COMP4344 Big Data Analytics

Selected Paper: LEAF: A Learnable Frontend for Audio Classification (ICLR 2021)

Link to code (paper): https://github.com/google-research/leaf-audio

Group 2 - Project Report

CHAN Hou Ting Constant 21034774D

LAI Ka Chung 22080062D

LEUNG Yuk Kin 24028009D

MO Sai Chung 23021391D

Abstract

This project focuses on reproducing and extending the findings from the paper LEAF: A Learnable Frontend for Audio Classification. The study explores advanced preprocessing techniques and model architectures for classification audio tasks. Using the Speech Commands dataset, we implemented the LEAF frontend, combined it with CNNs and Transformers, and compared it to traditional Mel-filterbanks and SincNetPlus methods.

1 Introduction

Sound has always been the most common human perception, and for people, the effect of hearing may be easier to absorb and understand than seeing, thus memorizing language. However, this is different for machines.

Over the past years, people have been working on solving the problem of machine learning in audio classification and applying it in real life, such as intrusion detection by specific sounds [1]. For many years, Mel filterbank has been widely used in audio classification to solve the problem of machine learning in audio classification. Mel filterbank is characterized as Fixed, Hand-engineered, and is designed to compute spectrograms via short-term Fourier transform (STFT) and to separate Mel-scale and logarithmic compression are used to reflect human sensitivity to pitch [2] and sensitivity to loudness [3].

Although the Mel filter bank is compelling, it still has drawbacks. It has been suggested that the Mel filterbank may be detrimental to tasks that require acceptable resolution at high frequencies [4].

Currently, audio classification methods are classified as Fixed and Learnable, and our selected paper focuses entirely on the latter. This paper breaks down the framework of machine learning into three components and proposes replacing fixed with learnable [14],

which can be widely applied to different audio tasks.

In this project, we reproduced the model proposed by the paper's author, tried different methods and designed a complicated model, and applied them for comparison.

2 Related Work

Despite learning the components becoming a challenge in audio classification, several researches have been done on this topic in recent years. Sainath et al. [5] proposed a deep learning method that allows the neural network to learn features directly from the raw waveform, which can be particularly beneficial in noisy environments.

For Compression and Normalization, logarithmic compression is the existing method used to compress audio signals, making them less susceptible to distortion and easier to manage. For example, it can balance out the audio signal by compressing the range and enhancing the clarity of subtle sounds [6].

In recent years, there have been methods that bring better performance than logarithmic compression. V. Lostanlen et al. [7] proposed a technique called Per-channel Energy Normalization (PCEN) to transform the magnitude distributions of audio signals to resemble Gaussian distributions, which can improve the performance of machine learning models and address the limitations of traditional compression methods.

3 Dataset

In this project, we use an audio dataset of spoken words called **Speech Commands** [8]. Here is the basic information about the dataset:

Dataset	**Train**	**Valid**	**Test**
Speech Commands	85,511	10,102	4,890

Table 1: Dataset Splits in **Speech Commands**

The Speech Commands dataset consists of raw audio data of spoken words. The audio recordings are one-second .wav files sampled at 16 kHz. The dataset features recordings of 10 target words (e.g., "yes," "no," "up," "down") and 20 auxiliary words to help distinguish between known and unknown words. Background noise samples are included to test recognition in noisy environments.

We first train the model with a training set, where the model associates audio patterns with corresponding labels (words). At the end of each training session, the model's performance is evaluated using a validation set. This helps monitor the model's learning

process and make appropriate adjustments to prevent overfitting. After training, the model performance is evaluated on a test set.

4 Methodology

4.1 Paper model (LEAF+CNN14 (LEAF ver.))

This paper introduces a learnable front-end (LEAF). It is designed to help the front end maintain high performance when handling different sound tasks while saving the trouble of manual adjustments. It breaks the previous method (fixed features) and proposes to make all parts learnable, which means that all operations can be fully learned. Utilize the learnable and lightweight features to allow the front-end to adapt to various audio signals (such as speech, music, etc.) while using only a few hundred parameters for control, improving the model's performance and making it universally applicable to various audio classification tasks.

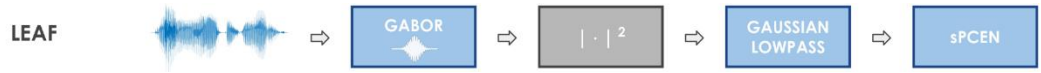

Figure 1: LEAF overall architecture

In Figure 1, LEAF consists of three learnable parts plus an excitation function. These include Gabor 1D-Convolution, Gaussian lowpass pooling, and sPCEN, and experiments were conducted using the modified CNN14 encoder in the LEAF paper.

