Deep Learning

PROJECT 2 - REPORT

April 25, 2023

Maciej Chylak, Agata Kaczmarek

Contents

1 Description of the problem

In this project, we focus on solving the problem of classifying voice commands from the collection TensorFlow Speech Recognition Challenge [1]. The training collection consists of multiple audio files of 1 second in length that present the commands: yes, no, up, down, left, right, on, off, stop, go, etc. The test dataset, on the other hand, contains only previously mentioned classes and special classess unknown and silence.

2 Instruction of the application

All classes and functions in our project are implemented in Jupyter notebook. We chose this setting, as in that way we can show examples of the usage right after code implementation. In our notebook examples of how to run each function and some of the results can be found. To run the code, Python 3.9 is recommended. All packages listed in the two first cells in the notebook are required to run all experiments.

3 Theoretical introduction

For this project, we trained Recurrent Neural Networks (RNN) with various hyperparameters. We trained them on Speech Commands dataset. Short theoretical introduction to used techniques is described below.

3.1 Recurrent Neural Networks

Recurrent Neural Networks are a special type of neural networks, which allows that during the training, output from some of the nodes in the network affect input of the other nodes. This allows them to process data, which consist of sequences, such as speech recognition.

3.1.1 Vanilla RNN

Vanilla RNN consists of a single layer of recurrent neurons, where each neuron receives an input vector and a feedback signal from its own output from the previous timestep. The output of each neuron is computed by applying a set of weights to the concatenation of the input vector and the feedback signal, and then passing the result through an activation function. In figure 1 we can see this architecture, where x is input data and h is output.

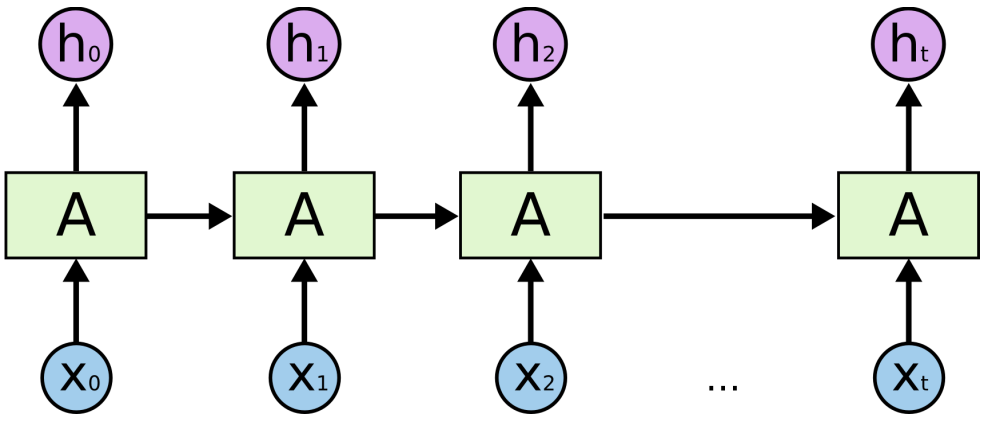

Figure 1: Architecture of SimplyRNN (source of the image).

3.1.2 Long short-term memory

LSTM is one of recurrent neural networks. LSTM unit contains four cells: memory cell that is designed to store information for long periods, input gate which influences the importance of new information given to cell, output gate that controls the flow of information out of the cell and forget cell that regulates retention of information from the memory cell. Thanks to that architecture this neural network is able to process sequence data such as text, time series, and audio. In figure 2, we can see the architecture of the LSTM network, in which the cell state (top horizontal line) plays a key role. The LSTM network has the possibility of adding or removing information located there by using the previously mentioned gates.

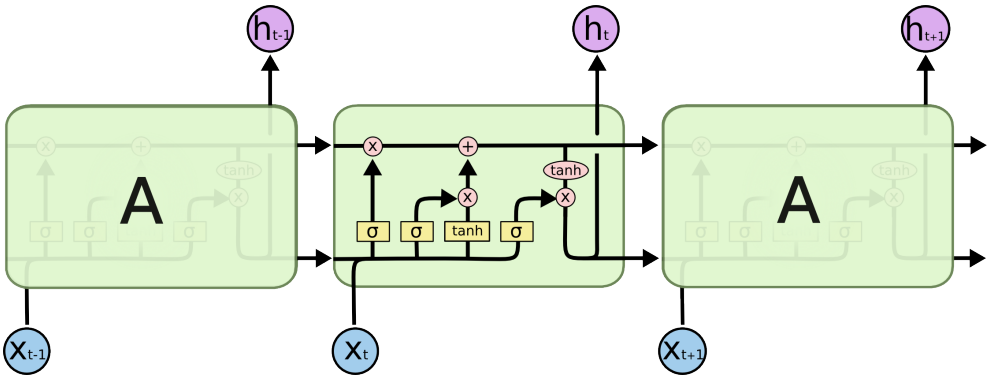

Figure 2: Architecture of LSTM (source of the image).

