CARDIAC ABNORMALITY CLASSIFICATION

MACHINE LEARNING FOR HEALTH CARE FS2022 MARCH 29, 2022

Frithiof Ekström **Kacper Kapuśniak** **Bartosz Rudnikowicz**

1 Introduction

The electrocardiogram (ECG) is a non-invasive, inexpensive diagnostic tool used for cardiac abnormalities prediction. Such abnormalities include an arrhythmia which is an irregularity of the heartbeat. Although usually harmless, undiagnosed arrhythmia can lead to severe or even lethal complications. Thus, it is crucial to detect it as early as possible using ECGs. However, manual analysis is often time-consuming, and there might be insufficient human resources for such analysis in underdeveloped countries. Thus, automatic detection can speed up the process and widen the accessibility to healthcare. Further, it can be used as the 'second' expert to support doctors in decision-making. Machine learning is a perfect candidate for such automatic detection. Especially recent development in Deep Learning can lead to a performance close to human doctors. That is why in this report, we focus on the analysis of the performance of vanilla and the design of more sophisticated and creative neural networks models for ECG datasets. [1, 2, 3]

We started by implementing vanilla models - Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN). Further, we dived into more state-of-the-art models, namely residual neural networks, transformer encoders, and autoencoders. We also experimented with different transfer learning approaches for each of the models. Finally, we developed two different ensemble methods combining our cutting edge methods - the first that stacks probabilities predictions', and the second that utilise the latent space produced by encoders and convolution layers.

We trained and tested our models on two datasets - Arrhythmia Dataset (MIT-BIH) with around 100k samples and five classes (regular, three abnormal, and unclassified), and The PTB Diagnostic ECG Database with around 15k samples and only two classes (regular and abnormal beat).

2 Methods

2.1 Recurrent Neural Network

We started with the implementation of an uncomplicated RNN. Since RNNs are famous for modelling time dependency in data, they seem like a reasonable choice for ECG signals, which are time-series.

Our model consists of one recurrent layer with 512 features in the hidden state. It takes one input value from our data at each time step. Output from the last timestep is passed to the fully connected layer. It consists of five output neurons for the multi-class classification (MIT-BIH dataset) and one output neuron for the binary classification (PTBDB dataset). For the former case, the softmax function is applied to the last layer, and it is trained with cross-entropy loss as a loss function, while for the latter case, we applied sigmoid to the output and used binary cross-entropy loss.

2.2 Convolutional Neural Network

Another influential architecture for classifying time series are Convolutional Neural Networks. Similarly, we began with a vanilla CNN, consisting of three one-dimensional convolutional layers. All of them use a kernel size of 10. The first two layers have 20 channels each, with the last layer having 40 channels. Output from the last convolutional layer is passed to the fully connected layer, with the same output as the previously described RNN model. For the training, we used the same loss functions as in the Vanilla RNN.

2.3 Residual Neural Network

Inspired by the ResNet [4] architecture used for computer vision tasks, we decided to implement a CNN with residual connections for one-dimensional input. Therefore, we followed the original ResNet paper and adapted it to the input dimensions of our ECG data.

The first layer is the one-dimensional convolutional layer, with a kernel size of 12 and 10 channels. Following the layer, there are three blocks, consisting of two convolutional layers, batch normalization and the ReLU activation function. As it is a residual network, the output of these layers is added to the original input to the block. If input and output have different numbers of channels, the original input goes through a convolutional layer with a kernel size equal to one to achieve the same number of channels as the output of the block.

The three blocks in our ResNet implementation have 20, 20 and 40 channels, respectively. The kernel size is the same for all of them and set to nine. The output of the last block is passed to the fully connected layer, which has five output neurons for the MIT-BIH dataset and one output neuron for the PTBDB dataset. The model uses the same loss functions as the vanilla RNN and CNN models.

2.3.1 Transfer learning

Further, we performed transfer learning for our ResNet model. Firstly, the model was trained on the MIT-BIH dataset. Next, the fully connected layer was swapped, and the weights in convolutional layers were kept for fine-tuning the model. Finally, we retrained the whole model on the PTBDB dataset without any frozen layers.

2.4 Attention

One of the most prominent approaches in deep learning are Transformers. Initially, they were designed for a sequence-to-sequence model for machine translation, but they achieved state-of-the-art performance in various tasks, including but not limited to Natural Language and Audio Processing [5]. The vital part of the transformer encoder is a multi-head attention mechanism that learns to

distinguish between more and less important parts of the input to provide efficient embedding [6].

