Project 1 Time Series

Anton Schäfer
ETH Zurich
Switzerland
scanton@ethz.ch

Lasse F. Wolff Anthony
ETH Zurich
Switzerland
laanthony@ethz.ch

Nidhi Agrawal
ETH Zurich
Switzerland
agrawaln@ethz.ch

ABSTRACT

Cardiovascular diseases are the prime cause of human deaths. Analysis and processing of ECG signals are crucial for timely diagnosis of these diseases and other cardiac abnormalities. In this project work, we investigate automatic ECG interpretation using the PTB Diagnostic ECG Database and heart arrhythmia classification using the MIT-BIH Arrhythmia Database. We develop several deep-learning based and traditional ML models and evaluate their performance across these tasks. Finally, we show that transfer learning and ensemble methods can further improve performance but may inherit problems such as decreased performance across rare events or classes.

1 INTRODUCTION

An electrocardiogram (ECG) records the heart's rhythm through monitoring its electrical activity that occur during depolarization and repolarization of cardiac muscle during each heartbeat. Cardiac abnormalities or cardiovascular diseases can cause changes to the ECG pattern and as such automatic classification and interpretation of these ECG signals may be essential for a timely diagnosis.

In this work, we exploit deep learning and its capabilities to recognize complex data patterns, extract features automatically, and capture long range dependencies for ECG interpretation using the PTB Diagnostic ECG Database and heart arrhythmia classification using the MIT-BIH Arrhythmia Database. We propose various deep-learning based models and evaluate their performance across these two tasks as well as compare them with traditional ML models. Transfer learning and ensemble methods are further explored to potentially increase performance.

2 MODELS AND METHODS

2.1 Datasets

We evaluate our models on the preprocessed[1] versions of the following two medical time series datasets:

MIT-BIH Arrhythmia Dataset (MIT-BIH) [5, 8]: The original dataset contains 48 half-hour excerpts of two-channel ambulatory ECG recordings, obtained from 47 subjects. The recordings were digitized at 360 samples per second per channel. We have been provided with 109 446 samples (80% train and 20% test samples), 188 features including labels. Labels have 5 classes denoting a normal beat, supraventricular premature beat, premature ventricular contraction, fusion of ventricular and normal beat, and unclassifiable beat. The dataset is highly imbalanced with ~83% corresponding to normal beats.

[1]Cropped, downsampled, and zero-padded.

PTB Diagnostic ECG Database (PTB DB) [3, 5]: The original dataset contains 549 records from 290 subjects. Each record includes 15 simultaneously measured signals from the conventional 12 leads along with the 3 Frank lead ECGs. Each signal was digitized at 1000 samples per second. We have been provided with 14 552 samples (80% train and 20% test samples), 188 features including labels. Labels have 2 classes denoting normal and myocardial infarction.

2.2 Models

We investigate several deep learning architectures as well as a tree-based model and compare them to the provided CNN-based baselines[2]. While we find that for the tree-based model, different architectures work better for the different datasets, we apply the same deep learning architectures on both datasets and vary only the final prediction layer. For the PTB DB dataset with binary targets, we generate a single logit and use the sigmoid activation, while we use softmax for the 5-class MITBIH dataset.

CNN models: We compare our models to the aforementioned CNN baseline with eight convolutional layers (Baseline) and to the deep residual CNN with 5 residual blocks containing two convolutional layers each proposed by Kachuee et al. [6] (Deep ResCNN). We propose a simple CNN model (VanillaCNN) that we found to yield the best validation accuracy during an architecture search. The model consists of four convolutional layers with comparably big kernel sizes (7, 9, 11, 11) followed by three dense layers. We further propose a deeper residual CNN (Deep++ ResCNN) comprised of 10 residual blocks each comprising two convolutional layers. Finally, we try to improve predictions on minority classes by training a VanillaCNN with a weighted loss function.

Attention model: Given the recent success of attention-based methods in various fields, we also introduce an architecture that combines the attention mechanism with convolution (AttCNN). We include convolution layers to enable the model to utilize positional or neighborhood information. This can be crucial for understanding the data without requiring additional positional embeddings. The model uses four multi-head dot-product self-attention layers that are each preceded by a residual block as in Deep++ ResCNN. Further skip connections are used and layer normalization is added before every attention layer.