3.2 Multi-layer gated recurrent unit

Gated Recurrent Unit (GRU) is an improved version of RNN. It also aims at solving the vanishing gradient problem and uses for this update gate and reset gate. The update gate controls how much information from the previous time step should be preserved, and the reset gate controls how much of the past information should be discarded.

3.3 Speech Commands dataset

The training collection consists of 65 000 one-second-long audio files that present the commands: *bed*, *bird*, *cat*, *dog*, *down*, *eight*, *five*, *four*, *go*, *happy*, *house*, *left*, *marvin*, *nine*, *no*, *off*, *on*, *one*, *right*, *seven*, *sheila*, *six*, *stop*, *three*, *tree*, *two*, *up*, *wow*, *yes*, *zero*. The test dataset, on the other hand, contains some of the above-mentioned commands and several others that should be classified as *silence* or *unknown*.

3.4 Hyperparameters

Hyperparameters in general are used to control models' learning process. In our project we decided to check the influence of four various, two related to the architecture (probability for dropout and bidirectional) and two related to training process (learning rate and batch size).

3.4.1 Dropout

One of the requlariasation techniques, its main goal is to prevent over-fitting of the neural network. With given probability some neurons are used and only they take part in one following learning step. Next step, next neurons are randomly chosen to be used. The result is, that the model is more robust on the noise in dataset. Single epoch in training process takes less time, however, the model needs more epochs to train.

3.4.2 Bidirectional flag

Flag indicating whether the network should operate bidirectionally. If yes, the input is given to the network in normal time order, and also to the other one in reverse order. Later, the output of those two is concatenated.

3.4.3 Learning rate

This hyperparameter controls how much do weights of a network change, with respect to the estimated error. If the chosen learning rate is too small, the training will be unnecessary longer. However, also too big value is not recommended - in that case the training process can be unstable.

3.4.4 Batch size

Batch size is the number of audio files from the dataset, that is passed to the network at the same time. If we have 1000 audio files, and the batch size equals to 10, The epoch contains 100 runs and updates of weights for the network. In general, the larger the batch size, the less time network needs to complete each epoch of the training process. However, this can cause worse metrics results of the model.

4 Methods of solving the problem of silence and unknown classes

4.1 Silence class

We decided to test two different ways of generating silence. In both cases of dealing with silence, we decided to use the audio files available to us in train datasets and process them. These two ways are:

- variant/mode 1: silence generated using cut fragments from all audio files. We decided to collect all the fragments of files where the volume was below 40 decibels and put them together in one-second fragments
- variant/mode 2: silence generated by cutting background_noise files into one-second sound fragments

4.2 Unknown class

When it comes to dealing with the unknown class we have decided to do it in three ways:

- variant/mode 1: all predictions that do not belong to the set of test classes had labels changed to unknown
- variant/mode 2: reworking the other audio files in various ways so that they do not look like the original sound and then setting their labels to unknown
- variant/mode 3: if the model is not sure about the result for several classes then set the output to unknown, as it is most likely to be one of the classes that we do not know

5 Experiments

To check how good the models are able to perform on the given data and to see the influence of the chosen hyperparameters on models accuracy, we conducted several experiments, which are described in details below. We also checked the influence of methods of dealing with silence and unknown classes on final accuracy.

5.1 Chosen technology

For this project, we used the Python language and its library, which supports the creation of neural networks and work with them - Keras.

5.2 Used architectures

In our experiments, we relied on the network layers implemented in the Keras library (SimpleRNN, LSTM, GRU [2] [3] [4]), to which we then added a dropout and optionally a bidirectional layer. During training, we used the adam optimiser which is the extended version of stochastic gradient descent. In addition, we added early stopping to prevent overtraining the network. As a loss function, we chose categorical cross entropy.

