CSVC-Net: Code-Switched Voice Command Classification using Deep CNN-LSTM Network

Arowa Yasmeen[§], Fariha Ishrat Rahman[§], Sabbir Ahmed and Md. Hasanul Kabir
Department of Computer Science and Engineering,
Islamic University of Technology, Dhaka, Bangladesh
Email: {arowayasmeen, farihaishrat, sabbirahmed, hasanul}@iut-dhaka.edu

***Abstract*—Colloquial Bengali has adopted many English words due to colonial influence. In conversational Bengali, it is quite common to speak in a mixture of English and Bengali, a phenomenon termed Code-switching (CS). To build a Voice Command Classifier in this era, when the usage of CS is ever-increasing, it is often necessary to map a single base command to its many different variants - spoken in multiple mixtures of languages. The works done with Bengali Speech have been primarily focused on single word classification and mostly incompetent in understanding the complex semantic relationships displayed in sentences. This paper proposes '*CSVC-Net*', a CNN-LSTM based architecture for classifying spoken commands that exhibit code-switching between Bengali and English. To effectively reflect the scenario, it also presents a newly curated dataset named '*Banglish*' containing 3,840 audio files of spoken computer commands belonging to 11 classes, considering 64 variations in total. The proposed pipeline passes the input audio signal through a series of appropriate transformation and augmentation steps enabling the model to achieve an accuracy of 92.08% on the curated dataset. Furthermore, the robustness of the proposed model has been justified by comparing with different architectures and tested under different noise levels with promising accuracy, which shows the applicability of the model in real-life scenarios.**

***Contribution*—Identifying the absence of a standardized dataset exhibiting Bengali-English code-switching, we have curated *'Banglish'*, a novel dataset of spoken computer commands, and proposed the *'CSVC-Net'* architecture that exploits the ability of Deep CNN-LSTM network for efficient classification of voice commands.**

***Index Terms*—CNN-LSTM, MFCC, Bengali/English Code-switching, Voice Command Classification, Bengali Voice Dataset, Speech Augmentation, Frequency masking**

I. Introduction

Speech Classification/Recognition is one of the bridges for human-machine communication. However, only recently has this domain enjoyed significant progress in terms of research. This late progress is mainly due to the efficiency of the alternative communication modalities, e.g. keyboard, mouse, touch screens etc. However, as of late, Speech Recognition technologies has garnered immense popularity through voice-based user interfaces like Microsoft's *Cortana*, Amazon's *Alexa*, Apple's *Siri*, and Google's *Google Now* [1].

According to Ethnologue's list of "200 Most Spoken Languages in the World", posted in 2021, Bengali ranks as the 6^{th} most popular language [2]. Despite having a speaker count of 268 million, it is still relatively unexplored compared to other popular languages. Moreover, most of the works done on Bengali assume the language to be spoken in its pure form. This assumption rouses certain limitations since many Bengali speakers are bilingual due to colonial influence and rapid globalisation. Therefore, it is pretty typical for a native speaker to talk in a mixture of Bengali and English. This particular situation of continuous alternation between two languages in a single conversation is called code-switching (CS) [3].

Although CS is commonly used in everyday conversations, there are not many works regarding this as a whole especially in Bengali Speech Classification. Moreover, the unavailability of vast public datasets contributes to the scarcity. Thus the classification of speech containing Bengali-English code-switching is a promising research problem to be considered.

In this work, we have explored code-switching in the context of classifying voice commands containing a mixture of Bengali-English words and introduced "Code-switched Voice Command Network (*CSVC-Net*)", a CNN-LSTM based architecture robust to this bilingual variance. Taking the phenomenon of CS into account, we have created "*Banglish*" - a dateset consisting of 3,840 audio files belonging to 10 different classes of computer commands - each of which has multiple variations in which it can be spoken - and one offset class. After the input signal is processed through a series of transformations, Mel-frequency Cepstral Coefficients (MFCC) features are extracted and fed into the model. Moreover, we have utilized different data augmentations techniques to make the model robust to the unseen real-life data. The model has achieved promising results even under moderate, high and extremely noisy condition of the dataset and its competency has been justified by comparing with some other standard choices of classifiers in the existing literature.

II. Literature Review

A. Related Works on Speech Recognition

Typically, Speech Recognition is done on either isolated words or continuous sentences. For the Isolated words, the assumption is that there is no need for prior knowledge of the word's context. In such a case, the recognition algorithm is simply a pattern-matching algorithm. The sequence of spectral vectors of the unknown audio input is matched against each set of spectral patterns of known audio, and the pattern with

[§]These authors have contributed equally to this work.

least time-aligned spectral distance is the recognised word [4]. Unlike isolated words, classifying continuous sentences requires sophisticated linguistic knowledge. The typical approach for building such a system is to have an acoustic model, a language model and a hypothesis search. Combined with the acoustic model and language model scores, the hypothesis search recognises the sentence with the maximum posterior probability as the output [5].

Most traditional systems use Gaussian Mixture Models (GMMs) - Hidden Markov Models (HMMs) to model the overall system. HMMs deal with the temporal variability of speech, while GMMs determine how well each state of each HMM fits a frame or a short window of frames. However, the use of Deep Neural Networks (DNNs) for speech recognition has proven to outperform the GMM-HMM architectures [6]. The extraordinary feature-learning capability of Deep Learning(DL) has alleviated the necessity of extreme feature engineering in the traditional approaches.

