11-14-2019

Deepfake detection and low-resource language speech recognition using deep learning

Bao Thai
btt4530@rit.edu

Deepfake detection and low-resource language speech recogntion using deep learning

by

Bao Thai

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Department of Computer Engineering

Kate Gleason College of Engineering

Rochester Institute of Technology

Rochester, NY

November 14th 2019

Approved By:

Dr. Raymond Ptucha — Date

Primary Advisor - R.I.T Dept. of Computer Engineering

Dr. Alex Loui — Date

Secondary Advisor - R.I.T Dept. of Computer Engineering

Dr. Emily Prud'hommeaux — Date

Secondary Advisor - Boston College Dept. of Computer Science

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank Dr. Ray Ptucha and Dr. Emily Prud'hommeaux for the guidance and continuous support in my research. I am also thankful for Dr. Alex Loui for being on the thesis committee. I would also like to thank Dr. Wright for the guidance on the audio spoofing research. I would like to thank Robert Jimmerson and Dominic Arcoraci for their help with the research on speech recognition for Seneca. Finally, I would like to thank all the friends and family for their support throughout my time at RIT.

Contents

List of Figures

List of Tables

Abbreviations

ASR Automatic Speech Recognition

ASV Automatic Speaker Verification

CER Character Error Rate

CNN Convolution Neural Network

CQCC Constant Q cepstrum coefficients

CQT Constant Q transform

CTC Connectionist Temporal Classification

DFT Discrete Fourier Transform

EER Equal Error Rate

FFT Fast Fourier Transform

FNR False Negative Rate

FPR False Positive Rate

GAN Generative Adversarial Network

GMM Gaussian Mixture Model

HMM Hidden Markov Model

LA Logical Access

LF-MMI Lattic-free Maximum Mutual Information

LM Language model

LSTM Long-Short Term Memory

LVCSR Large Vocabulary Continuous Speech Recognition

MFCCs Mel-frequency cepstrum coefficients

MHA Multi-Headed Attention

PA Physical Access

PSOLA Pitch Synchronous Overlap and Add

RNN Recurrent Neural Network

t-DCF Tandem detection cost function

TTS Text-to-Speech

VC Voice Conversion

WER Word Error Rate

1 Abstract

While deep learning algorithms have made significant progress in automatic speech recognition and natural language processing, they require a significant amount of labelled training data to perform effectively. As such, these applications have not been extended to languages that have only limited amount of data available, such as extinct or endangered languages. Another problem caused by the rise of deep learning is that individuals with malicious intents have been able to leverage these algorithms to create fake contents that can pose serious harm to security and public safety. In this work, we explore the solutions to both of these problems. First, we investigate different data augmentation methods and acoustic architecture designs to improve automatic speech recognition performance on low-resource languages. Data augmentation for audio often involves changing the characteristic of the audio without modifying the ground truth. For example, different background noise can be added to an utterance while maintaining the content of the speech. We also explored how different acoustic model paradigms and complexity affect performance on low-resource languages. These methods are evaluated on Seneca, an endangered language spoken by a Native American tribe, and Iban, a low-resource language spoken in Malaysia and Brunei. Secondly, we explore methods to determine speaker identification and audio spoofing detection. A spoofing attack involves using either a text-to-speech voice conversion application to generate audio that mimic the identity of a target speaker. These methods are evaluated on the ASVSpoof 2019 Logical Access dataset containing audio generated using various methods of voice conversion and text-to-speech synthesis.

2 Introduction

2.1 Low-resource automatic speech recognition

With modernization and globalization, many languages are becoming extinct. It is estimated that up to 90% of the seven thousand languages in the world may be extinct by the end of this century [17]. When a language becomes extinct, a big part of the culture associated with the language will also become unretrievable since documents, folklore, and many other artifacts can no longer be understood. Additionally, each language provide linguists with unique information on the development and characteristic of human speech. Thus, it is extremely important to document and preserves the languages that are endangered or close to extinct. However, documenting endangered languages often come with challenges. Generating transcriptions of recorded speech from native speakers often requires a lot of manual labor and linguistic expertise. With the evolution of technology, automatic speech recognition (ASR) has slowly become a very useful tool to speed up the process of obtaining linguistic data from recorded speech. Despite the improvement in technology, developing a capable ASR system for endangered languages still comes with numerous challenges. One such challenge is the amount of data available to train an effective system. Acutely under-resource languages are typically spoken by a very small number of people [34]. Additionally, some speakers are not willing to share information about the languages due to sensitive reasons. Therefore, an effective ASR system aimed at preserving such languages have to be able to perform well without a large volume of training data.

2.2 Audio spoofing detection

One of the cornerstones of democracy is a well-informed population. With internet evolution seen in the past 25 years and the popularity of social networking platforms since the start of the decade, information can now be shared and distributed much easier than before. While this evolution allows people to feel closer together, it also presents a platform for untrustworthy information (or disinformation) to be propagated very quickly. The advance of deep learning in recent years has allowed for the generation of media, such as videos or audio clips, that resemble real people doing or saying things that are not real. While most fake media is still easily discernable from bonafide, pristine media, many have achieved the quality that is able to fool human viewers. Such media threatens the legitimacy of information propagated on the internet. Additionally, adversaries can use such media to fuel unrest, political instability and conspiracy theories.

While videos often are the most common form of fake media being used for such purposes, fake, or spoofed, audio can also lead to mistrust in information as well. In a fake audio clip, speech is made to sound like it was spoken by a person, who did not actually said it. The ability to generate such audio clips, especially when produced with a good algorithm, can allow an individual to spread misinformation for political or economical gains, such as shorting a stock by spread false announcement by a company's leader.

While there are key differences in low-resource ASR and spoofing detection, both problems involve inferring information from input audio. As such, we also aim to create a module which can be useful for both tasks. A module which proves to be useful for both ASR and spoofing detection tasks can then be applied to other audio processing tasks such as music analysis, noise detection, or sentiment analysis. Additionally, being able to use the same module or architecture for both problems allow the pre-training of models on one problem and then fine-tune on the other.

2.3 Contributions

The contributions of this thesis are:

- An investigation of data augmentation techniques to improve low-resource speech recognition.

- A transfer learning strategy to leverage high-resource languages to enhance the performance of ASR systems on low-resource languages.
- Novel convolutional-based architectures targeting low-resource ASR.
- Deep learning models aimed at detecting computer-generated speech.

3 Background

3.1 Acoustic modelling

The goal of large vocabulary continuous speech recognition (LVCSR) systems is to map an input acoustic signal to the corresponding orthographic transcription, typically text. A typical LVCSR system often consists of several components as detailed in Figure 1.

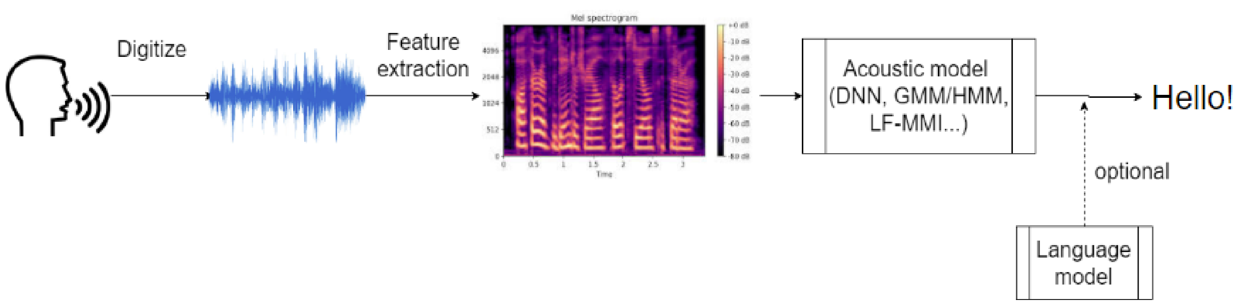

Figure 1: *A typical ASR system consists of digitization of acoustic signal, feature extraction, acoustic modelling, and optional language modelling.*

The acoustic modelling step in Figure 1 aims to perform the mapping of acoustic input O, which is a sequence of input frames $o_1, o_2, o_3, ..., o_t$, to a sequence of output symbols S ($s_1, s_2, s_3, ..., s_t$). Before the emergence of deep learning, this task was often performed using Gaussian Mixture Model (GMM) and then Hidden Markov Model (HMM) to convert input audio features such as Mel-frequency cepstrum coefficients (MFCCs) to characters, which are then corrected using a language model.

Improvements and breakthroughs in deep learning for automatic speech recognition (ASR) have resulted in significant improvements in ASR performance in high-resource languages such as English and Mandarin [31, 27, 10, 6, 14]. Such methods, however, require very large volumes of labelled training data to achieve these notable results. Deep learning ASR systems for languages with limited labelled training data typically must incorporate additional training resources such as cross-lingual acoustic models or in-domain synthetic acoustic data to begin to approach the word error rates found using traditional GMM/HMM frameworks.

The majority of deep learning ASR systems use recurrent neural networks (RNNs) to model temporal dependencies. In a RNN, the state and output of each timestep depends on the state of all previous timesteps in the sequence. Therefore, RNNs can use information from previous inputs as well as the current input to make predictions. For ASR, RNNs can be employed either to make one prediction per timestep, as seen in DeepSpeech 1 and 2 [28, 5], or to obtain an encoding of the audio sequence and then obtain the output text via a decoding network, as seen in Listen-Attend-Spell [10].

Despite good performance in sequence modelling tasks, RNNs often take a long time to train since the output of one timestep is dependent on the output of previous timesteps. As such, the computation of an RNN cell cannot be easily parallelized on modern GPUs. Meanwhile, convolution neural networks (CNNs) can easily take advantage of parallelized computation while still be able to model temporal dependencies by using different kernel sizes. CNNs have demonstrated superior performance on vision tasks such as image classification, image segmentation, and object recognition. Early CNN architectures require inputs to be of fixed size. However, as seen in object detection and image segmentation applications, fully convolutional variations can operate on multiple locations simultaneously and allow for variable-size inputs.

3.2 Mel-frequency cepstrum coefficients (MFCCs)

Speech signals can be represented digitally as an array of numbers with the same number of elements per second as the sampling rate. However, this representation does not contain information useful for speech recognition. To counter this problem, the raw audio signal can be converted to the frequency domain using fast Fourier transform (FFT) on small audio window. While the FFT contains energy information at each frequency band in the audio window, it does not put emphasis on the band that is important for human hearing, which is below roughly 1000 Hz. To overcome this problem, Stevens et al. [53] suggested a scale, as shown in (1) , to improve the emphasis on the frequency important to human hearing.

$$mel(f) = 1127ln(1 + \frac{f}{700}) \tag{1}$$

The mel-frequency cepstrum coefficient (MFCC) is a speech signal feature commonly used in ASR research as well as music classification research. Figure 2 shows the process of extracting MFCCs for a speech signal.

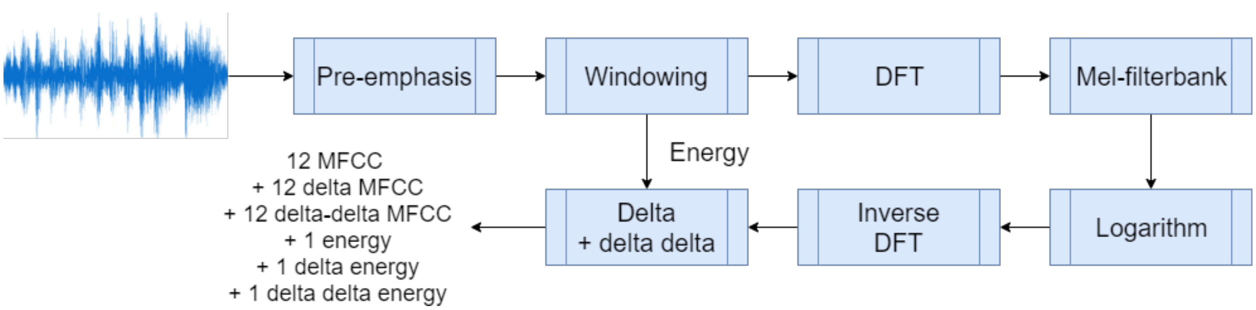

Figure 2: *The process of extracting MFCCs from a raw audio signal focuses on the frequencies that are important to human hearing.*

Since the characteristic of speech changes throughout an utterance, spectral features obtained over the entire utterance would not convey useful information. Instead, features are typically extracted over a small window, typically 25 ms, strided by 10 ms. Each window is then passed through the pipeline shown in Figure 2. The pre-emphasis step in Figure 2 aims to boost the energy of the signal at high frequency since the high frequency of human speech typically has lower energy than low frequency, but is also important to speech recognition task. After the pre-emphasis step, the windowing step involves multiplying the signal with a pre-defined window. While a rectangular window (Figure 3a) is the simplest window to use, it causes the signal to be abruptly cuts off at the edge. The cutoff causes problems when the discrete Fourier transform of the signal is obtained. Instead, a Hamming window (Figure 3b) is used. A Hamming window approaches 0 at its edges, which shrinks the value of the input signal toward 0 at the boundaries [36].