5.3 Tested hyperparameters

During the work on this project, we tested the influence of four hyperparameters on models behaviour. These hyperparameters were probability for dropout, bidirectional, learning rate, and batch size.

5.4 Description of conducted experiments

Before conducting all other experiments, we decided to train one, baseline model for each of the architectures. We did this to see what kind of performance, we will be looking for later.

Hyperparameter	Grid
dropout	[0.1, 0.3]
bidirectional	[True, False]
batch size	[32, 64]
learning rate	[0.001, 0.01]

Table 1: Hyperparameter grid for experiments

LSTM At the beginning of the experiments, we trained a baseline model with the following hyperparameters:

- probability of dropout - 0.1
- bidirectional - False
- learning rate - 0.001
- batch size - 64

We conducted the following experiments for the LSTM:

1. training of neural networks for the grid of hyperparameters presented in table 1 for silence prepared from all audio files and for the unknown class prepared by processing the audio files

2. training of neural networks using a subset of data (100 observations for each class) for the grid of hyperparameters presented in table 1 for silence prepared using the background_sample class and for the unknown class prepared by processing the audio files

3. for the four best networks in Experiments 1 and 2 train networks with the same architectures on subset of data (100 observations for each class), but with different ways of dealing with unknown classes

4. calculation of mean and standard deviation accuracy for the best neural network run on a subset of the data (100 observations for each class) for 3 iterations

SimpleRNN At the beginning of the experiments, we trained a baseline model with following hyperparameters:

- probability of dropout - 0.1
- bidirectional - False
- learning rate - 0.001
- batch size - 64

We conducted the following experiments for the SimpleRNN:

1. training of neural networks for the grid of hyperparameters presented in table 1 for silence prepared from all audio files and for the unknown class prepared by processing the audio files

2. training of neural networks using a subset of data (100 observations for each class) for the grid of hyperparameters presented in table 1 for silence prepared using the background_sample class and for the unknown class prepared by processing the audio files

3. for the four best networks in Experiments 1 and 2 train networks with the same architectures on subset of data (100 observations for each class), but with different ways of dealing with unknown classes

4. calculation of mean and standard deviation accuracy for the best neural network run on a subset of the data (100 observations for each class) for 3 iterations

Multi-layer gated recurrent unit At the beginning of the experiments, we trained a baseline model with following hyperparameters:

- probability of dropout - 0.1
- bidirectional - False
- learning rate - 0.001
- batch size - 64

We conducted the following experiments for the Multi-layer GRU:

1. training of neural networks for the grid of hyperparameters presented in table 1 for silence prepared from all audio files and for the unknown class prepared by processing the audio files

2. training of neural networks using a subset of data (100 observations for each class) for the grid of hyperparameters presented in table 1 for silence prepared using the background_sample class and for the unknown class prepared by processing the audio files

3. for the four best networks in Experiments 1 and 2 train networks with the same architectures on subset of data (100 observations for each class), but with different ways of dealing with unknown classes

4. calculation of mean and standard deviation accuracy for the best neural network run on a subset of the data (100 observations for each class) for 3 iterations

6 Results

6.1 Evaluation of the solution

Results of conducted experiments were evaluated either according to accuracy, the models achieved on validation set.

6.2 Obtained results

Below we present the results of conducted experiments.

6.2.1 LSTM

Experiment 1 As we can see in table 3, the results of our models differ quite significantly depending on the hyperparameters used. The biggest difference is noticeable in the case of learning rate and batch size selection, where the results differed by 10 percentage points. As far as the overall results that we can see in table 2 are concerned, the best model we obtained achieved an efficiency of 0.9 accuracy. The training accuracy can be seen in chart 3.

Best four				
bidirectional	dropout	lr	batch size	accuracy
True	0.3	0.01	32	0.900
True	0.1	0.001	32	0.899
True	0.3	0.001	64	0.886
True	0.1	0.01	64	0.882

Table 2: Best LSTM neural networks trained in experiment 1

hyperparameter	value	average	std
lr	0.01	**0.739**	0.225
	0.001	**0.837**	0.115
batch size	32	**0.736**	0.241
	64	**0.839**	0.075
bidirectional	True	0.804	0.233
	False	0.772	0.121
dropout	0.1	0.762	0.228
	0.3	0.813	0.127

