Deep Learning Project 2 – Transformers

Report

Authors: Grzegorz Zbrzeżny, Izabela Telejko May 2024

Repository Link: https://github.com/izabelatelejko/2024L-Deep-Learning-P2-Transformers

Contents

1 Introduction

The aim of the project is to test and compare different network architectures for audio classification task specified in TensorFlow Speech Recognition Challenge[1]. The short audio clips are classified into 10 specific categories representing the spoken word or to "unknown" category in case of not matching the featured classes. What's more some of the test samples are silent, containing only small background noise. Therefore we approached this task in a 3-step manner by developing one model for silence detection (binary classification), one model for for distinguishing between the featured words and the "unknown" ones (also binary classification), and the last model for classifying audio clips within the featured classes. The process of generating prediction is shown in Figure 1. What's more we generated additional samples for silence samples with the use of data augmentation techniques. For the models, we considered GRU (Gated Recurrent Unit), Bidirectional LSTM (Long Short-Term Memory) and Transformer architectures, implemented using PyTorch library in Python.

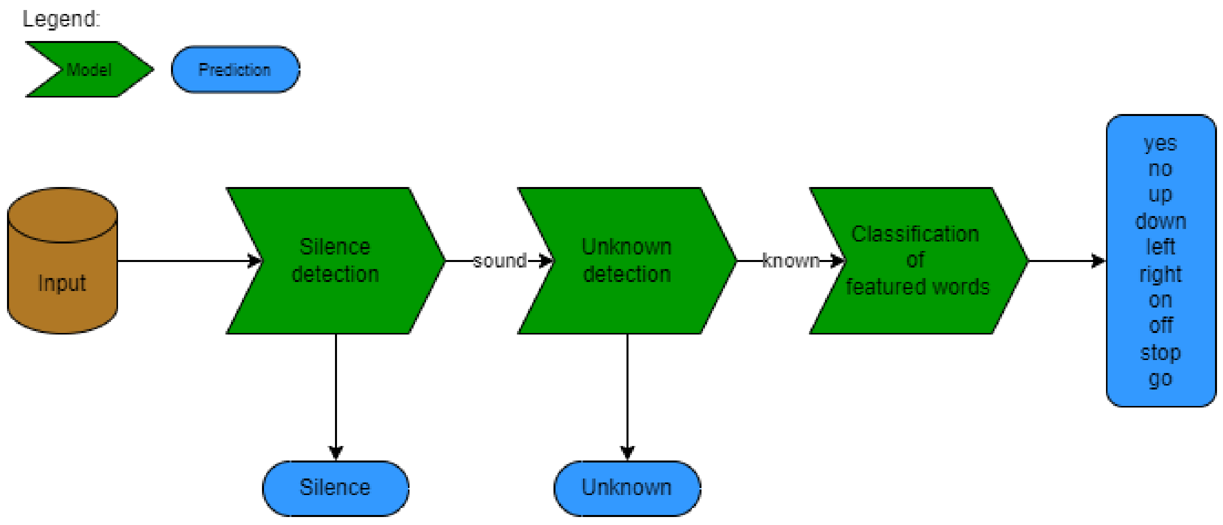


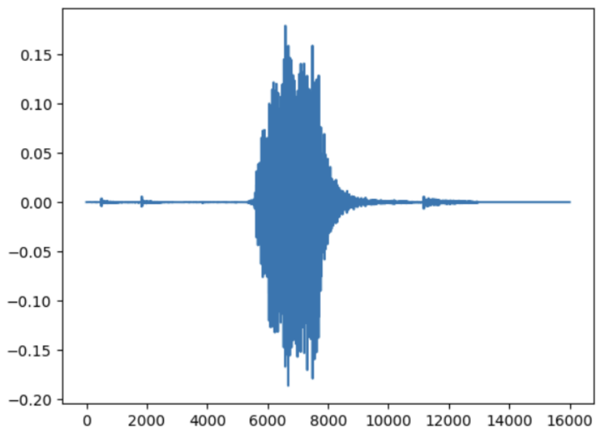
Figure 1: Task approach

2 Dataset

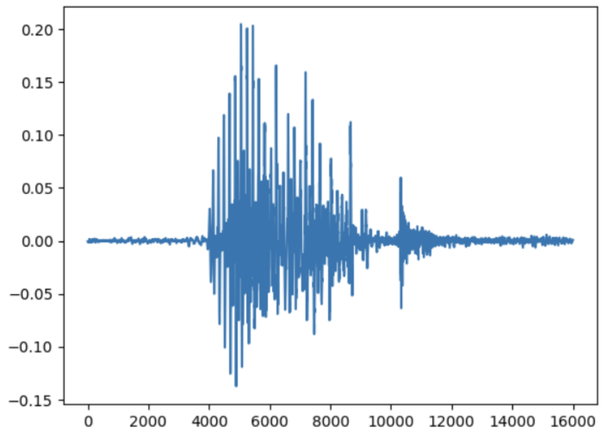
The dataset for this kaggle competition consists - Speech Commands Data Set v0.01[2] consists of one-second .wav audio files, each containing a single spoken English word. These words are from a small set of commands, and are spoken by a variety of different speakers. In the task 10 following words are featured and form a seperate classes: yes, no, up, down, left, right, on, off, stop, go (see audio samples plotted in Figure 2). Everything else should be classified as "unknown" if other word is spoken or "silence" in case when no word is spoken at all.

The dataset contains 64,727 audio files for training. In our solution we import this data directly from Hugging Face[3] and use the train-test-split dictated by this dataset. We evaluated the performance of our models using the test set provided in the competition, by collecting predictions from our models and submitting the results to the competition.

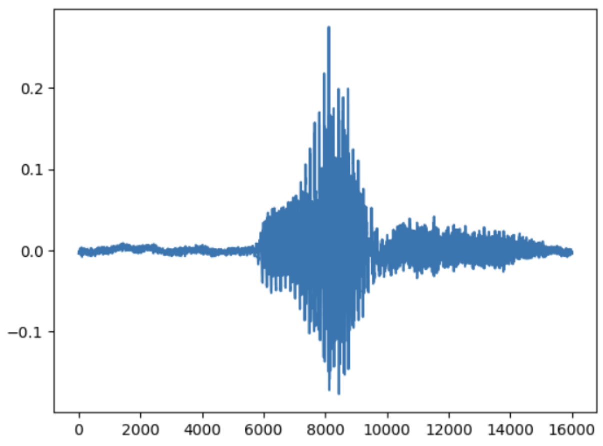
[1]https://www.kaggle.com/competitions/tensorflow-speech-recognition-challenge/overview
[2]https://www.kaggle.com/competitions/tensorflow-speech-recognition-challenge/data
[3]https://huggingface.co/datasets/speech_commands



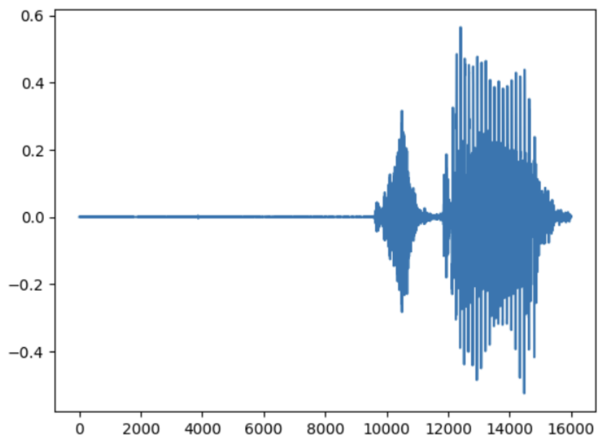
(a) up



(b) right



(c) yes



(d) stop

Figure 2: Audio samples plotted.

3 Network Architectures

In our experiments, we developed models based on three primary custom architectures with adjustable parameters (e.g. layer count or dropout rate). This flexibility allowed us to explore various iterations and select the most effective ones. At the end of each network we put an output fully connected layer with 1 neuron with sigmoid activation for binary classification task and with 10 neurons for main classification.

The first architecture, depicted in Figure 3, uses BI-LSTM (Bidirectional Long Short-Term Memory) layers. Initially, data served as the input for the BI-LSTM layer, capable of identifying dependencies in data sequences like audio. We opted for this layer over the standard LSTM version as it can traverse the data bidirectionally, capturing nuances from both ends. Following this, we incorporated a fully connected layer with batch normalization, dropout, and ReLU activation. This layer received the last output token from the BI-LSTM.

The second architecture shown in Figure 4 resembles the previous one, with the primary difference being the utilization of GRU (Gated Recurrent Unit) instead of BI-LSTM. Additionally, in few experiments we designed this network to be as compact as possible, so we directly connected the output from the GRU layer to the output fully connected layer.

The third architecture, illustrated in Figure 5, employs encoder part of transformers. While structurally akin to the previous architectures, this variant replaces the BI-LSTM/GRU with a Transformer Encoder block. Similarly, we took the last token value from Encoder and passed it to subsequent fully connected layer for processing.

