Speech Recognition-based Bengali Learning App for Non-Bengali Speakers

Sajib Biswas Shuvo
Department of EEE
BUET
Dhaka, Bangladesh
1806131@eee.buet.ac.bd

Md. Samiul Islam Shadhin
Department of EEE
BUET
Dhaka, Bangladesh
1806151@eee.buet.ac.bd

Supta Deb
Department of EEE
BUET
Dhaka, Bangladesh
1806154@eee.buet.ac.bd

Tonmoy Hossain
Department of EEE
BUET
Dhaka, Bangladesh
1806136@eee.buet.ac.bd

***Abstract*—This paper contains the design approaches and algorithms used for the development of a "Bengali Learning App" for non-Bengali Speakers. We have divided everything in two stages: front-end and back-end. Details of the user interface are described in the front-end section while details of speech-recognition and other audio processing algorithms are described in the back-end section.**

Keywords—Bengali, audio, feature-extraction, CNN, bark-spectrum, user-interface, PySimpleGUI.

I. Introduction

Bengali is the 7th most spoken language by total number of speakers in the world [1]. However, to this date, popular language learning apps like Duolingo, Babbel, or Beelinguapp still do not have Bengali in their "List of languages to learn," which is unfortunate. That is why we have taken the approach to develop a software that will help non-Bengali speakers to learn Bengali in a self-paced and effective way. For now, our intended primary users are tourists who plan to visit Bangladesh. We have included some of the most used words and sentences that are necessary for daily communication with others.

The main feature of our app is the "Practice Session." After the user learns a few things from the app, a practice session is automatically initiated. The app shows one of the words (or Sentences) the user just learned and asks the user to pronounce and record it. Then our app uses speech recognition to decide whether the user pronounced it right or wrong. Based on this the user gets an idea about his/her learning progress and the effectivity of learning a new language is increased manyfold.

We have used python to develop the user interface and MATLAB to incorporate speech recognition. Prediction models developed in MATLAB are called from python.

II. Methodology

A. Front-End Development

PySimpleGUI, a python GUI library is used to develop the front-end of our software. Figure 1 is an screen-shot of the user interface. We have categorized the learning topics in Numbers, Letters, Words, and Sentences (shown on the left side of the window). After the user selects a category, the related topics are displayed on the right side of the window. For example, in Figure 1, we can see the word "শহর" which is the Bengali word for "City" is displayed on the top. Also, an image of a city is included in the center for easy comprehension.

When the user clicks the "Listen" button, the app pronounces the word (or Sentence) using Google's online Text to Speech Engine. The "Previous" and "Next" buttons are for browsing through the entire list of the selected category.

After the user learns two things, a practice session is initiated as shown in Figure 2. The user clicks on the "Record" button and tries to pronounce the word shown on the top. Then the app tells the user whether he/she was right or wrong. Besides, a scoring algorithm is also added which shows the overall accuracy of the user at the end of the session. The score is visible when the user clicks on the "Show Score" button.

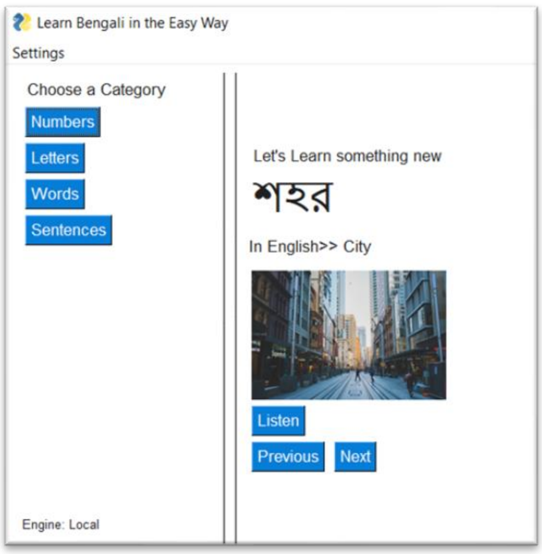


Figure 1: The User Interface

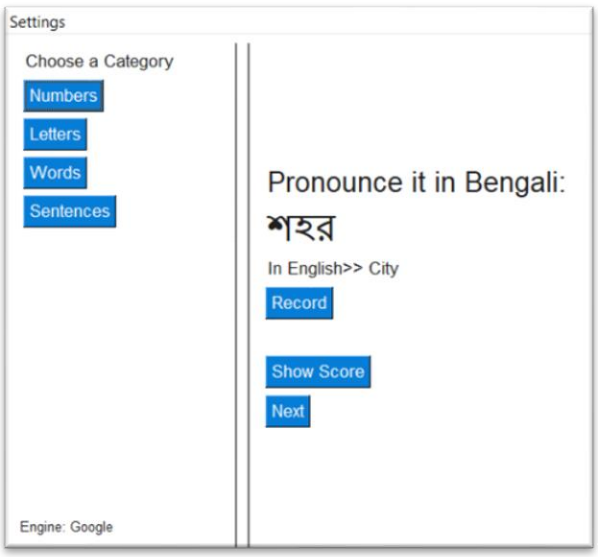


Figure 2: Practice Session

B. Back-end Development

When the user clicks on the record button, a MATLAB function to record and recognize the audio is invoked. The function works as shown in the flowchart.