RNN models: We evaluate two RNN-based models. A simple RNN (VanillaRNN) and GRU model (GRU). For both, we find that four 512-wide layers followed by average pooling over all hidden states and four linear layers perform best on a hold-out validation set. For VanillaRNN we use a simple RNN layer with Tanh activation. We further experiment with versions that instead use

[2]https://github.com/CVxTz/ECG_Heartbeat_Classification/tree/master/code

Machine Learning for Healthcare, 2022, ETH Zurich, Switzerland
March 29, 2022.

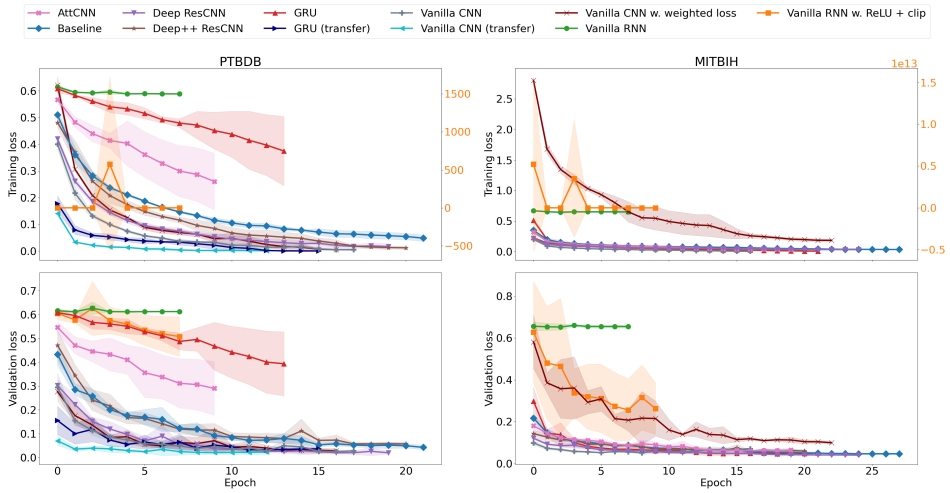

Figure 1: Learning curves for our deep-learning models showing the mean (upper) training and (lower) validation cross-entropy loss by epoch on the (left) PTB DB and (right) MIT-BIH datasets. The shaded areas depict ±1 standard deviation from the mean. The second y-axes (rhs) depicts the training loss for the Vanilla RNN w. ReLU + clip model.

ReLU activation and clipped gradients to a maximum value of 0.5 (VanillaRNN+ReLU+Clip). This helps combat vanishing and exploding gradients we observed when using Tanh activations or unclipped gradients. GRU uses Tanh activationsand clipped gradients resulting in a much more stable learning curve.

Transfer learning: As proposed by Kachuee et al. [6], we apply transfer learning to two of our models, VanillaRNN and GRU, by first training them on the bigger MIT-BIH dataset then freezing the convolutional, or respectively, GRU layers, and lastly only training the final dense layers on the PTB DB dataset.

Tree-based models: Deep-learning based models often require large amounts of compute and benefit from the massive parallel compute offered by GPUs. We explore the trade-off between compute required and model performance by further evaluating an averaging classifier using multiple randomized decision trees fit on sub-samples of the data [4] (Extra Trees).

Ensemble models: We employ two ensemble learning methods: a model averaging ensemble (Model avg. ensemble) and a stacking ensemble (Stacking ensemble). Model avg. ensemble uses an unweighted average of the predicted output probabilities across all trained models for its predictions. Stacking ensemble instead trains a logistic regression classifier on the predictions of all other trained models. Internally, both of these models examine all log files generated by training other models (except ensemble models) to create their predictions.

3 EXPERIMENTS

3.1 Experimental Setup

All of our experiments were performed on the ETH Euler cluster[3] using 1 unknown GPU and 1 unknown CPU with 8 GB of reserved RAM. We repeated every model run five times with consecutive seeds (seed 42 to 46) and report the mean and standard deviation. The tree-based models were developed using Scikit-learn [9]. The deep-learning models were developed using Tensorflow [1] and were trained for a maximum of 1000 epochs using the Adam optimizer [7] and decreasing the learning rate once validation accuracy plateaued for 3 epochs. We further used early stopping and assumed convergence when the validation accuracy showed no improvement for 7 (5 for baselines) consecutive epochs. The final models are the models with the highest validation accuracy.