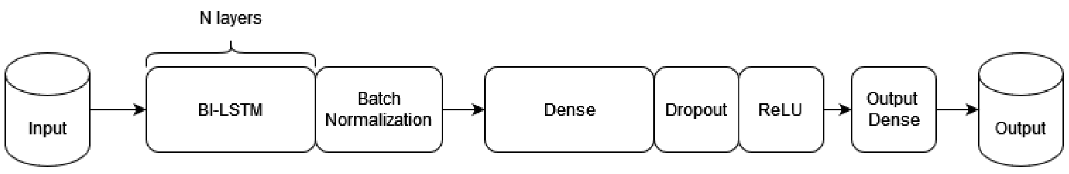


Figure 3: BI-LSTM architecture

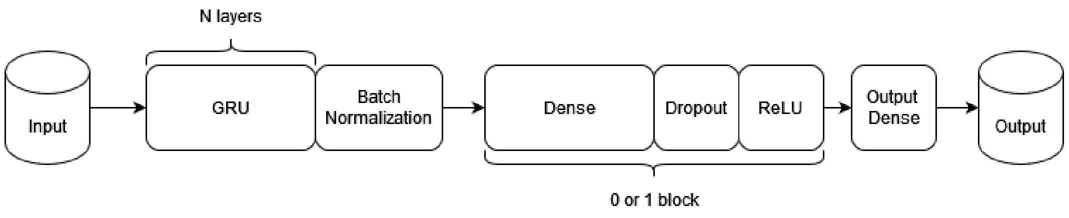


Figure 4: GRU architecture

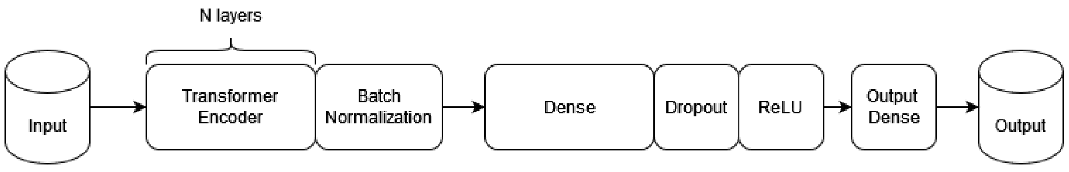


Figure 5: Transformer architecture

4 Data augmentation

We generated additional samples for silence clips by applying to the original samples exactly one of the listed below transformations:

- Addition of white noise,
- Addition of pink noise,
- Time stretching (slowing/speeding up the audio),
- Pitch shifting (changing the amplitudes),
- Shifting in time (the audio is shifted left or right).

We performed this process specifically on the silence set due to its relatively limited size, consisting of approximately 400 observations obtained by segmenting 1-second clips from silence samples. We generated 10 variations for each observation in the original silence dataset, each incorporating a random augmentation. Subsequently, we combined the original silence clips with the newly generated ones, resulting in a silence dataset comprising approximately 4400 samples. We also wanted to generate some more observations for the training dataset, but due to memory limitations we had to use only base versions of the clips during the training. In Figure 6 we can see the augmented samples from the training set (the augmented samples are plotted in pink color and the original ones in blue).

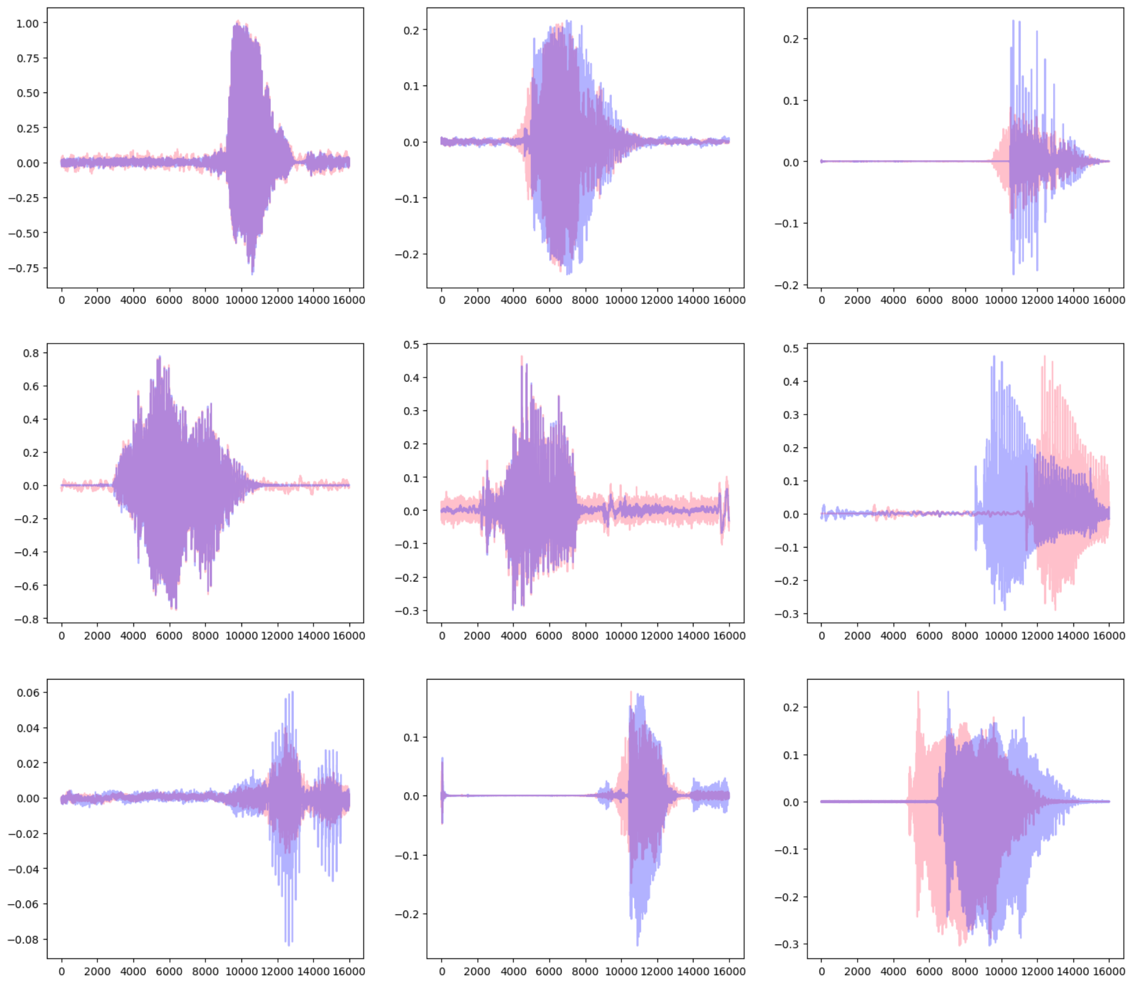


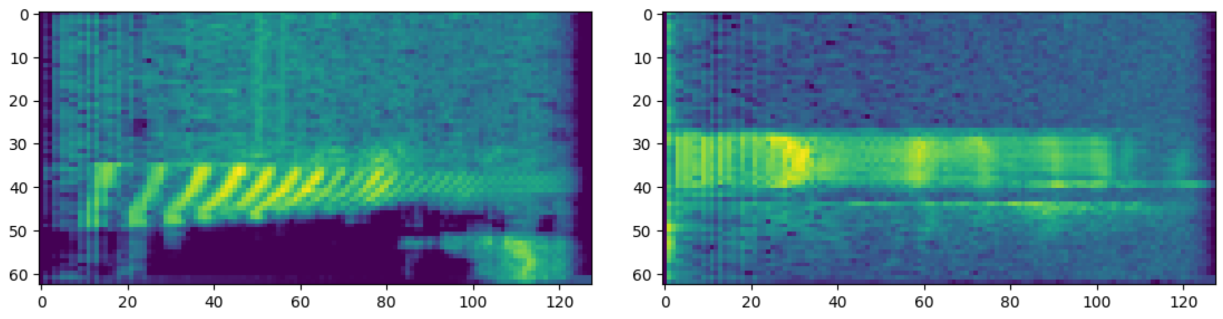
Figure 6: Augmented audio clips.

5 Preprocessing

We tested two quite popular approaches to handle preprocessing of audio data - transforming to spectograms and MFFCs. We compared the initial results on a simple model to choose one method of processing data for our models.

