SMU Data Science Review

Volume 3 | Number 2 Article 3

2020

Spoken Language Recognition on Open-Source Datasets

Brady Arendale
Southern Methodist University, barendale@smu.edu

Samira Zarandioon
Southern Methodist University, szarandioon@smu.edu

Ryan Goodwin
Southern Methodist University, rgoodwin@smu.edu

Douglas Reynolds
MIT Lincoln Laboratory, dar@ll.mit.edu

Spoken Language Recognition on Open-Source Datasets

Brady Arendale[1], Samira Zarandioon[1], Ryan Goodwin[1], and Douglas Reynolds[2]

[1] Master of Science in Data Science, Southern Methodist University, Dallas TX 75275 USA {barendale,szarandioon,rgoodwin}@smu.edu
[2] MIT Lincoln Laboratory, Lexington MA USA dar@ll.mit.edu

Abstract. The field of speaker and language recognition is constantly being researched and developed, but much of this research is done on private or expensive datasets, making the field more inaccessible than many other areas of machine learning. In addition, many papers make performance claims without comparing their models to other recent research. With the recent development of public multilingual speech corpora such as Mozilla's Common Voice as well as several single-language corpora, we now have the resources to attempt to address both of these problems. We construct an eight-language dataset from Common Voice and a Google Bengali corpus as well as a five-language holdout test set from Audio Lingua. We then compare one filterbank-based model and two waveform-based models found in recent literature, all based on convolutional neural networks. We find that our filterbank-based model achieves the strongest results, with a 90.5% test accuracy on our eight-language test set and a 74.8% test accuracy on our five-language Audio Lingua test set. We conclude that some models originally trained on private datasets are also applicable to our public datasets and make suggestions on how this performance can be improved further.

1 Introduction

Language comprehension is a major frontier towards achieving general artificial intelligence. New techniques are being developed constantly for both text and spoken language comprehension. There are many services commonly available that are capable of both identifying and translating written language from text or image inputs. There are also products such as Google Translate that have the capability to translate speech. However, this generally requires knowing what language one wants to translate. Even applications that can identify a language being spoken generally have only a few of the most spoken languages. A model that could accurately identify many languages would be extremely useful. For example, in call centers language routing is typically done by presenting options that require the caller to press a number. With a spoken language recognition model, the caller would only need to say a few words and would be routed automatically to an operator that speaks their language.

The field of language recognition is somewhat limited compared to speech and speaker recognition. Popular corpora for speaker recognition are typically mostly or only in English. Many datasets that do exist are not freely available, such as those used in the National Institute of Standards and Technology's Language Recognition Evaluation series and those from the Language Data Consortium. However, in recent years there has been an increase in the number of open-source datasets available, including Common Voice from Mozilla. This surge in available data now makes it possible to develop language recognition models without having access to private or expensive datasets.

These developments motivated us to develop a model that could identify a large number of languages from spoken audio only using open-source datasets. Unlike previous works that generally used data from one or two sources, we gathered large amounts of audio data from a few different sources and compiled them together into one large dataset. This gave us a broader range of speech for our model to learn the nuances of each language. We then transformed the audio data into a usable form for deep learning by extracting raw waveforms and filterbanks, which we will explain later. We built several deep learning models based on recent research that operate on either raw waveforms or filterbanks to classify the language spoken.

In the next section, we will examine common techniques used in spoken language recognition projects. We will also explore previous projects, what techniques they used, and what results they achieved. We will then explain the methods used for collecting, transforming, and modeling the data. Next, we will look at our process for data transformation and modeling. Then, we will examine the results and see how our models performed. Lastly, we will conclude how the project went and talk about challenges faced and possibilities for future analysis.

2 Overview of Common Techniques

2.1 Approaches

There are several approaches to spoken language recognition, and projects are typically categorized by the approach they take. The common approaches are acoustic-phonetics, phonotactics, prosodic, and lexical. These approaches attempt to extract features from speech data at varying levels of abstraction. In linguistics, distinct sound events are called phones, and the smallest units that distinguish meaning of one sound from another are called phonemes. The acoustic-phonetic approach is concerned with individual phones and phonemes. Different languages have different sets of phonemes, and the phonemes are used in different frequencies. The phonotactic approach considers phonotactic constraints, which dictate what phonemes can be used together in a language [7].

The prosodic approach analyzes suprasegmental features such as stress, duration, rhythm, and intonation. For example, many linguists divide languages into three rhythmic classes: stress-timed, syllable-timed, and mora-timed, although this classification is disputed [1]. However, prosodic features have been shown to be less effective and more challenging to extract than other approaches. The lexical approach consists of analyzing each language's unique phonology and syntax for identification. One option is to run each utterance, or period of speech separated by silence, into a semantic understanding model for each language, and choose the one that gives the highest likelihood. However, this is essentially equivalent to learning a whole language to identify it. This is much more complicated and costly than other approaches, so it is not generally used. The acoustic-phonetic and phonotactic approaches have historically delivered the best results with the most efficiency and are the most commonly used. We will examine the methodologies used for each approach [7].

2.2 Phonotactic Approach

The phonotactic approach is used because we believe that languages can be characterized by their lexical-phonographical constraints. That is, we expect certain sequences of phones to be more or less common, if present at all, in different languages, and we attempt to classify languages based on these sequences. One of the challenges with this approach is isolating and identifying phones. This is done by a speech tokenizer, which converts an utterance into a sequence of sound tokens, which may be sound frames, phones, syllables, or words. The goal of a tokenizer is to provide the most accurate phonotactic representation possible, greatly influencing the effectiveness of the model itself. One tokenization technique uses Gaussian mixture models (GMMs) at the frame level. GMM tokenization converts speech frames into a sequence Gaussian labels. The drawback to this technique is that the short timeframes fail to capture enough information to be useful. Another tokenization approach is the attribute-based approach, which tokenizes speech into articulatory attributes. These can be made to have a set of attributes common to all languages to build a universal attribute recognizer (UAR). A third technique called phone tokenization uses phone units. These are typically modelled using a hidden Markov model (HMM). Phone tokenizers, or phone recognizers, are often used in state-of-the-art systems due to the tradeoff between development cost and effectiveness. In practice, phone recognizers can be used to model languages that they have not been trained on [7].