Table 3: Experiment 1 on LSTM

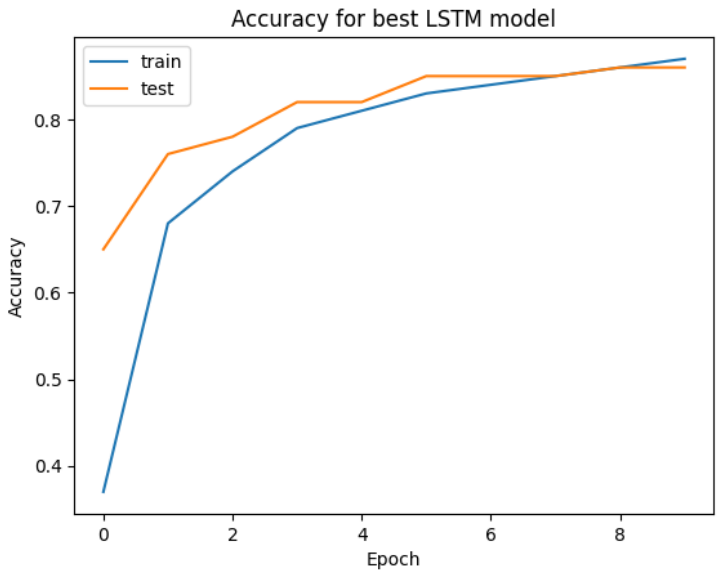

Figure 3: Chart showing the train and test accuracy while training for the best LSTM model.

Experiment 2 For silence prepared using the background_sample we run the grid search of hyperparameters that were chosen from the same values as for data with other kind of silence, as shown in the table 5. This time the results were slightly lower, as the models were trained on smaller amount of data (due to computational and time limits of our machines) and for fewer epochs. Even though the results were satisfying as can be seen in tables 4. The biggest difference in the average accuracy can be seen for bidirectional True and False, for True it was better.

Best four				
bidirectional	dropout	lr	batch size	accuracy
True	0.3	0.01	32	0.53
True	0.1	0.001	64	0.484
True	0.3	0.001	64	0.478
True	0.3	0.001	32	0.466

Table 4: Best LSTM neural networks trained in experiment 2.

hyperparameter	value	average	std
lr	0.01	0.31	0.133
	0.001	0.338	0.14
batch size	32	0.335	0.15
	64	0.31	0.12
bidirectional	True	**0.435**	0.10
	False	**0.213**	0.056
dropout	0.1	0.348	0.111
	0.3	0.299	0.156

Table 5: Experiment 2 on LSTM - for silence from background noise.

Experiment 3 This experiment was conducted for the best two models from Experiment 1 and Experiment 2. We can see the results of the experiment in table 6. As we can see, the results vary depending on the silence mode used. For the silence from variant 1, the unknown class dealing variant 3 turned out to be the best, while for the silence variant 2, variant 2 turned out to be the most successful.

silence_mode	unknown_mode	bidirectional	dropout	lr	batch size	balanced_accuracy
1	1	True	0.3	0.01	32	0.499
1	2	True	0.3	0.01	32	0.477
1	3	True	0.3	0.01	32	0.576
2	1	True	0.3	0.001	32	0.36
2	2	True	0.3	0.001	32	0.42
2	3	True	0.3	0.001	32	0.36

Table 6: Experiment 3 on LSTM

Experiment 4 As for the stability tests of the best models we obtained, the results of which we can preview in table 7, they perform very well. We can see that the results for the different iterations are very close to each other.

Model	Silence	Iter 1	Iter 2	Iter 3	Avg	Stdev
True 0.3 0.001 64	from audio	0.558	0.541	0.510	**0.5363**	0.0198
True 0.3 0.001 32	from background	0.454	0.479	0.500	**0.477**	0.018

Table 7: Experiment 4 on LSTM

6.2.2 SimpleRNN

Best four				
bidirectional	dropout	lr	batch size	accuracy
True	0.3	0.001	64	0.431
True	0.1	0.001	64	0.378
True	0.3	0.001	32	0.363
True	0.1	0.001	32	0.341

Table 8: Best SimpleRNN neural networks trained in experiment 1

hyperparameter	value	average	std
lr	0.01	**0.126**	0.073
	0.001	**0.316**	0.089
batch size	32	0.219	0.116
	64	0.223	0.143
bidirectional	True	0.24	0.153
	False	0.202	0.097
dropout	0.1	0.224	0.113
	0.3	0.218	0.144