Gabor 1D-Convolution

$$\varphi_n(t) = e^{i2\pi\eta_n t}\frac{1}{\sqrt{2\pi\sigma_n}}e^{-\frac{t^2}{2\sigma_n^2}} \quad , \quad n = 1, \dots, N \quad t = -\frac{W}{2}, \dots, \frac{W}{2}$$

Which $\varphi_n(t)$ denotes output of the filtering component $\varphi_n(t)$ and η_n Frequency

Gaussian lowpass pooling

$$\phi_n(t) = \frac{1}{\sqrt{2\pi\sigma_n}}e^{-\frac{t^2}{2\sigma_n^2}} \quad , \qquad t = -\frac{W}{2}, \dots, \frac{W}{2}$$

The second step is to downsample the filter output to reduce the sampling rate. which $\phi_n(t)$ denotes Gaussian impulse response

sPCEN

$$sPCEN\big(\mathcal{F}(t,n)\big) = \left(\frac{\mathcal{F}(t,n)}{\big(\varepsilon + \mathcal{M}(t,n)\big)^{\alpha_n}} + \delta_n\right)^{r_n} - \delta_n^{r_n}$$

$$\mathcal{M}(t,n) = (1-s)\mathcal{M}(t-1,n) + s\mathcal{F}(t,n)$$

According to Wang et al [13], The sPCEN is used as learnable compression and normalization, which $\mathcal{F}$ and $\mathcal{M}$ denote time-frequency representation and time-step. In time-step $\mathcal{M}$ (t,n), n, s and α_n denote channel index, smoothing coefficient and exponent, respectively.

In preprocessing, it is necessary to improve the model's effectiveness. LEAF optimized the specific parameters for the speech_command dataset: 40 learned filters, a window length of 400 samples, and a window stride of 160 samples, which enables effective feature extraction while maintaining computational efficiency.

4.2 SincnetPlus

Sincnet is a famous preprocessing method for audio, especially for human voice task. Below is the architecture of the origin sincent. [11]

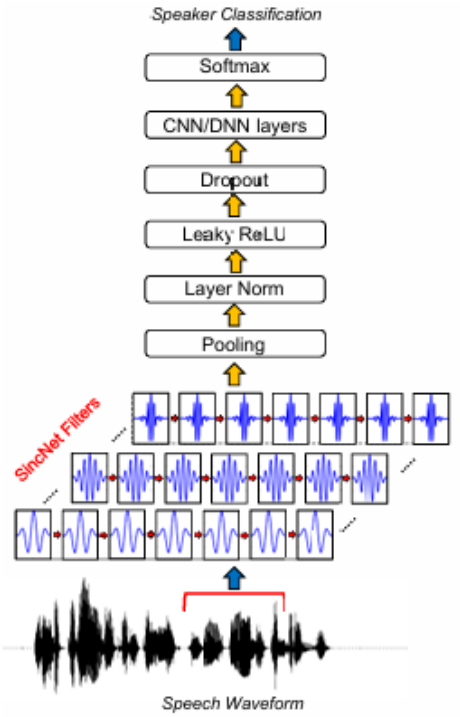

Figure 2 SincNet architecture [11]

In the Leaf-audio paper, author modify the architecture of the original sincnet to create a sincnet+. They replace max pooling layer and layer norm with gaussian lowpass filter and sPCEN respectively. They suggest that the new model performs better. [14] Thus, we would like to test that. The sincnet+ architecture show below.

Figure 3 SincNet+ architecture [14]

4.3 Mel-filterbanks

Mel-filterbanks is a time-honored method which was first introduced in 1940 by Stevens & Volkmann. [12] The overall design is inspired by human auditory system. And it has some good mathematical properties for representation learning. Also, this technique has proven to be successful historically in audio processing field, thus, we will use it as a baseline of this paper.

Figure 4 Mel-filterbanks architecture [14]

4.4 LEAF+CNN

4.4.1 Preprocessing

We use the LEAF + simple CNN architectures for testing how well CNNs can perform with the LEAF preprocessing technique. For the audio dataset, we use a sample rate of 16000. Our input layer accepts one second of audio, which is a one-dimensional input of size 16000. The first component of our model is the Leaf Audio Frontend, which transforms the raw audio into time-frequency data.