The resulting tokenized speech is used as the input to a model. One such model is a phone n-gram model. This model is trained on one of N target languages for a total of N models. Each model measures the probability of sequences of phones occurring in that language. The output of a phone tokenizer is inputted into each model, returning N likelihood values. We can fuse these likelihood values into one or use them as inputs into another model. We can optionally use multiple phone recognizers and train N phone n-gram models for each. This process is called phone recognition followed by language modeling (PRLM), and the process using multiple phone recognizers is called parallel phone recognition followed by language modeling (PPR-LM).

A further modeling option is to use vector space modeling (VSM) to map the output of a phone n-gram model into a high-dimensional vector. One such technique is the bag-of-sounds framework, analogous to the bag-of-words paradigm in information retrieval and document classification. The bag-of-words maps a document to a vector containing word counts over a known vocabulary, and the bag-of-sounds is much the same but with phones or phone n-gram statistics. We can take the vectorized output and treat it as a vector-based classification problem. Support vector machines (SVMs) are very powerful and efficient for this purpose. We train N one-versus-the-rest SVM models and generate N output scores. Again, we can take these outputs as-is and use the maximum likelihood as our decision, or we can input these into another classification model.

2.3 Acoustic-Phonetic Approach

The acoustic-phonetic approach is based on the idea that languages fundamentally differ in terms of the phones used and the frequency that each phone is used. These approaches attempt to model the acoustic-phonetic distribution of a language using the acoustic features. Instead of identifying and labeling phones as in the previous approach, the acoustic-phonetic approach attempts to extract features directly from the audio. The Mel-frequency cepstral coefficients (MFCCs) are one of the most popular feature extraction techniques for audio data. At a high level, the cepstrum extracts the characteristic features of the audio waveform, and it is transformed to the mel scale, which represents the range of human hearing. MFCCs are computed at short time intervals together with their first and second derivatives. Another popular alternative is the shifted-delta-cepstral (SDC) coefficients. They contain information about not only the time frame they are calculated for but also for a number of adjacent time frames [7].

A common model used after feature extraction is the universal-background-model-based GMM (GMM-UBM), popular in speaker recognition. A universal background model (UBM) is first trained on the entire dataset containing all languages. The UBM represents the general characteristics of speech among all languages in the data. Separate GMMs for each language are then trained using the maximum a posteriori (MAP) technique, with the UBM as the prior distribution. The GMM is used for its ability to approximate arbitrary distributions and to model the underlying components with individual Gaussian distributions. We can approximate the acoustic-phonetic distributions of a spoken language using this method. However, GMMs are susceptible to acoustic variability caused by non-language characteristics such as speaker and noise.

As with the phonotactic approach, we can use vector space modeling to map spectral features of an utterance into a high-dimensional vector. We can again use SVM methods to classify these vectors.

2.4 Deep Learning Approaches

Deep learning has emerged in the past several years as an approach to automatic speech regonition, most commonly using the acoustic-phonetic approach. Neural networks have been used for spoken language recognition, isolated word recognition, and audiovisual speech recognition. Neural networks make small assumptions about the statistical qualities of a model for spoken language identification. Language recognition has also benefitted from technological breakthroughs in related areas, such as signal processing, pattern recognition, cognitive science, and machine learning [13].

Initially, feed-forward deep neural networks (DNNs) were utilized, but only model by working with a limited and fixed size sliding window of acoustic frames. This precludes the models from utilizing speech dependencies on other parts of the speech input. To address these concerns, recurrent neural networks (RNNs) are used. RNNs utilize looped connections that allow for interactions between different parts of speech by allowing previous inputs or time steps to influence calculations for subsequent time steps by retaining information within internal states.

Nevertheless, RNNs still have issues with retaining memory, which can influence time steps farther away from each other. The next evolution is to include a mechanism that retains memory to influence multiple parts of a speech vector using LSTMs (Long Short-Term Memory). LSTMs are a special class of RNNs that include special memory blocks within each hidden recurrent layer. Additional improvements to LSTMs include BLSTMs (Bidirectional LSTMs) which operate by using input from both directions of a speech excerpt to calculate decisions on a current input.

One of the major advances in recent years is the development of x-vectors as a new way of extracting embeddings of utterances. A deep neural network is trained using extracted audio features as inputs and language as the target. Outputs are extracted from one of the last hidden layers and used as embeddings [14]. X-vectors extracted in this way have been shown to perform better than i-vectors extracted from GMM-UBM models in both language recognition and speaker recognition tasks [16].

Another advantage to the shift to deep learning frameworks is the types of feature extraction available. In addition to MFCCs, deep learning approaches can also use filterbanks [16] and bottleneck features [14]. Filterbanks are calculated during the derivation of MFCCs, but coefficients are highly correlated, which poses a problem for many earlier models. To account for this, the discrete cosine transformation (DCT) is applied, resulting in uncorrelated features. However, deep learning models are able to handle the correlated coefficients from filterbanks, resulting in potential gains by minimizing loss of information from the DCT[3]. Bottleneck features (BNFs), unlike MFCCs and filterbanks, are extracted from neural networks. A deep neural network is created with one “narrow” layer with relatively few neurons in between several “wide” layers with many more neurons. This creates a “bottleneck” where the information needed to predict the target from the input is assumed to be compressed. We can then take the weights

[3] https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html. Last accessed May 22, 2020.

from the bottleneck layer and use them as features in an x-vector network or other model [9].