Table 9: Experiment 1 on SimpleRNN

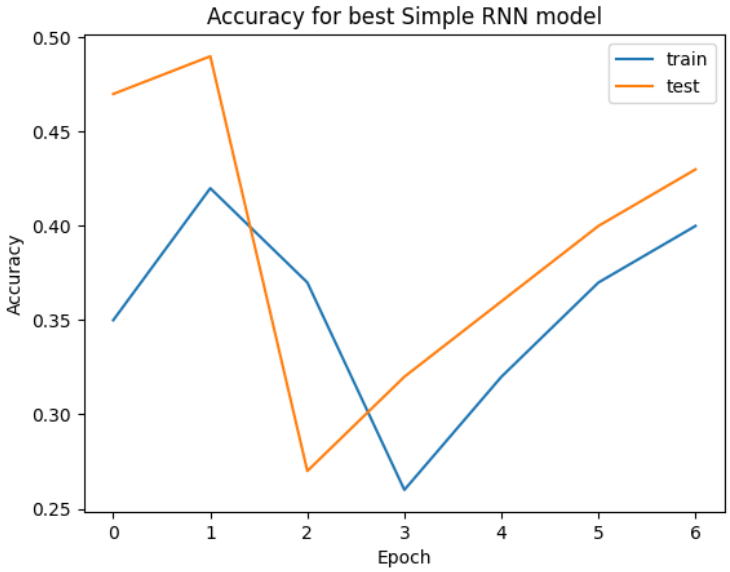

Figure 4: Chart showing the train and test accuracy while training for the best Simple RNN model.

Experiment 1 When it comes to hyperparameter grid search experiments for SimplerRNN networks, the results look generally poor. The best network achieved only 0.43 accuracy (table 10). However, when it comes to the impact of the hyperparameters, the results of which can be viewed in table 9, the biggest difference can be seen in the case of the learning rate selection, where for a value of 0.001 we have a jump of 20 percentage points. In figure 4 we can view the progression of the accuracy in training for the best network we have obtained.

Experiment 2 For silence prepared using the background_sample we run the grid search of hyperparameters that were chosen from the same values as for data with another kind of silence, as shown in table 11. This time the results were slightly lower, as the models were trained on smaller amount of data (due to computational and time limits of our machines) and for fewer epochs. Even though the results were satisfying as can be seen in table 10. The biggest difference can be seen between learning rates - for 0.001 the accuracy is much bigger.

Best four				
bidirectional	dropout	lr	batch size	accuracy
True	0.3	0.001	64	0.415
True	0.1	0.001	64	0.409
True	0.3	0.001	32	0.401
True	0.1	0.001	32	0.39

Table 10: Best Simple RNN neural networks trained in experiment 2 for silence from background noise.

hyperparameter	value	average	std
lr	0.01	**0.034**	0.006
	0.001	**0.334**	0.087
batch size	32	0.167	0.15
	64	0.259	0.159
bidirectional	True	0.234	0.18
	False	0.14	0.09
dropout	0.1	0.17	0.15
	0.3	0.19	0.15

Table 11: Experiment 2 on Simple RNN - for silence from background noise.

silence_mode	unknown_mode	bidirectional	dropout	lr	batch size	balanced_accuracy
1	1	True	0.3	0.001	64	0.403
1	2	True	0.3	0.001	64	0.4159
1	3	True	0.3	0.001	64	0.467
2	1	True	0.3	0.001	64	0.45
2	2	True	0.3	0.001	64	0.42
2	3	True	0.3	0.001	64	0.36

Table 12: Experiment 3 on SimpleRNN

Experiment 3 We can see, also in the case of the SimplerRNN network, that it is difficult to make clear conclusions about the impact of the choice of dealing with the unknown class, as the results we can see in Table 12 are different for the different variants of dealing with silence. For the first variant, unknown mode 3 was found to be the best, while for the second, unknown mode 1.

Model	Silence	Iter 1	Iter 2	Iter 3	Avg	Stdev
True 0.3 0.01 32	from audio	0.031	0.031	0.0484	**0.0368**	0.0082
True 0.3 0.001 64	from background	0.382	0.296	0.411	**0.363**	0.048

Table 13: Experiment 4 on SimpleRNN

Experiment 4 We can see that the results, shown in table 13, of our neural networks are similar for successive iterations for the first neural network. However, we can notice that for a network trained on silence from all audio files, the network no longer looks as good as when it was trained on all data. Concerning the second network, the results are very unstable.

