Deep Learning Project 2

Dutt Salveen Singh, Patryk Prusak

supervisor

mgr inż. Stanisław Kaźmierczak

Warsaw University of Technology

November 18, 2024

Contents

1 Problem Description

The aim of the project is to test and compare different network architectures (preferably Transformers) with different hyperparameters on a Speech Commands Dataset[1]. For the hyper-parameters related to the training process, we chose batch size, learning rate, type of optimizer, number of epochs, and train-test splits. The influence of the mentioned parameters is measured in terms of the model's (LSTM, Whisper) accuracy.

2 Application Instruction

The solution has been prepared in Python Jupyter Notebook. It is enough to install the required packages (visible at the beginning of the notebook) and run cells one by one. The preferable folder with the downloaded and extracted dataset is also visible at the beginning of the notebook. Note that seeds are predefined to ensure reproducible randomness.

3 Theoretical Introduction

During the experiments, a model architecture called Long Short-Term Memory (LSTM) is used. LSTM is a type of Recurrent Neural Network that handles sequential data[2]. The advantage of LSTM is that it addresses the vanishing gradient problem, it is capable of learning long-term dependencies. Experiments have been mainly performed on an LSTM with convolutional layers before the proper LSTM layers, but the performance of a bare-bones LSTM has also been investigated. Moreover, we examine the efficiency of a pretrained Transformer - Whisper[3]. Transformers take a much different approach by dropping the requirement for recurrence, they are based on attention mechanisms. This kind of architecture allows for concurrency and it learns context keeping track of relations between input data.

The multi-class classifier's performance can be measured in terms of accuracy (number of correct predictions divided by all predictions) and can be represented with a confusion matrix[4].

There are many parameters involved in the process of creating and training machine learning models. Starting from input data that can undergo the process of augmentation[5]. The process of learning consists of epochs, an epoch is a single pass through all the training data, in general, an increase in a number of epochs results in increased accuracy until a certain point. One can also modify the batch size, which is a number of samples that go through the network in a single pass during fitting or learning rate that dictates how quickly the model converges to the right solution.[6] With learning rates, we associate different optimizers, in our solution, we use Stochastic Gradient Descent, ADAM, and Lion.[7] These concepts should be enough to understand the performed experiments.

4 Experiments

The experiments have been designed with the best accuracy in mind. We have constructed adaptations of chosen models (LSTM, Whisper) to the problem at hand. Having that, a number of experiments have been performed to find the best possible values for the parameters under investigation. The procedure can be described as:

1. Choose a starting parameter X (such as batch size) and answer the following question: given a set of basic (commonly seen in literature) parameter values (such as a hyperparameter) excluding parameter X, what value for X results in best accuracy?
2. Choose the best-found value for parameter X and replace the parameter's basic initial value.
3. Repeat 1. with the new parameter's X value and choose a different parameter X.
4. The procedure finishes after investigating: batch size, learning rate, number of epochs, optimizer type, and test size

Each experiment has been repeated 5 times on 20% of the original dataset given time and computing power constraints. Lastly, each model has been trained for 50 epochs on the reduced dataset in order to examine the optimal number of epochs. The following parameter values have been taken into consideration:

- Batch size: 128, 64 and 32
- Learning rate: 0.01,0.001,0.0001
- Optimizers: Adam, Lion, SGD (with each of the different learning rates)
- Test sizes: 10%, 20%, 30%

4.1 Dataset

The used Speech Commands Dataset dataset consists of .wav audio files. There are 31 classes (including background noise). Apart from the background noise class, each of them contains on average 2157 data points of similar length. Background noise is a collection of 7 longer audio files that require some additional splitting before it can be used for training.

5 Results

The results for the described experiments are as follows. Created LSTM architecture consists of layers visible in Figure 1:

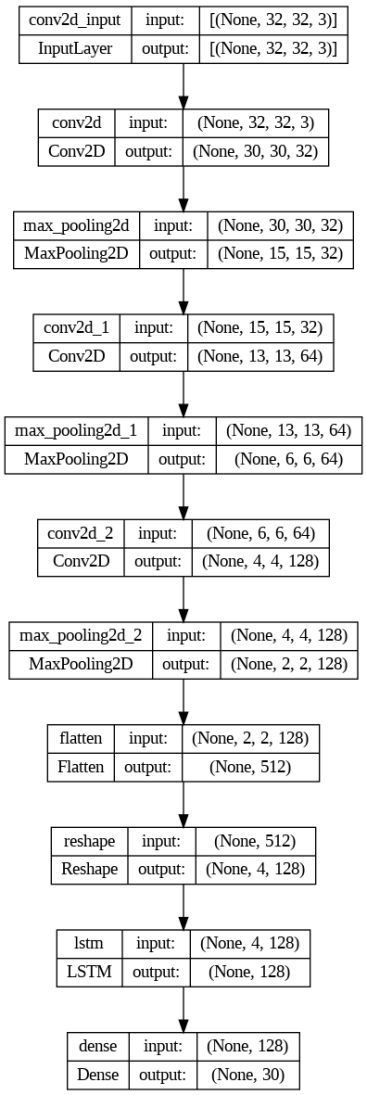

Figure 1: LSTM Architecture Diagram

From the tested parameters, the best that have been found for LSTM are 128 batch size, 0.001 learning rate with Lion optimizer, 0.1 test split, trained for 10 epochs. However, when the model was trained on a larger dataset it seemed to converge quite quickly using the Lion optimizer (visible in figures 11, 3 and 4) but in the end, results were worse than the results of the Adam optimizer. A possible conclusion might be that although Lion converges faster than Adam, it is more sensitive to the amount of data provided for the training process. Adam, on the other hand, seems to be more robust to parameters such as learning rate, batch size, or size of training data, following that

