservoir weight matrix W. Larger sr implies slower decay of impulse response and longer-range interactions, and smaller sr implies a smaller effect of initial conditions. A high sr gives chaotic dynamics in the reservoir, and low sr ensures more stable dynamics. RCo defines the density of the reservoir internal weight matrix, i.e. the inter-connectivity between neurons in the reservoir. ICo defines the connectivity of input neurons and reservoir neurons, and FCo defines the connectivity of the feedback neurons and the readout. The readout has only one hyperparameter ridge that defines the regularisation coefficient for ridge regression.

The optimisation of the reservoir hyperparameters in this project is done through using hyperopt optimisation tools from ReservoirPy. Similar to most estimators used in the majority of machine learning tasks, the optimisation algorithms are based on the hypothesis of convexity. We expect to reach one of the local minima by shifting the parameters. We explore 6 hyperparameters including lr, sr, RCo, ICo, FCo, $ridge$. We define a specific range for each of the parameters and use a random search algorithm to find the optimised values. We run two experiments with 3 hyperparameters being searched in each as optimising all parameters in one experimentation can lead to unpredictable interactions between the parameters, making the local minima search longer and more tedious. We use single ESNs with 1000 neurons in the reservoir, and they are run on 30 batches of the pre-processed data with a train/test split of 4:1. We use normalised root mean squared error metric as loss function and coefficient of determination R-squared (R^2) as our evaluation metric.

Table 1 and Figure 10 show the results of the hyperparameter exploration, and we found that leak rate, input connectivity, and ridge have the highest correlation to the performance. We jointly optimised lr and $ridge$ and found that $lr = 0.2$ and $ridge = 1e-5$ yield the best performance. Furthermore, jointly with those two optimised parameters, $ICo = 0.8$ yields the best performance but for $ICo = (0.5, \dots, 0.8)$ and for $lr = (0.2, \dots, 0.4)$, the performance is quite stable. The rest of parameters doesn't have significant influence on the performance, and we set sr to 0.4, RCo to 0.1, and FCo to 0.1 as an initialisation for the rest of the project.

The optimisation and the training of the reservoir in this project is based on Eq. 7**,** and its recurrent weights are randomly drawn from statistical distributions. Both the input weights and the feedback weights emerge from a Bernoulli distribution, and the recurrent weights are from a Gaussian distribution.

Hyperparameter	Best Value
Leak Rate (lr)	0.2
Spectral Radius (sr)	0.4
Ridge ($ridge$)	1e-5
Input Connectivity (ICo)	0.8
Reservoir Connectivity (RCo)	0.1
Feedback Connectivity (FCo)	0.1

Table 1. This table summarises the hyperparameters for all ESNs instance in this paper. The hyperparameters that are highly correlated to the performance, according to the experiment, are leak rate, ridge, and input connectivity.

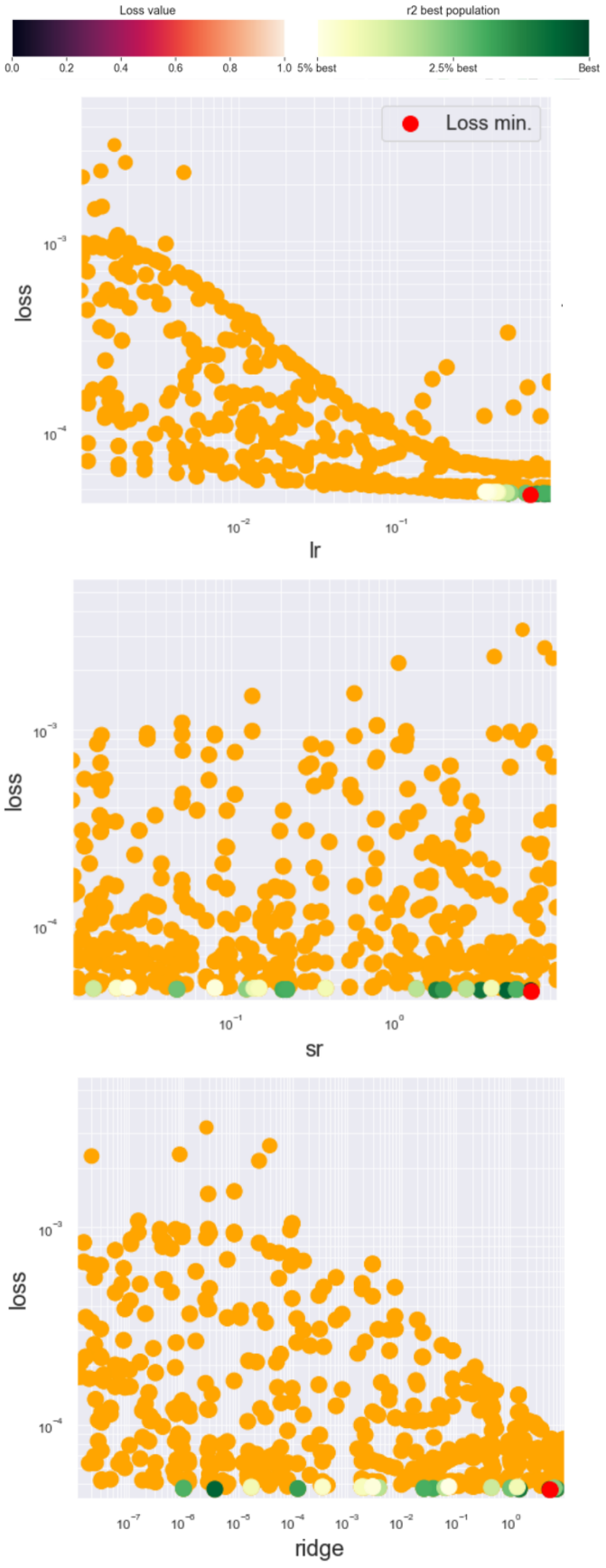


Figure 10. This figure shows the correlation between loss and hyperparameters SR, LR, and Ridge. Red dot indicates the value with minimal loss, and the shade of green represents how optimised the value is. Different combination of hyperparameters can influence the performance; therefore, the best value is not necessarily the value with least loss.

4.3.3 – LSTM Models

We use a single LSTM and a bi-LSTM in the experiments. For the single LSTMs, the model consists of one LSTM layer and a fully connected output linear layer. The LSTM layer has a hidden dimension of 64 input data, and as mentioned Section 4.2, data is pre-processed with an audio feature of 39 and phonetic label size of 41. The audio input data is arranged into a sequence of frames (with a dimension of 39), and a sequence in length of approximately 15 seconds. All sequences are padded to the same length. The LSTM takes an input vector of dimension (batch size, sequence length, number of labels) and produces an output vector in the same dimension. The bi-LSTM model used in the 2nd experiment has the same initialisation except having two bi-directional LSTM layers. Both the 1st and 2nd experiment have a learning rate of 0.01 and an Adam optimiser. The 1st experiment has a loss function of Negative Log Likelihood Loss Function with a training epoch of 10. In the 2nd experiment, Cross Entropy Loss Function is used as loss function and a training of 30 epochs is done for both single LSTMs and bi-LSTMs.

4.4 – Evaluation Metrics

The evaluation metrics adopted in this project are confusion metrics and PER. According to McCowan et al. (2004), an ideal ASR evaluation metric should be: (i) Direct; the measure should provide direct measures of the components in the systems, (ii) Objective; the measure should be generated in an objective, automated manner, and should not be resource-intensive nor application-dependent, (iii) Interpretable; the measure should give an absolute value that is intuitively related to system performance, and (iv) Modular; the measure is available to be generalised through applications.

Confusion matrix is one of the popular evaluation metrics used in natural language processing tasks as it meets all four

requirements (Prasanna, S. R. Mahadeva et al., 2025). Basically, a phone recognition task is to take in a segment of audio signal and tag the segment with the correct phonetic label. As a sequence classification task, a confusion matrix is an appropriate measure to evaluate the classifier's classification performance. It gets the ratio of the sum of principal diagonal values to the total sum of values in the confusion matrix, giving scores including an Accuracy Acc, a Recall R, a Precision P, and an F1 score F_1. Each of them is calculated as follow:

$$Acc = \frac{TP+TN}{N} \quad \text{Accuracy (8)}$$

$$P = \frac{TP}{TP+FP} \quad \text{Precision (9)}$$

$$R = \frac{TP}{TP+FN} \quad \text{Recall (10)}$$

$$F_1 = \frac{2PR}{P+R} \quad \text{F1 score (11)}$$

Where TP is the count of True Positive, and TN is the count of True Negative; FP is the count of False Positive, and FN is the count of False Negative. N is the total number of predictions.

PER comes from WER which is one of the standard evaluation metrics for automatic speech recognition systems. It provides a measure for quantifying the differences between model predictions and reference text. The calculation of WER involves observing each isolating word w generated within well-terminate audio input signals $a(t)$, and categorising their difference between the reference w' into three: Insertions I, Deletions D, and Substitutions S. The expected WER over whole audio signals a is calculated as

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

Where N is the total number of words in the reference. In our case, it would be the total number of phonetic labels in the reference.

In this project, we evaluate the model outcome at an audio-to-frame level and audio-to-phrase level. For frame level, we find the accuracy score and F1 score for each frame and get the average for each as a final score. Specifically, each frame is annotated with a corresponding phonetic label (Figure 8), and we check if the prediction matches the target. We then average the accuracy of each segment to get the frame accuracy score (FACC) and F1 score. In other words, conceptually, FACC and F1 score are used to measure the bars' alignment in length and colour in Figure 8 between the targets and the predictions.

Besides evaluating the transduction at frame level, we also evaluate the performance at an audio-to-phrases level. We reconstruct the predictions into a sequence of phonetic labels and compare them with the correct labels. We refer to this measurement as phone error rate (PER) as they are conceptually and mathematically the same as WER but in a context of evaluating the error rate of phones instead of words. In other words, conceptually, it would be comparing the error rate of the coloured bars in Figure 8 where there are 41 colours in total (41 phonetic labels), and PER measures the validity of colour sequence (i.e. label sequence).

5 – RESULTS

This following presents the results of the four experiments.

5.1 – Single ESNs' Performance to a Function of Data Size

In this section, we assess the performance of two single ESNs on phone recognition task and comparing them with two LSTMs. The two ESNs are respectively trained with 2 hours of annotated audio data and 16.6 hours of annotated audio data, and so are the two LSTMs. Each single ESNs used in this experiment contains a reservoir size of 1000 nodes and is

Model Name	Model Type	Training Data Size	FACC	F1 score	PER
ESN-2	Single ESN	2 hours	0.4405	0.2745	0.2788
ESN-16	Single ESN	16.6 hours	0.4926	0.2923	0.3450
LSTM-2	Single LSTM	2 hours	0.3828	0.1646	0.4903
LSTM-16	Single LSTM	16.6 hours	0.4769	0.2926	0.2928

Table 2. This table shows the results of the 1st experiment, including all ESNs and LSTMs performance in phone recognition at frame level and phrase level. ESN-2 performs the best of all instances and proves its strengths as a model that strikes balance between limited training data size and performance.

Model Name	Insertions	Deletions	Substitutions	Equals
ESN-2	(19,129)	(28, 680)	(23, 129)	(30, 995)
ESN-16	(25, 142)	(35, 608)	(58, 160)	(27, 912)
LSTM-2	(15, 73)	(36, 727)	(79, 206)	(26, 904)
LSTM-16	(36, 330)	(30, 313)	(69, 157)	(34, 790)

Table 3. This table shows all ESNs and LSTMs performance in the 1st experiment at phrase level by giving a breakdown of the PER calculation and the total frequency count of Equals (i.e. true predictions count). Each value is in pair where the first value is the number of label type and the second is the total frequency count of that category. For example, ESN-2 has (19, 129) for its Insertions, that means ESN-2 has 129 counts in false insertions and in these counts, there are 19 types of labels in total.

initialised with optimised hyperparameters listed in Table 1. Each single LSTMs used in this experiment is a model containing an LSTM layer and a fully connected output layer followed by a softmax layer. The hyperparameters for all single LSTMs are as those defined in Section 4.3.3. All four models, two single ESNs and two single LSTMs, are tested on the same dataset which is a 0.5-hour pre-processed annotated audio data. The FACC, F1 score, and PER was computed per sequence (i.e. per audio file, which each is approximately 15 seconds long), then averaged over the whole dataset. Each model is named after its model type and its training data size. For example, ESN-2 is the single ESN trained with 2 hours of data.

Table 2 shows the results of four models on the test dataset with their FACC, F1 scores, and PER. At frame level, while LSTMs demonstrate a greater improvement, ESNs have better performance than LSTMs both on small and large training data. For FACC, ESN-16 is 2% higher than LSTM-16, and ESN-2 is 6% higher than LSTM-2. Though ESN-16 and LSTM-16 have very close F1 scores, when training data is small, ESN-2 excels by having 11% higher in F1 score. This indicates that ESNs have a bigger advantage than LSTMs on small training data. At phrase level, ESN-2 has the lowest PER of all four models (0.2788). ESN-16, however, has a PER of 0.345, which is 5% higher than LSTM-16 (0.2928) and 7% higher than ESN-2 (0.2788). This further stresses ESN models advantage with small training data and exposes its weakness with large training data.