5.1 Spectograms

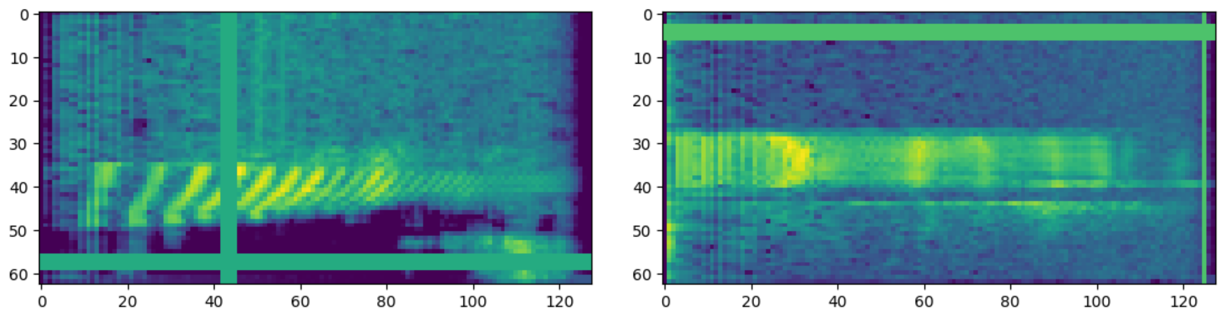
In the first approach each audio sample within the dataset undergoes transformation into a spectrogram representation. This conversion essentially breaks down the audio signal into its constituent frequencies over time, providing a detailed visual representation. Parameters such as the length of the Fast Fourier Transform (FFT), window size, and stride length are adjusted to optimize this process. Following this initial transformation, a technique known as mel-scaling is applied. This step recalibrates the spectrograms to better align with human auditory perception (human perception of sound intensity follows a logarithmic scale). In Figure 7 we can see some examples of generated spectograms from training samples.



(a) (b)

Figure 7: Sample spectograms.

To such spectograms some additional augmentation may be applied, by masking some frequencies ranges or time ranges (see examples in Figure 8).



(a) (b)

Figure 8: Sample masked spectograms.

5.2 MFCC

The second approach aims to transform each audio sample into MFCC representation. Mel-Frequency Cepstral Coefficients (MFCC) are derived from the power spectrum of an audio signal, capturing its spectral characteristics. They are generated by transforming the power spectrum into the mel-frequency scale and applying a series of mathematical operations including the discrete cosine transform (DCT). The sample MFCCs are plotted in Figure 9, at first glance they seem more chaotic than spectograms, however in our initial experiments performed much better than spectograms. Therefore we decided to use MFCCs in further model developing.

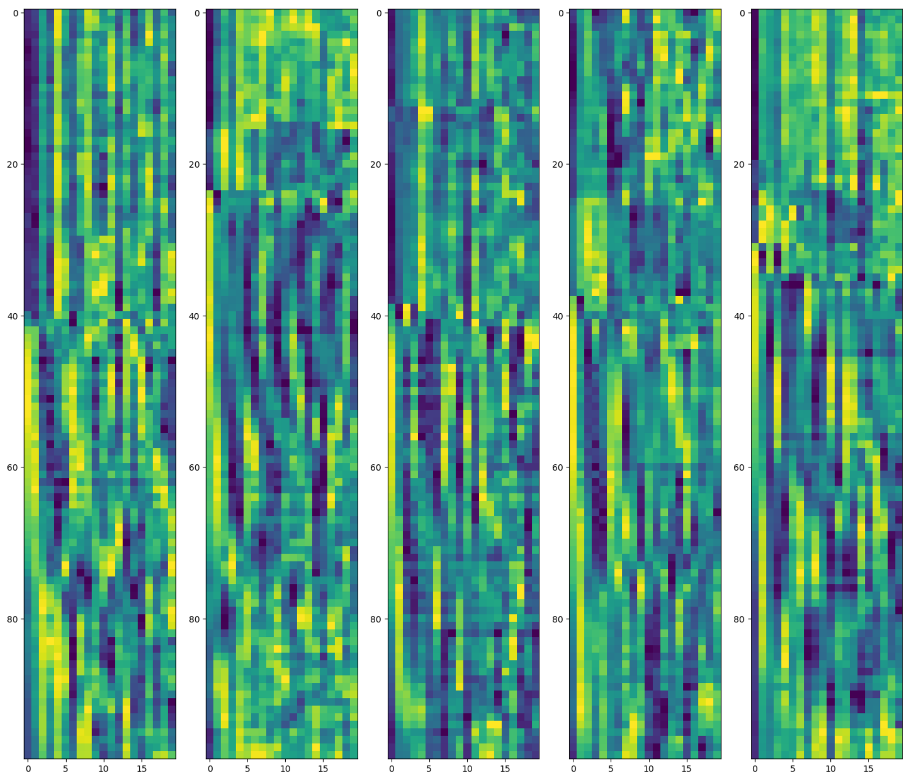


Figure 9: Sample MFCC plots.

6 Experiments

To tackle our prediction task effectively, we divided it into three distinct subproblems. The first subproblem focused on silence detection, where our goal was to develop models capable of identifying whether a given audio clip contains silence or some form of voice activity. The second subproblem, which we referred to as "Known vs. Unknown" involved distinguishing whether a given clip belonged to one of the main predefined classes or if it should be categorized as "unknown". Finally, the third subproblem entailed predicting the specific main class to which a given clip belonged. This task was approached assuming that any unknown or silent clips had been filtered out by the models from previous tasks. For models distinguishing we used the following prefix notation: **S** for silence detection task, **K** stands for "Known vs. Unknown" task, and finally **M** denotes the classification of main classes. For model validation purposes we split dataset into three subsets – training (51 088 samples), validation (6798 samples), and test (2824 samples), on which we will present the results in this section. Note that for part of the models we do not present training accuracy, because we do not have enough memory available to run the prediction on the whole training dataset.

6.1 Silence detection

To address the problem of silence detection, we created three models, each comprising one primary layer followed by two fully connected layers at the end, with dropout applied after the first of them. The primary layers employed were BI-LSTM, GRU, both with hidden size set of 32, or Transformer Encoder with 4 heads. Across all models, we maintained a dropout between 0.2 and 0.3. The size of the first fully connected layer were ranging from 16 to 64, as for this task a smaller architecture performed better. All configurations specific to each model are presented in Tables 1 - 2.

Throughout the training process, we employed binary cross-entropy loss, and the Adam optimizer, utilizing its default parameters: a learning rate of 0.001, betas set to (0.9, 0.999), and epsilon at 1e-08. Additionally, we applied a weight decay of 1e-5. To optimize efficiency, we implemented an early stopping rule, stopping training if there was no improvement in validation loss over three consecutive epochs.

To generate the training data, we segmented the samples from the "background noises" directory into 1-second clips. Subsequently, we augmented this dataset by creating 10 variations of each clip, resulting in a significantly larger training dataset comprising approximately 4500 samples. During training we used Adam Optimizer with 0.001 learning rate, Binary Cross Entropy loss function, and early stopping with patience set to 3 epochs. Moreover we performed train-validation split with 0.2 validation ratio.

In the following sections we present the results obtained by each model - accuracies on train and validation datasets, as well as plots of loss changes during training, and confusion matrices. Notably, this task posed minimal difficulty for our models, as each of them completed training within a dozen epochs, attaining high performance.

Model	Hidden size	Dropout	Num. layers	Fully conn. size
`S-GRU`	32	0.2	1	64
`S-BI-LSTM`	32	0.2	1	32

Table 1: GRU and BI-LSTM model configuration

Model	Heads count	Encoders count	Dropout	Fully conn. size
`S-Transformer`	4	1	0.3	16

Table 2: Transformer model configuration

6.1.1 Results of S-GRU

GRU model achieved results shown in Table 3.

Data	Accuracy
Train	0.987
Validation	0.985

Table 3: Results of S-GRU

Moreover we present loss changes during training (Figure 10) and confusion matrix for test data (Figure 11).

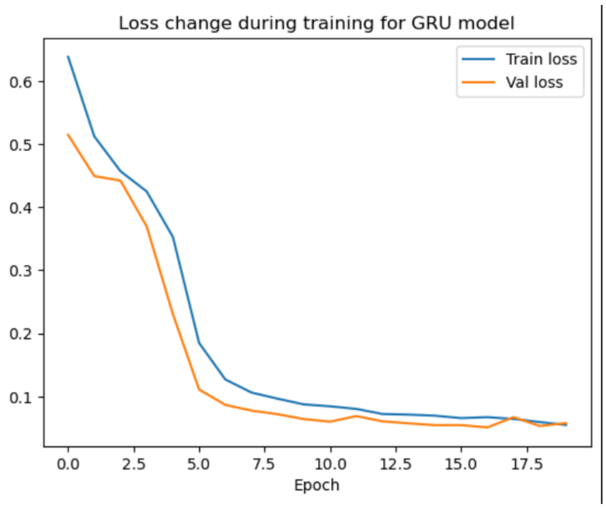


Figure 10: Loss changes during S-GRU training.