for the final results presented in figures 2 and 10 we decided to use Adam optimizer. It is quite possible that the training process could be yet made more efficient with the use of Lion Optimizer, perhaps the learning rate should be lowered or the batch size increased. Lowering the number of epochs could also be a possible solution. This matter requires further investigation in order to draw any satisfactory conclusions.

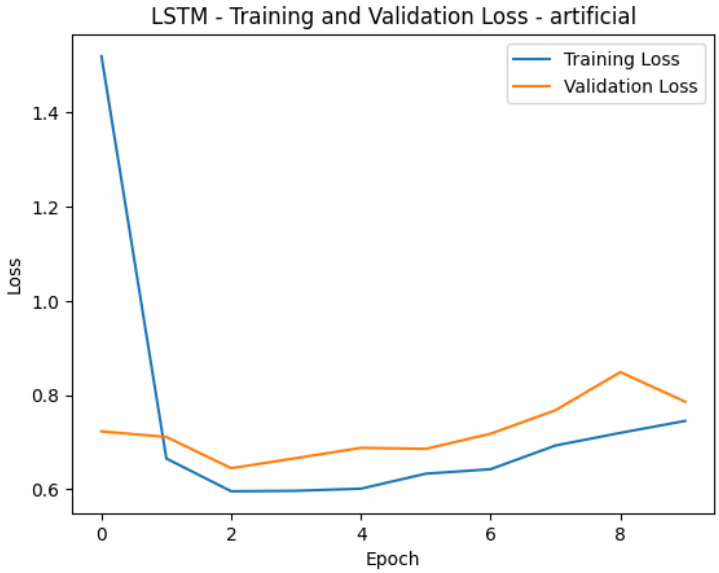

Figure 2: Training and Validation Loss (LSTM, Lion)

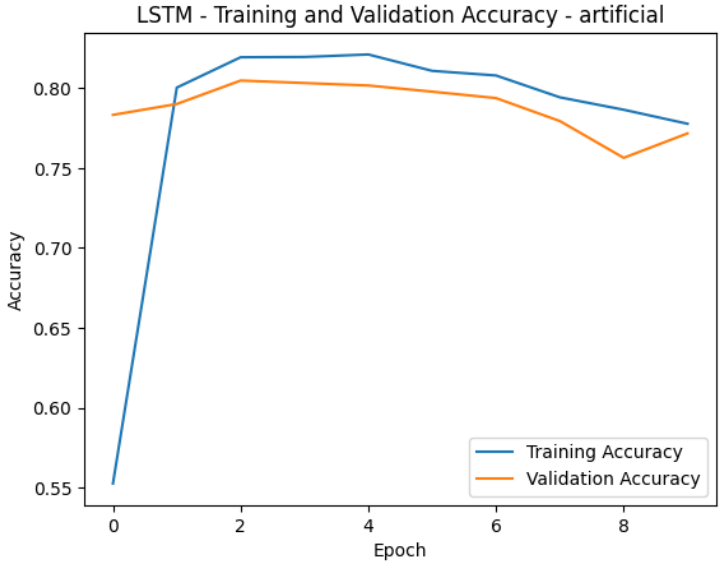

Figure 3: Training and Validation Accuracy (LSTM, Lion)

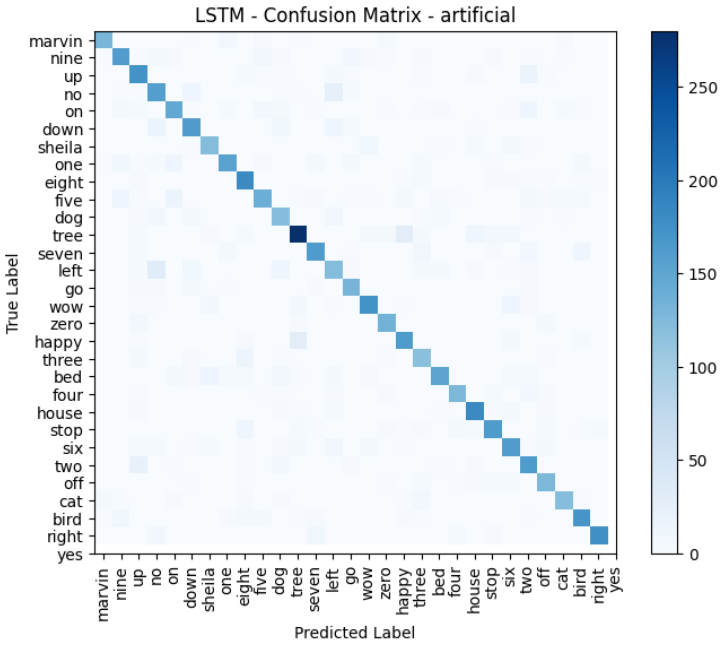

Figure 4: Confusion Matrix (LSTM, Lion)

