Machine Learning for Healthcare
Project 1: ECG Heartbeat Classification

Alicja Chaszczewicz [*], Simon Heinke [†]Anastasia Sycheva [‡]

Department of Computer Science, ETH Zurich

March 19, 2020

1 Introduction

The objective of the project was to experiment with different neural network architectures for ECG heartbeat classification. The analysis was performed on two datasets: MIT-BIH Arrhythmia dataset and PTB Diagnostics dataset. The former, larger dataset was classified into five categories, the latter into two. Since the input for both tasks is the same, there is a possibility that higher-level signal representations learned for one task are useful for another, thereby reducing the total number of observations required to fit the model. Thus in the last part of our report we explore the effects of transfer learning on the classification.

This report is a concise summary of our work. Following every subsection we provide links to corresponding notebooks in the submission directory.

2 Analysis

2.1 Data Preprocessing

We use the the code from the baseline models to load the data, extract labels and also perform train-test split. We standardise the amplitude values to have mean value of 0 and standard deviation of 1 before we train all our models.

2.2 Data Visualization

We embed high-dimensional intermediate activation tensors from a pretrained residual CNN into a two dimensional space using t-SNE ((MAATEN; HINTON, 2008)) and UMAP ((MCINNES *et al.*, 2018)) algorithms. As the datasets are highly imbalanced, for the visualisations we used balanced samples of the datasets with equal number of observations for each class. The created embedding could serve as a nice tool to visually assess the discriminating power of the models on one hand, and to see the effect of adding extra layers on the other.

MIT-BIH dataset: *./notebooks / MITBIH_visualize.ipynb*

PTB dataset: *./notebooks /PTB_visualize.ipynb*

2.3 Model Fitting

We have reused the model fitting framework from the baseline models. All models were trained using Adam optimization algorithm ((KINGMA; BA, 2014)). 10% of the training data was held back to serve as a validation set. In case of stagnation first the learning rate was decreased and if this did not help, the training was stopped before all the epochs were completed. Subsequent sections contain more details on the other model architectures.

2.3.1 Smaller CNN

Since the baseline CNN seemed rather large, we decided to implement a smaller CNN for comparison. Our new model combines different non-linearities in the convolutional layers and gets rid of the fully connected layers at the end of the network. It is roughly 4 times smaller than the baseline CNN (25.48% trainable weights) but still achieves marginally better results (+0.1% F1-score on MIT-BIH).

MIT-BIH dataset:

./notebooks / MITBIH_other_models.ipynb

PTB dataset:

./notebooks / PTB_other_models.ipynb

2.3.2 RNN and Bidirectional RNN

We have used a simple model with two LSTM layers and two Dense layers. This architecture performs markedly worse than our CNN model. To boost the model performance we have switched to Bidirectional LSTM layers and did a naive hyperparameter tuning. We have tried seven different architectures and were able to increase accuracy by 6% from 86% to 92.1%.

MIT-BIH dataset:

./notebooks / MITBIH_RNN.ipynb

PTB dataset:

./notebooks / PTB_RNN.ipynb

./notebooks / PTB_rnn_experiments.ipynb

2.3.3 Residual CNN

We have implemented the Residual CNN architecture (ResNet) to tackle the classification problem. Both baseline and our CNN models have more than 15 layers which makes them hard to train. (HE *et al.*, 2016) have suggested to partially eliminate this issue by introducing skip connections. Their architecture was successfully applied to many classification tasks and also yielded the best results on both of our datasets.

MIT-BIH dataset:

./notebooks / MITBIH_other_models.ipynb

PTB dataset:

./notebooks / PTB_other_models.ipynb

[*]achaszcze@ethz.ch
[†]sheinke@ethz.ch
[‡]asycheva@ethz.ch

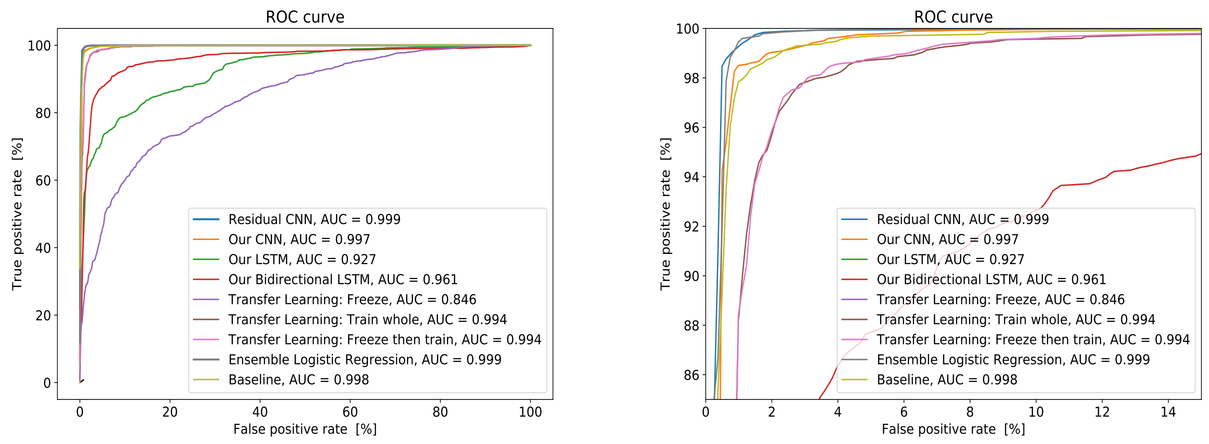

Figure 1: ROC curves for all models trained on the PTB dataset

Model	MIT-BIH		PTB	
	accuracy	F1	accuracy	F1
Baseline	**0.985**	**0.916**	**0.986**	**0.990**
Our CNN	0.985	0.917	0.988	0.991
Our LSTM	0.977	0.879	0.869	0.913
Our Bidirectional LSTM			0.967	0.977
Residual CNN	**0.988**	**0.930**	**0.993**	**0.992**
Ensemble naive	0.988	0.927	0.987	0.984
Ensemble logistic			**0.994**	**0.996**
Transfer learning: freeze			0.801	0.729
Transfer learning: train whole			0.971	0.963
Transfer learning: freeze then train whole			0.978	0.973

Table 1: Model performance comparison. Numbers are taken from *SUMMARY_project_1.ipynb* and averaged over several runs

2.4 Model Ensembles

We have combined the predictions from different classifiers in the hope of improving model performance. We test two approaches: a) simple averaging of the votes[1] b) logistic regression classifier. Results in Table 1 and ROC curves from Figure 1 indicate that ensemble classifiers slightly outperform all but one classifier (Residual CNN). Furthermore, a more elaborated aggregation method (logistic regression) yields better results w.r.t. accuracy and F1 score.

MIT-BIH dataset:

./notebooks / MITBIH_ensemble.ipynb

PTB dataset:

./notebooks / PTB_ensemble.ipynb

2.5 Transfer Learning

As we have previously mentioned in the introduction, MIT-BIH dataset is much larger than PTB dataset. In this section we explore, whether "knowledge" gained while training models on the former dataset could be transferred to improve classification rate on the latter.

We use our LSTM architecture to conduct three experiments. In the first experiment, we take the weights from the first two LSTM layers and add two dense layers. Only these newly added dense layers are then trained on the PTB dataset, while the LSTM layers remain frozen (*freeze*). In a second experiment, we simply initialize the model with the pretrained weights, but keep all weights trainable (*train whole*). In the third approach, we first freeze the layers, train the newly added layers, and then we unfreeze the whole model and train it (*freeze then train whole*). The third procedure yields the best result, significantly better than the first method, yet does not outperform neither the ResNet nor the baseline model.

PTB dataset:

./notebooks / MITBIH_to_PTB_transfer_learning.ipynb

[1]Could use weighted averages instead, where weights are proportional to accuracy / F1 score. These values do not vary significantly across the models

3 Conclusion

Table 1 summarizes the performance of different neural network classifiers. Residual CNN (HE *et al.*, 2016) appears to have the best performance on both datasets. In general, across all our models, CNN outperform RNN for the task of ECG classification. Model aggregation improves accuracy and F1-scores. In contrast, transfer learning does not seem to bring any substantial benefits. Furthermore, the worst performing model was obtained by freezing the weights trained on MIT-BIH dataset.

Please read the provided notebooks to see detailed task specific results.

References

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

MAATEN, L. v. d.; HINTON, G. Visualizing data using t-sne. **Journal of machine learning research**, v. 9, n. Nov, p. 2579–2605, 2008.

MCINNES, L.; HEALY, J.; MELVILLE, J. Umap: Uniform manifold approximation and projection for dimension reduction. **arXiv preprint arXiv:1802.03426**, 2018.