Project II - Transformers
Project Report

Kinga Frańczak, 313335
Grzegorz Zakrzewski, 313555

Contents

1 Description of the research problem

The project focuses on recognizing simple spoken commands using the Speech Commands dataset, sourced from [1]. This dataset comprises approximately 71,000 short audio clips in the training set and 158,538 unlabeled files in the test set, all stored as .wav files. Each clip is one second long, with a sample rate of 16000. Originally, the audio clips belong to 30 classes, but the task aims to predict one of ten base classes: yes, no, up, down, left, right, on, off, stop, and go. Any other voice command should be classified as *unknown*. Additionally, the test set includes clips that should be recognized as *silence*. The *_background_noises_* folder contains several longer files, useful for generating samples of the *silence* class. In summary, the task involves recognizing ten base classes, the *silence* class, and the *unknown* class. All training set observations were utilized throughout the project. Sample audio clips are depicted as waveforms in Figure 1.

As previously mentioned, the goal of this project is to develop a model capable of classifying voice commands from the Speech Commands dataset. The project instructions necessitate the testing and comparison of different network architectures, including at least one Transformer. Additionally, it is required to investigate the influence of hyper-parameters. The utilized network architectures and hyper-parameters are introduced in Section 3, and the experiments are described in Section 4.

2 Instruction of the application

The project solution was developed using the Python programming language and Jupyter Notebook. All the code needed to conduct experiments is contained in the included *experiments.ipynb* file. The work was facilitated by the `keras` library with TensorFlow backend. Other libraries such as `pandas`, `matplotlib`, and `seaborn` were utilized to process and visualize results. Version requirements for the