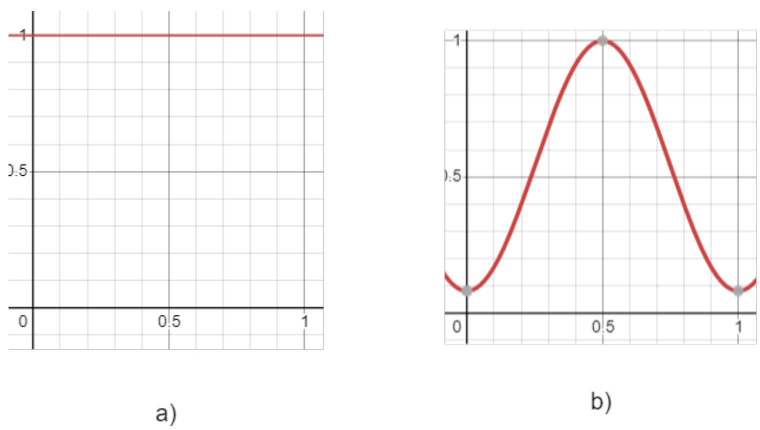

Figure 3: *A rectangular window (a) is simpliest to implement but results in audio being abruptly cuts off at the edges. A Hamming window (b) avoids this problem by fading toward 0 at the edges.*

The discrete Fourier transform (DFT) of each window is then computed. To approximate the mel scale described in (1), mel-filterbanks are then used to filter the output from the DFT. Typical mel-filterbanks consists of 10 linearly-spaced filters from 0 to 1000 Hz. Above 1000 Hz, the filters are spaced logarithmically. The spacing of the filters put more emphasis on the lower frequency, similar to how the mel scale operates. The logarithmic of the output is then taken to reduce variations in the input. While the log of the mel spectrum can be used as input features to an ASR system, it does not separate the information from the glottal pulse, which affects the speaker's fundamental frequency, from the information about the vocal tract, which shapes the phones. An inverse DFT operation is then performed to obtain the cepstrum of the signal. A cepstrum aims at separating the fundamental frequency, which is more important for speaker identification and ASR for tonal languages, from the shape of the vocal tract, which is important to speech recognition tasks. The lower cepstral values contain information related to the vocal tract, while the higher cepstral values contain information related to the glottal pulse. Prior research [36] have shown that the first 12 cepstral values are typically used in ASR application. In addition to the cepstral values, the average energy level over each window is also added as a feature. Finally, the first and second derivatives of the cepstrum as well as the energy level from each window is taken to obtain 39 MFCCs.

3.3 Connectionist Temporal Classification (CTC)

One of the challenges of LVCSR is the alignment of the audio and ground truth text. Because the input is an audio stream while the output is typically a string of characters or words, manual ground truth alignment can be very time consuming and labor intensive. Since the length of speech can be different between speakers for the same ground truth, each character of the transcription must be aligned to the exact location in the audio for training. For either recurrent or convolutional architectures, the number of input audio windows is often much greater than the number of characters in the target transcripts. The CTC operation [23], allows systems to be trained without the need for alignment.

The CTC loss function avoids the alignment problem by summing over the probability of all possible alignments between the encoded text and the ground truth. To obtain an alignment, an encoded text consisting of repeating characters in consecutive timesteps can be collapsed into a single character. A special CTC blank token ("_") is used to separate consecutive characters that should not be collapsed.

A CTC-trained neural network learns to produce the encoded text from a given input sequence. At each timestep, the network produces the probability for each character in the alphabet, including the CTC blank. The sum of the probability of all possible alignments corresponding to the ground truth yields a score for the encoded text. The CTC loss is simply the negative log likelihood of the score for the encoded text. Figure 4 shows an example of how CTC score can be computed using an example of 2 timesteps and 3 unique tokens.

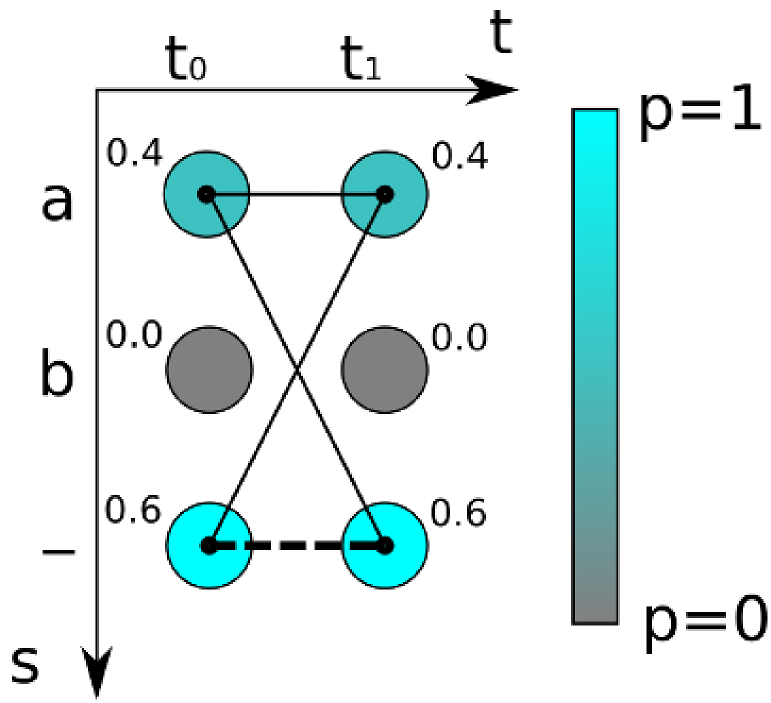

Figure 4: *The output of a neural network is color coded and also labelled for each timestep. The solid paths represent the alignments corresponding to the ground truth 'a', while the dashed path represents the only possible alignment for the output ''* [4].

As seen in Figure 4, there are 3 different paths which can lead to the output 'a', namely 'aa' with probability $0.4 \times 0.4 = 0.16$, 'a_' (probability $0.4 \times 0.6 = 0.24$), and '_a' (probability $0.6 \times 0.4 = 0.24$). There is only 1 path corresponding to the output '', namely '__' (probability $0.6 \times 0.6 = 0.36$). The CTC score for the output 'a' is the sum of the probabilities corresponding to all paths that produce output 'a', which is $0.16 + 0.24 + 0.24 = 0.64$. The CTC score

[4] www.towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c

for the output ‘’, computed in similar fashion, is 0.36. The CTC loss can be obtained by computing the negative log of the probability of the path leading to the correct ground truth.

3.4 Language modelling

As shown in Figure 1, an ASR system can include a language model (LM) along with the acoustic model to obtain better performance. While the use of a LM is not necessary to obtain output from the system, it is often critical to obtaining usable output. The goal of a LM is to determine the most likely sequence of words or characters given the output of the acoustic model. With the advancement of deep learning, neural language models such as RNN-based models [45, 54, 24], CNN-based model [18], or attention-based models [19, 50] have been proposed. However, n-gram language models are still common used for decoding in ASR tasks. In an N-gram language models, the probabilities of sequences of 1 to N tokens, where a token is a word, are computed from a given text corpora, such as the transcripts of the training data or written data obtained from books, webpages, historical documents, etc. Additional processing steps such as smoothing and pruning of the probability can be applied to simplify the language models. For the experiments performed in this research, N-gram language models generating with the KenLM tool [30] were utilized.

3.5 Generative Adversarial Networks (GANs)

While deep learning becomes prevalent due to discriminative models used in image classification, Goodfellow et al. [21] proposed a generative model that can learn the underlying distribution of the data based on adversarial training. Figure 5 shows the architecture of a basic generative adversarial network (GAN).

A GAN pits a discriminator D against a generator G. The goal of the generator is to generate an output that closely resembles a sample from a distribution, while the discriminator seeks to distinguish an input as either the output of the generator or a real sample from the

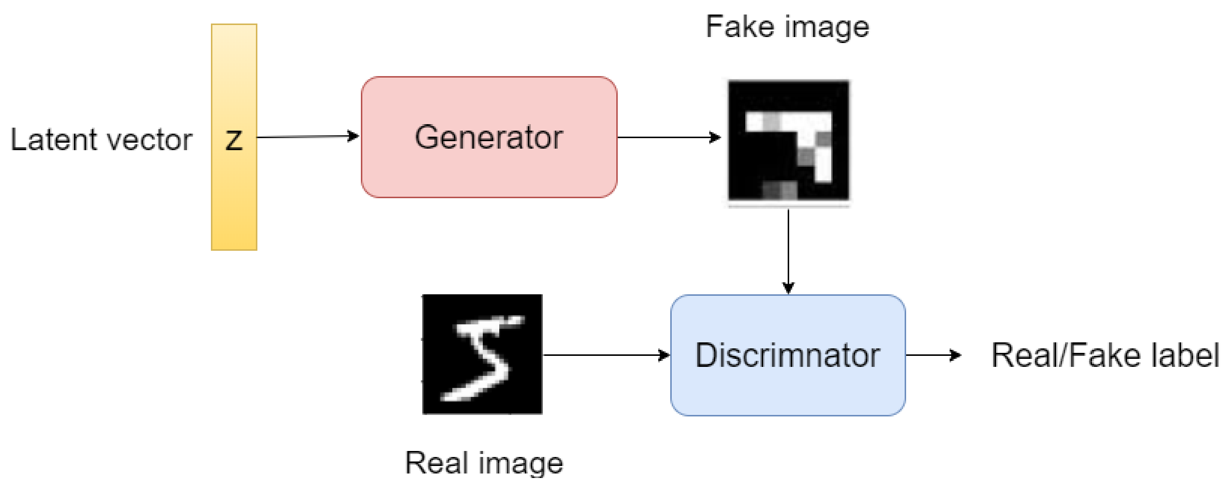

Figure 5: *A generative adversarial network aims to create a realistic image from a latent vector space by training a generator to generate the fake image and a discriminator to detect such fake images.*

distribution. These objectives can be represented by (2).

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim P_{data}(x)}[log D(x)] + \mathbb{E}_{z \sim P_z(z)} log(1 - D(G(z)))] \tag{2}$$

In (2), the generator aims to minimize the probability that the discriminator would classify the generated output as fake. The discriminator, on the other hand, aims to maximize the probability of making the correct prediction. As the discriminator gets better at detecting the fake images generated by the generator, the generator also learns to generate more realistic images that can fool the discriminator. As a result, the generator can eventually generate images that closely resemble those used to train the network.

By adding different constraints to the architecture of the network, GANs can be used to generate samples that maintain certain characteristics while changing others. Using convolution and transposed convolution instead of vanilla feed forward layers, Radford et al. [49] proposed DCGAN, which can represent local dependencies and texture better. Zhu et al.

[66] managed to generate images of a different style while still maintaining the structure of the images by using cycle-consistency loss to ensure images can be converted between two styles. Choi et al. [15] proposed a single network that can perform mutli-domain image translation by adding a domain classifier which can identify which domain the output image belongs to. With the domain classifier, the generator has to learn the characteristics of the target domain.

3.6 Recurrent Neural Network (RNN) and Long-Short Term Memory cells (LSTM)

For sequence understanding, it is important to have information from not only the current timestep, but also from tiemsteps before, and optionally after. A typical feed-forward neural network does not have the capability to process information at different timesteps. However, a RNN is designed to learn temporal dependencies. Figure 6 shows the difference between a feed forward layer and a recurrent layer.

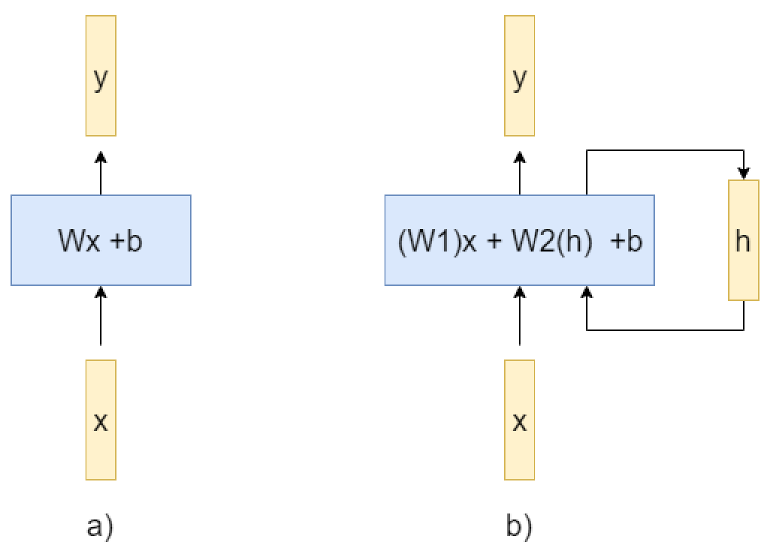

Figure 6: *A feed-forward layer (a) produces the output vector (y) based on an input vector x. A recurrent layer (b) produces the output vector y based on an input vector x and a hidden state h.*

In a feed-forward (or fully-connected) layer, the output is computed using (3), where W is a weight matrix, and b is a bias vector. The output of a recurrent layer, however, is computed using (4), where W_1, W_2, W_3, W_4 are weight matrices, h is a hidden vector, and b and c are a bias vectors.

$$y = Wx + b \tag{3}$$

$$y = W_1 x + W_2 h + b \tag{4}$$

$$h = W_3 x + W_4 h + c \tag{5}$$

At each timestep, a recurrent layer computes the prediction as well as the next hidden state based on the previous timestep hidden state and the input at the current timestep. Figure 7 demonstrates the use of the hidden state to obtain information across many timesteps. While Figure 7 only shows the information being passed in one direction, a bidirectional RNN can also propagate information from future timestep to past timestep.

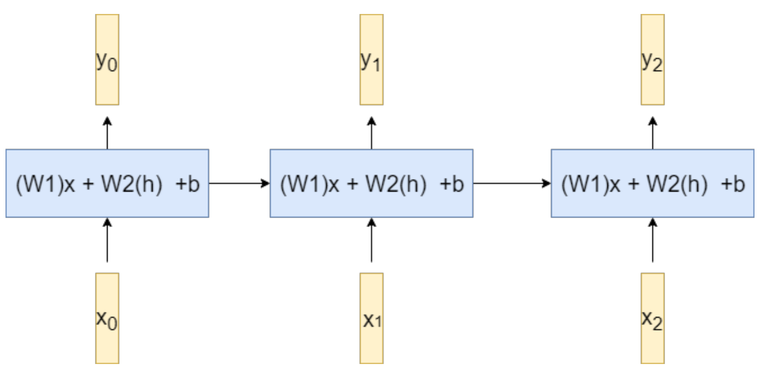

Figure 7: *At each timestep, a recurrent layer computes its output based on the current timestep input and the hidden state from the previous timestep.*

While RNN manages to model sequences fairly effectively, it typically experiences vanishing or exploding gradient problems as the number of timesteps in a sequence grows. The vanishing gradient problem is caused by small gradients being multiplied together, eventually approaching 0. The exploding gradient, on the other hand, is caused by large gradients being multiplied together, eventually approaching infinity. To tackle these problems, Long-Short Term Memory cells (Figure 8) was introduced.

