Transformers - report

Piotr Bielecki, Tomasz Krupiński

May 2024

Abstract

This document aims to consolidate all research, experiments and conclusions done for the project regarding audio classification and the application of transformer architecture [1] for this task. All codes for this project are available at github repository

1 Introduction

The data for this project comes from a 2017 Kaggle competition on speech recognition [2]. Aim of the project is to test and compare different neural network architectures, at least one of which ought to be a transformer. The research consists of investigation of the influence of certain hyperparameters, as well as the presentation and discussion of the outcomes.

2 Architectures

2.1 Wav2Vec2-based architecture

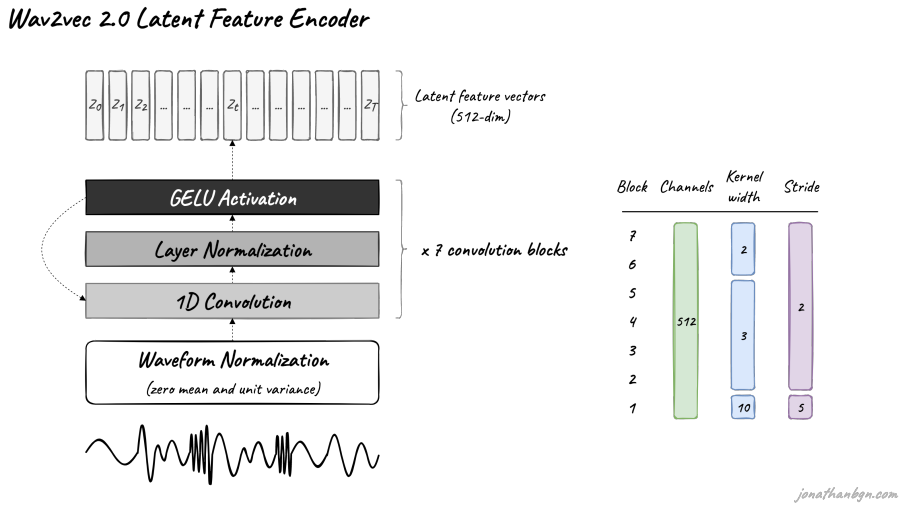


Figure 1: Wav2Vec2 feature extraction module

Wav2Vec2 model is composed of a multi-layer convolutional feature encoder which takes as input raw audio and outputs latent speech representations. These are fed to a transformer to build representations capturing information from the entire sequence. PyTorch base model consists of 12 encoding layers inspired by the transformer architecture and has feature dimension of 768 - this exact model is utilized in this project.

Wav2Vec2 is primarily used for automatic speech recognition, where it estimates the class probabilities frame-by-frame.

In this study we use Wav2Vec2 base [3] architecture with frozen weights on feature-encoding module (fig. 1). The model was Pre-trained on 960 hours of unlabeled audio from LibriSpeech dataset. In fig. 2 on the next page, example features extracted from the encoder, from one of our samples. These features come from the first, sixth, and last layers of the encoder stack.

In order to handle context representations we use pooling, to concatenate the 3D representations into 2D[4]. Finally, two dense layers serve as classification head.

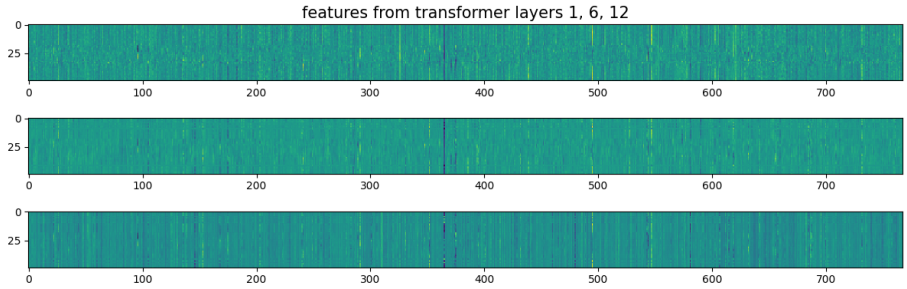


Figure 2: x axis is feature dimension, y axis represents time in frames

2.2 CNN

```
CNNet(
  (conv1): Conv2d(3, 32, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
  (conv2_drop): Dropout2d(p=0.5, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc1): Linear(in_features=51136, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=30, bias=True)
)
```

Figure 3: CNN architecture

CNN model is created by hand, rather than being prebuild. It is a simple model with only 6 layers. First two of them being convolutional layers, then by one dropout layer, one flattening layer, and two linear layers to classify inputs. The inputs to this model are spectrograms generated from our samples, saved as .png files. They are 81x201, but even if the input data would be of another dimensions, the dataloaders would rescale it to 81x201 images.

3 experiments

Two sets of experiments were conducted for each model architecture - one for binary classification (only yes/no classes), the other one for full dataset (30 classes, mostly balanced).

3.1 Wav2Vec2 hyperparameter experimentation

For this model we selected three different learning rates (10^{-5}, 10^{-4}, 10^{-3}) and three different pooling modes (max, mean, sum). The experiments were

conducted on binary classification, after which, one set of hyperparameters was chosen for training on the whole dataset.

