Spoken Language Detection

Speech Recognition

Michael G. Campiglia
Stanford University
mikecamp@stanford.edu

Nicholas A. Graziano
Stanford University
ngraz@stanford.edu

Gabriel Ortuno II
Stanford University
gortuno@stanford.edu

Abstract

Identifying a language only through audio can be a difficult task with the additional complexities not present in text such as accents, age, gender, pitch, speed, and speaking proficiency. Before any technology that takes audio commands can parse what is requested, it must identify the correct tongue. Hence this paper focuses on tackling this dilemma by taking in audio files and converting them into Mel Frequency Cepstral Coefficients (MFCC) that are fed as inputs to a convolutional neural network (CNN) for language detection. The model incorporates use of SpecAugment [1], to help generalize learning during training on twelve languages from the Voxlingua107 dataset [2] (English, Mandarin, Hindi, Spanish, French, Arabic, Bengali, Russian, Portuguese, Urdu, German, Japanese). Using this approach, the team obtained an 88% accuracy utilizing 10 second audio clips.

1 Introduction

With the advent of machine translation in recent years the ability to translate a source text to one's native language has become almost trivial. Even in the case that the source language is unknown, tools such as Google Translate have integrated a *Detect Language* feature that generally does a good job, provided the input text is transcribed correctly in the original language. The task of identifying an unknown spoken language, however, poses a more difficult challenge.

Firstly, orthography, or how a language is written, provides a major clue in identifying which language one is reading. On the other hand, if one simply hears a sound, they are at a significant disadvantage. Was that an English "sh", a French "ch" or a German "sch"? Note that all three of these examples use the Latin alphabet. Hence, there is an ever growing interest in creating effective models to distinguish between languages clearly and efficiently. Explicitly, our team will take approximately 10 second audio samples of English, Mandarin, Hindi, Spanish, French, Arabic, Bengali, Russian, Portuguese, Urdu, German, and Japanese as inputs that then get converted to MFCCs after a transformation to mel-spectrograms for a CNN to perform categorical cross-entropy and output the most likely language candidate for each audio sample.

2 Related Work

State-of-the-art approaches for this task have converted audio utterances into spectrogram images and feed them into CNNs to detect features for classification. The team used this approach as the basis for this report as it is an innovative technique that has one of the highest accuracy amongst all algorithms. In particular, the team leveraged the existing CNN implementation by Singh et al. [3] that achieved approximately 98% accuracy on the Spoken language identification [4] dataset. This work builds upon those such as Revay et al. [5], that performed language identification for audio spectrograms (LIFAS) with a 89% accuracy on six languages of 3.75 second audio clips. As well as Sarthak et al. [6] which used log-Mel spectrogram images for language identification that classified six languages (English, French, German, Spanish, Russian and Italian) with an accuracy of 95.4% from the VoxForge [7] dataset.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

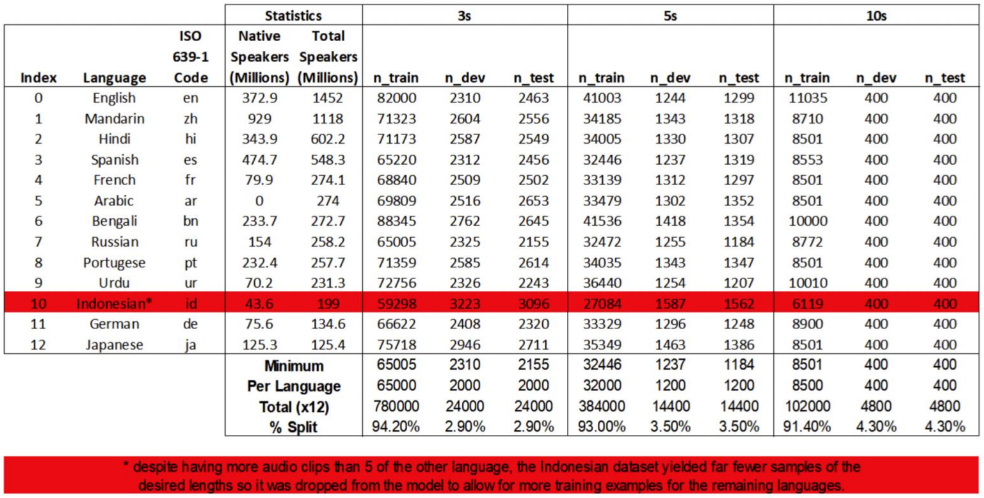
It is important to note that these references use a small set of languages, meanwhile our team would like to expand this concept to more languages and test with smaller audio samples. There is a known weakness in using CNNs for audio language identification however, and that is the risk of overfitting. SpecAugment [1], a data augmentation method for speech recognition, warps spectrogram features as well as masks blocks of frequency and time steps. The development of this method by Park et al. [1] achieves state-of-the-art performance on the LibriSpeech 960h and Switchboard 300h tasks, hence our team will utilize it for data augmentation.

