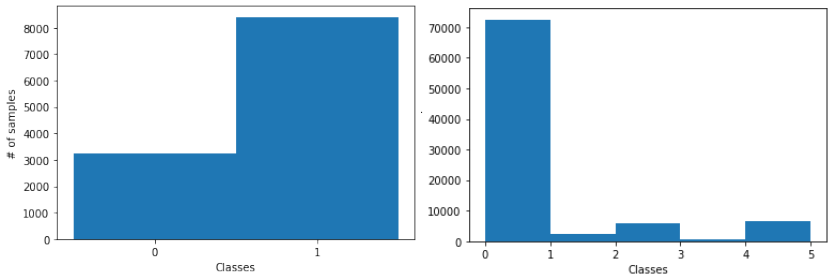
ML4HC Project 1

Rafael Bischof, Thierry Backes, Julia Ortheden, Levin Moser

1 Analysis of dataset



(a) PTB (b) MITBH

Figure 1: Visualization of the data distribution. (a) Shows the PTB dataset and (b) the MITBIH data set. Both data sets contain a higher number of controls than cases.

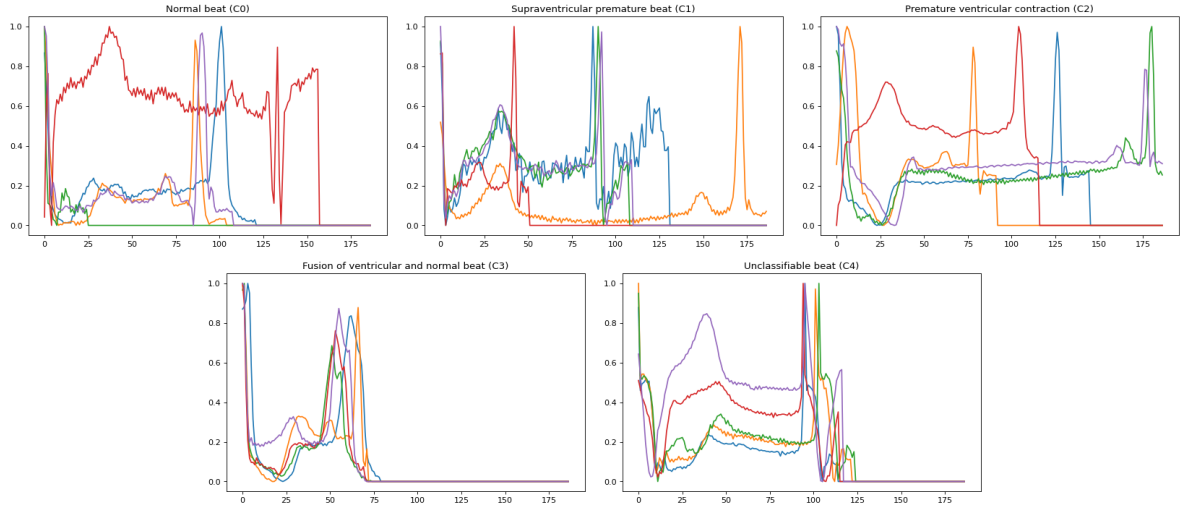


Figure 2: Plot of five random samples from the MIT dataset.

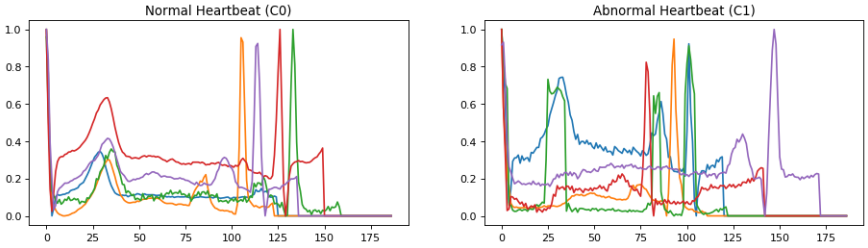


Figure 3: Plot of five random samples from the PTB dataset.

The Figures 1, 2 and 4 give an insight of the dataset distributions. Figure 1 displays the large imbalance of the data sets. The Plots in 2 and 4 emphasise the fluctuation of samples within classes. We observe that most classes do not have a clear unique pattern.

2 Models & Training

We tried different approaches to predict both datasets, which can be broadly categorised in three: standard networks, transfer learning, and ensemble learning. The following sections briefly describe the models.
All models were trained using grid search for parameter fine-tuning and evaluated with 5-fold cross-validation.

2.1 LSTM

The most straight-forward approach, using RNN, is to use a many-to-one RNN followed by a single densely connected layer for classification. As we're dealing with time series which generally have short- and long-term dependencies to be accounted for, this is intrinsically a task for an LSTM model. We tried several LSTM RNN models and used large gridsearches to find an optimal network layout as well as good hyper parameters. The models however never reached the baseline performance. Therefore we decided to look for more complex architectures. We realized that we are presented with fixed-size samples and can therefore make use of a bidirectional layer which is going to further facilitate extrapolating information.