In the dataset there is a special class "_background_noise_". First, we decided to investigate the performance of models at hand on a dataset with the background noise class removed. In such case, for LSTM, with the increase of used batch size, accuracy also increased (figure 5). In the case of optimizers, SGD proved to be quite inefficient resulting in very low accuracy regardless of the learning rate used. For Adam and Lion, similarly, a learning rate of 0.001 offers the best accuracy. Lion seems to outperform Adam at learning rates of 0.0001 and 0.001 but fails at a learning rate of 0.01 indicating that Lion is quite sensitive to large learning rates and requires more precise fine tuning as visible in figure 6. Figure 7 shows that all train test splits offer similar results, with 20% test size having the largest variance and 10% test size offering the best results. Accuracy and loss graphs (figures 8 and 9) show that the LSTM converges quite quickly and when we arrive at the 10th epoch there is not much improvement going further. Performance of the final model can be seen in figures 10 and 11, most of the improvement in accuracy and loss can be seen in the first 2 epochs, indicating a very fast convergence, moreover, confusion matrix presented in figure 12 indicates high performance of the model, classes are identified correctly shown by intense colors on the diagonal and very bleak colors in other cells. The model's performance is not perfect it seldom confuses "down" with "no" for example but it can be considered satisfactory.

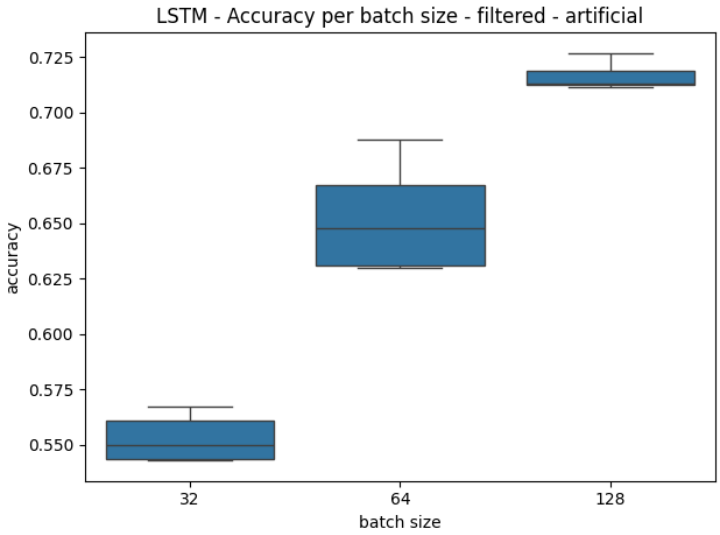

Figure 5: Accuracy vs Batch size (LSTM)

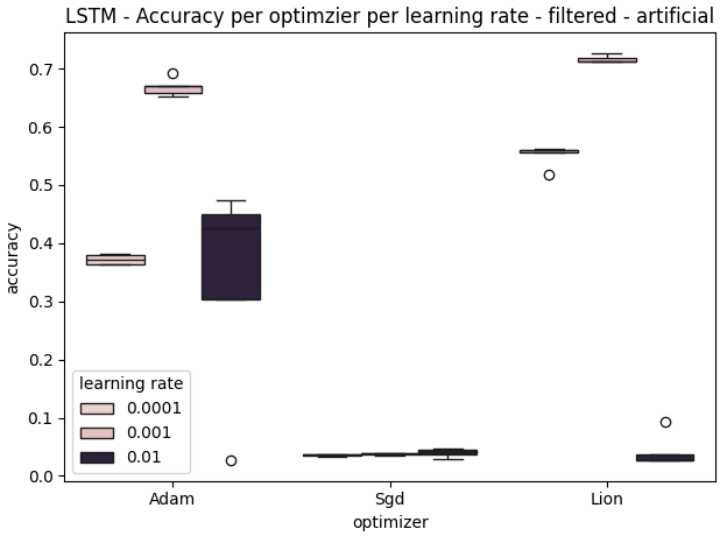

Figure 6: Accuracy vs Learning Rate (LSTM)