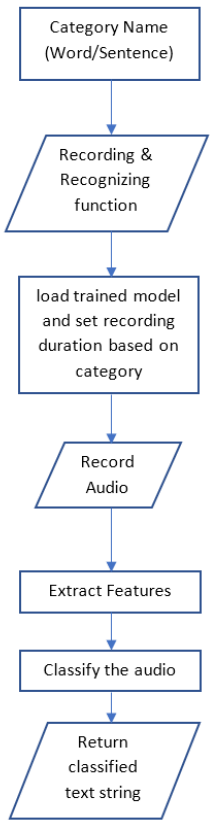

Category Name (Word/Sentence)
Recording & Recognizing function
load trained model and set recording duration based on category
Record Audio
Extract Features
Classify the audio
Return classified text string

Figure 3: Flowchart of the recognizing function

The pre-trained speech recognition model takes auditory-based spectrograms as inputs. So, we first convert the speech waveform to an auditory-based spectrogram. We use the MATLAB function audioFeatureExtractor to compute the auditory spectrogram. The feature extractor function's parameter are set such that it gives the bark spectrum coefficients as outputs.

Lastly the spectrum is sent to a pretrained voice recognition model. The model recognizes the voice and then sends the feedback. This feedback goes to the python interface and the GUI interface shows the result.

C. Training the Recognizer Model

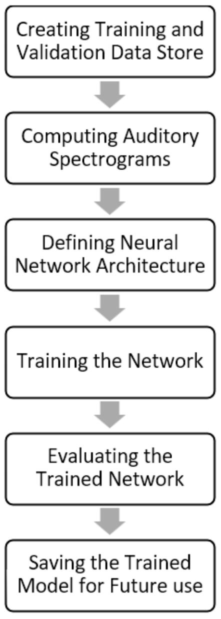

Creating Training and Validation Data Store
Computing Auditory Spectrograms
Defining Neural Network Architecture
Training the Network
Evaluating the Trained Network
Saving the Trained Model for Future use

Figure 4: Training Workflow

a. <u>Creating Training and Validation Data Store</u>:
At first, we created an audioDatastore object (from MATLAB Audio Toolbox) that points to the whole data set of a specific category (word/sentence). Then the dataset was split into training and validation dataset in a 70:30 ratio. The dataset distribution for different categories are shown in Figure 10 and Figure 11.

b. <u>Computing Auditory Spectrograms (Feature Extraction):</u> To prepare the data for efficient training of a Convolutional Neural Network, we converted the speech waveforms to auditory-based spectrograms. We defined the parameters of the feature extraction as follows:

- Sampling Frequency, Fs = 16KHz
- Window Type: Hanning
- Segment Duration: 2s
- Frame Duration = 0.025s
- Hop Duration = 0.010s
- FFT Length = 512
- Number of Bands = 50
- Size of spectrum coefficient matrix for an audio clip = 198 ×50

Here, Segment Duration is the duration of each speech clip (in seconds). Frame Duration is the duration of each frame for spectrum calculation. Hop Duration is the time step between each spectrum. Number of Bands is the number of filters in the auditory spectrogram. The extracted spectrum of three audio tracks are shown in Figure 6.

c. Defining the Neural Network Architecture: The structure of the Convolution Neural Network is as follows:

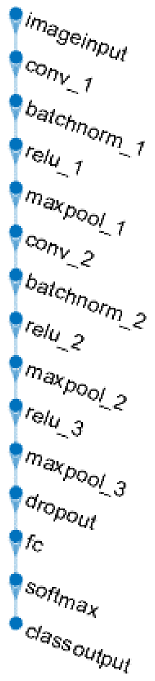


Figure 5: Neural Network Architecture for Sentence

d. Pre-detection audio Processing:
During live speech recognition, we first take the audio input via device microphone. For the case of digit recognition, we rescaled both the training and detection audio to the range [-0.8, 0.8]. Beside, we took the 10-based logarithm of the extracted feature to get a smoother distribution with a small offset.

III. Result analysis

The Training outputs are shown in Figure 7, Figure 8, and Figure 9. The confusion matrices obtained from training different categories are showed in Figure 12, Figure 13, and Figure 14.