The earlier approaches of Bengali Speech Recognition depended heavily on vigilant preprocessing and feature extraction steps fed into traditional machine learning classifiers where the performance was susceptible to noise and mostly failed to generalise [7], [8]. A handful of recent approaches by [9]–[11] employed the MFCC-CNN architecture. While the works of Shuvo *et al.* [9] and Sharmin *et al.* [10] achieved remarkably high accuracy of 93.65% and 98.37%, respectively, on isolated speech, the accuracy obtained by Sumon *et al.* [11] was as low as 74.01% due to the attempt of recognising short speech commands instead of single words. Thus, it can be concluded that as the classification problem moves from an isolated utterance domain to a continuous one, the classic MFCC-CNN architecture model is simply not enough. We need an additional mechanism to understand the contextual and semantic relationship present in a continuous audio stream.

Gupta et al. [12] proposed a cross-correlation method to recognise continuous Bengali voice commands for building a digital personal assistant. Their dataset consisted of 240 audio files of 12 different voice commands tested in noisy, moderate and noiseless environments with corresponding accuracy values of 75%, 83% and 83%. Although this work comes closest to be classified as a work similar to our problem statement of recognising computer commands, it does not consider the problem of having high intra-class variability within a multi-class Bengali speech classification problem. In fact, none of the works mentioned above does so.

Usage of mixed language is getting prevalent among bilingual/multilingual communities, raising the necessity of building Speech Recognition systems specially designed to tackle these challenges. Eventually, multilingual training of DNN-based speech recognition systems has improved recognition accuracy for both high and low-resource languages [13]–[15]. Furthermore, code-switching(CS) is prevalent in many existing works on different languages. Experiments on bilingual CS data for Frisian and Dutch were performed by Yılmaz *et al.* [16]. They used a conventional GMM-HMM along with MFCC features to train bilingual phonemes and achieved a CS-Word Error Rate (WER) of 59.5%. Hamed *et al.* [17] introduced a code-switched Arabic-English text corpus and an appropriate Language Model (LM). Moreover, an attention-based end-to-end speech recognition model was proposed by Shan *et al.* [18] which achieved a 6.49% Character error rate (CER) on a Mandarin-English CS corpus. A complete survey was done on Code-switching through reviewing the available corpus, datasets, works and computational approaches [3]. This work justifies the challenges and exhibits many opportunities and applications of CS, especially while building intelligent agents and designing systems for multilingual communities.

Although the impact of research with CS is emergent, to the best of our knowledge, the impact of Deep Neural Network (DNN) training for Bengali-English code-switched audio data is yet to be explored.

B. *Recurrent Architectures for Speech Recognition*

To suit the problem statement, the choice can be narrowed down to three Deep Learning models - Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Out of these three choices, the objective is suited by RNN the most. Continuous speech deals with sequential information which can't be captured using ANNs. The same applies to CNNs, as they perform better for spatial data than sequential. On the other hand, while making predictions, RNNs investigate the dependency between the words in the audio data and can take full advantage of capturing the sequential information [19].

Recurrent Neural Networks allows exploiting the previous context through a loop in the network; an RNN cell takes an input X_t and gives an output h_t. An RNN cell with an unrolled loop looks like any other sequential neural network. Thus, each cell passes on its output and hidden state value to the next cell, enabling it to hold the previous context [20].

The disadvantage of a traditional RNN is it suffers from short-term memory. If the audio sequence is long enough, it is certainly possible that the RNN cannot use the context learnt from the beginning of the audio sequence. RNNs compute gradients through backpropagation through time. When gradient values shrink exponentially fast to small values as they backpropagate through time, layers cannot learn much and forget the previous context; this is called the problem of *vanishing gradients* [21]. In our case, we aim to make our model invariant of how long the audio stream is, and so it must carry forward information from the earlier time steps to the later ones. Hence, an LSTM is a much better choice as it can learn long-term dependencies through learning to bridge time intervals even in noisy audio streams, without losing short time lag capabilities [22]. LSTM itself has various derivations of it; Unidirectional LSTM, Gated Recurrent Unit (GRU) and Bidirectional LSTM (BiLSTM) are some of them [23].

LSTMs have internal mechanisms called *Gates* to regulate what information to retain and what to forget. These gates can learn the relevancy of data in a sequence and pass relevant information down the long chain of sequences to make predictions. Four Neural Network layers interact to form the *Forget Gate*, *Input Gate*, and *Output Gate*. The Forget Gate decides which previous information to keep and which to discard (1). The Input Gate decides which of the present information is essential to keep (2).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f]) \tag{1}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i]) \tag{2}$$
$$\widetilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C]) \tag{3}$$
$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{4}$$

One of LSTM's core concepts is the *Cell State* - combined with the outputs of the Input and Forget Gates (4). Cell state contains new values that the neural network finds relevant. The Output Gate (5) decides what the next hidden state should be. The hidden state contains information on previous inputs and is also used for predictions. The new cell state (4) and the new hidden state (6) are then carried over to the next time step.

