Research Design and Pilot Study Report

1. Research project

1.1. Title

A vehicle type recognition model based on audio data from its drive-through

1.2. Supervisor

mgr inż. Szymon Zaporowski

1.3. Goals and short description

The goal of the project is to develop a deep neural network model that is capable of recognizing the type of passing vehicle based on the input of an audio file. The approach presented in the literature can be used as a reference for developing researchers own more advanced model using e.g. Transformers.

Main tasks to accomplish are:

1. Literature review
2. Training data analysis
3. Labelling additional data
4. Network architecture selection
5. Model training
6. Model verification
7. Analysis of results

Research design

2.1. Research goal (general goal of the research)

The goal of the research is to determine the best model type for our data in terms of architecture, optimized hyperparameters and input data.

2.2. Research gap (description of knowledge gap identified with SLR and other sources)

During the SLR process, we found out that the most common models used in vehicle type recognition based on sound are CNN, SVM, DNN and LSTM. Also, these models are frequently combined with each other to create more advanced network models and achieve better results.

Thanks to our SLR, we also discovered that sound feature extraction that goes into network input is as important, or even more important, than the model used. Thus, our input data will be a sound signal in the form of MFCC (Mel Frequency Cepstral Coefficient).

Some articles focus on creating a complex network and while others focus on best extraction features out of recorded vehicle sound.

In our approach we would like to combine well extracted and denoised feature vectors with advanced models to outperform existing networks for vehicle sound classification.

Because of the fact that our research is the continuation of the research carried out by Czyżewski, A., Kurowski, A., & Zaporowski, S. (2019). Application of autoencoder to traffic noise analysis. Journal of the Acoustical Society of America, 146, 2958-2958 (https://doi.org/10.1121/1.5137275), we will omit SVM model and focus on the Neural Network based models.

2.3. Research questions (3-5 questions to answer with own research)

a) What is the most optimal architecture type? choose from: [Dense, 2x Dense, 3x Dense, Convolutional, 2xConvolution, 3xConvolutional, Dense+Convolutional]
b) What is the optimal number of neurons (units) in each layer? choose from: [10, 30, 50, 80, 100]
c) What is the best optimizer? choose from: ['RMSprop', 'Adadelta', 'Adagrad', 'Adam', 'SGD']
d) What is the best activation function? choose from: ['tanh', 'sigmoid', 'relu']
e) During all experiments, the number of epochs will be set to 500 with an Early Stopping parameter equal to 25 epochs.

2.4. Research hypotheses (hypothetical answer to every research questions, falsifiable)

Convolutional Neural Network has better results then Dense Neural Network?
Optimal number of neurons is 100
The best optimizer is Adam
The best activation function is relu + softmax

2.5. Research subjects and sample (subjects to study, qualification criteria, sampling method)

Research subjects are audio records of vehicle passage

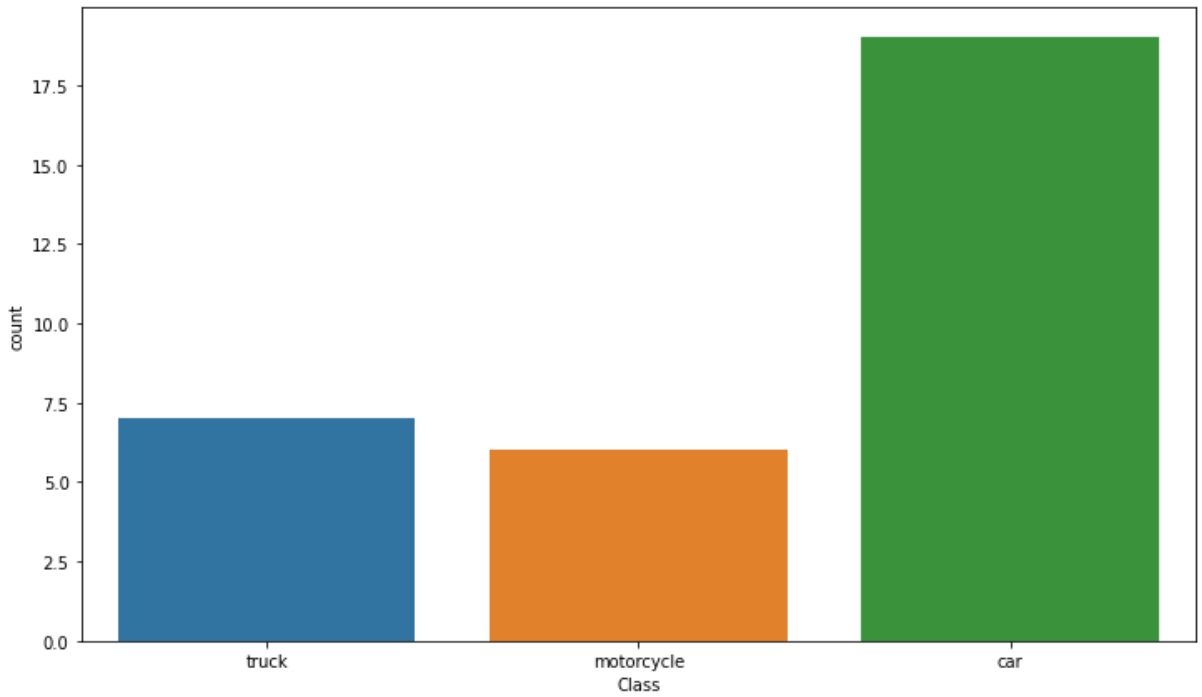
```
car_noises = [str(item) for item in actual_data_dir.glob('*.*')]
car_noises.sort(key = lambda x: int(x[x.rfind('e')+1:x.rfind('.')]))

print(f'Number of samples: {len(car_noises)}')
```

```
Number of samples: 32
```

We currently have 32 labeled samples.

```
df = pd.read_csv(str(data_dir)+'/data.csv')
df.head()
```

	id	Class
0	0	truck
1	1	motorcycle
2	2	motorcycle
3	3	car
4	4	car



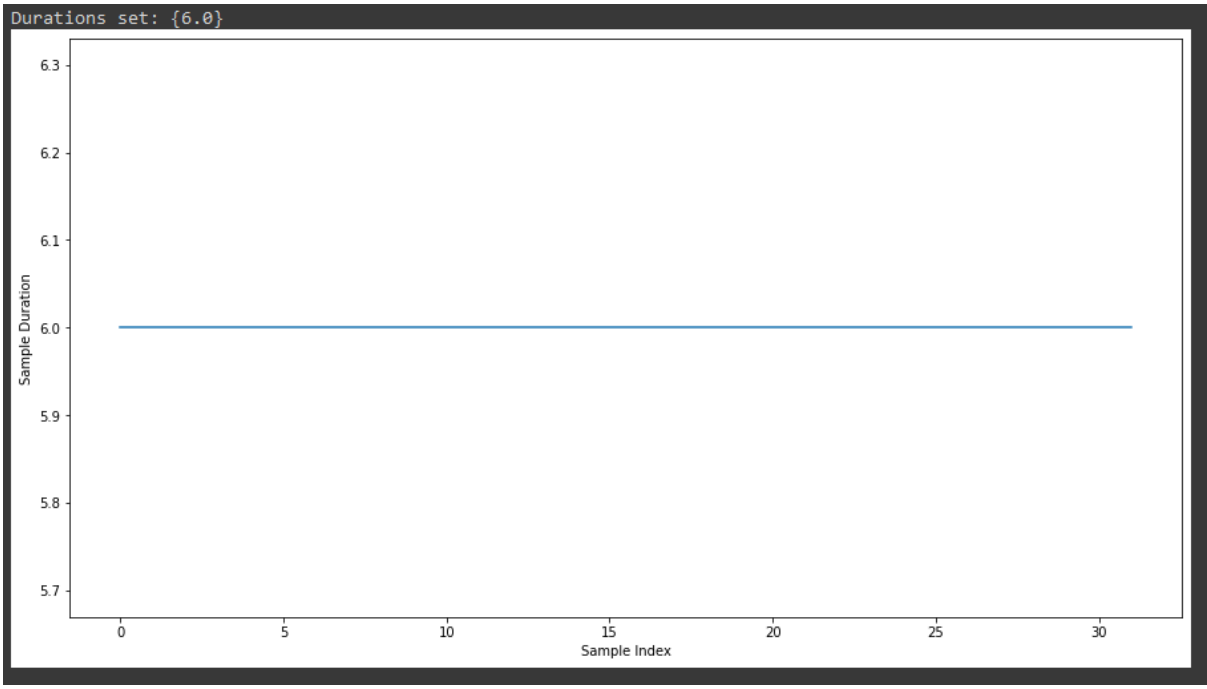
Most of the samples are car noise

It is important to check metadata of samples and look if it could be useful in experiments.

```
from tinytag import TinyTag

ipd.Audio(car_noises[0])

a_tag = TinyTag.get(car_noises[0])
a_tag
```

{"album": null, "albumartist": null, "artist": null, "audio_offset": 50, "bitrate": 3072.0, "channels": 1, "comment": null, "composer": null, "disc": null, "disc_total": null, "duration": 6.0, "extra": {}, "filesize": 2304058, "genre": null, "samplerate": 48000, "title": null, "track": null, "track_total": null, "year": null}

Most of the informations in metadata of out samples is useless but duration and channels could be useful in data analysis



Each sample has 6s duration it is important to have samples that have equal duration

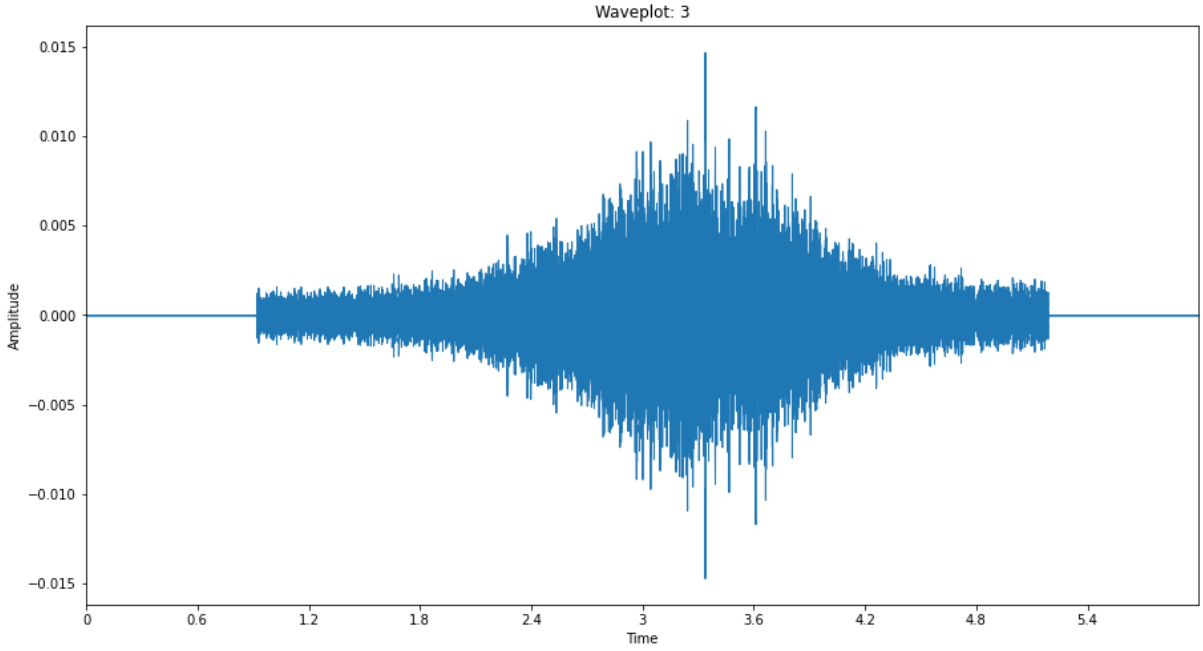
To use the data It is important to load and process them correctly