3 Dataset and Features

Through initial research, a plethora of datasets were found of numerous spoken languages with varying sentence lengths, speakers, pitch, and speed. The team's basis model by Singh et al. [3] uses the Kaggle Spoken Language Identification [4] speech samples from the English, German, and Spanish languages. The dataset has 73080 training samples as Free Lossless Audio Codec (FLAC) audio files with a duration of 10 seconds and sample rate of 22,050 Hz. In order to expand the capabilities of this model, a bigger dataset is needed to extend to more languages as well as test the robustness of variations such as sample length.

The team chose to use VoxLingua107 [2], a dataset consisting of short speech segments from YouTube videos. VoxLingua107 contains 107 languages, as the name suggests, with roughly 62 hours of data per language (over 6600 hours total). The speech segments range from approximately 4-15 seconds and were preprocessed by the team to get sample lengths of 3, 5, and 10 seconds. A complete breakdown of the total number of audio samples per time segment and train, dev, or test sets can be found in Table 1, yet this excludes additional data that will be created from SpecAugment [1]. The team only expanded the model to the twelve most spoken languages due to computation constraints, but will use this as a proof of concept that incorporating more languages is feasible with this plentiful and well-labeled dataset.

Table 1: Dataset Breakdown



			Statistics		3s			5s			10s		
Index	Language	ISO 639-1 Code	Native Speakers (Millions)	Total Speakers (Millions)	n_train	n_dev	n_test	n_train	n_dev	n_test	n_train	n_dev	n_test
0	English	en	372.9	1452	82000	2310	2463	41003	1244	1299	11035	400	400
1	Mandarin	zh	929	1118	71323	2604	2556	34185	1343	1318	8710	400	400
2	Hindi	hi	343.9	602.2	71173	2587	2549	34005	1330	1307	8501	400	400
3	Spanish	es	474.7	548.3	65220	2312	2456	32446	1237	1319	8553	400	400
4	French	fr	79.9	274.1	68840	2509	2502	33139	1312	1297	8501	400	400
5	Arabic	ar	0	274	69809	2516	2653	33479	1302	1352	8501	400	400
6	Bengali	bn	233.7	272.7	88345	2762	2645	41536	1418	1354	10000	400	400
7	Russian	ru	154	258.2	65005	2325	2155	32472	1255	1184	8772	400	400
8	Portugese	pt	232.4	257.7	71359	2585	2614	34035	1343	1347	8501	400	400
9	Urdu	ur	70.2	231.3	72756	2326	2243	36440	1254	1207	10010	400	400
10	Indonesian*	id	43.6	199	59298	3223	3096	27084	1587	1562	6119	400	400
11	German	de	75.6	134.6	66622	2408	2320	33329	1296	1248	8900	400	400
12	Japanese	ja	125.3	125.4	75718	2946	2711	35349	1463	1386	8501	400	400
			Minimum		65005	2310	2155	32446	1237	1184	8501	400	400
			Per Language		65000	2000	2000	32000	1200	1200	8500	400	400
			Total (x12)		780000	24000	24000	384000	14400	14400	102000	4800	4800
			% Split		94.20%	2.90%	2.90%	93.00%	3.50%	3.50%	91.40%	4.30%	4.30%

* despite having more audio clips than 5 of the other language, the Indonesian dataset yielded far fewer samples of the desired lengths so it was dropped from the model to allow for more training examples for the remaining languages.