ESNs not only has it PER increased after training with larger dataset, and they are outperformed by LSTMs. To investigate this phenomenon, we delve deep into the details contributing to the PER calculation (i.e. Insertions I, Deletions D, and Substitutions S) and the true prediction count (i.e. Equals. E) (Table 3). We found that ESN-16's high PER is largely contributed by the high total frequency count of Deletions $D_{ESN-16} = 608$, and it has very little improvement from $D_{ESN-2} = 680$. LSTMs have greater improvement in Deletions than ESNs (i.e. $D_{ESN-16} - D_{ESN-2} = 72$ and $D_{LSTM-16} - D_{LSTM-2} = 414$). Furthermore, as training data size increases, ESNs see a slight increase both in Insertions count I_{ESN} and Substitutions count S_{ESN} whereas LSTMs see an increase in I_{LSTM} and a decrease in the S_{LSTM}.

Model Name	Insertions	Deletions	Substitutions	Equals
ESN-2	*F, SH, OW*	*B, G, TH*	*HH, __SIL__, P*	*EH, SH, W*
ESN-16	*EY, UW, SH*	*B, P, V*	*AO, EY, AE*	*__SIL__, T, EH*
LSTM-2	*AA, AH, M*	*DH, G, F*	*W, D, SH*	*R, T, P*
LSTM-16	*AW, AH, Y*	*G, TH, UH*	*AW, F, UW*	*S, __SIL__, EH*

Table 4. This table shows all ESNs and LSTMs performance in the 1st experiment by giving labels in the top 3 highest frequency count in each category. For example, of all ESN-2's Insertions made, label F, SH, OW are the top 3 most frequency inserted labels.

There are similarities in ESNs and LSTMs. Firstly, as the training data size increases, we expect D, I, and S to decrease with an increasing E for all models. However, only D meets our expectation. I_{ESN} increases from 129 to 142 and I_{LSTM} increases from 73 to 330; E_{ESN} decreases from 995 to 912 and E_{LSTM} decreases from 904 to 790. In short, both systems have a drop on both true positive (i.e. E) and false positive (i.e. I), and, between the two systems, LSTMs see a more obvious degradation. Table 4 summarises the details of these false positives (i.e. I), we found that the top 3 most frequently inserted labels go from (*F*, *SH*, *OW*) to (*EY*, *UW*, *SH*), and that of LSTMs goes from (*AA*, *AH*) to (*AW*, *AH*). *SH* remains in one of ESN system's top 2 Insertions, and so as LSTM: that *AH* remains in LSTMs'. On the other hand, the true positives are not as consistent; ESN-2's top 3 true positive labels are (*EH*, *SH*, *W*), and ESN-16's are (*__SIL__*, *T*, *EH*); and LSTM-2's top 3 true positive labels are (*R*, *T*, *P*), and LSTM-16's are (*S*, *__SIL__*, *EH*).

Another similarity between ESNs and LSTMs is that regardless the training data size, there are remaining labels missing (i.e. deleted). ESN models remain missing *B* frequently. Even though the label sees an improvement in the missing rate as training data size increases, *B* remains as both ESN-2's and ESN-16's most frequently missed label; Similarly, LSTMs remain missing *G* regardless of the training data size. *G* is LSTM-2 2nd most frequently missed label and LSTM-16 topmost frequently missed label.

In terms of the differences, ESNs show a greater improvement in Substitutions as the training data increases. ESN-2 has a high tendency of replacing *HH* with *P*; after training data size increases, *HH*'s Equals count drastically drops from 43 to 5, meaning that ESN system gets better in recognising *HH*. Furthermore, the second most frequently substituted label of ESN-2 is *__SIL__*, and it becomes ESN-16's best in Equals count. On the other hand, LSTMs system does not have such obvious improvement in its false replacements. *W* and *D* are LSTM-2's top 2 Substitutions, yet after the training data size increases, the Equals count for the former drops from 60 to 0, and the latter rises from 16 to 38. Furthermore, regardless the training data sizes, E_{ESN} is larger than E_{LSTM}. This also explains how ESNs get higher FACC.

In summary, at frame level, although LSTMs show a dramatic improvement, ESNs have better performance both with small and large training data. At phrase level, ESNs have a performance degradation and is outperformed by LSTMs when training data size is larger. Regardless, ESNs performs the best of all when it has compact training data size.

Model Name	# of Parameters	FACC	F1 score	PER
ESN-500	20500	0.4209	0.1911	0.3531
ESN-1000	41000	0.4429	0.2124	0.3203
ESN-2000	82000	0.4630	0.2337	0.3102
ESN-4000	164000	0.4805	0.2515	0.3017
LSTM-1	26880	0.3717	0.1765	0.3989
bi-LSTM-1	153088	0.5143	0.3189	0.2787

Table 5. This table shows all ESNs and LSTMs performance in the 2nd experiment in phone recognition at frame level and phrase level. ESN-300 and LSTM-1 have similar parameter sizes, yet ESN-500 outperforms LSTM-1 both at frame level and phrase level. ESN-4000 and bi-LSTM-1 have similar parameter sizes, and bi-LSTM-1 outperforms ESN-4000. This result shows that when there are limited memory resources, ESNs is a better option to for the task.

5.2 – Single ESN's Performance to a Function of Reservoir Size

In this section, we evaluate the performance of single ESNs to a function of the reservoir size n (i.e. number of nodes in the reservoir). n goes from 500 to 4000, increasing two-fold, and the corresponding number of parameters PN_{ESN} rises two-fold from 20.5K to 164K. 5 instances are created for each reservoir size, and we find the average of their FACC, F1 scores, and PER as a performance representation for the single ESN in that size. We also compare the two systems' frequency count in Deletions D, Insertions I, Substitutions S, and Equals E. For ESNs, we randomly pick one instance each from reservoir size $n = 1000$ ($PN_{ESN} = 41K$) and $n = 4000$ ($PN_{ESN} = 164K$) as representation of small and large single ESN. For LSTMS, we pick an instance each from a single LSTM with number of parameters $PN_{LSTM} = 26.8K$ and a bi-LSTM with $PN_{LSTM} = 153K$. We use 1.25 hour of pre-processed annotated audio data with a train/test split of 4:1. Each single ESN is named after the model type and reservoir size (e.g. ESN-500 means a single ESN with $n = 500$). The single LSTMs and bi-LSTMs are named as LSTM-1 and bi-LSTMs-1 respectively.

Unsurprisingly, both ESNs and LSTMs have a significant improvement in their performance as n doubles (Table 5). ESNs have its FACC and F1 score increase by 2%, and eventually reach to a total increase of 6%. ESNs' PER shows a similar trend with a total decrease of 5%, yet the decrease rate gradually decrease as the reservoir size grows (Figure 11). PER drops by 3% when n increases from 500 to 1000, then after it drops by 1% each time as n increases from 1000 to 2000, then to 4000. LSTMs see a more dramatic improvement after the parameter size increases. PN_{LSTM} grows from 26,800 to 153,088, and the FACC and F1 score both increase by 14% with a drop of 12% in PER.

Regardless, LSTM-1 does not reach the same performance as ESN-500, even they share a similar parameter size. Bi-LSTM-1, however, surpasses ESN-4000 even though they have similar model sizes. This answers our research questions in terms of ESNs' efficiency, that when there are limited memory resources and only a compact model is allowed, ESNs are a better option than LSTMs.

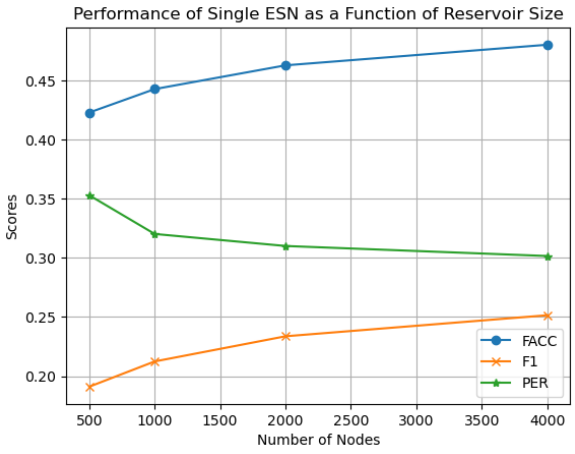


Figure 11. This figure plots the results of the 2nd experiment. The y-axis denotes the scores, and x-axis denotes the reservoir sizes of each single ESN. The rise of FACC and F1 score and the decline of PER are steeper when the reservoir sizes are relatively smaller. Once the reservoir size is over 2000 nodes, the changing rate starts decreasing.

Model Name	Insertions	Deletions	Substitutions	Equals
ESN-1000	(19, 109)	(25, 239)	(48, 114)	(32, 482)
ESN-4000	(31, 193)	(22, 159)	(39, 93)	(33, 472)
LSTM-1	(18, 60)	(34, 300)	(56, 117)	(29, 451)
bi-LSTM-1	(30, 135)	(29, 152)	(56, 79)	(35, 375)

Table 6. This table shows all ESNs and LSTMs performance in the 2nd experiment in phone recognition by giving a breakdown of the PER calculation and the total frequency count of Equals (i.e. true predictions count). Each value is in pair where the first value is the number of label type and the second is the total frequency count of that category. For example, ESN-1000 has (19, 109) for Insertions, that means ESN-1000 has 109 counts in false insertions and within these counts, there are 19 types of labels. For all instances, the numbers of label type increase in Insertions and Equals; this means that model has gained the ability to recognise more labels (even there are false positive, i.e. Insertions) after the parameter sizes increase.

Model Name	Insertions	Deletions	Substitutions	Equals
ESN-1000	*AW*, *AA*, *CH*	*P*, *B*, *G*	*SH*, *AY*, *IY*	*R*, *T*, *__SIL__*
ESN-4000	*AW*, *JH*, *Y*	*G*, *P*, *B*	*SH*, *S*, *AY*	*R*, *S*, *Z*
LSTM-1	*EY*, *D*, *UW*	*DH*, *TH*, *AW*	*U*, *W*, *NG*	*W*, *AH*, *AY*
bi-LSTM-1	*UW*, *Y*, *TH*	*G*, *NG*, *UH*	*Y*, *L*, *NG*	*W*, *G*, *AY*

Table 7. This table shows all ESNs and LSTMs performance in the 2nd experiment in phone recognition at phrase level by giving the top 3 labels with the highest frequency count in each category. For example, for ESN-1000, the top 3 most frequently (and falsely) inserted labels are AW, AA, CH. Both model types have labels remained falsely inserted (e.g. AW in ESNs' Insertions, UW in LSTMs' Insertions) or falsely substituted (e.g. SH in ESNs' Substitutions, NG in LSTMs' Substitutions), or correctly predicted (e.g. R in ESNs' Equals, W and AY in LSTMs' Equals).

As parameter size increases, both ESNs and LSTMs have a decrease in total frequency count of Deletions D, Substitutions S, and Equals E, and an increase in that of Insertions I (Table 6). The number of labels in some categories also follows a similar trend: that both types of models have the number of label type in Deletions d decrease, and the number of labels type in Insertions i increases. However, both ESNs and LSTMs have their number of label type in Equals e increases. This indicates that as n increases, the model has a greater ability to recognise more labels, hence an increase in e and i. However, this capability of recognising new labels does not accompany with good accuracy. Therefore, we see an increase in I and a decrease in E.

Comparing to LSTMs, ESNs have a less significant growth in e. LSTMs has its e_{LSTM} increasing from 29 to 35 while e_{ESN} increases from 32 to 33. This shows that ESN system has a more sluggish improvement in recognising more labels as n increases. However, while both systems see a decrease in E, ESN system, E_{ESN} has a smaller drop than E_{LSTM}; the former has a decrease of 10 and the latter has a decrease of 76.

As parameter size increases, both LSTMs and ESNs demonstrate a good robustness in keeping accuracy of the learnt labels (Table 7). For ESN system, *R*, *S*, and *T* remain as three of the four top frequency count in Equals. For LSTMs system, *W* and *AY* remain as 2 of the 3 top frequency count in Equals.

Reservoir Size n (# nodes)	# layers l	FACC	F1 score	PER
500	1	0.4391	0.2232	0.3611
500	2	0.4523	0.2474	0.3327
1000	1	0.4594	0.2457	0.3295
1000	2	0.4773	0.2782	0.3192

Table 8. This table shows HRC performance in the 3rd experiment at frame level and phrase level. Contrary to our expectation (and the results from Triefenbach et al., 2025), there is very little difference in the performance between adding layers and increasing reservoir sizes. In other words, HRC with 2 layers of $n = 500$ should outperform HRC with 1 layer of $n = 1000$.