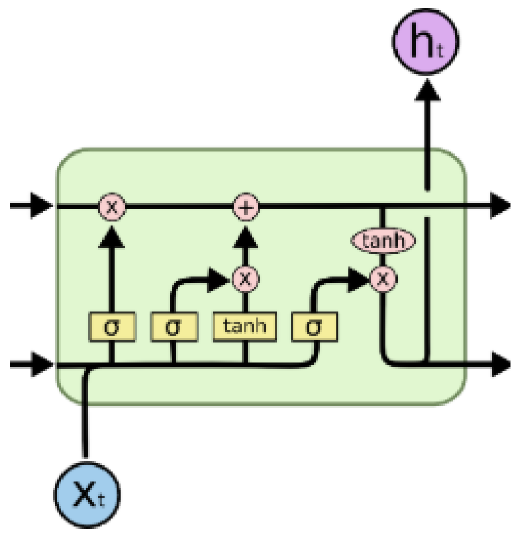

Figure 8: *An LSTM cell contains gates which allow for the control of how much information from previous timesteps is passed through*[6].

An LSTM consists of a forget gate, represented by the left-most sigmoid activation function, an input gate, represented by the middle sigmoid and tanh function, and an output gate, represented by the last sigmoid. The forget gate controls how much of information from past timesteps to affect the output of the current timestep. The input gate computes the new cell state C from past hidden state and the current input. This cell state is then added to the previous cell state, whose information has been controlled by the forget gate. Finally, the new cell state and the current timestep input is used to compute the current timestep output and the new hidden state.

[6]https://colah.github.io/posts/2015-08-Understanding-LSTMs/

3.7 DeepSpeech

The baseline model for deep learning ASR is DeepSpeech [28], implemented by Mozilla [7]. The model (Figure 9) consists of 6 layers: layers 1, 2, 3, and 5 are fully-connected layers of 2048 units, layer 4 is a recurrent layer with 2048 LSTM cells, and layer 6 is a fully-connected layer that predicts a character for a timestep. The input to the model consists of 39 MFCCs taken with window size of 25 ms with 10 ms stride. The model is trained using the CTC loss function. Using this model, different data augmentation techniques were evaluated.

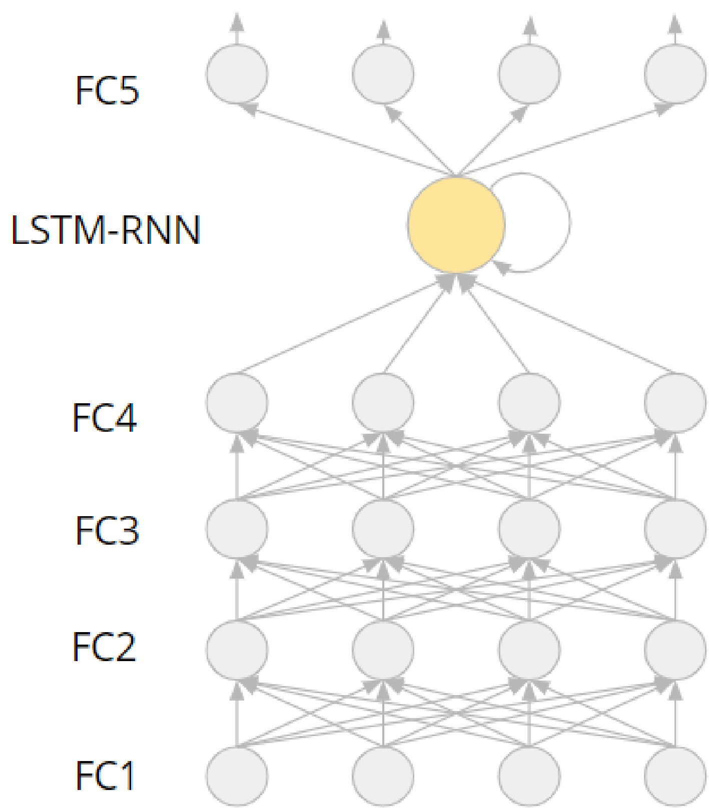

Figure 9: *The DeepSpeech architecture consists of fully-connected layers followed by a recurrent layer that captures temporal dependencies, and a fully-connected layer to predict output characters.*

[7] https://github.com/mozilla/DeepSpeech/tree/v0.3.0

3.8 Audio Spoofing

With the popularity of social media, the spreading of false information has recently become an important research area for public safety and cybersecurity. False information can be in the form of fake written news or fake media. The advancement of technology, in general, and deep learning, in particular, has lead to the creation of tools that allow the generation of images, videos, and audio segments that are not real but cannot be easily identified by human. While some of these tools can be helpful, they also lead to the generation of fake media which could have significant consequences in politics, economy, and public safety.

An audio spoofing attack happens when an attacker uses a fake audio signal to mimic the identity of another person. There are two main categories of audio spoofing attacks: Logical Access (LA) and Physical Access (PA). In the LA case, the spoofed audio is generated using either a text-to-speech or a voice conversion software. With a text-to-speech synthesis software, a text input can be converted to a speech output with a voice similar to that of a target speaker. A voice conversion software allows a speech given by a source speaker to be converted to a different utterance which has the same linguistic content (i.e. textual information) but with a different speaker identity. The PA scenario focuses on security systems that use voice detection. In this scenario, a pre-recorded spech from the target speaker is replayed in order to trick the security system to believe the target speaker is actually speaking. For this research, we focus on the LA scenario since this type of attack allows the adversary to easily create fake media by falsely representing the identity of the speaker or falsely creating the content with a given speaker identity.

LA attacks often involve the usage of text-to-speech (TTS) synthesis systems or voice conversion (VC) systems. Most voice conversion systems use a neural-network-based and spectral-filtering-based approaches [44]. Additionally, GANs have also allowed the creation of non-parallel voice conversion systems, which do not require the source and target speakers to say the same content.

3.9 Constant Q Cepstral Coefficients (CQCC)

The constant Q transform (CQT), first introduced for music processing [8], is a signal processing technique aimed at producing a spectrum with a constant Q factor. The Q-factor (6) of a filter is defined as the ratio between the the center frequency (f_c) and the bandwidth of the filter w.

$$Q = \frac{f_c}{w} \tag{6}$$

Human hearing have been shown to approximate a constant Q factor between 500 Hz and 20kHz [46]. The CQT aims at obtaining a spectrum with a constant Q factor by using geometrically spaced frequency bins instead of linearly spaced frequency bins as seen in Fourier-based methods. The advantage of the CQT over Fourier-based transformation is it provides higher frequency resolution at lower frequencies and higher temporal resolution at higher frequencies.

The CQCC can be obtained by first performing the CQT on the input signal. A power spectrum is then computed from the output of the CQT, and its logarithm is taken. Afterward, a uniform re-sampling of the output is then performed before a discrete cosine transform is taken to obtain the CQCC. Details of how to obtain the CQCC can be found at [57].

4 Previous works

4.1 Resource Constrained ASR

When given sufficient in-domain monolingual training data, deep neural network methods for ASR often perform significantly better than traditional methods based on HMMs and GMMs [31, 22, 27, 4, 10, 65, 14, 2]. Common approaches for deep learning ASR rely on RNNs: sequence-to-sequence models like that in Chen et al. [10] use RNNs to generate a latent representation of the utterance before decoding with RNNs, while DeepSpeech 1 and DeepSpeech 2 [27, 4] use RNNs to capture temporal dependencies before making predictions for each timestep. Methods that produce characters, such as versions of DeepSpeech, currently use Connectionist Temporal Classification (CTC) [23] to reduce streams of characters to plausible words by combining consecutive similar characters and pauses during speech.

Convolutional architectures have achieved remarkable results in computer vision tasks such as image classification [55, 64, 29, 55, 64] and image segmentation [40]. The first successful breakthrough of convolutional neural network came in 2012 when Krizhevsky et al. [38] achieves one of the best results in the ImageNet competition using AlexNet. Szegedy et al. [55] introduced the concept of an Inception block which consists of convolution with multiple filter sizes in a layer to capture different levels of regional dependencies. This concept can be applied to sequential data like speech by using filters with different widths to simultaneously capture different temporal dependencies. The Inception network also introduces 1× bottleneck filters to reduce the number of parameters in a model by reducing the number of filter maps being passed into the main filters in each Inception block. He et al. [29] came up with the idea of skip-connection to improve the flow gradient and improve convergence rate by allowing the model to learn the difference between the output and input rather than learning the transformation from input to output. Xie et al. [64] use Inception-like blocks but with similar filter sizes while adding skip connections similar to ResNet to allow for better gradient flow.

Previous experiments have shown that transfer learning from a model trained on resource-rich languages can improve the performance of ASR for low-resource languages [20, 32]. Using synthetic data has also been found to yield improvements in true low-resource, artificially low-resource, and resource-rich conditions [61, 7, 63]. Augmentation methods such as adding background noise, changing fundamental frequency, modifying speaking rate, and other distortions of the signal lead to WER reductions for ASR systems trained on less than two hours of audio [33]. Carmantini et al. [9] introduced sample overgeneration during initialization for low-resource ASR for improved semi-supervised training on lattice-free maximum mutual information (LF-MMI) [43]. Malhotra el at. [42] selected samples with lower confidence in an active learning scenario for low-resource ASR.

Rosenberg et al. [51] investigated the use of a CTC-based RNN and an RNN Encoder-Decoder network in character-based end-to-end ASR for low-resource languages. While recurrent-based models have demonstrated usefulness in ASR and other sequence modeling tasks, these models cannot easily take advantage of parallelization on modern hardware since the output of an RNN cell at each timestep depends on the results from the previous timestep. To mitigate this problem, Collobert et al. [16] relies on convolution to capture temporal dependencies.

The fully convolutional, character-based architecture proposed by Collobert et al. [16] still requires training models with large numbers of parameters. Additionally, these models have a high number of layers causing the models to converge more slowly. Our proposed model aims to reduce the complexity of the model without reducing performance by using bottleneck filters and skip connections. Additionally, instead of relying on different layers to capture different levels of temporal dependencies, we combine filters with different widths into one layer to reduce the number of layers in the model while still maintaining a wide context window.

Another approach to mitigate the training time problem with RNN is the use of self-attention models. Vaswani et al. [62] proposes a self-attention model, called Transformer model, as a way to perform sequence-to-sequence tasks without relying on recurrent connections. The Transformer model relies on the concept of Multi-Headed Attention (MHA) (Figure 10) to emphasize important parts of a sequence. Each MHA consists of multiple scaled dot-product attention modules, each compute different attention matrices to allow the MHA to focus on multiple parts of the sequence at once. Since the sequence is represented as a matrix, a MHA module can compute scores for all timesteps at once.

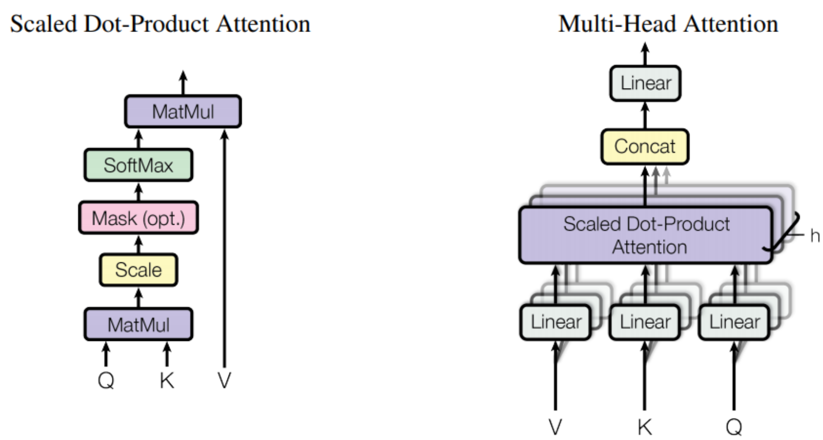

Figure 10: *The main building block of Transformer models are Multi-Headed Attention blocks (right), which computes multiple scaled dot-product attentions (left) for each sequence.*

Another proposed method for deep learning without a large volume of labelled data is self-supervised learning. In self-supervised learning, unlabelled data can be used to train a neural network on pre-tasks. Such pre-tasks do not require knowing the groundtruth of the data. Some examples of pre-tasks are: determining a missing element in a sequence or picture based on surrounding elements, determining the correct order of a scrambled sequence. When trained on such pre-tasks, the neural network can learn rich feature representations of the input, which can then be used to train the network on the targeted task such as image classification or speech recognition.

Noroozi et al. [47] proposed a method of self-supervised learning for images by training a network to solve a jigsaw puzzle of a small image patch. van den Oord et al. [48] proposes a method of self-supervised learning by asking a neural network to predict the next image patch, given several other incorrect image patches, based on previous image patches. Schneider et al. [52] takes this idea and applied it to audio representation, in a method called Wav2Vec, by training two neural networks: an encoding network and a context network (Figure 11. The encoding network downsamples a raw audio while also learning representations from the raw audio, called embeddings. The context network produces a vector which is used to predict the correct embedding of several future timesteps.

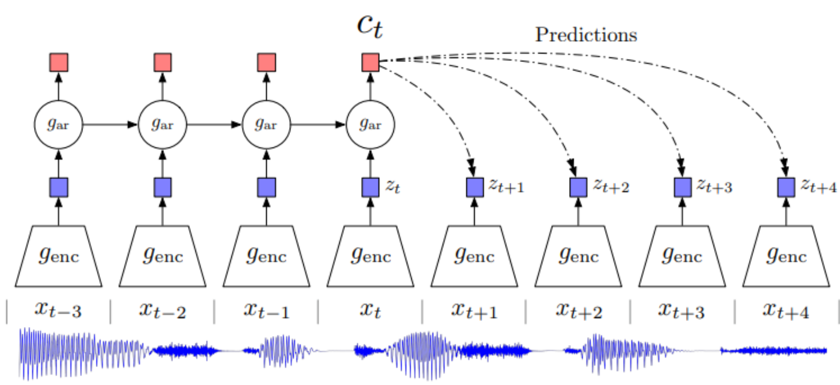

Figure 11: *In Wav2Vec, the context network (g_{ar}) is used to obtain a context vector c_t which is then used to predict future encoding z_t. The encoding is obtained from raw audio using an encoder g_{enc}.*

While transfer learning and data augmentation separately have both shown improvements, we explore the effectiveness of combining both concepts on low resource ASR, as well as a final finetuning step using only unaugmented data to prevent digital artifacts in augmented data from degrading performance.