6.2.3 Multi-layer GRU

Experiment 1 As we can see in table 15, the results of our models differ slightly depending on the hyperparameters used. The biggest difference is noticeable in the case of learning rate, where the results differed by 20 percentage points. The overall results, shown in table 14, show, the best models we obtained. They achieved an efficiency of over 0.9 accuracy. The training accuracy can be seen on chart 3.

Best four				
bidirectional	dropout	lr	batch size	accuracy
True	0.3	0.001	32	0.909
True	0.1	0.001	32	0.908
True	0.1	0.001	64	0.902
True	0.3	0.001	64	0.901

Table 14: Best GRU neural networks trained in experiment 1

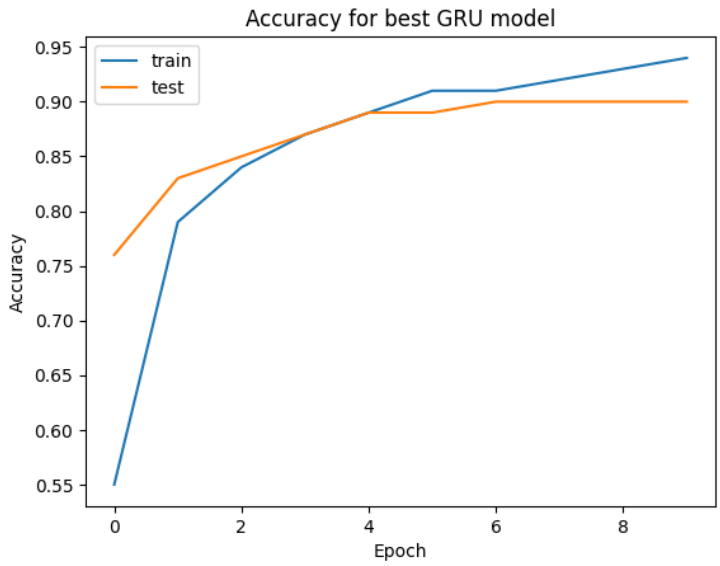

Figure 5: Chart showing the train and test accuracy while training for the best GRU model.

hyperparameter	value	average	std
lr	0.01	**0.692**	0.223
	0.001	**0.896**	0.011
batch size	32	0.837	0.084
	64	0.752	0.25
bidirectional	True	0.837	0.092
	False	0.751	0.247
dropout	0.1	0.836	0.089
	0.3	0.752	0.248

Table 15: Experiment 1 on GRU

Experiment 2 For silence prepared using the background_sample we run the grid search of hyperparameters that were chosen from the same values as for data with another kind of silence, as shown in the table 16. This time the results were slightly lower, as the models were trained on smaller amount of data (due to computational and time limits of our machines) and for fewer epochs. Even though the results were satisfying as can be seen in table 17. The biggest difference can be seen between various learning rate bidirectional values.

Best four				
bidirectional	dropout	lr	batch size	accuracy
True	0.1	0.01	64	0.749
True	0.3	0.01	64	0.734
True	0.1	0.001	32	0.725
True	0.3	0.01	32	0.724

Table 16: Best GRU neural networks trained in experiment 2 for silence from background noise.

Experiment 3 We conducted this experiment for the best two models from Experiment 1 and Experiment 2. We can see the results of the experiment in table 18. As we can see, the results vary depending on the silence used. For the silence from variant 1, the unknown class dealing variant 1 turned out to be the best, while for the silence variant 2, variant 2 turned out to be the most successful.

hyperparameter	value	average	std
lr	0.01	**0.406**	0.12
	0.001	**0.66**	0.071
batch size	32	0.543	0.151
	64	0.527	0.173
bidirectional	True	**0.628**	0.107
	False	**0.442**	0.155
dropout	0.1	0.53	0.165
	0.3	0.541	0.16

Table 17: Experiment 2 on GRU - for silence from background noise.

silence_mode	unknown_mode	bidirectional	dropout	lr	batch size	balanced_accuracy
1	1	True	0.3	0.001	32	0.692
1	2	True	0.3	0.001	32	0.610
1	3	True	0.3	0.001	32	0.674
2	1	True	0.1	0.01	64	0.362
2	2	True	0.1	0.01	64	0.754
2	3	True	0.1	0.01	64	0.361

Table 18: Experiment 3 on GRU

Experiment 4 As for the stability tests of the best models we obtained, the results of which we can see in table 19, they perform very well. We can see that the standard deviation of the results in each group is small.