Model	Training Data	# Parameters	PER
Single ESN-1000	2.5 hours	20.5K	0.2788
Wav2Vec2	960 (pretrained) + 100 hours	315M	0.0799

Table 9. This table summarises a comparison between Wav2Vec2 and the best ESNs instance by far in the three experiments.

5.3 – HRCs in Phone Recognition

In this section, we examine the performance of HRCs as a function of number of layers l and reservoir size n. l goes from 1 to 2, and for each layer, n goes from 500 to 1000 nodes. Each instance is named after the model type, reservoir size in each layer, and number of layers. For example, ENS-500-1 is an HRC with 1 layer of ESN $n = 500$. We use 1 hour of pre-processed annotated audio data with a train/test split of 4:1.

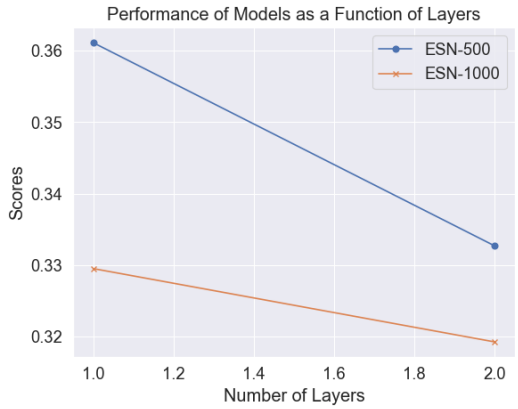


Figure 12. This figure plots the results of the 3rd experiment. We are expecting ESN-500 with 2 layers (ESN-500-2) has a lower PER than ESN-1000 with 1 layer (ESN-1000-1), yet the results suggest otherwise.

Table 8 shows the results of the four ESN instances, and the increase in the layer number does not induce a more significant improvement than the increase in the reservoir size; and this is not aligned with the findings of Triefenbach et al. (2025). ESN-500-1 has an FACC of 0.4391 and an F1 score of 0.2232; yet both adding another layer (ESN-500-2) and doubling the ESN reservoir size (ESN-1000-1) contribute a 2% increase in FACC and 2% increase in F1 score. Moreover, PER even suggests that, at phrase level, increasing reservoir size can improve the performance slightly more than increasing layers. ESN-500-1 has PER of 0.3611, and ESN-500-2 has a PER of 0.3327 and ESN-1000-1 has a PER of 0.3295 (Figure 12).

5.4 – Comparison with Transformers

ESN models by far excel in small training data; therefore, for this experiment, we choose ESN-1000 from the 1st experiment, a single ESN model with reservoir size $n = 1000$ (trained on 2.5h LibreSpeech train-clean data). We compare ESN-1000 results to the work from Shahin et al. (2025). They use a Wav2Vec2-base model pretrained on 960h of LibriSpeech data, fine-tuned and tested on 100h LibriSpeech train-clean dataset, which is 50 times more than ESN-1000's training data size. As CTC allows model using a blank node in some frames when uncertain, comparing the models at frame-level can be an unfair comparison. Therefore, we make the comparison on a phrase-level using PER (Table 9). Even though the Wav2Vec2 reaches a PER of 4.8%, which is far lower than single ESN-1000 results (0.27), the size of parameters goes beyond 95M, which is far larger than ESN-1000 (41K).

6 – DISCUSSION

In the first experiment, we found single ESNs performing better than single LSTMs, especially when training data is small. This result aligns with the findings in Jirak et al. (2020), when they compare ESNs and LSTMs for continuous gesture recognition. Similar to our findings, their ESN framework achieves comparably good performance comparing to LSTMs yet extending label sizes and data sizes together lead to a significant challenge to the single ESN architecture. In Chattopadhyay et al. (2020), LSTMs and ESNs are compared in short-term forecasting, and similar to our results, LSTMs see a drastic increase in the performance after the training data size increases whereas standard ESN architecture with small training data shows the best performance. The underlying reasons are that ESNs have weaker dependence on the size of the training set, making them a robust choice for situations when data collection is a constraint. Furthermore, the reservoir in ESNs has random, fixed recurrent weights and only the output linear layer is being trained. This not only enables rapid training but also reduces the risk of overfitting, especially in small-data regimes. Training LSTMs, however, involves training all weights in the model via back propagation through time; therefore, LSTMs are prone to overfitting when training data is scarce. However, even though ESNs excel with small datasets in both speed and performance, LSTMs can outperform single ESNs when it has multilayers. As shown in experiment two, Bi-directional layers allow LSTMs to learn complex long-term dependencies. López-Ortiz et al. (2024) found that standard ESN models have limitations in memory requirements, and multi-reservoir ESN architecture is the solution to the problem. Nevertheless, far from the expectation, in the 3rd experiment, we do not see drastic improvement after adding layers in multi-ESN framework. We see that 2-layered ESNs with reservoir size = $\frac{N}{2}$ nodes do not perform significantly different from single ESN of N nodes. There are a few possible reasons. Firstly, performance of ESNs highly depends on reservoir configuration, i.e. hyperparameters. The hyperparameters optimisation are done using single reservoir. Multi-reservoir ESNs could require a separate set of hyperparameters for optimisation. Secondly, multi-reservoir ESN has an optimal weight matrix density for the task. In Wringe et al. (2023), multi-reservoir ESNs (referred as Restricted ESN, a.k.a. rESNs, in their paper) are experimented with two types of weight density: Patch-consistent density and Overall-consistent density. Overall-consistent density, which is the set up in our experiment, can make rESNs show little performance improvements, or even performance degradations when added more than 4 reservoirs in the system. Furthermore, it can be the nature of tasks that affect the improvement of upscaling single ESNs to multi-reservoir ESNs. Pedrelli & Hinaut (2022) proposed a hierarchal-task reservoir (HTR) (Figure 13) architecture where in each layer ESN is trained to perform different tasks in the ASR pipeline. It is found that breaking down the task and distributing them to each layer in HTR can outperform having an entire multi-reservoir ESNs do the whole task. In Future work, the 3rd experiment can be extended to word recognition using an HTR, where the first layer takes audio signal as input and performs phone recognition, and the second layer takes phonetic labels as input and performs a sequence-to-sequence word prediction.

There are similarities found between ESNs and LSTMs. In the first experiment, even when the training data size increases, there are labels remaining missed (i.e. Deletions) or falsely inserted (Insertions) in both single ESN system and single LSTMs. However, for Equals, both

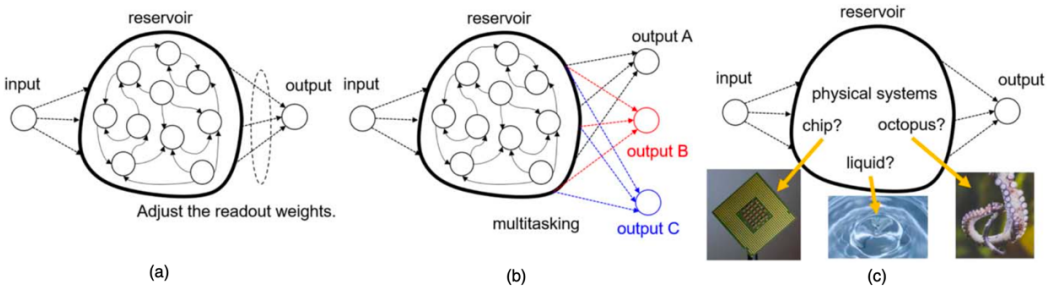


Figure 14. This figure (cited from Nakajima, 2020) illustrates different variations of RCs, including PRC in different materials.

systems fail to keep their best guessed labels after training data size increases. In other words, there are consistent labels in false positive and false negative but not in true positive. Furthermore, in both first and second experiment, ESNs and LSTMs see similar trend in the count of Deletions, Substitutions, Insertions, and Equals. After training data size increases, Deletions, Substitutions, and Equal count decrease but Insertions count rises.

In the 4th experiment, the results show that ESNs is still far from reaching the same performance as the state-of-the-art model, the transformers. However, ESNs has a form of a discrete time open dynamical system with a particular transfer function; the training of ESNs does not have access to internal parameters of the reservoir. This feature gives ESNs full access to any simulated or neuronal dynamical system as a replacement of the reservoir. Such material system is referred as Physical Reservoir Computing (PRC) (Figure 14). The possible substrates include but not limited to analogue electronic circuits, atomic switch systems, nanomagnetic devices, photonics, MEMS devices (Allwood et al., 2023; Caluwaerts et al., 2013; Dale et al., 2019, Barazani et al., 2020; Fujii & Nakajima, 2017). Larger et al. (2017) apply PRC on a photonic train and test of AURORA-2 database (Nakamura et al., 2024), and they achieve an WER of 4.5%. In N. Parihar et al. (2004), they apply PRC with a multistate architecture with 20K nodes and achieve the best normalised WER of about 1% on AURORA-2 dataset. Zhong et al. (2021) use a dynamic memristor-based reservoir train and test on NIST TI46 database, containing isolated spoken digits, and achieve a WER of 0.4%. Furthermore, Yan et al. (2024) explore the parameter size and required training petaflops between transformers-based models and ESNs and found that with same capacity (parameter size), ESNs have far less computational cost for training (i.e. petaflops), presenting ESNs as a compelling model for future wearable devices and workstations deployment (Figure 15).

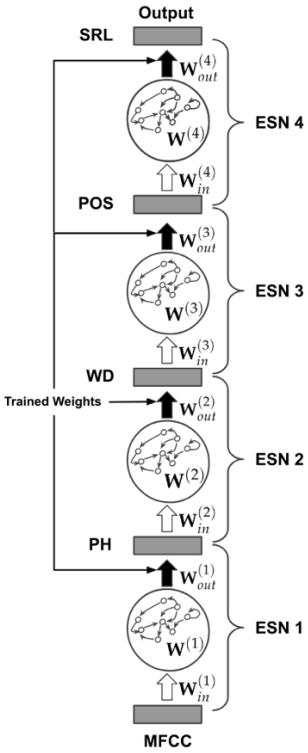


Figure 13. This figure (cited from Pedrelli & Hinaut, 2022) demonstrates the design of their hierarchal task reservoir computing (HTR). Each layer contains an ESN in charge of different tasks in ASR pipeline. The first layer takes the MFCCs as input, and performs phone recognition (output phones, PH). The second layer does word recognition (output words, WD). The third layer does Post-Of-Speech (POS) tagging on those words produced from 2nd layer. The last layer does semantic analysis (SRL).

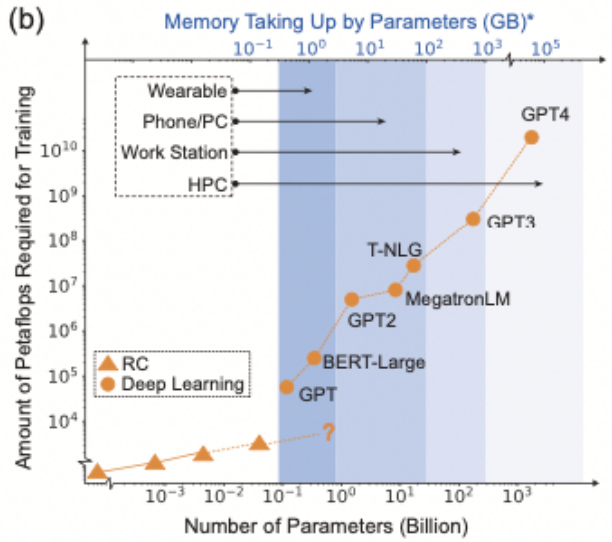


Figure 15. This figure (cited from Yan et al., 2024) summarises the current progress of RCs comparing to DNNs, particularly transformers-based models. It shows that RCs is still far being fully explored. Given to its compact size and efficiency, RCs have great potential in wearable devices or workstation applications.

7 – CONCLUSIONS & FUTURE WORK

In this project, we demonstrate how to complete phone recognition on LibreSpreech data using a RC-based system. We show how feature extraction is done with audio data, and how we complete acoustic model using ESNs. We also show its strength when there is limited training data available, and how it reaches an LSTMs' performance when they have similar model sizes. Our results support that reservoirs can exploit long-term dynamic properties of the articulatory system in phone recognition, and potentially achieve similar or better results as state-of-the-art with different architecture or an integration of physical reservoir.

For future work, the acoustic modules completed in this project can be adopted in word recognition pipelines by adding a language model and lexical module. With an ESN-based language model (Köster & Uchida, 2025), we can deploy a hierarchal task reservoir computing system. In this project, one of the possible reasons of HRCs not being successful is that we might fail to find the optimised hyperparameters for HRCs. This improvement can be completed in any future project experimenting HRCs. Alternatively, phone recognition can be recognised as groups of characters, and a CTC head on top could find the boundaries between the groups to complete word recognition. There is no existing research on ESNs+CTC yet, and this project provide an example of running experiments of ESN architecture with a comparison to other models, and in ESNs+CTC case, that can be other DNNs+CTC. Furthermore, this project focuses on developing a speaker dependent module using clean recordings; noises, accents, speaking style can be experimented in future to test the robustness of ESNs systems. All in all, this project present fair number of insights of ESNs in phone recognition task on clean English data.