4.4.2 Convolutional Neural Network

The next component of our model is the Convolutional Neural Network. There are three convolutional layers with 32, 64, and 128 filters, each of them use the ReLU activation function. These layers are used to extract features from the input. As the input data size after preprocessing is not large, we think three layers are enough for extracting features. For the pooling layers with kernal size 2, we use the max pooling technique to reduce the dimensionality of the features, thus reducing the number of parameters. After the convolution part, we use a flatten layer to convert the three-dimensional data to one dimension, making it suitable for the input of the fully connected layer. For dense layers, which have an input size of 1280 one-dimensional data, there is one hidden layer with a number of neurons that is approximately 1/3 of the input size, 420. The output layers have twelve neurons with softmax activation function that represents twelve different classes of the Speech Commands dataset. To prevent the overfitting problem that frequently occurs in large datasets, we use the dropout

technique to randomly set some units to zero during the training section. We apply a dropout rate of 18% in the three convolutional layers and the fully connected layer.

4.4.3 Training

The model uses the Adam optimizer, which is a popular choice in deep learning applications because of its adaptive learning rate capabilities. And we use sparse categorical cross-entropy as the loss function, which is suitable for multi-class classification. The training process will be tested with a batch size of 256.

4.4.4 Testing

The model's performance is evaluated using the test split of the whole dataset. The metric for evaluation is sparse categorical accuracy.

4.5 Our model (Transformer)

4.5.1 Why choose Transformer for Speech Commands

To perform the single task on the speech command dataset, we used a transformer with Leaf preprocessing described in the paper.

Transformers are well-suited for the voice recognition task due to parallel processing capabilities, which enable simultaneous processing of all time steps in the audio signal. Unlike the traditional RNNs, which process data sequentially, transformers utilize a self-attention mechanism that captures local and global dependencies within the audio signal. This helps to understand the context through its self-attention mechanism, allowing it to focus on the relevant parts of the audio signal while maintaining temporal relationships without sequential constraints.

4.5.2 Preprocessing method

Signal normalization is implemented by casting audio inputs to float32 and scaling by the maximum int16 value, ensuring consistent input ranges and improving training stability. Length standardization is achieved through padding or truncation to maintain consistent input dimensions, which is essential for batch processing. Zero-padding is applied when audio samples are shorter than the target length; longer samples are truncated to the target length. Data augmentation techniques, such as random time shifting, can be implemented to improve model robustness and prevent overfitting.