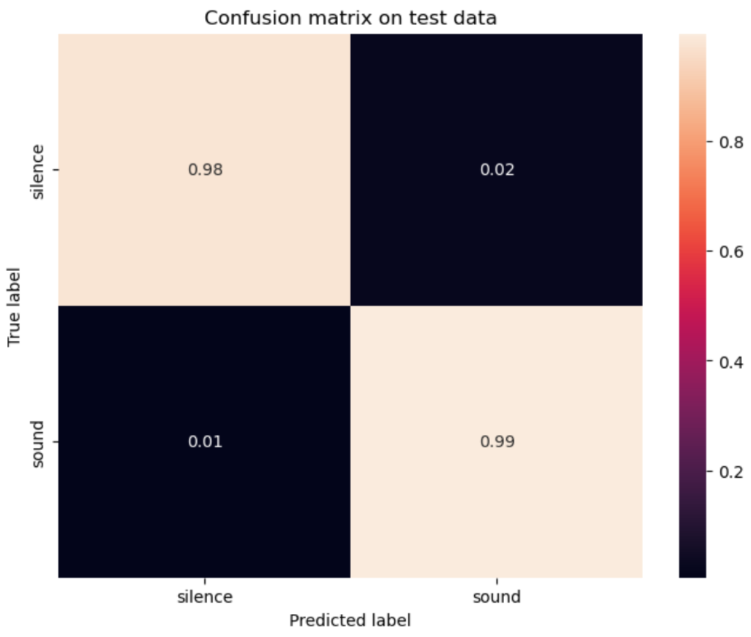


Figure 11: Confusion matrix for test data for S-GRU.

6.1.2 Results of S-BI-LSTM

BI-LSTM model achieved results shown in Table 4.

Data	Accuracy
Train	0.986
Validation	0.985

Table 4: Results of S-BI-LSTM

Moreover we present loss changes during training (Figure 12) and confusion matrix for test data (Figure 13).

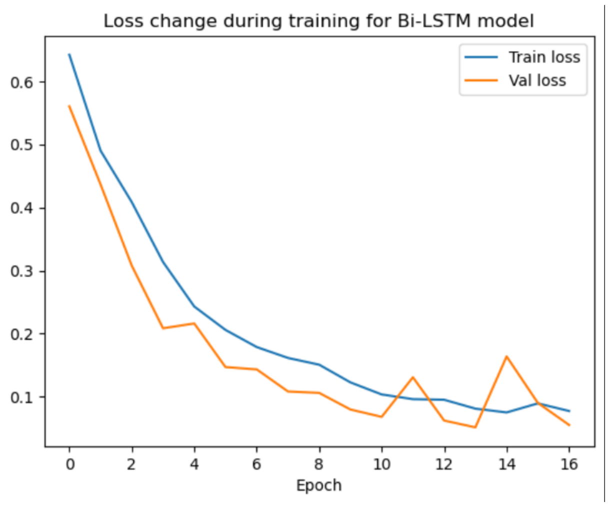


Figure 12: Loss changes during S-BI-LSTM training.

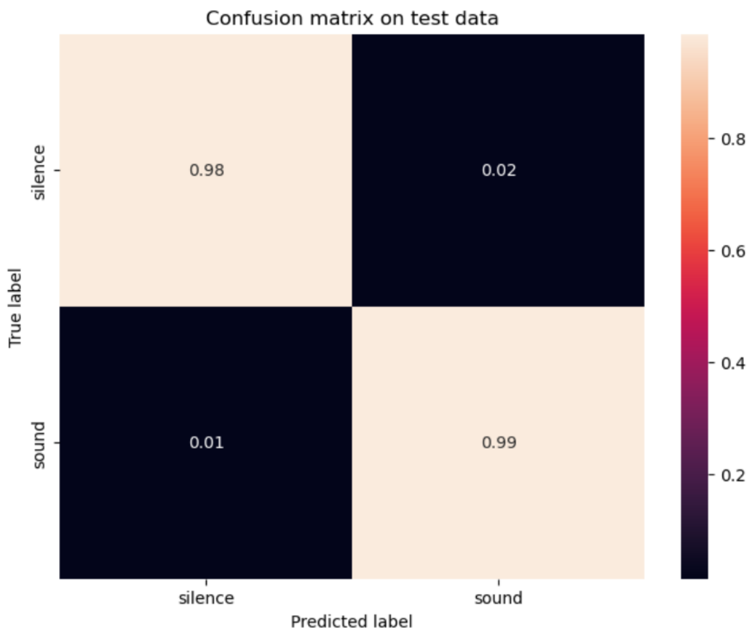


Figure 13: Confusion matrix for test data for S-BI-LSTM.

6.1.3 Results of S-Transformer

Transformer model achieved results shown in Table 5

Data	Accuracy
Train	0.996
Validation	0.983

Table 5: Results of S-Transformer

Moreover we present loss changes during training (Figure 14) and confusion matrix for test data (Figure 15).

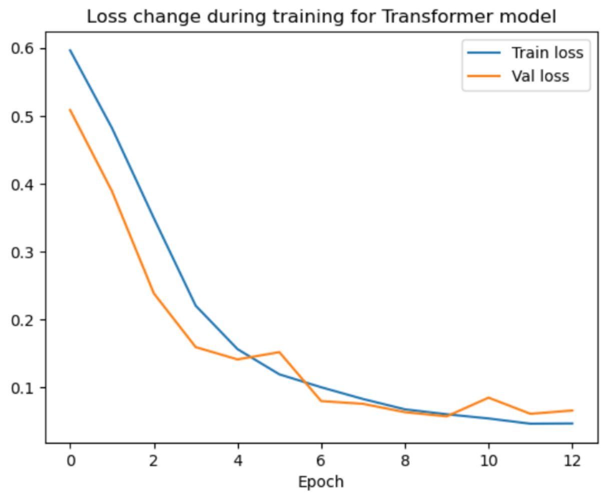


Figure 14: Loss changes during S-Transformer training.

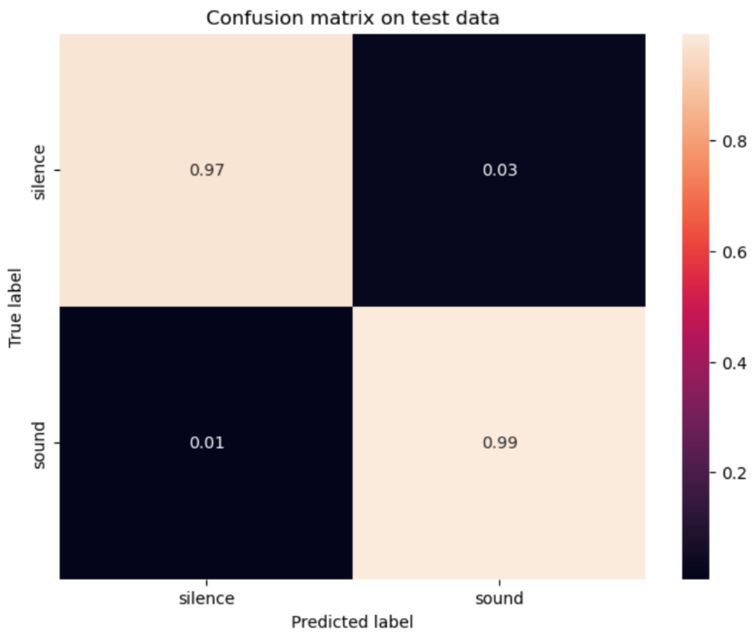


Figure 15: Confusion matrix for test data for S-Transformer.

6.2 Known vs. Unknown

To handle the detection of unknown words, we also created three models: GRU, BI-LSTM, and Transformer with 4 heads. However for this task the models had to be more complex to be able to recognize a variety of words that made up the "unknown" class. Therefore in each of these models we employed 4 primary layers, followed by two fully connected layers at the end, with dropout (0.3) applied after the first of them. We utilized 512 units in fully connected layer. For GRU we set the hidden size to 64, and for BI-LSTM to 32. All configurations specific to each model are presented in Tables 6 - 7.

Similarly as in previous task, during training, we employed binary cross-entropy loss, and the Adam optimizer with the same parameters. We also made use of the identical stopping rule with patience of 3 epochs.

In the following sections we present the results obtained by each model - accuracies on train, validation, and test datasets (please note that due to memory limitations train and validation accuracies are not always present), as well as plots of loss changes during training, and confusion matrices. Overall, our models obtained really good results in this task.

Model	Hidden size	Dropout	Num. layers	Fully conn. size
`K-GRU`	64	0.3	4	512
`K-BI-LSTM`	32	0.3	4	512

Table 6: GRU and BI-LSTM model configuration

Model	Heads count	Encoders count	Dropout	Fully conn. size
`K-Transformer`	4	4	0.3	512

Table 7: Transformer model configuration

6.2.1 Results of K-GRU

GRU model achieved following results shown in Table 8.

Data	Accuracy
Validation	0.96
Test	0.94

Table 8: Results of K-GRU

Moreover we present loss changes during training (Figure 16) and confusion matrix for test data (Figure 17).