REFERENCES

[1] Sethy, & Narayanan, S. (2003). Split-lexicon based hierarchical recognition of speech using syllable and word level acoustic units. *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, *1520-6149*. https://doi.org/10.1109/icassp.2003.1198895

[2] Al-Imam, A. (2020). Optimizing Linear Models via Sinusoidal Transformation for Boosted Machine Learning in Medicine. *DOAJ (DOAJ: Directory of Open Access Journals)*.

[3] Allwood, D. A., Matthew, Griffin, D., Hayward, T. J., Luca Manneschi, KH, M. F., O'Keefe, S., Stepney, S., Swindells, C., Trefzer, M. A., Vasilaki, E., Venkat, G., Vidamour, I., & Wringe, C. (2023). A perspective on physical reservoir computing with nanomagnetic devices. *Applied Physics Letters*, *122*(4). https://doi.org/10.1063/5.0119040

[4] Babu, A., Wang, C., Tjandra, A., Kushal Lakhotia, Xu, Q., Goyal, N., Singh, K., Patrick von Platen, Saraf, Y., Pino, J., Alexei Baevski, Conneau, A., & Auli, M. (2022). *XLS-R: Self-supervised Cross-*

lingual Speech Representation Learning at Scale. https://doi.org/10.21437/interspeech.2022-143

[5] Baevski, A., Hsu, W.-N., CONNEAU, A., & Auli, M. (2021). Unsupervised Speech Recognition. *Advances in Neural Information Processing Systems*, *34*, 27826–27839. https://proceedings.neurips.cc/paper/2021/hash/ea159dc9788ffac311592613b7f71fbb-Abstract.html

[6] Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *Advances in Neural Information Processing Systems*, *33*. https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html

[7] Barazani, B., Dion, G., Morissette, J.-F., Beaudoin, L., & Sylvestre, J. (2020). *Microfabricated Neuroaccelerometer: Integrating Sensing and Reservoir Computing in MEMS*. *29*(3), 338–347. https://doi.org/10.1109/jmems.2020.2978467

[8] Bhatt, S., Jain, A., & Dev, A. (2020). Acoustic Modeling in Speech Recognition: A Systematic Review. *International Journal of Advanced Computer Science and Applications*, *11*(4). https://doi.org/10.14569/ijacsa.2020.0110455

[9] Billa, J. (2017). *Improving LSTM-CTC based ASR performance in domains with limited training data*. ArXiv.org. https://arxiv.org/abs/1707.00722

[10] Bizzocchi, A. L. (2017). How Many Phonemes Does the English Language Have? In *International Journal on Studies in English Language and Literature* (Vol. 5, Issue 10). https://doi.org/10.20431/2347-3134.0510006

[11] Braga, D., Freitas, D., Coelho, L., Moura, A., & João Barros, M. (2005). *On the Identification of Word-Boundaries using Phonological Rules for Speech Recognition and Labeling*. http://www.conforg.fr/acoustics2008/cdrom/data/fa2005-budapest/paper/866-0.pdf

[12] Burget, L., Schwarz, P., Agarwal, M., Pinar Akyazi, Feng, K., Ghoshal, A., Ondrej Glembek, Goel, N., Karafiat, M., Povey, D., Ariya Rastrow, Rose, R. C., & Thomas, S. (2010). Multilingual acoustic modeling for speech recognition based on subspace Gaussian Mixture Models. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* . https://doi.org/10.1109/icassp.2010.5495646

[13] C, V., & Radha, V. (2012). *A Review on Speech Recognition Challenges and Approaches*. World of Computer Science and Information Technology Journal (WCSIT). http://www.conforg.fr/acoustics2008/cdrom/data/fa2005-budapest/paper/866-0.pdf

[14] Caluwaerts, K., Michiel D'Haene, D. Verstraeten, & Schrauwen, B. (2013). *Locomotion Without a Brain: Physical Reservoir Computing in Tensegrity Structures*. *19*(1), 35–66. https://doi.org/10.1162/artl_a_00080

[15] Chattopadhyay, A., Hassanzadeh, P., & Subramanian, D. (2020). Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, *27*(3), 373–389. https://doi.org/10.5194/npg-27-373-2020

[16] Chen, L., & Mehra, R. K. (2008). Reformulating Kalman Filter Based Optimal Dynamic Coverage Control. *IFAC Proceedings Volumes*, *41*(2), 4174–4179. https://doi.org/10.3182/20080706-5-KR-1001.00702

[17] Chen, S. F., Kingsbury, B., Lidia Mangu, Povey, D., Saon, G., Soltau, H., & Zweig, G. (2006). Advances in speech transcription at IBM under the DARPA

EARS program. *IEEE Transactions on Audio, Speech and Language Processing*, *14*(5), 1596–1608. https://doi.org/10.1109/tasl.2006.879814

[18] Cheng, J., Dong, L., & Lapata, M. (2016). *Long Short-Term Memory-Networks for Machine Reading*. https://arxiv.org/pdf/1601.06733

[19] Conneau, A., Baevski, A., Collobert, R., Mohamed, A., & Auli, M. (2021). Unsupervised Cross-Lingual Representation Learning for Speech Recognition. *Interspeech 2021*. https://doi.org/10.21437/interspeech.2021-329

[20] Dale, M., Miller, J. F., Stepney, S., & Trefzer, M. A. (2019). A substrate-independent framework to characterize reservoir computers. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *475*(2226), 20180723. https://doi.org/10.1098/rspa.2018.0723

[21] Davis, S., & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *28*(4), 357–366. https://doi.org/10.1109/tassp.1980.1163420

[22] Deng, L., Li, J., Huang, J.-T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., Williams, J., Gong, Y., & Acero, A. (2013, May 1). *Recent advances in deep learning for speech research at Microsoft*. IEEE Xplore. https://doi.org/10.1109/ICASSP.2013.6639345

[23] Fernandes, V., Mascarehnas, L., Mendonca, C., Johnson, A., & Mishra, R. (2018). Speech Emotion Recognition using Mel Frequency Cepstral Coefficient and SVM Classifier. *2018 International Conference on System Modeling & Advancement in Research Trends (SMART)*, 200–204. https://doi.org/10.1109/sysmart.2018.8746939

[24] Fu, W. (2020). Application of an Isolated Word Speech Recognition System in the Field of Mental Health Consultation: Development and Usability Study. *JMIR Medical Informatics*, *8*(6), e18677. https://doi.org/10.2196/18677

[25] Fujii, K., & Nakajima, K. (2017). Harnessing Disordered-Ensemble Quantum Dynamics for Machine Learning. *Physical Review Applied*, *8*(2). https://doi.org/10.1103/physrevapplied.8.024030

[26] G. Evermann, Chan, H. Y., Gales, M. J. F., Hain, T., Liu, X., Mrva, D., Wang, L., & Woodland, P. C. (2004). Development of the 2003 CU-HTK conversational telephone speech transcription system. *IEEE International Conference on Acoustics Speech and Signal Processing*, *1520-6149*. https://doi.org/10.1109/icassp.2004.1325969

[27] Gallicchio, C., Micheli, A., & Pedrelli, L. (2017). Deep reservoir computing: A critical experimental analysis. *Neurocomputing*, *268*, 87–99. https://doi.org/10.1016/j.neucom.2016.12.089

[28] Gauthier, D. J., Bollt, E., Griffith, A., & Barbosa, W. A. S. (2021). Next generation reservoir computing. *Nature Communications*, *12*(1). https://doi.org/10.1038/s41467-021-25801-2

[29] Gerlach, M., & Font-Clos, F. (2020). A Standardized Project Gutenberg Corpus for Statistical Analysis of Natural Language and Quantitative Linguistics. *Entropy*, *22*(1), 126. https://doi.org/10.3390/e22010126

[30] Ghai, W., & Singh, N. (2012). Literature Review on Automatic Speech Recognition. *International Journal of Computer Applications*, *41*(8), 42–50. https://doi.org/10.5120/5565-7646

[31] Ghoraani, B., & Krishnan, S. (2011). Time–Frequency Matrix Feature Extraction and Classification of Environmental Audio Signals. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(7), 2197–2209. https://doi.org/10.1109/tasl.2011.2118753

[32] Gowda, T., & May, J. (2020). Finding the Optimal Vocabulary Size for Neural Machine Translation. *ArXiv (Cornell University)*. https://doi.org/10.18653/v1/2020.findings-emnlp.352

[33] Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5-6), 602–610. https://doi.org/10.1016/j.neunet.2005.06.042

[34] Grivel, E., Gabrea, M., & Najim, M. (2002). Speech enhancement as a realisation issue. *Signal Processing*, *82*(12), 1963–1978. https://doi.org/10.1016/S0165-1684(02)00251-7

[35] Güçlü, U., & van Gerven, M. A. J. (2017). Modeling the Dynamics of Human Brain Activity with Recurrent Neural Networks. *Frontiers in Computational Neuroscience*, *11*. https://doi.org/10.3389/fncom.2017.00007

[36] Hannan, M. A., Ker, P. J., Lipu, M. S. H., Choi, Z. H., Rahman, M. S. Abd., Muttaqi, K. M., & Blaabjerg, F. (2020). State of the Art of Solid-State Transformers: Advanced Topologies, Implementation Issues, Recent Progress and Improvements. *IEEE Access*, *8*(2169-3536), 19113–19132. https://doi.org/10.1109/access.2020.2967345

[37] Hintz, J., & Siegert, I. (2024). CommonBench: A larger Scale Speaker Verification Benchmark. *4th Symposium on Security and Privacy in Speech Communication*, 17–20. https://doi.org/10.21437/SPSC.2024-3

[38] Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *06*(02), 107–116. https://doi.org/10.1142/s0218488598000094

[39] Hu, Y., Huber, A., Anumula, J., & Liu, S.-C. (2018). *OVERCOMING THE VANISHING GRADIENT PROBLEM IN PLAIN RECURRENT NETWORKS*. https://arxiv.org/pdf/1801.06105

[40] Ibrahim, H., & Varol, A. (2020, June 1). *A Study on Automatic Speech Recognition Systems*. IEEE Xplore. https://doi.org/10.1109/ISDFS49300.2020.9116286

[41] J.Arora, S., & Pal Singh, R. (2012). Automatic Speech Recognition: A Review. *International Journal of Computer Applications*, *60*(9), 34–44. https://doi.org/10.5120/9722-4190

[42] Jaeger, H. (2006). Reservoir riddles: suggestions for echo state network research (extended abstract). *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, *3*, 1460–1462. https://doi.org/10.1109/ijcnn.2005.1556090

[43] Jaurigue, L., & Lüdge, K. (2022). Connecting reservoir computing with statistical forecasting and deep neural networks. *Nature Communications*, *13*(1). https://doi.org/10.1038/s41467-021-27715-5

[44] Jirak, D., Tietz, S., Ali, H., & Wermter, S. (2020). Echo State Networks and Long Short-Term Memory for Continuous Gesture Recognition: a Comparative Study. *Cognitive Computation*. https://doi.org/10.1007/s12559-020-09754-0

[45] Kaas, J. H., & Hackett, T. A. (2000). Subdivisions of auditory cortex and processing streams in primates.

Proceedings of the National Academy of Sciences, *97*(22), 11793–11799. https://doi.org/10.1073/pnas.97.22.11793

[46] Kesarkar, M., & Rao, P. (2003). *FEATURE EXTRACTION FOR SPEECH RECOGNITON*. https://www.ee.iitb.ac.in/~esgroup/es_mtech03_sem/sem03_paper_03307003.pdf

[47] Koroteev, M. V. (2021). BERT: A Review of Applications in Natural Language Processing and Understanding. *ArXiv:2103.11943 [Cs]*. https://arxiv.org/abs/2103.11943

[48] Köster, F., & Uchida, A. (2025). *Reservoir Computing as a Language Model*. ArXiv.org. https://arxiv.org/abs/2507.15779

[49] Kröger, B. J., Kannampuzha, J., & Neuschaefer-Rube, C. (2009). Towards a neurocomputational model of speech production and perception. *Speech Communication*, *51*(9), 793–809. https://doi.org/10.1016/j.specom.2008.08.002

[50] Kubo, Y., Shigeki Karita, & Michiel Bacchiani. (2022). Knowledge Transfer from Large-Scale Pretrained Language Models to End-To-End Speech Recognizers. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8512–8516. https://doi.org/10.1109/icassp43922.2022.9746801

[51] Kumar, K., Aggarwal, R. K., & Jain, A. (2012). A Hindi speech recognition system for connected words using HTK. *International Journal of Computational Systems Engineering*, *1*(1), 25. https://doi.org/10.1504/ijcsyse.2012.044740

[52] Larger, L., Baylón-Fuentes, A., Martinenghi, R., Udaltsov, V. S., Chembo, Y. K., & Jacquot, M. (2017). High-Speed Photonic Reservoir Computing Using a Time-Delay-Based Architecture: Million Words per Second Classification. *Physical Review X*, *7*(1). https://doi.org/10.1103/physrevx.7.011015

[53] Lee, C., Hyun, D., Choi, E., Go, J., & Lee, C. (2003). Optimizing feature extraction for speech recognition. *IEEE Transactions on Speech and Audio Processing*, *11*(1), 80–87. https://doi.org/10.1109/tsa.2002.805644

[54] Li, B., Zhou, E., Huang, B., Duan, J., Wang, Y., Xu, N., Zhang, J., & Yang, H. (2014). Large scale recurrent neural network on GPU. *2022 International Joint Conference on Neural Networks (IJCNN)*, 4062–4069. https://doi.org/10.1109/ijcnn.2014.6889433

[55] Li, J., Deng, L., Gong, Y., & Haeb-Umbach, R. (2014). An Overview of Noise-Robust Automatic Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *22*(4), 745–777.