Accuracies obtained while training different categories are as follows:

Table 1: Validation Accuracy

Category	Validation Accuracy
Digits	96.74%
Words	96.18%
Sentences	83.15%

Table 2: Comparison of Different Feature Extraction Techniques

Feature	Average Validation Accuracy
Bark Spectrum	90 %
Mel Spectrum	73

IV. Future Prospects

Currently, we live in the era of communication. Because of globalization, the entire planet has shrunk to the size of a single city. And there are so many people who speak Bengali that it ranks as the seventh most spoken language worldwide. The country of Bangladesh is a promising destination for vacationers. Our model can understand numerical, alphabetical, and linguistic input. You can use it as a tool to study Bangla. The absence of data is our project's biggest hurdle. We anticipate that the model's accuracy will increase if more varied data is introduced. A refined version of our prototype concept could serve as a platform for communication between Bengalis and those who are not from the region. We can create a version with the most used words and sentences, which could be useful as a means of communication for visitors. The potential of this methodology to teach Bangla to infants is enormous. Everything these days has a technological slant. They can be used as a tool for self-learning and to pique babies' interest with a sophisticated and fancy interface. This approach is widely embraced as a means of combining educational content with enjoyable activities. Illiterate people may be taught the basic essentials with this model. Teaching them at the elementary level will equip them to comprehend any writing.

V. Conclusion

We have successfully developed an app to recognize Bengali digits letters words and sentences. Our model's results are respectable. Google's search engine has consistently surpassed local engine in terms of recognition.
The model could be trained to recognize objects using a KNN, SVM, decision tree, or ensembling technique. In every case, the neural network model has been found to be superior. When given fewer data points, neural networks struggle. Consequently, the limitation of data and the lack of diversity in data samples pose the largest difficulty to training local engines. We need a large and versatile dataset to fully exploit the potential of this software.
Unfortunately, we have had some difficulty incorporating the Bangla typeface into MATLAB. The problem has been addressed via our Python interface. To be of practical use, this model can be enhanced.
Our goal is to get local engine performance worthy of note by providing it with a suitable dataset and a few tweaks.

VI. References

[1] "Bengali Language," 1 9 2022. [Online]. Available: https://en.wikipedia.org/wiki/Bengali_language#cite_ref-7.