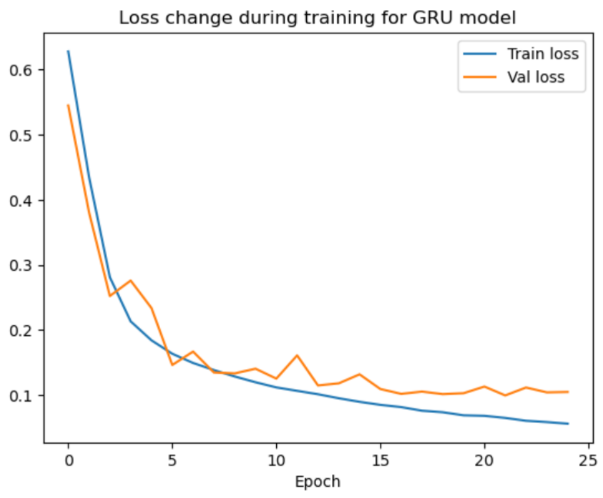


Figure 16: Loss changes during K-GRU training.

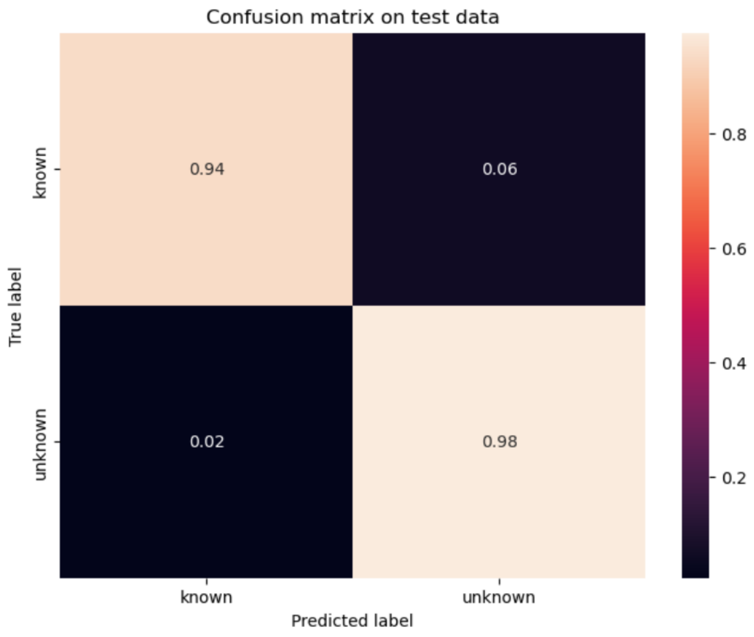


Figure 17: Confusion matrix for test data for K-GRU.

6.2.2 Results of K-BI-LSTM

BI-LSTM model achieved following results shown in Table 9.

Data	Accuracy
Validation	0.93
Test	0.95

Table 9: Results of K-BI-LSTM

Moreover we present loss changes during training (Figure 18) and confusion matrix for test data (Figure 19).

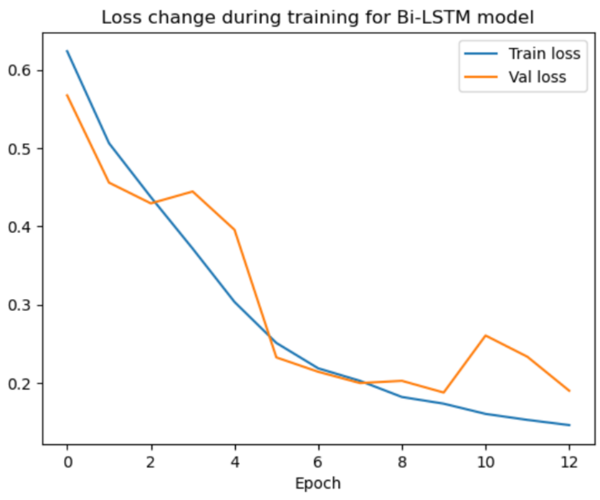


Figure 18: Loss changes during K-BI-LSTM training.

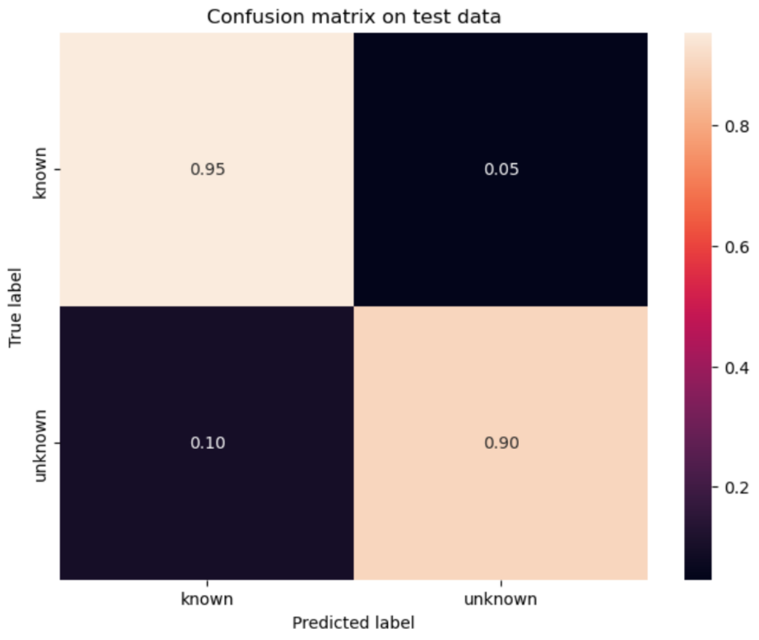


Figure 19: Confusion matrix for test data for K-BI-LSTM.

6.2.3 Results of K-Transformer

Transformer model achieved following results shown in Table 10.

Data	Accuracy
Test	0.89

Table 10: Results of K-Transformer

Moreover we present loss changes during training (Figure 20) and confusion matrix for test data (Figure 21).

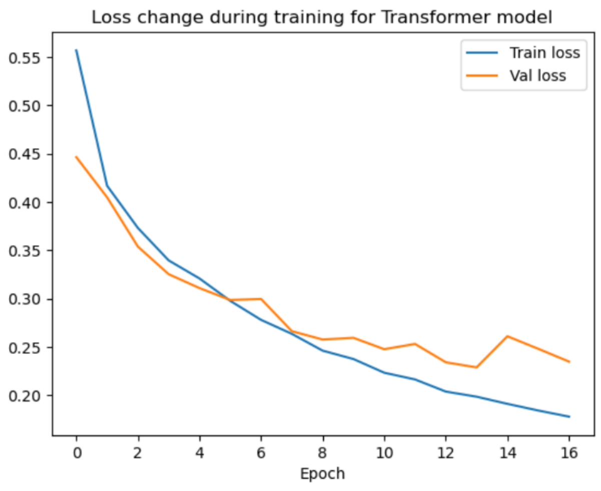


Figure 20: Loss changes during K-Transformer training.

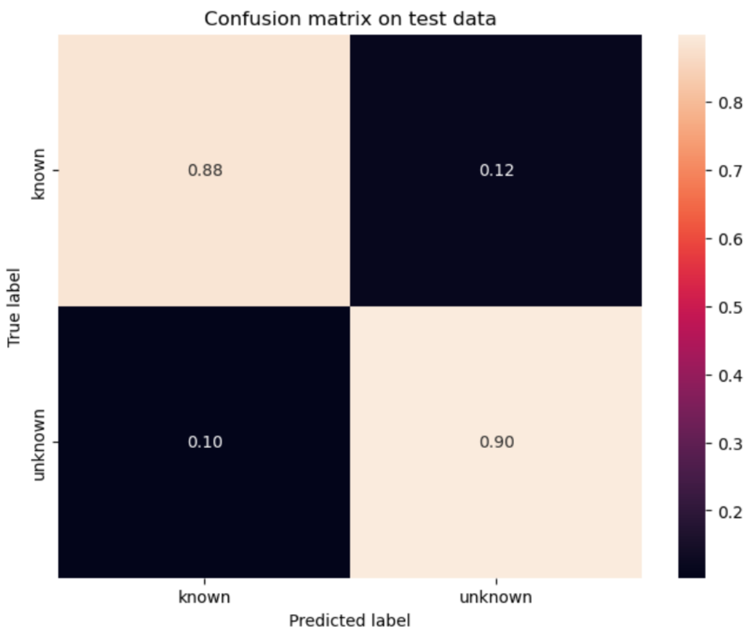


Figure 21: Confusion matrix for test data for K-Transformer.

6.3 Main class detection

In the last task in order to classify each input sample into one of the main categories, we explored different parameter values for each model type (BI-LSTM, GRU, Transformer) and selected configurations that yielded the most favorable results. Note that since the 0.001 learning rate (default for ADAM optimizer used by us) yielded best results almost every time during our initial experiments, we decided that we would use this value for every final model. For the loss function we chose Cross Entropy and we utilized early stopping with patience set to 10 for transformers and 5 for other models. In the Table 11, Table 12, and Table 13 we put our best models along with their parameters and later we show their results.