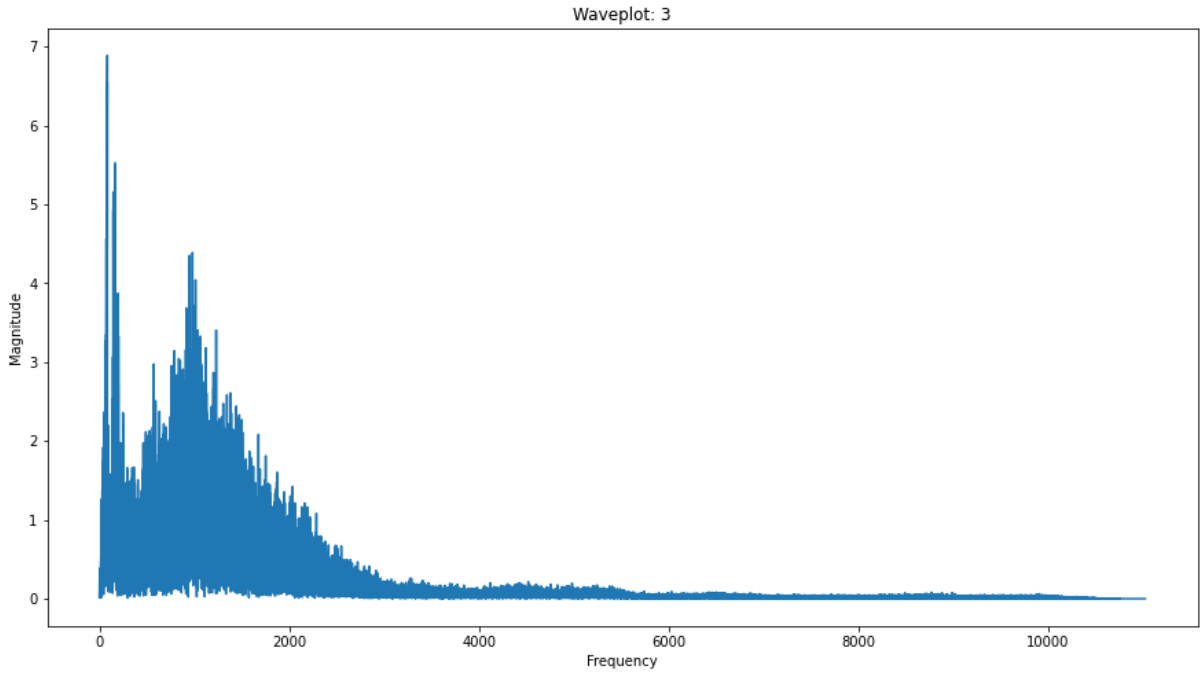
Librosa is used to load data

```
signal, sr = librosa.load(car_noises[0], sr=22050)
print("Signal: ", signal)
print("Typ: ", type(signal))
print("Dlugosc: ", len(signal), "\n")

print("Sample Rate: ", sr)
print("Typ: ", type(sr))
```

```
Signal:  [0. 0. 0. ... 0. 0. 0.]
Typ:  <class 'numpy.ndarray'>
Dlugosc:  132300

Sample Rate:  22050
Typ:  <class 'int'>
```

Using librosa it is possible to plot the wave and magnitude of frequency

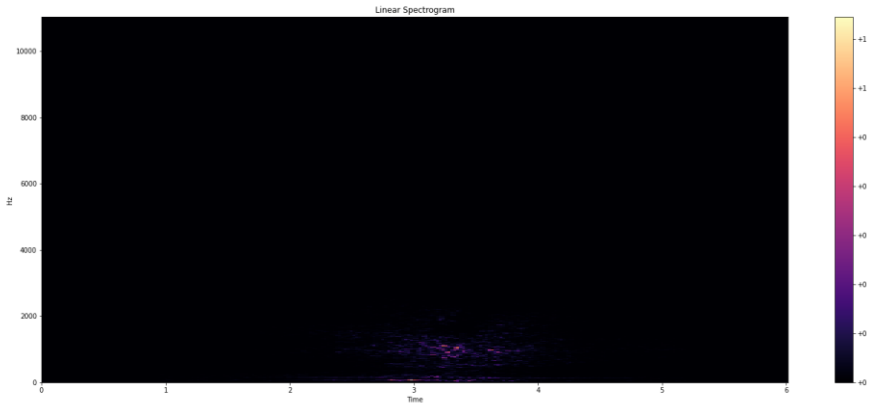


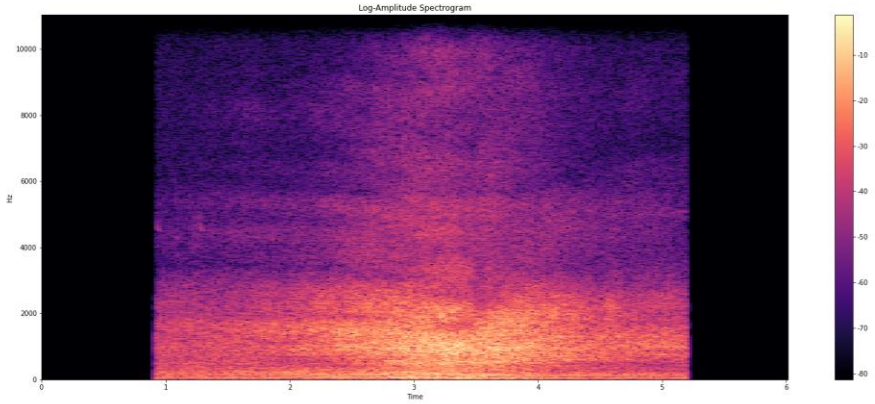


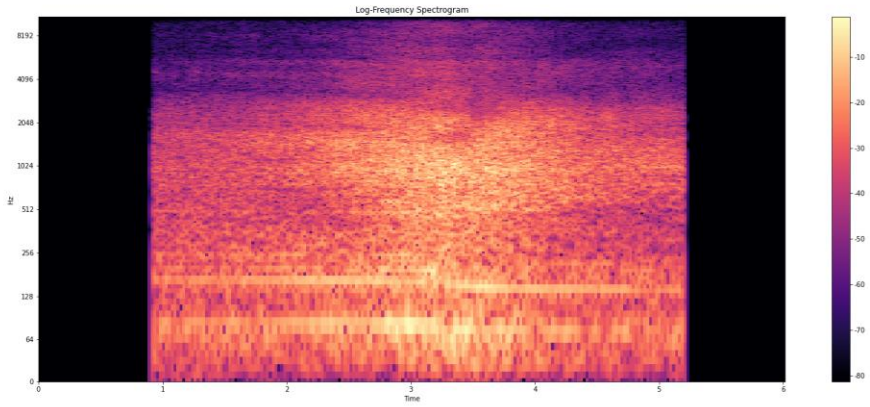
Two most popular ways to process the audio data to training purpose are spectrogram and MFCCs

Spectrograms:

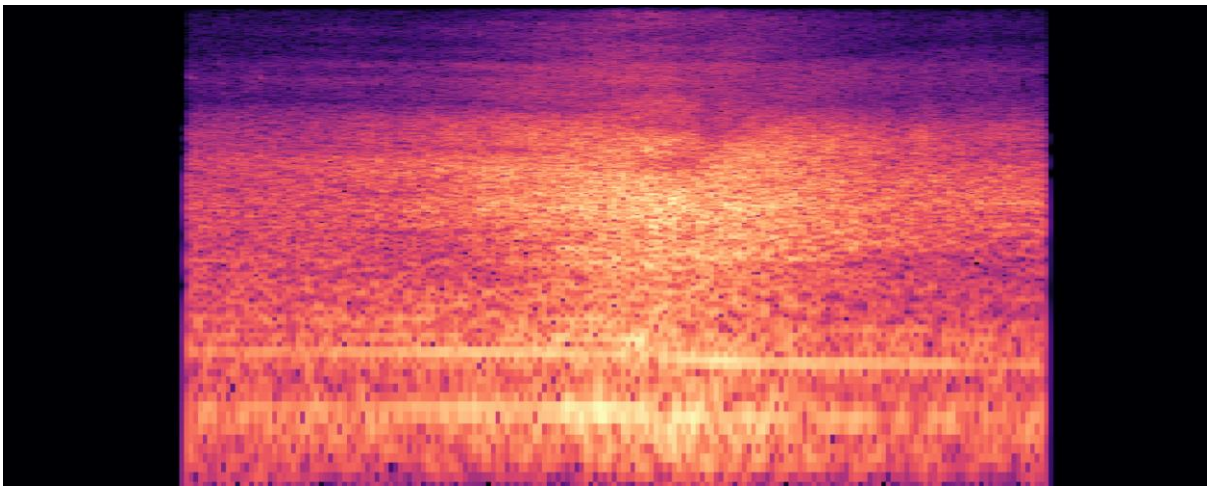
```
for x in car_noises[3:4]:
  name = str(x)
  name = name[name.rfind('/')+1:name.rfind('.')]

  ipd.Audio(str(x))
  mat, sr = librosa.load(str(x))
  S_f0 = librosa.stft(mat, n_fft=FRAME_SIZE, hop_length=HOP_SIZE)
  S_f0.shape
  Y_f0 = np.abs(S_f0)**2

  fig = plot_spectrogram(Y_f0, sr, HOP_SIZE)
  plt.title("Linear Spectrogram")
  plt.savefig(plots_dir+'/spectrogram_'+ name)

  Y_log_scale = librosa.power_to_db(Y_f0)
  fig =plot_spectrogram(Y_log_scale, sr, HOP_SIZE)
  plt.title("Log-Amplitude Spectrogram")
  plt.savefig(plots_dir+'/spectrogram_Log-Amplitude'+ name)
  fig =plot_spectrogram(Y_log_scale, sr, HOP_SIZE, y_axis="log")
  plt.title("Log-Frequency Spectrogram")
  plt.savefig(plots_dir+'/spectrogram_Log-Frequency'+ name)
```