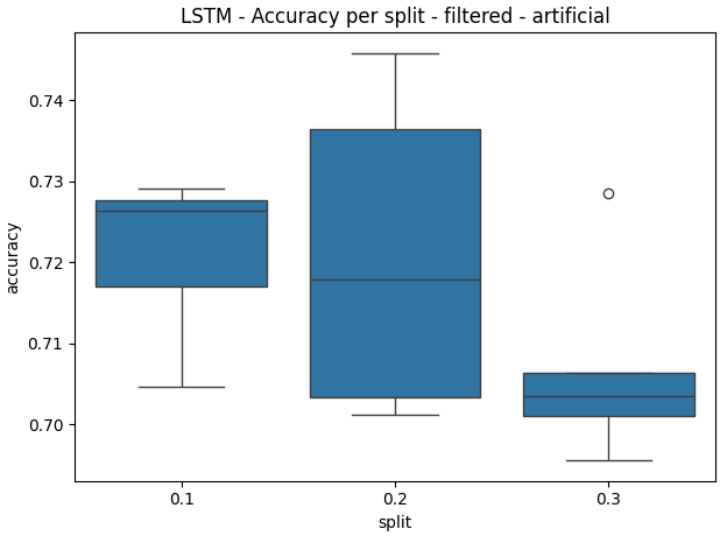

Figure 7: Accuracy vs Train-Test Split (LSTM)

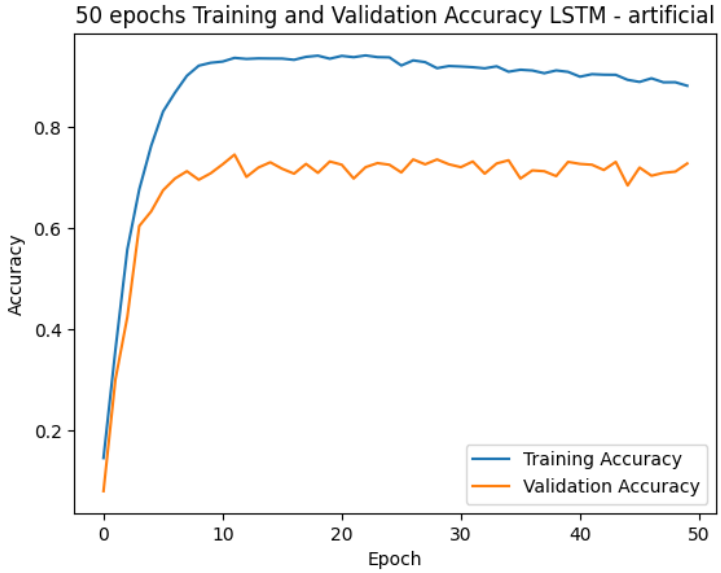

Figure 8: 50 epochs accuracy (LSTM)

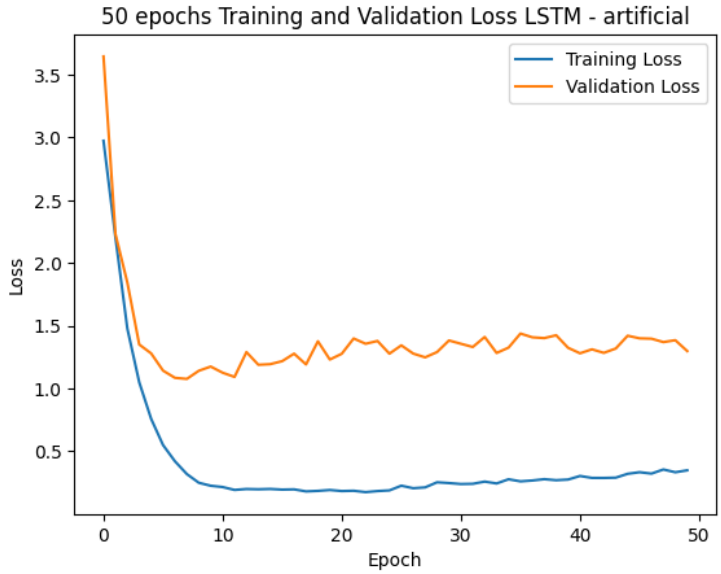

Figure 9: 50 epochs loss (LSTM)

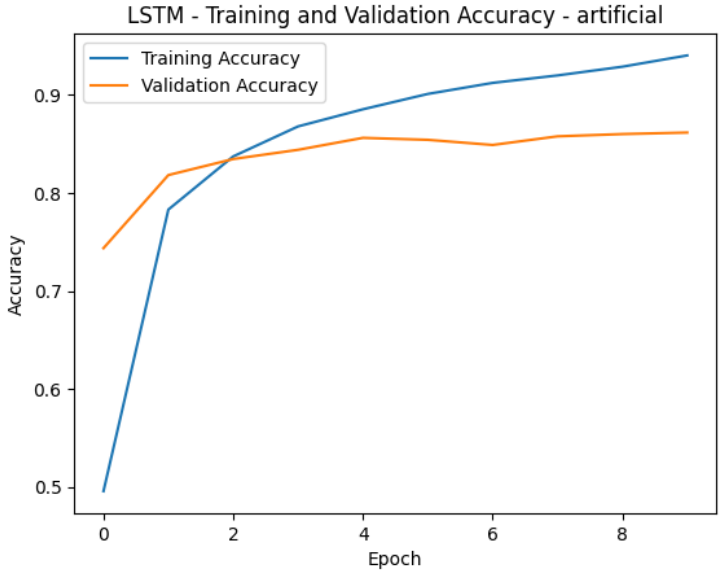

Figure 10: Final Accuracy (LSTM)