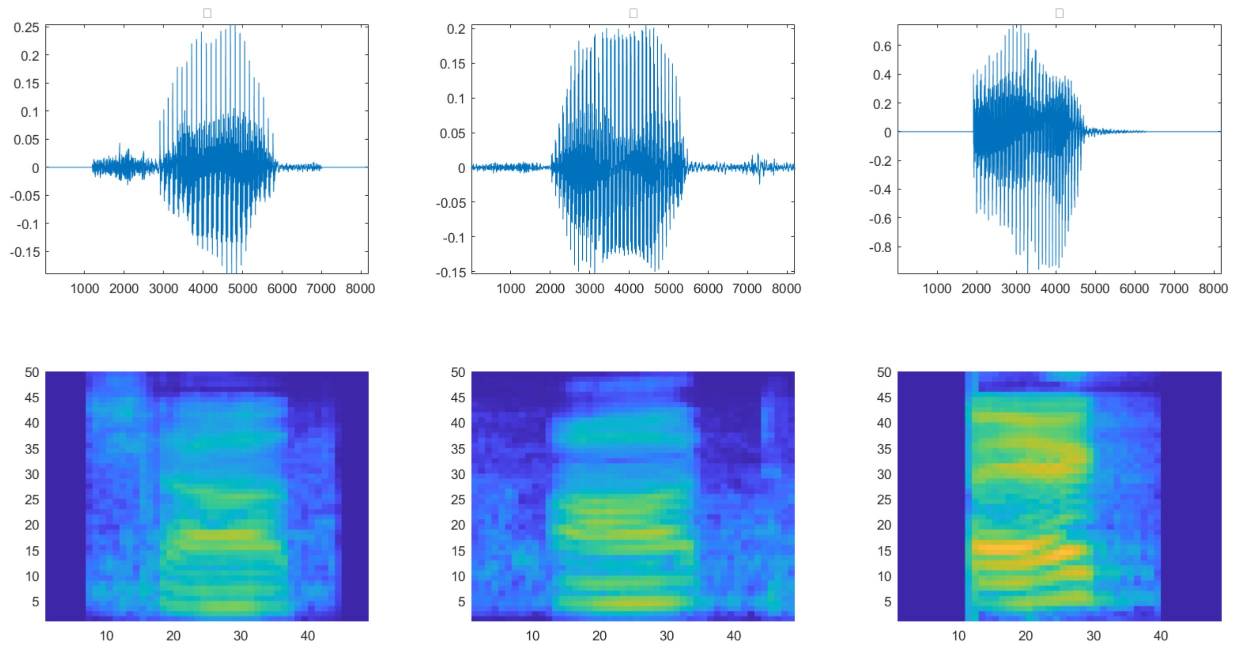
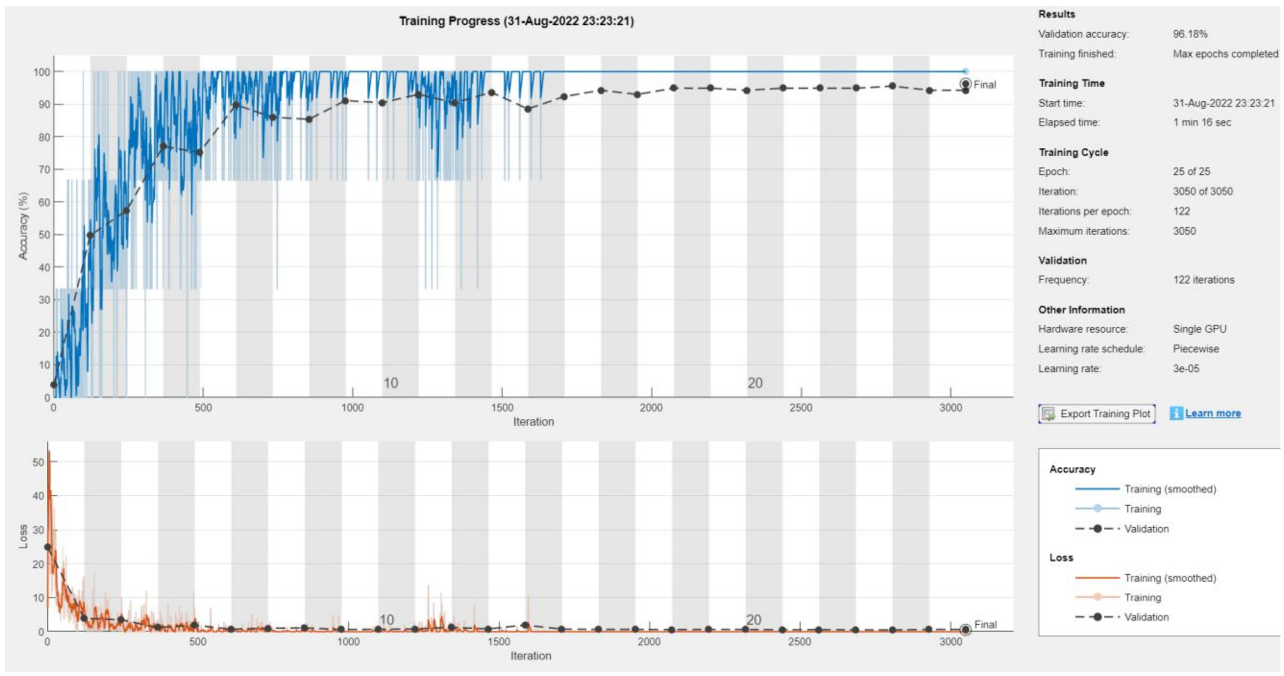


Figure 6: Feature Extraction and Spectrum Creation



Results

Validation accuracy:	96.18%
Training finished:	Max epochs completed

Training Time

Start time:	31-Aug-2022 23:23:21
Elapsed time:	1 min 16 sec

Training Cycle

Epoch:	25 of 25
Iteration:	3050 of 3050
Iterations per epoch:	122
Maximum iterations:	3050