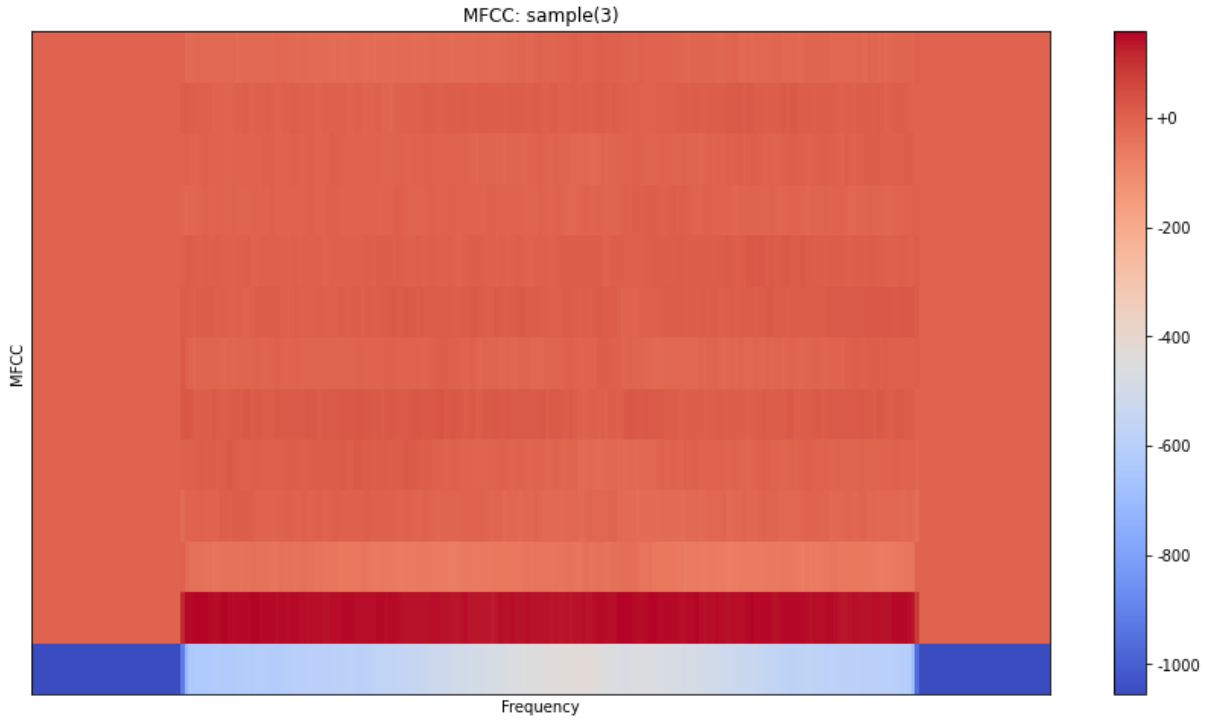


Spectrograms may be saved as an images and processed like them



MFCCs:

```
for x in car_noises[3:4]:
  name = str(x)
  name = name[name.rfind('e')+1:name.rfind('.')]
  index = int(name)

  signal, sr = librosa.load(str(x), sr=22050)

  MFFCs = librosa.feature.mfcc(signal, n_fft = FRAME_SIZE, hop_length = HOP_SIZE, n_mfcc = 13)

  fig=plt.figure(figsize=(15, 8))
  librosa.display.specshow(MFFCs, sr=sr, hop_length=HOP_SIZE)
  plt.xlabel("Frequency")
  plt.ylabel("MFCC")
  plt.title(f'MFCC: sample({index})')
  plt.colorbar(format="%+2.f")
```



To use MFCCs in training it is important to extract features

```
signal, sr = librosa.load(car_noises[0], sr=22050)
MFFCs = librosa.feature.mfcc(signal, n_fft = FRAME_SIZE, hop_length = HOP_SIZE, n_mfcc = 13)
feature = np.mean(MFFCs.T, axis=0)
print("MFFCs: ")
print(MFFCs)
print("MFFCs type: ", type(MFFCs))
print("MFFCs shape: ", MFFCs.shape)
print("Feature: ")
print(feature)
print("Feature type: ", type(feature))
print("Feature shape: ", feature.shape)
```

```
MFFCs:
[[-971.6235 -971.6235 -971.6235 ... -971.6235 -971.6235 -971.6235]
 [   0.        0.        0.     ...    0.        0.        0.    ]
 [   0.        0.        0.     ...    0.        0.        0.    ]
 ...
 [   0.        0.        0.     ...    0.        0.        0.    ]
 [   0.        0.        0.     ...    0.        0.        0.    ]
 [   0.        0.        0.     ...    0.        0.        0.    ]]
MFFCs type:  <class 'numpy.ndarray'>
MFFCs shape:  (13, 259)
Feature:
[-6.1432727e+02  8.6552910e+01 -6.2932177e+00 -4.4856491e+00
 -8.4625950e+00  1.2782081e+01  2.6140790e+00  1.0740006e+01
  1.0855153e+01  5.1412400e-02  1.5524321e+00  5.7838502e+00
 -5.4327564e+00]
Feature type:  <class 'numpy.ndarray'>
Feature shape:  (13,)
```

Now it is possible to use the MFCCs features to create training data

```
data[0]
```

```
[array([-6.18957764e+02,  9.30773239e+01, -1.27259626e+01,  1.79644728e+00,
        -1.45376043e+01,  1.86015091e+01, -2.90124631e+00,  1.59156685e+01,
         6.05080032e+00,  4.45966816e+00, -2.44410872e+00,  9.36136723e+00,
        -8.58996105e+00,  7.25627899e+00, -7.45218372e+00,  6.12963390e+00,
        -6.41582298e+00,  1.59493446e+00, -5.13423204e+00,  4.04451656e+00,
        -3.93139005e+00,  1.75449681e+00, -3.03771472e+00,  4.94094849e-01,
        -3.13434315e+00, -7.90502071e-01, -8.35597396e-01,  3.72299016e-01,
         1.64196634e+00,  3.98923308e-01, -1.45177424e+00, -8.03995192e-01,
        -2.41189122e+00,  7.99351096e-01, -2.30536008e+00, -4.25885111e-01,
        -1.66353726e+00, -9.52477813e-01, -9.46766138e-01, -1.29049861e+00],
       dtype=float32), 'truck']
```

2.6. Operationalization – variables (independent, dependent, confounding, and hidden variables)

Experiment variables:

- independent:
  - type of neural network layers
  - amount of neurons/channels in each layer
  - type of optimizer
  - type of activation function
  - type of input (MFCC / Mel-Spectrogram)
- dependent
  - number of epochs / running time
  - average accuracy on training set
  - average loss (cross-entropy / mean square error)
- confounding
  - libraries used (TensorFlow / PyTorch)
  - random network parameters initialization
  - random data split
- hidden:
  - functions implementation in libraries

2.7. Research methods (real-life experiments, simulations, surveys, interviews, FGIs, action research, case studies, structural equation modeling etc.)

The selected experiments will be conducted in the form of python scripts execution using a small subset of data, divided into training, validation and testing sets. Achieved validation and testing results will be assessed using the MSE (Mean Square Error) metric.

In order to ensure the reproducibility of the experiments, the dataset is sampling with constant seed random value.

```
df = df.sample(frac=1, replace=True, random_state=7)
```

The experiment is to train different models with different parameters on processed train data and check the results. The **data** that is being used are **spectrograms and MFCCs features**.

In order to ensure greater reliability of the experiments each training is executed 10 times and results are the mean result of each execution.

2.8. Research tools (experiment design, survey design, interview guide, FGI guide, action research plan, case study scope and execution plan etc.)

Google Colab, PyCharm IDE / VSCode in order to execute the experiments.

Github for code sharing and version control.

Google Drive to share data and plots.

Additionally to find the best suited solution for our project and compare two most popular machine learning libraries, we implemented our experiments in two different environments.

First one is the Google Colab machine with the experiments implemented in the TensorFlow library.

And the second one is a personal computer with low end Nvidia video card and experiments are implemented with PyTorch library.

2.9. Expected results (qualitative and quantitative results expected)

According to our knowledge the CNN model is more popular in sound recognition than DNN, so we can predict that that's for a reason and CNN will perform better than DNN. Again we've seen that in many research researchers used Adam optimizer as first choice so we think that it will have the best performance. As sound recognition is a rather complicated task, we think that it needs a fairly complex network, so the number of neurons in the layers should be quite big. We predict that best results will be achieved with more than 30 neurons in each layer. In terms of activation functions ReLU is our choice.

2.10. Validity threats (threats to construct, internal, external and conclusion validity; means to minimize these threats)

Internal validity is the extent to which you can be confident that a cause-and-effect relationship established in a study cannot be explained by other factors. Without high internal validity, an experiment cannot demonstrate a causal link between two variables.

Threats to validity that we discovered are:

- our small sample of data may not represent the general features of the vehicles
- randomness in sets creation may create imbalanced classes
- our recordings comes from one road with fixed speed limit, so our samples may not represent general vehicle sound with any speed

2.11. Research plan (process, phases, tasks, assignments to team members)

ZADANIE	DATA WYKONANIA (DEADLINE)	OSOBA ODPOWIEDZIALNA	Uwagi	CZY WYKONANE
Przygotowanie kamieni milowych na stronie projektu	10.03.2022	Dariusz Kobiela	-	TAK
Zapoznanie się z tematem przetwarzania audio: nauka wskaźników, celów projektu itd. (jak coś znajdziecie fajnego - to podsyłać na Discord na kanał)	10.03.2022	Mateusz, Michał, Dariusz	-	TAK
Przygotowanie raportu 1 (Metody badawcze w Informatyce)	07.03.2022	Michał	-	TAK
Systematyczny przegląd literatury (SLR): bazy IEEE, dalej Scopus i Web of Science (w razie problemów piracka "Scihub"). Robić to na kompach na uczelni (otwarte laboratorium) - wtedy po IP uczelni mamy dostęp do baz artykułów [KAŻDY PO 1/3 liczby artykułów]	19.03.2022	Mateusz, Michał, Dariusz	-	TAK
Przeczytanie i ocena artykułów [KAŻDY PO 1/3 liczby artykułów]	22.03.2022	Mateusz, Michał, Dariusz	-	TAK
Snowball	24.03.2022	Mateusz, Michał, Dariusz	-	TAK
Uzupełnienie raportu i wysłanie na enauczanie (+ PREZENTACJA)	24.03.2022	Michał Hajdasz	Deadline: 28.03	TAK
Wstepny wybór model których użyjemy	31.03.2022	Mateusz, Michał, Dariusz	-	TAK
Plan plac z Panem Szymonem	04.04.2022	Mateusz, Michał, Dariusz	-	TAK
Analiza dostępnych danych + Przygotowanie danych (EDA)	07.04.2022	Mateusz, Michał, Dariusz	-	TAK
Tagowanie części danych	14.04.2022	Mateusz, Michał, Dariusz	-	TAK
Szkic raportu 2 na MBI (plan prac + opis danych)	21.04.2022	Mateusz, Michał, Dariusz	Może się opóźnić o 2 dni	TAK
Trenowanie i testowanie modeli (w ograniczonym zakresie - na części) [bez frameworku - tymczasowe testy]	28.04.2022	Mateusz, Michał, Dariusz	-	TAK
Analiza wyników (czy działa jak	07.05.2022	Mateusz, Michał,	Deadli	TAK

powinno)		Dariusz	ne: 11.05	
Dokończenie raportu 2 i wstawienie go na enauczanie	09.05.2022	Mateusz, Michał, Dariusz	-	TAK
Wybór artykuł do recenzji + Szkic raportu 3	12.05.2022	Mateusz, Michał, Dariusz	-	
Ocena artykułu	19.05.2022	Mateusz, Michał, Dariusz	-	
Dokończenie raportu 3 i wysłanie na enauczanie	26.05.2022	Mateusz, Michał, Dariusz	-	
Sporządzenie raportu końcowego (w formie publikacji naukowej)	02.06.2022	Mateusz, Michał, Dariusz	-	
Zrobienie Postera z wynikami	02.06.2022	Mateusz, Michał, Dariusz	-	
Przerwa sesyjna	16-30.06.2022	-	-	
WAKACJE			-	
Puszczenie eksperymentów na pełnych danych (wszystkich danych) - COLAB lub kompy uczelnia	październik 2022	Mateusz, Michał, Dariusz	-	
Ewentualna poprawa jakości działania modelu	październik 2022	Mateusz, Michał, Dariusz	-	
Sporządzenie raportu końcowego (w formie publikacji naukowej)	pierwsza połowa listopad 2022	Mateusz, Michał, Dariusz	-	
Zrobienie Postera z wynikami	druga połowa listopad 2022	Mateusz, Michał, Dariusz	-	

2.12. Publication goals (what results of the research could be published and possibly where)

Although there are some publications that describe good results in vehicle recognition, there is still some room for improvement, especially among the more difficult classes like buses.

Also line detection of a transpassing vehicle would be something worth publishing.

The achieved result could be published in “The Journal of the Acoustical Society of America” as a continuation of the research carried out by Czyżewski, A., Kurowski, A., & Zaporowski, S. (2019). Application of autoencoder to traffic noise analysis. Journal of the Acoustical Society of America, 146, 2958-2958.