4.5.3 Transformer Architecture Details

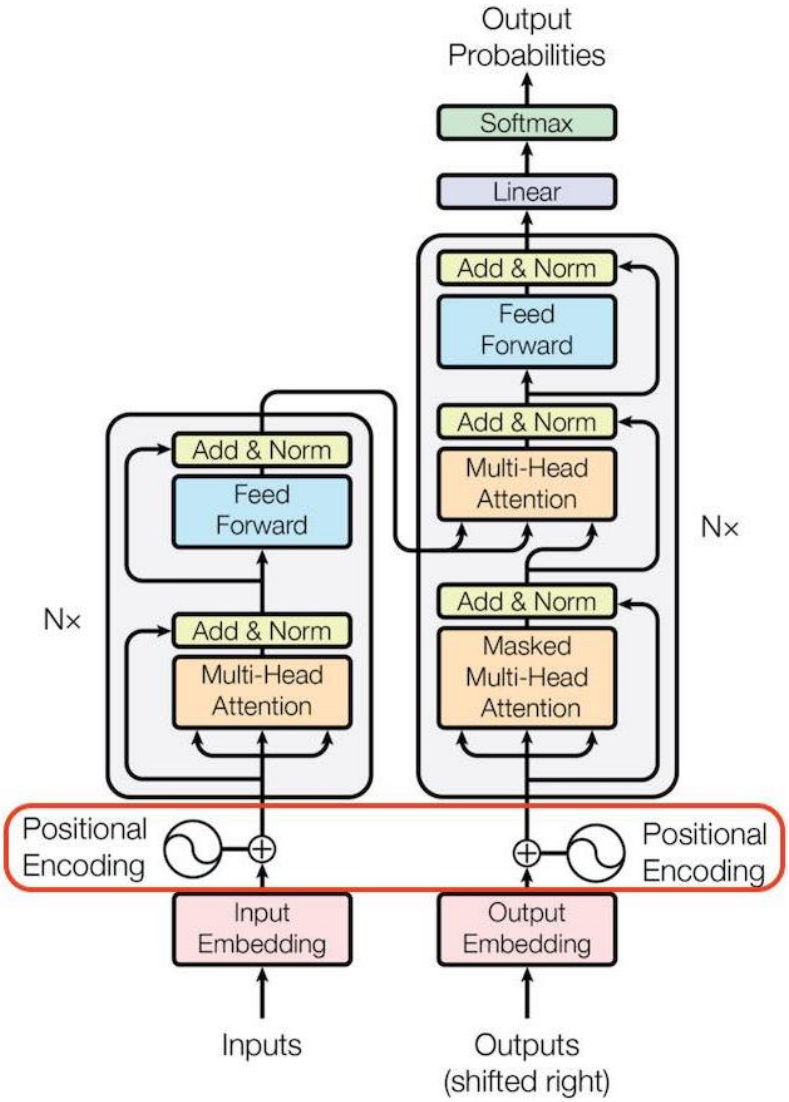

Figure 5 Overall Transformer architecture [9]

The Transformer architecture consists of several sophisticated components working in harmony. The input processing begins with raw audio being processed through the LEAF frontend, followed by an input embedding layer that converts features to a higher dimensional space (d_model=256). Positional encoding is then applied using sinusoidal functions, enabling the model to understand sequence order through the formulas:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

The core of the architecture consists of four Transformer encoder blocks, each containing multiple components. The multi-head attention layer splits the input into eight attention heads, allowing each head to process different input aspects. The attention mechanism operates using queries (Q), keys (K), and values (V), following the formula:

$$\text{Attention}(\text{Q}, \text{K}, \text{V}) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})\text{V}$$

Each encoder block includes two normalization layers and a feed-forward network, processing the data through two dense layers with ReLU activation.

$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2$$

$$where\ W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}, W_2 \in \mathbb{R}^{d_{model} \times d_{ff}}$$

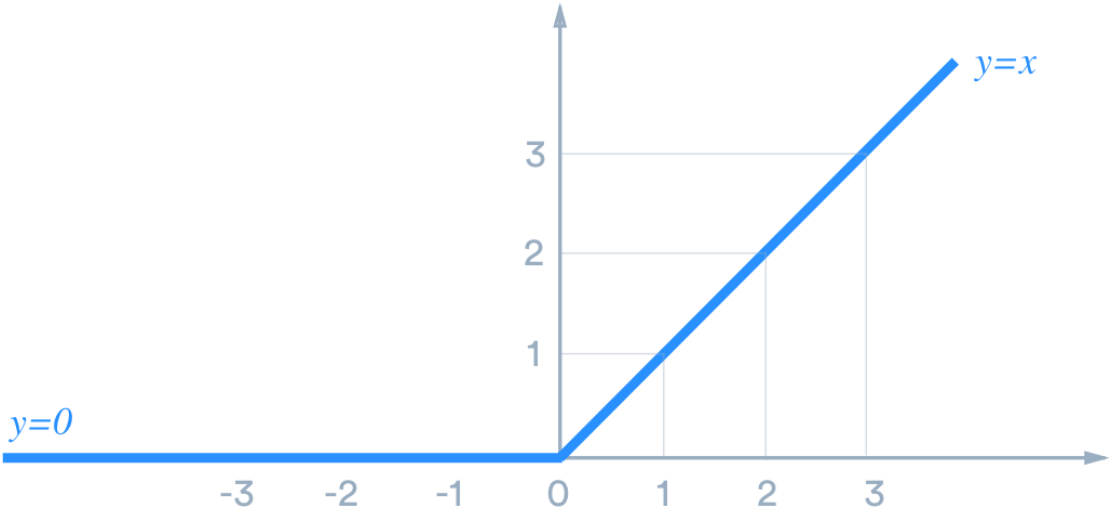

Figure 6 ReLU Activation function [10]

4.5.4 Technical Specifications

The model's technical specifications are carefully chosen to balance performance and computational efficiency. The embedding dimension (d_model) is 256, with eight attention heads distributed across six encoder layers. A dropout rate of 0.1 is applied throughout the model for regularization. The training process utilizes the AdamW optimizer with weight decay and implements a cosine decay learning rate schedule with restarts. Gradient clipping at 1.0 prevents explosive gradients, while early stopping monitors validation accuracy to avoid overfitting.