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{5}$$
$$h_t = o_t * \tanh(C_t) \tag{6}$$

Another version of an LSTM is the GRU [24]. In GRU, instead of separate Forget, Input and Output Gates, there are the *Reset* and *Update* Gates. It also gets rid of the Cell State and uses only the Hidden State. The Update Gate (10) decides which of the new information to throw away and keep. The Reset Gate (9) regulates how much of the past information is retained. GRUs are simpler compared to LSTMs.

$$z_t = \sigma\left(W_z \cdot \left[h_{t-1}, x_t\right]\right) \tag{7}$$
$$r_t = \sigma\left(W_r \cdot \left[h_{t-1}, x_t\right]\right) \tag{8}$$
$$\widetilde{h}_t = \tanh\left(W \cdot \left[r_t * h_{t-1}, x_t\right]\right) \tag{9}$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h}_t \tag{10}$$

BiLSTM is essentially the same as an LSTM. A BiLSTM is two LSTMs - one which processes input in a forward direction and the other in the backward direction. Having the input being processed in both directions allows BiLSTM to exploit not only previous context but also the future context [25].

III. Dataset: ***Banglish***

Our curated dataset contains 3,840 audio samples obtained from 30 female and 30 male contributors. Audio file lengths vary from 1s to 6s. The voice commands are divided into eleven classes. Ten classes define ten different commands, each with five intra-class variations representing five different ways of speaking the same command. One extra class termed "Offset Class" consists of 14 variations, each with a different meaning. This class is included to avoid biased training of only ten commands since the system also needs to be able to identify if a command does not belong to any of the executable ten commands. The classes are listed as follows:

i) Close Tab (5 intraclass variations)
ii) Close Window (5 intraclass variations)
iii) Switch Tab (5 intraclass variations)
iv) Open New Tab (5 intraclass variations)
v) Minimise Window (5 intraclass variations)
vi) Maximise Window (5 intraclass variations)
vii) Restore Up (5 intraclass variations)
viii) Restore Down (5 intraclass variations)
ix) Refresh (5 intraclass variations)
x) Take Cursor to Search (5 intraclass variations)
xi) Offset Commands (14 commands not belonging to the other 10 classes)

Our dataset has robust and practical use cases, and so it has been constructed to have a more realistic representation of colloquial Bengali. Thus, this "bilingual" dataset consists of a mix of both Bengali and English words - hence the name "***Banglish***". For example, we have shown the five intra-class variations of the first command of the dataset in Table I. The command "Close Tab" can both be spoken as "Ei tab ta close koro" or "Ei tab ta bondho koro", where "bondho" is the Bengali equivalent of the English word "close". In this way, the dataset contains a mixture of Bengali and English words in each variations of each of the commands.[1]

Existing Bengali datasets are not reflective of the Code-switching phenomenon. In addition, these datasets also do not consider the intra-class variation of each command, i.e. one single command spoken in multiple mixtures of Bengali and English. *Banglish*, on the other hand, takes into consideration all such features. In addition to variance in speech utterance of speakers, it poses high intra-class variations. Two commands of the same class can sound seemingly different but mean the same thing. To exemplify, two of the variations of command-3 are "Ei tab theke arek tab e jao" and "Tab switch koro". Other than having the word "tab" in common, they sound entirely different. However, they both mean "Switch tab". The dataset also manifests low inter-class variations. Two commands of different classes can sound quite similar yet display different meanings. For instance, the commands "Window ta boro koro" and "Window ta choto koro" respectively mean the window to "Restore up" and "Restore down". Despite being two different commands, they only differ by a single word. Hence, to make accurate predictions on this dataset, a model needs to be robust and distinguish the semantic information in these commands.

TABLE I
Intra-class variations of Computer Command-1

SL #	**Computer Command 1: Close the Tab**	
	English Transliteration	Bengali Command
1	Ei Tab ta Close koro	এই ট্যাব টা ক্লোজ কর
2	Ekhon jei Tab e achi eita Close koro	এখন যেই ট্যাব এ আছি এইটা ক্লোজ কর
3	Ei Tab bondho koro	এই ট্যাব বন্ধ কর
4	Ekhon jei Tab e achi ta bondho koro	এখন যেই ট্যাব এ আছি তা বন্ধ কর
5	Tab ta bondho koro	ট্যাব টা বন্ধ কর

[1]The entire dataset with detailed description and the codes are publicly available here: https://github.com/space-urchin/CSVC-Net.

IV. Methodology

A. Dataset Preprocessing and Feature Extraction Pipeline

All audio samples are converted to the '.wav' format, having a mono (single) channel, and sampled at 16kHz with a bit-rate of 512kbps. The audio samples then pass through a series of feature extraction and data augmentation processes as shown in Fig. 1.

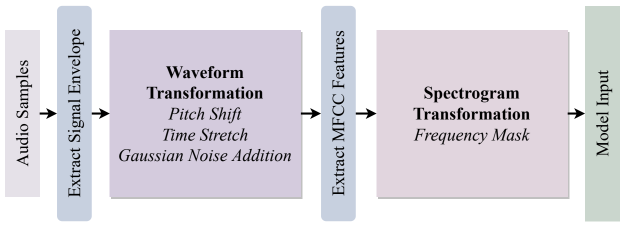


Fig. 1. Data Preprocessing Pipeline

1) Extract Sound Envelope: The envelope of a signal is defined as the boundary which encloses an audio signal. We have applied an envelope mask over the audio to remove regions of silence or "dead zones". This step extracts the relevant waveform from the signal and removes the aforementioned "dead zones" that are usually present at the start and end of the audio clips, as observed in Fig. 2(a) and 2(b).

