Machine Learning for Health Care - Project 4 Report

Manuel Burger, Tobias Peter, Anej Svete [*]

June 2021

1 Introduction & Overview

This report provides a concise summary and discussion of our submission for the fourth project for the class *Machine Learning for Health Care*.

The objective of the project was to classify ECG time series data, specifically pre-processed individual heartbeat templates, on two different datasets. Evaluation of the classification is based on the ROC (receiver operating characteristic) and PR (precision-recall) curves (binary case only), conventional accuracy and F1-score (with macro averaging). In the following, we will describe the various models that we used, their training as well as their performance. Details can be found in the corresponding Jupyter notebooks, which are referenced in the respective sections.

2 Analysis

The provided data has already been preprocessed very well. All templates are scaled to a $[0, 1]$ range, cut to same lengths, and have been sampled at the same frequency. Class-wise distribution plots (Fig. 1) of the templates even show visually distinctive features for some classes.

The MITBIH dataset shows a large imbalance between class 0 (the normal healthy heartbeats) and the minority classes with the different anomalies. The PTB dataset also shows an imbalance albeit much smaller.

3 Modeling & Training

This section lists and describes the models we used in the assignment. Architecture search was usually based on the more challenging classification task for the MIT-BIH dataset. However, all models were trained on both datasets while some were also used for transfer learning, as per the assignment description.[1]

3.1 Baseline & Modifications

`baselines.ipynb`
`baseline_improvements.ipynb`

The provided baseline model is a multi-layer one-dimensional CNN, which already has a very good performance. We therefore first tried to fine-tune its architecture by for example modifying the number of filters and the sizes of the kernels. Secondly, we also used a simple LSTM model.

3.2 CNN with Dilated Convolutions

`additional_cnn.ipynb`

Dilated convolutions have been successfully used in semantic image segmentation or machine translation as they deliver a wider field of view resp. context at the same computational cost.

We therefore also tested a CNN architecture with dilated convolutions, which in our case captures a larger part of the signals. To further improve upon it, we additionally added a Gaussian noise layer resp. a residual connection with features extracted from the signals (e.g. in the spectral domain). However, both additions slightly reduced the performance of the model and are not reported in Tables 1, 2 and 3.

3.3 CNN with Residual Layers

`residual_cnn.ipynb`

We also implemented a CNN model that feeds the signals directly into a residual connection instead of separately extracted features. The configuration used in the end was determined through a hyperparameter search. Compared to a standard implementation of the network, we experimented with the *swish* activation function, which often outperforms the standard *ReLu* function.

3.4 CNN with Attention Layers

`attention_cnn.ipynb`

To build upon the baseline CNN architecture we introduced a custom attention layer. The attention layer creates a more informed average over all the extracted filters in the last convolutional layer. The attention layer learns multiple context vectors to pay varying

[*]{burgerm, topeter, asvete}@ethz.ch

[1]Transfer learning is restricted to weight initialization as the baseline showed a significant negative effect when freezing layers.

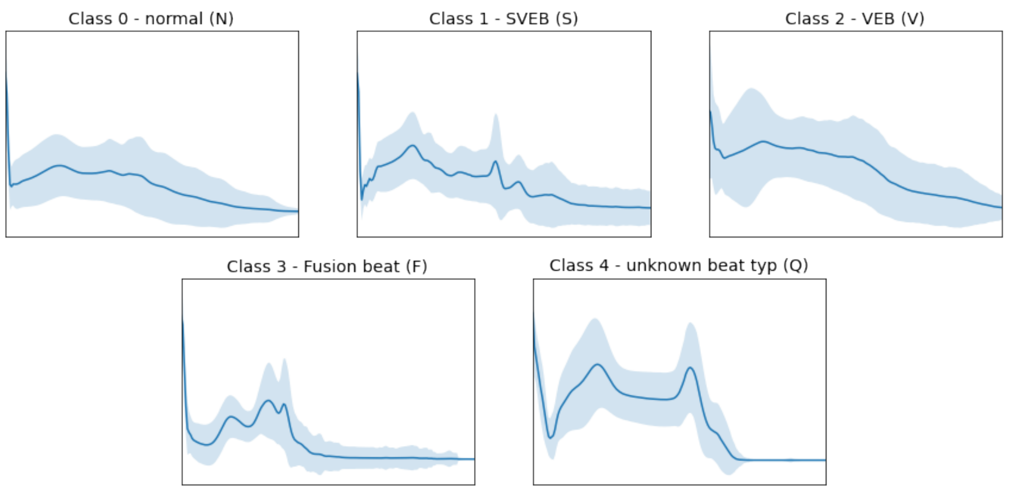

Figure 1: Class-wise distribution plots of the heartbeat templates

amounts of attention to different sets of learned features. Additionally, we feed in extracted fourier features and again use an attention layer to make an informed combination with the CNN features.