Thus, our next approach involves using the attention layers to form transformer encoders. The output of that encoders is passed to the Dense and Softmax layers. Our implementation uses two sequential blocks of encoders, each consisting of a Multihead Attention layer. The layer output is added to its input to form a residual connection. Then, it is passed to two one-dimensional convolutional layers and the dropout. It is followed by the second residual connection, this time from the input of the convolutional layers to their output. It roughly resembles the design from [6]. However, we decided not to use normalisation layers. After testing the Batch Normalisation and the Layer Normalisation, it turned out that both of them hindered the performance in contrast to no normalisation at all. The detailed model schematics can be found in A.

2.4.1 Transfer Learning

Next, we explored transfer learning methods for the 'Attention' model involving pretraining on the MIT-BIH dataset and fine-tuning on the PTBDB dataset. This time, we decided to compare two different approaches to transfer learning. For the first one, we froze all layers in encoders after pretraining and only tuned the dense layers. Our second method, analogical to ResNet transfer learning, did not involve freezing layers but tuning the entire model. For both approaches, we changed the dimension of the last layers to match the number of classes in the new dataset.

2.5 Autoencoder

Autoencoder-based methods for feature extraction in ECG signals have been explored in the past with promising results [7, 8]. The idea of these methods is to project the heartbeat signal onto a latent space, in which the features of the signal can be more meaningfully represented than in the data space. The latent representations can then be used as input to a classifier.

Our model consists of a convolutional encoder transforming the input from a signal of length 188 to a latent representation of size 47 x 2. The decoder consists of transpose convolution layers transforming the latent representation back to the original shape. In training, the encoder and decoder are chained, and the loss is calculated by computing the mean squared error using the decoder output and the original input.

The classification is done using the scikit-learn implementation of extremely randomized trees [9]. It takes as labels the flattened latent representation produced by the encoder, and fits an ensemble of decision trees using random feature selection. The detailed model schematics can be found in A.

2.6 Transfer Learning

The autoencoder training step is self-supervised and does not rely on labels. Thus, for the transfer learning part of the experiment, the autoencoder was simply trained on the two datasets combined. Then, extremely randomized trees were applied as before.

2.7 Ensembles

It has been shown that combining methods may yield better results than simply selecting the best one [10]. Therefore, we decided to implement the ensemble method by stacking our models. More precisely, we are using the Vanilla CNN, Residual Neural Network, Attention and Autoencoder models. We used two different approaches to stack these models.

In the first approach, we use the class probabilities as returned by the softmax layer of the neural network models. These probabilities are treated as features for the training of the ensemble classifier. To select the most suitable classifier we compared various manually hyper tuned models, including Random Forests, LGBM [11] and XGBoost [12], where ExtraTrees [9] achieved the best performance.

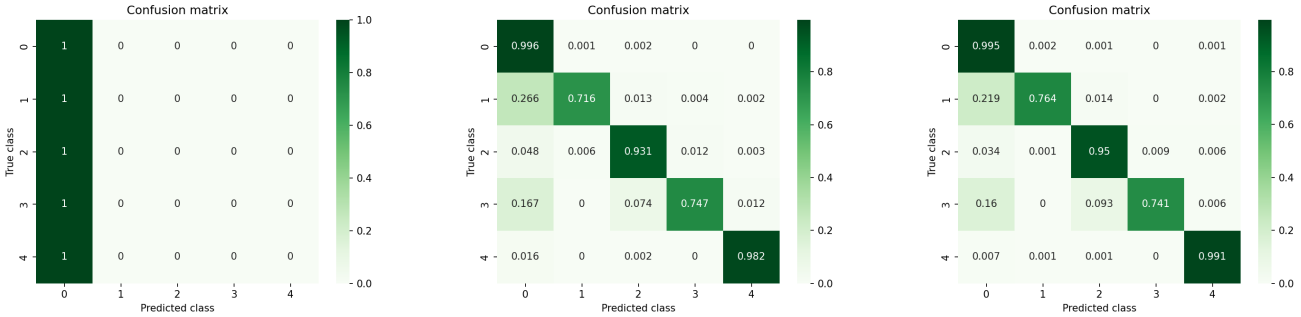
Our second method involves combining 'hidden' features generated by each model. For the CNN and ResNet models, they are the outputs of the last convolutional layer. For the Attention and Autoencoder models, they are the output of the corresponding encoders. Again, we tested various classifiers and selected the XGBoost [12] as the final one. However, the number of features is significantly larger than for the'probabilities' method, and due to memory constraints, we decided to perform memory-efficient

Table 1: Performance metrics for models trained without transfer learning.