3. Pilot study

3.1. Research subjects (description of subjects involved in the pilot study, qualification criteria)

3 executors of the prepared code, executed on a small subset of data.

All executors have an engineer title and some experience in quite big projects.

Some of us specialize in programming and some in data analysis.

3.2. Study execution (who, when, how executed the pilot study)

To answer to our research question we needed to test four components:

- different types of layers of the network
- number of neurons in each layer
- types of optimizers
- types of activation function

These experiments were conducted sequentially, one after another starting from choosing the type of the model. For the next experiment the best approach from the previous phase was used.
So if in the first experiment the CNN model was found to be the best, then the amount of neurons in the layer in the second experiment will be tested with CNN model.
We won't test every possible combination, firstly because it would cost a lot of time and secondly because we don't think that there is a correlation between tested components. Usually in machine learning projects just selected combinations are checked.

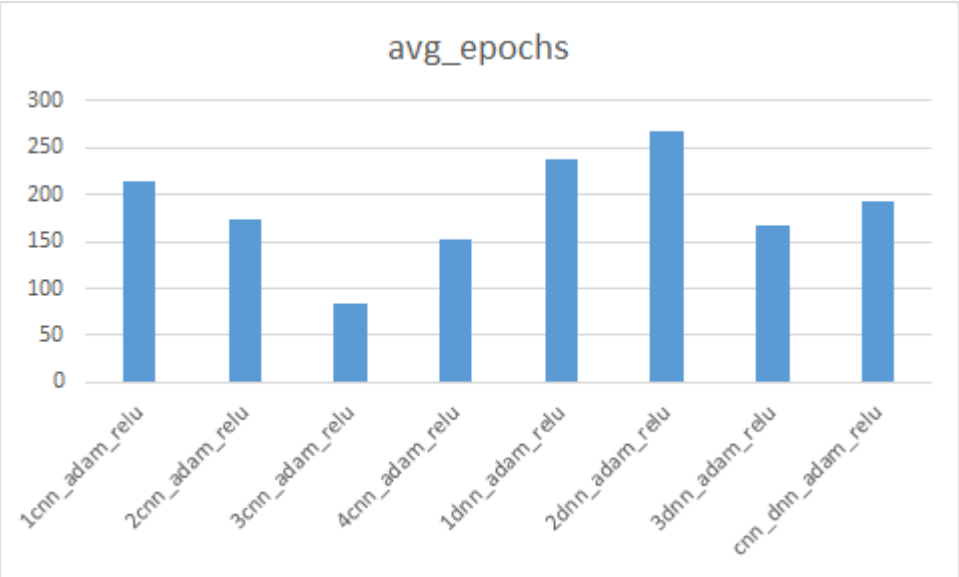
Experiments were executed on processed data and some models have been trained. Selected metrics were measured in each epoch to see the trend.

Each training was executed 10 times in order to ensure greater reliability of the experiments. Results in each epoch are the mean value of 10 values.
We are interested in the type of model, number of neurons, activation functions and optimizers.

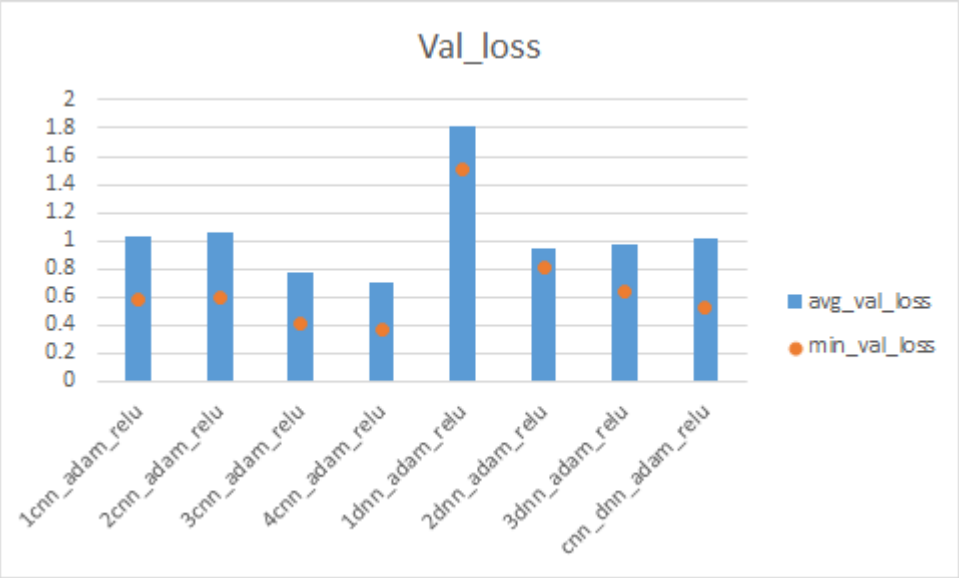
To avoid library and environment influence all experiments were conducted on different setups.

3.3. Results (research results from the pilot study, quality evaluation of the results)

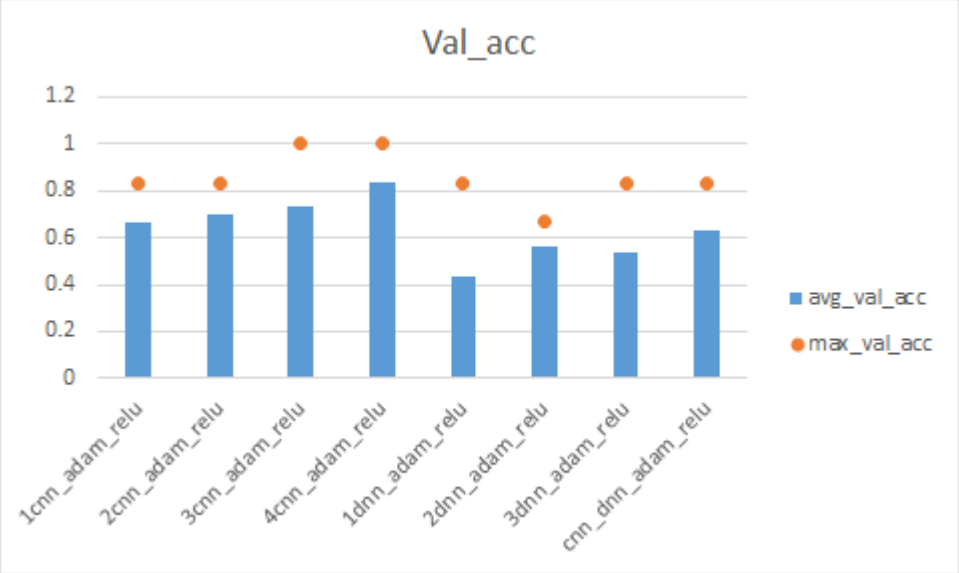
Our first experiment was to find the best composition of layers for our network model. We tested simple layers like CNN and DNN (fully connected layers) and their number in the model.
The results are as followed:



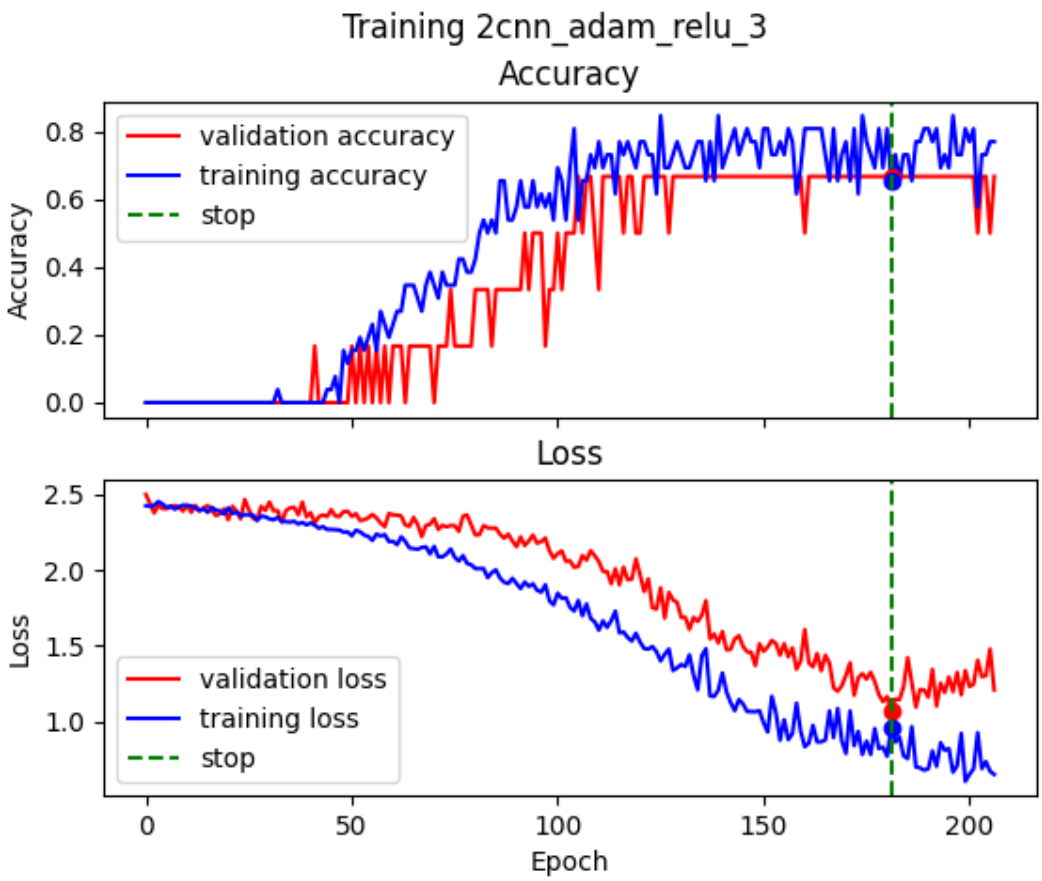
Here we can see that the number of epochs depends on the number of layers in the model. Overally, the more layers there are, the less epochs the model needs to be trained. However, that trend seems to end when the number of layers achieves 4.



Now, in terms of validation loss (we used validation set for testing because of the lack of data) which seems to be the most important statistic here, we can see that on average CNN layers give better results then DNN layers. Moreover, one DNN layer has far worse results then the rest of the models, probably that model was too simple for the task.

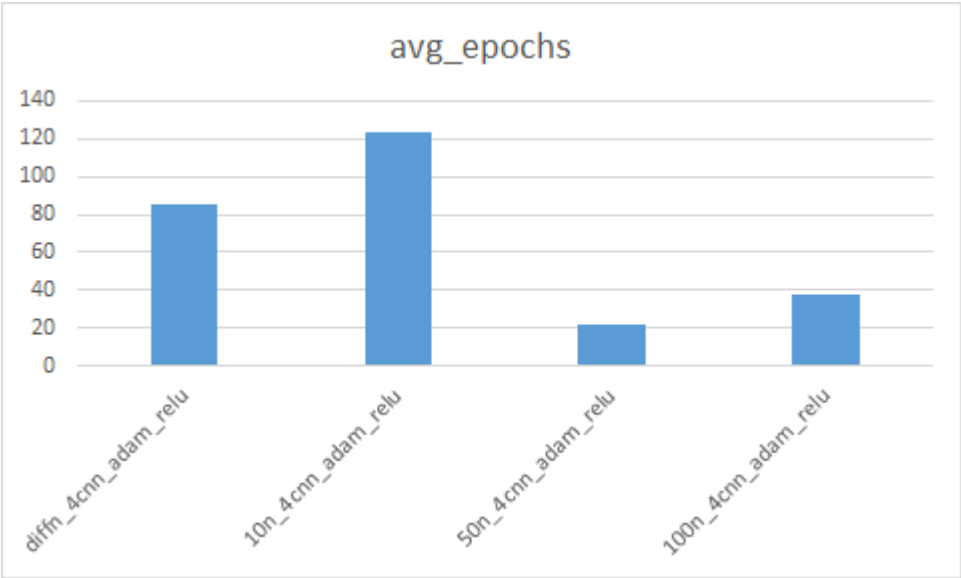


When it comes to accuracy there is also the tendency that the CNN layers have higher scores then DNN layers.Two models (3CNN and 4CNN) achieved 100% accuracy, but we shall remember that our validation set was very small (8 elements) and could be imbalanced.
The CNN+DNN behaves mediocre, but it's probably because of just two small layers, using 3 or 4 CNN layers with DNN could result in better scores.