[56] Li, J., Mohamed, A., Zweig, G., & Gong, Y. (2015). LSTM time and frequency recurrence for automatic speech recognition. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. https://doi.org/10.1109/asru.2015.7404793

[57] Li, S., Akita, Y., & Kawahara, T. (2016). Semi-Supervised Acoustic Model Training by Discriminative Data Selection From Multiple ASR Systems' Hypotheses. *IEEE/ACM Transactions on Audio Speech and Language Processing*, *24*(9), 1524–1534. https://doi.org/10.1109/taslp.2016.2562505

[58] Liao, H., McDermott, E., & Senior, A. (2013, December 1). *Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription*. IEEE Xplore. https://doi.org/10.1109/ASRU.2013.6707758

[59] Lin, T., Horne, B. G., & Giles, C. Lee. (1998). How embedded memory in recurrent neural network architectures helps learning long-term temporal

dependencies. *Neural Networks*, *11*(5), 861–868. https://doi.org/10.1016/s0893-6080(98)00018-5

[60] Liu, Q., Wu, S., Wang, L., & Tan, T. (2016). Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts. *Proceedings of the ... AAAI Conference on Artificial Intelligence*, *30*(1). https://doi.org/10.1609/aaai.v30i1.9971

[61] López-Ortiz, E. J., Perea-Trigo, M., Soria-Morillo, L. M., Sancho-Caparrini, F., & Vegas-Olmos, J. J. (2024). Exploring deep echo state networks for image classification: a multi-reservoir approach. *Neural Computing and Applications*, *36*(20), 11901–11918. https://doi.org/10.1007/s00521-024-09656-4

[62] Loubser, A., De Villiers, P., & De Freitas, A. (2024). End-to-end automated speech recognition using a character based small scale transformer architecture. *Expert Systems with Applications*, *252*, 124119. https://doi.org/10.1016/j.eswa.2024.124119

[63] M. Benzeguiba, Mori, R. D., Deroo, O., Dupont, S., D. Jouvet, L. Fissore, Laface, P., Mertins, A., Ris, C., Rose, R., Tyagi, V., & C. Wellekens. (2006). Impact of variabilities on speech recognition. *International Conference on Speech and Computer*, 3–16.

[64] Maass, W., Natschläger, T., & Markram, H. (2002). Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Computation*, *14*(11), 2531–2560. https://doi.org/10.1162/089976602760407955

[65] Markov, N. T., & Kennedy, H. (2013). The importance of being hierarchical. *Current Opinion in Neurobiology*, *23*(2), 187–194. https://doi.org/10.1016/j.conb.2012.12.008

[66] McCowan, I. A., Moore, D., Dines, J., Gatica-Perez, D., Flynn, M., Wellner, P., & Hervé Bourlard. (2004). *On the Use of Information Retrieval Measures for Speech Recognition Evaluation*.

[67] McFee, B. (2015). *Librosa: Audio and music signal analysis in python*. SciPy. McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python. SciPy, 2015, 18-24.

[68] Medsker, L., & Jain, L. C. (1999). *Recurrent Neural Networks*. CRC Press.

[69] Miao, Y., Gowayyed, M., Xingyu Na, Ko, T., Metze, F., & Waibel, A. (2016). An empirical exploration of CTC acoustic models. *International Conference on Acoustics, Speech, and Signal Processing*. https://doi.org/10.1109/icassp.2016.7472152

[70] Mikolov, T., Kombrink, S., Burget, L., Černocký, J., & Khudanpur, S. (2011, May 1). *Extensions of recurrent neural network language model*. IEEE Xplore. https://doi.org/10.1109/ICASSP.2011.5947611

[71] Min, Z., & Wang, J. (2023). Exploring the Integration of Large Language Models into Automatic Speech Recognition Systems: An Empirical Study. *Communications in Computer and Information Science*, 69–84. https://doi.org/10.1007/978-981-99-8181-6_6

[72] Miranda, I., Diacon, A. H., & Niesler, T. (2019). A Comparative Study of Features for Acoustic Cough Detection Using Deep Architectures. *PubMed*. https://doi.org/10.1109/embc.2019.8856412

[73] Monomi , T. (2025). *Feedback-enhanced quantum reservoir computing with weak measurements*. Arxiv.org. https://arxiv.org/html/2503.17939v1#S4

[74] N. Parihar, Picone, J., Pearce, D., & Hirsch, H. G. (2004). Performance analysis of the Aurora large vocabulary baseline system. *European Signal Processing*

Conference, 553–556. https://doi.org/10.5281/zenodo.38362

[75] Nakajima, K. (2020). Physical reservoir computing—an introductory perspective. *Japanese Journal of Applied Physics*, *59*(6), 060501. https://doi.org/10.35848/1347-4065/ab8d4f

[76] Nakamura, T., Mishra, M., Tedeschi, S., Chai, Y., Stillerman, J. T., Friedrich, F., Yadav, P., Laud, T., Chien, V. M., Zhuo, T. Y., Misra, D., Bogin, B., Vu, X.-S., Karpinska, M., Varma, D. A., Kusa, W., Furlanello, T., Yokota, R., Muennighoff, N., & Pai, S. (2024). *Aurora-M: Open Source Continual Pre-training for Multilingual Language and Code*. ArXiv.org. https://arxiv.org/abs/2404.00399

[77] Nasersharif, B., & Namvarpour, M. (2024). Exploring the potential of Wav2vec 2.0 for speech emotion recognition using classifier combination and attention-based feature fusion. *The Journal of Supercomputing*, *80*(16), 23667–23688. https://doi.org/10.1007/s11227-024-06158-x

[78] O'Shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks*. https://arxiv.org/pdf/1511.08458

[79] Oh, Y. R., Park, K., & Park, J. G. (2021). Fast offline transformer-based end-to-end automatic speech recognition for real-world applications. *ETRI Journal*, *44*(3). https://doi.org/10.4218/etrij.2021-0106

[80] Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). *LIBRISPEECH: AN ASR CORPUS BASED ON PUBLIC DOMAIN AUDIO BOOKS*. https://www.danielpovey.com/files/2015_icassp_librispeech.pdf

[81] Patlar, F. (2009). *A CONTINUOUS SPEECH RECOGNITION SYSTEM FOR TURKISH LANGUAGE BASED ON TRIPHONE MODEL*. https://openaccess.iku.edu.tr/server/api/core/bitstreams/60f772b5-da7e-40b6-a3f4-f10b15a51b55/content

[82] Pedrelli, L., & Hinaut, X. (2022). Hierarchical-Task Reservoir for Online Semantic Analysis From Continuous Speech. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(6), 2654–2663. https://doi.org/10.1109/TNNLS.2021.3095140

[83] Prasanna, S. R. Mahadeva, Yegnanarayana, B., Pinto, J. P., & Hermansky, H. (2025, August 16). *Analysis of Confusion Matrix to Combine Evidence for Phoneme Recognition*. Epfl.ch; IDIAP. https://infoscience.epfl.ch/entities/publication/2ba86c71-ceb0-4fd5-ba50-6dff1ef375ca

[84] Rabiner, L., & Levinson, S. (1981). Isolated and Connected Word Recognition--Theory and Selected Applications. *IEEE Transactions on Communications*, *29*(5), 621–659. https://doi.org/10.1109/tcom.1981.1095031

[85] Raza, A. A., Athar, A., Randhawa, S., Tariq, Z., Saleem, M. B., Bin Zia, H., Saif, U., & Rosenfeld, R. (2018). Rapid Collection of Spontaneous Speech Corpora Using Telephonic Community Forums. *Interspeech 2018*. https://doi.org/10.21437/interspeech.2018-1139

[86] ReserviorPy. (2021). *A quick start to ReservoirPy — ReservoirPy 0.3.13 documentation*. Readthedocs.io. https://reservoirpy.readthedocs.io/en/latest/user_guide/quickstart.html

[87] S. Joshua Johnson, M. Ramakrishna Murty, & I. Navakanth. (2023). A detailed review on word embedding techniques with emphasis on word2vec. *Multimedia Tools and Applications*, *83*. https://doi.org/10.1007/s11042-023-17007-z

[88] Samy Bengio, & Georg Heigold. (2014). Word embeddings for speech recognition. *Interspeech 2022*, 1053–1057.

https://doi.org/10.21437/interspeech.2014-273

[89] Shahin, M., Epps, J., & Ahmed, B. (2025). Phonological level wav2vec2-based Mispronunciation Detection and Diagnosis method. *Speech Communication*, *173*, 103249–103249. https://doi.org/10.1016/j.specom.2025.103249

[90] Sharrab, Y. O., Attar, H., Eljinini, M. A. H., Al-Omary, Y., & Al-Momani, W. E. (2025). Advancements in Speech Recognition: A Systematic Review of Deep Learning Transformer Models, Trends, Innovations, and Future Directions. *IEEE Access*, *13*(13), 46925–46940. https://doi.org/10.1109/access.2025.3550855

[91] Shrawankar, U., & M, T. V. (2025). *Techniques for Feature Extraction In Speech Recognition System : A Comparative Study*. ArXiv.org. https://arxiv.org/abs/1305.1145

[92] Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019). The Performance of LSTM and BiLSTM in Forecasting Time Series. *2019 IEEE International Conference on Big Data (Big Data)*, 3285–3292. https://doi.org/10.1109/bigdata47090.2019.9005997

[93] Spyros Matsoukas, Gauvain, J.-L., Gilles Adda, Colthurst, T., Kao, C.-L., Kimball, O., Lamel, L., Franck Lefèvre, Ma, J., Makhoul, J., Nguyen, L., Prasad, R., Schwartz, R. S., Schwenk, H., & Xiang, B. (2006). Advances in transcription of broadcast news and conversational telephone speech within the combined EARS BBN/LIMSI system. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(5), 1541–1556. https://doi.org/10.1109/tasl.2006.878257

[94] Srivastava, B. M. L., & Sitaram, S. (2018). *Homophone Identification and Merging for Code-switched Speech Recognition*. https://www.isca-archive.org/interspeech_2018/srivastava18_interspeech.pdf

[95] Sulistyaningsih, Putranto, P., Daud, P., & Desvasari, W. (2018, October 1). *Fast Fourier Transform (FFT) Data Sampling using Hamming and Blackman Method for Radar*. IEEE Xplore. https://doi.org/10.1109/ICECOS.2018.8605193

[96] Thrun, S., Saul, L., & B Schölkopf. (2004). Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference. *Neural Information Processing Systems*.