2) Waveform Transformation: We have adopted the data augmentation techniques suggested for a low-resource speech recognition system. The augmentation techniques are transformations made to the audio data waveform to introduce different variations into the existing dataset - thus resulting in an increase in the size of dataset [26]. To do so, we have augmented the audio samples by performing a sequence of waveform transformations as follows:

- *Pitch Shift*: Male voices are generally low-pitched due to longer and thicker vocal cord folds while the opposite is true for females. Pitch shifting mimics different lengths and thicknesses of vocal folds to introduce variation in the dataset. Fig. 2(c) and 2(d) show the change in the waveform when the pitch of the same audio sample is decreased and increased, respectively.
- *Time Stretch*: Not everyone speaks at the same rate; some speak faster while others are slower. Taking this into consideration, we employ time stretching techniques. Fig. 2(e) and 2(f) display the difference when audio is manipulated to make speech faster and slower, respectively.
- *Add Gaussian Noise*: As observed by the increase in amplitude in the Fig. 2(g), Gaussian noise is introduced into the samples. This addition can help our model become more resilient to noise that is likely to be present in practical scenarios.

3) Extract MFCC features: While handling audio data, different speech feature extraction techniques can be adopted such as Linear Prediction Coefficients (LPC), Linear Prediction Cepstral Coefficients (LPCC), Discrete Wavelet Transform (DWT), Perceptual Linear Prediction (PLP), Line Spectral Frequencies (LSF) and Mel Frequency Cepstral Coefficients (MFCC). Investigating the comparison between different feature extraction techniques, we come to the conclusion that MFCC suits our case the best as it is modelled after the human auditory system. In addition, it also has high reliability and good enough noise-resistance which justifies its extensive usage in speech recognition systems [27].

Higher frequencies' signals are more challenging to differentiate for humans than the lower frequencies. Our perception might find two sounds to be identical despite being different. For example, to the human ear, the difference between low frequencies, e.g. 500Hz and 200Hz, is much more notable than the difference between high frequencies, e.g. 1500Hz and 1200Hz. Logarithmic transformation (11) of a signal's frequency mimics this human perception of sound. This transformation is called the Mel Scale [28].

$$m = 1127 \cdot \log\left(1 + \frac{f}{700}\right) \tag{11}$$

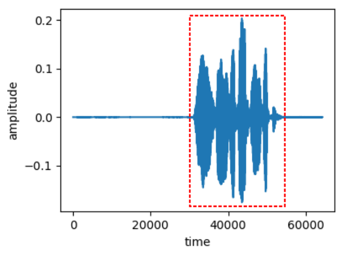
Mel Spectrograms visualise an audio signal to represent both its frequency-domain component as well as time-domain components. MFCC features are obtained by taking the logarithm of the Mel spectrogram, then performing Discrete Cosine Transform (DCT) (12). We concluded that 26 is the number of cepstrum deemed the best fit for our model through hyper-parameter tuning.

$$\begin{aligned} F(u,v) =& \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cdot \\ & \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right] \end{aligned} \tag{12}$$

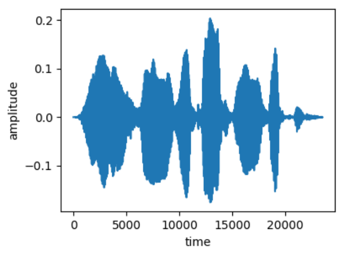
4) Spectrogram Transformation: For a final augmentation, we apply a Spectrogram Transformation technique to our MFCC features through a *frequency mask*. The Spectrogram Transformation technique named *SpecAugment* augments the spectrogram through masking blocks of frequency channels and time steps to improve the performance of speech recognition systems. In our system, we have only applied frequency masking and avoided time masking since the audio files' lengths range between 1s to 6s and taking away a fixed chunk from an already small audio might prove to be fruitless. Applying specAugment deliberately warps the audio data through partial loss of frequency information. This further makes our system robust to possible data deformations. By removing certain random frequency bands, our model learns to adapt to such loss in information. The use of a frequency mask allows to train a model to be more robust to frequency deformations [29]. The state of Mel Spectrogram before and after applying the frequency mask is respectively shown in Fig. 3(a) and 3(b). After these series of preprocessing steps, the input is finally fed into the model for classification.

B. Proposed Model: ***CSVC-Net***

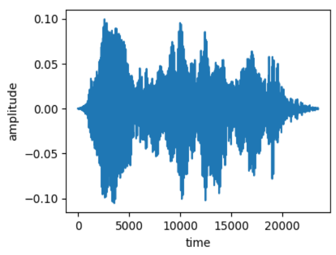
Our proposed architecture is a CNN-LSTM model. It consists of three main components - a CNN block, an LSTM



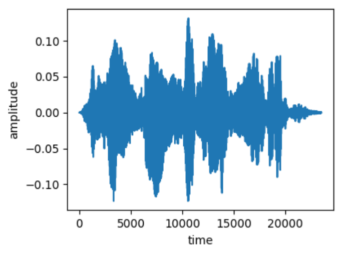
(a) Raw Audio (red box shows the envelope of audio signal without dead zones)