2.2 CNN-GRU

Adding two convolutional layers on top of the RNN lets the model operate on a higher level of abstraction provided by the CNNs kernels. As the baseline model works very well, we reused its layers for this model, and reduced the perceptive field to guarantee that the RNN is of any use.

2.3 CNN with Residuals

Residual blocks allow us to train deeper networks without worrying about the vanishing gradient. Each residual block does BatchNormalization, two 1D convolutions of kernel size 3, and has a skipping link. We tried different residual blocks, ranging from 16 kernels per convolution up to 64, and combinations of them. The best performance was reached with residual blocks that have 32 filters of size 3. The last two layers are unskippable dense layers to extract the information from the kernels.

2.4 Transfer Learning

While we do have a lot of training samples in the MIT data set, this is not so true for PTB. Looking at section 1, it becomes apparent that the two problems to be solved are very similar and that we should be able to transfer knowledge between the two data sets to improve performance. We tried doing this in two different manners:

- Transferring the weights from a model trained on MIT to the same model for PTB, only re-initialising the last dense layers that are responsible for the output. During training, we allow the weights of the entire model to be updated. We will hereafter be referencing to this method as *Retrain*.
- Transferring all weights from MIT to PTB except for the last dense layers. But this time, we freeze all layers that got transferred from MIT, thus only training the re-initialised dense layers. This method will hereafter be called *Freeze*.

2.5 Ensembling

The idea of classifier ensembles is to *"create a highly accurate prediction rule by combining many relatively weak and inaccurate rules"* [Sch13]. In other words: reduce variance through (more or less intelligent) averaging without drastically increasing the bias. We have presented a number of very accurate models. But by looking at their respective accuracies per class, we realised, that we could build an even stronger model by combining their strengths. We created the following ensemble models:

- AdaBoost: AdaBoost classifier using a random forest as the base classifier
- Averaging: Take the results of all the *weak* learners[1] and average them.
- Neural Layer Ensembling: Concatenate the *weak* learners' results and add a densely connected layer which combines the results.
- LGBM: Concatenate the *weak* learners' results and use the gradient boosting framework LightGBM to create decision trees.

3 Results

In general, it can be noted that although the classes are very unevenly represented, the models didn't tend to overfit to a particular class. We tried penalising errors w.r.t. class sizes, but we never got better results than without this measure. All numbers in tables 1, 2 and 3 are medians from tests conducted 5 times per model. The standard deviation is less than 1% for all models and therefore omitted for the sake of readability. The exact results can be consulted in model_results.xlsx.

[1] Weak learners for MIT: Baseline CNN, LSTM, CNN-GRU, CNN w. Residuals; PTB: Baseline CNN, LSTM, CNN-GRU, CNN w. Residuals and all the models with transferred weights

3.1 Base Models

Table 1 shows the performance of the models introduced in section 2. Adding the convolutional layers on top of the RNN has proven to make the model more robust compared to having none. In fact, CNN-GRU is the best performing model on the MIT dataset. Interestingly, CNN with Residuals, which is vastly inferior to CNN-GRU on MIT, wins the price of the best model on PTB. This is probably due to the large difference in trainable parameters, since CNN-GRU has over 230'000, while CNN w. Residuals has only 82'000. MIT has enough samples to leverage the large number of parameters, but PTB is too small and hence a simpler model is to be preferred over a complex one in this case.

	MIT		PTB			
Model	Accuracy	F1	Accuracy	F1	AUROC	AUPRC
Baseline CNN	0.9847	0.9111	0.9904	0.9880	0.9880	0.9915
LSTM (Bidirectional)	0.9844	0.9111	0.9289	0.9110	0.9090	0.9376
CNN-GRU (Bidirectional)	**0.9885**	**0.9325**	0.9923	0.9904	0.9883	0.9917
CNN w. Residuals	0.9824	0.8973	**0.9948**	**0.9936**	**0.9930**	**0.9949**

Table 1: Comparison of the models' performances

3.2 Transfer Learning

From table 2 we can infer that *Retrain* performs considerably better than blocking learning for the transferred layers. Apparently, having the flexibility of adapting all layers to the fact that the number of classes went from five to two benefits the accuracy. Although *Freeze* converges faster due to the fewer number of trainable parameters, accuracy is usually preferred over training speed in medical applications, which makes *Retrain* the more preferable method. Overall, transferring knowledge from MIT to PTB has increased accuracy by more than a percent, which is a considerable amount given that the accuracy was already very high before.