4 Methods

Our team tackled this challenge through an image-based approach in which we converted raw audio data into Mel spectrograms, transformed them into MFCCs, then applied a CNN architecture to extract features for detection.

Mel spectrograms are commonly used in audio classification tasks instead of traditional spectrums since humans can better detect differences in lower frequencies than higher frequencies. Mel accounts for this by adjusting relative pitches to better replicate how they would be perceived by an actual human listener [8]. An MFCC is generated by performing a Discrete Cosine Transform (DCT) on the Mel spectrogram. This step is critical as using the Mel spectrogram directly contains superfluous data that makes training difficult, hence reducing the dimensionality through MFCC is a needed step.

Briefly, CNNs take input images and traverse them with filters to slowly learn and detect features such as edges. Multiple layers can be placed in sequence to detect higher level features, gradually building a network of weights and biases that can detect items even as complex as language. We created our own version of the CNN framework within Singh et al. [3], employing deep learning generalizing techniques and best practices such as adding additional drop out layers, regularization, use of He initialization to reduce chance of exploding or vanishing gradients, and splitting a single dense layer into two smaller dense layers with less parameters to learn. Our CNN consists of five separate convolutional layers, each individually followed by ReLU activation, batch normalization, and max pooling. The output was then fed into a series of flattening, normalizing, ReLU activations, and dropouts along the way. Finally, the CNN ended with a softmax activation function to identify each audio sample as one of the twelve we trained on. The entire CNN architecture can be found in Appendix A, but a simplified representation of this can be seen in Figure 1.

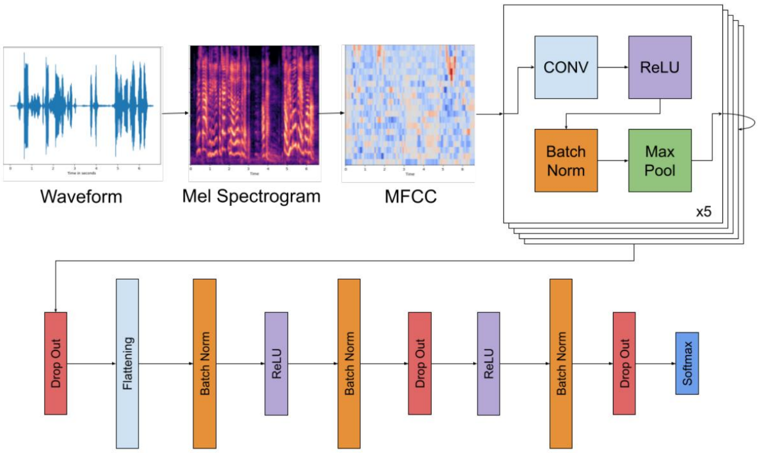


Figure 1: CNN Architecture

We used categorical cross-entropy (AKA Softmax loss) as our loss function since each sample can only belong to one out of many possible languages. This allowed our CNN to learn to output a probability over the amount of languages we have, selecting the top probability as the language most likely being spoken. Equation 1 demonstrates this function where $\hat{y}_i$ is the probability the model outputs that the language is class i while y_i is the true target value. Lastly, $n_{classes}$ is the number of classes the model can output.

$$Loss = -\sum_{i=1}^{n_{classes}} y_i \cdot \log \hat{y}_i \tag{1}$$

5 Experiments/Results/Discussion

In order to begin training the model, good hyperparameters for our model had to be selected and were chosen by leveraging the trial and error results from the works of Singh et al. [3], Revay et al. [5] and Sarthak et al. [6]. Building upon their success and standard deep learning parameters, we selected softmax as the activation output, ReLU as hidden layer activation functions, as well as the number of hidden layers depicted in Figure 1. We also increased the typical batch size to 64 and epochs to 100 due to the higher amount of data samples we have, explicitly detailed in Table 1.