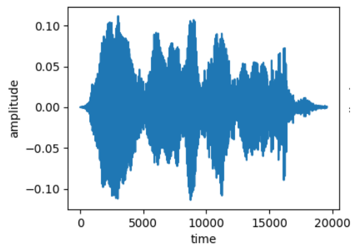
(b) Extracted Sound Envelope



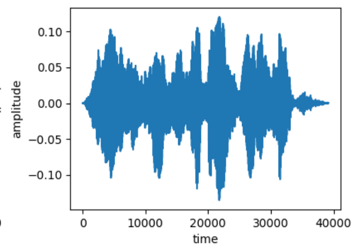
(c) Decreased Pitch



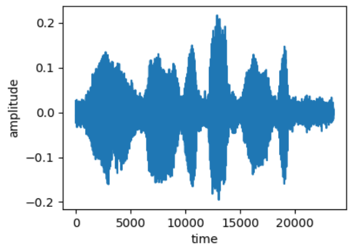
(d) Increased Pitch



(e) Sped Up

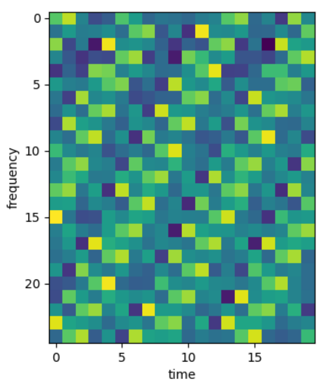


(f) Slowed Down

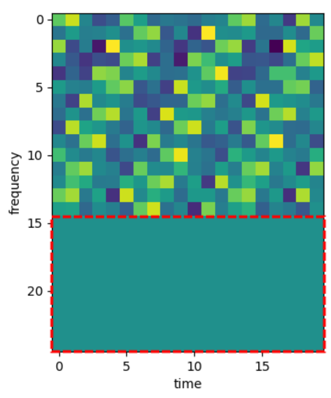


(g) Noise added

Fig. 2. (a) Original audio signal in time domain (milliseconds), (b) signal after extracting sound envelope and (c)-(g) signal after applying waveform transformations on the extracted envelope



(a) Initial Mel Spectrogram



(b) After applying Frequency mask (red box denotes the frequency band that was removed from the spectrogram)

Fig. 3. Mel Spectrogram Transformation using Frequency Mask

block and Time Distributed Dense Layers. To enhance the performance of the model, we also utilized Dropout and Batch Normalization layers in different parts.

1) CNN block: The CNN receives a $250 \times 26 \times 1$ MFCC feature vector as input from the preprocessing pipeline. It then proceeds to extract a feature map that is then passed on to the LSTM network. The CNN backbone consists of three two-dimensional convolutional layers. The three CNN layers contain 32, 64 and 128 filters of size 3×3 respectively. ReLU is used for activation to introduce non-linearity. The second and third CNN layers are followed by 2-dimensional Max Pooling layers of size 2×2. Pooling helps to reduce the learnable parameters and pass only the most important features to the succeeding feature map. The output from the CNN layer is of size $62 \times 6 \times 128$.

The use of CNN provides us three advantages - robustness against non-white noise, reduction of overfitting, handling of small frequency shifts. Local convolutions allow good features to be computed from the clean parts of the spectrum - thus reducing the effect of noise on features. The strong low-level features from earlier layers are forward propagated to the next layers, and ultimately, the high-level features computed in the deeper layers are more noise-robust [30]. Overfitting is reduced through weight-sharing in CNN. Not one but multiple frequency bands contribute to the learning of each weight. Furthermore, pooling allows our system to handle slight frequency shifts which occur frequently due to speakers having different vocal tract lengths.

2) LSTM Network: The feature map from the CNN is reshaped to 62×768 before being propagated to the LSTM network. This network comprises of four LSTM layers, each consisting of 128 cells. Hyperbolic tangent (*tanh*) is used for non-linear activation. The 'return sequence' parameter for the LSTM is set to true exposing both the hidden output state and the cell state to the LSTM layers. This also entails that the outputs from the LSTM layers are sequences rather than single vectors. The LSTM network can understand the semantic information of the audio stream that the traditional MFCC-CNN architecture is incapable of. In fact, it carries on and learns long-term dependencies. This helps our system retain information regardless of the audio file length. LSTM does this by propagating previous context on to the next step through its Cell State and Hidden State.