3.5 Bidirectional LSTM

`bilstm.ipynb`

A bidirectional LSTM (BiLSTM) enables additional training by traversing the input data twice; from beginning to end and reversed, respectively. It is therefore often observed that the bidirectional LSTM outperforms their one-directional counterpart.[2]

3.6 Patch-based transformer

`patch_transformer.ipynb`

Transformers have been established as state of the art models in NLP and have recently also shown great success on image based tasks. To use a transformer architecture on a discretized continuous signal such as an image or a sensor signals (e.g. our ECG data) it is common practice to divide the input into different patches. This keeps the quadratically growing computational effort of the attention mechanism controlled.

We thus extract patches of fixed length from our input signal and join them with positional embeddings. The patches are then fed through 3 transformer layers. The transformed patches are then concatenated, flattened and passed through a dense head classifier.

3.7 Variatonal Autoencoder

`conditional_cnn_vae.ipynb`

In addition to all the discriminative classifiers, we also explored generative modeling. We construct a CNN (meaning the encoder and decoder network use convolutional operators) based conditional variatonal autoencoder. In contrast to a classical autoencoder the variational autoencoder enforces a structured latent space through a prior distribution and with its probabilistic component allows for generative modeling and sample generation. The conditional component additionally allows to benefit from the shared feature space of the classes, while at the same time adapt the encoding and decoding to and from the latent space to the classes.

[2]This is confirmed in our case, since the BiLSTM - without any particular hyperparameter search - reaches a very competitive score in contrast to the one-directional model, cf. Section 4.

3.8 Ensemble

`ensembles.ipynb`

Finally, we combined the probabilistic predictions of the well performing models by averaging them or training a logistic regression classifier on top of them.

4 Results

In this section, we will briefly discuss the results of the various models. As can be seen in Table 1, 2 and 3, the provided baseline shows very good results and was difficult to beat. We restrict transfer learning to weight initialization as the baseline as well as our experiments showed a clear detrimental effect of freezing layers.

Moreover, we would like to emphasize that for some models the differences in scores are very small, such that it would need to be carefully tested whether these differences are significant and robust, and whether one model truly outperforms another one based on these metrics.

Model	Accuracy	F1 (macro)
Baseline[3]	0.9839	0.9099
CNN (Modified)	0.9872	0.9247
CNN (Dilated)	0.9888	0.9331
CNN (Residual)	**0.9897**	**0.9396**
CNN (Attention)	0.9855	0.9180
Simple RNN	0.9708	0.8541
BiLSTM	0.9867	0.9239
Patch Transformer	0.9850	0.9157
VAE	0.8275	0.5121
Ensemble Average	0.9850	0.9169
Ensemble Log. Reg.	0.9840	0.8927

Table 1: Results on the MITBIH dataset

4.1 Discriminative classifiers

4.1.1 MITBIH dataset

The various CNN, the BiLSTM as well as the patch-based transformer perform very similar to the baseline CNN on the MITBIH dataset, whereby the CNN with a residual connection from the signals themselves performs best by a slight margin.

The simple RNN has only a slightly lower Accuracy than the aforementioned models. However, it is clearly outperformed in terms of F1-score.

The used ensembles do not provide any additional benefit. This can be attributed to the fact that the individual models are probably already close to reaching the Bayesian error rate, thus any further improvement becomes generally difficult, and that there might be not enough variance among the different predictions to benefit from ensembling.

4.1.2 PTB dataset

On the PTB dataset, a similar picture emerges, but with the baseline being even harder to beat. In particular, without transfer learning we could not improve its already very high Accuracy and F1-score. Yet, the CNN with dilated convolutions at least reaches the same Accuracy and F1-score, and the modified baseline architecture lead to even slightly higher AUROC and AUPRC scores.

On the PTB dataset, we observe that the results tend to be better when transfer learning first on the MITBIH dataset, as these more complex models benefit from the larger amount of exposed data. However, this does not consistently hold; see AUROC and AUPRC score for the CNN with attention. Moreover, the positive effect is also not necessarily robust for a single model, e.g. it can change with a different seed.

[3]Note that we report our reproduced scores, which slightly differ from the ones on https://github.com/CVxTz/ECG_Heartbeat_Classification, cf. baselines.ipynb.

4.2 Variational Autoencoder

The results in Tables 1, 2 and 3 for this model have been computed in the following way: We train the complete VAE, then generate a dataset by sampling from the latent space and feeding it through the trained decoder. Then, using this purely generated dataset, we train the baseline CNN and validate the performance on the test set from the real distribution.