For each set of hyperparameters, the models were trained for 10 epochs - This resulted in 9 different binary classifier models, some of which did not increase in accuracy (fig. 4). There was no experimentation done for the whole dataset. The models which were able to learn the dataset, had their accuracy/loss similar to the one in fig. 5. In table 1 we present the results of our experiments. The loss function is cross entropy on the training dataset.

learning rate	pooling mode	loss	accuracy
1e-3	mean	0.692	0.497
1e-3	max	0.691	0.5
1e-3	sum	0.696	0.497
1e-4	mean	0.0001	**0.989**
1e-4	max	4.8e-5	**0.99**
1e-4	sum	0.717	0.50
1e-5	mean	0.001	**0.986**
1e-5	max	0.001	**0.985**
1e-5	sum	4.48e-5	**0.984**

Table 1: results of the binary classification experiments for wav2vec2

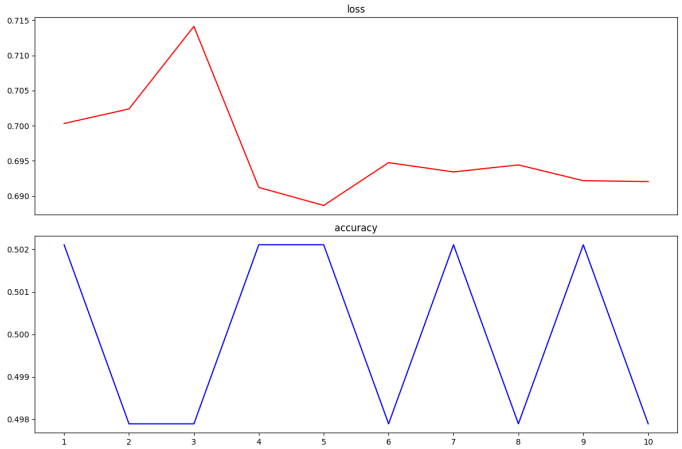


Figure 4: Example loss/accuracy for a model that did not train, training loss - red, validation accuracy - blue

The confusion matrices are presented in figures 6, 7, 8. For the highest of our learning rates, none of the models learned, blindly classifying all samples to a single class.

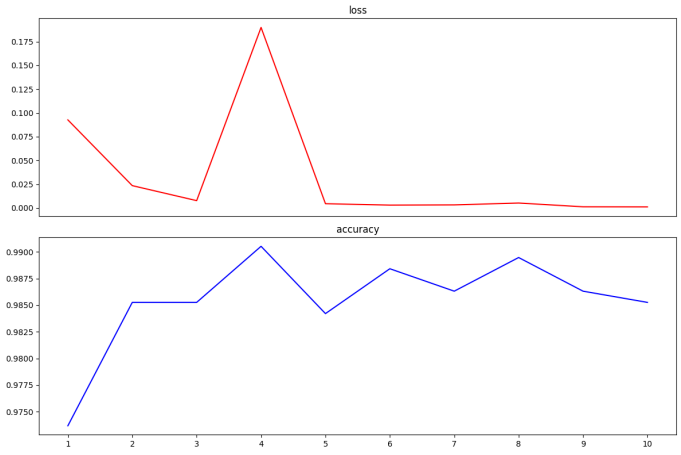


Figure 5: loss/accuracy for the model with mean pooling strategy and learning rate of 10^{-5}, training loss - red, validation accuracy - blue

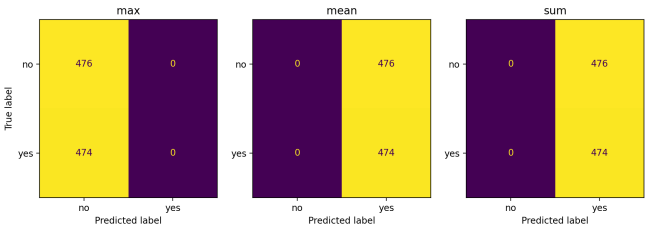


Figure 6: confusion matrices for binary classification for model with learning rate = 10^{-3}

For learning rate of 10^{-4} two of the three pooling strategies proved useful, while with the smallest of the tree learning rates, all three models gained the ability to generalize on our dataset.

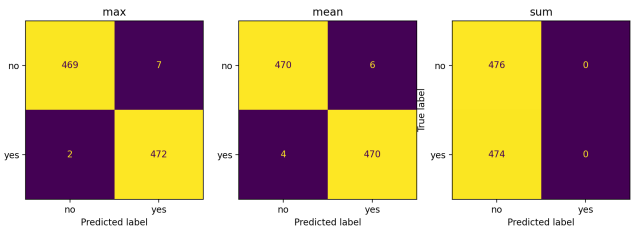


Figure 7: confusion matrices for binary classification for model with learning rate = 10^{-4}

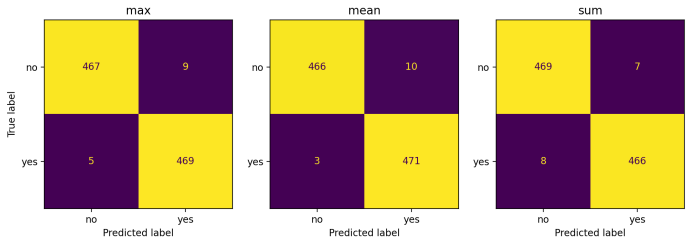


Figure 8: confusion matrices for binary classification for model with learning rate = 10^{-5}

3.2 CNN hyperparameter experimentation

For the second model, we decided to only experiment with different values of learning rates (10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}). However, this time the experiments