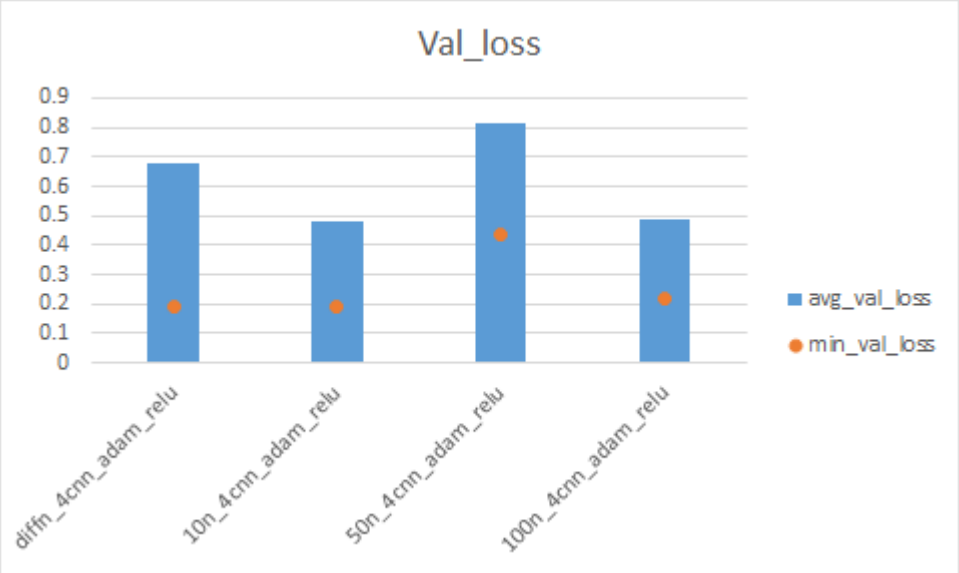


On the example chart that shows the history of the loss function during the training we can see that our early stopping function works great.
The training loss is decreasing for the whole range but the validation loss stops decreasing in one point and starts increasing. That's the point where the overfitting may start and our algorithm stops the training.

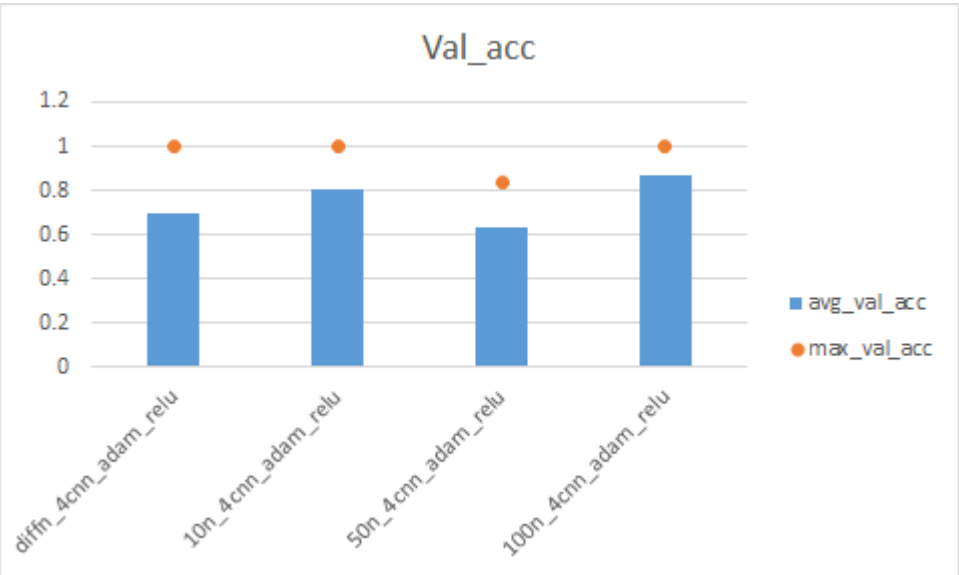
For the next step we've choose the **4*CNN** model and proceed to the next experiment which is testing the number of neurons in the layers.
Charts with results:



We can define models as:
diffn - model with different increasing number of neurons in layers, exactly: 6, 8, 16, 32
10n - 10 neurons in each layer
50n - 50 neurons in each layer
100n - 100 neurons in each layer
If we look closely, we'll see that models with more neurons have much lower number of epochs. But what we can't see here is that the training time spent in each epoch increased significantly also. Moreover, the usage of the GPU was much higher.

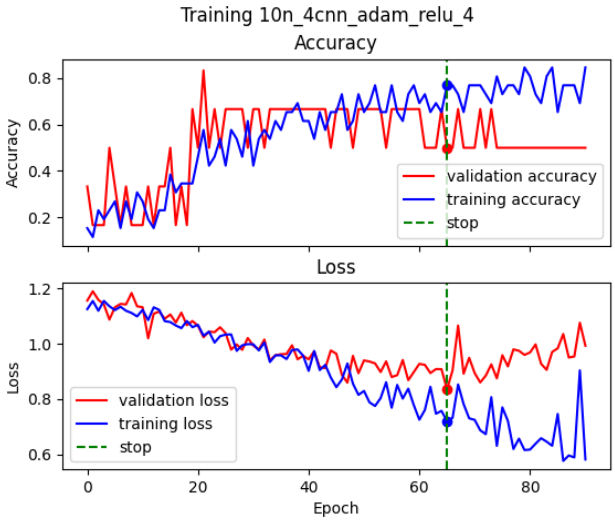


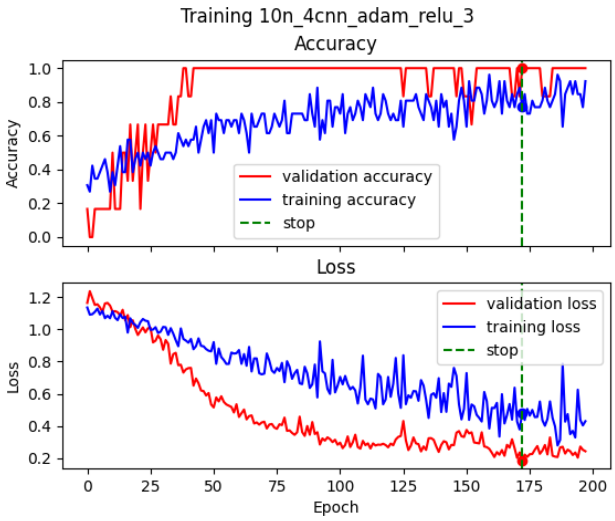
The chart shows that the gain from more neurons in terms of value of loss function is rather small if any. It looks like the best model here is the one with 10 neurons, but more testing should be done to confirm this conclusion.

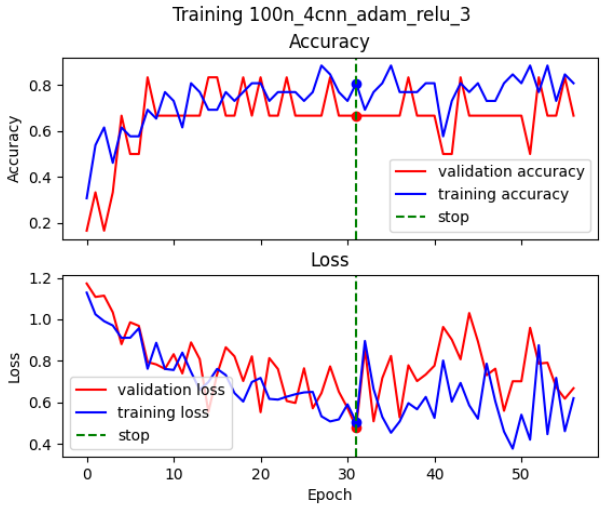


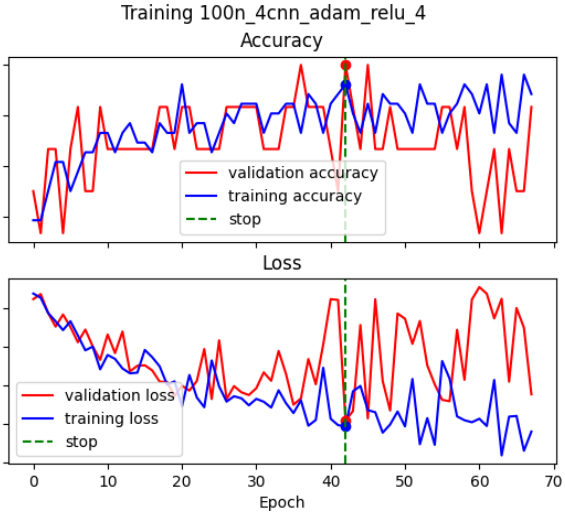
Here we can also see the similarity of the results for 10n and 100n architecture. The model with 50 neurons looks surprisingly bad, but it also has the smallest number of average epochs, so there may be some badly drawn starting conditions for that model. Again more tests are needed to conclude that.