4.2 Audio spoofing detection

To counter the spoofing problem, different speech features, such as constant-Q cepstrum coefficients (CQCC) [58], or MFCCs, have been combined with GMM and DNN to produce a detection system. While these systems have performed very well against PA scenarios, there have not been any performance benchmark on LA scenarios. Additionally, the result of the Voice Conversion Challenge 2018 [41] shows many promising parallel and non-parallel voice conversion techniques, which allow fake audio to be created much easier.

In detecting fake images and videos, two main families of neural networks are often used: RNN-based architecture with CNN feature extractors, and pure CNN. One problem with fake audio and images is the frequency or resolution at which digital artifacts remain. To combat this problem in image forgery, Afchar et al. [1] opts to examine the mesoscopic level of images to identify forgery in a network called MesoNet. MesoNet consists of layers with different kernel sizes, which are aimed to examine different resolution levels. This idea can be extended to audio spoofing detection by combining kernels of multiple sizes to detect artifacts at different frequencies.

Another method that has been used to detect fake videos is Convolutional-Recurrent Neural Network (C-RNN) [26]. In this method, a convolutional neural network is used as feature extractor for each frame in the video. The features are then passed into a Long-short Term Memory (LSTM) cell, which will produce a vector which can be used to determine whether the entire video is fake or not. Similarly, for audio, audio features, obtained via signal processing techniques or via a CNN, can be passed into an LSTM and then through a classification layer to determine whether the audio is spoofed.

For spoofing countermeasure, Chettri et al. [13] uses an ensemble of 2-D CNN trained on cropped or padded audio, convolution-recurrent networks, 1-D CNN, Gaussian Mixture Model, and Support Vector Machines to perform spoofing detection on the ASVSpoof 2019 dataset. For the same dataset, Alzantot et al. [3] uses a 2-D CNN with skip-connection, trained on fixed size crops of audio to perform countermeasure.

5 Datasets

5.1 Resource Constrained ASR

5.1.1 Seneca dataset

The primary target language for this research is Seneca, a morphologically complex and endangered language spoken by Native Americans in the Western New York area. Currently, there are roughly only 50 individuals who speak Seneca as their first language, and a few hundred who speak Seneca as their second language. The raw audio recordings used in our experiments consist of roughly 720 minutes of conversational, spontaneous speech from eleven adult speakers, eight male and three female. All speakers use Seneca as their first language, with English as a second language. All speakers are middle-aged or elderly. The audio was recorded using various equipment, in different environemnts, and was made over many years. As such, the recordings have a diverse range of audio quality. The recordings were segmented at the utterance level and transcribed using Seneca's current orthographic conventions by second-language Seneca speakers.

The transcribed audio data was partitioned into a 10-hour training set and a 2-hour test set as follows. Using the utterance boundaries provided in the reference transcripts, we randomly selected individual utterances from the full corpus of twelve hours until we had compiled ten hours of audio for training. The remaining two hours comprise the test set. We deliberately selected utterances in a random fashion in order to maximize diversity of gender, age, dialect, voice quality, and content (e.g., narrative vs. conversation) of both the training and test sets and to avoid overfitting to any particular speaker or speaker characteristic. We note that selecting the test data in this way has the effect that certain speakers appear in both the testing and training data, a compromise we are obliged to make given the very small number of available speakers of the language.

Approximately six hours of the audio data consists of casual conversations between one of the authors and a Seneca elder dealing with current events, the weather, and anecdotes from the elder's childhood. The remaining audio data was collected from a variety of other speakers and consists of community narratives and information about the natural world. In addition to transcriptions of this audio (roughly 35,000 words), the text data used to train the language model includes an additional 6000 words of previously transcribed texts for which there are no corresponding audio recordings.

5.1.2 Iban dataset

In order to determine how well the architectures and techniques investigated work for low-resource languages in general, the Iban language was also used to evaluate the performance of the system. Iban is a Malayic language spoken in Malaysia, Indonesia, and Brunei. There are about 800,000 estimated native speakers.

The Iban dataset [35] consists of 8 hours of audio obtained from local radio and television stations in Malaysia. The transcription was performed by Iban native speakers. The dataset was split into 71 minutes of testing data, consisting of 6 speakers (2 males, 4 females), and 408 minutes of training data, consisting of 17 speakers (7 males, 10 females).

5.2 Audio Spoofing

5.2.1 ASVspoof2019 dataset

To evaluate the fake audio detection algorithm, we use the ASVspoof 2019 dataset [59]. The dataset consists of two spoofing scenarios: Logical access (LA) and Physical access (PA). In the LA scenario, the fake audio is created using a speech synthesis or speech conversion software. In the PA scenario, the fake audio is created by replaying a pre-recorded audio using an output device, such as a speaker. For this research, the focus will be on the logical access data since it is easier, and more dangerous, to create speeches from any given text using logical access methods. Table 1 shows the composition of speakers' trait in the dataset.

Table 1: *ASVspoof 2019 LA dataset speaker and attack type composition*

Partition	**Male speakers**	**Female speakers**	**Attack algorithms**
Training	8	12	Known
Development (Validation)	8	12	Known
Evaluation (Test)	21	27	Known, Unknown

No speaker appears in more than one partition (training, development, or evaluation). All non-spoofed recordings were performed in the same condition. The training and development sets contain spoofed audio generated using the same algorithms. The evaluation set contains audio generated the algorithms used in the training and development sets as well as different algorithms. In total, there were 17 different spoofing systems used, with 6 systems designated as known attacks, and 11 designated as unknown attacks. The six known attack systems consist of 2 VC systems and 4 TTS systems, while the 11 unknown attacks consist of 2 VC systems, 6 TTS systems, and 3 hybrid TTS-VC systems. The training and development sets use only the 6 known attacks, while the evaluation set use 2 known attacks and all 11 unknown attacks. Thus, a good algorithm should be able to detect spoofing via unseen algorithms.

Table 2: *ASVspoof 2019 LA dataset ground truth composition*

Partition	**Spoofed samples**	**Bonafide samples**
Training	22800	2580
Development (Validation)	22296	2548
Evaluation (Test)	63882	7355

Table 2 shows the ground truth distribution of the dataset. Notably, in all partitions, the number of spoofed samples significantly outnumbers the number of bonafide samples. The ratio of spoofed samples to bonafide samples is approximately 10 : 1. This ratio can potentially lead to challenges in training the model since the model can achieve fairly high accuracy just by predicting any samples to be spoofed.

6 Methodology

6.1 Resource Constrained ASR

6.1.1 Data augmentation

Data augmentation involves generating new, synthetic data from existing data. For speech, data augmentation aims at creating new utterances with the same linguistic content (i.e. transcription) while modifying acoustic characteristics. Jimerson et al. [33] explored data augmentation via distortion of the speech signal, in which they added to the training corpus copies of the existing audio data that were modified to adjust the fundamental frequency (F0) and speaking rate or to include background noise. Here we focus on modifying pitch and speaking rate using the Pitch Synchronous Overlap and Add (PSOLA) algorithm [11]. For pitch augmentation, the F0 of the speech signal was varied in fractions of octaves ranging from 0.10 to 0.30 with a step size of 0.05. Speaking rate was adjusted by re-sampling the audio at multiples of the sampling frequency of the utterance ranging from 0.75 to 1.25 with a step size of 0.05. Each utterance in the training corpus was distorted 10 times with parameters randomly chosen and added to the existing training corpus, resulting in an additional 6000 minutes of audio data.

Another way of generating synthetic training data is via non-parallel voice conversion. Non-parallel voice conversion techniques do not require the source and target speakers to say the same speech in the training process. As such, these techniques are suitable for generating new data for low-resource languages where getting speakers to say the same speech is difficult. The StarGAN-VC model [37] modifies the image-based StarGAN [15] to acoustic features to perform many-to-many voice conversions. StarGAN implements a cycleGAN [66] architecture with an additional domain classifier, where the speaker identity was used as the domain. Figure 12 shows the overall architecture of StarGAN.

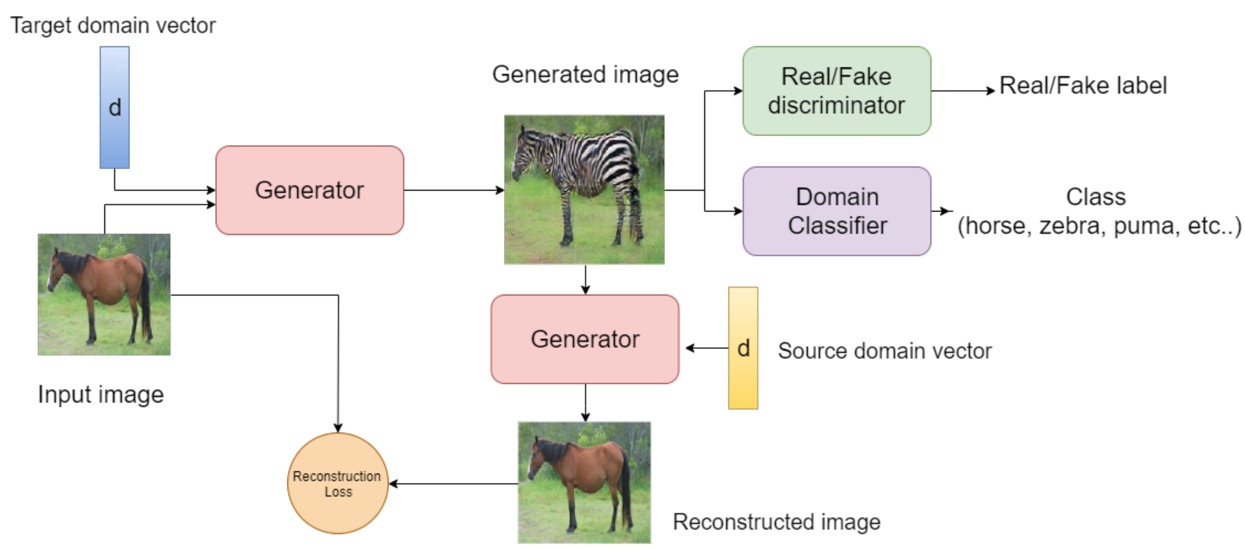

Figure 12: *StarGAN uses a domain classifier as well as domain vectors to allow many-to-many style conversion.*

As seen in Figure 12, the generator G takes an input image as well as a vector representing the target domain (zebra in the given example). Its goal is to generate an image that maintains the structure of the original image while has the style of the target domain. To ensure the same structure, a reconstructed image is created using the generated image and the domain vector from the source domain. The difference between the input image and the reconstructed image is the reconstruction loss. To ensure the style of the generated image matches the target domain style, a domain classifier is used to determine which domain the generated image belongs to. Lastly, similar to the GAN model proposed by Goodfellow et al. [21], a real/fake discriminator is used to make sure the generated image looks as realistic as possible. The loss function used to train the discriminator, which consists of a real/fake classifier and a domain classifier, is shown in (7), and the loss function for the generator is shown in (8).

$$\mathscr{L}_D = -\mathscr{L}_{adv} + \lambda_{cls}\mathscr{L}_{cls}^{real} \tag{7}$$

$$\mathscr{L}_G = \mathscr{L}_{adv} + \lambda_{cls}\mathscr{L}_{cls}^{fake} + \lambda_{rec}\mathscr{L}_{rec} \tag{8}$$

In (7) and (8), $\mathscr{L}_{adv}$ represents the loss obtained from the real and fake detection. $\mathscr{L}_{cls}$ represents the loss of the domain classifier, which is an n-class classifier, where n is the number of domains. The recovery loss, which captures the differences between the reconstructed image and the original input image, is represented by $\mathscr{L}_{rec}$. λ_{cls} and λ_{rec} represent the weights given for each of the loss to adjust the emphasis on either the domain similarity or the reconstruction similarity.

By replacing images with spectrograms and representing speaker identity with domain vectors, Kameoka et al. [37] proposed the architecture of StarGAN can be employed to perform many-to-many voice conversion. The output of StarGAN-VC is then converted back to raw audio using the Griffin-Lim algorithm [25].

For each of the voice conversion methods, we selected the three speakers with the largest volume of labelled data: Speaker A (94 minutes), Speaker B (250 minutes), and Speaker C (156 minutes). Since StarGAN-VC enables many-to-many voice conversion, only one StarGAN-VC model was trained to perform voice conversion among the three speakers. The StarGAN-VC model was trained for a total of 500,000 iterations, with sample outputs taken at every 50,000 iterations to subjectively determine whether the model produced intelligible utterances. A total of six VAWGAN models were trained to perform voice conversion among the three speakers since VAWGAN only allows for one-to-one voice conversion. Each VAWGAN model was trained for 100 epochs, with samples taken every 10 epochs to determine whether synthesized utterances were intelligible. The trained StarGAN-VC and VAWGAN models were then used to convert utterances from each of the three speakers to the other two. From the original 500 minutes of audio produced by Speakers A, B, and C, we obtained 1000 minutes of synthetic data for each voice conversion model.

6.1.2 Acoustic modelling

Mini-Gated Convolutional Neural Network (m-GCNN) Based on the models suggested by Collobert et al. [16] and Liptchinsky et al. [39], fully convolutional approach for ASR was explored. Since speech and acoustic signals can be represented as a 1-D array of numbers, 1-D convolution operations can be used to learn temporal dependencies similar to how 2-D convolution operations have been used to learn spatial dependencies for vision tasks. At each timestep, the combination of all feature maps in a finite temporal window can be used to obtain a feature map in the next layer. After several layers, the network makes a character prediction at each timestep. These character predictions can then be decoded using the CTC function. Since predictions are made for each timestep, the input to the network can be sequences of variable lengths.

