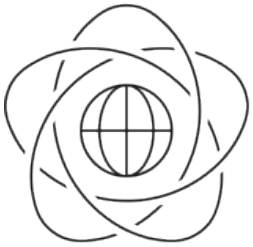

Wydział Matematyki
i Nauk Informacyjnych

Politechnika Warszawska

Deep Learning, Deep Learning Methods

Project 2: Transformers

Mikołaj Rzepiński, Krzysztof Wolny
student record book number 276890, 305765

supervisor
Zinelabidine Leghelimi

WARSAW April 30, 2025

Contents

1. Introduction . . . 3
2. Theoretical Background and Literature Review . . . 3
2.1. Transformers for Audio . . . 3
2.2. Models . . . 3
2.3. Data Augmentation . . . 4
2.4. Hyperparameter Experiments . . . 4
2.5. Literature and resources . . . 4
3. Methodology and Experimentation . . . 5
3.1. Hyperparameters . . . 5
3.2. Data Augmentation . . . 5
3.3. Fine-tuning vs. Training from Scratch . . . 6
3.4. Evaluation Metrics . . . 6
4. Results and Discussion . . . 6
4.1. Parameters . . . 9
4.1.1. Learning Rate . . . 9
4.1.2. Batch Size . . . 11
4.1.3. Augmentation . . . 13
4.2. Compare results of best models . . . 16
4.3. Compare results of models trained on all data . . . 19
5. Conclusions and Future Work . . . 23
5.1. Conclusions . . . 23
5.2. Future Work . . . 23
6. Application Instruction . . . 23
References . . . 25

1. Introduction

The primary objective of this project was to compare the performance of several deep learning architectures on the audio classification task using the Speech Commands dataset. Beyond simple benchmarking, we focused on understanding how key training components (hyperparameters and data augmentations) impact overall model performance in practice.

We used these three models:

- LSTM-CNN network,
- Wav2Vec2.0,
- Audio Spectrogram Transformer (AST).

AST and Wav2Vec2.0 models were initialized with pretrained weights and fine-tuned with frozen feature extractors. The LSTM-CNN network was trained from scratch without any pretraining. All experiments were conducted using the TensorFlow framework and selected Hugging Face libraries within the Google Colab environment.

2. Theoretical Background and Literature Review

This project is based on modern ideas in deep learning, especially on the use of transformers for audio classification. Before starting the experiments, we studied several important research papers and existing solutions that helped us choose the models, training methods, and augmentation strategies.

2.1. Transformers for Audio

Transformers are a special type of deep learning model that were originally created for language tasks but are now used for many different types of data, including images and audio. Instead of just looking at small parts of the input like CNNs do, transformers can look at the whole input at once and find long-range patterns. This makes them very powerful for audio signals, where understanding context over time is important.

2.2. Models

In our project, we tested three different models:

- Wav2Vec2.0: A model that processes raw audio signals directly without converting them into images first. It was pre-trained on a lot of unlabeled audio data and can be fine-tuned for specific tasks with much less labeled data. It is widely used for speech recognition.
- LSTM-CNN Network: A combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. CNN layers find patterns in the audio

features, and the LSTM layers help the model remember important information over time. This model was trained from scratch.

- Audio Spectrogram Transformer (AST): A transformer-based model designed specifically for audio data. It works on spectrograms, which are pictures showing how sounds change over time. AST is a strong model because it uses the same ideas that made transformers successful in text and vision tasks.

2.3. Data Augmentation

Data augmentation means making small changes to the training data to help the model learn better and become more robust. In this project, we used two techniques:

- **Background Noise Injection (BN):** Adding random noise to the audio samples, making the model better at handling real-world noisy data.
- **Time Masking (TM):** Randomly hiding parts of the audio signal over time, helping the model focus on important patterns instead of relying too much on specific moments.

2.4. Hyperparameter Experiments

We studied how two key hyperparameters affect the models:

- **Learning Rate (LR):** Controls how fast the model updates during training. Too high can make training unstable, too low can make it slow.
- **Batch Size (BS):** The number of samples processed at once. Larger batches can make training faster but require more memory.

Each model was tested with different learning rates and batch sizes to find the best training setup.

2.5. Literature and resources

The following research papers and resources helped us choose the models, understand their architectures, and design our training process:

- **AST: Audio Spectrogram Transformer[1]**
  - Huggingface pretrained model: Huggingface Documentation
- **Wav2Vec[2]**
  - Huggingface pretrained model: facebook/wav2vec2-base
  - Tensorflow Model
- **LSTM-CNN Network[3]**
  - Helpful article: Machine Learning Mastery
  - TensorFlow layer documentation: LSTM Layer API

These resources guided us in model selection, augmentation strategies, and training methodology.

3. Methodology and Experimentation

In this chapter, we explain how we trained our models and what steps we followed. We tested different settings to see what works best. This helped us learn how model architecture, hyperparameters, and data augmentations affect the results.

All experiments in this project were done using the Speech Commands dataset, which contains short audio recordings of spoken words from different classes like `yes`, `no`, `stop`, and `go`. It is a good test for models that classify simple speech commands.

Models were initially trained on data with four labels: `yes`, `no`, `stop`, and `go`. Each label had 1000 examples. Models were trained on such data with different hyperparameters and data augmentations. After evaluating model performances, the ones with highest accuracy were selected to train on full dataset with all labels. Data was split during all phases with proportions: 0.7 to training, 0.15 to validation, 0.15 to testing.

We worked with three models following this plan:

- AST and Wav2Vec2.0 models were initialized with pretrained weights and fine-tuned with frozen feature extractors.
- LSTM-CNN model was trained from scratch without any pretraining.
- All models were trained and evaluated using the TensorFlow framework and run on Google Colab.

Each model was trained for 10 epochs, and we monitored accuracy and loss during training and validation.

3.1. Hyperparameters

Hyperparameters are settings we choose before training the model. They control how fast and how well the model learns. We tested different values for key hyperparameters to find which settings give the best results.

Here are the hyperparameters we tested:

- Learning Rate (LR): Controls how fast the model updates its weights. We tested values such as 0.0001, 0.001, and 0.01.
- Batch Size (BS): Defines how many audio samples the model processes at once. We tested batch sizes like 32 and 64.

We tested all combinations (including all augmentations) on simplified data. Then for best configuration for each model we tested them on full data.

3.2. Data Augmentation

Data augmentation means changing the audio recordings in small ways to make the training set more varied. This helps the models become more robust to real-world conditions.

We tested the following augmentations:

- Background Noise Injection (BN): Adding random noise to the audio to simulate noisy environments.
- Time Masking (TM): Randomly hiding small sections of the audio signal over time to make the models less dependent on specific sounds.
- No augmentation

3.3. Fine-tuning vs. Training from Scratch

The models were handled differently depending on their architecture:

- AST and Wav2Vec2.0 were fine-tuned: we froze the backbone weights and only trained the final classification layers.
- LSTM-CNN was trained from scratch: no pretrained weights were used, and the whole model learned from random initialization.