3 Mel-Frequency Cepstral Coefficients and Filterbanks

Audio data presents many unique challenges. Such challenges include background noise, accents, variation in tone and potentially limitless vocabulary that can be spoken. To mitigate these obstacles, a variety of techniques have been researched and must be used to get the data in a format that can be analyzed. Each step of the speech recognition system must deal with its respective challenges before the next step can be approached. Because of this, the entire process as a whole must be broken down into steps, each step must have its challenges laid out, and each challenge must have a solution in place to allow for the ability to move to the next step in the spoken language classification process.

The most primitive challenge is first getting the recorded audio in a format that can be interpreted by a computer for analysis. This is known as analog to digital conversion and can generally done by any modern computer automatically when inputting audio signal into an instrument such as a microphone that is directly connected with the computer.

After getting the data in a computer-usable format, background noise is the first challenge faced when dealing with spoken language data. Before the speech in the audio can be interpreted, the background noise must be filtered out allowing a model to focus solely on the intended sound. Although microphones and other speech recording systems are constantly being upgraded to reduce background noise and pick up only intended audio, background noise continues to be present in almost every scenario. The process of extracting actual speech from background noise is known as voice activity detection (VAD). This is a necessary step in any speech recognition system as the background noise can heavily affect a model's interpretation of audio data. Most VAD methods exist for the primary reason of feature extraction [3]. Numerous methods of VAD have been researched, and our analysis works with a method known as Mel-Frequency Cepstral Coefficients, or MFCC. The idea behind MFCCs is to break the audio down into very short intervals that can be considered stationary. Figure 1 below [8] simulates a high-level overview of MFCC generation.

Audio signals are constantly changing, and it is very difficult to evaluate data that is non-stationary, therefore we need to break the audio down into timeframes in which we can view the data as statistically stationary. This timeframe is generally determined to be within 5-100 milliseconds when working with audio data. After about 0.2 seconds, signals are non-stationary and begin to reflect more holistic speech sounds that can be too complex for current machine learning techniques [5].

MFCCs gained popularity in the early days of speech and language recognition because they were easy to calculate and uncorrelated, making them ideal for the popular models in use at the time, including GMMs and HMMs. With recent advances in deep learning, however, there is not as much of a need for un-

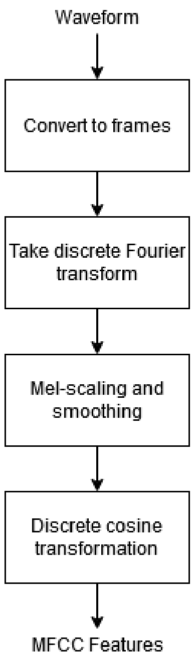

Waveform
Convert to frames
Take discrete Fourier transform
Mel-scaling and smoothing
Discrete cosine transformation
MFCC Features

Fig. 1. MFCC feature generation

correlated features. During the calculation of MFCCs, we calculate filterbanks, which are highly correlated representations of the audio data. We then apply the discrete cosine transformation (DCT) to decorrelate the filterbank features before mean normalizing the data, which results in our MFCCs. Since we now have models that are able to make use of correlated features, we can exclude the DCT and mean normalize the filterbanks directly. Since the DCT is a linear transformation, it can remove some of the nonlinearities in the audio data, which can cause loss of information. This can lead to both performance benefits and faster extraction of audio features since filterbanks take one less step to compute. For these reasons, filterbanks have increased in popularity as audio features with the domination of deep learning models in the fields of speech and language recognition. An example illustration of waveforms and filterbanks is shown in Figure 2.

4 X-Vectors

X-vectors were developed for speaker recognition with the goal of producing embeddings that generalize to speakers not seen in the training data [15]. X-vector embeddings are extracted from a deep neural network trained on the extracted audio features. The neural network is made up of time-delay neural

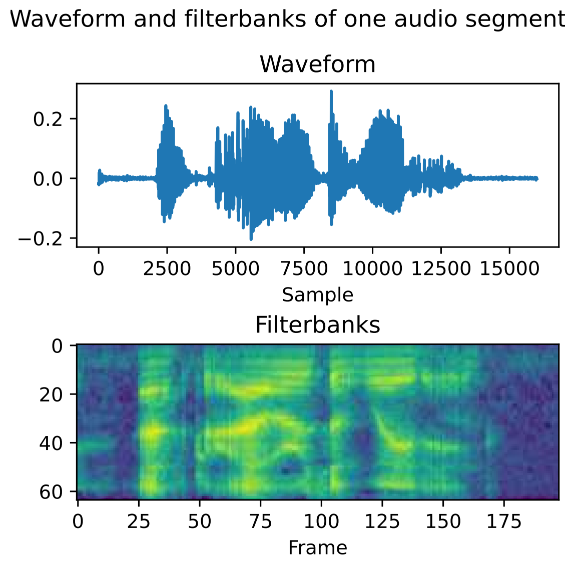


Fig. 2. Waveform and filterbanks for the same 2-second audio segment at 8000 Hz

network layers, also known as 1D convolutional layers. The original architecture is shown in Table 1 [14].

Table 1. Architecture of DNN for extracting x-vectors

Layer	Layer context	Total context	Weight dimensions
frame1	$[t-2, t+2]$	5	$5F \times 512$
frame2	$\{t-2, t, t+2\}$	9	1536×512
frame3	$\{t-3, t, t+3\}$	15	1536×512
frame4	$\{t\}$	15	512×512
frame5	$\{t\}$	15	512×1500
stats pooling	$[0, T)$	T	$1500T \times 3000$
segment6	$\{0\}$	T	3000×512
segment7	$\{0\}$	T	512×512
softmax	$\{0\}$	T	$512 \times L$

The input to the DNN is a segment consisting of T speech frames. Each speech frame t has F features, which can be MFCCs, filterbanks, or BNFs [14, 16]. The first five layers operate on the frame level, and then they are pooled and operated on as a whole segment. Each speech frame t is input into the first layer along with the previous two frames and next two frames, for a context of five frames for each t. The total number of features inputted per frame is thus $5F$. The first layer expands the input into 512 dimensions. This is equivalent to a 1D convolutional layer with a kernel size of 5 and 512 convolutional filters. A ReLU activation is applied, and this is also the activation function for each frame and segment layer. The frame1 layer outputs for $t-2$, t, and $t+2$ are spliced together to form a $512 \times 3 = 1536$-dimensional vector with a total context of nine frames, then reduced to 512 dimensions. This is equivalent to a 1D convolutional layer with a kernel size of 3, a dilation of 2, and 512 convolutional filters. A similar process is used for the frame3 layer, taking the $t-3$, t, and $t+3$ outputs from frame2, for a total context of fifteen frames. The frame4 layer adds no additional context. The frame5 layer expands the output into a 1500-dimensional vector.