From the training history we can conclude that the model with 10 neurons behaves

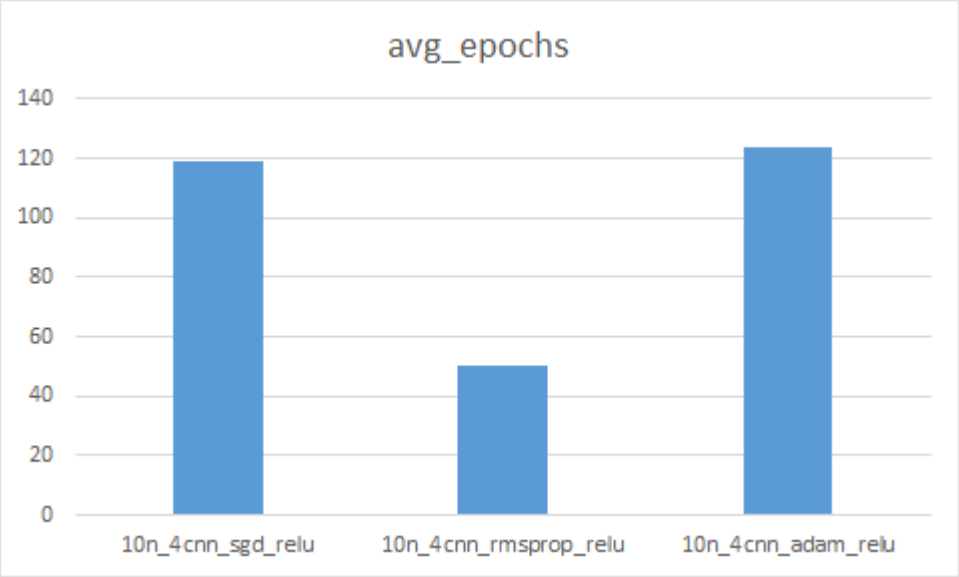








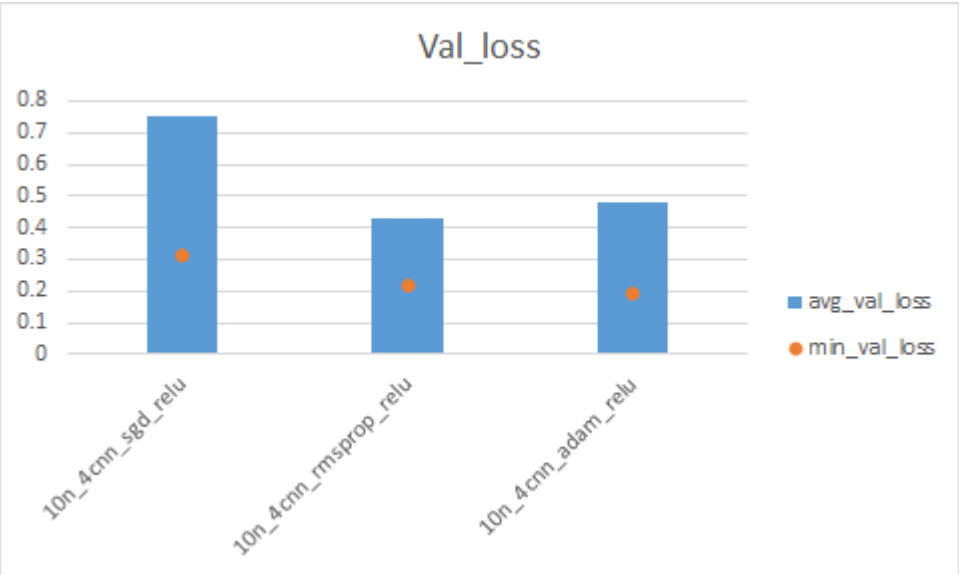
much more stable then the one with 100 neurons, because of that as the best one here we've chosen the **10n model**.



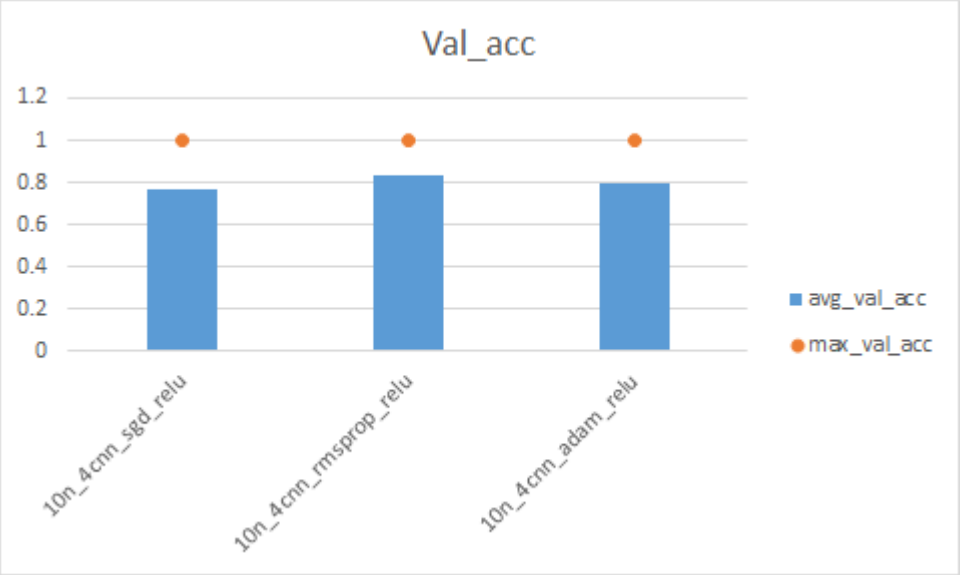
Proceeding to the optimizers we have:

- SGD - stochastic gradient descent
- Adam - stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments
- RMSprop - optimizer that implements the RMSprop algorithm

In terms of number of epochs Adam and SGD optimizers are similar, whereas RMSprop vastly improves the speed of the training.

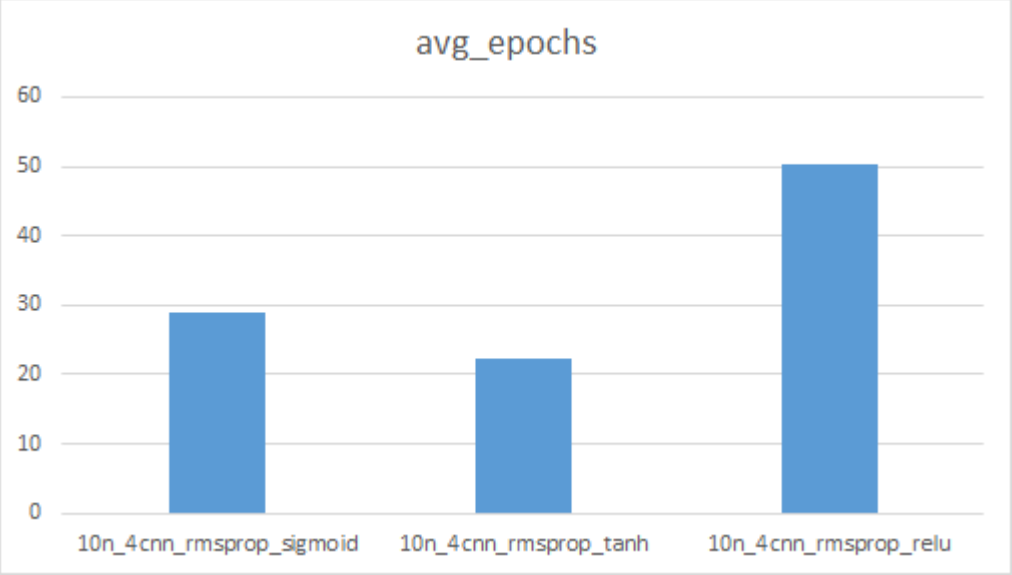


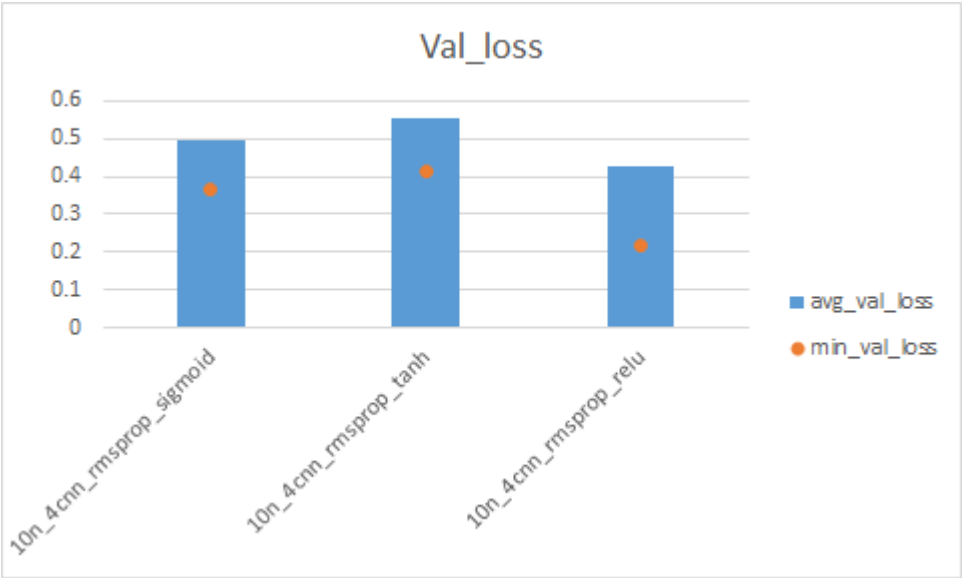
When it comes to loss results on validation set SGD performs far the worst and still RMSprop is in the lead.

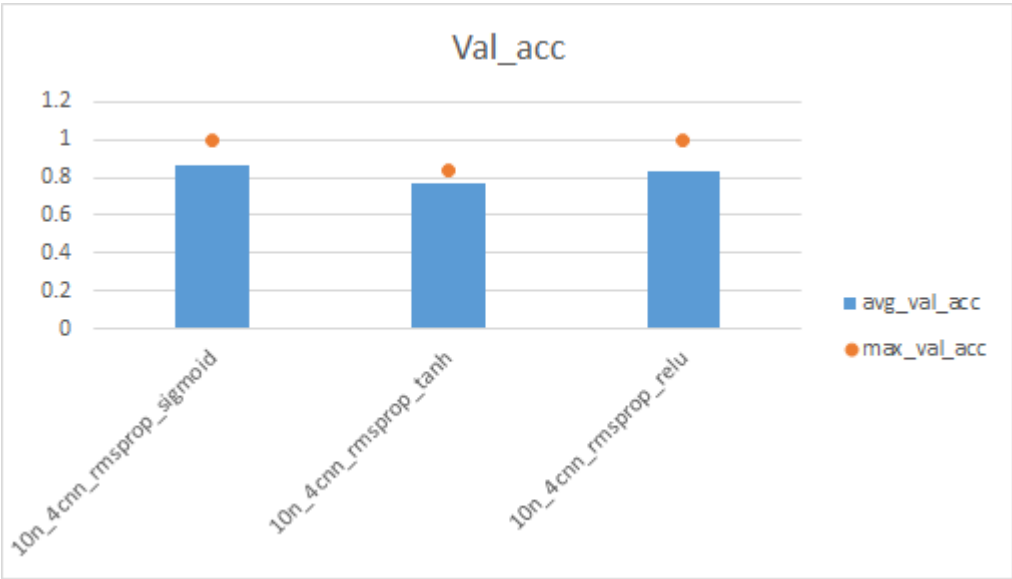


Finally, comparing the accuracy there is no big difference between these three optimizers.

All in all, as the best one we could consider RMSprop or Adam optimizer, they perform similarly, but **RMSprop** allows to speed up the training process, so here we'll pick RMSprop.