Model	MIT-BIH		PTBDB			
	Accuracy	F1 (macro)	Accuracy	F1 (binary)	AUROC	AUPRC
Vanilla RNN	0.828	0.181	0.722	0.839	0.500	0.611
Vanilla CNN	0.980	0.885	0.965	0.976	0.950	0.996
ResNet	0.983	0.907	0.980	0.986	0.965	**0.999**
Attention	0.979	0.887	0.961	0.973	0.956	0.994
Autoencoder + ExtraTrees	0.975	0.871	0.974	0.982	0.961	0.997
Ensemble (combine hidden features)	**0.986**	**0.924**	0.992	0.994	0.990	**0.999**
Ensemble (combine probabilities)	0.985	0.913	0.990	0.993	0.987	**0.999**
Baseline	0.983	0.903	**0.993**	**0.995**	**0.991**	**0.999**

Table 2: Performance metrics for transfer learning models on the PTBDB dataset.

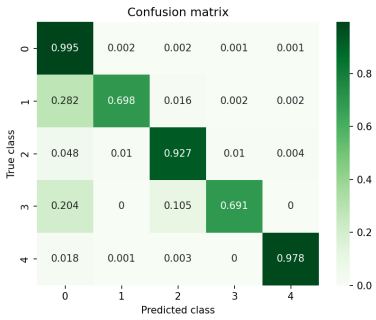
	Accuracy	AUROC	AUPRC	F1-score
ResNet	**0.993**	**0.99**	**0.998**	**0.993**
Attention - freezed layers	0.964	0.956	0.996	0.975
Attention - finetuning	0.973	0.967	0.996	0.981
Autoencoder	0.975	0.962	0.997	0.983



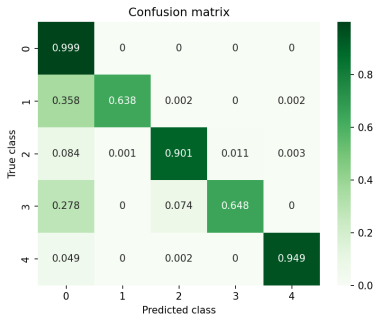
(a) Vanilla RNN

(b) Vanilla CNN

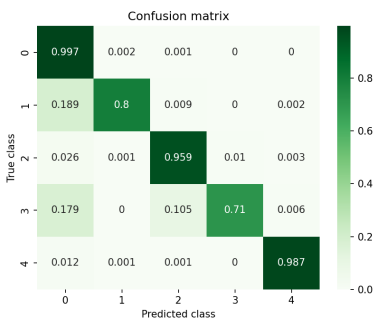
(c) ResNet



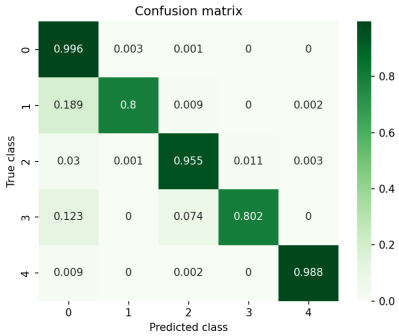
(d) Attention



(e) Autoencoder



(f) Ensemble Softmax



(g) Ensemble Hidden

Figure 1: Normalised Confusion Matrices for MIT-BIH dataset

feature selection prior to the ensemble classifier training (only for the MIT-BIH dataset). Firstly, we deleted all the features with zero variance. Then, we selected the 200 best features by computing the ANOVA F-value [13] on the training set.

3 Results and Analysis

The following sections present the results and analysis for each model. They were trained with 'Adam' optimiser [14], and using early stopping on the validation set (10% of the training set). The summary of results can be found in Table 1 for models trained without Transfer Learning and in Table 2 for results of models trained with Transfer Learning. Further, the confusion matrices can be found in Table 1 for the MIT-BIH dataset and in B for the PTBDB dataset.