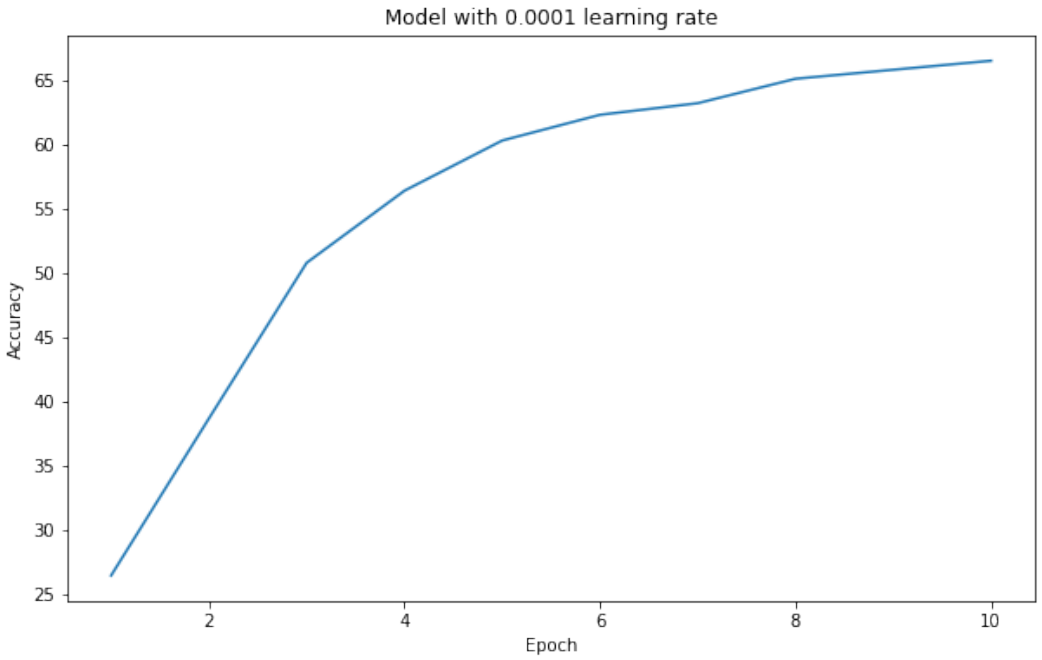
were done on all 30 classes, instead of just the yes/no classification. Thats the reason for only trying out learning rates, as it takes much longer for each epoch of the whole dataset instead of just the two classes. The loss function is cross entropy.

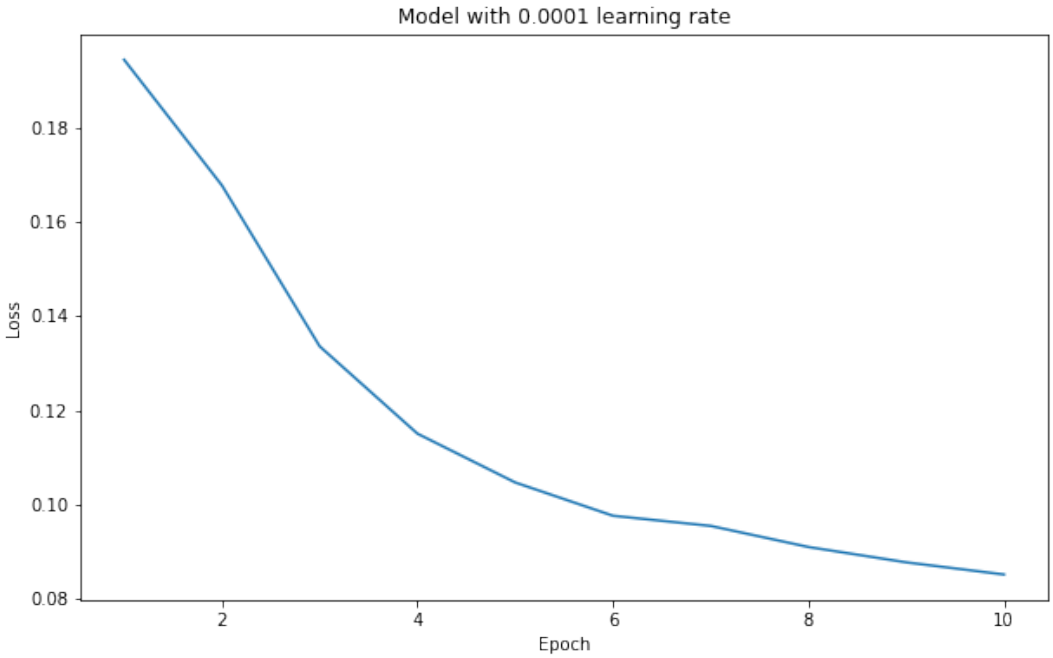
The training was done in a loop, each consisting of 10 epochs. The learning rates were fixed during a particular training session.

This table shows the results of training for ten epoch:

Learning rate	**Accuracy**	**Loss**
10^{-4}	66.5%	0.085201
10^{-3}	3.7%	0.226467
10^{-2}	2.6%	0.226747
10^{-1}	2.6%	0.226747

As we can see, only one model had the ability to generalize, given the dataset. Therefore, only the plots for that one will be shown next:





3.3 CNN multiclass classification

The experiment of classification of all the classes with CNN, was done by using the different models with the learning rates out of the hyperparameter experimentation. Three out of the four models, did not learn anything at all, and kept on classifying all data as one label. The one model that did learn, actually performed somewhat well. The biggest issue was the word UP.

Because of the amount of labels, and therefore the size of plots, all the visualizations have to be on separate pages. The first plot is of the learning rate $= 10^{-4}$ model, which was able to learn, and the other three are of the remaining models. The achieved accuracy was 66.5%.