To evaluate the effectiveness of fully convolutional architectures for ASR, we uses a more compact version of the Gated Convolutional Neural Network (GCNN) proposed in [39], called mini-GCNN or mGCNN. Figure 13 shows the overall architecture of mGCNN.

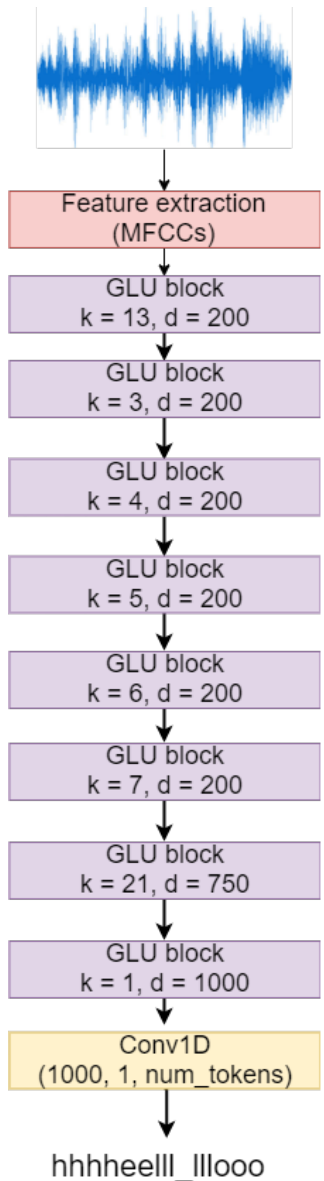

Figure 13: *The mGCNN consists of 8 Gated Linear Unit (GLU) blocks with different kernel widths and a final linear layer which predicts one character per timestep*

Compared to the original GCNN, mGCNN removes layers with kernel sizes from 8 to 20. This change was made to simplify the model in an attempt to reduce the number of parameters, making it more suitable for the small volume of training data. After the feature extraction step, a mid-size Gated Linear Unit (GLU) block is used to capture contextual information before a series of GLU blocks with increasing kernel size extracts features with increasingly wider temporal dependencies. Finally, a GLU block with kernel size 1 and a linear layer (represented as a convolution block with kernel size of 1) perform the prediction. Figure 14 shows the components of a GLU.

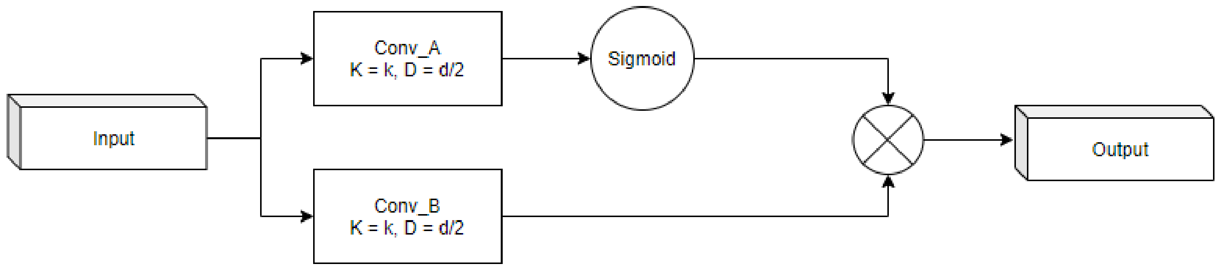

Figure 14: *A GLU consists of two paths: one performs the feature extraction (Conv_B) while the other (Conv_A) determines how much information from each feature map is passed to the next layer.*

A GLU consists of two convolution operations of the same kernel size. One of the convolution (the bottom path in Figure 14) performs feature extraction similar to a normal convolution operation. The other convolution (the top path in Figure 14), however, learns the importance of each feature map produced by the other path. As such, the output of this path is passed through a sigmoid layer, which is then multiplied with the output of the feature extraction path. With the ability to learn the importance of individual feature maps, a GLU allows for better convergence and gradient flow in the network. Optionally, a skip connection can also be used between the input and the output of the GLU to further improve the gradient flow.

Wide Inception REsidual Network (WIRENet) While mGCNN uses GLU blocks of different kernel sizes in series to capture different level of temporal dependencies, convolution operations can also be performed in parallel path, as demonstrated in Inception models [56]. Based on this intuition, we propose a novel network, called Wide Inception Residual Network or WIRENet, comprised of multiple blocks of convolutions with different filter sizes. Figure 15 shows the architecture of a Wide Block, the key building block of this network.

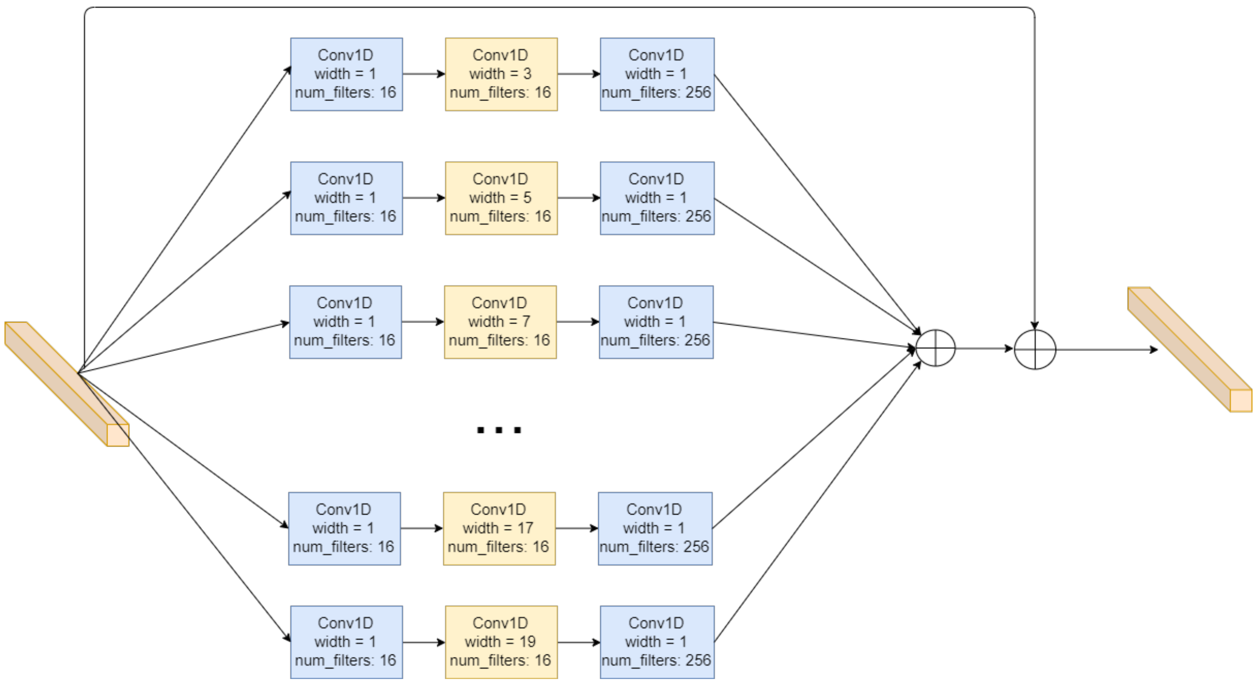

Figure 15: *A Wide Block contains multiple paths, each with different convolution kernel size, to learn different ranges of temporal dependencies.*

The main building block of our architecture is the WideBlock (Figure 15), named for the high number of paths in each block. The architecture of the block, taking inspiration from ResNeXt blocks used in image classification [64], consists of several parallel streams, each consisting of bottleneck 1×1 convolution layers before and after a normal convolution layer. The bottleneck layers reduce the complexity of the model by reducing the number of parameters required by the middle convolution operation. Instead of keeping the same filter size for all paths, we draw inspiration from Inception networks and employ filters with different sizes in each layer. The filter widths are odd numbers between 3 and 19. This choice is suitable for speech-related tasks since temporal dependencies in audio typically have more variance than spatial dependencies in visual tasks. The different filter sizes allow the model to pick up both short-term and long-term temporal dependencies. The output from each path is then summed before being added to the input of each block, forming a skip connection.

Figure 16 shows the overall architecture of WIRENet. The network consists of a mid-size and

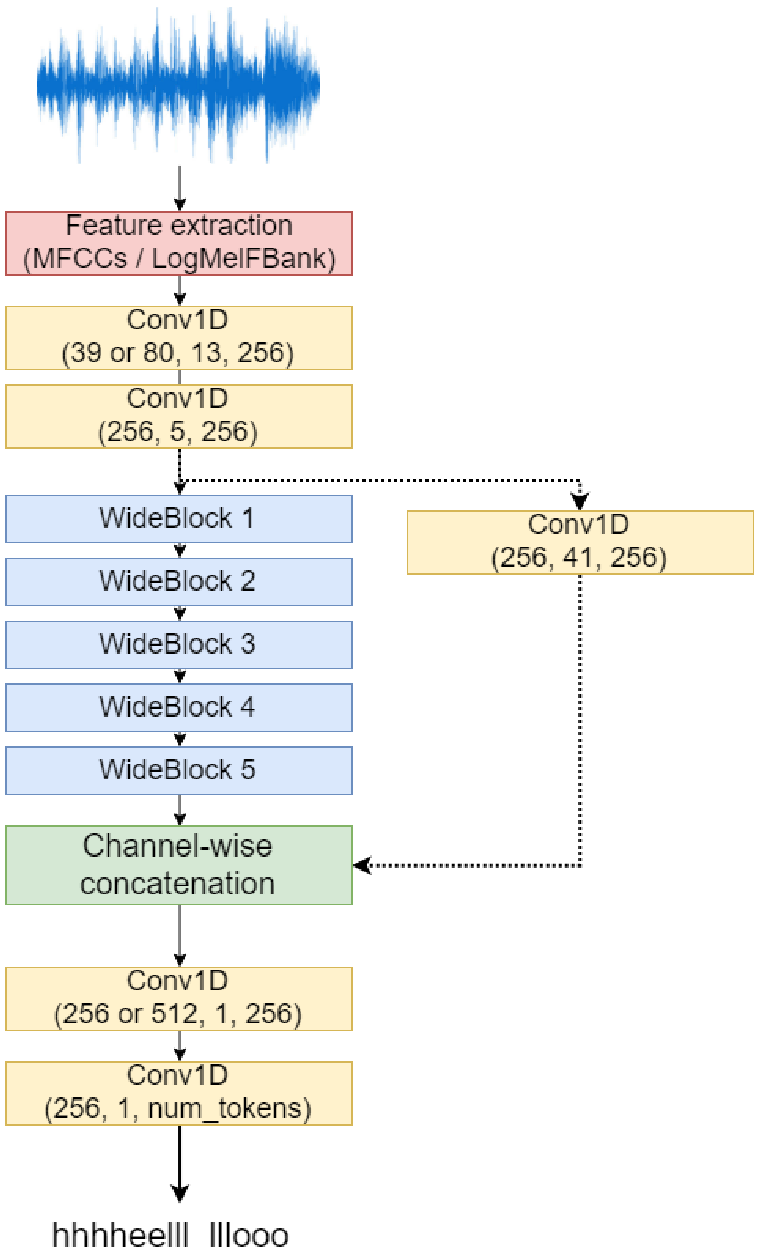

Figure 16: *WIRENet consists of 1-D convolution layers and Wide Blocks. The dimensions of the filters are: input channels, kernel width, output channels.*

small-size convolution layer after the feature extraction step. These embedding layers convert input audio features into a vector of desired depth and temporal content. The WIRENet architecture continues with five WideBlocks, then two 1×1 convolution layers which act as fully-connected layers. The final layer outputs a vector with size corresponding to the number of tokens to be predicted. Optionally, a context path, consisting of a convolution layer with very wide kernel width, can be used in parallel with the Wide Blocks to obtain context information from a wide temporal window before the prediction is made. Batch normalization and ReLU are used after each convolution operation. To prevent overfitting due to limited data, dropout layers of 0.25 are added after each Wide Block. To train the network, the CTC loss function is used.

6.1.3 Multi-staged learning

A proposed learning strategy to mitigate the lack of training data is to train the models in multiple stages. Figure 17 shows the proposed multi-stage learning scheme.

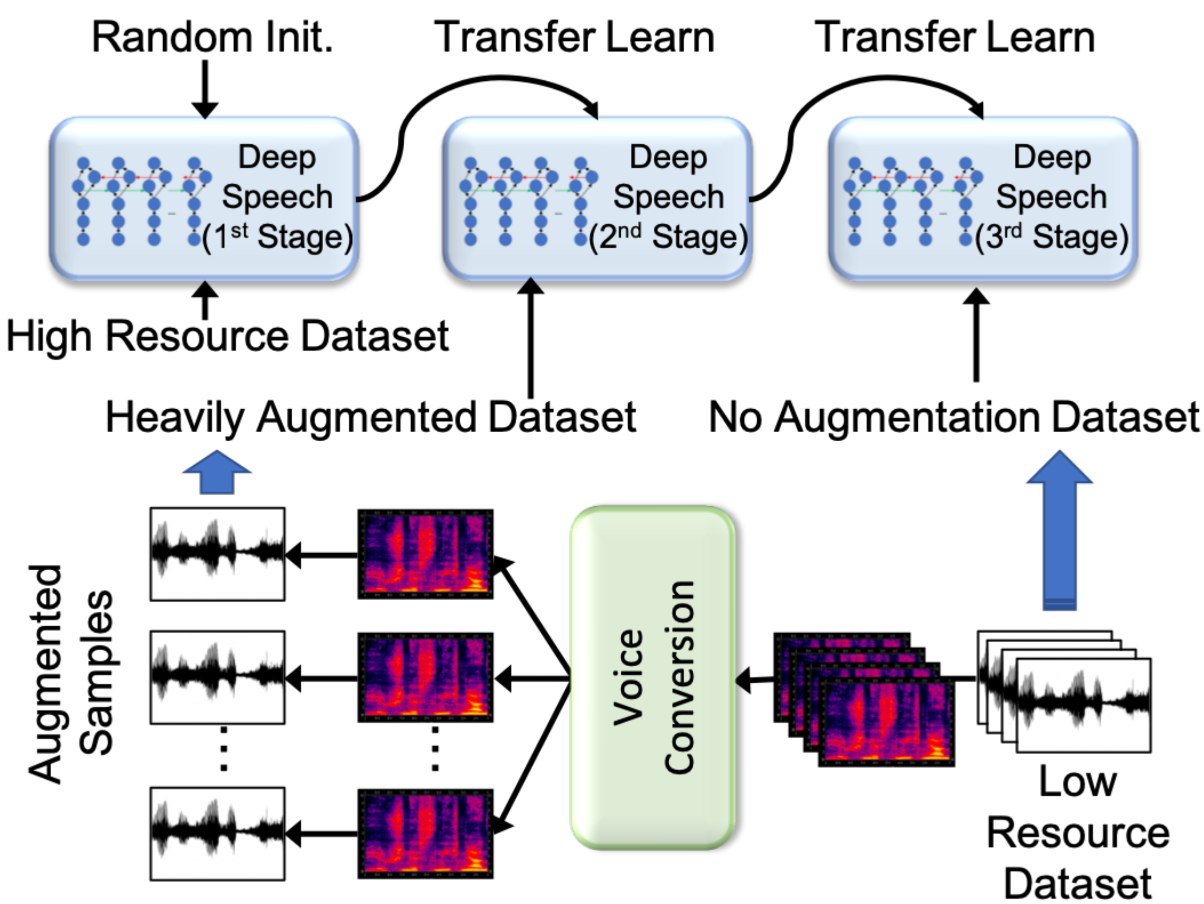

Figure 17: *Multiple learning stages can help extract useful features related to human speech (stage 1), the specific language (stage 2), and cleaning up the artifact from augmentations (stage 3)*

In the first stage, an acoustic model is trained on 960-hour of LibriSpeech English corpus, with randomized initial weights. The models are trained for 75 epochs, with the best models saved to initialize weights in later stage. The best models were determined by word-error rate on the LibriSpeech validation set.