Model	Silence	Iter 1	Iter 2	Iter 3	Avg	Stdev
True 0.3 0.001 32	from audio	0.6915	0.6093	0.6741	**0.6583**	0.0082
True 0.1 0.01 64	from background	0.734	0.737	0.733	**0.734**	0.0016

Table 19: Experiment 4 on GRU

Additional experiment While creating this project, we conducted additional experiment, we trained Logistic Regression model for preprocessed data. Surprisingly, it was able to score over 0.45 accuracy, what we think is a good result.

6.2.4 Kaggle competition

In order to give credibility to our solutions, we decided to upload the predicted values for the test set for one best-performing network for LSTM and GRU:

- LSTM - bidirectional True, dropout 0.3, learning rate 0.01, batch size 32
- GRU - bidirectional True, dropout 0.3, learning rate 0.01, batch size 64

The results can be seen in figure 6. They turned out to be worse than those obtained on the validation data.

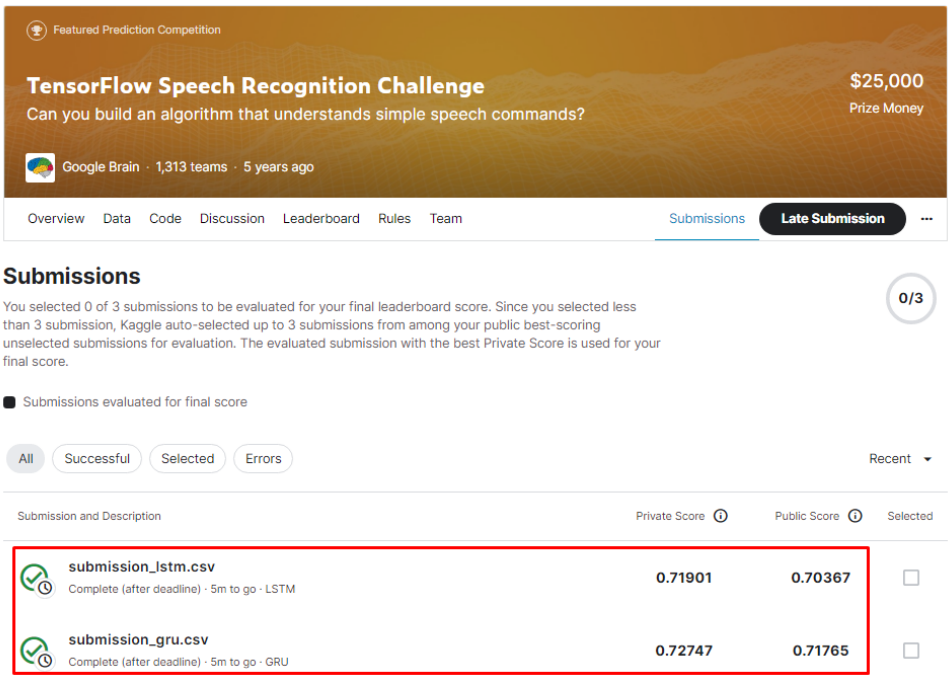

Figure 6: Kaggle competition results

7 Conclusions

During work on this project, we tried to find the best approach to classify the given recordings. We tested various hyperparameters settings and various methods of handling *silence* and *unknown* classes and observed the changes in the model's accuracy. The most important outcome of this project is, that not all models (even RNN) are able to learn to this problem, for example our SimpleRNN. This network had problems with accuracy, no matter which silence and which unknown setting, the results were much worst then the other networks.
For the hyperparameters:

- the LSTM and GRU bidirectional were learning better than the ones not bidirectional,
- batch size can influence the learning pace,
- for not suitable learning rate, the models tend to learn slower, or even do not learn anything,
- dropout did not have much influence on the performance,
- the results for various silence settings were similar.

Future work It would be worth training all neural networks in grid search on the whole dataset. Additionally, all manipulations we did with the recordings were quite simple as we are not voice specialists. Having more domain specific knowledge, would be helpful to chose parameters for preprocessing, which might improve the results.

References

[1] Competition page - https://www.kaggle.com/competitions/tensorflow-speech-recognition-challenge/

[2] SimpleRNN - https://www.tensorflow.org/api_docs/python/tf/keras/layers/SimpleRNN

[3] LSTM - https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

[4] GRU - https://www.tensorflow.org/api_docs/python/tf/keras/layers/GRU