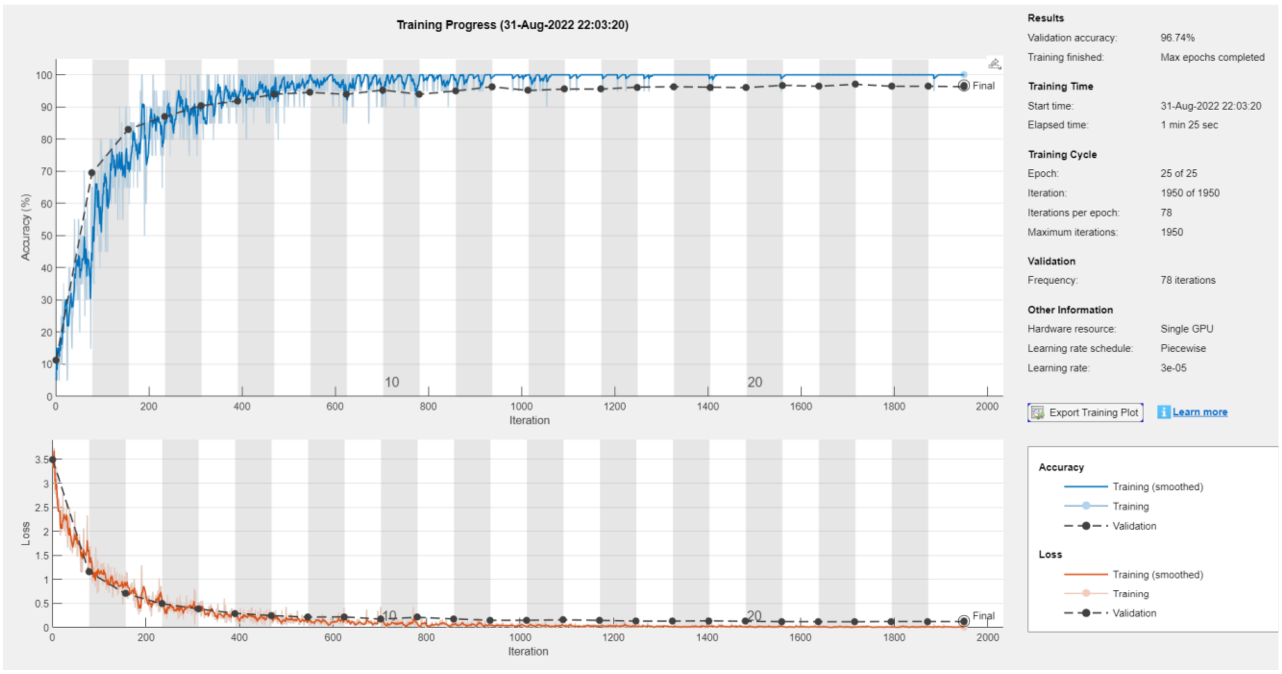
Validation

Frequency:	122 iterations

Other Information

Hardware resource:	Single GPU
Learning rate schedule:	Piecewise
Learning rate:	3e-05

Figure 7: Training Output for Words



Training Progress (31-Aug-2022 22:03:20)

Results

Validation accuracy:	96.74%
Training finished:	Max epochs completed

Training Time

Start time:	31-Aug-2022 22:03:20
Elapsed time:	1 min 25 sec

Training Cycle

Epoch:	25 of 25
Iteration:	1950 of 1950
Iterations per epoch:	78
Maximum iterations:	1950

Validation

Frequency:	78 iterations

Other Information

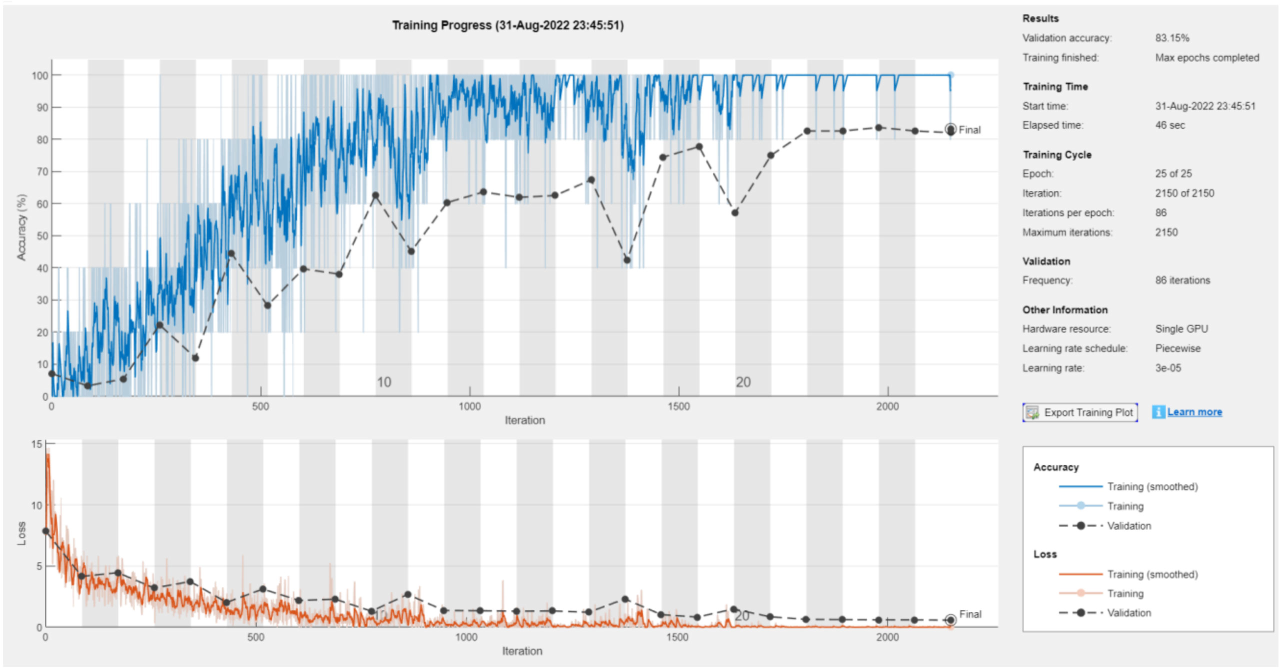
Hardware resource:	Single GPU
Learning rate schedule:	Piecewise
Learning rate:	3e-05