This allowed us to compare the benefits of using pretrained models versus training a model completely on our own data.

3.4. Evaluation Metrics

We used several methods to evaluate model performance:

- Accuracy: The percentage of correct predictions.
- Per-class Accuracy: How well the model performs for each individual word class.
- Confusion Matrix: A table showing where the models made mistakes, helping us see which words were often confused.
- Training and Validation Curves: Graphs showing how the model's performance changed over time during training.

All evaluations were performed on the same validation set to ensure fair comparisons between models and settings.

4. Results and Discussion

This section presents an analysis of the results obtained from experiments. In the first part of our project the models were trained using the following parameters:

- **Learning Rate:** 0.01, 0.001, 0.0001
- **Batch Sizes:** 32, 64
- **Data augmentation:** None, BN, TM
- **Loss Function:** Cross-entropy
- **Optimizer:** Adam

In total each model was trained 18 times. The results of the training are presented in table 4.1.

Next there were selected parameters with best accuracy score for each model. These three models were trained on data with all labels and evaluated **??**.

Table 4.1. Comparison of Model Performance Across Hyperparameters and Augmentations

Model	ID	Learning Rate	Batch Size	Augmentation	Accuracy
Wav2Vec	0	0.01	32	None	0.927
Wav2Vec	1	0.01	64	None	0.938
Wav2Vec	2	0.001	32	None	0.958
Wav2Vec	3	0.001	64	None	0.945
Wav2Vec	4	0.0001	32	None	0.863
Wav2Vec	5	0.0001	64	None	0.825
Wav2Vec	6	0.01	32	BN	0.897
Wav2Vec	7	0.01	64	BN	0.910
Wav2Vec	8	0.001	32	BN	0.905
Wav2Vec	9	0.001	64	BN	0.918
Wav2Vec	10	0.0001	32	BN	0.848
Wav2Vec	11	0.0001	64	BN	0.822
Wav2Vec	12	0.01	32	TM	0.912
Wav2Vec	13	0.01	64	TM	0.928
Wav2Vec	14	0.001	32	TM	0.940
Wav2Vec	15	0.001	64	TM	0.933
Wav2Vec	16	0.0001	32	TM	0.818
Wav2Vec	17	0.0001	64	TM	0.782
AST	0	0.01	32	None	0.867
AST	1	0.01	64	None	0.868
AST	2	0.001	32	None	0.848
AST	3	0.001	64	None	0.847
AST	4	0.0001	32	None	0.748
AST	5	0.0001	64	None	0.698
AST	6	0.01	32	BN	0.865
AST	7	0.01	64	BN	0.888
AST	8	0.001	32	BN	0.883
AST	9	0.001	64	BN	0.868
AST	10	0.0001	32	BN	0.792
AST	11	0.0001	64	BN	0.768
AST	12	0.01	32	TM	0.813
AST	13	0.01	64	TM	0.858
AST	14	0.001	32	TM	0.830
AST	15	0.001	64	TM	0.815
AST	16	0.0001	32	TM	0.722
AST	17	0.0001	64	TM	0.703
LSTM	0	0.01	32	None	0.225
LSTM	1	0.01	64	None	0.237
LSTM	2	0.001	32	None	0.842
LSTM	3	0.001	64	None	0.860
LSTM	4	0.0001	32	None	0.728
LSTM	5	0.0001	64	None	0.783
LSTM	6	0.01	32	BN	0.237
LSTM	7	0.01	64	BN	0.703
LSTM	8	0.001	32	BN	0.885
LSTM	9	0.001	64	BN	0.895
LSTM	10	0.0001	32	BN	0.755
LSTM	11	0.0001	64	BN	0.688
LSTM	12	0.01	32	TM	0.455
LSTM	13	0.01	64	TM	0.640
LSTM	14	0.001	32	TM	0.893
LSTM	15	0.001	64	TM	0.837
LSTM	16	0.0001	32	TM	0.502
LSTM	17	0.0001	64	TM	0.722

4.1. Parameters

4.1.1. Learning Rate

Table 4.4 presents the best test accuracies for three models—Wav2Vec, AST, and LSTM—across different learning rates. The Wav2Vec model achieves its highest accuracy of 95.8% at a learning rate of 0.001. The AST model performs best at a learning rate of 0.01 with an accuracy of 88.8%, while lower learning rates result in slightly smaller accuracy. The LSTM model also have the best accuracy at a learning rate of 0.001 with an accuracy of 89.5%, and shows significantly lower performance at learning rates of 0.01 (70.3%) and 0.0001 (78.3%). Overall, a learning rate of 0.001 appears optimal for Wav2Vec and LSTM, whereas AST performs best with a learning rate of 0.01.

Figures 4.1, 4.2, 4.3 show the training loss and accuracy of the Wav2Vec during the training process using a learning rate of 0.01, 0.001 and 0.0001. Batch size in all plots is set to 64 and there is no data augmentation. Figure 4.1 with 0.01 learning rate display sharp drop in loss, but it quickly flattens indicating that learning rate is too high. In figure 4.2 learning rate is converging slower and in last epoch it looks that it starts flattening. The smallest learning rate 4.3 was too small to get to optimal result. It would require training model on more epoch to find out whether it can get to better accuracy than other values.

Table 4.2. Best test accuracy for different learning rates

Model	LR=0.01	LR=0.001	LR=0.0001
Wav2Vec	0.938	0.958	0.863
AST	0.888	0.883	0.792
LSTM	0.703	0.895	0.783

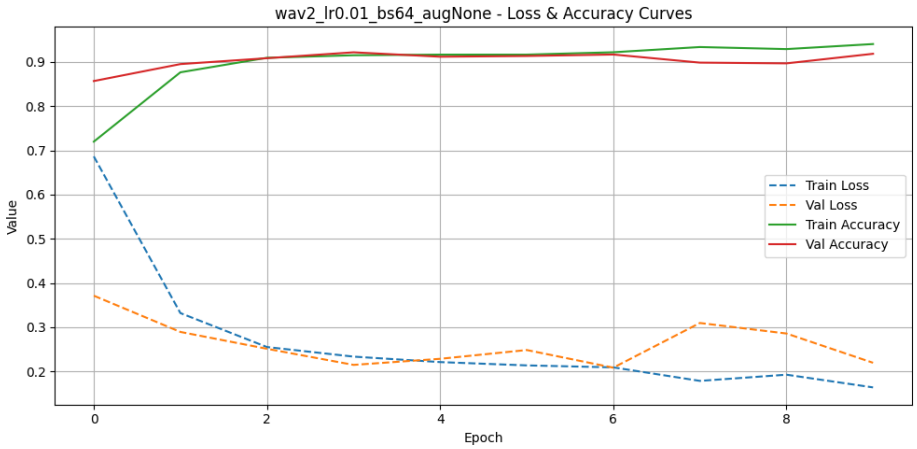

Figure 4.1. Wav2Vec learning curve with learning rate = 0.01, batch size = 64, and no data augmentation.