Model		Accuracy	F1	AUROC	AUPRC
Baseline CNN	Retrain	**0.9966**	**0.9958**	**0.9950**	**0.9963**
	Freeze	0.9813	0.9769	0.9808	0.9872
LSTM (Bidirectional)	Retrain	0.9880	0.9851	0.9864	0.9906
	Freeze	0.9680	0.9604	0.9627	0.9743
CNN-GRU (Bidirectional)	Retrain	**0.9966**	0.9957	**0.9950**	**0.9963**
	Freeze	0.9828	0.9786	0.9790	0.9853
CNN w. Residuals	Retrain	0.9946	0.9933	0.9920	0.9941
	Freeze	0.9888	0.9860	0.9920	0.9941

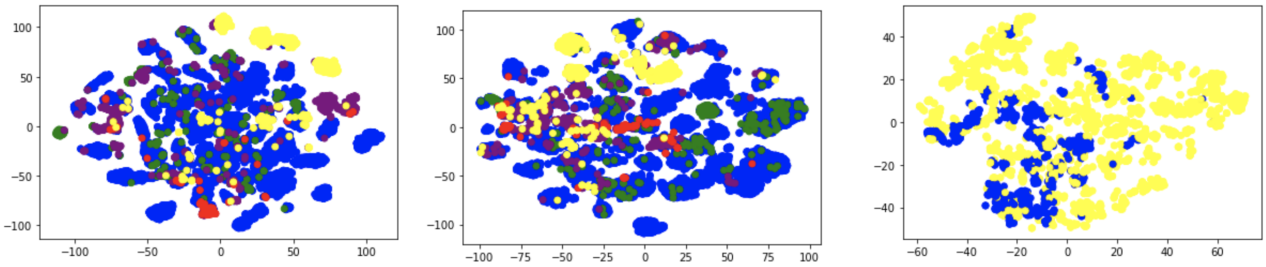
Table 2: Comparison of the models' performances when transferring weights from MIT to PTB

3.3 Ensembling

In ensembling it becomes apparent that it is beneficial to combine the *weak* learners' results using an *intelligent* model. For MIT, all learners except CNN-GRU perform relatively poorly at classifying samples from class 1 and 3 due to them being heavily underrepresented. Averaging bets on the assumption that the majorities' opinion is right, thus not allowing for experts' opinions. In our case, even though CNN-GRU performs reasonably well on the underrepresented classes, it may very well be overruled by the majority of less knowledgeable models. Small neural networks or decision trees (LGBM) do not suffer from the same weakness and show 5% resp. 3.5% higher accuracy for class 1 and 4% resp. 6% higher accuracy for class 3, while the accuracy on other classes is similar across all ensembling methods. This conclusion is further backed by the fact that Averaging is as good as the other ensembling methods on PTB, which has only two classes and therefore provides no possibility for a model to become an expert.
In contrast to these methods, traditional ensembling models such as random forest or AdaBoost cannot match their scores. For the PTB dataset we observe scores slightly lower than the Baseline CNN whereas for the MIT dataset the results do not match the baseline.

	MIT		PTB			
Model	Accuracy	F1	Accuracy	F1	AUROC	AUPRC
AdaBoostClassifier	0.9724	0.8661	0.9780	0.9721	0.9635	0.9740
Averaging	0.9883	0.9301	0.9962	0.9951	0.9950	0.9963
NN Layer	0.9883	0.9305	0.9960	0.9950	0.9948	0.9963
LGBM	**0.9889**	**0.9332**	**0.9966**	**0.9957**	**0.9957**	**0.9970**

Table 3: Comparison of the ensembling models' performances



(a) Bidirectional LSTM (b) CNN w/ Residual (c) Transferred bidirectoinal LSTM

Figure 4: Visualization of the learned representations with TSNE, colored to easier visualize the different classes. (a) Samples from the Bidirectional LSTM classification results (b) Samples from the CNNR classification results (c) Samples from the transferred bidirectional LSTM, retrained on the PTB data set.

4 Conclusion

In this report we presented multiple methods for ECG heartbeat classification. We thoroughly investigated transfer learning [STS18] and ensembling methods, both very interesting concepts in real-world applications due to an increasing number of pretrained, well-performing methods that are already at hand and ready to be adapted and combined to solve new problems. Our obtained results demonstrate that these methods achieve scores close to state of the art methods in the literature, and that they absolutely need to be considered when conceiving machine learning frameworks in the medical domain.

References

[Sch13] Robert Schapire. "Explaining AdaBoost". In: Oct. 2013, pp. 37–52. ISBN: 978-3-642-41135-9. DOI: 10.1007/978-3-642-41136-6_5.

[STS18] Milad Salem, Shayan Taheri, and Jiann Shiun-Yuan. "ECG Arrhythmia Classification Using Transfer Learning from 2-Dimensional Deep CNN Features". In: *arXiv e-prints*, arXiv:1812.04693 (Dec. 2018), arXiv:1812.04693. arXiv: 1812.04693 [cs.LG].