**Introduction**

Time Series classification is a growing field in medical application. ECG is a common time-series recorded in the medical field. Apart from diagnostic in a clinical setting analyzing ECG is also of interest for application in smartwatches and home monitoring devices. With the increasing amount of data, it gets challenging to classify all the recordings by hand. An automatic Algorithm helping the doctor to analyze the ECG recordings would be desirable.

Two applications are from interest, first classifying between normal and abnormal ECG and second classifying the ECG after certain diagnoses.

**Task**

In the task both classification problems were considered. Both datasets come with there on challenges regarding the development/deployment of ML algorithms. The dataset ptbdb was relatively small, which reduces training time, but at the same time, the data isn’t distributed well between the two classes, which might negatively affect training performance.

**Baseline**

Together with the datasets a baseline code was provided. The Network consists of simple Conv1D layers. The performance of the baseline code is:

|  |  |  |
| --- | --- | --- |
| Dataset | F1 score | Acc Test Set |
| Ptbdb (2 classes) | 1.5 | 98.8 |
| Mitbih (5 classes) | 98.5 | 98.3 |

**The following Approaches were evaluated**

1. **Vanilla CNN**

The provided baseline code is already a good realization of a Vanilla CNN. The baseline code was adapted by a small manual grid search. Since the baseline model already performed very good, a full grid search did not seem reasonable with respect to resources and time.

For both datasets 19 Models were evaluated exploring different structures of the CNN. Number of layers, activation function, filter and kernel size were varied. The three best models are listed here:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Dataset | F1 score | Acc Test Set | AUCROC | AUCPR |
| Ptbdb (2 classes) | Model\_1p | 0.9959 | 0.9942 | 0.9997 | 0.9997 |
| Model\_2p | 0.994 | 0.991 | 0.9997 | 0.9999 |
| Model\_3p | 0.993 | 0.99 | 0.9993 | 0.9997 |
|  |  |  | Acc | | |
| Mitbih (5 classes) | Model\_1m | 0.9147 | 0.9856 | | |
| Model\_2m | 0.917 | 0.9854 | | |
| Model\_3m | 0.911 | 0.9845 | | |

Model\_1p: The number of layers was decreased by two compared to baseline

Model\_2p: The structure from the baseline Model wasn’t changed. However instead of relu, tanh was used for activation layer.

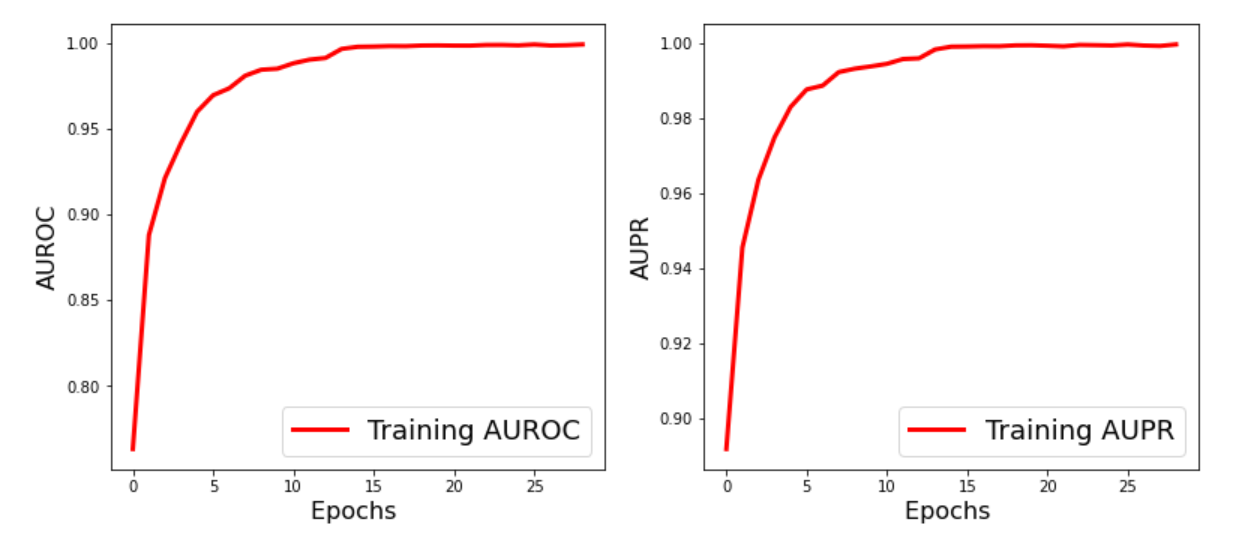
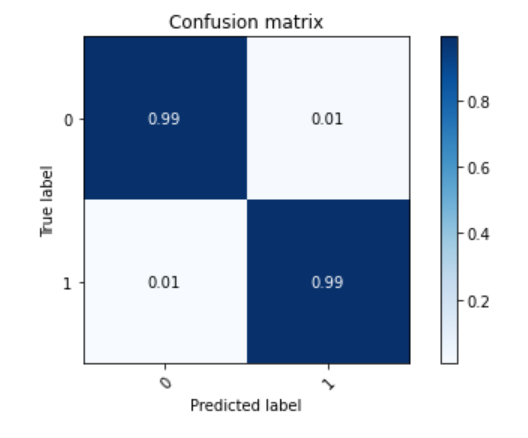
Model\_3p: For each Conv-Block of the baseline model an additional layer was added.