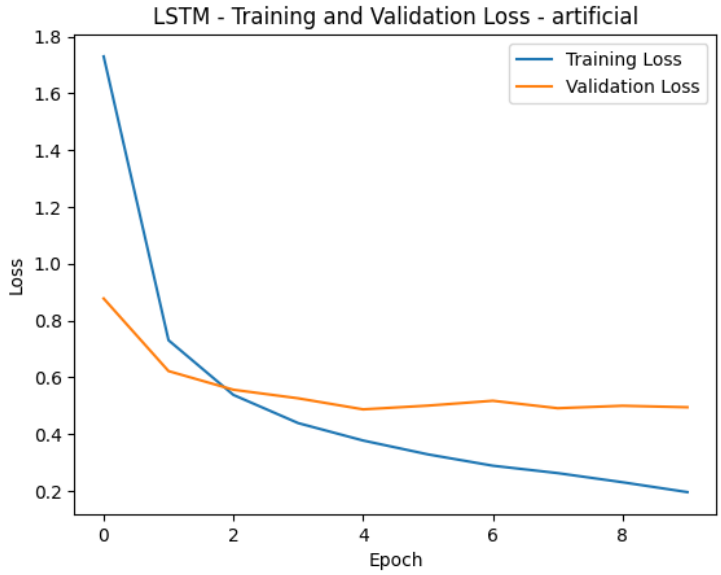

Figure 11: Final Loss (LSTM)

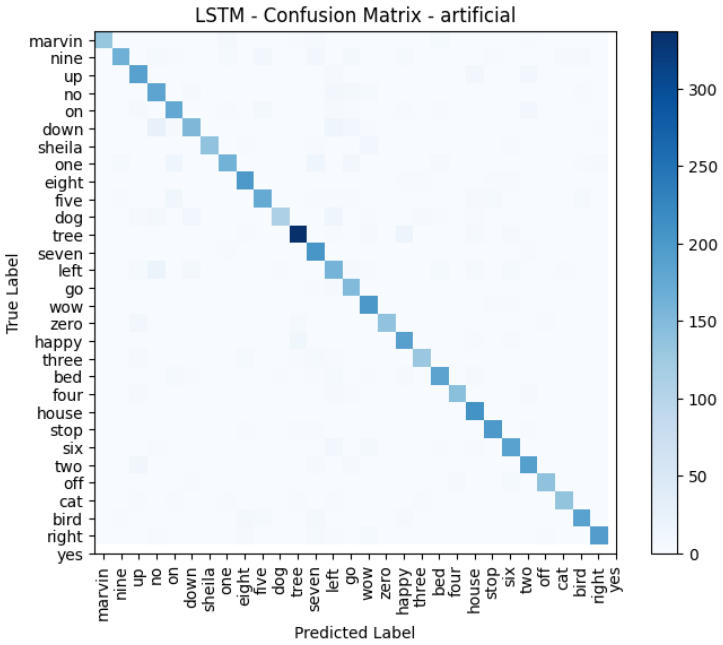

Figure 12: Final Confusion Matrix (LSTM)

Having those results LSTM with the same parameters was trained on a dataset with split background noise (split into 1-second clips) and then normalized by reducing the amount of background noise datapoints to an average number of datapoints in other classes. The results of such an experiment are visible in the figures 16, 17 and 18. A quick glance is enough to see that the model performs well, the confusion matrix holds the most intense colors on the diagonal indicating correct predictions, and other cells are very slightly more intense than in the confusion matrix from the previous experiment (figure 12), therefore one can conclude that in order to handle the unusual background noise class, one can simply split the .wav files to comply with the format of the files in other classes. Since the accuracy of the LSTM on the dataset with background noise is slightly worse than on the dataset without background noise one could build a separate network for the detection of such class, the dataset could then be filtered by the first network and passed to the beforementioned LSTM. In pursuit of such an idea, we have tried to construct another LSTM architecture to differentiate between "_background_noise_" and the rest of the classes, however, we have failed at our attempts, the constructed model overfitted no matter what parameters we have provided, a number of different batch sizes (32, 64, 128), optimizers (Adam, SGD, Lion), learning rates (0.01, 0.001, 0.0001) and epochs (even trying as little as 2 epochs) but none provided a much higher than 50%, which is definitely not enough for such task, the results are visible on figures although the loss and accuracy graphs look decent, confusion matrix shows that the model stubbornly predicts all datapoints as non background noise.