3.1 Recurrent Neural Network

The RNN model did not learn any significant patterns in data. The training was very unstable - while training loss was 'jumping' up and down, the validation loss barely decreased by 0.001 in the first epochs and saturated in the very next epoch. We tested 256, 512 and 1024 features in hidden states, but they all behaved the same. Then, we tried gradient clipping to avoid the exploding gradient problem, but it did not improve the performance. The RNN model ended up predicting only one class for all examples, as shown in Figure 1b.

3.2 Convolutional Neural Network

On the other hand, the vanilla CNN model yielded good results for both datasets. The model outperformed the attention and autoencoder models on the MIT-BIH dataset in terms of accuracy, achieving an accuracy of 0.980, compared to 0.986 for the best performing model. The gap was more significant for the PTBDB dataset, where the CNN achieved an accuracy of 0.965, and the best model 0.993.

3.3 Residual Neural Network

Our residual neural network model beats the Attention, Autoencoder and both vanilla models on the MIT-BIH dataset, both in

terms of accuracy (where only the baseline model achieves the same performance) and F1 score. On the PTBDB dataset, our ResNet model also performs well, being outperformed only by the Ensemble and Baseline models in terms of accuracy and F1 score. In the AUROC metric, it was also outperformed by the Autoencoder model. However, it achieves the best AUPRC score together with two other models.

3.3.1 Transfer learning

Applying transfer learning further improved the performance of the model. It achieved the best accuracy (together with the baseline model) and improved every other tested metric compared to ResNet trained from scratch. This shows how useful transfer learning is and explains its popularity.

3.4 Attention

The Attention model performance is significantly better on the larger MIT-BIH dataset than on the smaller PTBDB dataset. It might result from the complexity of the Attention model being significantly higher than other non-ensemble models. Thus, it requires more samples than other models. It can also explain why the performance on the MIT-BIH dataset, while being satisfactory, is only better than the Vanilla RNN and the Autoencoder models. The further research direction might include training that model on larger datasets. Furthermore, from Figure 1d we observe that the model especially has problems with distinguishing between classes 0 and 1 (normal beats and supraventricular premature beats) and classes 0 and 3 (normal beats and fusion of ventricular and normal beats).

3.4.1 Transfer Learning

Our Transfer Learning experiments for the attention model showed that both methods (frozen and unfrozen layers) improved upon the method without pretraining on the largest dataset. It reconfirms the phenomenon suggested before - the complexity of attention layers require significantly more samples than we used, as with an increase in the number of samples, a performance increased could be observed. However, the performance is still not competitive enough to rank among the best models.

Moreover, from Table 2 we can observe that freezing layers achieved worse performance than retraining the entire model. It suggests that while the datasets are similar, their latent representation still needs to be tuned for each respective classification task.

3.5 Autoencoder

The features generated by the autoencoder have similar predictive performance when compared to using the same classifier with the original data (identical accuracy and F1 score for the MIT-BIH dataset and a discrepancy of <0.002 in both metrics for the PTBDB dataset), despite representing the signal in half as many dimensions. This indicates that the signal can be significantly compressed with negligible information loss. However, the encoding does not seem to improve predictions.

Figure 2 shows the latent representations computed by the autoencoder projected onto a two-dimensional plane by the UMAP algorithm [15]. Some clusters, like the bottom parts of the clusters of classes 0 and 4, are coherent and well separated from the rest. This information can clearly be used to classify new data points in these regions with high accuracy. However, such regions are few, and a high degree of entanglement can be observed in the remainder of the plot. This calls for a highly complex classifier to be used in conjunction with the autoencoder in order to achieve good classification performance.

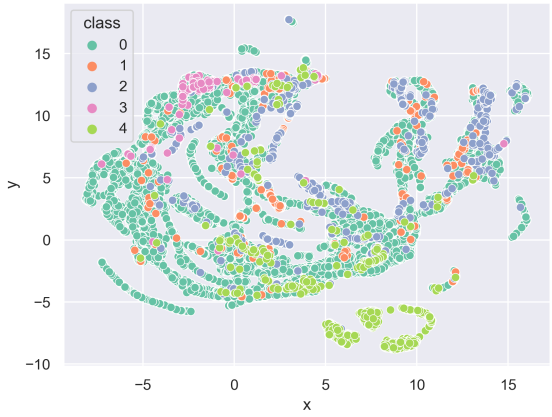


Figure 2: UMAP projection of the latent representations of the autoencoder.