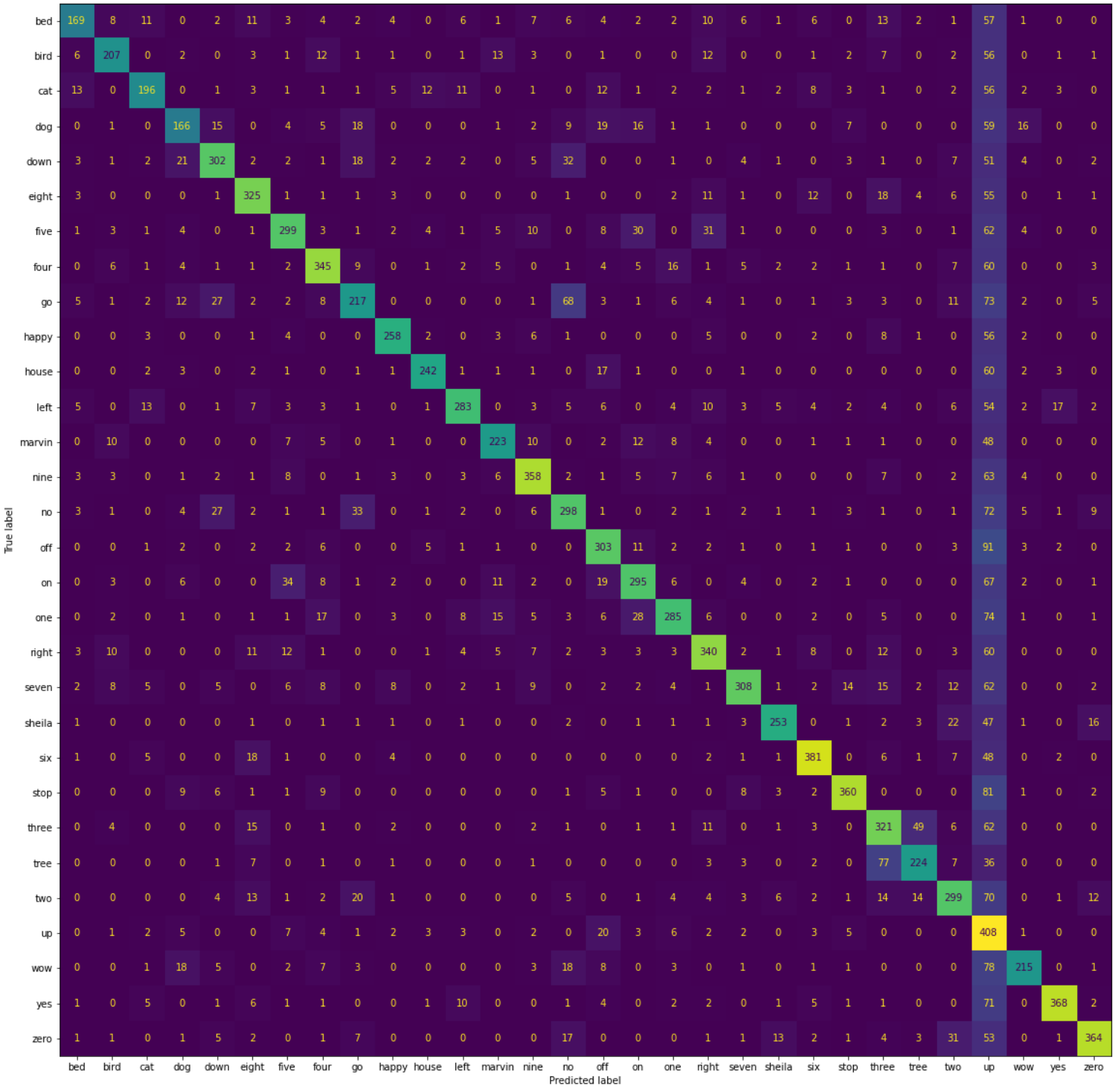


Figure 9: Confusion matrix for model with learning rate = 10^{-4}

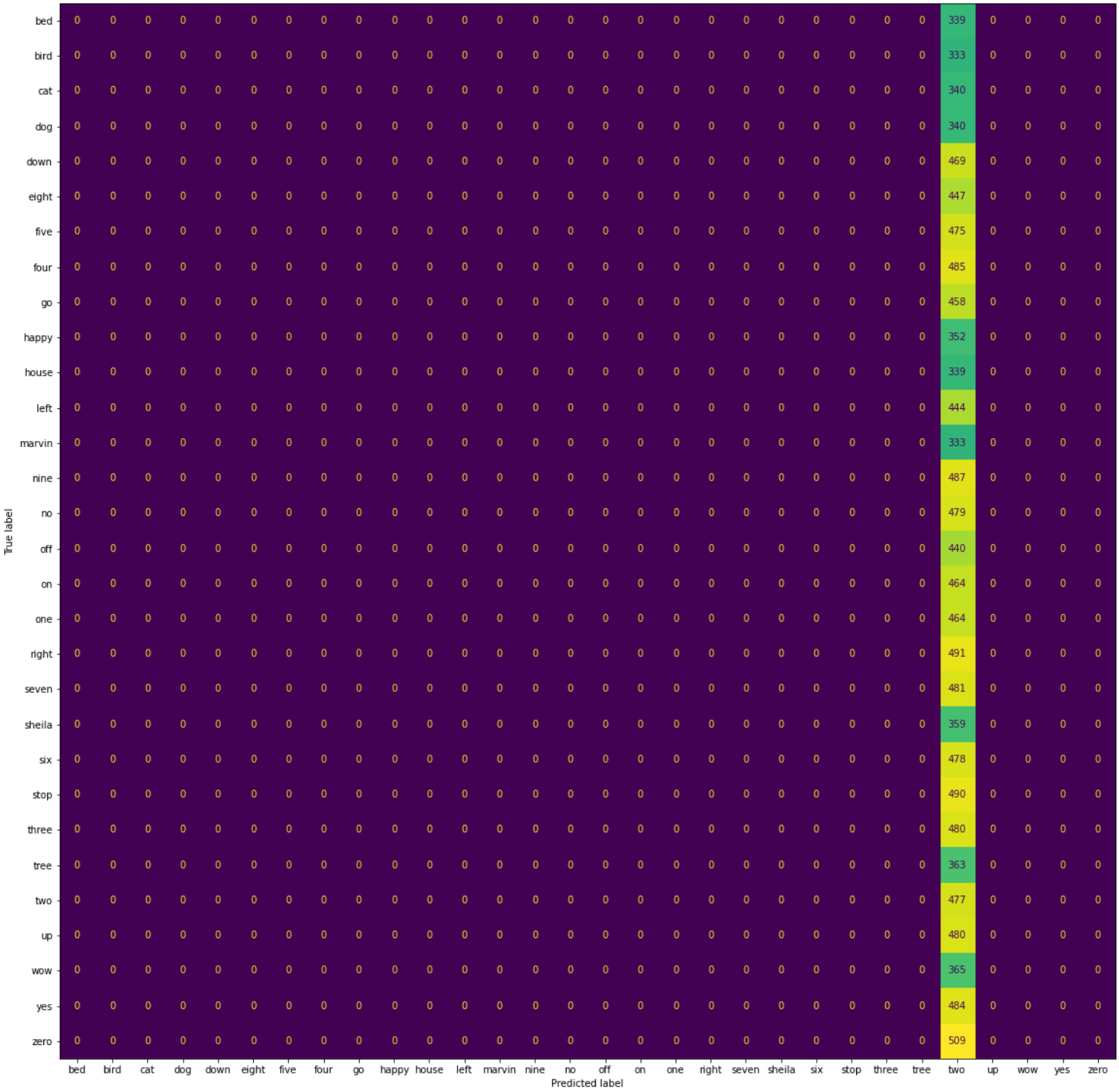


Figure 10: Confusion matrix for model with learning rate = 10^{-3}

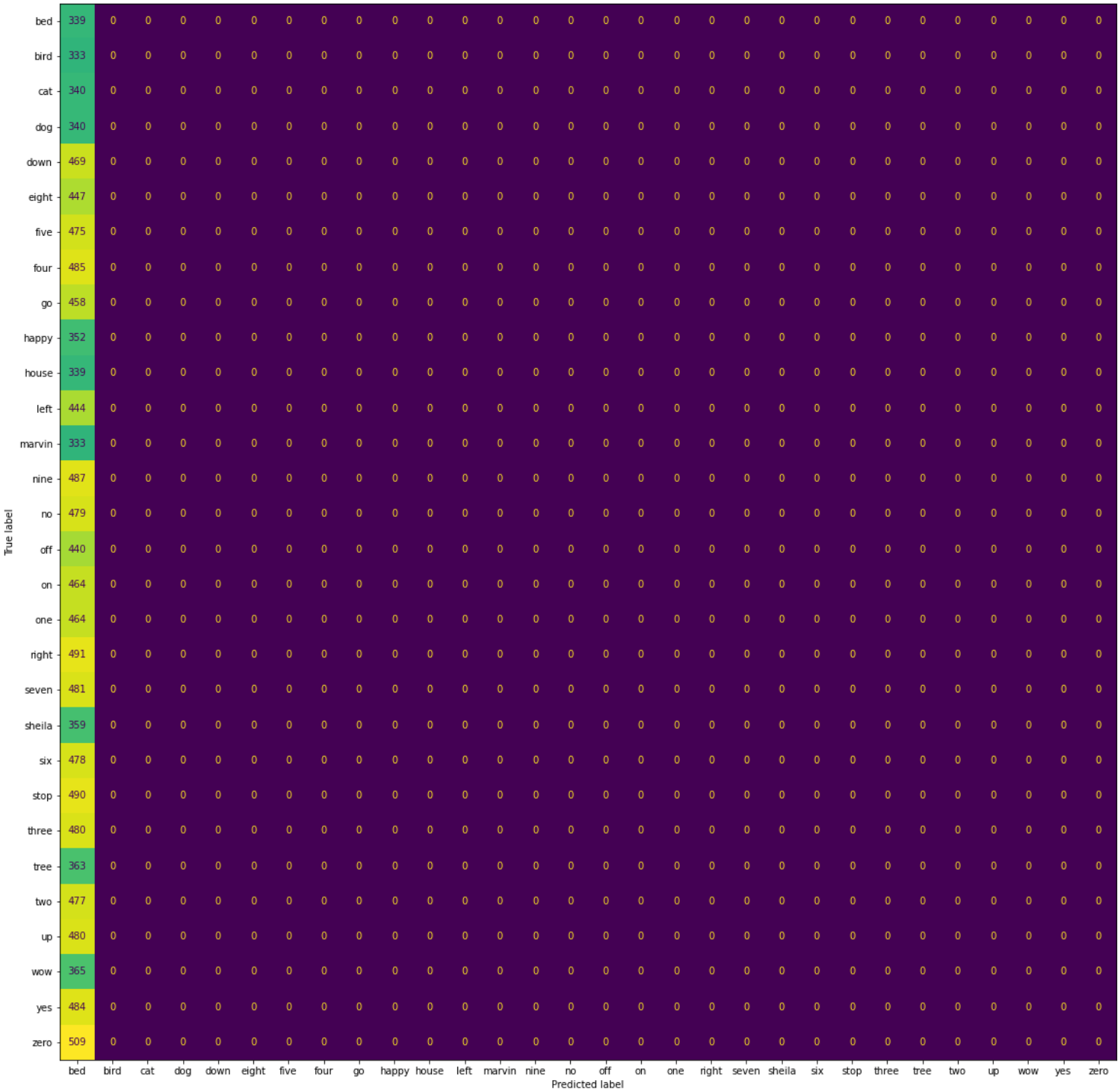