Model	Hidden size	Dropout	Num. BI-LSTM layers	Fully connected size
M-BI-LSTM-1	32	0.3	4	512
M-BI-LSTM-2	32	0.5	3	32

Table 11: BI-LSTM models configuration

Model	Hidden size	Dropout	Num. GRU layers
M-GRU-1	32	0.5	1
M-GRU-2	64	0.5	4

Table 12: GRU models configuration

Model	Heads count	Encoders count	Dropout	Fully conn. size
M-Transformer-1	4	4	0.4	64
M-Transformer-2	4	5	0.3	512

Table 13: Transformer models configuration

6.3.1 Results of M-BI-LSTM-1

BI-LSTM model with configuration 1 achieved following results shown in Table 14.

Data	Accuracy
Train	0.97
Validation	0.92
Test	0.92

Table 14: Results of M-BI-LSTM-1

Moreover we present loss changes during training (Figure 22) and confusion matrix for test data (Figure 23).

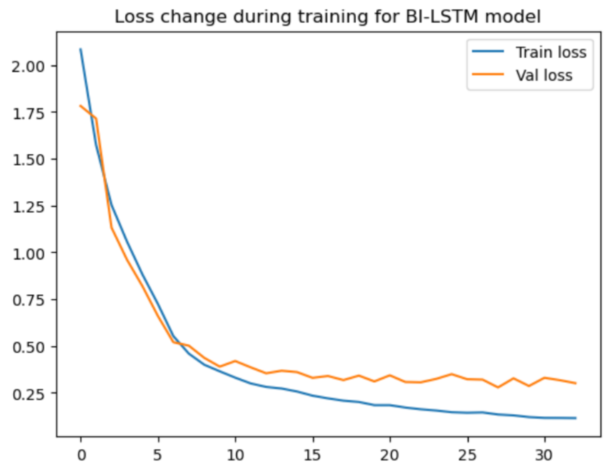
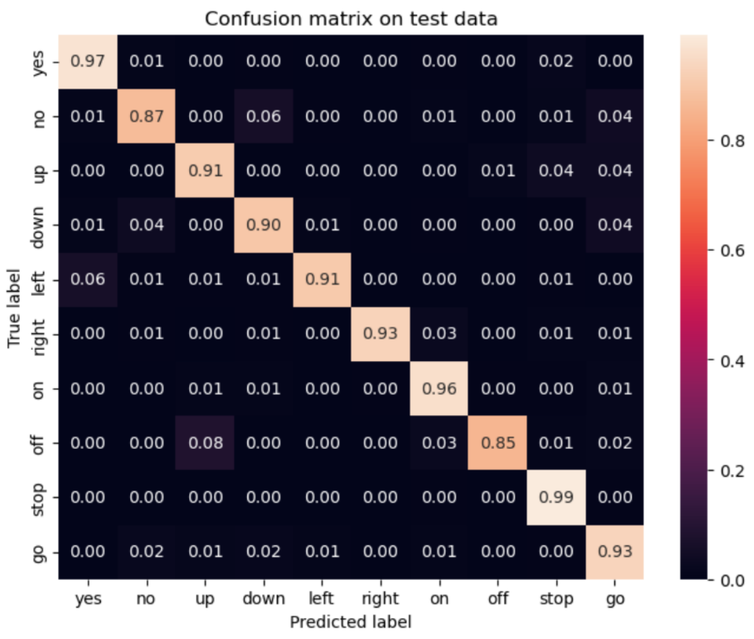


Figure 22: Loss changes during BI-LSTM-1 training.



	yes	no	up	down	left	right	on	off	stop	go
yes	0.97	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00
no	0.01	0.87	0.00	0.06	0.00	0.00	0.01	0.00	0.01	0.04
up	0.00	0.00	0.91	0.00	0.00	0.00	0.00	0.01	0.04	0.04
down	0.01	0.04	0.00	0.90	0.01	0.00	0.00	0.00	0.00	0.04
left	0.06	0.01	0.01	0.01	0.91	0.00	0.00	0.00	0.01	0.00
right	0.00	0.01	0.00	0.01	0.00	0.93	0.03	0.00	0.01	0.01
on	0.00	0.00	0.01	0.01	0.00	0.00	0.96	0.00	0.00	0.01
off	0.00	0.00	0.08	0.00	0.00	0.00	0.03	0.85	0.01	0.02
stop	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00
go	0.00	0.02	0.01	0.02	0.01	0.00	0.01	0.00	0.00	0.93

Figure 23: Confusion matrix for test data for BI-LSTM-1.

6.3.2 Results of M-BI-LSTM-2

BI-LSTM model with configuration 2 achieved results shown in Table 15.

Data	Accuracy
Train	0.94
Validation	0.90
Test	0.90

Table 15: Results of M-BI-LSTM-2

Moreover we present loss changes during training (Figure 24) and confusion matrix for test data (Figure 25).

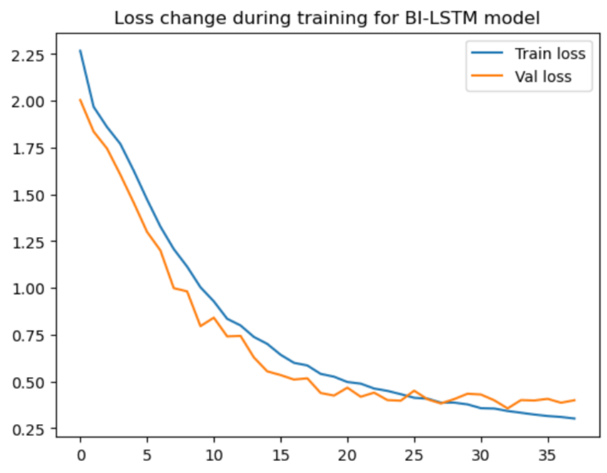


Figure 24: Loss changes during BI-LSTM-2 training.

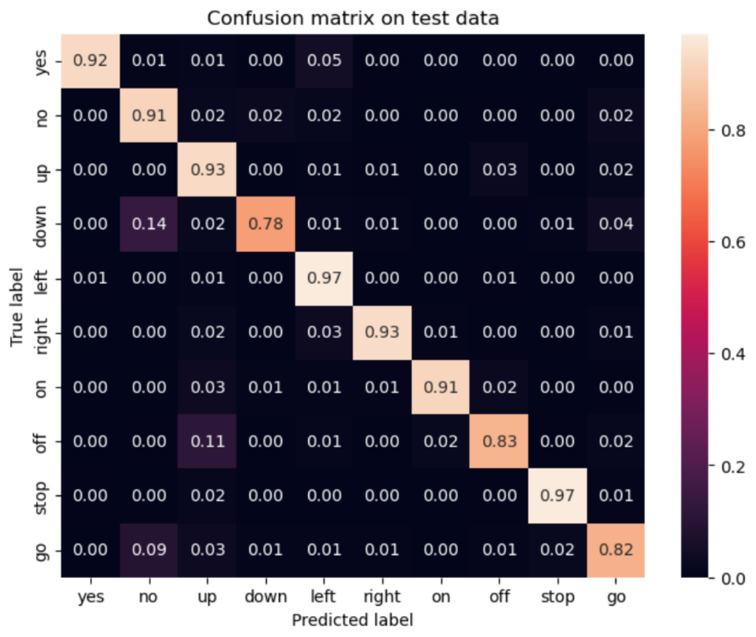


Figure 25: Confusion matrix for test data for BI-LSTM-2.

6.3.3 Results of M-GRU-1

GRU model with configuration 1 achieved results shown in Table 16.

Data	Accuracy
Train	0.92
Validation	0.89
Test	0.88

Table 16: Results of M-GRU-1

Moreover we present loss changes during training (Figure 26) and confusion matrix for test data (Figure 27).

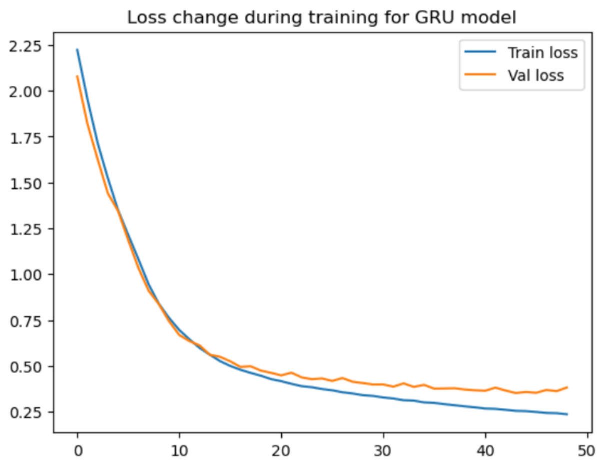


Figure 26: Loss changes during GRU-1 training.