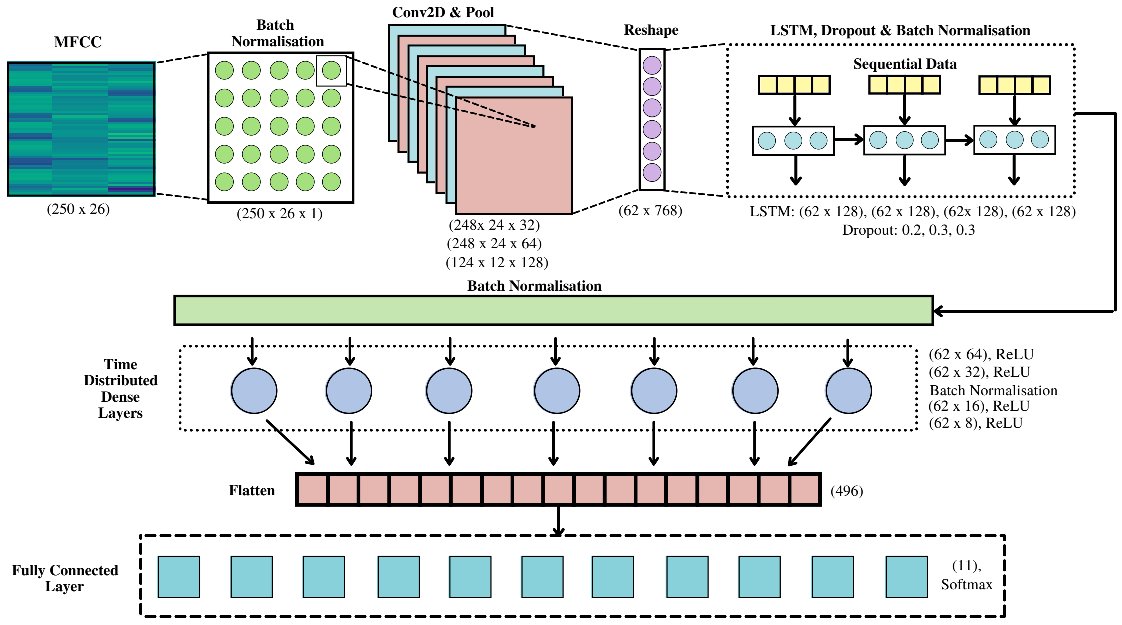


Fig. 4. Proposed Model Architecture: CSVC-Net

3) Time Distributed Dense Layers: The output from the LSTM network is then passed on to the Time Distributed Dense (TDD) layers. Our architecture consists of 4 TDD layers of sizes 64, 32, 16 and 8 that use ReLU activation. Since the LSTM network returns a sequence, the fully connected dense layers have to be applied to each step's input. The Time Distributed wrapper makes this possible, which applies the same weight to each temporal slice of input. The output from the TDD layer is flattened and passed on to a final fully connected layer for softmax classification.

4) Regularisation and Batch Normalization: To fully utilise our large dataset, it is essential to train the data on deep neural networks and make the model more susceptible to overfitting and optimisation issues. Awni Hannun *et al.* [31] suggest using Dropout as a regularisation technique, and Dario Amodei *et al.* [32] suggest using Batch Normalization to accelerate training and improve generalisation. Using Dropout, a neuron in the network has a certain probability of not being activated in the network, such that in that instance, it will not be connected to any other neurons. This enforces neuron independence and helps in generalisation. Batch Normalization transforms the data such that the mean is close to 0 and the standard deviation is close to 1. We apply three dropout layers of probabilities 0.2, 0.3, 0.3 after the second, third and fourth LSTM layers respectively. Four Batch Normalization layers are applied, one before the MFFC vector is sent to the CNN, two after the first and fourth LSTM layer and one after the second TDD layer.

V. Result and Discussion

A. Experimental Setup

The experiments were carried out on a local machine running Ubuntu 20.04.2 LTS that uses AMD Ryzen-5 3600 6-Core Processor with a base clock speed of 2.2kHz and total usable memory of 18 GB. The models were trained under a python environment (Python 3.8.5) and built using the Keras Deep Learning API. The models were trained for 70 epochs, adopting mini-batch gradient descent with a batch size of 20. Categorical cross-entropy was selected as the loss function, and Adam [33] was chosen as the optimization algorithm.

B. Creating the train and test set:

We split our original dataset into a train and a test set, maintaining a split ratio of 75:25. The train set consists of 2,944 samples obtained from 23 female and 23 male contributors. These samples then underwent various augmentation techniques, and the final train set amounted to 35,328 audio samples. 26,496 samples were reserved for training, and the remaining 8,832 samples made up the validation set.

The test set consists of 896 samples obtained from seven female and seven male contributors. The test set was further modified with varying degrees of Gaussian noise(δ) to produce additional test sets to evaluate the resilience of our model to noise (13). This resulted into four test sets ranging from noiseless to extremely noisy, termed L_0, L_1, L_2, and L_3. The original test set is L_0 with no added noise. L_1, L_2, and L_3 have added Gaussian noise of degree(δ) 0.0001, 0.0025 and 0.005 respectively.

$$GaussianNoise = \delta \times \mathcal{N}(\mu, \sigma) \tag{13}$$

C. Result Analysis

As illustrated in Table II, we empirically prove our hypothesis by comparing the efficacy of our proposed model against other methods. Our first motivation was to establish the rationale behind preferring MFCC over a simple Spectrogram.

TABLE II
COMPARISON OF DIFFERENT METHODS ON DIFFERENT NOISE LEVELS

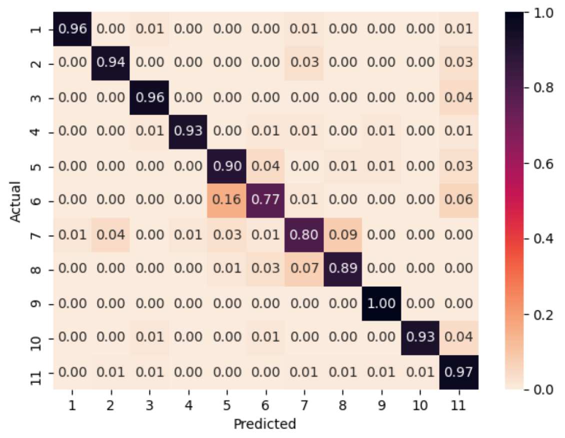
Feature Extraction	Classifier	Accuracy[a]			
		L_0	L_1	L_2	L_3
Spectrogram	CNN	16.07	15.62	-	-
MFCC	CNN	75.11	72.66	63.73	48.55
MFCC	GRU	89.73	87.39	74.00	43.53
MFCC	LSTM	91.52	89.29	81.03	55.47
MFCC	BiLSTM	90.62	89.73	86.61	66.29
MFCC	CSVC-Net	**92.08**	**92.41**	**87.50**	**72.66**