In the second stage, the weights of a second acoustic model are initialized using the weights from the best English models obtained in the first stage. The training data from this second model includes the original unaltered 10-hour Seneca dataset as well as up to a 10× augmentation of each utterance using one of the three data augmentation methods: speed and pitch modification (Augment10), or voice conversion via StarGAN-VC. This model is trained until convergence on the training dataset.

The motivation for the final stage is that the augmented data often contains heavy digital artifacts. Since the amount of augmented data is significantly higher than the amount of original data, the networks trained on augmented and original data might be skewed towards improving performance on augmented data. However, it is hoped that the network can still learn valuable representation with augmented data, which will allow the final network trained on original data to perform better.

In the final stage, the weights of the third acoustic model are initialized using the final weights from the second model. The training data for the third models includes only the original 10 hours of unaltered Seneca dataset with no augmented data. Models in this fine-tuning stage are trained until convergence on the training dataset. For the final stage, the learning rate is reduced by an order of magnitude.

6.2 Audio Spoofing

6.2.1 Convolution-Recurrent Neural Network

The first proposed model for spoofing detection is a convolution-recurrent neural network. Figures 18 show the architecture of the proposed model. The model takes raw audio signal as an input and produces the log probability of two classes: bonafide speech and spoofed speech.

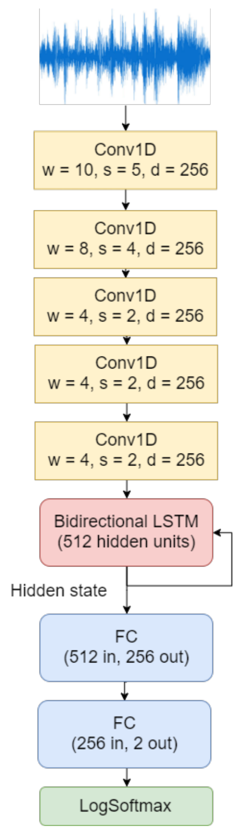

Figure 18: *The architecture of the proposed convolution-recurrent network consists of convolution layers to extract features and downsample the input audio, and a recurrent layer to obtain information from all the timesteps.*

Since raw audio is used instead of spectral features as the input to the network, five 1-D convolution layers are used to learn useful representation. Additionally, the convolutions are strided to downsample the input signal from 16kHz to 100 Hz, reducing the memory footprint while also speeding up the training process. The extracted features are then passed to a bidirectional Long-Short Term Memory (LSTM) layer. After the features from the last timestep is passed through the LSTM, the hidden state of the LSTM is used to perform prediction using two fully-connected layers. Dropout and batch normalization are used after each layer to perform regularization. The network is trained using negative log-likelihood loss function. Due to the unbalanced nature of the dataset, a miss classification of a spoofed speech incurs heavier loss than a miss classification of a bonafide speech.

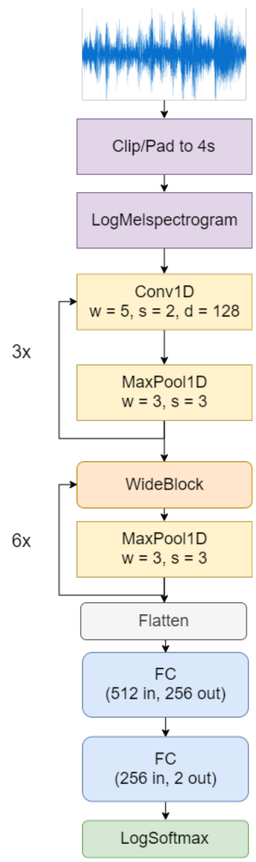

Figure 19: *The architecture of the proposed convolution approach for spoofing countermeasure uses Wide Blocks to extract information from different temporal window.*

6.2.2 Convolutional-based detection

Similar to the low-resource ASR problem, convolution neural networks are also examined for spoofing countermeasure. Since Wide Block shows promising results for ASR tasks, it is used as a key building block for the CNN countermeasure (Figure 19). Since a countermeasure only needs to produce one score for an entire utterance instead of one prediction per timestep, the architecture in Figure 19 uses strided convolution and max pooling operations to reduce the length of the feature map as it passes through the network. For similar reason, the input audio is either clipped or repeated to a fixed length of four seconds before the log-mel spectrogram is obtained. Dropout and batch normalization are used after each layer to avoid overfitting. Similar to the CRNN model, the convolution model is also trained using weighted negative log-likelihood.

7 Evaluation metrics

7.1 Low-resource ASR

7.1.1 Character/Word Error Rate

The performance of augmentation methods, acoustic models, and learning strategy is evaluated using word-error-rate (WER) and character-error-rate (CER). Both metrics are computed by aligning the output of the ASR system with the manually generated reference transcript. WER is the minimum edit distance over a word alignment, aggregated across utterances and normalized by the total number of words in the reference. CER is computed in the same manner, except the error is computed over characters instead of words. The minimum edit distance, normalized by the number of tokens, between two sequences is defined in (9), where I, D, and S represent the number of insertion, deletion, and substitution from to obtain one sequence from the other.

$$ED = \frac{I + D + S}{num\_tokens} \tag{9}$$

7.2 Audio spoofing detection

7.2.1 Equal Error Rate (EER)

To evaluate the effectiveness of a spoofing detection algorithm, the equal error rate (EER) is used. EER is typically used to evaluate biometric security system. For a biometric system, EER is used to determine the threshold value at which the false acceptance rate, or false positive rate (FPR), shown in (10), is equals to the false rejection rate, or false negative rate (FNR), shown in (11). When $FPR = FNR$, the common error rate is the EER of the system. Typically, lower EER corresponds to better accuracy of the system.

$$FPR = \frac{FP}{FP + TN} \tag{10}$$

$$FNR = \frac{FN}{FN + TP} \tag{11}$$

7.2.2 Tandem Detection Cost Function (t-DCF)

When a spoofing countermeasure is evaluated alone, EER is a good metric to evaluate different systems. However, spoofing countermeasures are often used in tandem with an automatic speaker verification (ASV) system. Thus, the effectiveness of a countermeasure system also needs to be evaluated with the ASV system in mind. To this end, the t-DCF is proposed as an evaluation metric by the organizer of ASVSpoof 2019 challenge [59]. Equation 12 shows how the t-DCF is computed, where P^{cm}_{miss} and P^{cm}_{fa} corresponding to the FNR and FPR of the countermeasure system, respectively. β represents the performance of the ASV system. For the ASVSpoof 2019 challenge, the organizer provides the ASV score, so β is a fixed value. β is inversely proportional to the false acceptance rate of the ASV system to a specific attack.

$$t - DCF_{norm}^{min} = \min_{s}\{\beta P_{miss}^{cm}(s) + P_{fa}^{cm}(s)\} \tag{12}$$

8 Results and Analysis

8.1 Low-resource ASR

8.1.1 DeepSpeech with data augmentation

Figure 20 shows the mel-spectrogram of an unaltered Seneca utterance along with its corresponding synthetically generated spectrograms for randomly chosen speed and pitch distortions, as well as the StarGAN-VC and VAWGAN augmentation methods.

As seen in Figure 20, the speed augmentation resulted in a longer utterance while still maintaining the shape of the spectrogram. The fundamental frequency of the pitch augmentation is higher compared to the original sample. The two voice conversion methods synthesize the utterance from speaker A as if it were spoken by speaker B. The peak locations in the two voice conversion methods are maintained, but the signature characteristics of the voice are transformed.

Table 3 shows the result of different augmentation methods and transfer learning stages on DeepSpeech. For each acoustic model, we evaluate with: 1) no language model (NO LM); and 2) a tri-gram language model ((3-GRAM). All WERs greater than 1.0 are replaced with 1.0, indicating no useful output was produced.

The baseline DeepSpeech model using only the 10 hours of unaugmented Seneca data yields

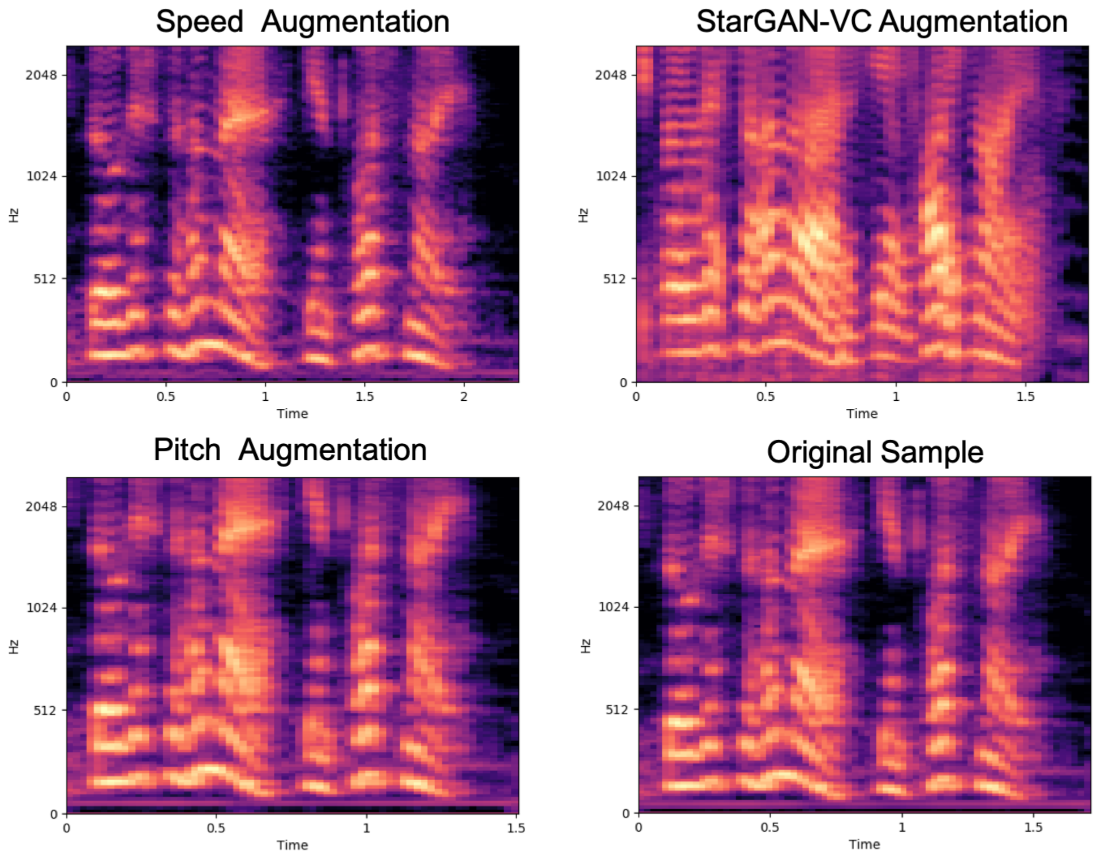

Figure 20: *Mel-spectrograms of a randomly selected utterance. The original utterance (mean F0=153, duration=2048msec) is on the bottom right, while the various augmentation methods, counter-clockwise from bottom left, are: pitch modification (mean F0=179), speaking rate modification (duration=3132msec), and StarGAN-VC.*

Table 3: *WER and CER across different acoustic models (Kaldi and DeepSpeech) and augmentation strategies (rows) vs. language models (columns).*

	WER		CER	
	NO LM	3-GRAM	NO LM	3-GRAM
Baseline: Kaldi	N/A	0.530	N/A	0.307
Baseline: DeepSpeech	1.000	0.970	0.891	0.872
Baseline: DeepSpeech w/TL	0.859	0.727	0.436	0.409
TL + Augment10	1.000	0.975	0.716	0.698
TL + Augment10 + finetune	0.850	0.693	0.427	0.421
TL + StarGAN-VC	0.911	0.790	0.497	0.474
TL + StarGAN-VC + finetune	0.772	**0.571**	0.364	**0.333**

a WER greater than 1.0, meaning it yielded no usable output. Applying a language model slightly reduces the error rate for this model. Applying transfer learning from English yields substantial improvement, with and without the use of a language model. However, the WER is still significantly higher than that obtained via a tri-phone acoustic model using the Kaldi toolkit. Using transfer learning and augmented data without finetuning the model did not significantly improve the performance for both simple signal distortion as well as voice conversion via StarGAN-VC. However, the result from StarGAN-VC augmentation is slightly better than that of signal distortion augmentation.