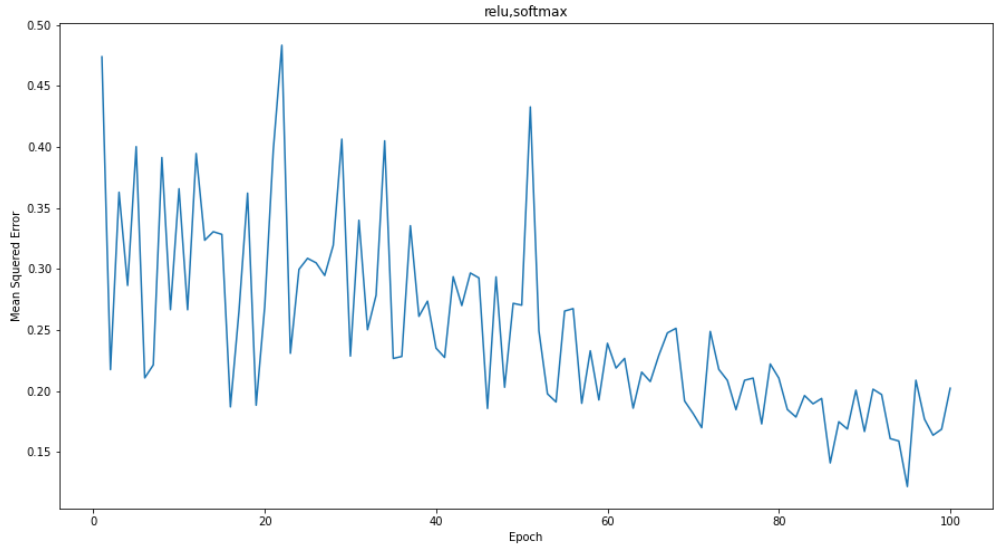


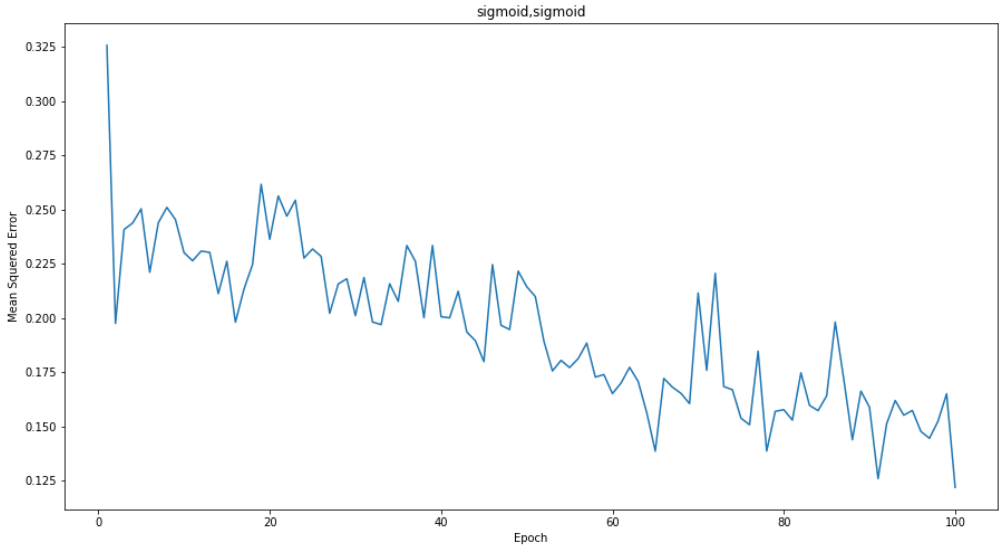


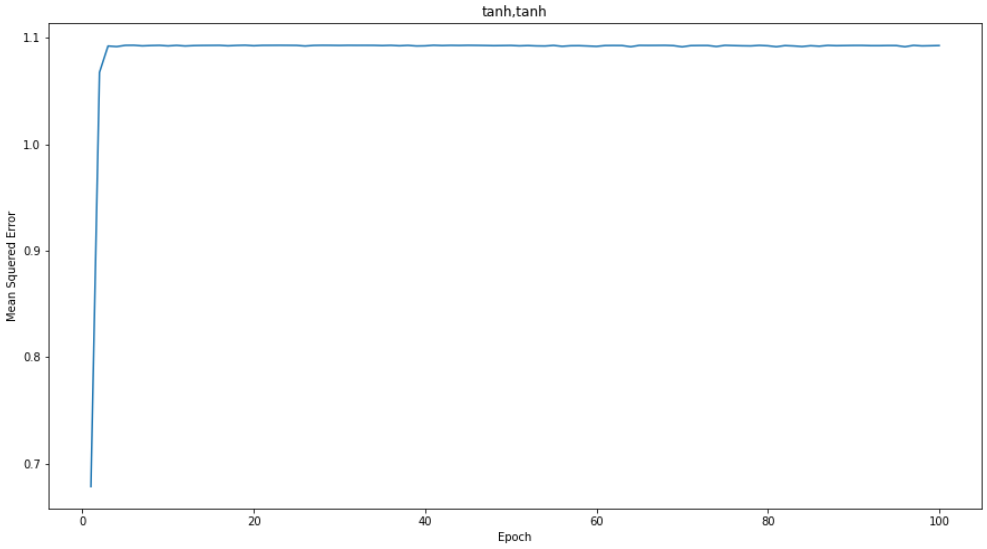


ReLu activation function achieved the best result. It also behaved most stable, probably because of that the amount of epochs was the largest.
Pretty easy choice here: **ReLu**

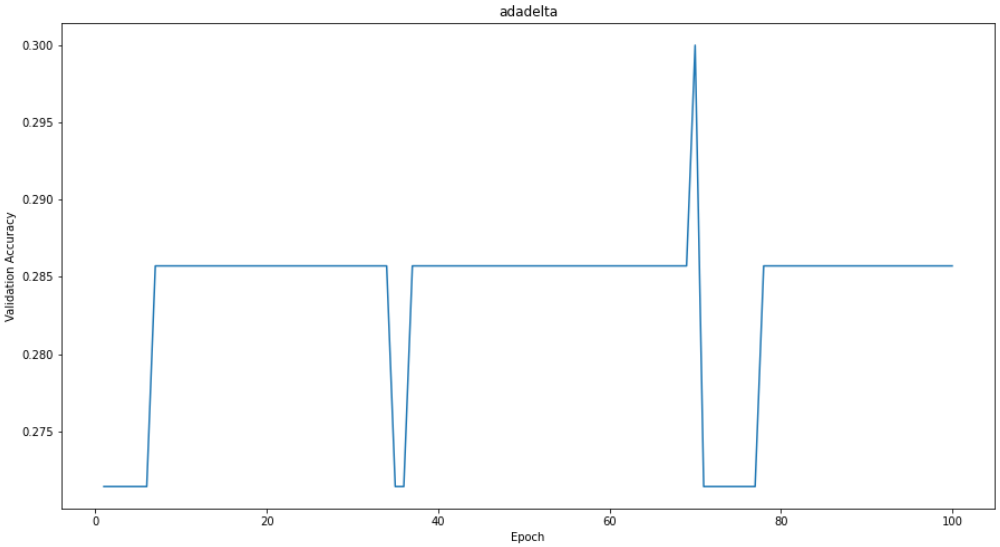
Comparison of MSE trends depending on the activation function shows that the results of relu + softmax functions and sigmoid function are satisfactory but results of tanh function are completely wrong.

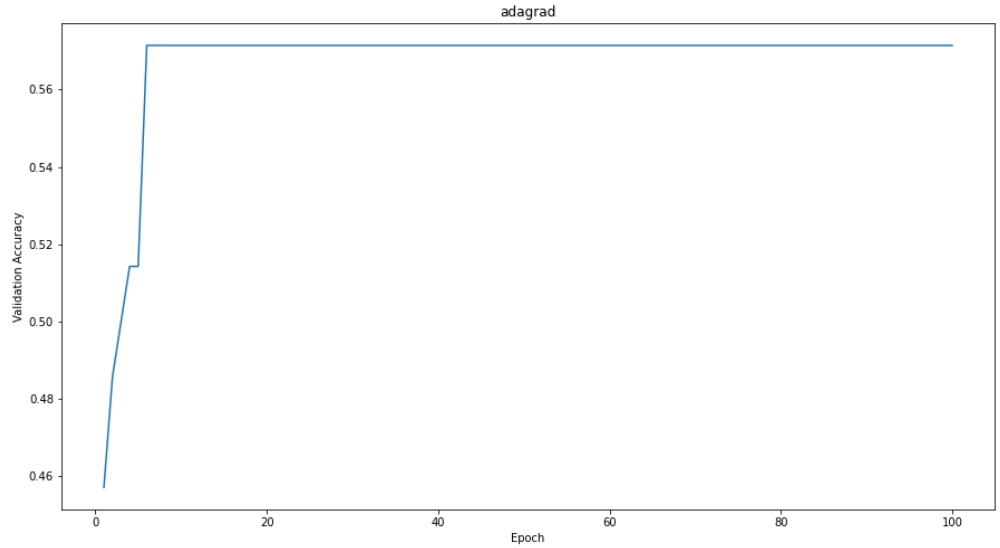

relu,softmax
Mean Squered Error
Epoch
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100

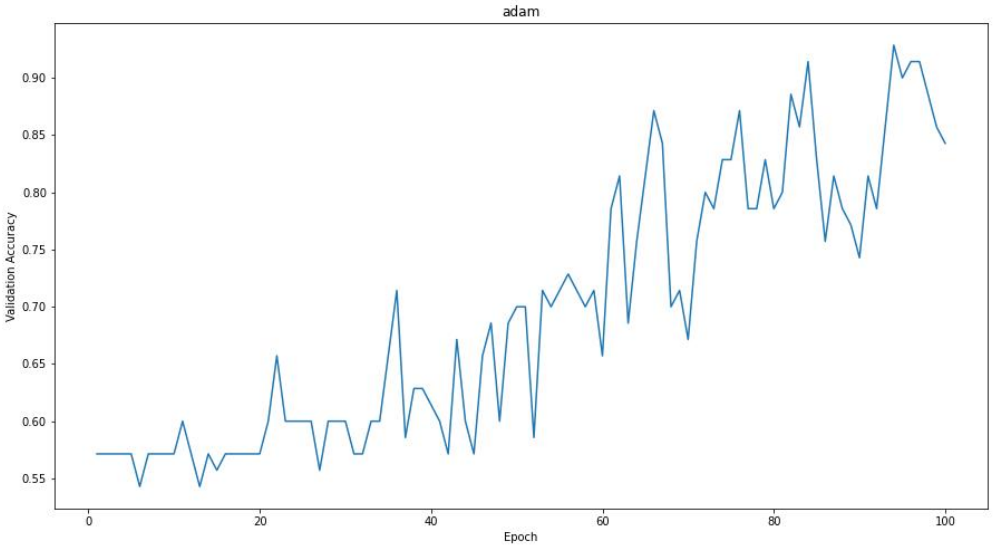

sigmoid,sigmoid
Mean Squered Error
Epoch
0.325
0.300
0.275
0.250
0.225
0.200
0.175
0.150
0.125
0
20
40
60
80
100

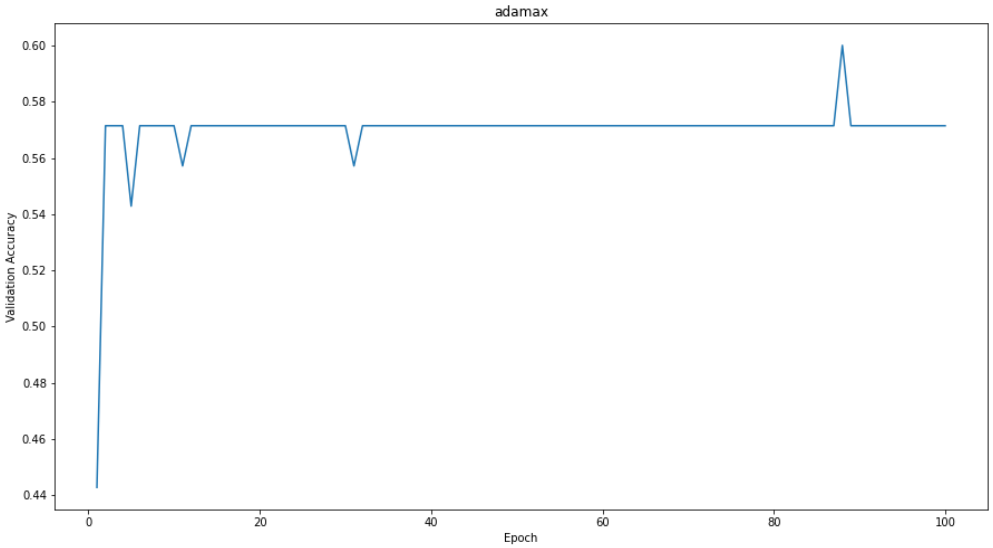


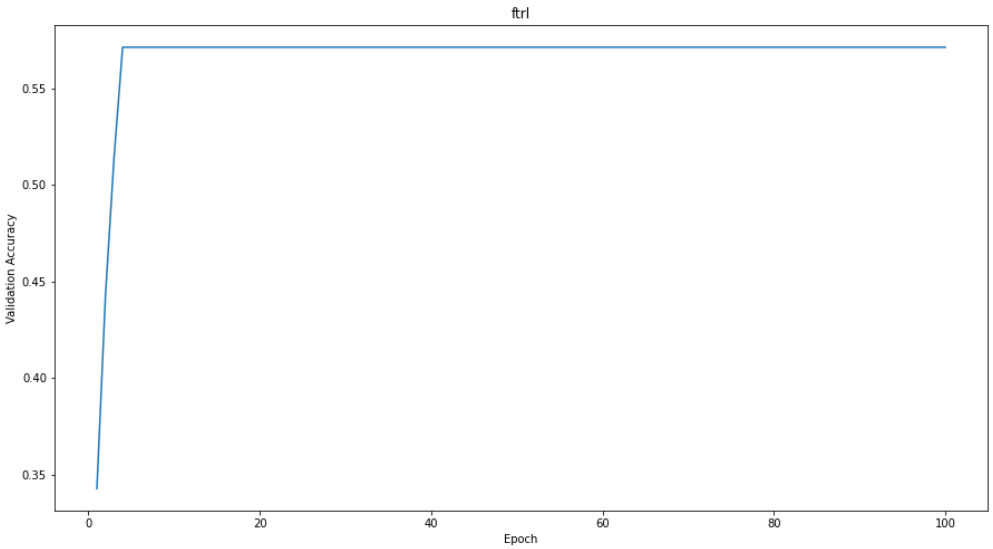
Validation Accuracy trends using different optimizers shows that the Adam, Nadam and Msprop optimizers achieved the best results and got stable behavior. Adadelta is interesting because we can see several sudden spikes.