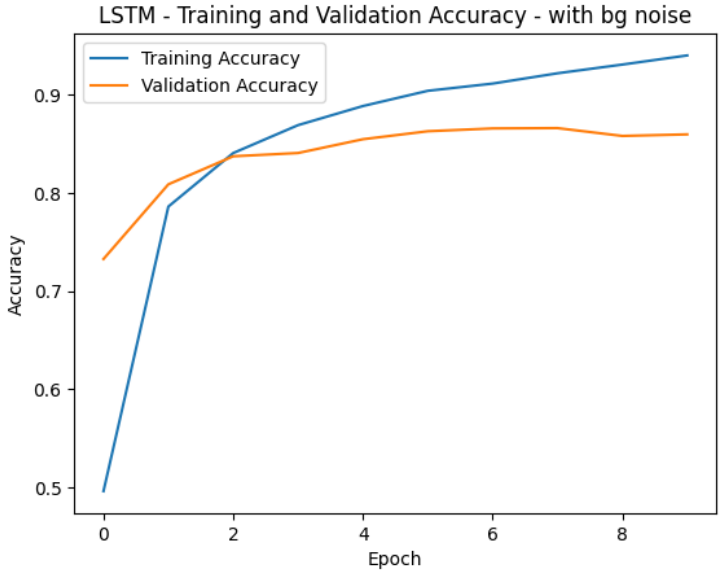

Figure 13: Final Accuracy (LSTM, split background noise)

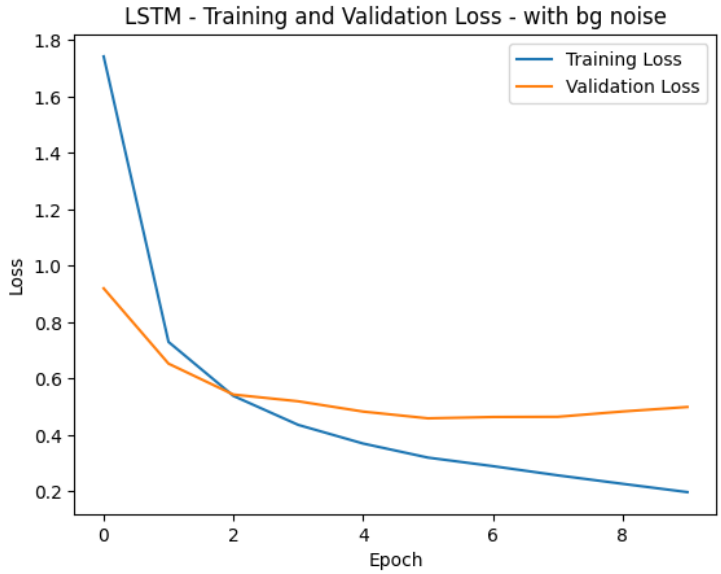

Figure 14: Final Loss (LSTM, split background noise)

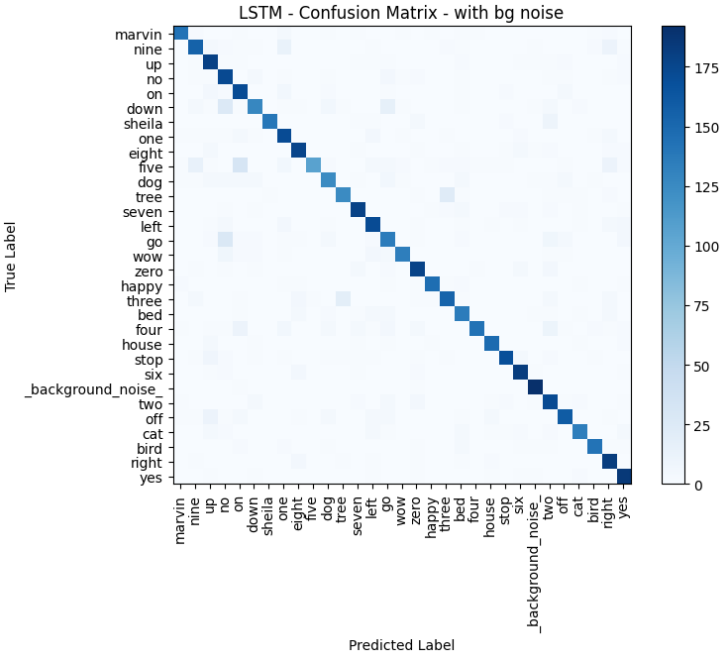

Figure 15: Final Confusion Matrix (LSTM, split background noise)

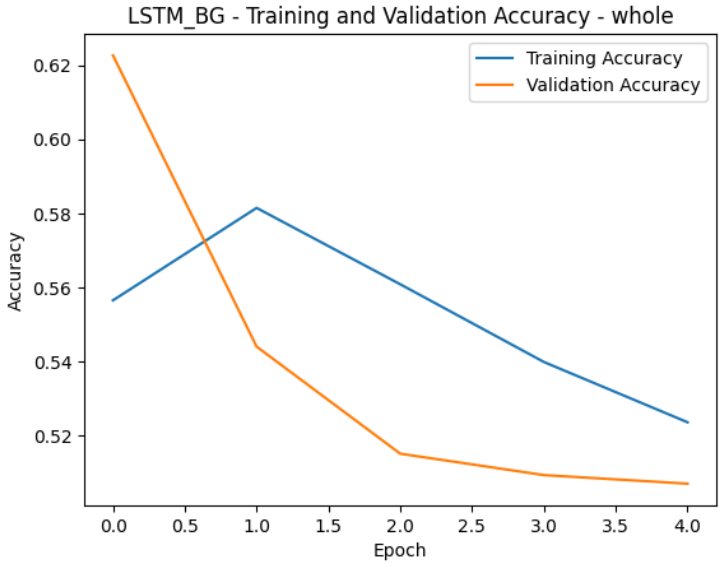

Figure 16: Final Accuracy (LSTM for background noise detection)