The most promising results are those of the Stage 3 data augmentation models, particularly StarGAN-VC and Augment10, both of which improve substantially over the baseline model using transfer learning, particularly when an n-gram language model is used during decoding. While the finetuned Augment10 model still has fairly high WER, the StarGAN-VC model begins to approach the WER of the traditional tri-phone HMM/GMM Kaldi model. We also see with this model a 40-point improvement over the Seneca-only baseline DeepSpeech model, and a 15-point improvement over the transfer learning baseline model.

This result indicates that transfer learning is critical to improving the performance of DeepSpeech for low-resource languages. In addition, data augmentation does provide performance increases for DeepSpeech when combined with transfer learning and, importantly, a fine-tuning step using un-augmented data. The improvement with fine-tuning justifies the need for the third stage of the training pipeline in order to reduce the effect of digital artifacts on WER.

8.1.2 mini-Gated Convolutional Neural Network (mGCNN)

Since data augmentation and multi-stage learning shows promising results for DeepSpeech, the same training regime was applied to train mGCNN acoustic model. Similar to DeepSpeech, the mGCNN models are evaluated using two language model configurations: no language model (No LM) and with a tri-gram language model (w/LM). Table 4 shows the results obtained for mGCNN models.

Table 4: *WER (word error rate) and CER (character error rate) for various transfer learning (TL) and augmentation strategies (rows) for mini-GCNN (m-GCNN) with and without a tri-gram language model.*

	m-GCNN (no LM)		m-GCNN (w/LM)	
	WER	CER	WER	CER
no TL, no Aug	0.839	0.365	0.426	0.257
+ TL, no Aug	0.766	0.299	0.350	0.186
Augment10 Stage2	0.702	0.266	0.372	0.184
Augment10 Stage3	**0.686**	**0.256**	0.364	0.179
StarGAN-VC Stage 2	0.817	0.364	0.488	0.311
StarGAN-VC Stage 3	0.691	0.263	**0.343**	**0.173**

The baseline m-GCNN model, trained without any transfer learning or data augmentation, produces better result compared to the baseline DeepSpeech model. With a language model, the baseline m-GCNN model achieves WER of 0.426 and CER of 0.257, which means the output from the model contains valid linguistic content, as opposed the WER of 0.970 and CER of 0.872 for DeepSpeech. This baseline m-GCNN model also shows a 10.4% absolute improvement in WER compared to the tri-phone GMM/HMM model obtained via Kaldi, and a 14.5% absolute improvement over the best DeepSpeech model in Table 3.

Similar to DeepSpeech, applying transfer learning from English yields further signficant improvement to the WER and CER. When combining transfer learning and data augmentation without fine-tuning, the WER does not improve compared to using only transfer learning. Notably, the WER for StarGAN-VC degrades. The fine-tuning step, however, still provides improvement in WER for the StarGAN-VC model compared to just using transfer learning or transfer learning and data augmentation. However, the improvement is not as significant as one saw in DeepSpeech model. The model trained with transfer learning, StarGAN-VC augmentation, and finetuning only achieves 0.7% absolute WER improvement over the model trained with only transfer learning. The minor improvement indicates that data augmentation does not have as significant impact on performance for convolution-based model as opposed to recurrent-based model.

8.1.3 Wide Inception REsidual Network (WIRENet)

Similar to DeepSpeech and m-GCNN, WIRENet models are evaluated using two language model configurations: no language model and a tri-gram language model. With WIRENet, only signal distortion augmentation was evaluated. Table 5 shows the results of different WIRENet models.

Table 5: *Seneca WER and CER for various transfer learning (TL) and augmentation (Aug) strategies (rows) for WIRENet without (*NO LM*) and with (*W/LM*) a trigram language model.*

	WIRENet (NO LM)		WIRENet (W/LM)	
	WER	CER	WER	CER
No TL, no Aug	0.839	0.365	0.421	0.257
TL, no Aug	0.785	0.328	0.337	0.199
TL + Augment10	0.730	0.303	0.319	0.194
TL + Augment10 + finetune	0.699	0.278	**0.299**	**0.175**

Without any transfer learning or data augmentation, the WIRENet model achieves approximately the same result as the baseline m-GCNN model. With transfer learning and no language model, the WIRENet model performs slightly worse than the m-GCNN model. However, when a language model is applied, the WIRENet model performs slightly better in terms of WER, despite a slightly worse CER. With transfer learning and data augmentation, and without the final finetuning stage, the WIRENet model produces minor improvement over the transfer learning-only model. The final finetuning stage further improves the performance of WIRENet, achieving 29.9% WER when a tri-gram model is applied. This result shows 4.4% absolute WER improvement over the result of the best m-GCNN model, and 6.5% absolute WER improvement over the m-GCNN model with the same augmentation, transfer learning, and finetuning.

Since the WIRENet model shows promising results using MFCCs as input features, we experimented with using the effectiveness of using a different input feature. Log-melfilterbank is obtained in a simlar fashion to MFCC, as seen in Figure 2, but without any steps following taking the log of the filterbank. Table 6 shows the result obtained by training WIRENet models using log-melfilterbank as input features instead of MFCCs.

Table 6: *Seneca WER and CER using WIRENet when substituting log mel-filterbank for MFCCs as input features without (*NO LM*) and with (*W/LM*) a trigram language model.*

	WIRENET (NO LM)		WIRENet(W/LM)	
	WER	CER	WER	CER
No TL, no Aug	0.784	0.307	0.369	0.197
TL, no Aug	0.768	0.309	0.302	0.172
TL + Augment10	0.758	0.324	0.307	0.187
TL + Augment10 + finetune	0.656	0.247	**0.243**	**0.130**

The baseline WIRENet model using log-melfilterbank achieves a WER of 36.9%, showing a 5.2% improvement over the baseline WIRENet model trained using MFCCs. Similar to all previous model, this WIRENet model benefits from transfer learning, achieving a further 6.7% WER improvement when transfer learning is applied. Finally, with signal distortion data augmentation and finetuning, this WIRENet model achieves 24.3% WER. This result shows 5.6% WER improvement over the best WIRENet model trained on MFCCs, 32.8% WER improvement over the best DeepSpeech model, and 28.7% WER improvement over the tri-phone HMM/GMM model.

We also experimented with adding a context path to extract information from a wider temporal window before making the character prediction, as seen in Figure 16. This model is trained on MFCCs, and the results are shown in Table 7.

The addition of the context path improves the performance of the model significantly compared to the results in Table 5. Without transfer learning, this WIRENet model improves by 12.9% absolute WER compared to the model without the context path, trained on MFCCs,

Table 7: *Seneca WER and CER using WIRENet with context path, without (NO LM) and with (W/LM) a trigram language model.*

	WIRENET(+C) (NO LM)		WIRENet(+c)(W/LM)	
	WER	CER	WER	CER
No TL, no Aug	0.774	0.324	0.292	0.179
TL, no Aug	0.732	0.289	0.252	0.154
TL + Augment10	0.655	0.257	0.236	0.144
TL + Augment10 + finetune	0.632	0.238	**0.234**	**0.130**

and by 7.7% compared to the model trained on Log-Melfilterbanks. Transfer learning and data augmentation still show minor improvement compared to the baseline model. With transfer learning, data augmentation and finetuning, we achieved a WER of 23.4%, which is slightly better than the best WER obtained without the context path. Figure 21 summarizes the performance, in terms of WER, for different acoustic models using different augmentation and transfer learning strategies.

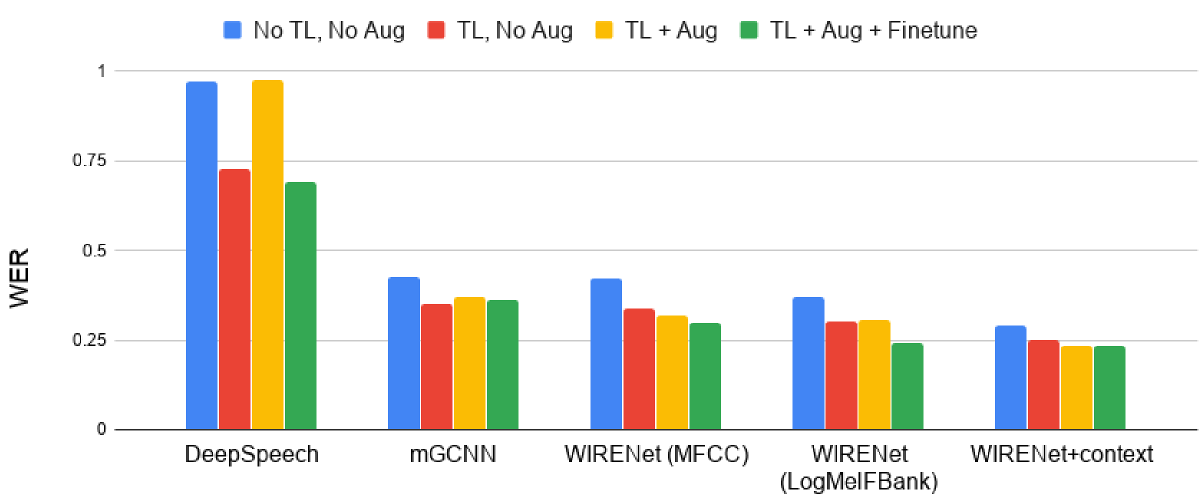

Figure 21: *WERs for all convolution-based models are lower than all DeepSpeech models on Seneca*

To understand the effectiveness of WIRENet given the amounts of training data, we performed an experiment with different amount of labelled Seneca training data. For this experiment, ten WIRENet with context models are trained on subsets containing 1 to 10 hours of labelled Seneca. Similarly, ten 3-gram language models are trained using only the transcripts of the labelled Seneca used to train the corresponding acoustic model. Figure 22 shows the result of this experiment.

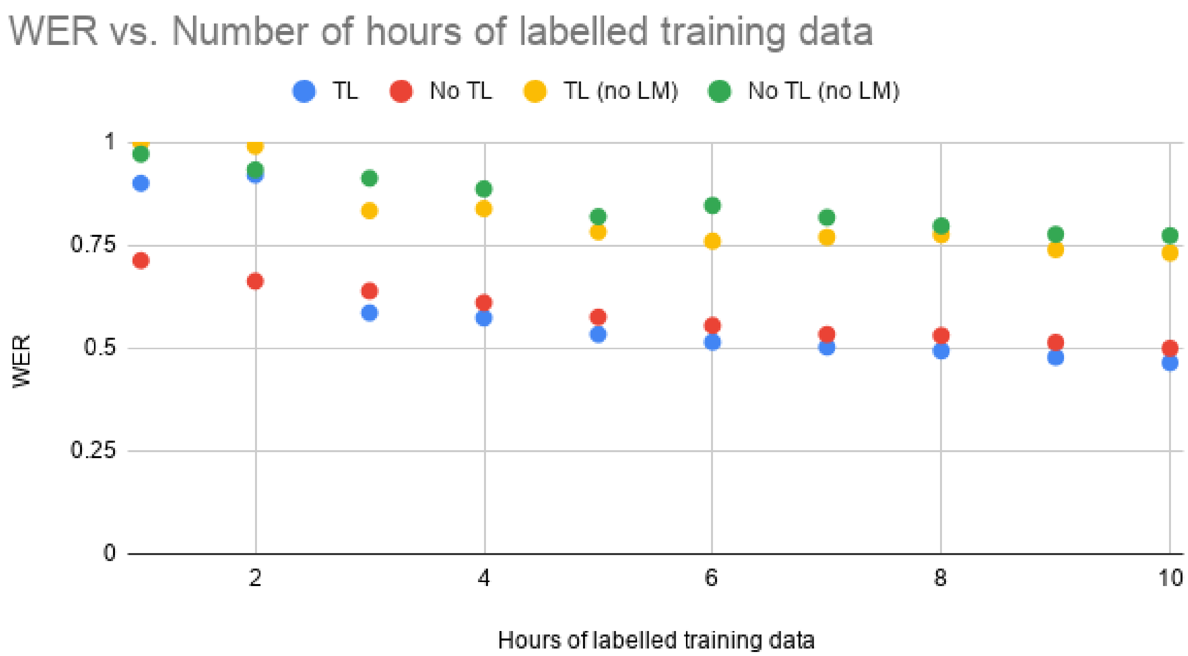

Figure 22: *In general, WER decreases with more training data*

As seen in the Figure 22, the performance of WIRENet improves with more training data, evidenced by the decreasing trend in WER for models that do not use a LM. Interestingly, with one and two hours of training data, the models with transfer learning perform worse than one trained directly on Seneca. This result could be due to the models not having enough Seneca data to overwrite the knowledge of English. With at least three hours of training data, the models with transfer learning outperform the models trained directly on Seneca. With more text data available, the effect of language models on the performance of the system also stabilizes.

To verify the effectiveness of WIRENet on low-resource language ASR, the WIRENet model using log-melspectrogram is trained on Iban dataset. Table 8 shows the results obtained on this dataset.

Table 8: *Iban WER and CER for various transfer learning and augmentation stategies using WIRENet with log mel-filterbanks as input features without (*NO LM*) and with (*W/LM*) a trigram language model.*

	WIRENET (NO LM)		WIRENet(W/LM)	
	WER	CER	WER	CER
No TL, no Aug	0.856	0.463	0.487	0.286
TL, no Aug	0.668	0.287	0.413	0.257
TL + Augment10	0.665	0.226	0.420	0.286
TL + Augment10 + finetune	0.518	0.160	**0.266**	**0.116**

Similar to the results for Seneca, the WIRENet model trained on Iban benefits from transfer learning, as well fine-tuning after data augmentation. Table 9 shows three Kaldi results on this same dataset: two standard HMM/GMM models (monophone and triphone) and one deep architecture, TDNN with LF-MMI. For Seneca, the WIRENet architecture substantially outperforms all three of these models, including the TDNN. For Iban, however, the TDNN model yields a lower error rate than the best WIRENet model. We suspect that the quality of the speech recordings and the content of the language might be contributing to these differences. Recall that the Iban data is professionally recorded and at least partially scripted broadcast news, while the Seneca recordings were captured by community members in casual settings with a wide array of equipment of varying quality. WIRENet could be more robust to the variability in recording quality found in the Seneca data. We also note that our model yields comparable WER error rates in both languages, which points to its superior ability to generalize to new datasets.