4.2.1 MITBIH dataset

The variational autoencoder provides an interesting additional insight into the data. By inspecting their distribution we concluded that the generated heartbeat templates are visually astonishingly close to the original distribution (see Fig. 1) for each class. By considering the confusion matrix in Fig. 2 we can see that the VAE was able to capture and generate the most essential features distinctive for class 0 and class 4, as the trained CNN performs quite decently on these two classes. The lacking capabilites of the VAE to capture class 1, 2, 3 better, can be attributed to the lack of data and the more subtle differences between these classes. For example class 2 is visually hard to separate from class 0 for the uninformed eye.

4.2.2 PTB dataset

On the PTB dataset the generative approach has been less successful. When transfer learning, the VAE does not seem to be able to capture any reasonable features of the true distribution, as can be seen by the 0.52 AuROC score in Table 3. The situation improves slightly without transfer learning (see Table 2) as now an F1 score of 0.64 is achieved. However, analysis of the confusion matrix still shows many false positives i.e. a strong tendency to just predict the majority class and thus we conclude that the VAE was not able to capture the underlying distribution of the data well enough. This can be attributed to the lack of data, as sufficient data is essential for good generative modeling.

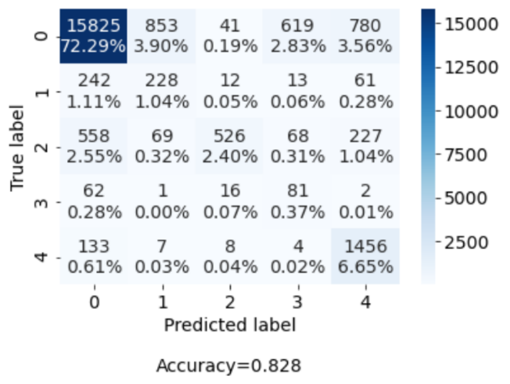

Figure 2: Confusion Matrix of test set predictions by the baseline CNN trained on VAE generated data of the MITBIH dataset

Model	AUROC	AUPRC	Accuracy	F1 (macro)
Baseline	0.9982	0.9991	**0.9941**	**0.9927**
CNN (Modified)	**0.9985**	**0.9993**	0.9924	0.9906
CNN (Dilated)	0.9974	0.9979	**0.9941**	**0.9927**
CNN (Residual)	0.9871	0.9904	0.9918	0.9897
CNN (Attention)	0.9979	0.9990	0.9825	0.9779
Simple RNN	0.8120	0.9210	0.7606	0.7049
BiLSTM	0.9955	0.9968	0.9824	0.9781
Patch Transformer	0.9969	0.9981	0.9886	0.9859
VAE	0.7537	0.8886	0.7128	0.6389

Table 2: Results on the PTB dataset *without transfer learning*

Model	AUROC	AUPRC	Accuracy	F1 (macro)
Baseline	0.9979 (−0.03%)	0.9989 (−0.02%)	0.9951 (+0.10%)	0.9940 (+0.13%)
Baseline frozen	0.9881 (−1.01%)	0.9947 (−0.44%)	0.9532 (−4.09%)	0.9425 (−5.02%)
CNN (Dilated)	0.9986 (+0.12%)	0.9991 (+0.12%)	0.9955 (+0.14%)	0.9944 (+0.17%)
CNN (Residual)	0.9957 (+0.86%)	0.9968 (+0.64%)	**0.9966** (+0.48%)	**0.9957** (+0.60%)
CNN (Attention)	0.9968 (−0.11%)	0.9969 (−0.21%)	0.9918 (+0.93%)	0.9897 (+1.18%)
BiLSTM	**0.9989** (+0.34%)	**0.9995** (+0.27%)	0.9955 (+1.31%)	0.9944 (+1.63%)
Patch Transformer	0.9969 (±0.00%)	0.9979 (−0.02%)	0.9904 (+0.18%)	0.9880 (+0.21%)
VAE	0.5204 (−23.3%)	0.7612 (−12.7%)	0.7221 (+0.93%)	0.4193 (−22.0%)

Table 3: Results on the PTB dataset *with transfer learning.*
In parentheses, absolute difference to the score without transfer learning, see Table 2.

5 Further Work

The baseline already offers great performance that proved difficult to beat. Since combining results of different models often offers a boost to the performance, one possible improvement to our work might be to train the logistic regression model on top of our predictions on a *separate* training dataset that the *original models were not trained on* (obtained for example by splitting the original training data into two parts), since our current approach, where logistic regression was fitted on the same training data as the individual models, runs a high risk of overfitting to these samples.

Main Contributions

M. Burger	CNN with Attention Layers, Patch-based Transformer and Variatonal Autoencoder.
T. Peter	CNN with Dilated Convolutions, Gaussian layer and Skip Connection, Bidirectional LSTM, Baselines
A. Svete	Analysis of the baseline and the simple enhancements, CNN with residual layers, an RNN with attention layers, work on the bidirectional RNN