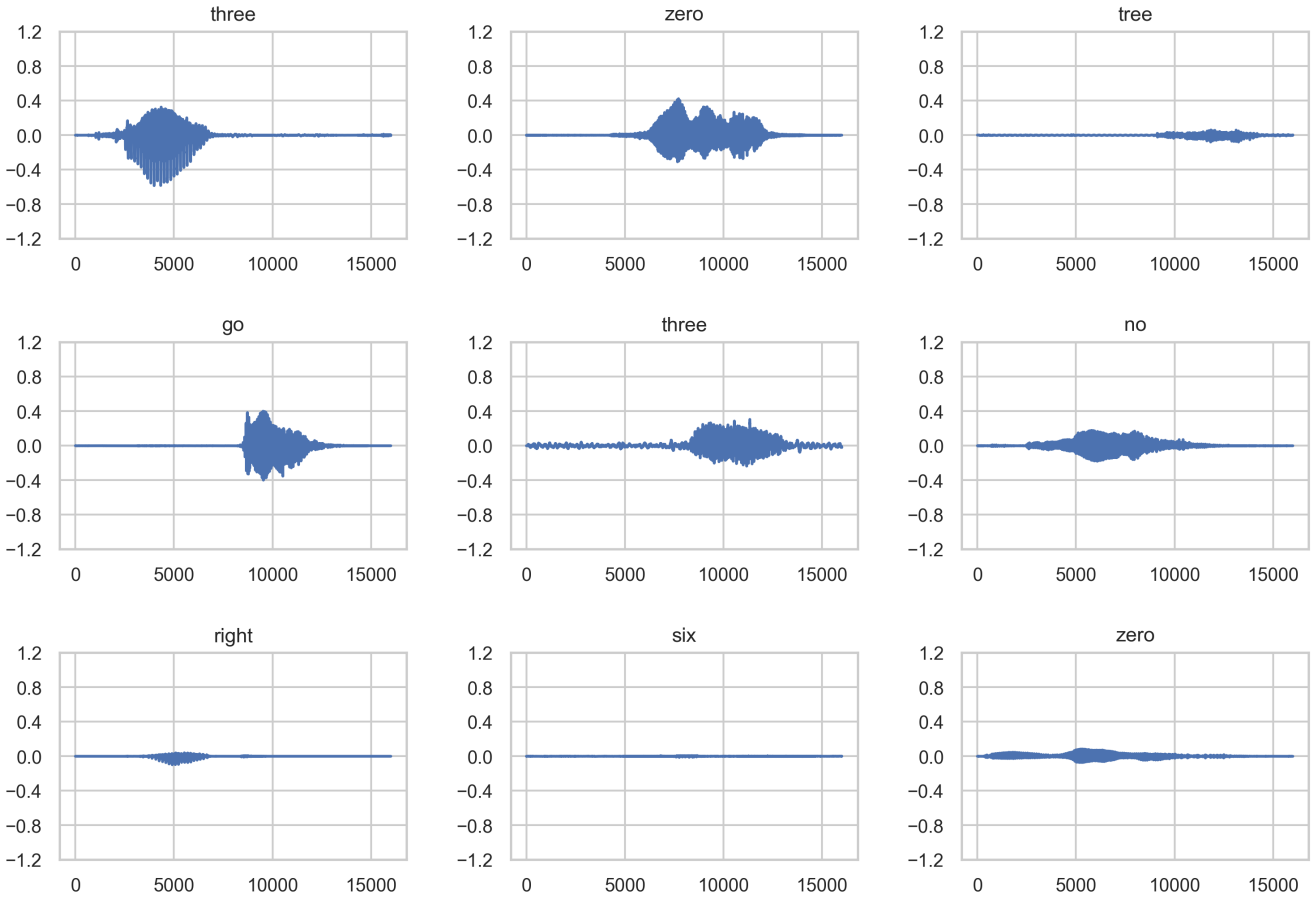


Figure 1: Sample audio clips from Speech Commands dataset.

mentioned libraries are listed at the top of the *experiments.ipynb* file. Additionally, the random number generator was initialized with a constant seed to ensure reproducibility of the experiments.

3 Theoretical introduction

Due to our lack of prior experience with Transformers, the main idea was to replicate the original architecture presented in the "Attention is All You Need" paper [3] as closely as possible. However, we soon realized that this was an impossible task. The paper describes sequence-to-sequence tasks, and the introduced architecture is an encoder-decoder model. In contrast, our project focuses on voice command classification, which does not involve an output sequence. As a result, only the encoder part of the "Attention is All You Need" Transformer could be utilized throughout the project.

Put simply, Transformers are a type of deep learning model architecture that rely on self-attention mechanisms to process input data, particularly sequences like text or audio. The encoder component of the Transformer focuses on encoding input sequences by using self-attention to capture dependencies between different parts of the sequence. In classification tasks, the output of the Transformer encoder can be used as input to a classifier, where it is mapped to class probabilities, allowing the model to make predictions based on the encoded information.

The original "Attention is All You Need" Transformer diagram is depicted in Figure 2. The encoder part is marked in red. The encoder, consisting of Multi-Head Attention, Dense, and Norm layers, was successfully implemented thanks to code snippets from the TensorFlow Transformer tutorial [2]. The result of the implementation was an Encoder layer, with the number of attention heads and the number of multi-head attention and feed-forward blocks as hyper-parameters. These hyper-parameters were later

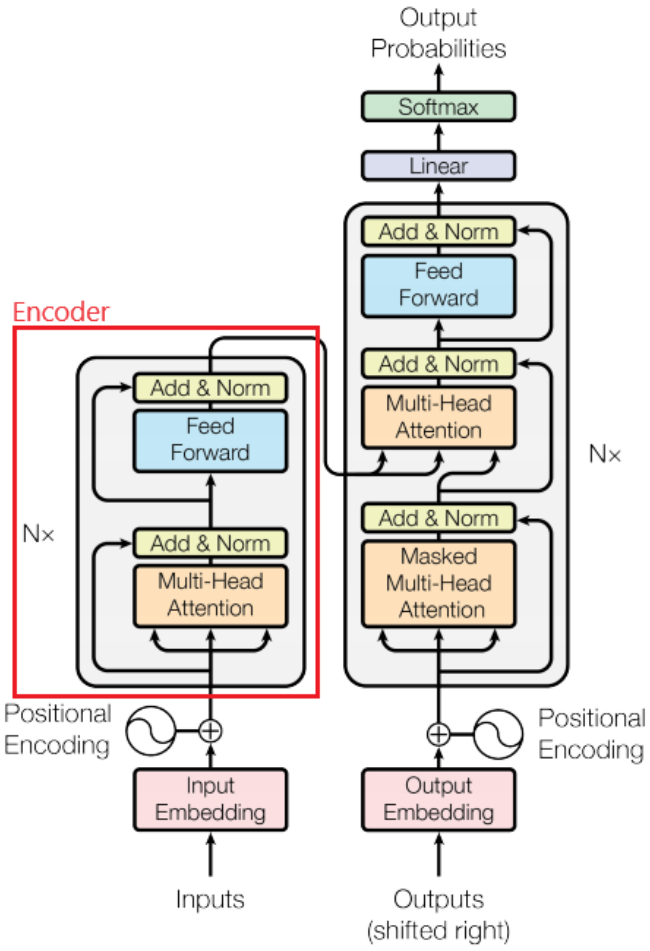


Figure 2: The original "Attention is All You Need" Transformer diagram.

tested in experiments.