Table 9: *Seneca and Iban WER for Kaldi HMM-GMM models and TDNN with LF-MMI.*

Method	Seneca	Iban
Monophone	0.608	0.392
Triphone	0.524	0.329
TDNN LF-MMI	0.421	0.174

8.2 Audio spoofing detection

Both audio spoofing countermeasure models trained for 100 epochs. The best model of each architecture was determined by the prediction accuracy on the development set. The countermeasure score, used to indicate the likelihood of a sample being a bonafide speech, is computed by subtracting the score of the "spoof" class from the score of the "bonafide" class in the output vector.

Table 10 shows the EER and t-DCF obtained by the two proposed model on the development and evaluation dataset, along with other benchmarks.

Table 10: *Results of proposed countermeasures with other benchmarks and baseline methods*

	Development set		Evaluation set	
	t-DCF	EER%	t-DCF	EER%
Baseline 01	0.066	2.71	0.211	8.09
Baseline 02	0.012	0.43	0.236	9.57
CRNNSpoof	0.069	2.94	0.132	4.27
WIRESpoof	0.072	3.01	0.243	7.37
Benchmark A	0.007	0.32	0.179	7.66
Benchmark B	0.101	3.34	0.204	9.33
Benchmark C	0.002	0.11	0.274	9.68
Benchmark D	0.000	0.01	0.217	7.69
Benchmark E	0.000	0.00	0.157	6.02

Baseline 01 and Baseline 02 are GMM models trained on LFCC and CQCC input features, respectively. These baselines are given by the organizer of the ASVSpoof 2019 contest. Benchmark A is a CNN architecture proposed as countermeasure for replay attacks [12]. Benchmark B, C, and D are the same CNN architecture [3] trained using MFCC, power spectrogram, and constant Q cepstrum coefficients input features, respectively. Benchmark E is a fusion model using benchmarks B, C, and D.

While both models fail to perform better than the given baselines as well as all other benchmarks on the the development set, the CRNNSpoof model achieves significantly better result, both in t-DCF and EER, compared to other models on the evaluation set. The CRNNSpoof model outperform the best baseline model (Baseline 01) by 47% in terms of EER and 37% in terms of t-DCF. Meanwhile, the WIRESpoof model achieves better EER than other models, except for the CRNNSpoof model, on the evaluation set. However, the t-DCF score of the WIRESpoof model is only better than that of Benchmark C.

Comparing the result from Table 10 to the results published at Interspeech 2019 [60], the CRNNSpoof model achieves the 9th best result in terms of t-DCF, the primary metric of the challenge, and 5th best in terms of EER when evaluated on the evaluation set. It is notable that all models that perform better in terms of t-DCF as well as EER uses an ensemble of classifiers. When only single classifiers are considered, the proposed CRNNSpoof model achieves the best result in both t-DCF and EER. The next best model achieves t-DCF of 0.1404 and EER of 5.74%.

One reason for the better performance of the CRNNSpoof model on the evaluation set is due to the simplicity of the architecture compared to other architectures. Since the main difficulty of the challenge lies being able to detect unknown attacks, architectures that are too complex can overfit to the known attacks. With a simpler architecture and heavy regularization such as dropout and batch normalization, the architectures are forced to learn more generalizable features.

Similarly, the performance of WIRESpoof suggests that the model overfit the training data due to the complexity of the WideBlock. We believe that reducing the number of filters in the WideBlock, reducing the number of WideBlocks, or eliminating the bottleneck filters might allow the WIRESpoof model to generalize better.

9 Conclusion

The goal of this thesis is to develop a deep learning pipeline for low-resource automatic speech recognition and an effective, generalizable fake audio detection. For low-resource automatic speech recognition, we explored data augmentation techniques using simple signal distortion, such as speed and pitch modifications, as well as more complex methods, particularly voice conversion using generative adversarial networks. In addition, we proposes a three-step transfer learning pipeline for acoustic models: training on high resource language, then training on augmented data from low-resource language, and finetune on unaugmeneted data from low-resource language. We evaluated the different augmentation methods and transfer learning schemes using DeepSpeech, a popular recurrent-based acoustic model. The result indicates that transfer learning significantly enhances the performance of the acoustic model on low-resource dataset. Additionally, finetuning after training on augmented data, especially data generated via voice conversion techniques, also provides significant improvement on ASR performance.

We also explored the use of fully convolutional neural networks for low-resource ASR purposes. Convolutional neural network is faster to train compared to recurrent-based models, while still maintain the ability to capture temporal dependencies by using different kernel sizes. We explored a small version of Gated Convolutional Neural Network on the Seneca dataset, using the same transfer learning and data augmentation as the DeepSpeech model. In addition, we proposed a novel acoustic model architecture using blocks of parallel convolution operations with different kernel sizes, called WIRENet. The result shows that both convolutional models outperform DeepSpeech as well as triphone GMM/HMM models built using the Kaldi toolkit on Seneca, and produces competitive result compared to Kaldi models on Iban.

For audio spoofing detection, we experimented with two models: a convolution-recurrent neural network and a fully convolutional neural network. Each model is evaluated on the effectiveness of detecting spoofed audio from bonafide human speech as a standalone system using equal error rate. In addition, the models are also evaluated in conjunction with a given automatic speech verification system using tandem detection cost function. Comparing the performance of the proposed models against benchmark models shows that the convolution-recurrent model achieves the best single-system performance in the ASVSpoof 2019 contest in both equal error rate and tandem detection cost function. This model also achieves the 9th best overall result based on tandem detection cost function and 5th best result based on equal error rate.

10 Future works

As architectures using Wide Block have shown effectiveness in both low-resource ASR and audio spoofing detection, we plan to extend the use of Wide Block to other audio processing tasks such as noise classification, music genre classification, and sentiment analysis in speech or music. Additionally, we would also like to explore multi-task learning, where a single model is trained to perform multiple audio related tasks, such as speech recognition and speaker identity verification. Similarly, we will also explore the effectiveness of training a model on multiple languages at the same time.

While our initial experiments with Transformer models did not yield successful results, we plan to further explore different architecture using MHA. Similarly, we also would like to experiment with different algorithms of self-supervised learning to leverage the available unlabelled Seneca data.

While the research on audio spoofing detection has yielded great results, it is important to combine the visual, audio, and linguistic contents to identify whether a video has been manipulated or generated by a computer. Additionally, being able to generate adversarial samples that would overcome the proposed countermeasures would yield great knowledge about the possible attacks against our system.

11 References

References

[1] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. Mesonet: a compact facial video forgery detection network. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7. IEEE, 2018.

[2] W. Agenbag and T. Niesler. Automatic sub-word unit discovery and pronunciation lexicon induction for asr with application to under-resourced languages. *Computer Speech & Language*, 57:20–40, 2019.

[3] Moustafa Alzantot, Ziqi Wang, and Mani B Srivastava. Deep residual neural networks for audio spoofing detection. *arXiv preprint arXiv:1907.00501*, 2019.

[4] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, and G. Chen. Deep Speech 2: End-to-end speech recognition in English and Mandarin. In *ICML*, pages 173–182, 2016.

[5] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[6] K. Audhkhasi, B. Kingsbury, B. Ramabhadran, G. Saon, and M. Picheny. Building competitive direct acoustics-to-word models for english conversational speech recognition. In *ICASSP*, pages 4759–4763, 2018.

[7] J. Billa. Isi asr system for the low resource speech recognition challenge for indian languages. *Interspeech*, pages 3207–3211, 2018.

[8] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[9] A. Carmantini, P. Bell, and S. Renals. Untranscribed web audio for low resource speech recognition. *Interspeech*, pages 226–230, 2019.

[10] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP*, 2016.

[11] Ff Charpentier and M Stella. Diphone synthesis using an overlap-add technique for speech waveforms concatenation. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 2015–2018. IEEE, 1986.

[12] Bhusan Chettri, Saumitra Mishra, Bob L Sturm, and Emmanouil Benetos. Analysing the predictions of a cnn-based replay spoofing detection system. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 92–97. IEEE, 2018.

[13] Bhusan Chettri, Daniel Stoller, Veronica Morfi, Marco A Martínez Ramírez, Emmanouil Benetos, and Bob L Sturm. Ensemble models for spoofing detection in automatic speaker verification. *arXiv preprint arXiv:1904.04589*, 2019.

[14] C. Chiu, T. N Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. Weiss, K. Rao, E. Gonina, et al. State-of-the-art speech recognition with sequence-

to-sequence models. In *ICASSP*, 2018.

[15] Yunjey Choi, Min-Je Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2017.

[16] Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*, 2016.

[17] David Crystal. *Language death*. Ernst Klett Sprachen, 2000.

[18] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org, 2017.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[20] M. Gales, K. Knill, A. Ragni, and S. Rath. Speech recognition and keyword spotting for low-resource languages: Babel project research at CUED. In *SLTU*, pages 16–23, 2014.

[21] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks.

ArXiv, abs/1406.2661, 2014.

[22] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649, 2013.

[23] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[24] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE, 2013.

[25] Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[26] David Güera and Edward J Delp. Deepfake video detection using recurrent neural networks. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[27] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, et al. Deep Speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[28] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[29] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[30] Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197. Association for Computational Linguistics, 2011.

[31] G. Hinton, G. Dahl, A Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, B. Kingsbury, and T. Sainath. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29:82–97, November 2012.

[32] D. Imseng, P. Motlicek, H. Bourlard, and P. Garner. Using out-of-language data to improve an under-resourced speech recognizer. *Speech Communication*, 56:142–151, 2014.

[33] Robbie Jimerson, Kruthika Simha, Raymond W Ptucha, and Emily Prudhommeaux. Improving asr output for endangered language documentation. In *SLTU*, pages 187–191, 2018.

[34] Robert Jimerson and Emily Tucker Prud'hommeaux. Asr for documenting acutely under-resourced indigenous languages. In *LREC*, 2018.

[35] Sarah Samson Juan, Laurent Besacier, and Solange Rossato. Semi-supervised g2p bootstrapping and its application to asr for a very under-resourced language: Iban. In *Proceedings of Workshop for Spoken Language Technology for Under-resourced (SLTU)*, May 2014.

[36] Dan Jurafsky. *Speech & language processing.* Pearson Education India, 2000.

[37] Hirokazu Kameoka, Takuhiro Kaneko, Kou Tanaka, and Nobukatsu Hojo. Stargan-vc: Non-parallel many-to-many voice conversion using star generative adversarial networks. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 266–273. IEEE, 2018.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[39] Vitaliy Liptchinsky, Gabriel Synnaeve, and Ronan Collobert. Letter-based speech recognition with gated convnets. *arXiv preprint arXiv:1712.09444*, 2017.

[40] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[41] Jaime Lorenzo-Trueba, Junichi Yamagishi, Tomoki Toda, Daisuke Saito, Fernando Villavicencio, Tomi Kinnunen, and Zhenhua Ling. The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods. *arXiv preprint arXiv:1804.04262*, 2018.

[42] K. Malhotra, S. Bansal, and S. Ganapathy. Active learning methods for low resource end-to-end speech recognition. *Interspeech*, pages 2215–2219, 2019.

[43] V. Manohar, H. Hadian, D. Povey, and S. Khudanpur. Semi-supervised training of acoustic models using lattice-free mmi. In *ICASSP*, pages 4844–4848, 2018.

[44] Driss Matrouf, J-F Bonastre, and Corinne Fredouille. Effect of speech transformation on impostor acceptance. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE, 2006.

[45] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[46] Brian CJ Moore. *An introduction to the psychology of hearing*. Brill, 2012.

[47] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[48] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[49] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[50] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[51] A. Rosenberg, K. Audhkhasi, A. Sethy, B. Ramabhadran, and M. Picheny. End-to-end speech recognition and keyword search on low-resource languages. In *ICASSP*, pages 5280–5284, 2017.

[52] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

[53] Stanley S Stevens and John Volkmann. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, 53(3):329–353, 1940.

[54] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015.

[56] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[57] Massimiliano Todisco, Héctor Delgado, and Nicholas Evans. Constant q cepstral coefficients: A spoofing countermeasure for automatic speaker verification. *Computer Speech & Language*, 45:516–535, 2017.

[58] Massimiliano Todisco, Héctor Delgado, and Nicholas WD Evans. A new feature for automatic speaker verification anti-spoofing: Constant q cepstral coefficients. In *Odyssey*, volume 45, pages 283–290, 2016.

[59] Massimiliano Todisco, Xin Wang, Md Sahidullah, H ector Delgado, Andreas Nautsch, Junichi Yamagishi, Nicholas Evans, Tomi Kinnunen, and Kong Aik Lee. ASVspoof 2019: Future Horizons in Spoofed and Fake Audio Detection. In *Proc. of Interspeech 2019*, 2019.

[60] Massimiliano Todisco, Xin Wang, Ville Vestman, Md Sahidullah, Hector Delgado, Andreas Nautsch, Junichi Yamagishi, Nicholas Evans, Tomi Kinnunen, and Kong Aik Lee. Asvspoof 2019: Future horizons in spoofed and fake audio detection. *arXiv preprint arXiv:1904.05441*, 2019.

[61] Z. Tüske, P. Golik, D. Nolden, R. Schlüter, and H. Ney. Data augmentation, feature combination, and multilingual neural networks to improve asr and kws performance for low-resource languages. In *Interspeech*, 2014.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[63] M. Wiesner, A. Renduchintala, S. Watanabe, C. Liu, N. Dehak, and S. Khudanpur. Low resource multi-modal data augmentation for end-to-end ASR. In *Interspeech*, 2018.

[64] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017.

[65] Y. Zhang, W. Chan, and N. Jaitly. Very deep convolutional networks for end-to-end speech recognition. In *ICASSP*, pages 4845–4849, 2017.

[66] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.