Hyperparameter tuning was done using a random search [2] over a large parameter space for both model and optimizer configurations. Optimal configurations can be found in the source code. We further experimented with Bayesian optimization to tune some models but this showed no improvement over a vast random search. It had a slightly higher compute efficiency but increased implementation complexity significantly.

For the MIT-BIH dataset, the F1-score is computed as the macro-averaged or unweighted mean of all per-class F1-scores. For the PTB DB dataset, we use AUPRC to denote the average precision score since directly computing the area under the precision-recall curve using the trapezoidal rule is prone to overestimating[4]

3.2 Results and Discussion

3.2.1 Training behavior. The training behavior of all deep-learning models models is shown in Figure 1. As expected, larger models train slightly slower. They also exhibit greater variance in loss and final accuracy depending on the seeding. For the VanillaRNN model with Tanh activation and no gradient clipping, we find that it suffers from the vanishing gradient problem causing stagnating loss. VanillaRNN+ReLU+Clip performs much better, however, we observe divergence of loss, presumably due to large gradients, temporarily for some runs.

[3] https://scicomp.ethz.ch/wiki/Euler

[4] See https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html.

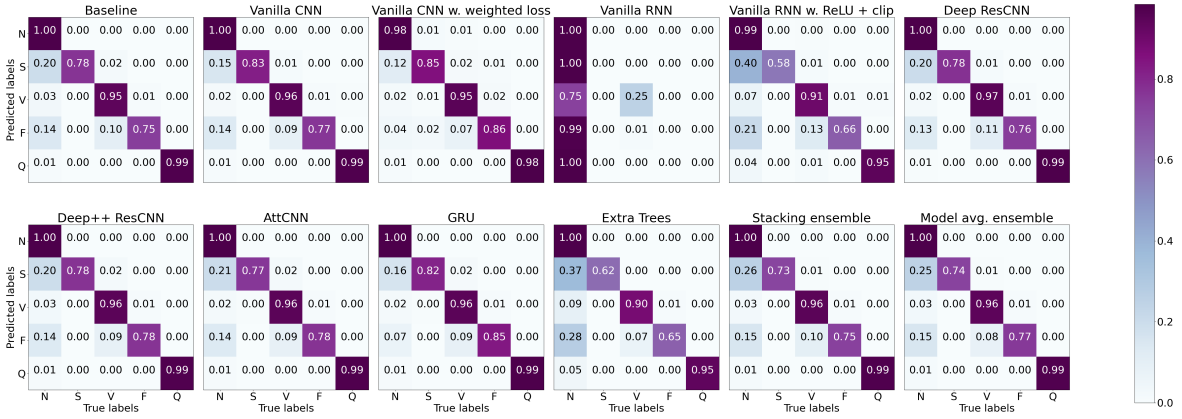

Figure 2: Confusion matrices associated with our model performances on the MIT-BIH test dataset. The class abbreviations are as follows: 'N' is normal beat, 'S' is supraventricular premature beat, 'V' is premature ventricular contraction, 'F' is fusion of ventricular and normal beat, and 'Q' is unclassifiable beat.

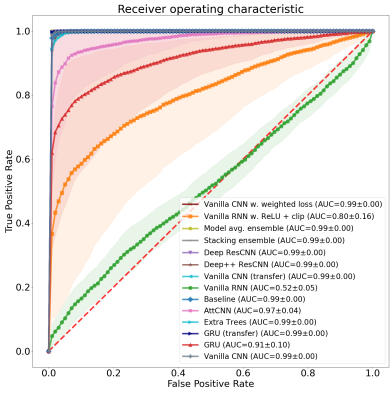

Figure 3: Receiver operating characteristic curve for our models on the PTB DB test dataset. The shaded areas depict ±1 standard deviation from the mean.

3.2.2 Test scores. Table 1 and Table 2 show the performance metrics on the test sets. Generally, we observe good performance close to the baseline for most models. Notably our VanillaCNN shows good performance, outperforming the baseline on both datasets. However, adding model complexity does not yield better results as evident by the performance of Deep++ ResCNN and AttCNN.