3.6 Ensemble

In Table 1, it is evident that the ensemble method using hidden/latent features consistently outperforms the ensemble method using probabilities. However, on the MIT-BIH dataset, both methods still yield results superior to all the other models, including the baseline. From Figure 1, we can see that both ensemble methods are better in distinguishing between classes 0 and 1 (normal beats and supraventricular premature beats), while the ensemble using hidden features is particularly good at distinguishing between classes 0 and 3 (normal beats and fusion of ventricular and normal beats).

On the other hand, for the smaller PTBDB dataset, the ensemble methods are inferior to the ResNet trained using transfer learning, which achieves the best scores in this category. This is likely a result of the absence of transfer learning base models in our ensembles, since these models, in general, achieve substantially better performance with transfer learning than without. Thus, further research could include building a 'hidden' ensemble of transfer learning models.

4 Conclusion

In summary, our best models exceeded or matched the baseline performance on both datasets. For the MIT-BIH dataset, the best performing model was the Ensemble combining latent representations of the samples created by hidden layers of our models. It achieved an accuracy of 0.986, where the baseline achieved 0.983. Further, on PTBDB dataset, our one-dimensional ResNet implementation with transfer learning achieved the same accuracy as the baseline. We also showed that a vanilla RNN is not a suitable solution for the task. The experiments in further research might include testing the attention network with larger datasets and using pre-trained models in the Ensembles. Such models are pushing forward the area of cardiac abnormality classification. When applied in a clinical environment, they might support medical doctors in decision-making and improve the detection of harmful conditions among patients.

References

[1] P. Melillo, R. Castaldo, *et al.*, "Wearable technology and ecg processing for fall risk assessment, prevention and detection," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2015, pp. 7740–7743.

[2] G. Sannino and G. De Pietro, "A deep learning approach for ecg-based heartbeat classification for arrhythmia detection," *Future Generation Computer Systems*, vol. 86, pp. 446–455, 2018.

[3] A. Isin and S. Ozdalili, "Cardiac arrhythmia detection using deep learning," *Procedia computer science*, vol. 120, pp. 268–275, 2017.

[4] K. He, X. Zhang, *et al.*, *Deep residual learning for image recognition*, 2015. DOI: 10.48550/ARXIV.1512.03385. [Online]. Available: https://arxiv.org/abs/1512.03385.

[5] T. Lin, Y. Wang, *et al.*, "A survey of transformers," *arXiv preprint arXiv:2106.04554*, 2021.

[6] A. Vaswani, N. Shazeer, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[7] P. Liu, X. Sun, *et al.*, "Arrhythmia classification of lstm autoencoder based on time series anomaly detection," *Biomedical Signal Processing and Control*, vol. 71, p. 103 228, 2022, ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2021.103228. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1746809421008259.

[8] C. Jiang, S. Song, and M. Q.-H. Meng, "Heartbeat classification system based on modified stacked denoising autoencoders and neural networks," in *2017 IEEE International Conference on Information and Automation (ICIA)*, 2017, pp. 511–516. DOI: 10.1109/ICInfA.2017.8078961.

[9] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006, ISSN: 0885-6125. DOI: 10.1007/s10994-006-6226-1. [Online]. Available: https://doi.org/10.1007/s10994-006-6226-1.

[10] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Machine learning*, vol. 54, no. 3, pp. 255–273, 2004.

[11] G. Ke, Q. Meng, *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[12] T. Chen, T. He, *et al.*, "Xgboost: Extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[13] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data classification: Algorithms and applications*, p. 37, 2014.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