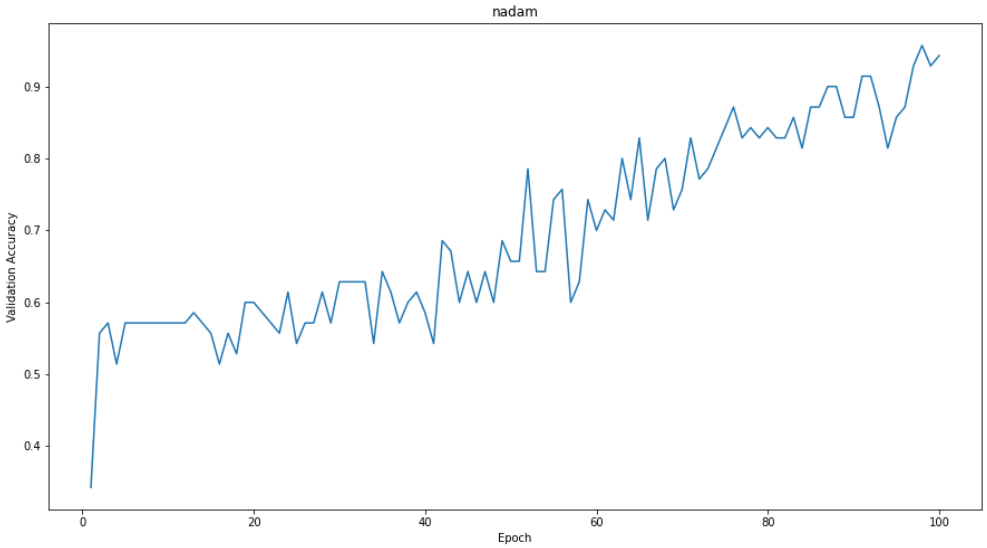

adadelta
Validation Accuracy
0.300
0.295
0.290
0.285
0.280
0.275
0
20
40
60
80
100
Epoch

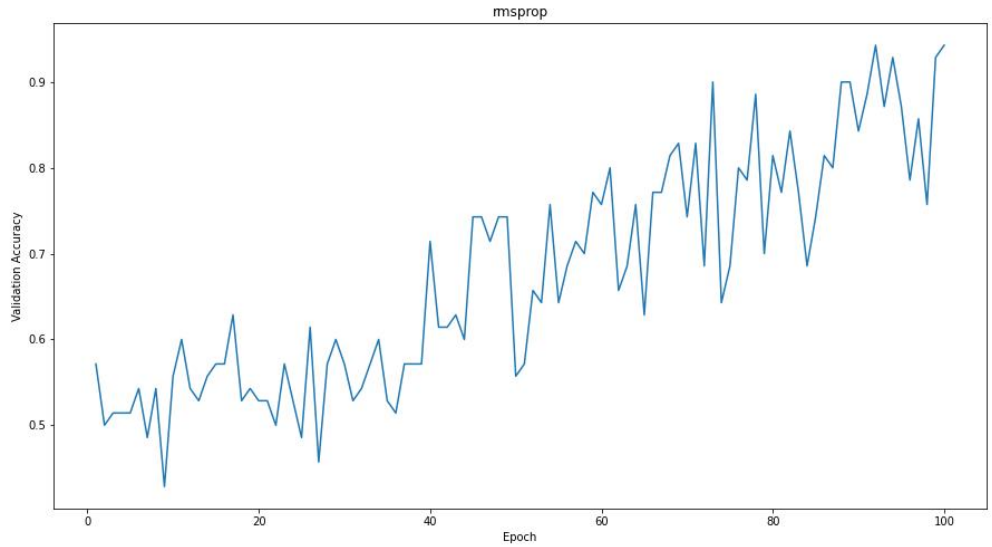

adagrad
Validation Accuracy
0.56
0.54
0.52
0.50
0.48
0.46
0
20
40
60
80
100
Epoch

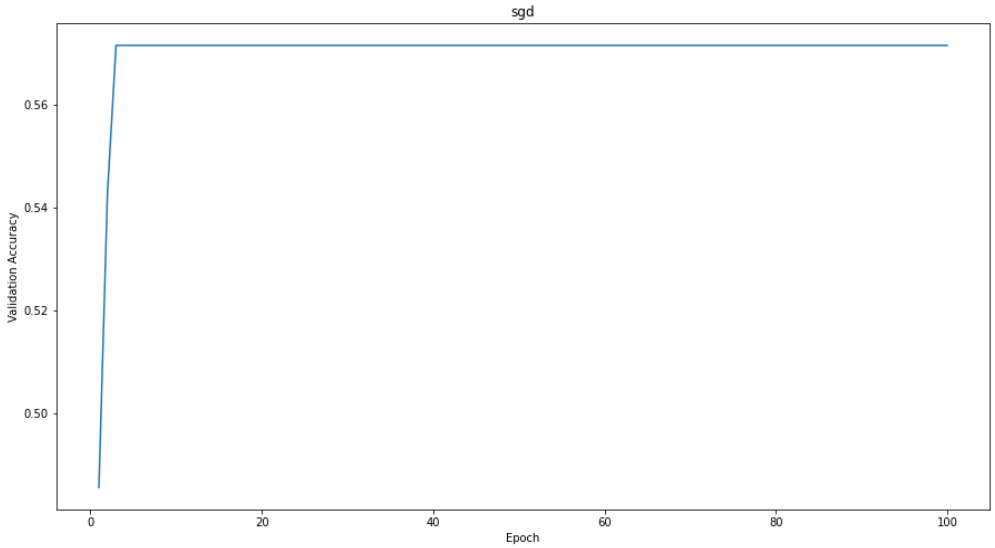

adam
Validation Accuracy
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
20
40
60
80
100
Epoch


adamax
Validation Accuracy
0.60
0.58
0.56
0.54
0.52
0.50
0.48
0.46
0.44
0
20
40
60
80
100
Epoch


ftrl
Validation Accuracy
0.55
0.50
0.45
0.40
0.35
0
20
40
60
80
100
Epoch


nadam
Validation Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0
20
40
60
80
100
Epoch





3.4. Conclusions (conclusions about the research method and tools, their validity)

Achieved results may not be valid because of the too small subset of data (validation set consists of only 3-6 samples). However, the PyTorch implementation may be a little bit more valid thanks to online augmentation (we were able to use more data for the validation set).

We will try the founded best hyperparameters in the real experiment execution.

To get better results the process of experiments may be easily applied on bigger set of data in the future.

4. Conclusions

4.1. Design (conclusions and lessons learnt about the research design)

The design of the application is well-prepared. However, without enough data it will not work properly, so we now should focus on data gathering and labeling.

4.2. Pilot study (conclusions and lessons learnt about the pilot study)

The best hyperparameters found are: Adam optimizer and RMSprop (which allows to reduce training length), ReLu activation function, 4xCNN architecture with 100 neurons in each layer. It took on average 150 epochs to train the model.

5. Literature (list of literature used in the research, articles from SLR)

- Systematic Literature Review, Research methods in informatics lecture, Jakub Miler
- Systematic Literature Review Plan Example, J. Miler, O. Springer
- 1D convolutional context-aware architectures for acoustic sensing and recognition of passing vehicle type, A. Kurowski; S. Zaporowski; A. Czyżewski, 2020
- Acoustic Emission Signal Classification Using Feature Analysis and Deep Learning Neural Network, Wu J.-D., Wong Y.-H., Luo W.-J., Yao K.-C., 2021
- Acoustic Signal Classification Using Symmetrized Dot Pattern and Convolutional Neural Network, Wu J.-D., Luo W.-J., Yao K.-C., 2022
- Audio Feature Extraction for Vehicle Engine Noise Classification, L. Becker; A. Nelus; J. Gauer; L. Rudolph; R. Martin, 2020
- Automatic labeling of traffic sound recordings using autoencoder-derived features, A. Kurowski; A. Czyżewski; S. Zaporowski, 2019
- Deep Recurrent Neural Networks for Audio Classification in Construction Sites, M. Scarpiniti; D. Comminiello; A. Uncini; Y. -C. Lee, 2021
- Hybrid Neural Network based on Feature Fusion for Vehicle Type Identification, H. Chen; Z. Zhang; W. Yin; M. Wang; M. Lifan; X. Hao, 2020
- Recognition of Moving Tracked and Wheeled Vehicles Based on Sound Analysis and Machine Learning Algorithms, Jackowski J., Jakubowski J., 2021
- Sound-Convolutional Recurrent Neural Networks for Vehicle Classification Based on Vehicle Acoustic Signals, Luo Y., Chen L., Wu Q., Zhang X., 2021
- Sound-based Transportation Mode Recognition with Smartphones, L. Wang; D. Roggen, 2019
- Spectral features for audio based vehicle and engine classification, Alicja Wieczorkowska, Elżbieta Kubera Tomasz Słowik Krzysztof Skrzypiec, 2018
- Transportation mode detection using cumulative acoustic sensing and analysis, Dinesh VijNaveen Aggarwal, 2020
- Unsupervised vehicle recognition using incremental reseeding of acoustic signatures, Sunu J., Percus A.G., Hunter B., 2018
- Vehicle Identification Based on Improved 1/3 Octave and Bark-Scale Wavelet Packet Methods, Q. Wang; Y. He; Z. Chen; Y. Luo, 2021
- Environmental Intelligence for Embedded Real-time Traffic Sound Classification, P. Montino; D. Pau, 2019
- Sound event classification using neural networks and feature selection based methods, A. Ahmed; Y. Serrestou; K. Raoof; J. -F. Diouris, 2021
- Machine Learning Inspired Sound-Based Amateur Drone Detection for Public Safety Applications, M. Z. Anwar; Z. Kaleem; A. Jamalipour, 2019

- Research on Environmental Sound Classification Algorithm Based on Multi-feature Fusion, Ruixue Li; Bo Yin; Yongchao Cui; Zehua Du; Kexin Li, 2020
- VehicleSense: A Reliable Sound-based Transportation Mode Recognition System for Smartphones, Sungyong Lee, Jinsung Lee, Kyunghan Lee, 2017
- Moving Vehicle Noise Classification using Multiple Classifiers, N. Abdul Rahim, Paulraj M P, A. H. Adom, and S. Sathish Kumar, 2011