In the conducted experiments, described in Section 4, three network architectures were tested. The first network was a simple feed-forward neural network with three dense layers with ReLU activation functions, separated by dropout layers. The second network was a convolutional neural network. This architecture was chosen as the best architecture from Project 1 and consists of three convolutional layers separated by pooling layers, with some dropout applied at the end. The last architecture was a Transformer. The main component of this architecture is the Encoder layer. It includes a specified number of sub-layers, each consisting of Multi-Head Attention, normalization, feed-forward, and a second normalization layer. In the first experiment, these three architectures were tested and compared, and the two subsequent experiments concerned Transformer hyper-parameters and setup only. These architectures are illustrated in the diagram presented in Figure 3.

An audio clip in its raw form is a one-dimensional vector of numbers, 16,000 elements long. The "Attention is All You Need" paper mentions input embedding and positional encoding steps. Moreover, feed-forward or convolutional neural networks would encounter problems if trained on input in the form of 16,000-element long vectors. Because of this, the original waveforms were transformed into spectrograms using Short-time Fourier Transformation. The obtained spectrograms are two-dimensional vectors of shape 132×129.

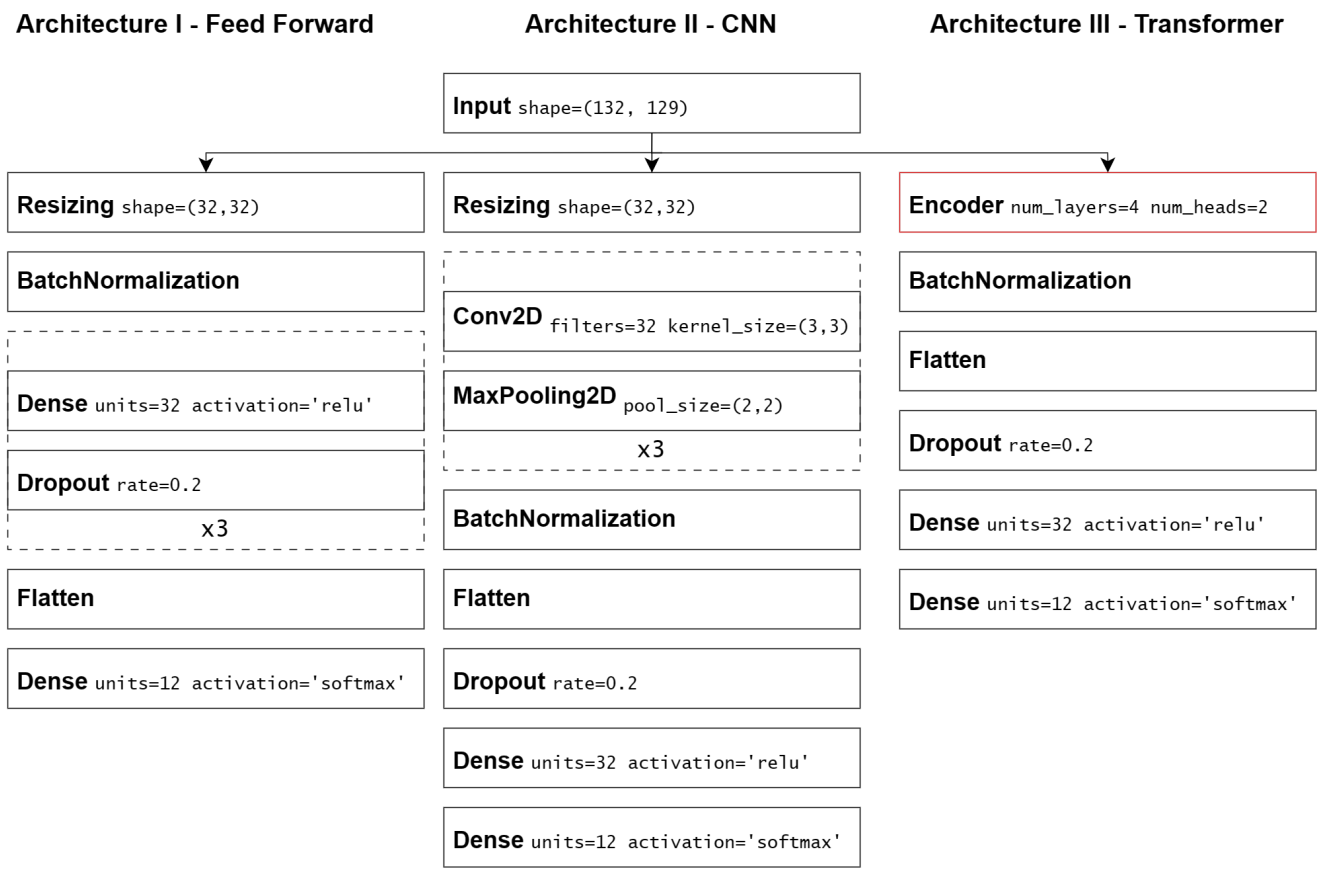


Figure 3: Three neural network architectures (keras nomenclature) used in the experiments.

4 Description of the conducted experiments

Training observations of the *silence* class were constructed using clips from the *_background_noises_* folder. This was accomplished using a moving window of length 16,000 samples (one second) with a step size of 200 samples. The window was moved across all clips from the *_background_noises_* folder, and every time the window stopped, a new *silence* clip was saved. As a result, 31,950 clips were created.

After the creation of *silence* clips, there was a total of 96,671 labeled observations. The dataset was partitioned into two segments: 77,337 clips for training (which accounts for 80% of the subset) and the remaining 19,334 clips for validation. The data was divided into batches of 32 clips each.

All metrics presented in this report pertain to the accuracy achieved on the validation dataset. Each neural network model was trained using the *sparse_categorical_crossentropy* loss function for 20 epochs. Furthermore, each model was trained five times with the same parameters.

There were three experiments conducted: the first concerning neural network architecture (Figure 3), the second investigating the influence of hyper-parameters, and the third involving a separate neural network for classification of the *silence* and *unknown* classes. Details of the experiments are provided in Table 1.