For evaluation metrics, we employed precision, recall, and F1 score. Precision is the ratio of correct positive predictions from the total positive class. Recall is the ratio of correct positive predictions from the total real positive cases. Lastly, F1 score represents the harmonic mean of precision and recall. Mathematically, these metrics are listed as Equation 2, 3, and 4 below.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{3}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

After parameter and evaluation metric selection, the team attempted to do a simple addition of twelve total languages to the base CNN architecture within Singh et al. [3] to see the results and pitfalls. We decided to purposely stress the model by training with the original VoxLingua107 audio clips of length 3, 5, and 10 seconds independently with the results depicted in Figure 2. It is extremely evident in the results that the model is overfitting on the short 3 and 5 second clips as the loss function diverges over the course of training. The team was expecting this based on related work, hence we took this into consideration when developing our own CNN architecture described within the Methods section.

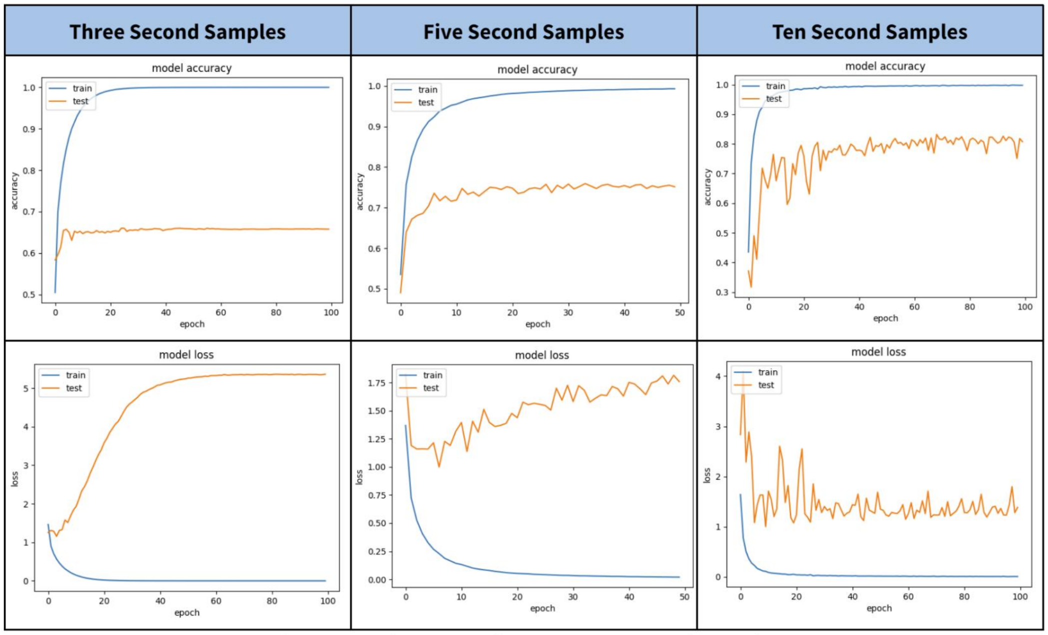


Figure 2: Three vs. Five vs. Ten second samples

Though key architecture changes were already noted, the most important expansion to our model was the use of SpecAugment [1]. This data augmentation allowed our training data to go from 102 thousand examples to 510 thousand. Besides just having more data, it helped generalize the model and limit overfit with the removal of segments it would perform such as the example in Figure 3. Therefore, when our improved model was trained on 10 second samples with this augmented dataset, we saw a clear improvement in precision, recall, and F1 score percentages as seen in Table 2.

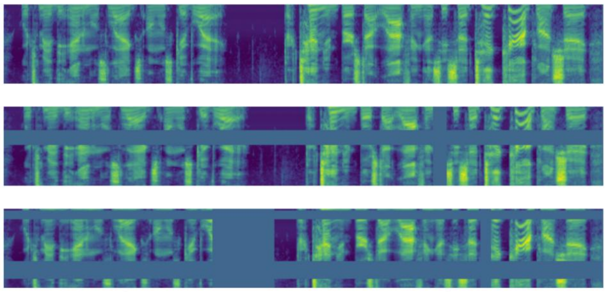


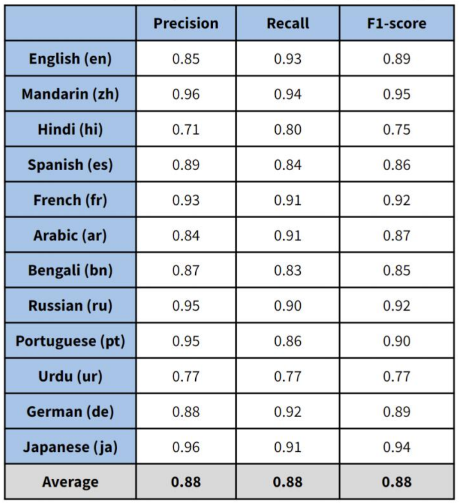
Figure 3: SpecAugment Example [1]