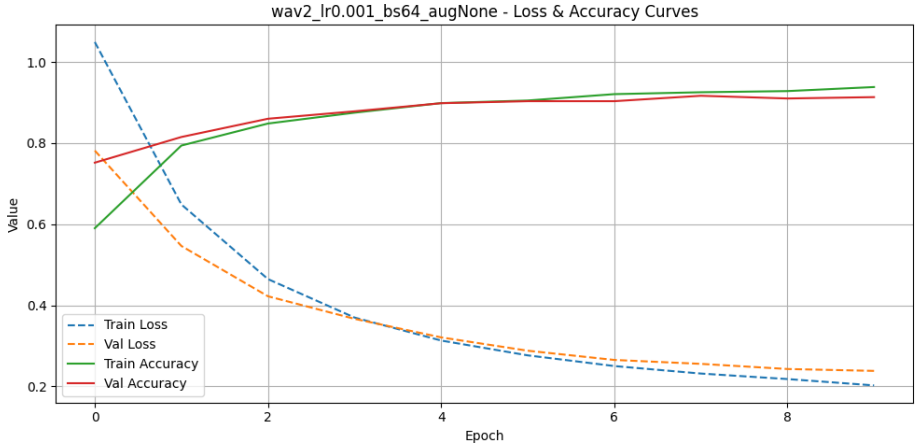

Figure 4.2. Wav2Vec learning curve with learning rate = 0.001, batch size = 64, and no data augmentation.

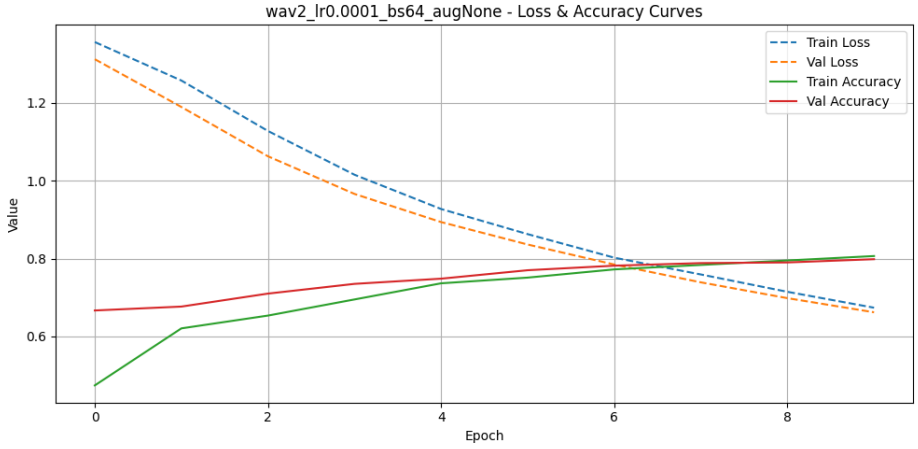

Figure 4.3. Wav2Vec learning curve with learning rate = 0.0001, batch size = 64, and no data augmentation.

4.1.2. Batch Size

Table 4.3 summarizes the best test accuracies of three models—Wav2Vec, AST, and LSTM—for batch sizes of 32 and 64. Wav2Vec achieves higher accuracy with a batch size of 32 (95.8%) compared to 64 (94.5%). In contrast, AST slightly improves with a larger batch size, increasing from 88.3% (BS=32) to 88.8% (BS=64). LSTM shows a small gain, with accuracy rising from 89.3% to 89.5%. Overall, Wav2Vec performs best with a smaller batch size, while AST and LSTM benefit slightly from a larger batch size.

Figures 4.4, 4.5, 4.6 displays the training loss and accuracy of the models during the training process, using learning rate of 0.001, batch size of 32 and no data augmentation. All models show proper convergence, with loss decreasing and accuracy increasing. Notably, the Wav2Vec model have a much steeper drop in loss than the other models. Meanwhile, AST and LSTM models converge at a similar pace. Although the models have reached high accuracy, there is still a reasonable assumption that further training with additional epochs can increase the results of models.

Table 4.3. Best test accuracy for different batch sizes

Model	BS=32	BS=64
Wav2Vec	0.958	0.945
AST	0.883	0.888
LSTM	0.893	0.895

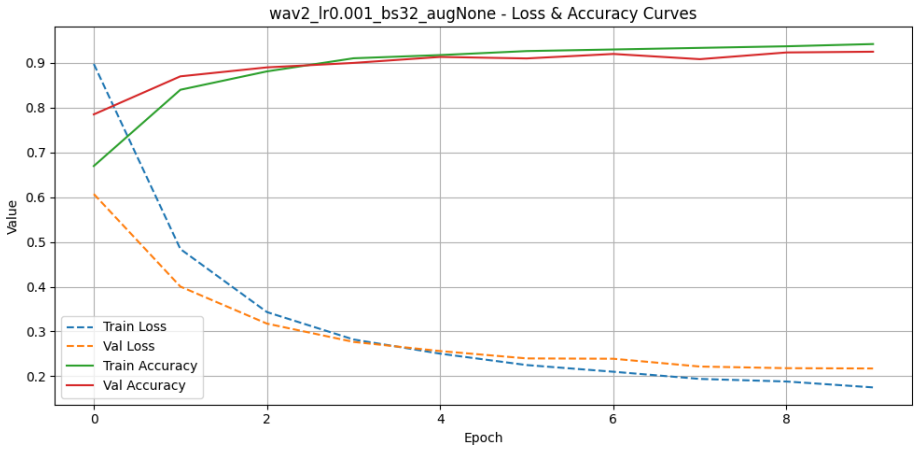

Figure 4.4. Wav2Vec learning curve with learning rate = 0.001, batch size = 32, and no data augmentation.

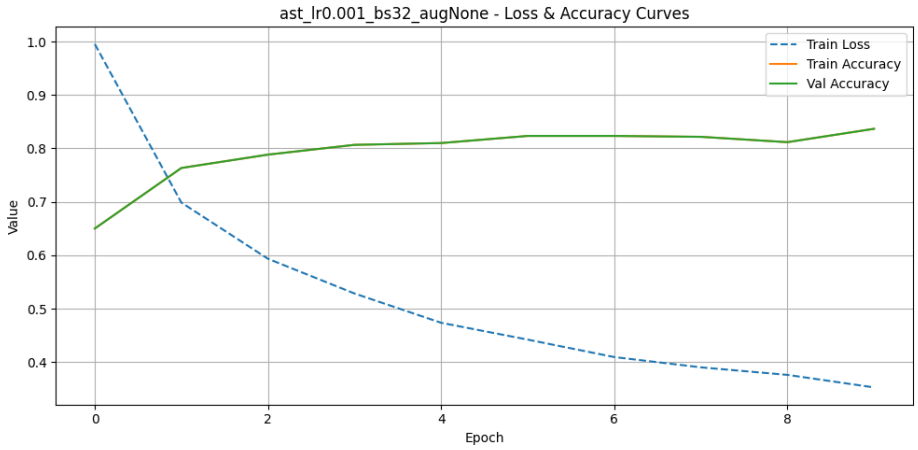

Figure 4.5. AST learning curve with learning rate = 0.001, batch size = 32, and no data augmentation.