Model\_1m: The Conv layers were the same as in the baseline model in addition LAyerNormalization was carried out after Conv layers.

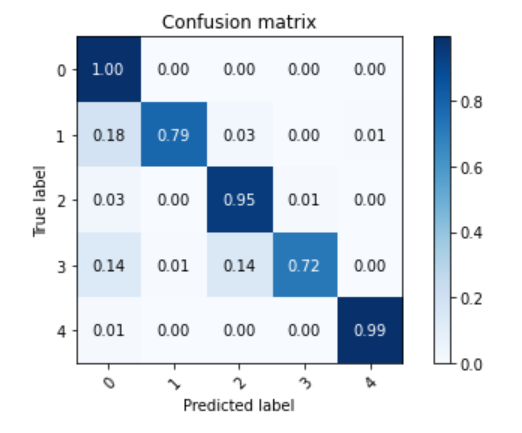
Model\_2m: The number of layers was decreased by two compared to baseline

Model\_3m: An additional Conv block with two layers with filter size 64 was added to the baseline model.

Model\_3p:

Model\_2m:



From the presented models it is clear, that for the different datasets different models perform better. This is helpful information for the following Tasks, since it shows that it is worth it to look at both datasets separately.

From the confusion matrices it was visible that that some classes werde predicted much better than others. By looking at the distribution of the classes in the training set, it is obvious that the classes are not equally distributed. Changing the training set and distributing classes equally might improve prediction performance for classes which are rare.

However due to time reasons this wasn’t performed for this task.

In addition to the presented models, very deep models with more than twice as much layers than the baseline model were evaluated as well. However, making the network to deep lead to complete failure.

1. **Parallel CNN with data transformation**

Since the baseline model already yielded very good results, we continued exploring different approaches and architectures instead of continuing grid search for the Vanilla CNN.

One of the new approaches was to implement data transformation. Forming a parallel pipeline of several CNNs.

baseline

Raw data

baseline

Sampled data

output

Dense layer

baseline

Smoothed data

The architecture was evaluated for different concatenations of smoothed and sampled data. The idea was that by applying transformations the main features would be easier to extract. As we expect the important features to be unaffected by data transformation.

The smoothing transformation however led to a complete failure of prediction. Adding parallel pipelines with sampled data lead to reasonable results.

The best model yielded accuracy 0.9835 and f1 0.903.

Applying data transformation might be promising for some applications. However especially in a medical content it must be ensured that data transformation really conserves the meaning of the data. For ECG data transformation might lead to information quite fast, since ECG series have a very special structure.

1. **CNN with residual blocks**

Since the input data represents time series adding residual blocks is expected to yield better results by adding a memory. Different architecture using identity blocks and convolutional blocks were evaluated.

For the mitbh dataset CNN with residual blocks with a very deep structure performed approximately the same as CNN with shallower structure. (E.g. only one residual block) The performance with residual blocks however did not significantly improve compared to CNNs without res blocks.

The evaluation showed that adding residual blocks to the CNN lead to a complete failure of prediction for the ptbdb dataset. Always predicting class 1.