Table 2: Training Results

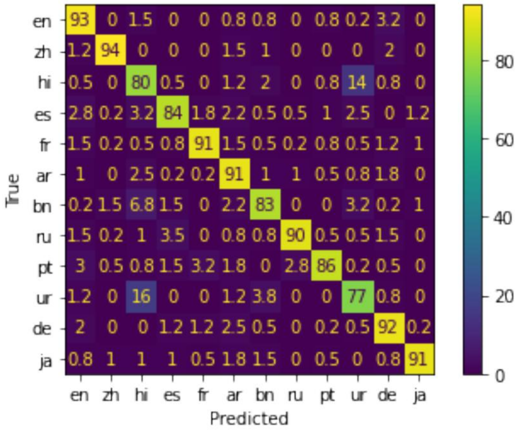
	Baseline Model 3 second	Baseline Model 5 second	Baseline Model 10 second	Our Model 10 second
Precision	.67	.77	.83	.88
Recall	.67	.76	.82	.88
F1	.67	.76	.82	.88

Performing a deeper analysis, a breakdown of the 10 second results for each language using our model is shown in Table 3 followed by a confusion matrix in Figure 4. A very interesting thing to see is that the languages with the lowest scores are those that are very similar such as Hindi and Urdu. The model struggles on these languages because they originally developed from the same dialect and share over 70% of their vocabulary at the beginner level [10]. Cognates, words that look and sound similar in languages because of a common origin, make adding more languages a tough task. Even with this in mind, the model performs better than the original with an 88% average F1 score amongst the twelve languages.

Table 3: Language Metrics, 10 sec samples



	Precision	Recall	F1-score
English (en)	0.85	0.93	0.89
Mandarin (zh)	0.96	0.94	0.95
Hindi (hi)	0.71	0.80	0.75
Spanish (es)	0.89	0.84	0.86
French (fr)	0.93	0.91	0.92
Arabic (ar)	0.84	0.91	0.87
Bengali (bn)	0.87	0.83	0.85
Russian (ru)	0.95	0.90	0.92
Portuguese (pt)	0.95	0.86	0.90
Urdu (ur)	0.77	0.77	0.77
German (de)	0.88	0.92	0.89
Japanese (ja)	0.96	0.91	0.94
Average	**0.88**	**0.88**	**0.88**



	en	zh	hi	es	fr	ar	bn	ru	pt	ur	de	ja
en	93	0	1.5	0	0	0.8	0.8	0	0.8	0.2	3.2	0
zh	1.2	94	0	0	0	1.5	1	0	0	0	2	0
hi	0.5	0	80	0.5	0	1.2	2	0	0.8	14	0.8	0
es	2.8	0.2	3.2	84	1.8	2.2	0.5	0.5	1	2.5	0	1.2
fr	1.5	0.2	0.5	0.8	91	1.5	0.5	0.2	0.8	0.5	1.2	1
ar	1	0	2.5	0.2	0.2	91	1	1	0.5	0.8	1.8	0
bn	0.2	1.5	6.8	1.5	0	2.2	83	0	0	3.2	0.2	1
ru	1.5	0.2	1	3.5	0	0.8	0.8	90	0.5	0.5	1.5	0
pt	3	0.5	0.8	1.5	3.2	1.8	0	2.8	86	0.2	0.5	0
ur	1.2	0	16	0	0	1.2	3.8	0	0	77	0.8	0
de	2	0	0	1.2	1.2	2.5	0.5	0	0.2	0.5	92	0.2
ja	0.8	1	1	1	0.5	1.8	1.5	0	0.5	0	0.8	91

Figure 4: Confusion Matrix

6 Conclusion/Future Work

Spoken language identification is no trivial task, particularly when languages share a common ancestry. As we saw in this report, languages with similar traits can be hard to distinguish. Hence, expanding the model to incorporate more languages produces an overall lower accuracy due to lexical borrowings. Furthermore, the longer an audio sample is, the higher the likelihood of our model discerning the language. Longer clips give the model a higher chance of finding distinct features only present in a particular language, catching the lexical dissimilarities.