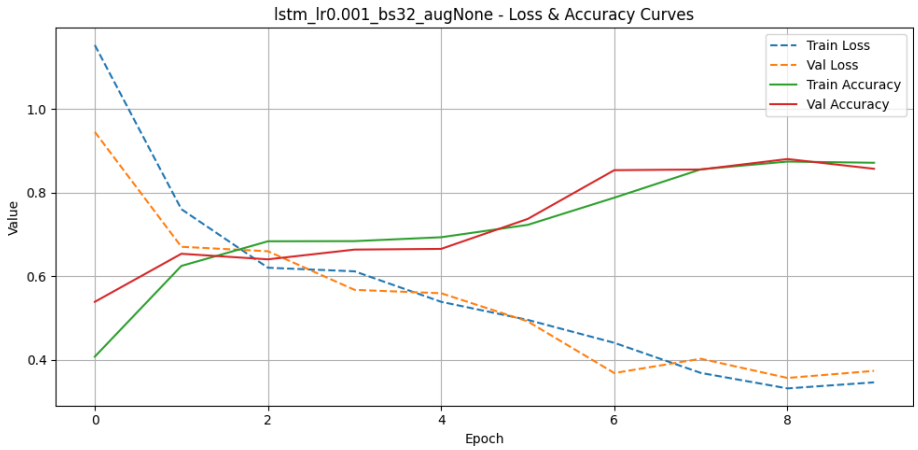

Figure 4.6. LSTM learning curve with learning rate = 0.001, batch size = 32, and no data augmentation.

4.1.3. Augmentation

Table 4.4 presents the best test accuracies for three models—Wav2Vec, AST, and LSTM—under different data augmentation strategies: None, Background Noise (BN), and Time Noise (TN). Wav2Vec achieves its highest accuracy without data augmentation (95.8%), while both BN and TN slightly reduce its performance. AST performs best with BN (88.8%). It then outperforms both no augmentation (86.8%) and TN (85.8%). LSTM benefits most from data augmentation, with BN (89.5%) and TN (89.3%) both improving over no augmentation (86.0%). Overall, BN augmentation is particularly beneficial for AST and LSTM, while Wav2Vec performs best without augmentation.

Three augmentation strategies were compared using confusion matrices: Batch Normalization4.8, No Augmentation4.7, and Time Masking4.9. All models performed best on the `on` class (up to 0.96 accuracy), while the `off` class consistently showed the highest confusion. `augNone` achieved the highest `yes` accuracy (0.86) but performed worst on `off` (0.80). `augBN` and `augTM` balanced performance better across classes, with augBN excelling in `no` (0.92) and augTM reducing confusion more evenly. Overall, augmentation impacts class-specific accuracy and error distribution significantly.

Table 4.4. Best test accuracy for different data augmentations

Model	DA=None	DA=BN	DA=TN
Wav2Vec	0.958	0.918	0.940
AST	0.868	0.888	0.858
LSTM	0.860	0.895	0.893

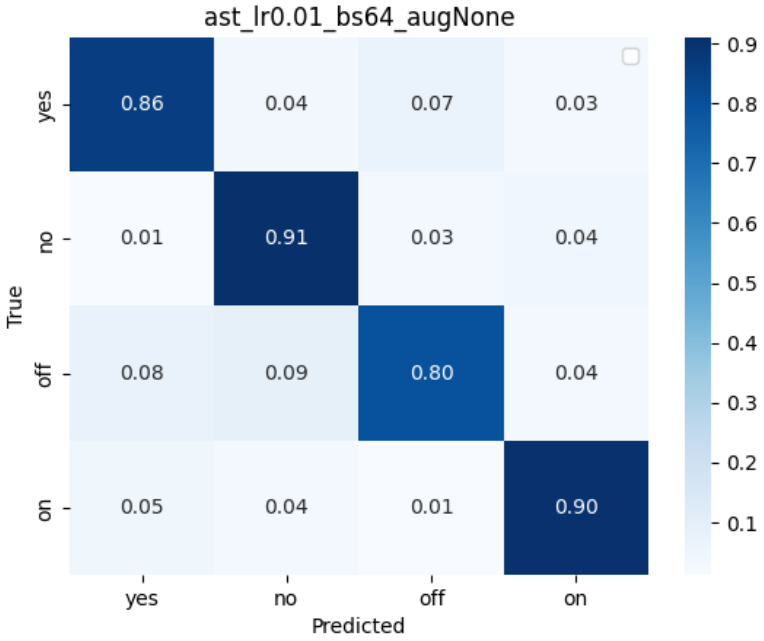

Figure 4.7. Confussion matrix of AST results with learning rate = 0.01, batch size = 64, and no data augmentation.

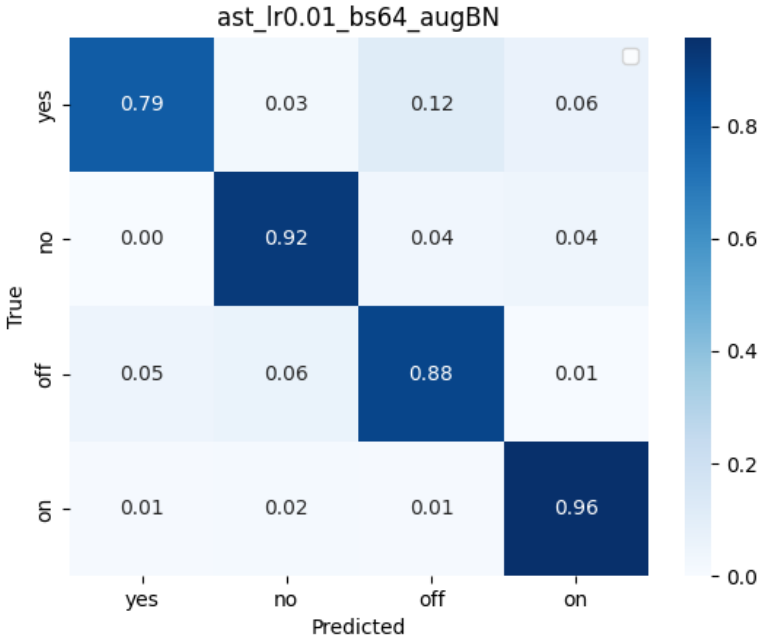

Figure 4.8. Confussion matrix of AST results with learning rate = 0.01, batch size = 64, and BN data augmentation.