Next, a statistical pooling layer takes each 1500-dimensional vector from all T speech frames and calculates the mean and standard deviation of each dimension, resulting in a 3000-dimensional output representing the entire segment. The segment6 layer takes this vector and reduces it to a 512-dimensional linear combination, representing the full context T. Another segment layer calculates nonlinearities in the data, which are finally used to predict one of L languages after a softmax activation.

The outputs of either segment6 or segment7 can be extracted pre-activation and used as embeddings, but segment6 embeddings were found to be preferable and came to be known as x-vectors [15, 16]. X-vector-based classification models have been found to outperform state-of-the-art i-vector models in a wide variety of applications [11].

Once the network is trained, languages are "enrolled". This is done by extracting x-vectors for each utterance in a given language and averaging all vectors together to form the language model vector. Dimensionality of vectors is then typically reduced using LDA for better performance. Vectors are standardized with zero mean and unit variance. They are then length normalized. Finally, x-vectors for each test sample are calculated and compared with the language model vectors using cosine similarity. The most similar language is chosen as the classification.

X-vectors can be augmented with various types of background noise, such as babble, music, noise, and reverberation, to artificially increase the amount and diversity of training data [16]. This has been shown to significantly improve performance in certain tasks including speaker recognition and language recognition [11, 14].

5 Methodology

5.1 Data Collection and Preparation

We collected data from a variety of open-source speech corpora. We used seven languages from Common Voice[4] and a crowd-sourced Bengali speech corpus from researchers at Google [6] and available at OpenSLR[5]. Table 2 shows the

[4] Data available at https://voice.mozilla.org/, accessed 28 May 2020.
[5] Data available at http://www.openslr.org/53/, accessed 30 May 2020.

languages used and the number of hours of recordings per language. Languages were chosen mainly for the amount of data available. We were unable to find large amounts of data for certain widely spoken languages, such as Arabic or Hindi. We had much more data for English, French, and German than we actually used, but wanted to keep a reasonable class balance and were constrained by other languages, such as Spanish.

Table 2. Languages used (Common Voice/Google Bengali dataset)

Language	Hours of recordings
Bengali	144.4
Catalan	144.4
English	135.9
French	135.2
German	136.8
Kabyle	122.8
Persian (Farsi)	133.1
Spanish	117.0

To test our models' generalization onto new data, we also used audio from Audio Lingua[6] as a holdout test set. Audio Lingua submissions are intended to be listened to by human speakers for the purpose of learning languages, rather than for use in speech and language recognition projects. This may result in Audio Lingua clips having different audio characteristics than those in the Common Voice or Google Bengali corpora, making this data ideal for testing generalization. They are also much longer on average. We used the five languages in common with our Common Voice/Google Bengali dataset. In total, our Audio Lingua test set consists of 2.3 hours of Catalan, 30.9 hours of English, 16.9 hours of French, 21.4 hours of German, and 11.4 hours of Spanish. Due to class imbalance, we will examine per-class recall to get a clearer picture of how our models performed on this test set.

Audio files were manually downloaded from Common Voice and OpenSLR. We then used the LibROSA[7] package for Python to extract raw waveforms. We resampled each audio file to 8000 Hz for consistency. We then split each waveform into 2-second (16000-sample) segments. The remainder, after splitting the audio waveform into as many 2-second segments as possible, was taken equally from the beginning and end of the waveform. We also extracted additional durations including 5 seconds, 10 seconds, and 30 seconds chosen at random, but we found that some languages had much more representation in the longer audio segments than others, so we only used the 2-second segments. Table 2 reflects only the

[6] https://www.audio-lingua.eu/, accessed 28 May 2020.
[7] https://librosa.org/librosa/

durations of the 2-second segments used. A side effect of segmenting the audio in this way is that it also removed some of the silence from the beginning and end of each audio file, which could be helpful since we did not use voice activity detection to remove silence.

After segmenting the audio into equal-sized chunks, we used torchaudio's Kaldi-compliant implementation of filterbank extraction to extract 64 filterbanks for each frame length of 25 milliseconds. We then mean-normalized the filterbanks by subtracting the filterbank's overall mean from each element. We labelled each set of waveforms and filterbanks with the language spoken and the speaker (if available).

We randomly assigned sets of waveforms and filterbanks to training, validation, and test sets with probabilities of 70% for training and 15% each for validation and test sets. We ensured that waveforms and filterbanks belonging to the same speaker were all in the same set to prevent data leakage. However, not all speakers were identified, so there is still a possibility that the same unidentified speaker could have ended up in two different sets. The amount of hours of recordings for each language in each split for both the Common Voice/Google Bengali (CV/GB) dataset and the Audio Lingua (AL) dataset can be seen in Table 3.