[a]L_0: Original dataset; L_1: Low to moderately noisy dataset; L_2: Moderate to highly noisy dataset; L_3: Extremely noisy dataset

To do so, in our first experiment, we converted our audio samples to a spectrogram and trained it on a CNN architecture. We achieved a poor accuracy of only 16.07% on the L_0 dataset and 15.62% on the L_1 dataset. Owing to the poor accuracy displayed, for our next experiment, we had replaced the spectrogram with MFCC for feature extraction and trained the dataset on the same CNN architecture. We noticed a drastic improvement in results which achieved 75.11%, 72.66%, 63.73% and 48.55% accuracy on L_0, L_1, L_2, and L_3. This demonstrates that the compressed representation of MFCC provides more meaningful features for audio classification.

However, the accuracy was still representative of the opinion that a CNN architecture is unable to pick up the semantics in a sentence. A more suitable model would be one that can process this data sequentially and understand the context, as enabled by RNNs. Hence, for our third, fourth and fifth experiment, we replaced the CNN architecture with RNN variants - GRU, LSTM and BiLSTM. They all achieved better accuracy compared to the CNN model. GRU, the simplest of the RNN variants achieved the lowest accuracy among the three of 89.73%, 87.39%, 74.00% and 43.53% on L_0, L_1, L_2, and L_3. While performing better than CNN by most metrics, it displayed less noise resilience compared to CNN on L_3. The LSTM and BiLSTM models showed comparable results. LSTM achieved a better accuracy of 91.52% on L_0 when compared to 90.62% accuracy of BiLSTM. Although BiLSTM displayed more noise resilience on L_1, L_2, and L_3, BiLSTM models are computationally more expensive to train and LSTM models provide a more lightweight solution. For our final experiment, we tested our proposed model CSVC-Net. It obtained the best accuracy on all metrics - 92.08%, 92.41%, 87.50% and 72.66% accuracy on L_0, L_1, L_2, and L_3 respectively. Our proposed model is not only more accurate but also more robust to noise. This outcome reflects our intuition of proposing a hybrid CNN-LSTM model. The CNN block provides resilience to noise, while the LSTM block can interpret the semantic information of the audio stream by exploiting the previous context.

From the confusion matrix in Fig. 5 we can observe that our model performs reasonably well in classifying most commands. Out of the eleven classes, the prediction accuracy for nine classes lies in the 90^{th} percentile or higher. This outcome is promising and asserts that our model can look past individual words and understand the semantics of the sentence as a whole. However, unsurprisingly, we can also observe that the model suffers from inaccuracy when identifying classes 5, 6, 7 and 8. It is unsurprising because, as mentioned earlier, classes 5 and 6 share minimal inter-class variation, with most commands differing by one or two words while at the same time displaying high intra-class variance. For example, “Window ta minimise koro” and “Jei Window ta khola ache oita niche namao” are both commands from class 5 but only have one keyword - “window” in common. At the same time, “Window ta minimise koro” and “Window ta maximise koro” are commands from classes 5 and 6, respectively, but differ only by a single word - “minimise/maximise” . The same applies to commands from class 7 and 8. Despite these challenges, our model can still predict with an average in the 80^{th} percentile when classifying these classes.



	1	2	3	4	5	6	7	8	9	10	11
1	0.96	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01
2	0.00	0.94	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.03
3	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
4	0.00	0.00	0.01	0.93	0.00	0.01	0.01	0.00	0.01	0.00	0.01
5	0.00	0.00	0.00	0.00	0.90	0.04	0.00	0.01	0.01	0.00	0.03
6	0.00	0.00	0.00	0.00	0.16	0.77	0.01	0.00	0.00	0.00	0.06
7	0.01	0.04	0.00	0.01	0.03	0.01	0.80	0.09	0.00	0.00	0.00
8	0.00	0.00	0.00	0.00	0.01	0.03	0.07	0.89	0.00	0.00	0.00
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
10	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.93	0.04
11	0.00	0.01	0.01	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.97

Fig. 5. Confusion Matrix of results from CSVC-Net on L_0

VI. CONCLUSION

The prevalence of code-switching displayed by most bilingual or multilingual speakers makes it challenging to develop viable speech classification systems. In order to do so, we need to build models that are resilient to these variances. We have presented a dataset containing code-switching between English and Bengali words and a complete pipeline for this classification. Our proposed model CSVC-Net can understand most of the semantic differences in sentences and predict with reasonable accuracy. Consequently, this model can be extended to classify multilingual commands beyond English and Bengali. However, the model may still fail to distinguish subtle variances, which leaves scope for future improvements. Incorporating different dialects of the same language introduces further intra-class variance which the model needs to be effective on. The performance of the model can be investigated using a larger dataset along with incorporating more noise-handling techniques. While, noise levels have been added externally to augment the dataset, it is not entirely reflective of a real-world scenario. Training and testing CSVC-Net on such real-world audio streams remains as a possible future work. We can further delve into a different architecture by substituting LSTM cells with a Transformer model. Introducing self-attention might enable our model to understand the audio semantics even more deeply. We sincerely hope that this work will be helpful to pave the way for fellow researchers in the days to come.