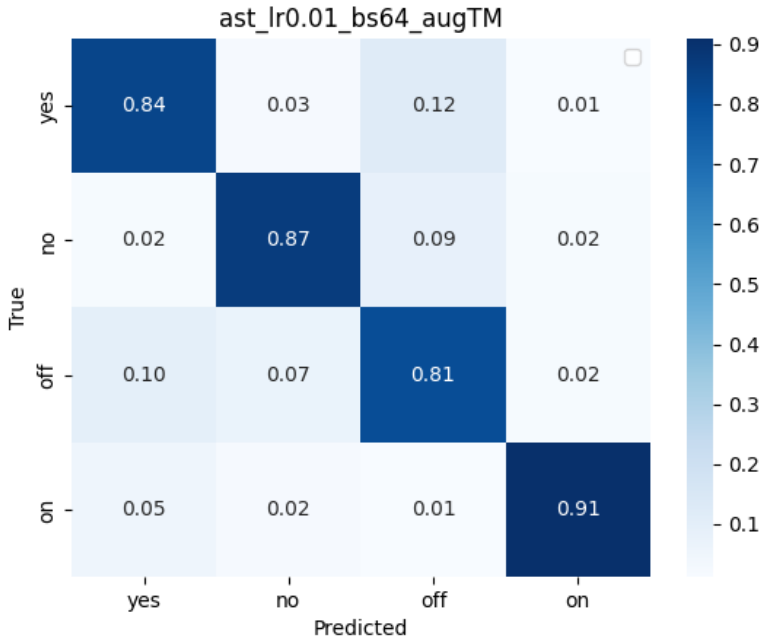

Figure 4.9. Confussion matrix of AST results with learning rate = 0.01, batch size = 64, and TM data augmentation.

4.2. Compare results of best models

Table 4.6 4.10 reports the best-performing configurations for each model on the 4-label dataset. The Wav2Vec model achieved the highest accuracy of 95.8% using a learning rate of 0.001, batch size of 32 and no data augmentation. The LSTM model performed best with a learning rate of 0.001, batch size of 64 and background noise (BN) augmentation. It reached an accuracy of 89.5%. AST model obtained similar accuracy of 88.8% , using a learning rate of 0.01, batch size of 64 and BN augmentation.

Analysis based on the confusion matrix shows that the Wav2Vec model was the most effective at classifying between all word classes. It achieved over 94% accuracy for each label 4.11. AST model struggled to correctly classify the words *yes* and *off*, leading to lower overall performance 4.12. The LSTM model demonstrated strong performance on the *on* and *yes* classes, both exceeding 90% accuracy 4.13. However, it was confusing the *no* and *off* classes, which contributed to a drop in these labels accuracy to approximately 86%.

Table 4.5. Best performing configuration for each model for 4 label dataset

Model	Learning Rate	Batch Size	Augmentation	Accuracy
AST	0.01	64	BN	0.888
LSTM	0.001	64	BN	0.895
Wav2Vec	0.001	32	None	0.958