Table 3. Hours of recordings for each language by split

Language	CV/GB training	CV/GB validation	CV/GB test	AL test
Bengali	101.3	22.0	21.1	—
Catalan	94.9	22.5	27.0	2.3
English	97.0	19.7	19.2	30.9
French	102.0	17.5	15.6	16.9
German	91.2	22.5	23.1	21.4
Kabyle	85.6	18.9	18.3	—
Persian (Farsi)	104.9	11.8	16.3	—
Spanish	85.3	14.2	17.4	11.4

5.2 Modeling

A number of effective speaker recognition and language recognition models have been developed in recent years. We decided to compare a few different models to determine which one worked best on our open-source data. We trained two models that employ raw waveforms: one based on the M5 architecture by Dai et al. [2] and one based on SincNet [12]. We also trained a model based on the recent E-TDNN architecture used for x-vector extraction [4]. Instead of extracting x-vectors from the network and comparing test vectors to model vectors as in the standard method, we simply used the output of the E-TDNN network as our language classification.

All of our models were trained in PyTorch on a P100 GPU. We used a batch size of 128 and cross-entropy loss for all models. We used an early stopping criterion of 10 epochs with no improvement in validation loss and chose the epoch with the lowest validation loss as our final weights for each model.

Our M5 model is a deep convolutional neural network that operates on the raw input signal, based on [2]. The network is composed of four convolutional blocks, each consisting of a 1D convolutional layer, batch normalization, dropout, and max pooling. These are followed by global average pooling and two dense layers, ending in a softmax activation. The M5 network was optimized using the Adam optimizer in its default configuration. The full architecture can be found in Table 4.

Table 4. M5 architecture

Layer	Dimensions
Conv1D (128 filters, kernel size 80, stride 4)	128 × 3981
Batch norm, 10% dropout, max pooling (size 4)	128 × 995
Conv1D (128 filters, kernel size 3)	128 × 993
Batch norm, 10% dropout, max pooling (size 4)	128 × 248
Conv1D (256 filters, kernel size 3)	256 × 246
Batch norm, max pooling (size 4)	256 × 61
Conv1D (512 filters, kernel size 3)	512 × 59
Batch norm, max pooling (size 4)	512 × 14
Global average pooling	512
Dense (100 neurons), ReLU activation	100
Dense (8 neurons), softmax activation	8

The SincNet model was taken from the authors' GitHub repository[8] with hyperparameters based off of their TIMIT configuration. SincNet attempts to learn more meaningful filterbanks from the raw waveform data using bandpass filters, as opposed to the human-created filterbanks used in models such as the E-TDNN. The first layer is a convolutional layer that learns filterbanks. This is followed by two more convolutional layers and four dense layers. The network was optimized using RMSprop as suggested by the authors. More detailed information on the architecture can be found in the authors' GitHub and the original paper [12].

Our E-TDNN model is based on the recent architecture for x-vector extraction [4]. In comparison to the original architecture mentioned earlier, the E-TDNN adds an extra TDNN (1D convolutional) layer with dense layers in be-

[8] https://github.com/mravanelli/SincNet/

tween the TDNN layers and an extra dense layer at the end. It also adds batch normalization layers after each TDNN and dense layer and uses Leaky ReLU activations instead of the standard ReLU. Our final layer is a softmax activation over our 8 languages, giving us predicted probabilities for each language. This network was optimized using the Adam optimizer in its default configuration. The full architecture can be found in Table 5.

Table 5. E-TDNN architecture

Layer	Context	Output dimension
TDNN, Leaky ReLU, 1D batch norm	$[t-2, t+2]$	198
Dense, Leaky ReLU, 1D batch norm	t	512
TDNN, Leaky ReLU, 1D batch norm	$\{t-2, t, t+2\}$	198
Dense, Leaky ReLU, 1D batch norm	t	512
TDNN, Leaky ReLU, 1D batch norm	$\{t-3, t, t+3\}$	198
Dense, Leaky ReLU, 1D batch norm	t	512
TDNN, Leaky ReLU, 1D batch norm	$\{t-4, t, t+4\}$	198
Dense, Leaky ReLU, 1D batch norm	t	512
Dense, Leaky ReLU, 1D batch norm	t	1536
Mean & SD pooling	Full	3072
Dense, Leaky ReLU	Full	512
Dense, Leaky ReLU	Full	512
Dense, Softmax	Full	8

6 Results

The summary of results can be found in Table 6. The E-TDNN was by far the best performing model with a test accuracy of 90.5% on our eight-language Common Voice/Google Bengali (CV/GB) test set. It trained in an hour and a half per epoch and converged very quickly. The M5 model performed the best of the waveform-based networks, with a test accuracy of 77.5%. It also trained the fastest of all our models, at only 11 minutes per epoch. SincNet performed much worse than M5 and took nearly four times as long per epoch to train, making it a poor choice for our data.

As mentioned before, we also created a holdout test set using data from Audio Lingua (AL) to test our models' generalization to different data sources. This test set consists of five languages: Catalan, English, French, German, and Spanish. We also evaluated our models' performance on a subset of our Common Voice/Google Bengali test set containing only these languages to compare our models' performance between the two test sets. We calculated the difference in between these two accuracy numbers as a proxy for how well our models generalized. The Audio Lingua test set results can be found in Table 7.

Table 6. Model comparison

Model	Test accuracy (CV/GB)	Training time per epoch
E-TDNN	90.5%	1.5 hours
M5	77.5%	11 minutes
SincNet (20% dropout)	63.0%	40 minutes

Table 7. Comparison of models on Audio Lingua test set

Model	Test accuracy (AL)	Test accuracy (CV/GB 5 languages)	Difference
E-TDNN	74.8%	88.8%	−14.0%
M5	49.7%	73.6%	−23.8%
SincNet (20% dropout)	36.6%	59.5%	−23.0%

6.1 Problems with SincNet and Overfitting

In the default configuration with no dropout, SincNet massively overfit our training data, reaching over 99% accuracy while only hitting 65% validation accuracy. To combat this, we tried adding different amounts of dropout in the dense layers. We tested both 20% and 50% dropout. The training curves for each model can be seen in Figure 3.