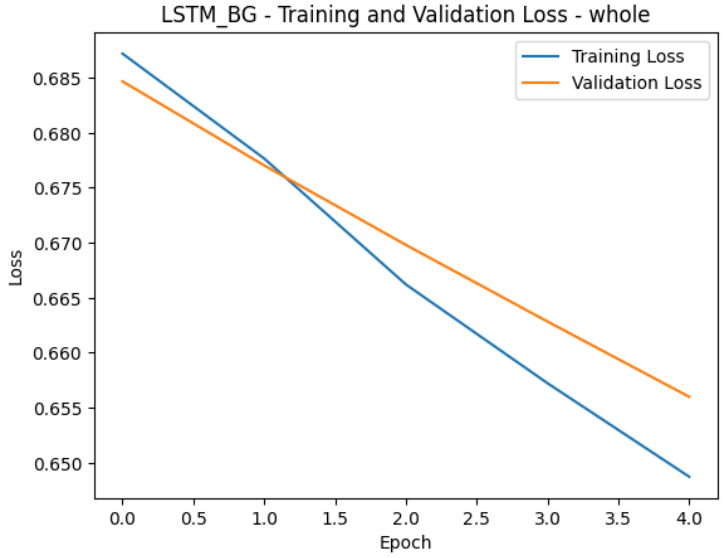

Figure 17: Final Loss (LSTM for background noise detection)

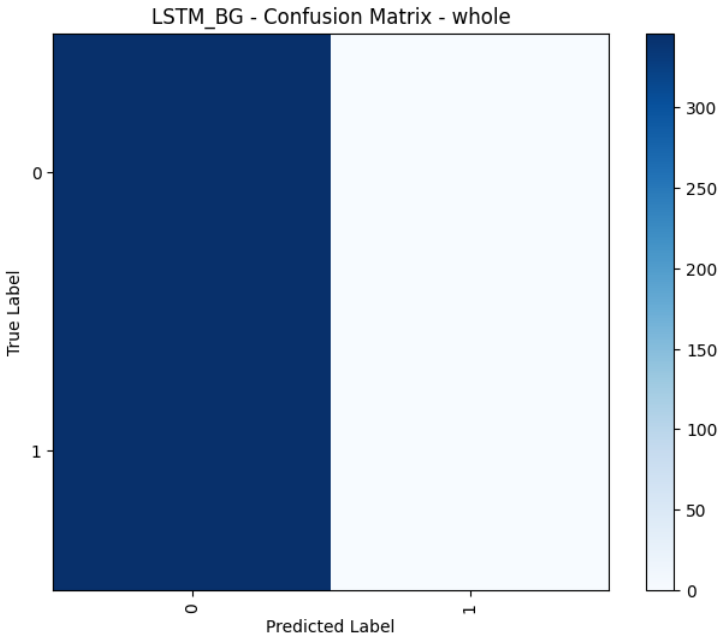

Figure 18: Final Confusion Matrix (LSTM for background noise detection)

Moreover, we have examined a slightly different LSTM model by getting rid of the convolutional layers, making the architecture more simple. For a dataset with background noise, the results are visible in figures 19, 20 and 21. It performs only slightly worse than the bigger LSTM model, making it a good candidate for a more lightweight alternative.

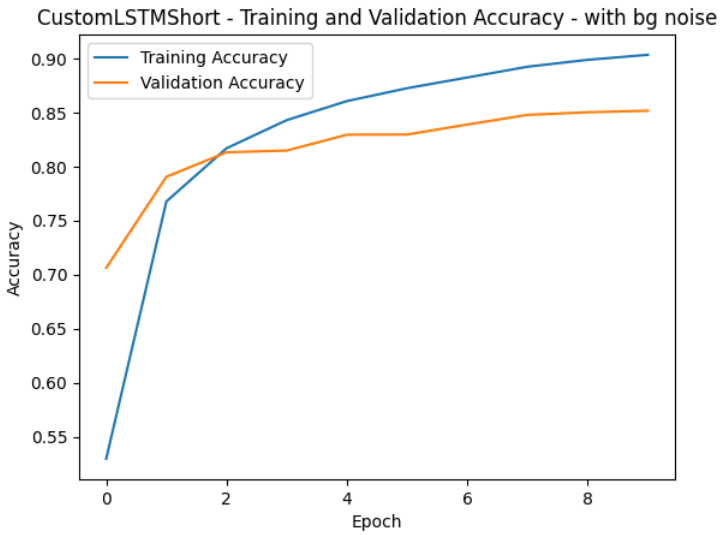

Figure 19: Final Confusion Matrix (LSTM simple)