In order to improve the model in the future, the team would need to add more data of each language, particularly those close in nature, increase GPU capability for the amount of processing needed, and perform an in-depth evaluation of the dataset for absent characteristics or overly repeated attributes. Are the unique aspects of said language even present in the dataset? Lastly, the team can use the results of the evaluation metrics to tune hyperparameters in an attempt to gain the best results.

Contributions

Michael Campiglia - Chief Deep Learning Architect Performed initial research for project, created overall convolutional neural network model/helper functions, trained and enhanced model, produced plots of training and validation results

Nick Graziano - Director of Dataset Exploration and Parsing Performed initial research for project, created scripts for data parsing, shuffling, and labeling, produced CNN graphic

Gabriel Ortuno - Editor-in-Chief of Project Reports Performed initial research for project, created Git repository, formed cohesive reports, performed dataset investigation for proper selection

References

[1] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, Q. V. Le “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition” April 2019

[2] VoxLingua107 http://bark.phon.ioc.ee/voxlingua107/ Accessed May 2022

[3] Gundeep Singh, Sahil Sharma, Vijay Kumar, Manjit Kaur, Mohammed Baz, Mehedi Masud, "Spoken Language Identification Using Deep Learning", Computational Intelligence and Neuroscience, vol. 2021

[4] Oponowicz, T “Spoken language identification” (2018), https://github.com/tomasz-oponowicz/spoken_language_identification Accessed May 2022

[5] S. Revay, M. Teschke, and Novetta, “Multi-class language identification using deep learning on spectral images of audio signals,” 2019

[6] S. S. Sarthak, S. Shukla, and G. Mittal, “Spoken language identification using convNets,” Lecture Notes in Computer Science book series (LNISA,volume 11912), 2019, LNCS

[7] voxforge.org: Free speech recognition - http://www.voxforge.org/ Accessed May 2022

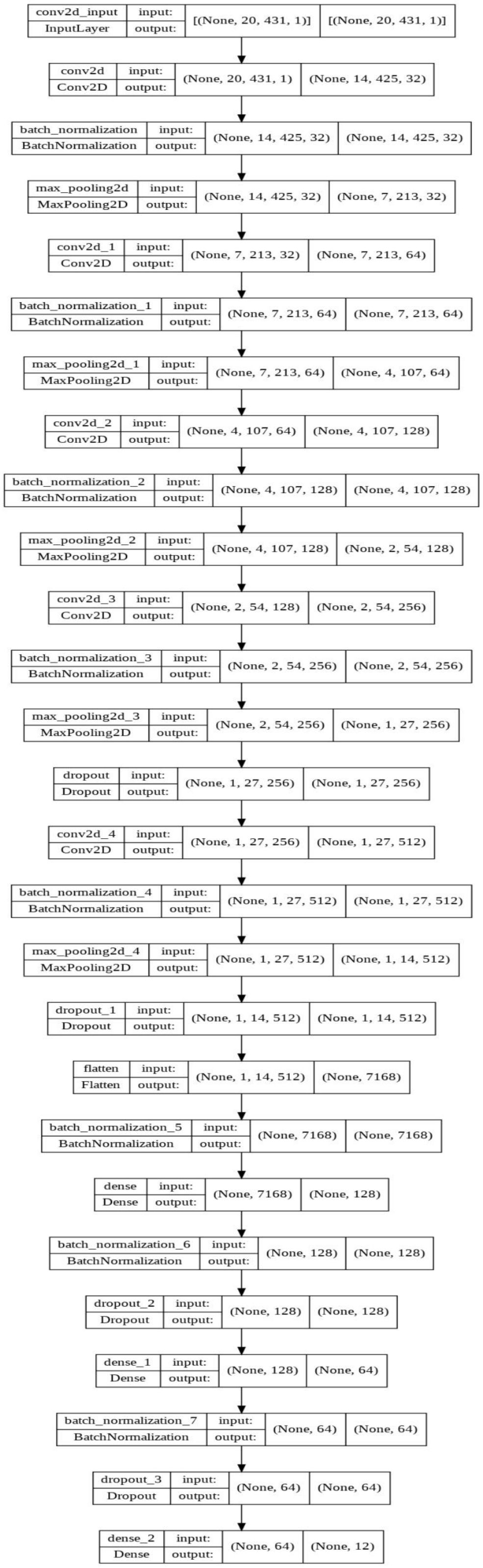
[8] Roberts, Leland “Understanding the Mel Spectrogram”, Mar 5, 2020 https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53