We can see that lower levels of dropout rapidly increased to arbitrarily high levels of accuracy while not improving in validation accuracy or loss. However, although adding 20% and 50% dropout did help in preventing overfitting, they did not actually improve the validation accuracy or loss. In the end, we selected the 20% dropout configuration for our final SincNet model due to its slightly lower minimum validation loss. It also gave the best performance on our test sets, as seen in Table 8.

Table 8. Comparison of dropout levels in SincNet

Dropout	Test accuracy (CV/GB)	Test accuracy (Audio Lingua)
None	62.1%	31.5%
20% dropout	63.7%	36.6%
50% dropout	63.0%	35.5%

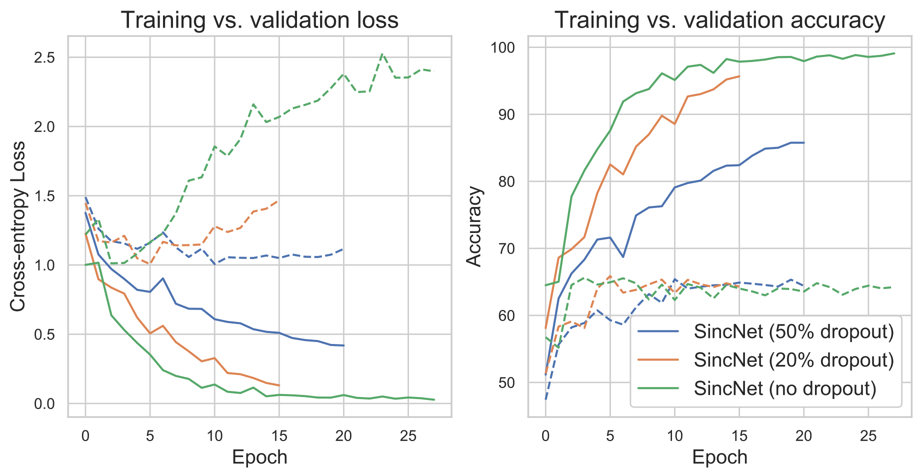


Fig. 3. Comparison of different levels of dropout in SincNet

6.2 Training Curves

Training curves for our models can be seen in Figure 4. The E-TDNN model outperforms even the best results of the M5 and SincNet models from the first epoch. The M5 model experiences a bit of erratic behavior in a couple epochs, but otherwise the training curves change fairly smoothly over each epoch, especially for the E-TDNN. The E-TDNN model converged in eight epochs, M5 converged in twelve epochs, and SincNet converged in five epochs.

6.3 Confusion Matrices for E-TDNN

The Common Voice/Google Bengali test set confusion matrix for the E-TDNN model can be seen in Figure 5. Each entry on the diagonal shows the recall for that language. We can see that Bengali, French, and German performed especially strongly. The most frequent error is Catalan being mistaken as Spanish by the model, which is understandable considering the similarity of these two languages. One possible concern that we can see is how well Bengali performed, achieving a 99.8% recall. This could be a genuinely high number, but more likely the model is learning different audio characteristics of the Google Bengali corpus compared to the Common Voice corpus.

Figure 6 shows the confusion matrix for the Audio Lingua test set. We also show a confusion matrix of a subset of the CV/GB test set containing only the Audio Lingua languages for comparison. English has the strongest generalization in the Audio Lingua test set with an 80% recall, compared to an 86% recall in the CV/GB test set. French and German also perform fairly well. Catalan performs very poorly, being mistaken mostly for Spanish, but also for English and French. Spanish is commonly mistaken for both Bengali and Catalan.