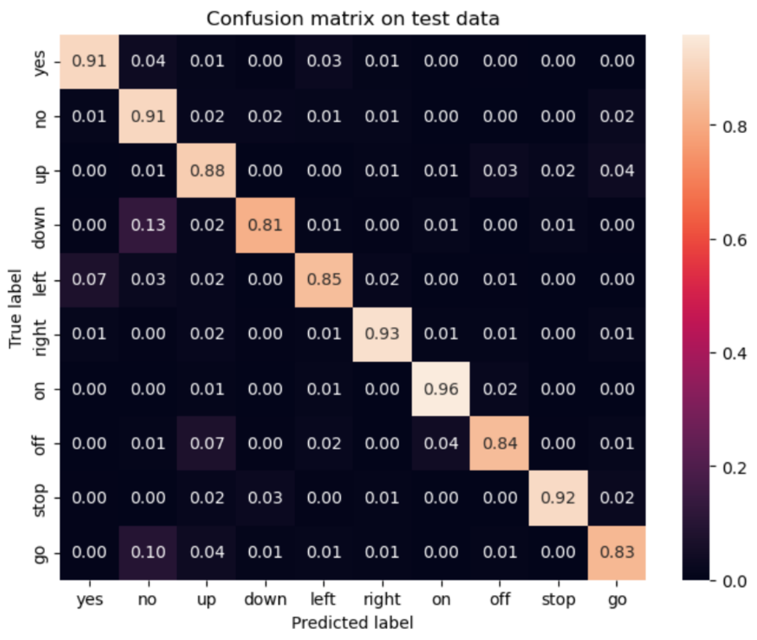


Figure 27: Confusion matrix for test data for GRU-1.

6.3.4 Results of M-GRU-2

GRU model with configuration 2 achieved results shown in Table 17.

Data	Accuracy
Train	0.97
Validation	0.93
Test	0.93

Table 17: Results of M-GRU-2

Moreover we present loss changes during training (Figure 28) and confusion matrix for test data (Figure 29).

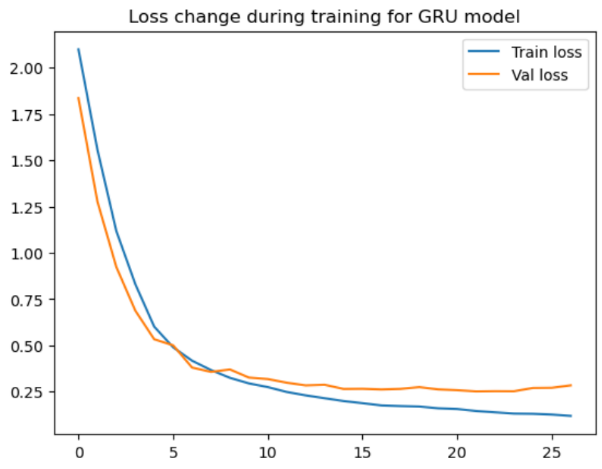
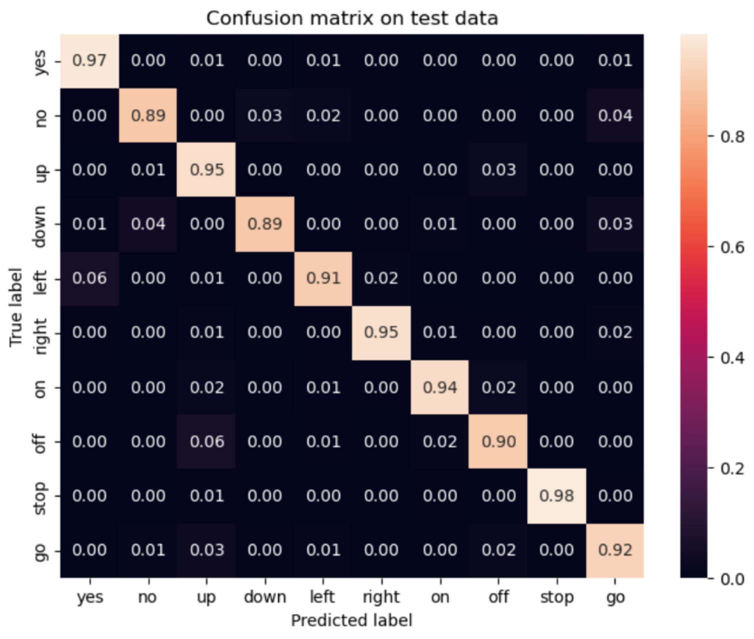


Figure 28: Loss changes during GRU-2 training.



	yes	no	up	down	left	right	on	off	stop	go
yes	0.97	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.01
no	0.00	0.89	0.00	0.03	0.02	0.00	0.00	0.00	0.00	0.04
up	0.00	0.01	0.95	0.00	0.00	0.00	0.00	0.03	0.00	0.00
down	0.01	0.04	0.00	0.89	0.00	0.00	0.01	0.00	0.00	0.03
left	0.06	0.00	0.01	0.00	0.91	0.02	0.00	0.00	0.00	0.00
right	0.00	0.00	0.01	0.00	0.00	0.95	0.01	0.00	0.00	0.02
on	0.00	0.00	0.02	0.00	0.01	0.00	0.94	0.02	0.00	0.00
off	0.00	0.00	0.06	0.00	0.01	0.00	0.02	0.90	0.00	0.00
stop	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.98	0.00
go	0.00	0.01	0.03	0.00	0.01	0.00	0.00	0.02	0.00	0.92

Figure 29: Confusion matrix for test data for GRU-2.

6.3.5 Results of M-Transformer-1

Transformer model with configuration 1 achieved results shown in Table 18.

Data	Accuracy
Validation	0.83
Test	0.81

Table 18: Results of M-Transformer-1

Moreover we present loss changes during training (Figure 30) and confusion matrix for test data (Figure 31).

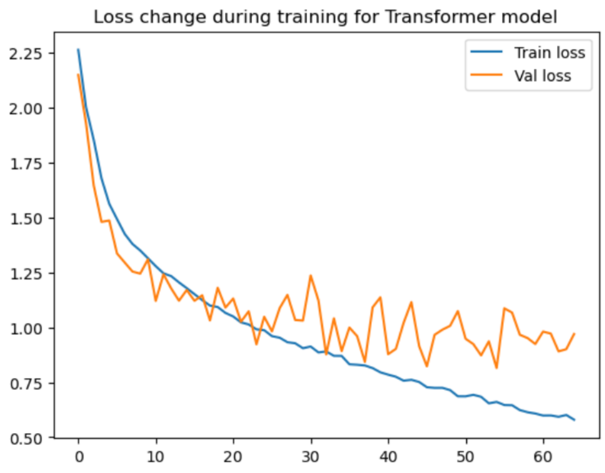
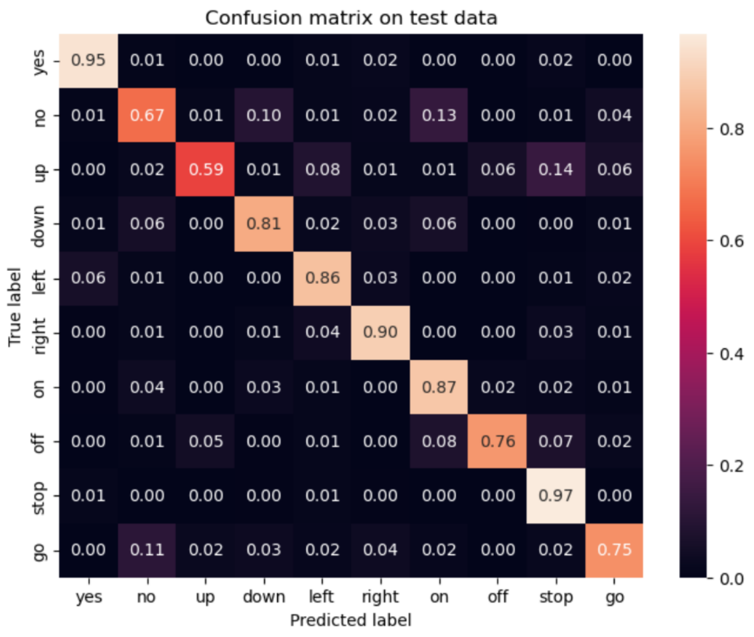


Figure 30: Loss changes during Transformer-1 training.



True \ Predicted	yes	no	up	down	left	right	on	off	stop	go
yes	0.95	0.01	0.00	0.00	0.01	0.02	0.00	0.00	0.02	0.00
no	0.01	0.67	0.01	0.10	0.01	0.02	0.13	0.00	0.01	0.04
up	0.00	0.02	0.59	0.01	0.08	0.01	0.01	0.06	0.14	0.06
down	0.01	0.06	0.00	0.81	0.02	0.03	0.06	0.00	0.00	0.01
left	0.06	0.01	0.00	0.00	0.86	0.03	0.00	0.00	0.01	0.02
right	0.00	0.01	0.00	0.01	0.04	0.90	0.00	0.00	0.03	0.01
on	0.00	0.04	0.00	0.03	0.01	0.00	0.87	0.02	0.02	0.01
off	0.00	0.01	0.05	0.00	0.01	0.00	0.08	0.76	0.07	0.02
stop	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.97	0.00
go	0.00	0.11	0.02	0.03	0.02	0.04	0.02	0.00	0.02	0.75

Figure 31: Confusion matrix for test data for Transformer-1.