Acknowledgment

Curation of the dataset would not have been possible without all the contributors who took out their precious time for helping out with this research. We would like to express our gratitude to Adham Arik Rahman, Md. Mushfiqul Haque Omi, Md. Sabbir Rahman, Minhaz Kamal, Syed Mohammed Sartaj Ekram (alphabetical order) for their active participation and help with dataset acquisition and quality checking. Special thanks to Md. Bakhtiar Hasan, for all of the support and suggestions throughout this journey.

References

[1] H. Tabani, J.-M. Arnau, J. Tubella, and A. González, "Performance analysis and optimization of automatic speech recognition," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 4, pp. 847–860, 2017.

[2] S. Int., "What are the top 200 most spoken languages?" Available at https://www.ethnologue.com/guides/ethnologue200, 2021, accessed: 2021-02-27.

[3] S. Sitaram, K. R. Chandu, S. K. Rallabandi, and A. W. Black, "A survey of code-switched speech and language processing," *Computing Research Repository (CoRR)*, 2019.

[4] L. R. Rabiner and B. Juang, *Fundamentals of speech recognition*, ser. Prentice Hall signal processing series. Prentice Hall, 1993.

[5] J. Benesty, J. Chen, and Y. Huang, "Automatic speech recognition: a deep learning approach," 2008.

[6] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[7] A. K. Paul, D. Das, and M. M. Kamal, "Bangla speech recognition system using lpc and ann," in *2009 Seventh International Conference on Advances in Pattern Recognition*. IEEE, 2009, pp. 171–174.

[8] G. Muhammad, Y. A. Alotaibi, and M. N. Huda, "Automatic speech recognition for bangla digits," in *2009 12th International Conference on Computers and Information Technology*. IEEE, 2009, pp. 379–383.

[9] M. Shuvo, S. A. Shahriyar, and M. Akhand, "Bangla numeral recognition from speech signal using convolutional neural network," in *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*. IEEE, 2019, pp. 1–4.

[10] R. Sharmin, S. K. Rahut, and M. R. Huq, "Bengali spoken digit classification: A deep learning approach using convolutional neural network," *Procedia Computer Science*, vol. 171, pp. 1381–1388, 2020.

[11] S. A. Sumon, J. Chowdhury, S. Debnath, N. Mohammed, and S. Momen, "Bangla short speech commands recognition using convolutional neural networks," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*. IEEE, 2018, pp. 1–6.

[12] D. Gupta, E. Hossain, M. S. Hossain, K. Andersson, and S. Hossain, "A digital personal assistant using bangla voice command recognition and face detection," in *2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON)*. IEEE, 2019, pp. 116–121.

[13] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8619–8623.

[14] K. M. Knill, M. J. Gales, S. P. Rath, P. C. Woodland, C. Zhang, and S.-X. Zhang, "Investigation of multilingual deep neural networks for spoken term detection," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013, pp. 138–143.

[15] A. Ghoshal, P. Swietojanski, and S. Renals, "Multilingual training of deep neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 7319–7323.

[16] E. Yılmaz, H. van den Heuvel, and D. Van Leeuwen, "Investigating bilingual deep neural networks for automatic recognition of code-switching frisian speech," *Procedia Computer Science*, vol. 81, pp. 159–166, 2016.

[17] I. Hamed, M. Elmahdy, and S. Abdennadher, "Building a first language model for code-switch arabic-english," *Procedia Computer Science*, vol. 117, pp. 208–216, 2017, arabic Computational Linguistics.

[18] C. Shan, C. Weng, G. Wang, D. Su, M. Luo, D. Yu, and L. Xie, "Investigating end-to-end speech recognition for mandarin-english code-switching," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6056–6060.

[19] A. Pai, "Cnn vs. rnn vs. ann – analyzing 3 types of neural networks in deep learning," Available at https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/, 2020, accessed: 2021-04-29.

[20] L. R. Medsker and L. Jain, *Recurrent neural networks: Design and Applications*. CRC press, 2001, vol. 5.

[21] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, ser. JMLR Workshop and Conference Proceedings, vol. 28. JMLR.org, 2013, pp. 1310–1318.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlüter, and H. Ney, "A comprehensive study of deep bidirectional lstm rnns for acoustic modeling in speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2462–2466.

[24] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," pp. 1724–1734, 2014.

[25] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[26] J. Billa, "Improving lstm-ctc based asr performance in domains with limited training data," *Computing Research Repository (CoRR)*, vol. abs/1707.00722, 2017.

[27] S. A. Alim and N. K. A. Rashid, "Some commonly used speech feature extraction algorithms," in *Natural to Artificial Intelligence - Algorithms and Applications*, R. Lopez-Ruiz, Ed. London: IntechOpen, 2018.

[28] A. Sabra, "Learning from audio: The mel scale, mel spectrograms, and mel frequency cepstral coefficients," Available at https://towardsdatascience.com/learning-from-audio-the-mel-scale-mel-spectrograms-and-mel-frequency-cepstral-coefficients-f5752b6324a8, 2021, accessed: 2021-05-06.

[29] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," pp. 2613–2617, 2019.

[30] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[31] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep speech: Scaling up end-to-end speech recognition," *Computing Research Repository (CoRR)*, vol. abs/1412.5567, 2014.

[32] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*. PMLR, 2016, pp. 173–182.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2015.