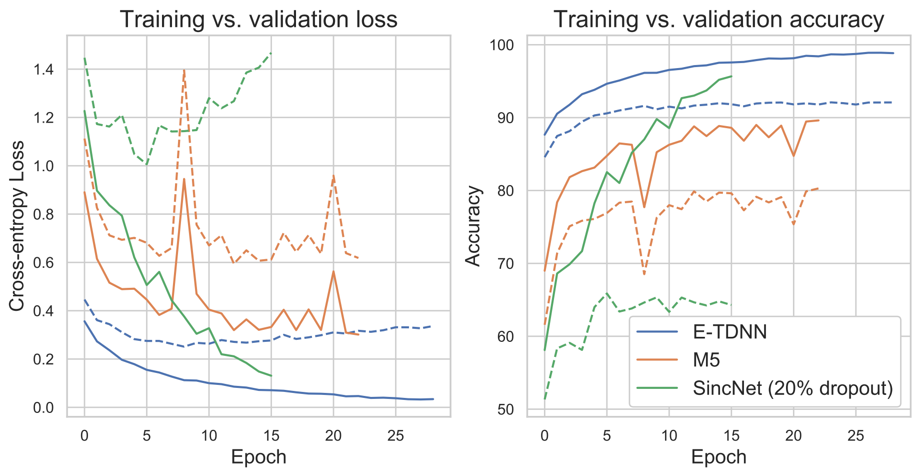


Fig. 4. Training curves

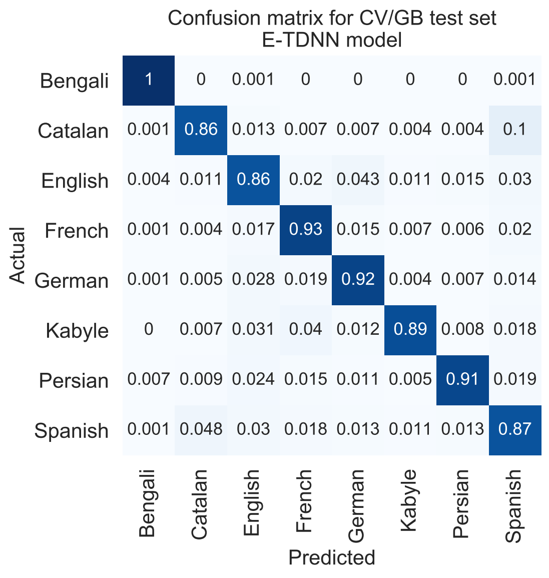
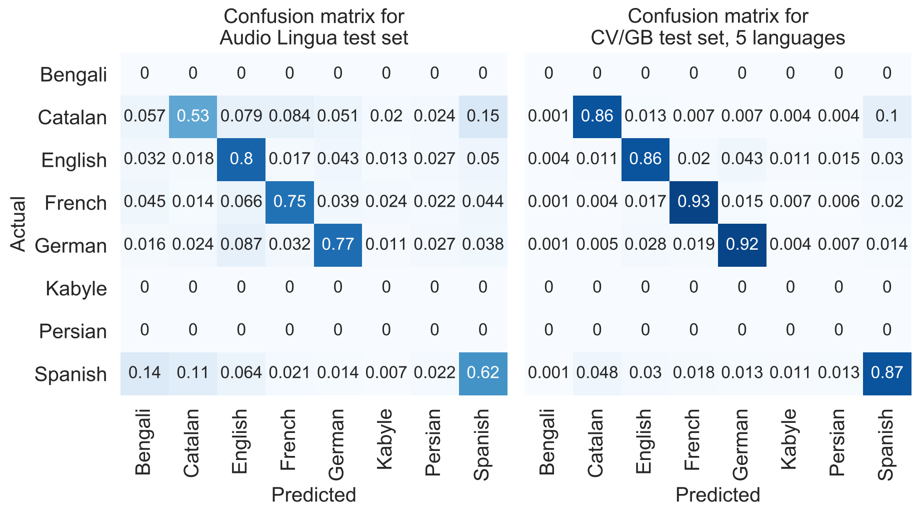


Fig. 5. Confusion matrix for E-TDNN model (CV/GB test set)



Actual \ Predicted	Bengali	Catalan	English	French	German	Kabyle	Persian	Spanish
Bengali	0	0	0	0	0	0	0	0
Catalan	0.057	0.53	0.079	0.084	0.051	0.02	0.024	0.15
English	0.032	0.018	0.8	0.017	0.043	0.013	0.027	0.05
French	0.045	0.014	0.066	0.75	0.039	0.024	0.022	0.044
German	0.016	0.024	0.087	0.032	0.77	0.011	0.027	0.038
Kabyle	0	0	0	0	0	0	0	0
Persian	0	0	0	0	0	0	0	0
Spanish	0.14	0.11	0.064	0.021	0.014	0.007	0.022	0.62

Actual \ Predicted	Bengali	Catalan	English	French	German	Kabyle	Persian	Spanish
Bengali	0	0	0	0	0	0	0	0
Catalan	0.001	0.86	0.013	0.007	0.007	0.004	0.004	0.1
English	0.004	0.011	0.86	0.02	0.043	0.011	0.015	0.03
French	0.001	0.004	0.017	0.93	0.015	0.007	0.006	0.02
German	0.001	0.005	0.028	0.019	0.92	0.004	0.007	0.014
Kabyle	0	0	0	0	0	0	0	0
Persian	0	0	0	0	0	0	0	0
Spanish	0.001	0.048	0.03	0.018	0.013	0.011	0.013	0.87

Fig. 6. Confusion matrices for E-TDNN model (AL test set and CV/GB 5-language test set)

7 Ethics

Using audio data from the internet always raises ethical concerns, including legality and privacy. We only obtained data licensed under Creative Commons and followed all restrictions imposed by those licenses carefully. We had considered using other sources such as free audiobooks and movies, but this turned out not to be necessary. If we had chosen to use these sources, careful investigation of those licenses would have been necessary to ensure compliance so that we were not using the data in a way that the authors would not approve of.

Privacy concerns are another issue. Speakers in the Common Voice and Bengali corpora are identified by a secret identifier. This is done to protect the privacy of the speakers who recorded audio for these datasets. We did not attempt to determine the identity of any of the speakers in the data we used, nor did we attempt to determine whether clips by unknown speakers were from the same speaker. Audio Lingua speakers voluntarily provided information including their first name and location. We stored this information intending to use it when separating speakers into training, validation, and test sets, but in the end we only used the Audio Lingua data as a test set. Therefore, we did not end up using the speaker information at all, nor did we attempt to identify speakers from their provided information.

Audio Lingua data was not submitted for the purpose of being used in speech and language recognition projects, but we believe that this use falls under the Creative Commons license it was published under.

8 Conclusion

8.1 Challenges Faced

We encountered a number of challenges over the course of this project that we did not anticipate going in. The first challenge we encountered was in research. Many of the papers we read assumed some prior level of knowledge about speaker and language recognition techniques. This information was hard to find, especially considering the relative obscurity of audio classification techniques compared to more popular tasks like image classification and object detection. Another challenge on this front is that many cutting-edge papers are implemented in Kaldi, a command-line toolkit for automatic speech recognition written in C++. We wanted to work in Python for this project, so we had to find other ways to implement our process. One final challenge is that many papers in speech and language recognition use metrics such as equal error rate (EER) that do not have a standard implementation in many of the popular machine learning packages, making it more difficult for us to compare our results to other papers.

Data availability was another issue. As mentioned earlier, many of the multilingual speech corpora used in this field are either private or expensive. Common Voice and OpenSLR were big assets in this regard, but many languages, including several widely spoken languages such as Hindi, Arabic, Portuguese, Japanese, and Korean, only had a small amount of data or none at all. In addition, the amount of data did not necessarily correlate with the number of speakers, as some small languages such as Kabyle had large amounts of data. This data is useful in modeling, but not necessarily in practice. We found a few corpora that had papers published in conference proceedings, but we were unable to locate the associated datasets. The data that was available was generally limited to short recordings of 2-5 seconds, causing us to have to use only very short segments in our models.