In line with the observed stagnating loss, VanillaRNN with Tanh activation and no gradient clipping fails to produce useful outputs for both datasets. As seen in the confusion matrix in Figure 2 as well as the ROC curve in Figure 3, it is clear that it does not perform better than random chance. The RNN-based models, VanillaRNN+ReLU+Clip and GRU, perform much better, especially on the MIT-BIH dataset. On the smaller PTB DB dataset they lack performance. This can likely be explained by the fact that they require more training data due to a large amount of parameters. This may also help explain why GRU benefits greatly from transfer learning, outperforming all other non-ensemble models on PTB DB. The smaller VanillaCNN does not benefit as much from transfer learning, showing only a small improvement.

The tree-based models, Extra Trees, produce predictions that are useful, yet not competitive with the baseline. They, however, train much faster than the deep learning models leading to a trade-off between performance and training time.

As seen in Figure 2, a weighted loss may balance class frequency effects as shown by the improved performance for VanillaCNN in predicting minority classes after being trained with the weighted loss. Overall accuracy and F1 score does decrease as another trade-off. We note that this behavior may be desired in some settings where, e.g., false negatives are more problematic for specific classes such as in medical domains.

Finally, as expected, the ensemble methods building upon all models and their variations perform very well. Both of their accuracies are within around a standard deviation of the best performing model on both datasets, outperforming all other models at times. Still, Figure 2 shows that the ensemble methods inherit the minority class inaccuracies of the models that they build upon.

4 CONCLUSION

In this work, we investigated the performance of several deep-learning based models across the tasks of ECG interpretation and heart arrhythmia classification. Having benchmarked all of our models, we showed that a simple CNN model is able to achieve good performance on both tasks and added model complexity may not yield better results. Further, introducing class weights on the loss can lead to better results for minority classes trading off some overall performance. Finally, we showed that model performance can be improved using transfer learning or ensemble methods across diverse models but these may inherit problems from the sub-learners such as decreased performance across rare events or classes.

[5]Note that the ensemble models require all other models to be trained beforehand as they are trained on their prediction outputs. The training times are therefore not directly comparative to other models.

Model	Training time (s)	Accuracy	F1	AUROC	AUPRC
Stacking ensemble	583±10	**0.9969**	**0.9961**	**0.9989**	**0.9995**
GRU (transfer)	321±30	0.9954±0.0011	0.9943±0.0014	0.9988±0.0002	0.9993±0.0002
Deep ResCNN	52±11	0.9951±0.0011	0.9938±0.0013	0.9971±0.0002	0.9973±0.0003
Model avg. ensemble		0.9945	0.9931	0.9981	0.9988
VanillaCNN (transfer)	35±9	0.9945±0.0011	0.9931±0.0014	0.9975±0.0006	0.9982±0.0005
VanillaCNN w. weighted loss	41±9	0.9937±0.0014	0.9921±0.0017	0.9966±0.0004	0.9973±0.0004
VanillaCNN	39±7	0.9926±0.0016	0.9907±0.0020	0.9967±0.0007	0.9975±0.0006
Baseline	49±9	0.9893±0.0042	0.9866±0.0053	0.9981±0.0009	0.9991±0.0005
Deep++ ResCNN	72±7	0.9842±0.0048	0.9802±0.0060	0.9966±0.0015	0.9984±0.0009
Extra Trees	**12**	0.9799±0.0005	0.9747±0.0006	0.9963±0.0001	0.9982±0.0001
AttCNN	95±30	0.9504±0.0604	0.9310±0.0893	0.9734±0.0429	0.9881±0.0188
GRU	328±111	0.8850±0.1055	0.8440±0.1468	0.9135±0.0971	0.9636±0.0409
VanillaRNN+ReLU+Clip	356±279	0.8201±0.1200	0.6350±0.2642	0.8032±0.1582	0.9096±0.0742
VanillaRNN	134±18	0.7221	0.4193	0.5166±0.0476	0.7559±0.0374

Table 1: Model performances on the PTB DB test dataset.[5]