[97] Triefenbach, F., Jalalvand, A., Demuynck, K., & Martens, J.-P. (2013). Acoustic Modeling With Hierarchical Reservoirs. *IEEE Transactions on Audio, Speech, and Language Processing*, *21*(11), 2439–2450. https://doi.org/10.1109/tasl.2013.2280209

[98] Triefenbach, F., Jalalvand, A., Schrauwen, B., & Martens, J. (2025). Phoneme Recognition with Large Hierarchical Reservoirs. *Advances in Neural Information Processing Systems*, *23*. https://proceedings.neurips.cc/paper/2010/hash/2ca65f58e35d9ad45bf7f3ae5cfd08f1-Abstract.html

[99] Trouvain, N., & Hinaut, X. (2021). Canary Song Decoder: Transduction and Implicit Segmentation with ESNs and LTSMs. *Lecture Notes in Computer Science*, *12895*, 71–82. https://doi.org/10.1007/978-3-030-86383-8_6

[100] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems*, *30*, 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[101] Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir

computing methods. *Neural Networks*, *20*(3), 391–403. https://doi.org/10.1016/j.neunet.2007.04.003

[102] Wang, D., Wang, X., & Lv, S. (2019). An Overview of End-to-End Automatic Speech Recognition. *Symmetry*, *11*(8), 1018. https://doi.org/10.3390/sym11081018

[103] Wester, M. (2003). Pronunciation modeling for ASR – knowledge-based and data-derived methods. *Computer Speech & Language*, *17*(1), 69–85. https://doi.org/10.1016/s0885-2308(02)00030-x

[104] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., & Drame, M. (2020). *Transformers: State-of-the-Art Natural Language Processing* (pp. 38–45). https://aclanthology.org/2020.emnlp-demos.6.pdf

[105] Wringe, C., Stepney, S., & Trefzer, M. A. (2023). Modelling and Evaluating Restricted ESNs on Single- and Multi-Timescale Problems. *Research Square (Research Square)*. https://doi.org/10.21203/rs.3.rs-3758288/v1

[106] Yan, M., Huang, C., Bienstman, P., Tino, P., Lin, W., & Sun, J. (2024). Emerging opportunities and challenges for the future of reservoir computing. *Nature Communications*, *15*(1), 2056. https://doi.org/10.1038/s41467-024-45187-1

[107] Yu, D., Seide, F., Li, G., & Deng, L. (2012). Exploiting sparseness in deep neural networks for large vocabulary speech recognition. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. https://doi.org/10.1109/icassp.2012.6288897

[108] Zaman, K., Melike Şah, Cem Direkoğlu, & Masashi Unoki. (2023). A Survey of Audio Classification using Deep Learning. *IEEE Access*, *11*(2169-3536), 106620–106649. https://doi.org/10.1109/access.2023.3318015

[109] Zheng, Y., Yang, X., & Dang, X. (2020). *Homophone-based Label Smoothing in End-to-End Automatic Speech Recognition*. ArXiv.org. https://arxiv.org/abs/2004.03437

[110] Zhong, Y., Tang, J., Li, X., Gao, B., & Qian, H. (2021). Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing. *Nature Communications*, *12*(1). https://doi.org/10.1038/s41467-020-20692-1

[111] Zhou, Z., Liu, L., Chandrasekhar, V., Zhang, J., & Yi, Y. (2020). Deep Reservoir Computing Meets 5G MIMO-OFDM Systems in Symbol Detection. *Proceedings of the ... AAAI Conference on Artificial Intelligence*, *34*(01), 1266–1273. https://doi.org/10.1609/aaai.v34i01.5481

[112] Zhou, Z., Odedeyi, T., Kelly, B., O'Carroll, J., Phelan, R., Darwazeh, I., & Liu, Z. (2020). Impact of Analog and Digital Pre-Emphasis on the Signal-to-Noise Ratio of Bandwidth-Limited Optical Transceivers. *IEEE Photonics Journal*, *12*(2), 1–12. https://doi.org/10.1109/jphot.2020.2966617

[113] Zweig, G., & Picheny, M. (2004). Advances in Large Vocabulary Continuous Speech Recognition. *Advances in Computers*, 249–291. https://doi.org/10.1016/s0065-2458(03)60007-0

APPENDIX A – Figures of the 1st Experiment Results

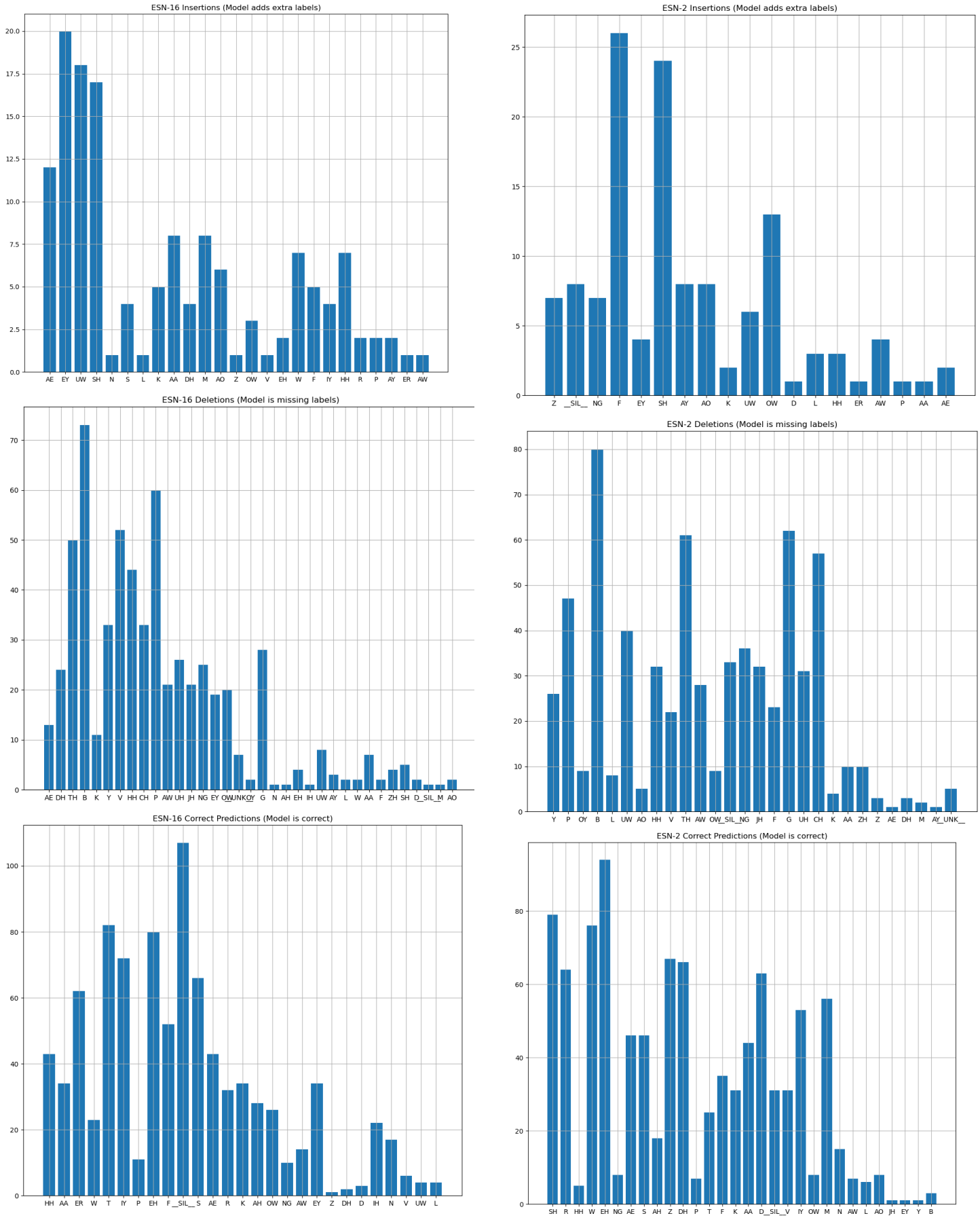


Figure 16. These figures show the frequency count of Insertions I, Deletions D, and Equals E for each single ESNs. The x-axis is the label type in each category, and y-axis is the frequency count for each label type.

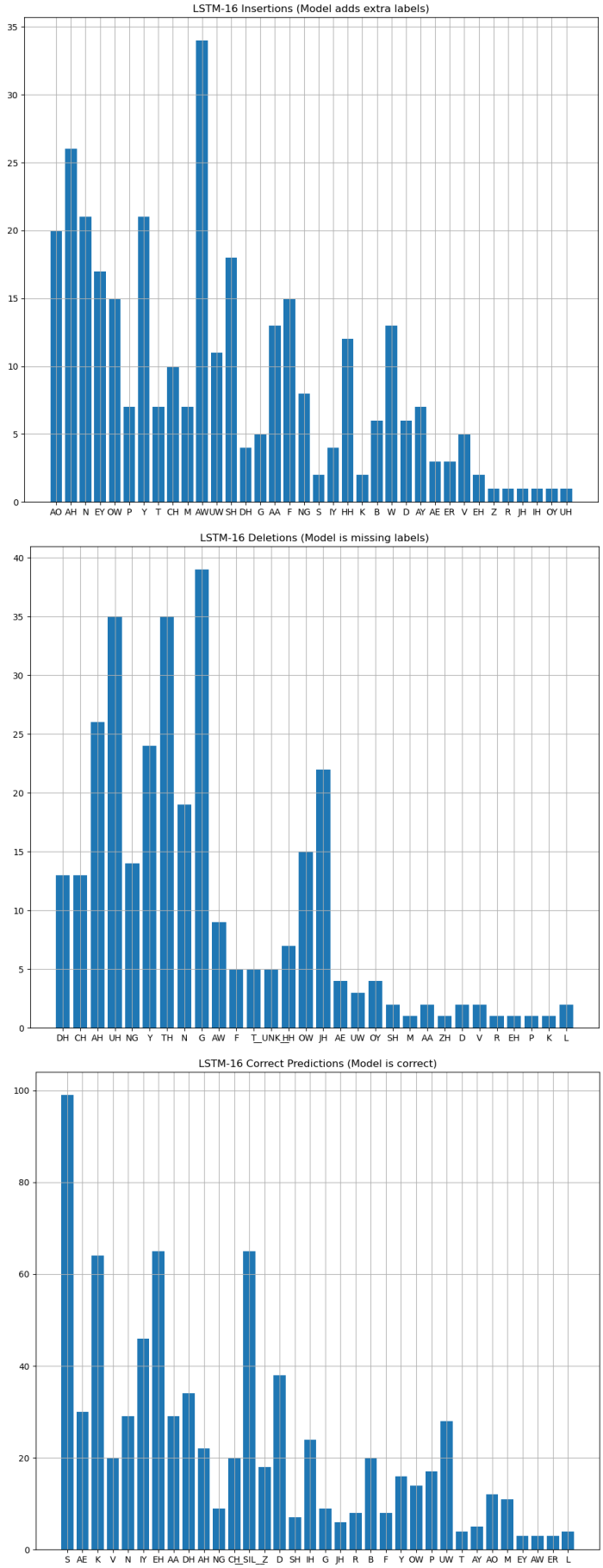
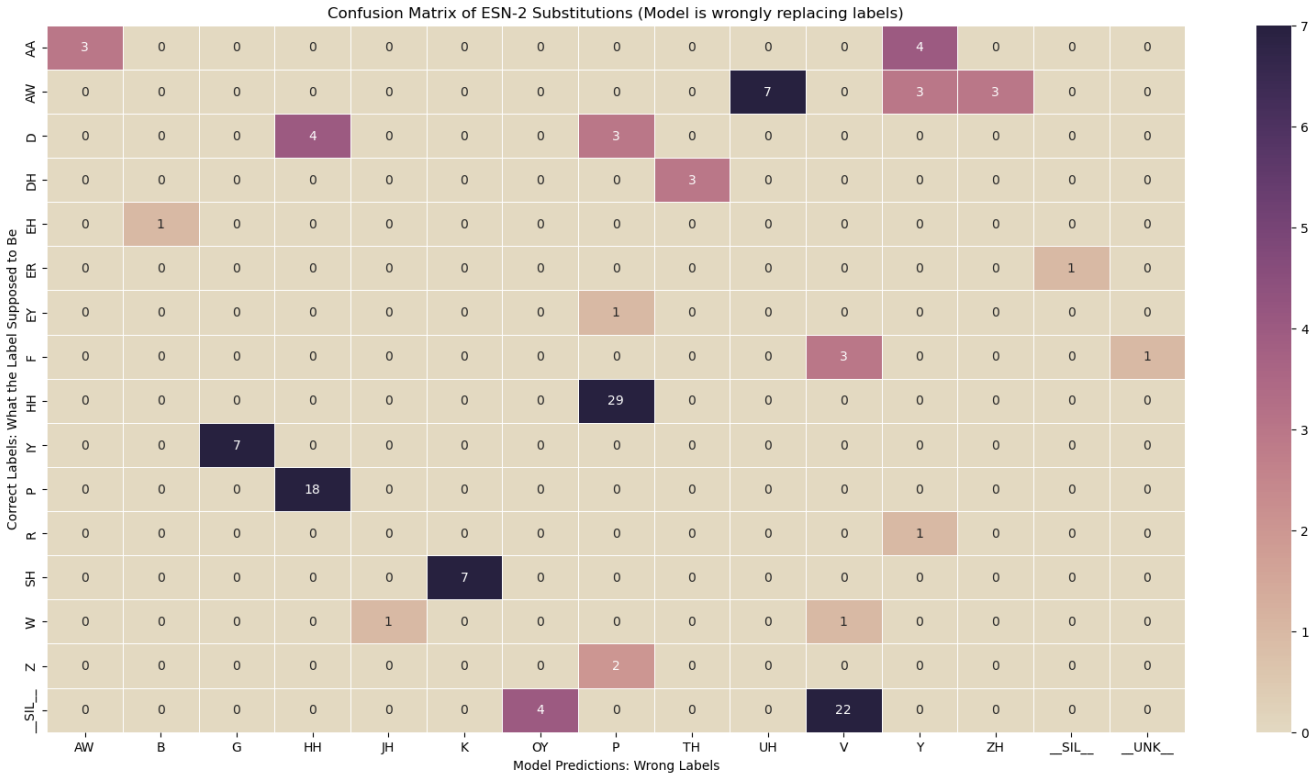


Figure 17. These figures show the frequency count of Insertions I, Deletions D, and Equals E for each single LSTMs. The x-axis is the label type in each category, and y-axis is the frequency count for each label type.