Another challenge was the availability of computing resources we had as students. We had access to our university's compute cluster, but due to upload constraints we had to store our audio recordings elsewhere in the cloud. We often ran into memory limitations importing our data or extracting raw waveforms and filterbanks, causing us to have to split our data into several chunks. We learned that torchaudio's Kaldi-compliant filterbank function does not support GPU processing, causing the extraction process to go slower on top of bandwidth limitations from downloading each file from the cloud. We found that reading MP3 files is a large bottleneck in the filterbank extraction process. It took about a day to extract filterbanks from 200,000 FLAC files for the Google Bengali dataset, compared to three or more days to extract the same number of MP3 files for each language in the Common Voice dataset, even though both datasets have roughly the same average clip duration.

8.2 Results and Future Work

We found that the E-TDNN model performed very well for our open-source datasets, achieving a 90.5% test accuracy on our Common Voice/Bengali dataset and a 74.8% accuracy on our Audio Lingua dataset. This filterbank-based model outperformed our waveform-based models, with M5 achieving better results than SincNet. Although the E-TDNN took much longer to train per epoch than either the M5 or SincNet models, it converged in a small number of epochs. The E-TDNN model performed very well on Bengali, French, German, and Persian, with over 90% recall in each. It struggled most with Catalan and English, confusing them most often with the related languages Spanish and German, respectively.

Generalization was a problem for all of our models, with the E-TDNN having a difference in accuracy of 14% between the Audio Lingua and Common Voice/Google Bengali test sets. This also showed in our model's ability to predict Bengali almost perfectly due to it being taken from a different dataset. We can attempt to address this in the future using data augmentation techniques. One popular data augmentation technique, as mentioned earlier, is to artificially insert background noise into audio files [16]. Another approach is to use techniques such as time masking, frequency masking, and time warping directly on the filterbanks, as in SpecAugment [10]. An interesting experiment in the future would be to compare these techniques to see which performs better, or perhaps if they perform better when used together.

Overall, we are very pleased with the results of this project. We have shown that some of the state-of-the-art models over the past few years can be applied very well to open-source datasets. We hope this inspires others to use open-source datasets in their research for easier reproducibility and to contribute to open-source speech databases such as Common Voice.

References

1. Arvaniti, A., Ross, T.: Rhythm classes and speech perception. Proceedings of Speech Prosody 2010 (2010)
2. Dai, W., Dai, C., Qu, S., Li, J., Das, S.: Very deep convolutional neural networks for raw waveforms. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 421–425 (2017)
3. Drugman, T., Stylianou, Y., Kida, Y., Akamine, M.: Voice activity detection: Merging source and filter-based information. IEEE Signal Processing Letters **23**, 1–1 (01 2015). https://doi.org/10.1109/LSP.2015.2495219
4. Garcia-Romero, D., Snyder, D., Sell, G., McCree, A., Povey, D., Khudanpur, S.: x-vector dnn refinement with full-length recordings for speaker recognition. In: INTERSPEECH (2019)
5. Kardava, I., Antidze, J., Gulua, N.: Solving the problem of the accents for speech recognition systems. International Journal of Signal Processing Systems **4**, 235–238 (06 2016). https://doi.org/10.18178/ijsps.4.3.235-238
6. Kjartansson, O., Sarin, S., Pipatsrisawat, K., Jansche, M., Ha, L.: Crowd-sourced speech corpora for javanese, sundanese, sinhala, nepali, and bangladeshi bengali. In: Proc. The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages. pp. 52–55 (2018). https://doi.org/10.21437/SLTU.2018-11, http://dx.doi.org/10.21437/SLTU.2018-11

7. Li, H., Ma, B., Lee, K.A.: Spoken language recognition: From fundamentals to practice. Proceedings of the IEEE **101**, 1136–1159 (05 2013). https://doi.org/10.1109/JPROC.2012.2237151
8. Logan, B.: Mel frequency cepstral coefficients for music modeling. Proc. 1st Int. Symposium Music Information Retrieval (11 2000)
9. Matejka, P., Zhang, L., Ng, T., Mallidi, S.H., Glembek, O., Ma, J., Zhang, B.: Neural network bottleneck features for language identification. In: Odyssey. pp. 299–304 (6 2014)
10. Park, D.S., Chan, W., Zhang, Y., Chiu, C.C., Zoph, B., Cubuk, E.D., Le, Q.V.: Specaugment: A simple data augmentation method for automatic speech recognition. Interspeech 2019 (Sep 2019). https://doi.org/10.21437/interspeech.2019-2680, http://dx.doi.org/10.21437/interspeech.2019-2680
11. Raj, D., Snyder, D., Povey, D., Khudanpur, S.: Probing the information encoded in x-vectors. In: 2019 IEEE Automatic Speech Recognition and Understanding Workshop. pp. 726–733 (12 2019). https://doi.org/10.1109/ASRU46091.2019.9003979
12. Ravanelli, M., Bengio, Y.: Speaker recognition from raw waveform with sincnet. 2018 IEEE Spoken Language Technology Workshop (SLT) (Dec 2018). https://doi.org/10.1109/slt.2018.8639585, http://dx.doi.org/10.1109/SLT.2018.8639585
13. Sak, H., Senior, A., Beaufays, F.: Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH pp. 338–342 (01 2014)
14. Snyder, D., Garcia-Romero, D., McCree, A., Sell, G., Povey, D., Khudanpur, S.: Spoken language recognition using x-vectors. In: Odyssey. pp. 105–111 (06 2018). https://doi.org/10.21437/Odyssey.2018-15
15. Snyder, D., Garcia-Romero, D., Povey, D., Khudanpur, S.: Deep neural network embeddings for text-independent speaker verification. In: Interspeech. pp. 999–1003 (08 2017). https://doi.org/10.21437/Interspeech.2017-620
16. Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., Khudanpur, S.: X-vectors: Robust dnn embeddings for speaker recognition. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 5329–5333 (2018). https://doi.org/10.1109/ICASSP.2018.8461375