In the training dataset, the *silence* and *unknown* classes have approximately ten times more observations than each of the base classes. To address class imbalance issues, for the first two experiments - focusing on architecture and hyper-parameters - 90% of the clips from these special classes were randomly dropped.

As mentioned, the third experiment involved training a separate network to handle special cases. This time, the entire training dataset was used, but it was re-labeled, and the ten base classes were treated as one. Every observation was classified as *silence*, *unknown*, or generally as one of the base classes. Observations classified as one of the base classes were then classified again by the second network, which

assigned them to the proper class. This second network was trained only on observations from the base classes.

Experiment	Objective	Values
1	Architecture	Simple feed-forward CNN Transformer
2.1	Number of attention heads	2 4 6 8 10
2.2	Number of Encoder sub-layers	2 4 6 8
3	Handling *silence* and *unknown* classes	one network for all classes separate network for special cases

Table 1: Details of the experiments.

5 Results

The results of the experiments are presented in the form of tables and plots. Table 2 displays the mean (and standard deviation) of the best values of the loss function and accuracy achieved by models in each experimental setup. Similarly, the plots shown in Figure 4 depict the accuracy computed on the validation subset for each epoch. In the following paragraphs, each experiment is briefly discussed.

Experiment 1 (architecture) (Table 2a, Figure 4a) provides us with very reassuring results as everything aligns with our intuition. As expected, the worst performance is observed with the feed-forward neural network, which is the simplest model. The convolutional neural network yields slightly worse results compared to the Transformer. The disparity in the performance of these two models is noticeable, with the Transformer model achieving an accuracy of almost 0.9 on the validation dataset. Given that the Transformer model represents the most state-of-the-art approach, these results suggest that the implementation was carried out correctly.

Experiment 2.1 (number of attention heads) (Table 2b, Figure 4b) reveals that as the number of attention heads in the Multi-Head Attention layer within the Encoder increases, the results deteriorate. This monotonic dependency is somewhat challenging to explain. One possible explanation is that the classification task being performed is relatively simple, and complex self-attention calculations may not be necessary. It is worth noting that the standard deviation of the accuracy measure is lowest for two attention heads.