Figure 11: Confusion matrix for model with learning rate = 10^{-2}

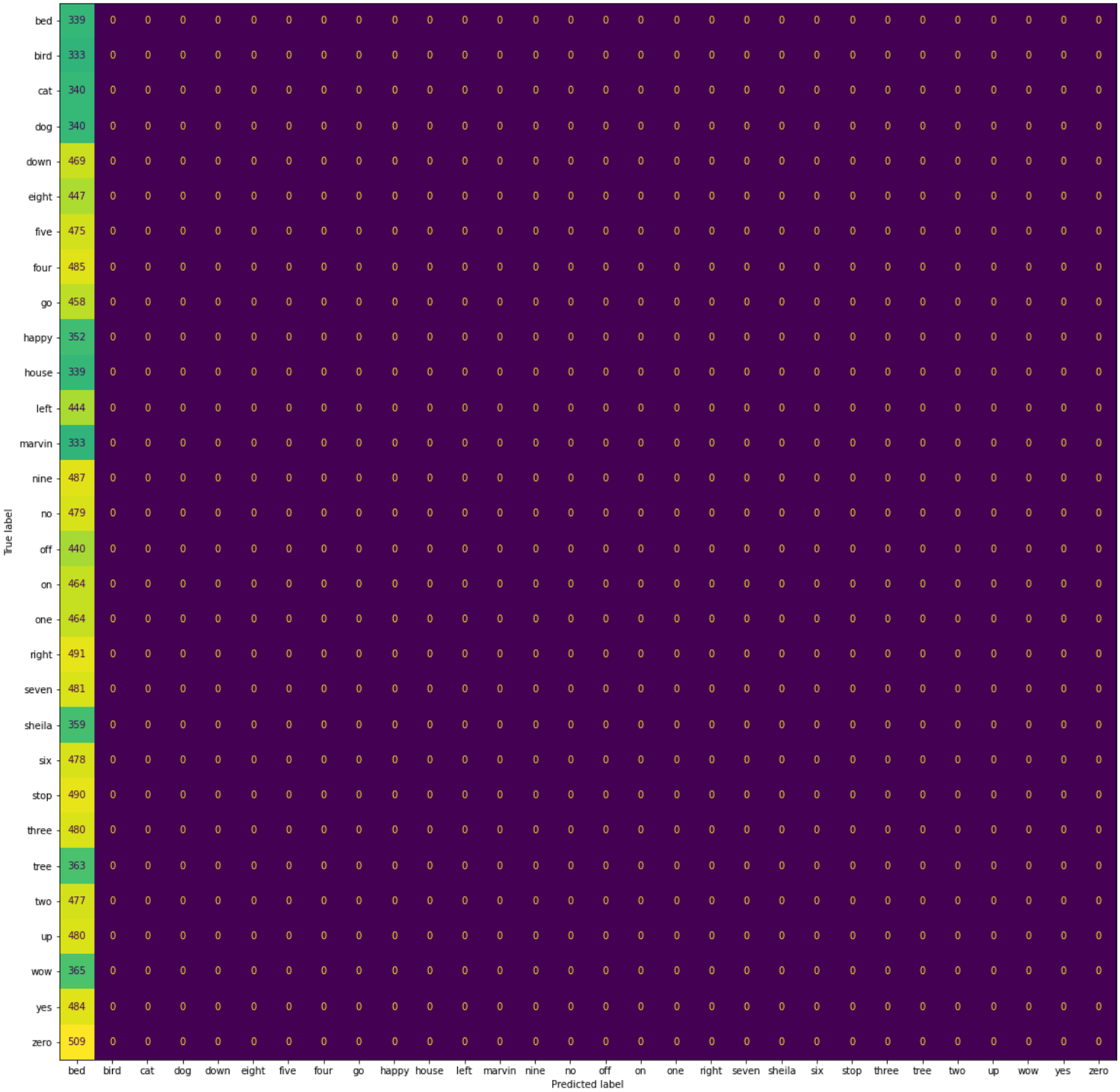
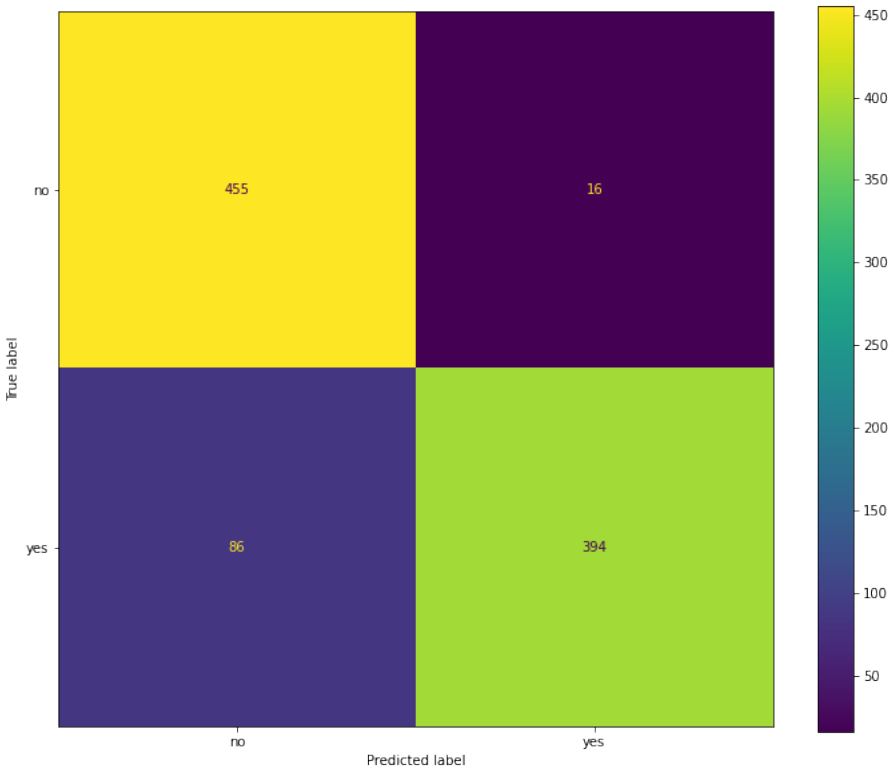


Figure 12: Confusion matrix for model with learning rate = 10^{-1}

3.4 CNN binary classification

Because the hyperparameter search was already done for multiclass task, we have only selected the model that was learning on the input data. Therefore, there is only one confusion matrix. As we can see, in general it has predicted test data correctly. The achieved accuracy was 89.3%.