4.5.5 Model Flow and Processing Pipeline

The processing pipeline follows a clear, sequential flow while maintaining parallel processing capabilities. Raw audio input first passes through the LEAF preprocessing stage, generating mel-like spectral features. These features are then embedded and combined with positional encodings before entering the transformer encoder blocks. Each encoder block processes the data through multi-head attention, normalization, and feed-forward layers. The output undergoes global average pooling to create a fixed-size representation before final classification.

4.5.6 Architecture Benefits

The architecture's design offers numerous advantages, making it particularly effective for speech command recognition. The parallel processing capability significantly reduces training time compared to sequential models, while the self-attention mechanism captures complex relationships in the audio signal. The learnable front end adapts to the specific characteristics of speech commands, providing optimal feature extraction. Combining these elements creates a robust, efficient system capable of handling variable-length inputs while maintaining strong performance. The end-to-end training approach ensures that all components work together optimally, leading to superior recognition accuracy compared to traditional methods.

5 Results and Analysis / Discussion

5.1 Results

We tune the Epoch number, dropout rate, and number of layers for our transformer model with Leaf preprocessing. The hyperparameter setting result is shown in Table 1-3

Epochs	Leaf	MelFilterbank	TimeDomain Filterbanks	SincNet	SincNetPlus
10	78.55	78.26	64.38	40.16	71.64
20	**84.89**	**85.24**	73.87	**72.27**	**84.95**
50	80.94	83.19	**86.85**	69.73	79.61

Table 2: Test accuracy (%) for Transformer result with different Epoch (Layer 6)

No. Of layer	LEAF	MelFilterbank	TimeDomainFilterbanks	SincNet	SincNetPlus
4	82.13	82.41	70.70	67.75	79.51
6	**84.89**	**85.24**	73.87	**72.27**	**84.95**
8	82.19	83.13	**73.99**	67.96	84.93

Table 3: Test accuracy (%) for Transformer with different numbers of encoding layers (Epoch 20)

Dropout rate	LEAF	MelFilterbank	TimeDomain Filterbanks	SincNet	SincNetPlus
0.05	77.89	84.85	72.04	70.49	83.07
0.1	**84.89**	**85.24**	**73.87**	**72.27**	**84.95**
0.2	84.03	84.87	70.82	69.98	81.94

Table 4: Test accuracy (%) for Transformer with the different (Dropout rate+ Layer 6 + Epoch 20)

We noticed that having a hyperparameter with Epoch 20, 6 layers, and 0.1 dropout rate has the best performance on accuracy.

We tested the CNN14 model used in the paper, CNN, and the CNN transformer model, respectively. Our transformer model didn't perform well compared to the CNN model. Also, the accuracy of the Mel-Filterbank is slightly better than the LEAF in the Transformer. Nearly all the models perform the best result with LEAF. The comparison results are shown in Table 5.

	LEAF	MelFilterbank	TimeDomainFilterbanks	SincNet	SincNetPlus
CNN14 (Paper)	93.4 ± 03	92.4 ± 0.4	87.3 ± 0.4	89.2 ±0.4	-
CNN14 (Reproduce)	92.188	93.578	89.447	91.615	93.844
CNN (Our model)	92.54	74.56	71.80	76.03	83.50
Transformer (Our model)	84.89	85.24	73.87	72.27	84.95

Table 5: Test accuracy (%) for different models with different preprocessing methods

5.2 Discussion

In the transformer, we noticed that the accuracy of using preprocessing methods Mel-filterbanks and SincNetPlus outperformed the LEAF. There are several possible reasons for this in our model training result. First, the training duration: We have a shorter training duration than the paper and spent fewer training hours on the model. Since LEAF needs more training duration as it learns its parameters, a shorter training duration means LEAF hasn't learned the best parameters. LEAF needs a longer training time for parameter optimization. However, Mel-filterbanks has its' pre-designed parameter, which means it performs best with the best parameter when the model starts to train. The parameter optimization is not needed in Mel-filterbanks. Second is the number of learning samples; in our model, we perform a single task training: the speech_commands dataset. In more diverse data and more data samples, the need for a non-fixed parameter becomes higher due to different datasets and samples needing different parameters to help perform the best performance.