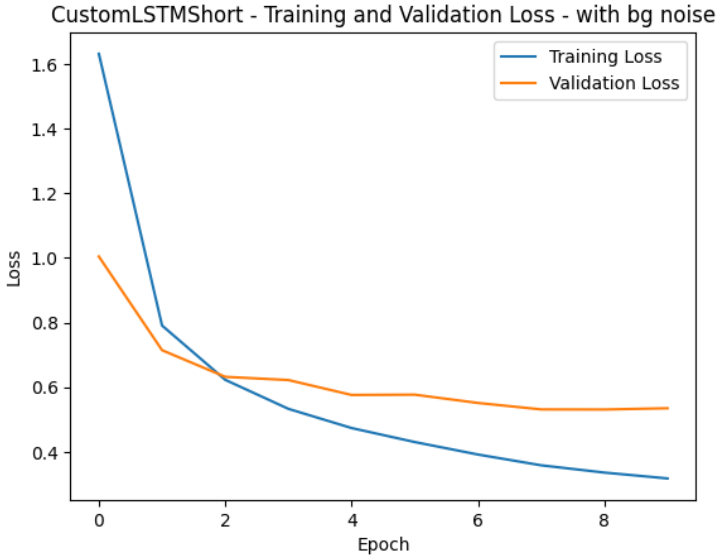

Figure 20: Final Confusion Matrix (LSTM simple)

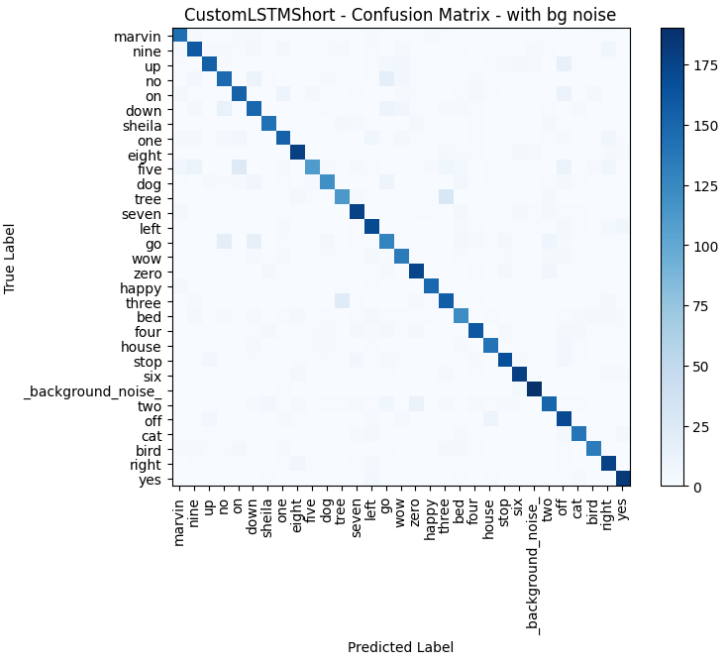

Figure 21: Final Confusion Matrix (LSTM simple)

We have also tested a transformer - Whisper, which is a pre-trained model created by OpenAI. On an artificially filtered dataset, it achieved an accuracy of 86.5%, and on a dataset with split background noise, the result was 84.2%. In comparison, LSTM on an artificially filtered dataset achieved an accuracy of 86.6% and on a dataset with split background noise 85.6%. These results are visible in the table 1 below. This shows that the created LSTM model outperforms Whisper on that particular task. Lastly, we have attempted to create a transformer of our own, however, due to technical difficulties we were unable to finalize that experiment.

	dataset without "_background_noise_" class	**dataset with split "_background_noise_" class**
Whisper	86.5%	84.2%
LSTM	86.6%	85.6%
LSTM simple	85.4%	83.8%

Table 1: Final Accuracies

6 Conclusions

We have successfully investigated the effects of the following parameters: batch size, learning rate, number of epochs, optimizer type, and test size. Many experiments have been performed to analyze their impact on the accuracy of the chosen models in the task of extracting text from speech. Having processed all the data from experiments we have chosen the most promising parameters with which to train and test the final models. We have obtained satisfactory results with LSTM with Whisper both on the dataset with and without background noise. However, we have failed to successfully create a model that would filter out the background noise, moreover, our attempts at creating our own transformer model turned out to be inconclusive. Lastly, certain matters could be investigated further, such as the comparison of Adam and Lion optimizers, since there is a possibility that a fine-tuned Lion optimizer could produce similar results with less training time.

List of Figures

List of Tables

References

[1] *Speech Commands Dataset*. URL: https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/data. (accessed: 12.05.2024).

[2] *LSTM*. URL: https://d2l.ai/chapter_recurrent-modern/lstm.html. (accessed: 12.05.2024).

[3] *What is a Transformer model*. URL: https://blogs.nvidia.com/blog/what-is-a-transformer-model/. (accessed: 12.05.2024).

[4] *Classification Performance*. URL: https://c3.ai/introduction-what-is-machine-learning/evaluating-model-performance/. (accessed: 26.03.2024).

[5] *Data Augmentation*. URL: https://aws.amazon.com/what-is/data-augmentation. (accessed: 26.03.2024).

[6] Jason Brownlee PhD. *Difference Between a Batch and an Epoch in a Neural Network*. URL: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/. (accessed: 26.03.2024).

[7] Sanket Doshi. *Various Optimization Algorithms For Training Neural Network*. URL: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6. (accessed: 26.03.2024).