Experiment 2.2 (number of Encoder sub-layers) (Table 2c, Figure 4c) shows that there is little difference between two, four, or six encoder sub-layers. As a reminder, each encoder sub-layer (or encoder block) consists of Multi-Head Attention, normalization, feed-forward, and a second normalization layer, as shown in Figure 2. Models with four or six Encoder sub-layers take longer to achieve the same results as the model with two Encoder sub-layers. After a sufficient number of epochs, the results are almost at the same level. The most concerning aspect is the very poor performance of the models with eight Encoder sub-layers. Once again, it appears that the task is too simple for such a number of layers, or the default hyper-parameters are highly mismatched for this setting.

Experiment 3 (handling *silence* and *unknown* classes) (Table 2d, Figure 4d) concerned the usage of two separate neural networks. The first Transformer, later denoted as Transformer A, was trained

Architecture	Loss	Accuracy	Validation loss	Validation accuracy
Feed-forward	1.088 (0.007)	0.648 (0.002)	1.039 (0.006)	0.687 (0.005)
CNN	0.368 (0.011)	0.876 (0.004)	0.451 (0.018)	0.857 (0.007)
Transformer	0.207 (0.010)	0.930 (0.003)	0.360 (0.018)	0.896 (0.004)

(a) Experiment 1 - architecture.

Number of attention heads	Loss	Accuracy	Validation loss	Validation accuracy
2	0.206 (0.010)	0.930 (0.004)	0.355 (0.017)	0.901 (0.003)
4	0.256 (0.024)	0.914 (0.008)	0.369 (0.032)	0.888 (0.012)
6	0.402 (0.107)	0.866 (0.036)	0.480 (0.059)	0.845 (0.025)
8	0.623 (0.061)	0.794 (0.021)	0.645 (0.039)	0.791 (0.016)
10	0.605 (0.072)	0.803 (0.023)	0.632 (0.048)	0.798 (0.012)

(b) Experiment 2.1 - number of attention heads.

Number of Encoder sub-layers	Loss	Accuracy	Validation loss	Validation accuracy
2	0.170 (0.009)	0.941 (0.003)	0.484 (0.017)	0.875 (0.004)
4	0.206 (0.012)	0.930 (0.004)	0.380 (0.023)	0.897 (0.006)
6	0.284 (0.044)	0.905 (0.015)	0.390 (0.032)	0.884 (0.009)
8	1.503 (0.416)	0.468 (0.132)	1.685 (0.508)	0.424 (0.174)

(c) Experiment 2.2 - number of Encoder sub-layers.

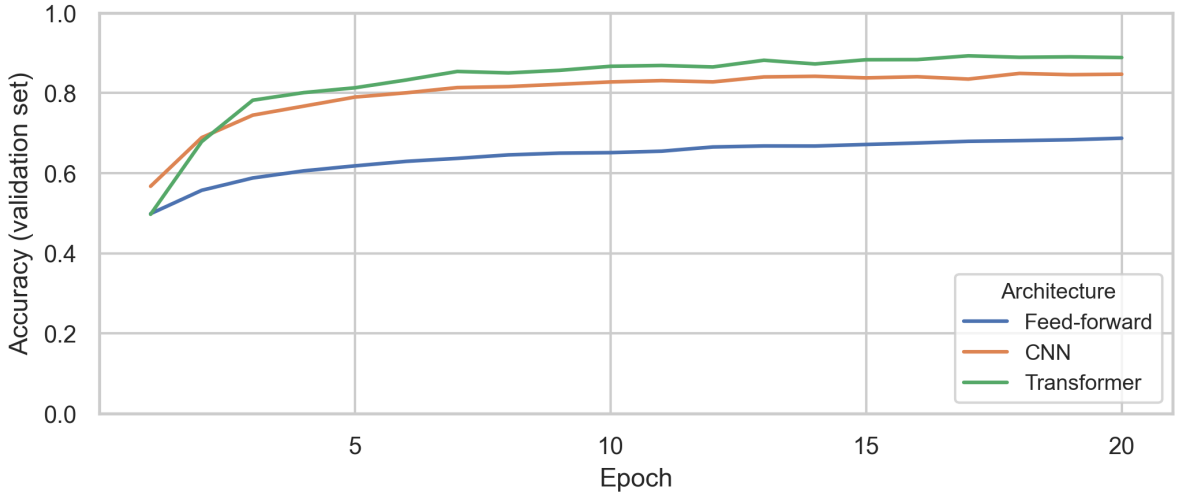
	Loss	Accuracy	Validation loss	Validation accuracy
Transformer A	0.127 (0.034)	0.947 (0.015)	0.164 (0.021)	0.934 (0.011)
Transformer B	0.177 (0.009)	0.940 (0.003)	0.331 (0.025)	0.913 (0.005)
Combination		0.944 (0.014)		0.917 (0.011)