Export Training Plot   Learn more

Accuracy
- Training (smoothed)
- Training
- Validation

Loss
- Training (smoothed)
- Training
- Validation

Figure 8: Training Output for Digits



Training Progress (31-Aug-2022 23:45:51)

Results

Validation accuracy:	83.15%
Training finished:	Max epochs completed

Training Time

Start time:	31-Aug-2022 23:45:51
Elapsed time:	46 sec

Training Cycle

Epoch:	25 of 25
Iteration:	2150 of 2150
Iterations per epoch:	86
Maximum iterations:	2150

Validation

Frequency:	86 iterations

Other Information

Hardware resource:	Single GPU
Learning rate schedule:	Piecewise
Learning rate:	3e-05

Export Training Plot   Learn more

Accuracy
- Training (smoothed)
- Training
- Validation

Loss
- Training (smoothed)
- Training
- Validation

Figure 9: Training Output for Sentence

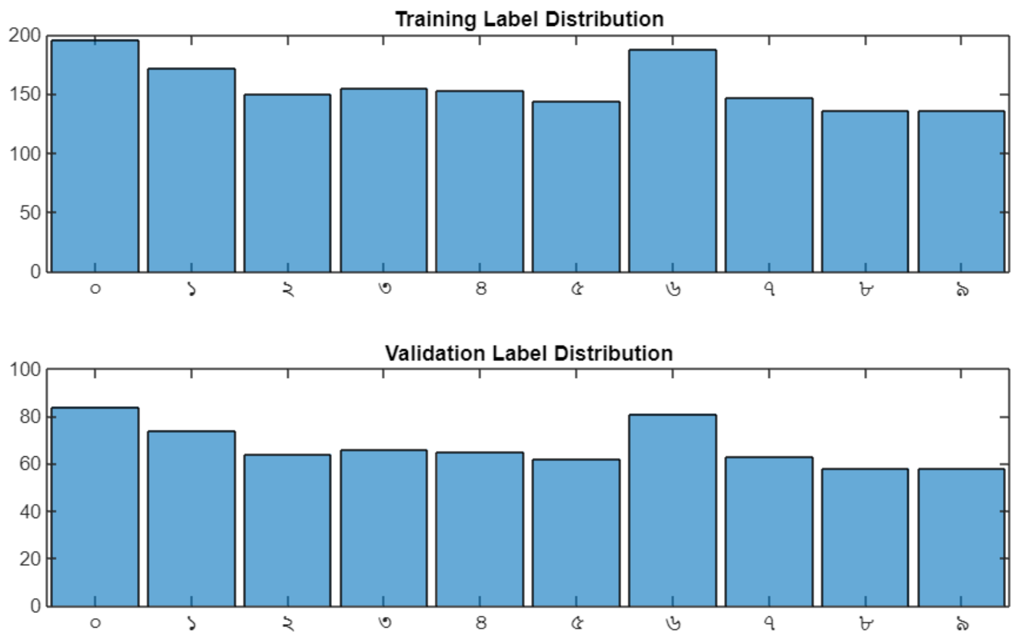


Figure 10: Training Validation Sample Distribution for Digits

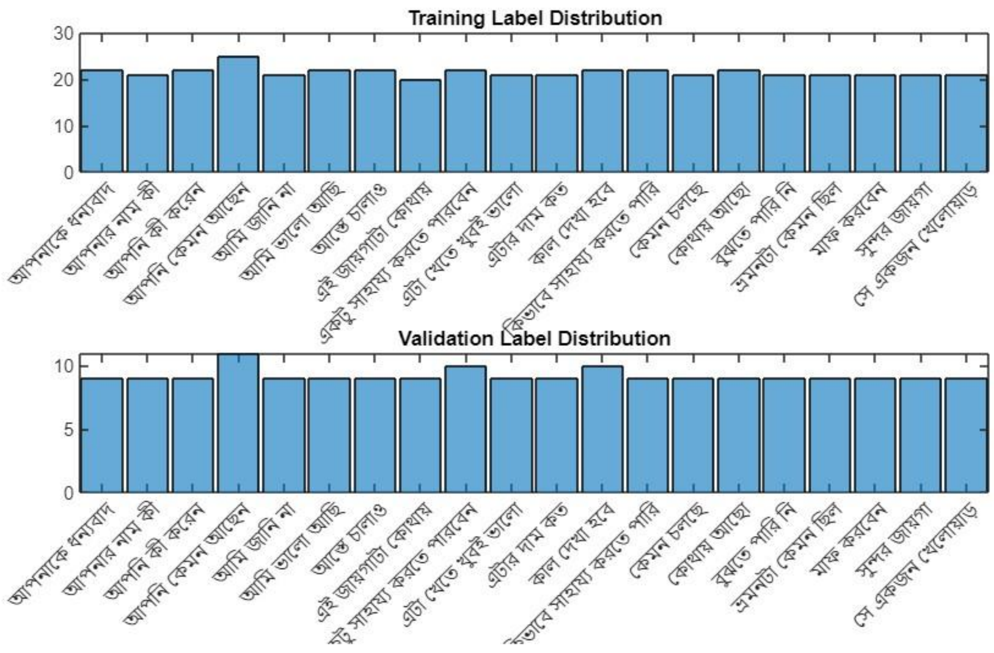


Figure 11: Feature Extraction and Spectrum Creation

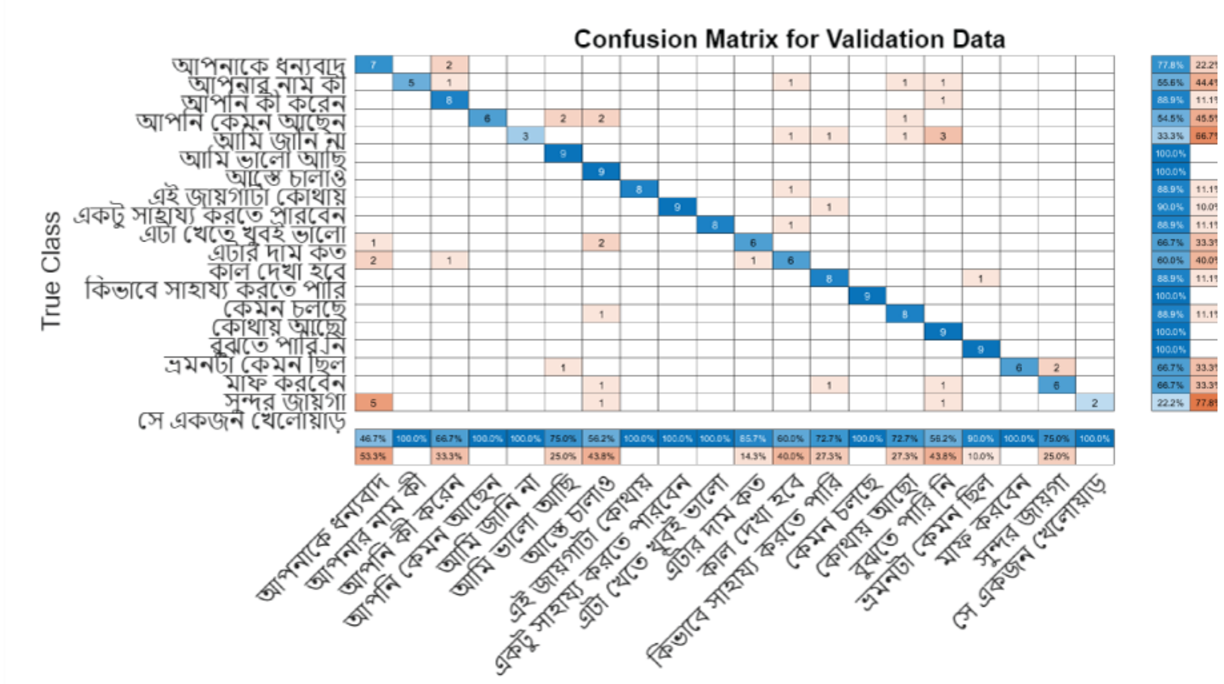


Figure 12: Confusion Matrix for Sentence Training