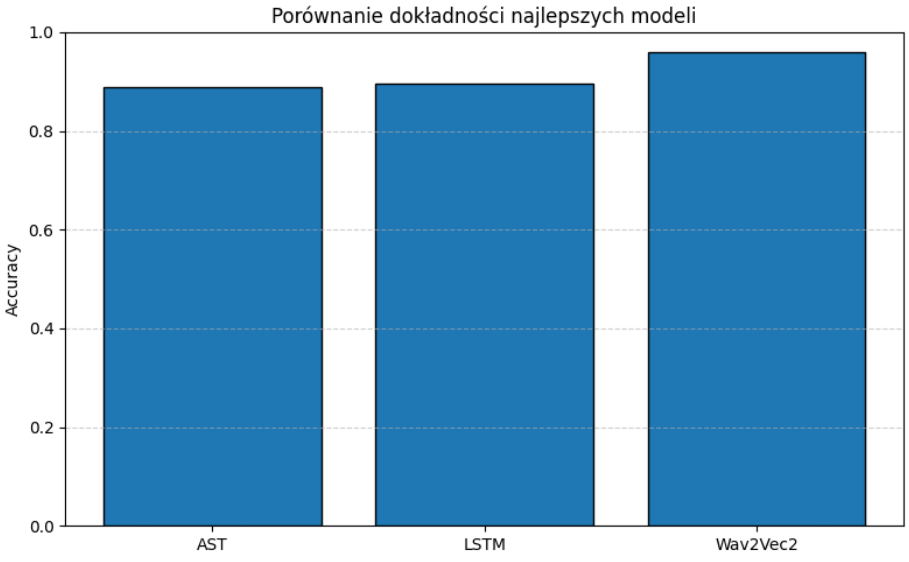

Figure 4.10. Accuracy for best models

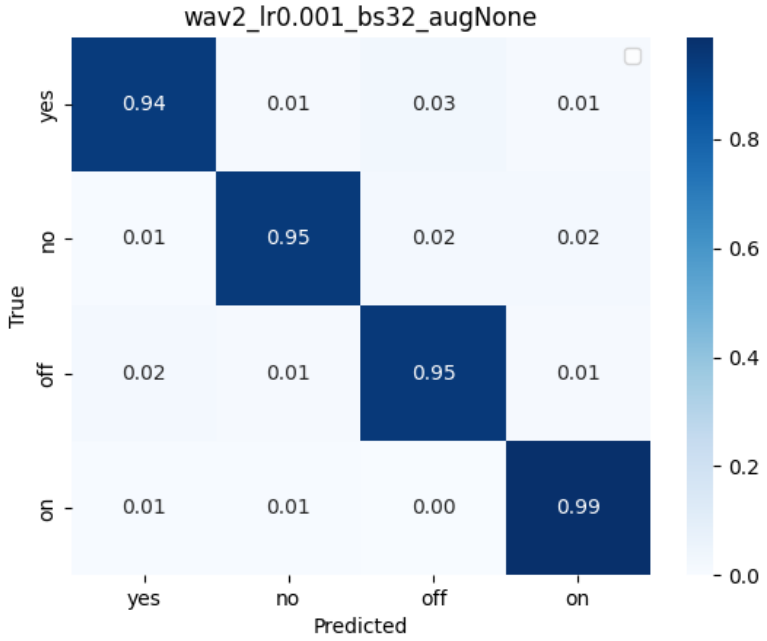

Figure 4.11. Best Wav2Vec model confusion matrix

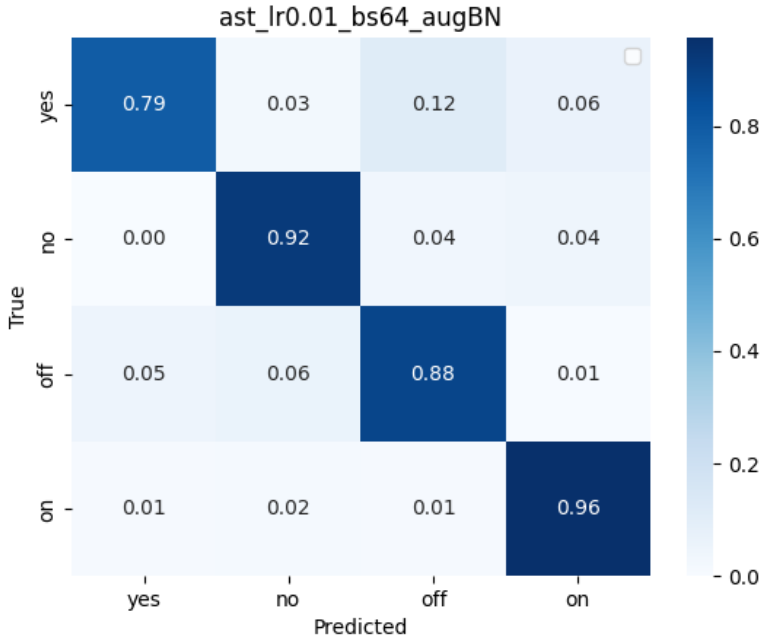

Figure 4.12. Best AST model confusion matrix

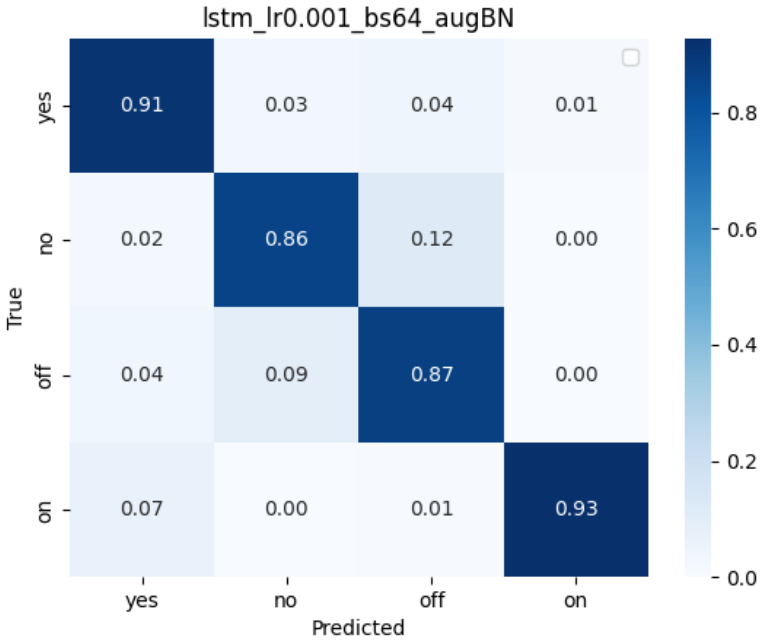

Figure 4.13. Best LSTM model confusion matrix

4.3. Compare results of models trained on all data

Table 4.6 4.14 presents the best-performing configurations for each model when trained on the full dataset. Wav2Vec achieved the highest accuracy of 86.7% using a learning rate of 0.001, batch size of 32 and no data augmentation. The LSTM model followed with an accuracy of 81.3%, using a learning rate of 0.001, batch size of 6 and background noise (BN) augmentation. The AST model had best accuracy of 67.0% , using a learning rate of 0.01, batch size of 64 and BN augmentation. These results indicate that Wav2Vec consistently outperforms the other models on the full dataset. It also indicate that augmentation strategies and learning rates play an important role in optimizing performance.

Figures 4.15, 4.16, and 4.17 show the confusion matrices for Wav2Vec, AST, and LSTM models, respectively. Wav2Vec achieves the highest classification accuracy, with most predictions tightly concentrated along the diagonal. AST struggles more with class separation, especially among phonetically similar words like *"three"* and *"tree"*. LSTM performs better than AST, though some confusion remains in similar cases. Overall, Wav2Vec demonstrates the most robust and consistent performance across all word classes.

Table 4.6. Best Performing Configurations for Each Model on all data

Model	Learning Rate	Batch Size	Augmentation	Accuracy
AST	0.01	64	BN	0.670
LSTM	0.001	64	BN	0.813
Wav2Vec	0.001	32	None	0.867