6.3.6 Results of M-Transformer-2

Transformer model with configuration 2 achieved results shown in Table 19.

Data	Accuracy
Validation	0.84
Test	0.84

Table 19: Results of M-Transformer-2

Moreover we present loss changes during training (Figure 32) and confusion matrix for test data (Figure 33).

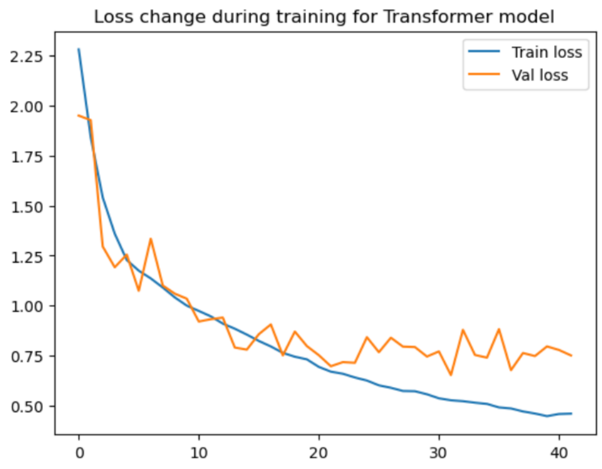
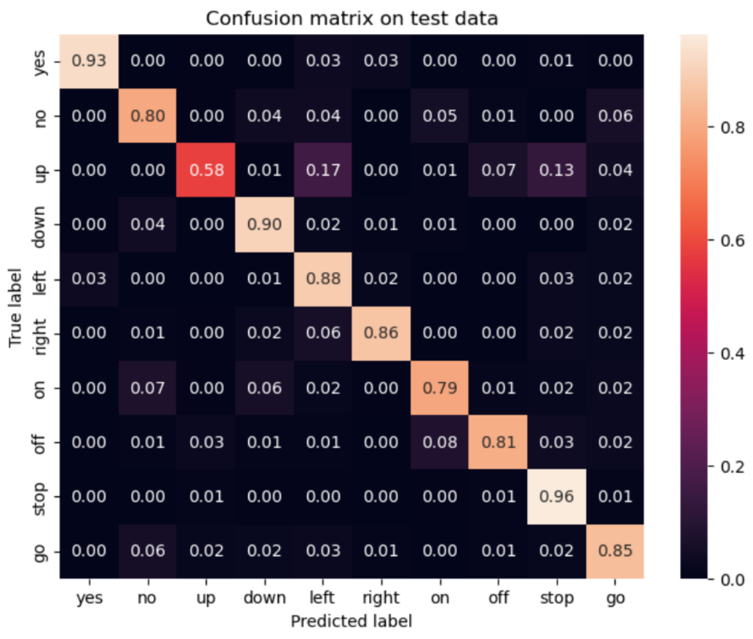


Figure 32: Loss changes during Transformer-2 training.



True \ Predicted	yes	no	up	down	left	right	on	off	stop	go
yes	0.93	0.00	0.00	0.00	0.03	0.03	0.00	0.00	0.01	0.00
no	0.00	0.80	0.00	0.04	0.04	0.00	0.05	0.01	0.00	0.06
up	0.00	0.00	0.58	0.01	0.17	0.00	0.01	0.07	0.13	0.04
down	0.00	0.04	0.00	0.90	0.02	0.01	0.01	0.00	0.00	0.02
left	0.03	0.00	0.00	0.01	0.88	0.02	0.00	0.00	0.03	0.02
right	0.00	0.01	0.00	0.02	0.06	0.86	0.00	0.00	0.02	0.02
on	0.00	0.07	0.00	0.06	0.02	0.00	0.79	0.01	0.02	0.02
off	0.00	0.01	0.03	0.01	0.01	0.00	0.08	0.81	0.03	0.02
stop	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.96	0.01
go	0.00	0.06	0.02	0.02	0.03	0.01	0.00	0.01	0.02	0.85

Figure 33: Confusion matrix for test data for Transformer-2.

6.4 Experiment conclusions

In our experimentation, we explored both simpler and more complex architectures, employing various layers tailored for processing sequential data. Despite our initial expectation that transformers would offer the most promising outcomes, Bi-LSTM and GRU consistently outperformed them in the majority of scenarios. We think this happened due to the fact that we employed relatively basic transformer architectures due to limitations in computational resources, encountering difficulties in loading larger models into memory. The other reason might be that transformer models are learning much longer than BI-LSTM and GRU, so we were not able to check as much parameter configurations for them.

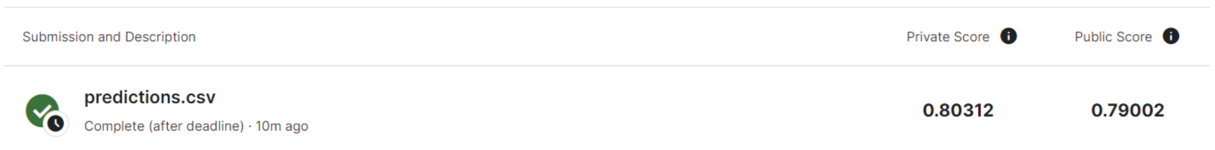
However, we attained quite reasonable results for each task. For the silence detection task, the top-performing model achieved a test accuracy of 0.99 using GRU (`S-GRU`). Similarly, for the Known vs. Unknown task, BI-LSTM (`K-BI-LSTM`) emerged as the best model with a test accuracy of 0.95. Lastly, for the main classification task, the GRU (`M-GRU-2`) model attained the highest accuracy with 0.93 on the test set.

We selected these best-performing models for each task and utilized them in the final classification phase, the outcomes of which are detailed in the subsequent section.

Upon analyzing the confusion matrices, it can be noticed that our models encountered no challenges in correctly identifying classes for the silence detection and Known vs. Unknown tasks. However, in the main task transformers had difficulty in accurately identifying the ”up” class, often confusing it with ”stop” or ”left”. Additionally, there is a tendency for ”go” to be misclassified as ”no”, which is understandable given the auditory similarity between the two words. Furthermore, both BI-LSTM and GRU models occasionally struggle with correctly classifying ”off”, predicting it as ”up”. These observations shown which words are difficult to distinguish and it might need additional insight to solve this problem. For instance specific models for binary classification between problematic classes might be developed.

7 Final Prediction

We obtained the test dataset from Kaggle and applied our best-performing models from previous experiments to make final predictions. Initially, we used the top model for the ”Silence” task to identify and classify any silence clips in the test samples. Then, we applied the best model from the ”Known vs. Unknown” task to categorize the remaining samples as either ”known” or ”unknown.” Finally, for the ”known” samples, we utilized the highest-performing model from the ”Main” task to assign them to their respective main class. This approach enabled us to classify each observation into one of the 12 classes (including Silence, Unknown, and the 10 main classes). The results of this classification were submitted to Kaggle for evaluation and in the Figure 34 we can presented accuracy score achieved by us on test data. What’s more, we counted number of occurrences of each class within our predictions and summarized the results in Table 20.



Submission and Description	Private Score	Public Score
predictions.csv Complete (after deadline) · 10m ago	0.80312	0.79002

Figure 34: Results of our models from Kaggle competition.

Class	No. of occurrences	Per. of all data
unknown	70331	0.44
silence	18500	0.12
up	8634	0.05
go	8581	0.05
left	7680	0.05
no	7063	0.04
stop	6786	0.04
right	6424	0.04
on	6375	0.04
off	6347	0.04
yes	6318	0.04
down	5499	0.03

Table 20: Occurrences of each class in our predictions.

8 Final conclusions

In summary, we have achieved promising results using relatively small models, as indicated by the final accuracy scores for each sub-task and the main classification problem, as shown in Table 21. The 0.8 accuracy score on the final test data from Kaggle demonstrates that our three primary models are able to accomplish audio classification task. However, it is notable that GRU and BI-LSTM models outperformed Transformer architectures in the majority of scenarios. We attribute this to the utilization of smaller Transformer architectures and the inability to explore more parameter configurations due to memory limitations. Therefore, it is possible that if we could train larger Transformer architectures (e.g. by stacking more encoder layers), they would surpass other models. Furthermore, due to memory constraints, we did not train models using augmented data, opting instead to augment only silence clips. Despite this, we achieved high scores with models that did not exhibit significant overfitting. However, as part of future work, it would be beneficial to train our models using a larger dataset comprising not only base data but also augmented clips. This could potentially enhance model performance and generalization capabilities.

Task	**Model**	Accuracy on test
Silence	`S-GRU`	0.99
Unknown	`K-BI-LSTM`	0.95
Main	`M-GRU-2`	0.93
Kaggle test	All combined	0.80

Table 21: Final results of the project.