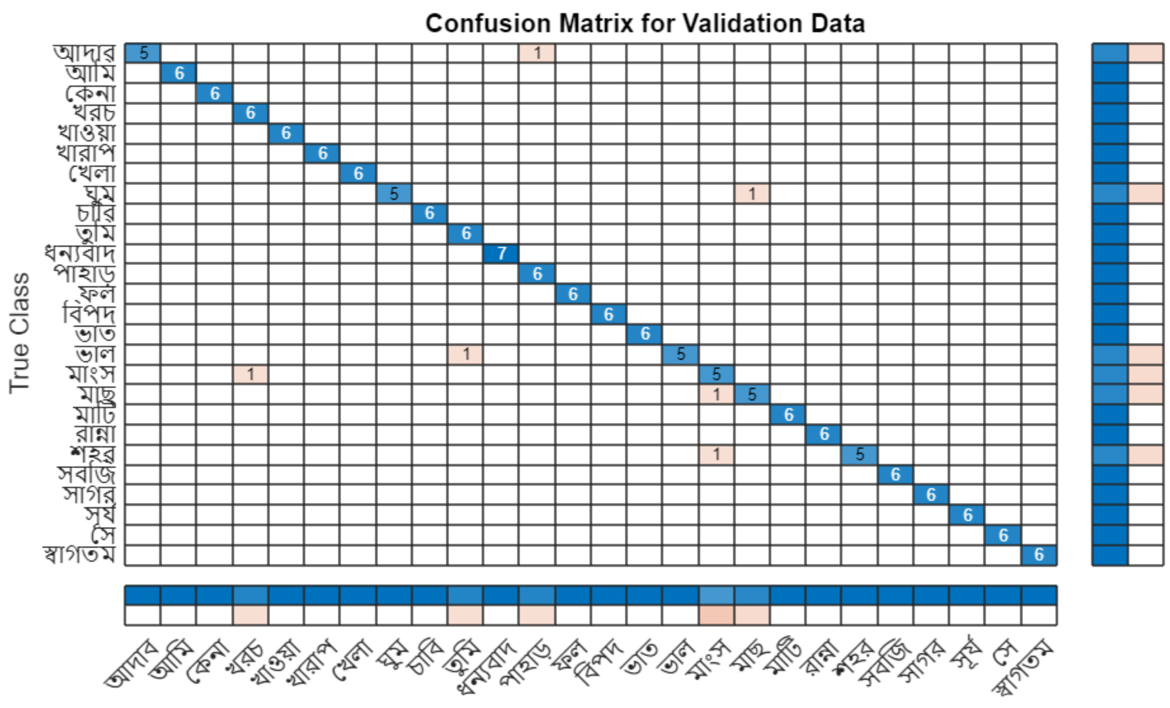
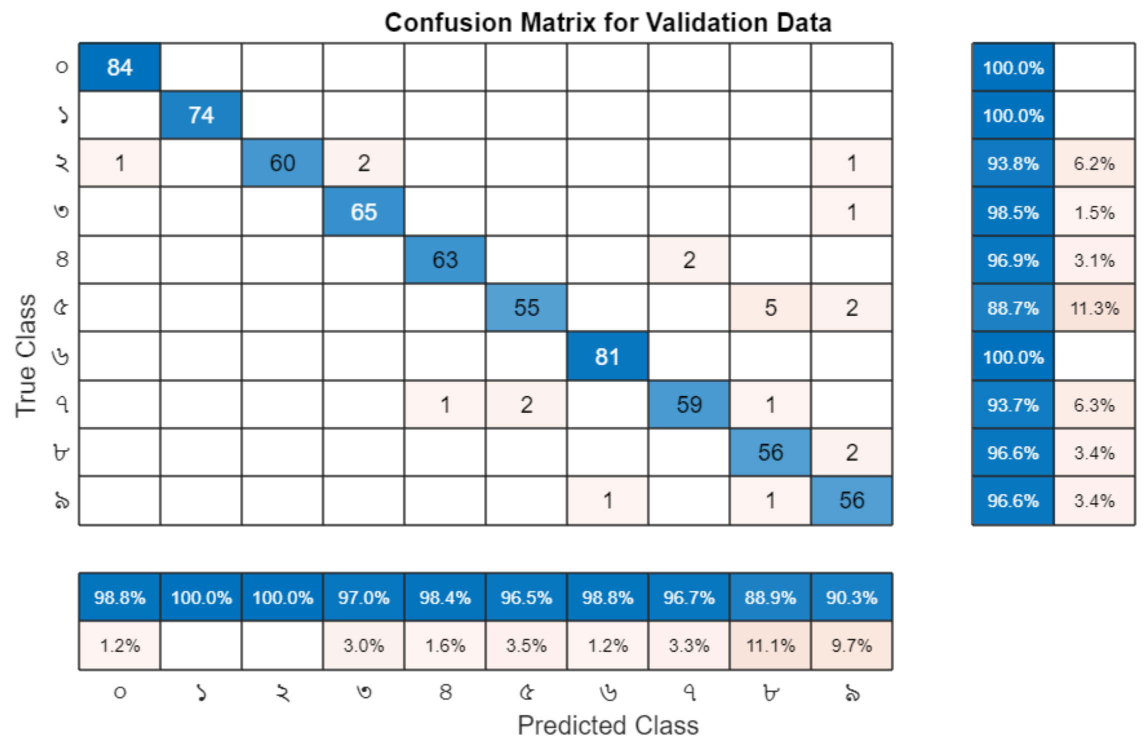


Figure 13: Confusion Matrix for Word Training



	০	১	২	৩	৪	৫	৬	৭	৮	৯		
০	84										100.0%	
১		74									100.0%	
২	1		60	2						1	93.8%	6.2%
৩				65						1	98.5%	1.5%
৪					63			2			96.9%	3.1%
৫						55			5	2	88.7%	11.3%
৬							81				100.0%	
৭					1	2		59	1		93.7%	6.3%
৮									56	2	96.6%	3.4%
৯							1		1	56	96.6%	3.4%
	98.8%	100.0%	100.0%	97.0%	98.4%	96.5%	98.8%	96.7%	88.9%	90.3%		
	1.2%			3.0%	1.6%	3.5%	1.2%	3.3%	11.1%	9.7%		

Figure 14: Confusion Matrix for Digit Training