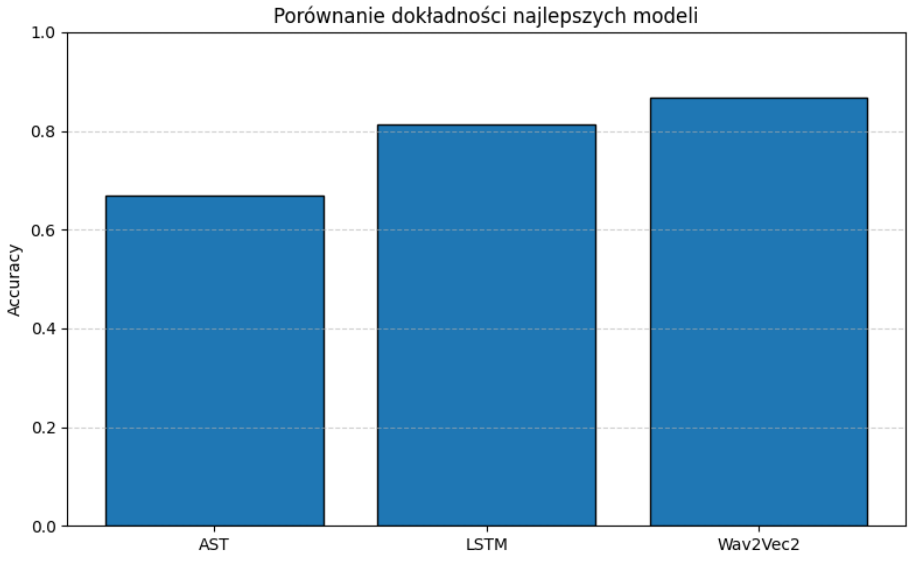

Figure 4.14. Accuracy for each model trained on all available labels

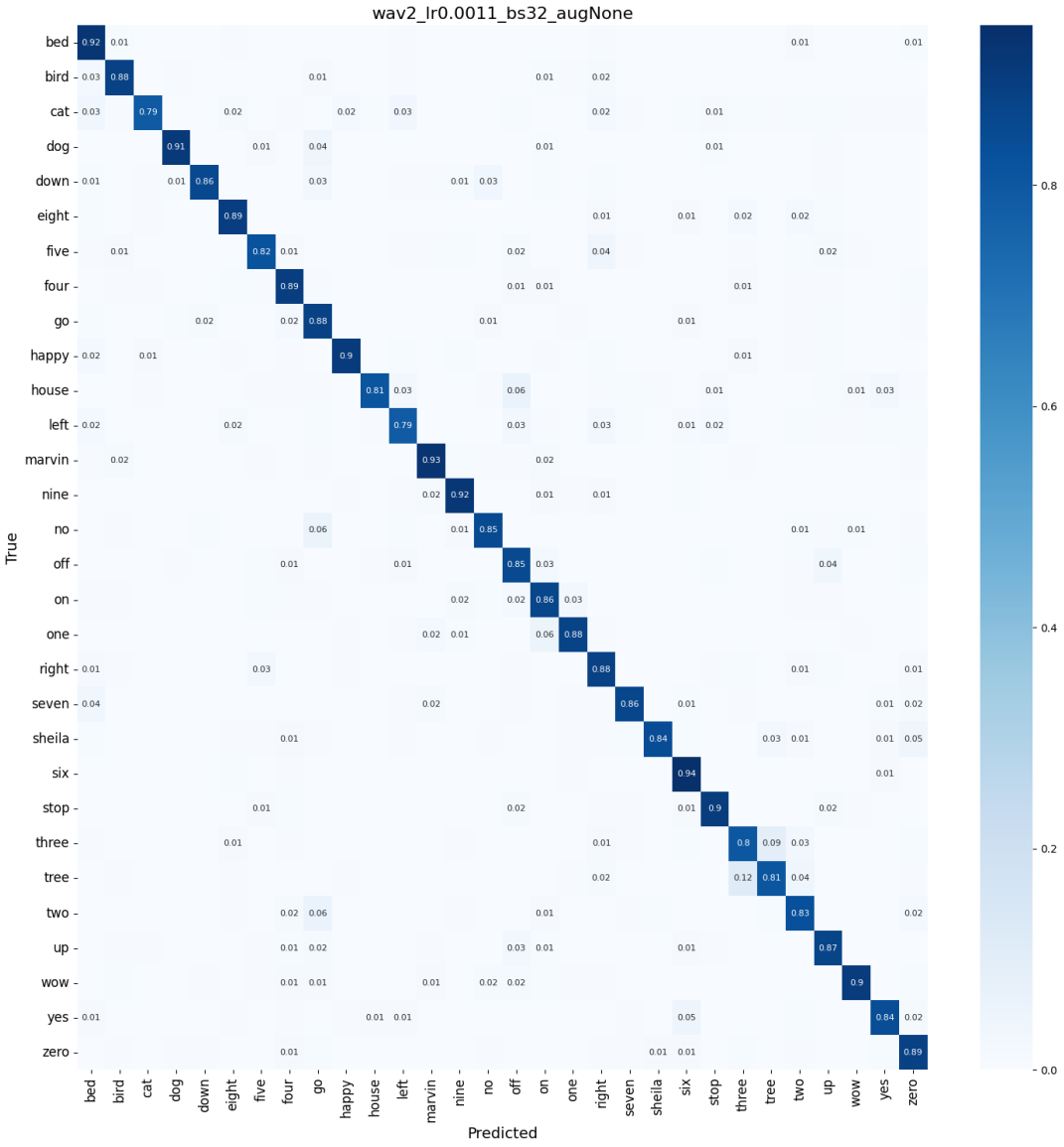

Figure 4.15. Best Wav2Vec model confusion matrix for all label dataset

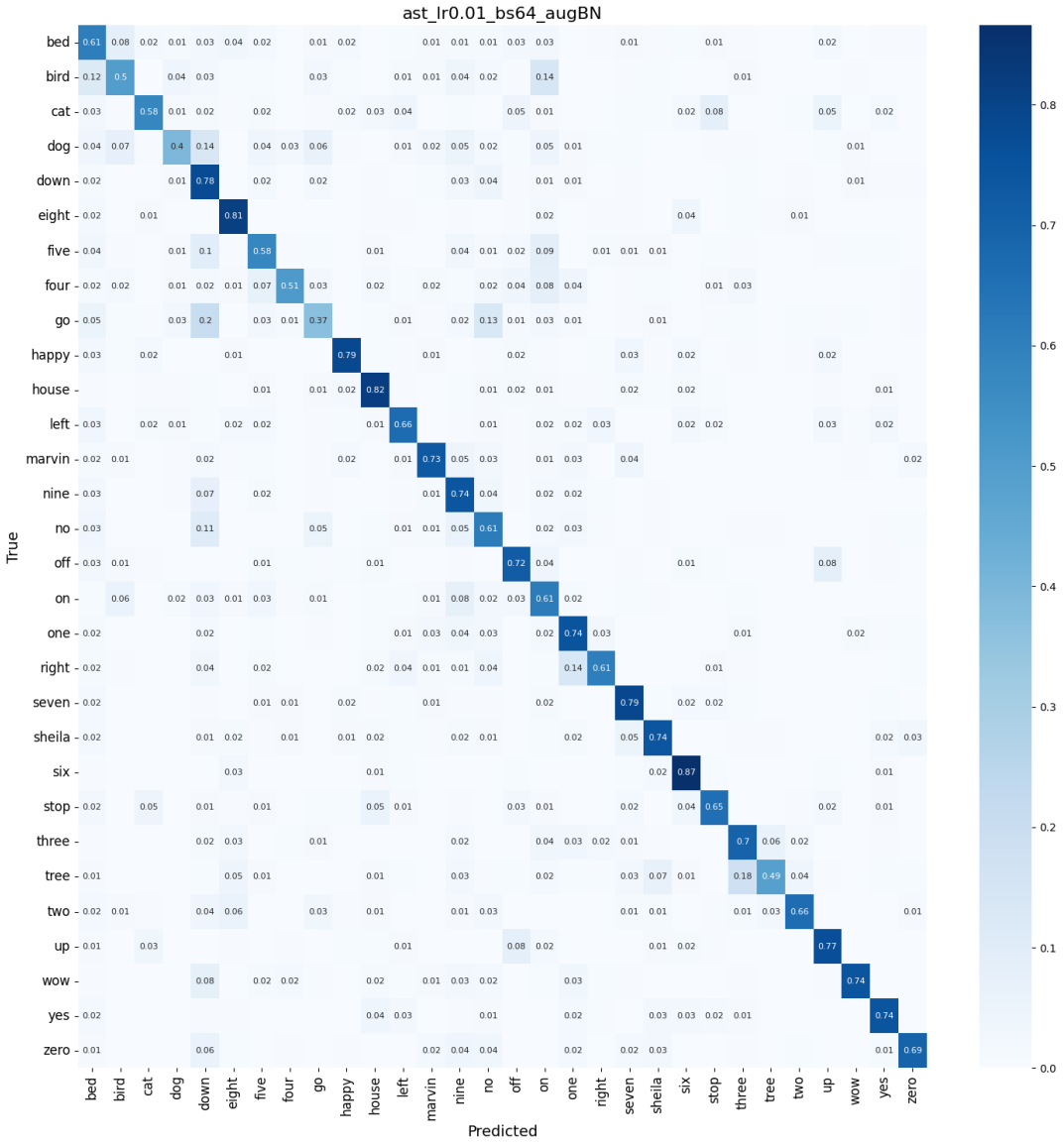

Figure 4.16. Best AST model confusion matrix for all label dataset

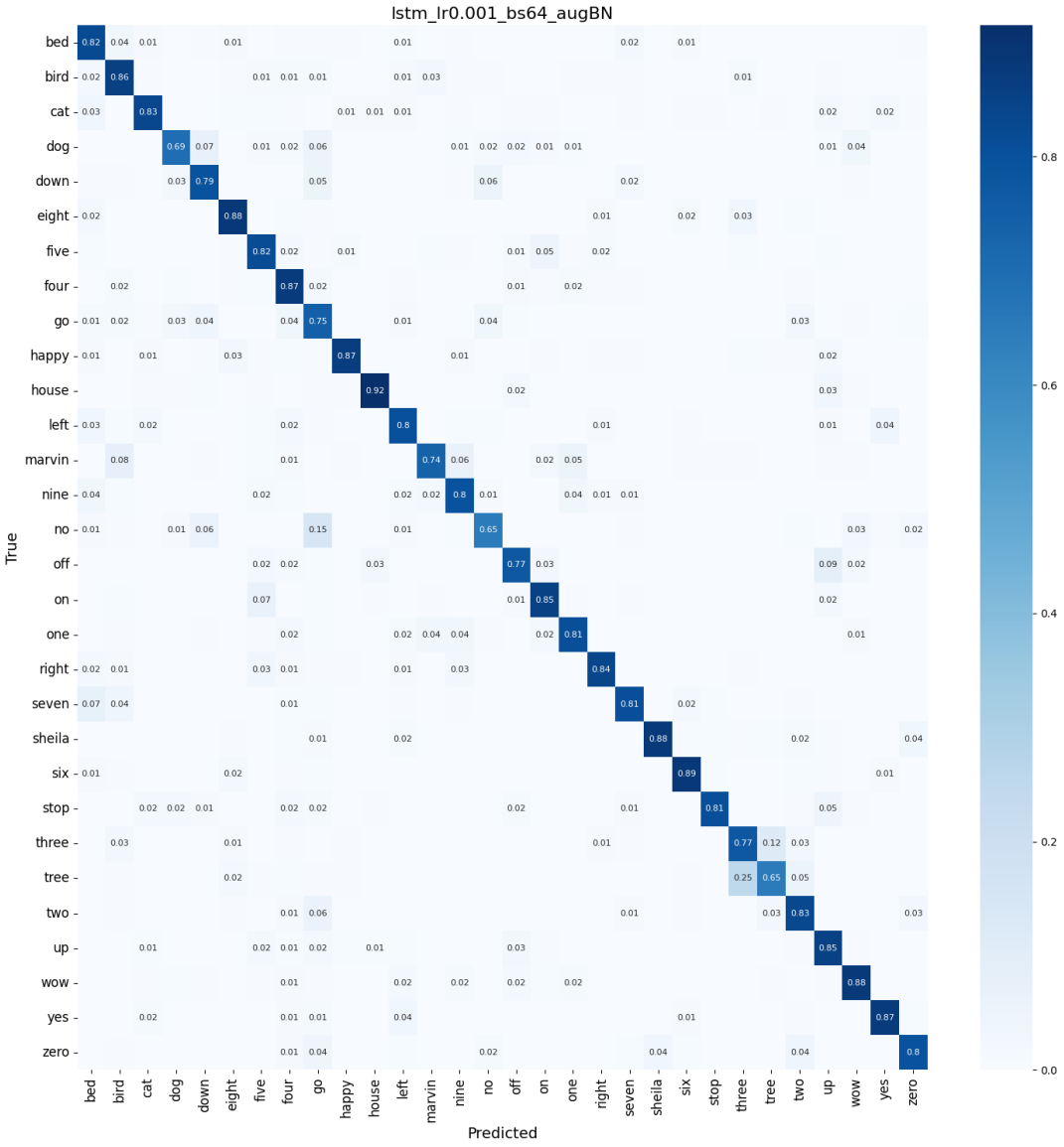

Figure 4.17. Best LSTM model confusion matrix for all label dataset

5. Conclusions and Future Work

5.1. Conclusions

- A learning rate of 0.001 generally got the best performance for both Wav2Vec and LSTM models, while AST performed best with a learning rate of 0.01.
- Wav2Vec achieved its highest accuracy with a smaller batch size (32), whereas AST and LSTM slightly benefited from a larger batch size (64).
- Data augmentation using background noise (BN) improved performance for AST and LSTM, while Wav2Vec performed best without any augmentation.
- On the 4-label dataset, Wav2Vec achieved the highest accuracy (95.8%), followed by LSTM (89.5%) and AST (88.8%) with their respective best configurations.
- On the dataset with all labels, Wav2Vec also outperformed other models (86.7%), with LSTM reaching 80.2% and AST achieving 65.7%.

5.2. Future Work

- Explore additional data augmentation techniques
- Experiment with training models with smaller learning rate and bigger amount of epochs.
- Analyze model interpretability and attention mechanisms to better understand classification decisions, especially in cases of confusion between similar labels.
- Conduct further analysis on model operations in order to better understand model confusion classifying similar labels