Model	Training time (s)	Accuracy	F1
Model avg. ensemble		**0.9875**	0.9247
VanillaCNN	181±31	0.9874±0.0008	0.9248±0.0058
GRU	1775±239	0.9872±0.0007	**0.9251±0.0049**
Stacking ensemble	3818±62	0.9870	0.9211
Deep ResCNN	321±45	0.9864±0.0006	0.9199±0.0026
Baseline	290±33	0.9850±0.0009	0.9143±0.0040
Deep++ ResCNN	442±65	0.9849±0.0011	0.9147±0.0066
AttCNN	577±138	0.9836±0.0012	0.9076±0.0084
Extra Trees	**67±7**	0.9764±0.0002	0.8792±0.0022
VanillaCNN w. weighted loss	260±39	0.9681±0.0027	0.8554±0.0074
VanillaRNN+ReLU+Clip	2994±1911	0.9543±0.0224	0.7443±0.1483
VanillaRNN	886±84	0.8306±0.0060	0.1972±0.0322

Table 2: Model performances on the MITBIH test dataset.[6]

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).

[3] R Bousseljot, D Kreiseler, and A Schnabel. 1995. Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet. (1995).

[4] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine learning* 63, 1 (2006), 3–42.

[5] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *circulation* 101, 23 (2000), e215–e220.

[6] Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh. 2018. Ecg heartbeat classification: A deep transferable representation. In *2018 IEEE international conference on healthcare informatics (ICHI)*. IEEE, 443–444.

[7] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[8] George B Moody and Roger G Mark. 2001. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine* 20, 3 (2001), 45–50.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[6] See footnote 1.

A ADDITIONAL FIGURES

Figure 4 and Figure 5 show the confusion matrices and precision-recall curve for the PTB DB dataset, respectively.

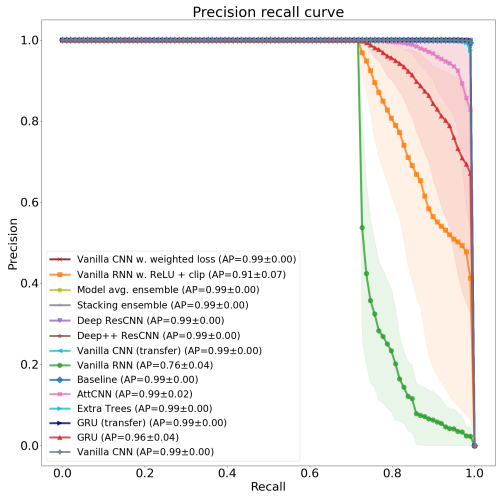

Figure 5: Precision-recall curve for our models on the PTB DB test dataset. The shaded areas depict ±1 standard deviation from the mean.

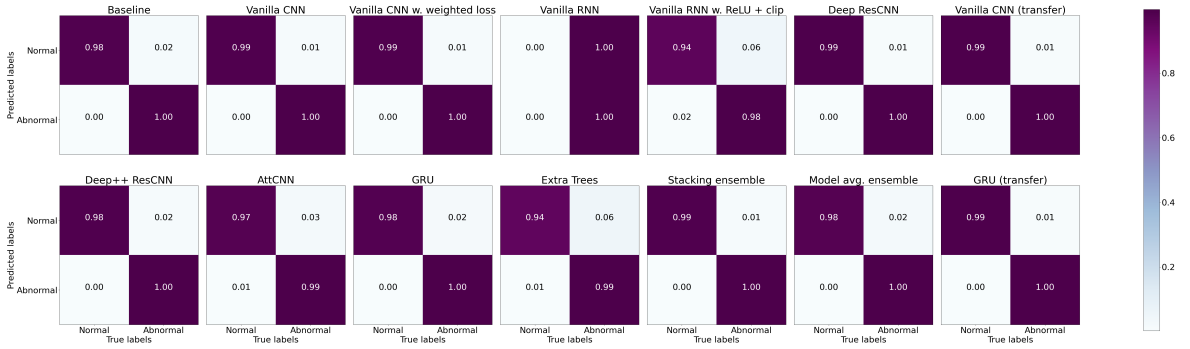

Figure 4: Confusion matrices associated with our model performances on the PTB DB test dataset. The abnormal class means myocardial infarction.