(d) Experiment 3 - handling *silence* and *unknown* classes.

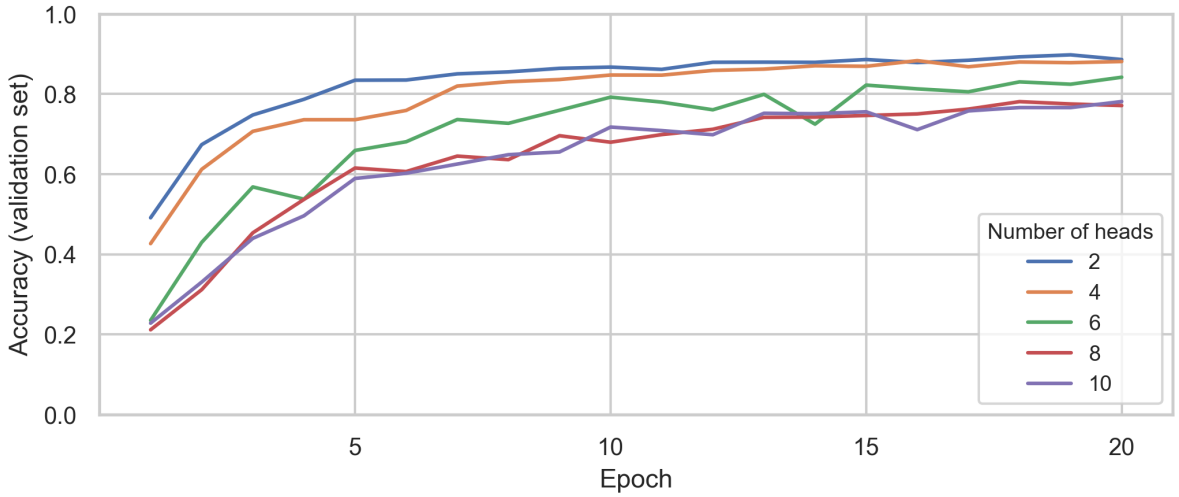
Table 2: Tables presenting the mean (and standard deviation) of the best values of the loss function and accuracy achieved by models in each experimental setup.

on the entire training dataset with the aim of correctly classifying observations into one of three classes: *silence*, *unknown*, or *base class*. The second model (Transformer B) was trained only on the observations from one of the base classes. As shown in Table 2d, both models achieved very good results. Accuracy on the validation dataset in both cases was higher than with the usage of single Transformer trained to recognize all twelve classes (e.g. the second row of Table 2c). Subsequently, the output of Transformer A and Transformer B was combined. Observations recognized as one of the base classes by Transformer A were then passed to Transformer B, which performed specific classification. As indicated in the last row of Table 2d, the combination of these two models gives us excellent results.