- Examine ensemble methods to potentially improve accuracy.

6. Application Instruction

1. Open the `.ipynb` file from the ZIP archive in Google Colab First, unzip the project folder on your computer. Then open Google Colab, click `File` → `Upload notebook`, and select the `DeepLearning_Project_2_Rzepinski__Wolny.ipynb` file from the extracted folder.
2. Use Google Colab
   Make sure you select a GPU runtime.
   (Click `Runtime` → `Change runtime type` → choose `GPU`).
3. Download JSON with Kaggle API token
   Go to kaggle.com/account, scroll to `API`, and click `Create New API Token`.
   This will download a file called `kaggle.json` — you will need it to access datasets from Kaggle.
4. Run the notebook
   Run each cell step by step (from top to bottom).

When the first cell asks you to upload a file, upload the `kaggle.json` file with your Kaggle API token.

References

[1] Y. Gong, Y. Chung, and J. R. Glass, "AST: audio spectrogram transformer", *CoRR*, vol. abs/2104.01778, 2021. arXiv: 2104.01778. [Online]. Available: https://arxiv.org/abs/2104.01778.

[2] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "Wav2vec 2.0: A framework for self-supervised learning of speech representations", *CoRR*, vol. abs/2006.11477, 2020. arXiv: 2006.11477. [Online]. Available: https://arxiv.org/abs/2006.11477.

[3] J. Donahue, L. A. Hendricks, S. Guadarrama, *et al.*, "Long-term recurrent convolutional networks for visual recognition and description", *CoRR*, vol. abs/1411.4389, 2014. arXiv: 1411.4389. [Online]. Available: http://arxiv.org/abs/1411.4389.