4 results and conclusions

4.1 Wav2Vec2

4.1.1 Multiclass classification

For this task, we did not perform any further experimentation, We selected learning rate of 10^{-5} and mean pooling strategy for training on the whole dataset. in figure 13 training loss and validation accuracy is presented for the

training process. We can notice, after 4 epoch loss increased, but accuracy did as well. We should have stopped training then, to avoid overtraining.

In figures 14 and 15 confusion matrices for this model. the second of the figures has the diagonal zeroed out, for clarity of the errors made by the model. conclusion from figure 15 is that the model had a "go-to" for samples which proved problematic - this class was "three". Another interesting fact is that the model had trouble differentiating between "three" and "tree". Having listened to some of these samples, we would not be able to classify these correctly (these don't sound like "three" at all, but that's just a personal opinion)

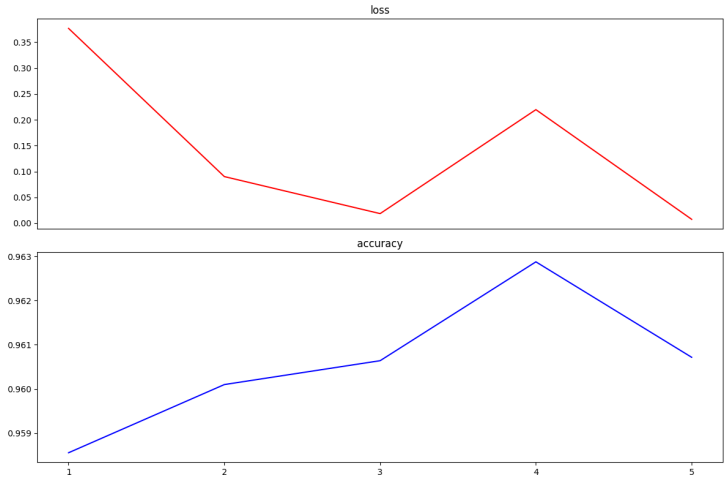


Figure 13: loss/accuracy plots for whole dataset classification, red - training loss, blue - validation accuracy