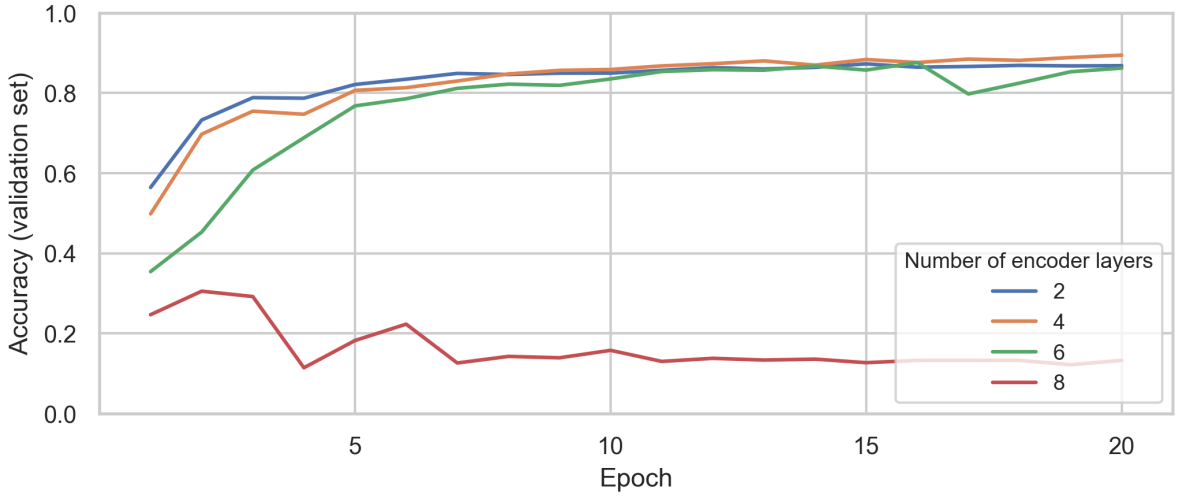
In conclusion, the best performance in terms of accuracy on the validation set was achieved with the combination of two separate Transformers (Transformer A and Transformer B). Figure 5 presents the confusion matrix prepared on the validation dataset, normalized by true conditions. The most obvious observation from this matrix is that many samples from one of the ten base classes were misclassified as an *unknown* command. This is particularly true for *on*, *go*, and *right* audio clips. Besides that, the combination of two Transformers has no significant problems with the classification task. The biggest, but still scarce issue, was with predicting the *left* command as *yes* and the *go* command as *no*. It is worth mentioning that all *silence* audio clips were correctly classified, and no other class was considered as *silence*.



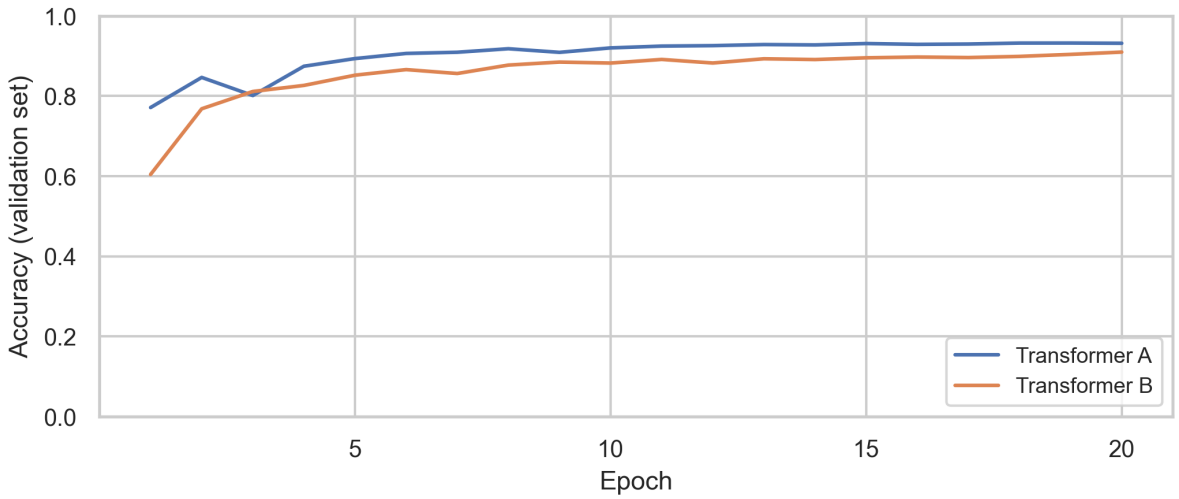
(a) Experiment 1 - architecture.



(b) Experiment 2.1 - number of attention heads.



(c) Experiment 2.2 - number of Encoder sub-layers.



(d) Experiment 3 - handling *silence* and *unknown* classes.

Figure 4: Plots presenting the accuracy computed on the validation subset for every epoch.