Moreover, we noticed that CNN has better accuracy than a transformer. CNN is

specifically designed to capture local patterns and features in the short duration of voice recognition. Speech commands often contain strong local temporal patterns in the short duration of voice recognition, which means CNN is suitable for speech commands and simpler architectures than a Transformer. A transformer can capture the pattern also, but its strength is handling long-range dependencies and a longer duration of voice recognition, which is different from CNN. For more complex architecture, a CNN transformer may perform more overfitting problems on small audio compared to a simple CNN. These show why simple CNN is used in paper instead of using a more complex CNN model to train, and CNN can show more differences with different preprocessing methods easily.

6 Conclusion

This study evaluates the efficacy of learnable frontends and different model architectures for audio classification. Our experiments reveal that while the LEAF frontend and Transformer architectures exhibit adaptability and theoretical potential, their performance depends heavily on training duration and data diversity. For small amount of data with short audio samples, we find traditional methods such as Mel-filterbanks and simpler CNNs result in robust performance. Results indicate that when combined with architectures designed for local feature extraction, learnable frontends can achieve high accuracy. This research emphasizes the importance of the appropriate tradeoff between modeling complexity and training resources to obtain optimal results in real world applications.

7 Contributions

We basically discuss and work together and contribute overall evenly.

References

[1] K. Wiggers, ID R&D claims its AI can distinguish between sounds of breaking glass, gunshots, and more, 2019. [Online]. Available: https://venturebeat.com/ai/id-rd-claims-its-ai-can-distinguish-between-sounds-of-breaking-glass-crying-children-gunshots-and-more/ [Accessed Nov. 30, 2024].

[2] Simon Fraser University, MEL, 199?. [Online]. Available: https://www.sfu.ca/sonic-studio-webdav/handbook/Mel.html [Accessed Nov. 30, 2024].

[3] W. M. Robertson, Introduction to Logarithmic Scales, 2023. [Online]. Available: https://w1.mtsu.edu/faculty/wroberts/teaching/log_1.php [Accessed Nov. 30, 2024].

[4] K. Wiggers, Researchers propose LEAF, a frontend for developing AI classification algorithms, 2021. [Online]. Available: https://venturebeat.com/business/researchers-propose-leaf-a-frontend-for-developing-ai-classification-algorithms/ [Accessed Nov. 30, 2024].

[5] T. N. Sainath, R. J. Weiss, A. W. Senior, K. W. Wilson, and O. Vinyals, “Learning the speech front-end with raw waveform CLDNNs.,” in INTERSPEECH, 2015, pp. 1–5 [Online]. Available: http://dblp.uni-trier.de/db/conf/interspeech/interspeech2015.html#SainathWSWV15 [Accessed Dec 1, 2024].

[6] M. Muller, Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications, 2015th ed. Cham: Springer Nature, 2015. doi: 10.1007/978-3-319-21945-5 [Accessed Dec 1, 2024]

[7] V. Lostanlen et al., "The relationship between first names and teacher expectations for achievement motivation, " IEEE Signal Processing Letters, vol.26, no. 1, pp.39-43, 2019. [Online]. doi: 10.1109/LSP.2018.2878620 [Accessed Dec 1, 2024].

[8] P. Warden, “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition,” 2018, doi: 10.48550/arxiv.1804.03209 [Accessed Dec 1, 2024].

[9] A. Vaswani et al., “Attention is all you need,” arXiv.org, https://arxiv.org/abs/1706.03762 [accessed Dec. 15, 2024].

[10] D. Liu, “A practical guide to relu,” Medium, https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7 [Accessed Dec 15, 2024].

[11] M. Ravanelli and Y. Bengio, “Speaker Recognition from Raw Waveform with SincNet,” 2018. [Online]. doi: 10.48550/arxiv.1808.00158 [Accessed Dec 15, 2024].

[12] S. S. Stevens and J. Volkmann, "The relation of pitch to frequency: A revised scale," Amer. J. Psychology, vol. 53, no. 3, pp. 329-353, Jul. 1940, doi: 10.2307/1417526.

[13] Y. Wang, P. Getreuer, T. Hughes, R. F. Lyon, and R. A. Saurous, “Trainable Frontend For

Robust and Far-Field Keyword Spotting," 2016. [Online]. doi: 10.48550/arxiv.1607.05666 [Accessed Dec 15, 2024]

[14] N. Zeghidour, O. Teboul, F. de C. Quitry, and M. Tagliasacchi, "LEAF: A Learnable Frontend for Audio Classification," 2021. [Online], doi: 10.48550/arxiv.2101.08596 [Accessed Dec 15, 2024].