[9] Peltarion, “Categorical crossentropy”, https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy Accessed May 2022

[10] Miyatsu, Rose, “Hindi and Urdu in conversation” https://artsci.wustl.edu/ampersand/hindi-and-urdu-conversation Accessed May 2022

Appendix

A Full Convolutional Network


conv2d_input | InputLayer | input: | output: | [(None, 20, 431, 1)] | [(None, 20, 431, 1)]
conv2d | Conv2D | input: | output: | (None, 20, 431, 1) | (None, 14, 425, 32)
batch_normalization | BatchNormalization | input: | output: | (None, 14, 425, 32) | (None, 14, 425, 32)
max_pooling2d | MaxPooling2D | input: | output: | (None, 14, 425, 32) | (None, 7, 213, 32)
conv2d_1 | Conv2D | input: | output: | (None, 7, 213, 32) | (None, 7, 213, 64)
batch_normalization_1 | BatchNormalization | input: | output: | (None, 7, 213, 64) | (None, 7, 213, 64)
max_pooling2d_1 | MaxPooling2D | input: | output: | (None, 7, 213, 64) | (None, 4, 107, 64)
conv2d_2 | Conv2D | input: | output: | (None, 4, 107, 64) | (None, 4, 107, 128)
batch_normalization_2 | BatchNormalization | input: | output: | (None, 4, 107, 128) | (None, 4, 107, 128)
max_pooling2d_2 | MaxPooling2D | input: | output: | (None, 4, 107, 128) | (None, 2, 54, 128)
conv2d_3 | Conv2D | input: | output: | (None, 2, 54, 128) | (None, 2, 54, 256)
batch_normalization_3 | BatchNormalization | input: | output: | (None, 2, 54, 256) | (None, 2, 54, 256)
max_pooling2d_3 | MaxPooling2D | input: | output: | (None, 2, 54, 256) | (None, 1, 27, 256)
dropout | Dropout | input: | output: | (None, 1, 27, 256) | (None, 1, 27, 256)
conv2d_4 | Conv2D | input: | output: | (None, 1, 27, 256) | (None, 1, 27, 512)
batch_normalization_4 | BatchNormalization | input: | output: | (None, 1, 27, 512) | (None, 1, 27, 512)
max_pooling2d_4 | MaxPooling2D | input: | output: | (None, 1, 27, 512) | (None, 1, 14, 512)
dropout_1 | Dropout | input: | output: | (None, 1, 14, 512) | (None, 1, 14, 512)
flatten | Flatten | input: | output: | (None, 1, 14, 512) | (None, 7168)
batch_normalization_5 | BatchNormalization | input: | output: | (None, 7168) | (None, 7168)
dense | Dense | input: | output: | (None, 7168) | (None, 128)
batch_normalization_6 | BatchNormalization | input: | output: | (None, 128) | (None, 128)
dropout_2 | Dropout | input: | output: | (None, 128) | (None, 128)
dense_1 | Dense | input: | output: | (None, 128) | (None, 64)
batch_normalization_7 | BatchNormalization | input: | output: | (None, 64) | (None, 64)
dropout_3 | Dropout | input: | output: | (None, 64) | (None, 64)
dense_2 | Dense | input: | output: | (None, 64) | (None, 12)