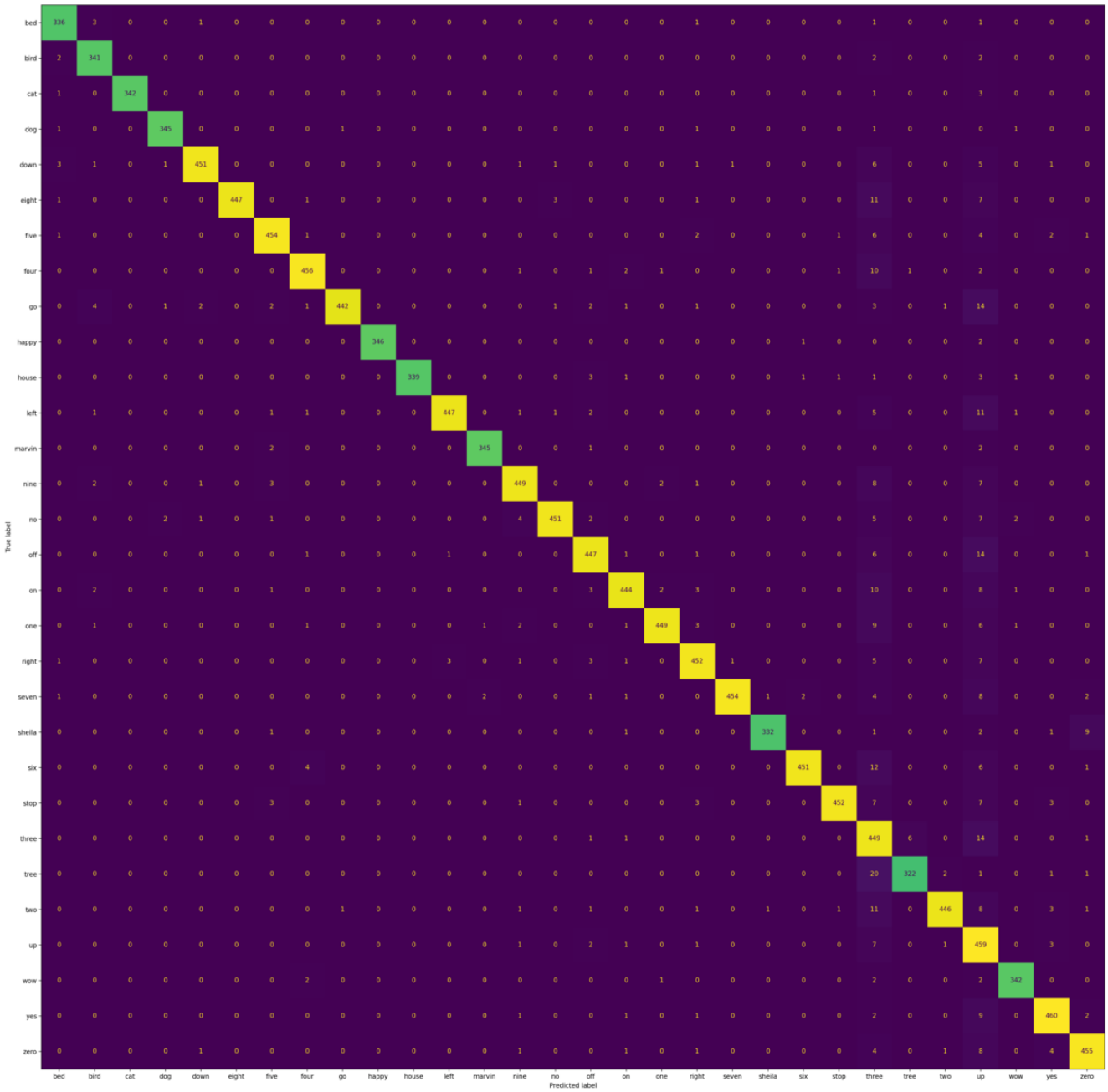


Figure 14: full dataset model confusion matrix for wav2vec2

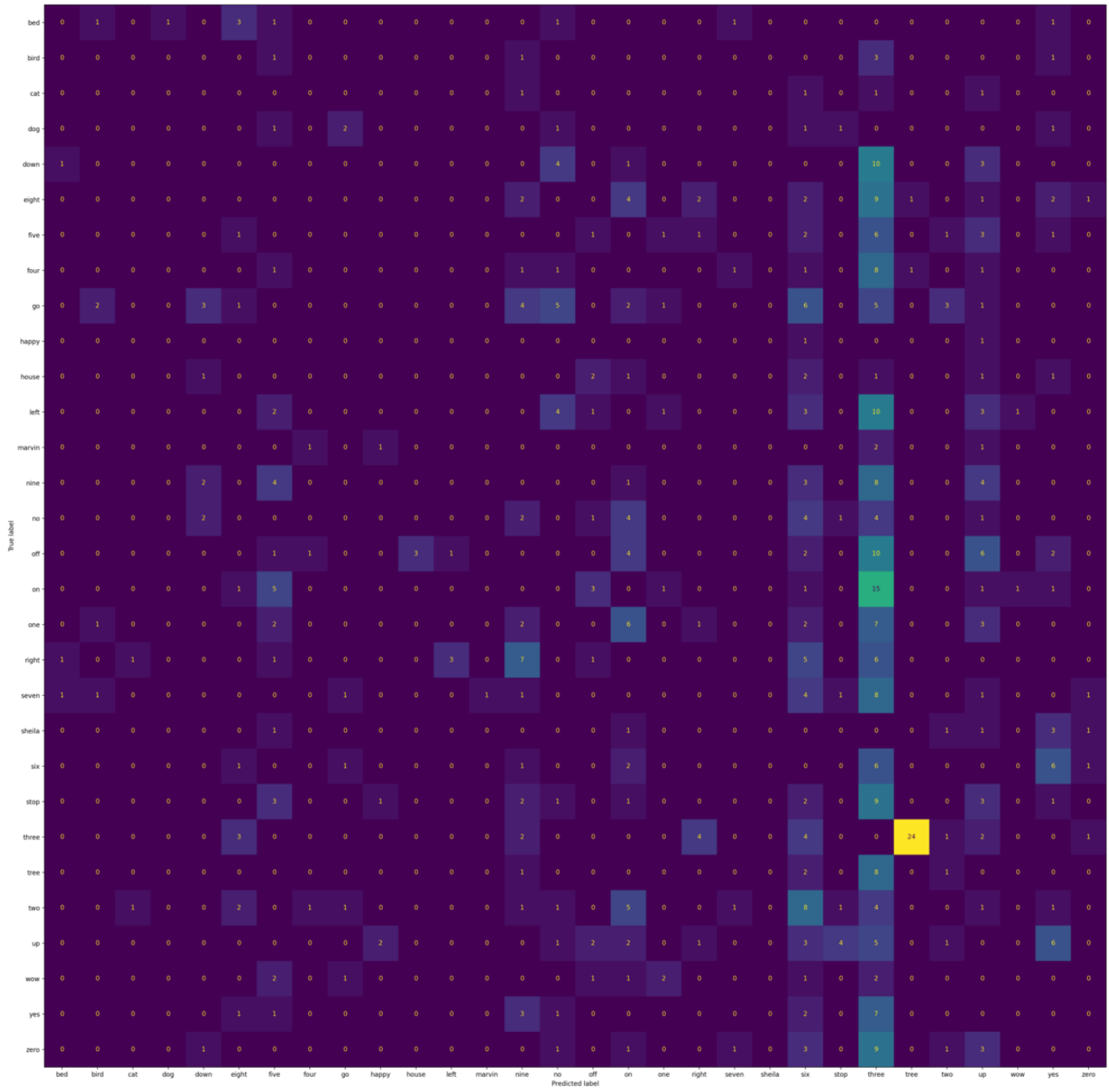


Figure 15: full dataset model confusion matrix with diagonal removed

4.2 comparison of the models

While the Wav2Vec2 based model performs extremely well for this task, and the CNN model could not achieve such results (although it did well, for its size and the specificity of the task), we have to point out, that the transformer based model is much larger. Other than that, the feature-endoding module of the wav2vec2 model was pretrained, which makes plain comparison unfair. The wav2vec2 model on full dataset achieved peak validation accuracy of 96.2%, while the CNN model achieved the accuracy of 66.5%. Both models had a "go - to" class, to which they classified samples for which the model did not know what to do with, although the prevalence of this in wav2vec was not as strong. Both models had trouble differentiating between the classes "three" and "tree", which, after having investigated the recordings, does not seem all that surprising - these are similar words, after all.

4.3 Conclusions

In this study we approached the task of speech classification from two different angles. The CNN takes as input images generated from the spectral analysis of the audio files, while the transformer-based model takes as input raw waveform data. Our assumption is - even without pretraining, the transformer model would be a stronger classifier for this task, because of the ability to look at the temporal context in the data.

References

[1] Attention is all you need

[2] TensorFlow Speech Recognition Challenge

[3] Wav2Vec2

[4] audio classification with the use of Wav2Vec2 model