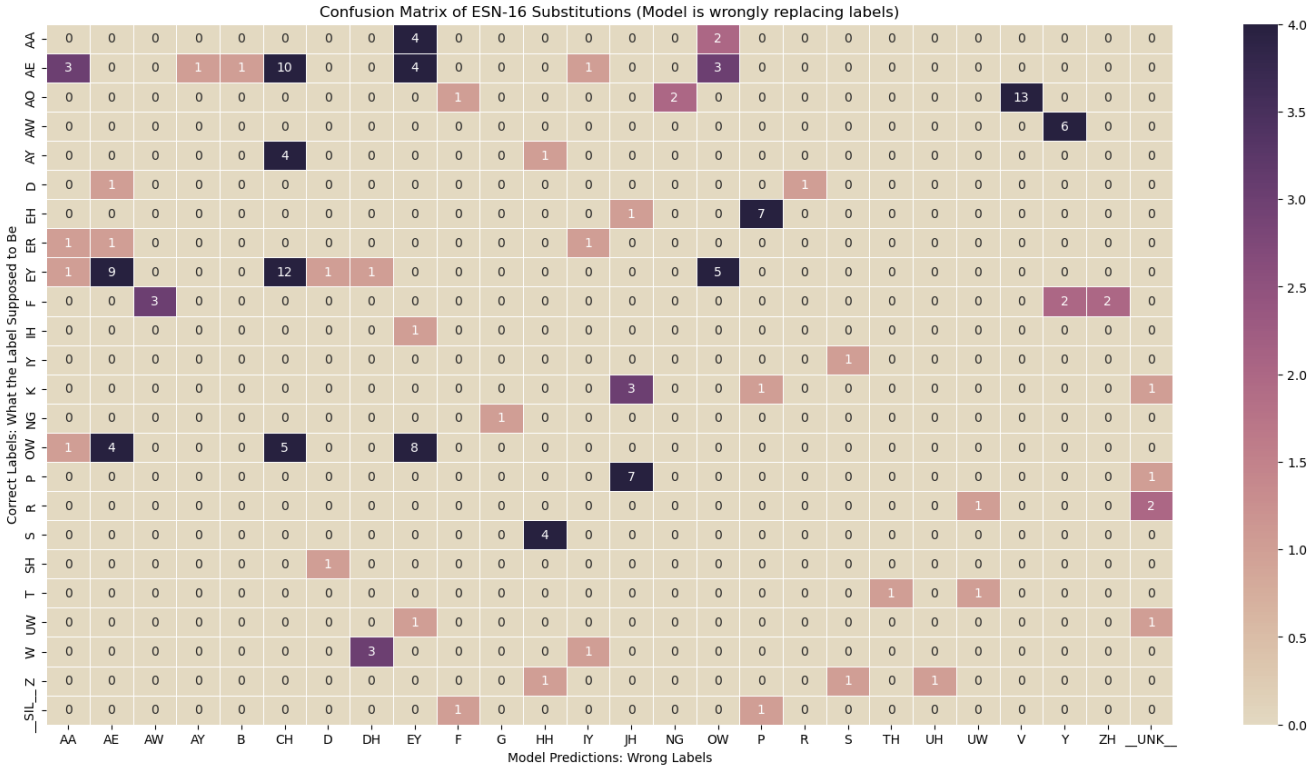


Figure 18. These figures show the frequency count of Substitutions S label pairs for each single ESNs. The x-axis is the phonetic labels from model predictions, and the y-axis is the correct labels.

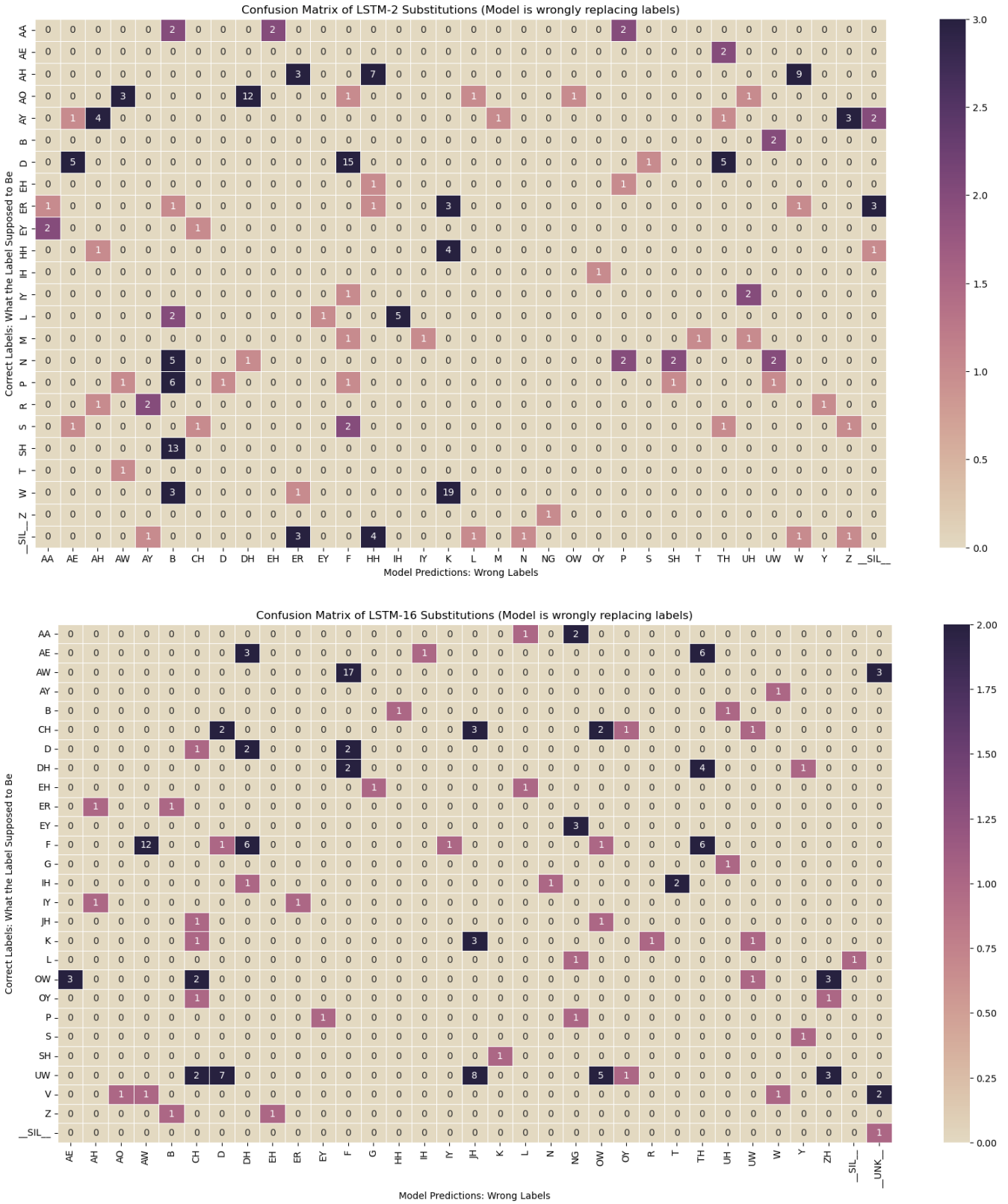
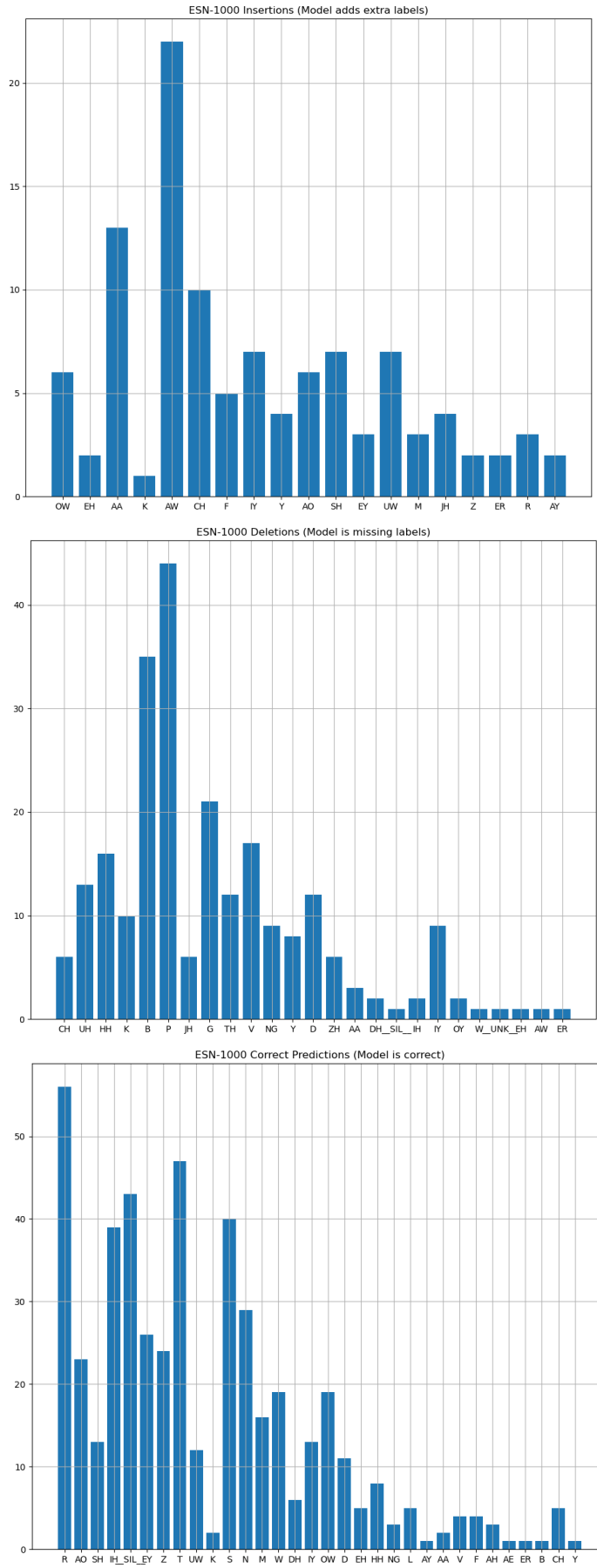


Figure 19. These figures show the frequency count of Substitutions S label pairs for each single LSTMs. The x-axis is the phonetic labels from model predictions, and the y-axis is the correct labels.

APPENDIX B – Figures of the 2nd Experiment Results



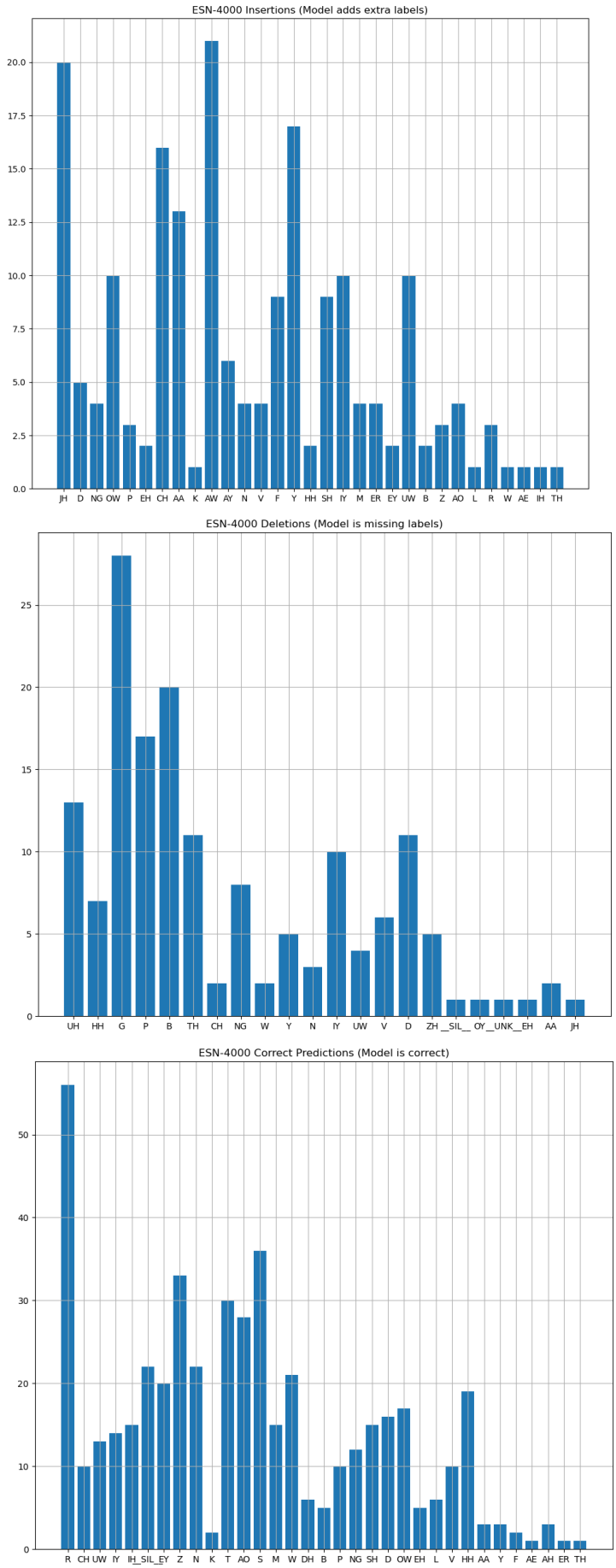


Figure 20. These figures show the frequency count of Insertions I, Deletions D, and Equals E for each single ESNs. The x-axis is the label type in each category, and y-axis is the frequency count for each label type.