Appendix A Keras Models' schematics

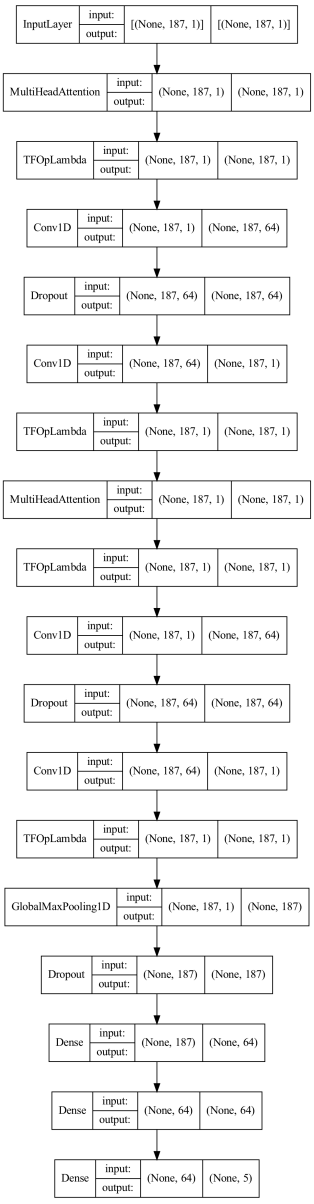


Figure 3: Attention model

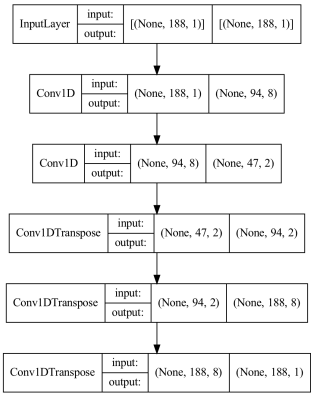
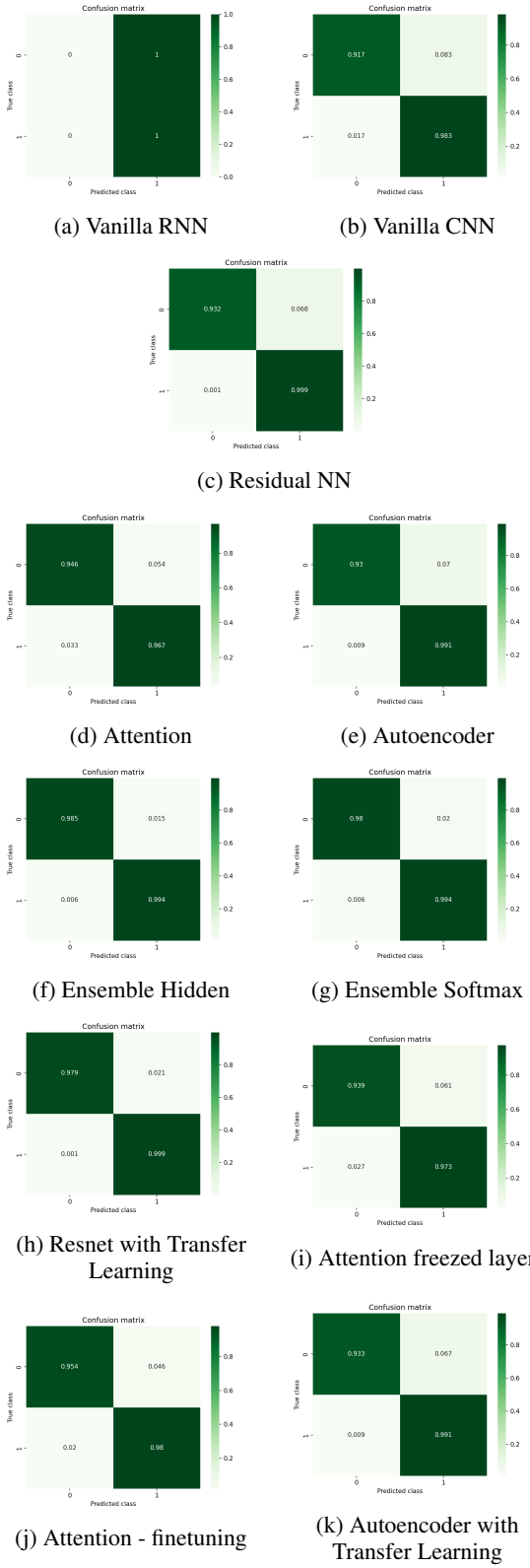


Figure 4: Autoencoder model

Appendix B Confusion matrices - PTBDB dataset



(a) Vanilla RNN

(b) Vanilla CNN

(c) Residual NN

(d) Attention

(e) Autoencoder

(f) Ensemble Hidden

(g) Ensemble Softmax

(h) Resnet with Transfer Learning

(i) Attention freezed layers

(j) Attention - finetuning

(k) Autoencoder with Transfer Learning

Figure 5: Normalised Confusion Matrices for PTBDB dataset