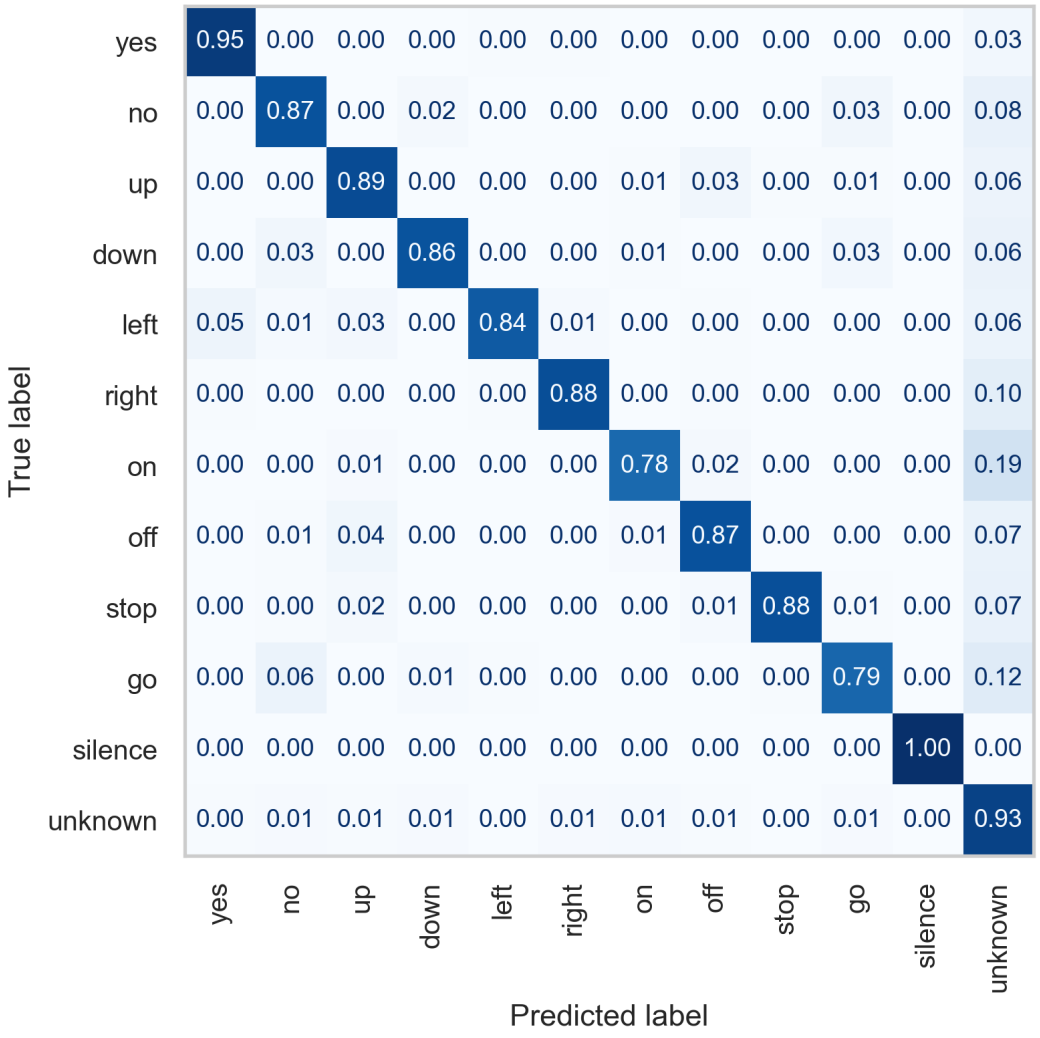


Figure 5: Confusion matrix prepared on the validation dataset, normalized by true conditions (rows).

6 Conclusions

The primary goal of this project, which is to prepare a model to classify images from the Speech Commands dataset, was successfully achieved. With the help of internet resources, a Transformer Encoder implementation was prepared. Experiments comparing network architectures and the influence of hyper-parameters were conducted. The results of these experiments confirmed the superiority of the Transformers and enabled us to select the best model setup.

The accuracy scored on the validation subset (around 0.91) appears to be a very good result. It was achieved thanks to the combination of the two Transformer models, one for handling special cases, and the second one designated for base classes. Additionally, a submission was made to the Kaggle Speech Recognition Challenge to check the performance of the solution on the test set. The submission achieved a score of about 0.67. This outcome is much lower than the accuracy calculated on the training and validation subsets. This may be due to overfitting on the training set and the relatively small size of the validation set. Additionally, more attention could be paid to cases from unknown classes - the largest drawback of our solution was misclassifying base classes as *unknown* clips.

References

[1] Speech Commands Dataset. `https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/data`.

[2] TensorFlow tutorial - Transformer. `https://www.tensorflow.org/text/tutorials/transformer`.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.