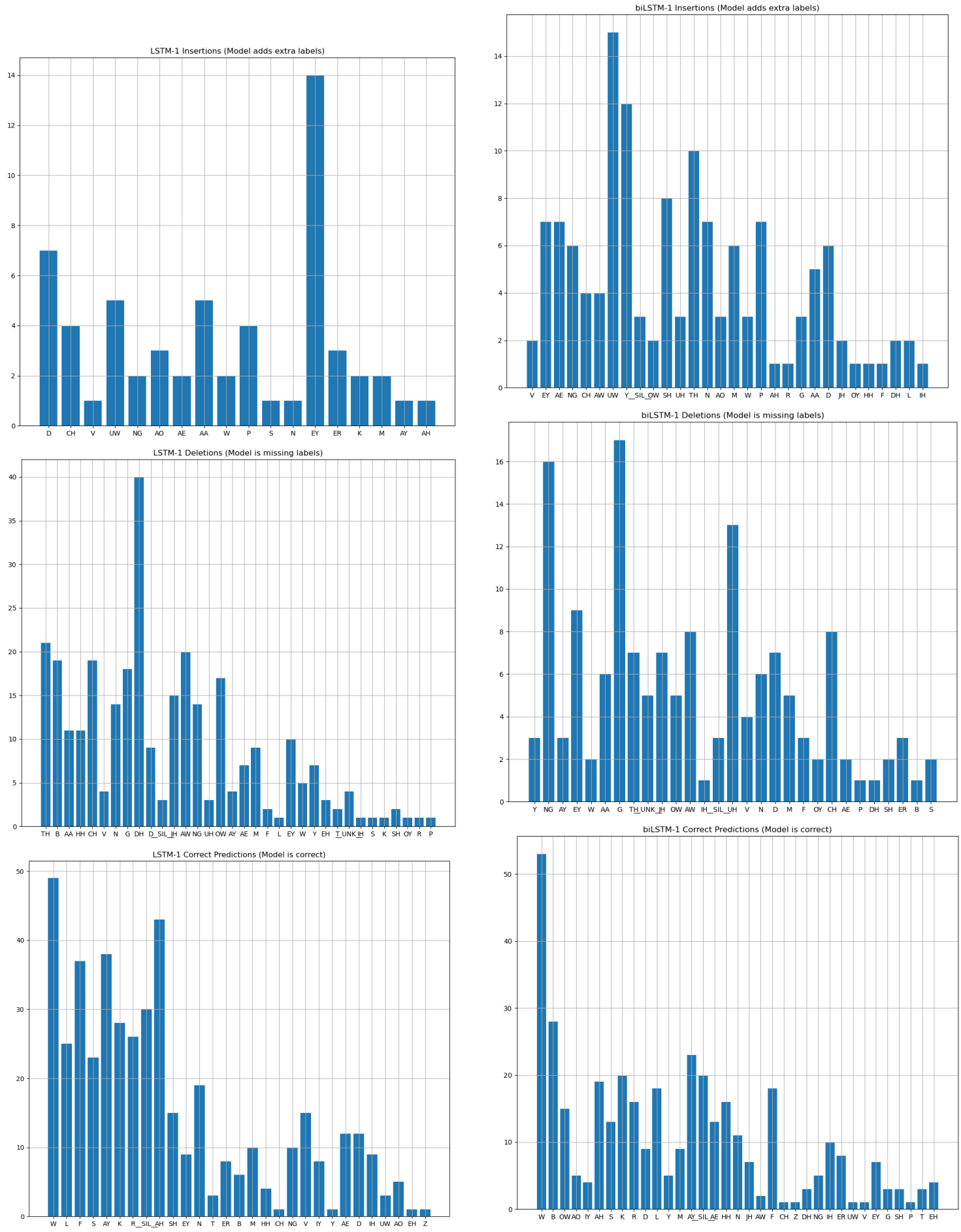


Figure 21. These figures show the frequency count of Insertions I, Deletions D, and Equals E for each LSTMs (including the single LSTMs, LSTM-1, and the bi-LSTMS, bi-LSTM-1). The x-axis is the label type in each category, and y-axis is the frequency count for each label type.

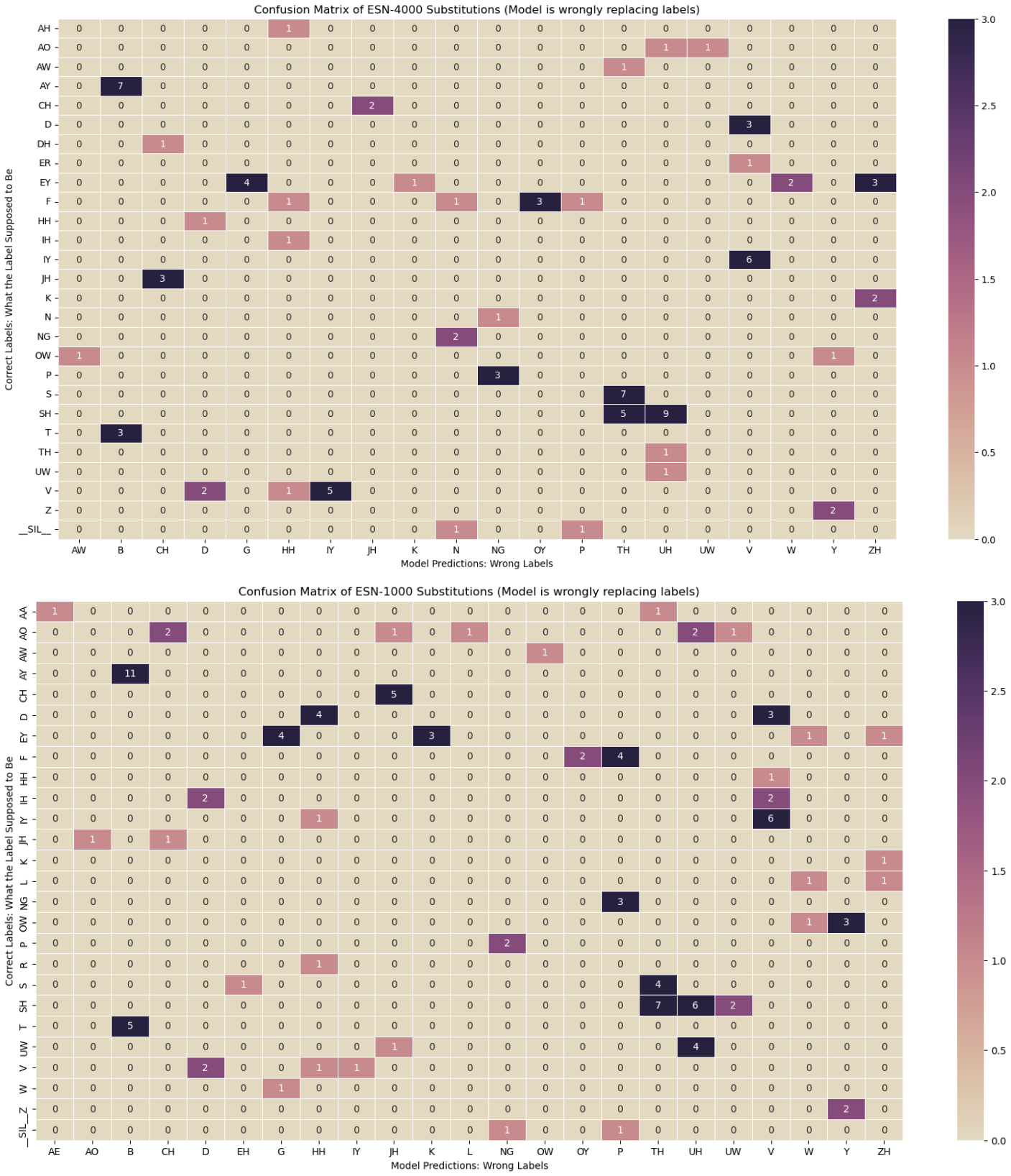


Figure 22. These figures show the frequency count of Substitutions S label pairs for each single ESNs. The x-axis is the phonetic labels from model predictions, and the y-axis is the correct labels.

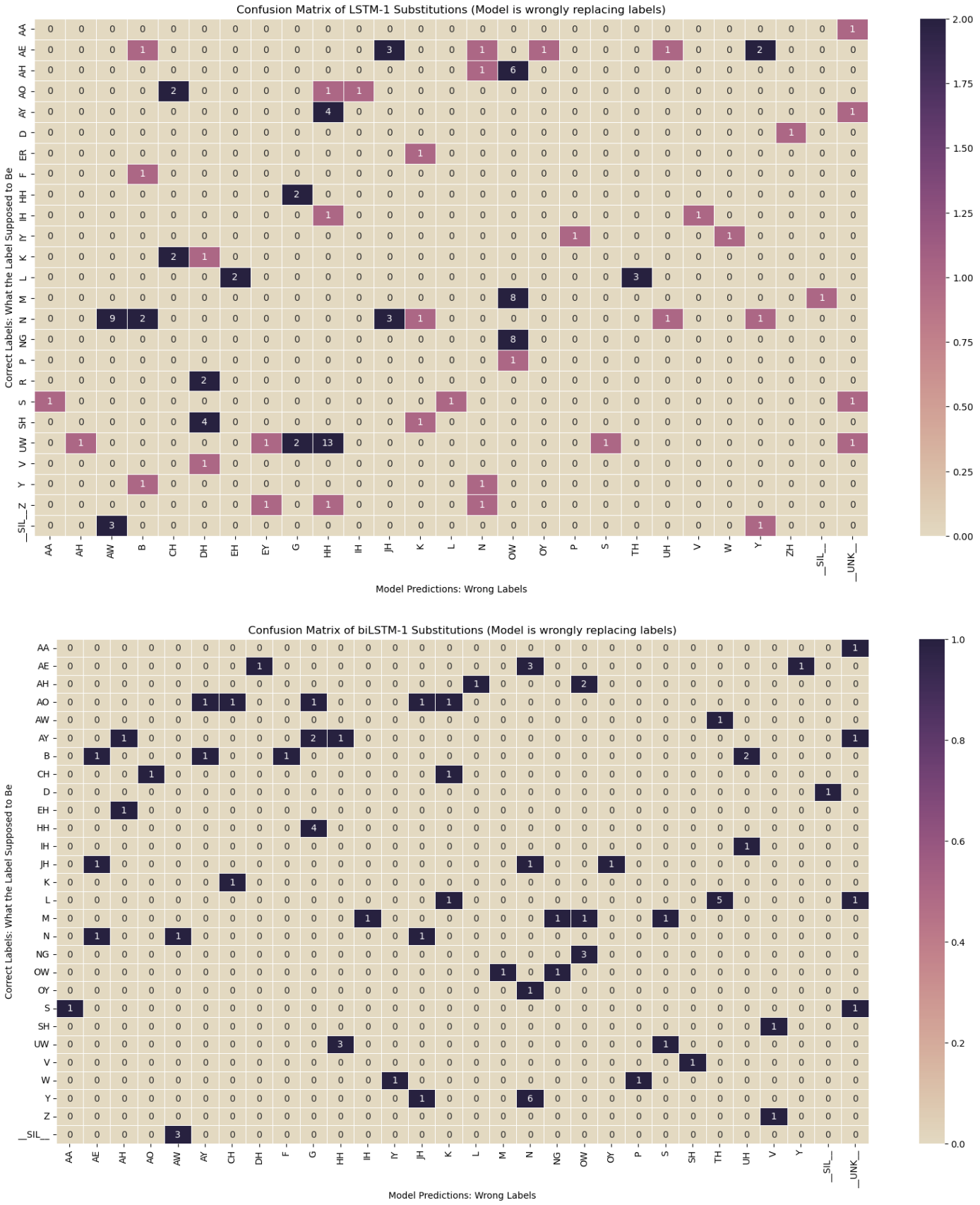


Figure 23. These figures show the frequency count of Substitutions S label pairs from LSTM-1 and bi-LSTM-1. The x-axis is the phonetic labels from model predictions, and the y-axis is the correct labels.