Implementation of a Deep Learning Network for Arrhythmia Classification

Spencer Gritton and Wesley Thio

University of Michigan

Abstract

Electrocardiograms (ECGs) record the voltage across time of electrical activity in a patient's heart, which assists doctors in diagnosing medical conditions and providing proper treatment. One of the most important applications of the ECG is in detecting irregular heartbeats called *arrhythmia*, as they can lead to the onset of a stroke, heart failure, or other life-threatening outcomes. Thus, deep learning methods are being explored as a way to rapidly and accurately diagnose heart conditions from ECG data in order to assist medical professionals in saving lives. In this work, we explore a recent deep learning ECG classification model and validate its accuracy on a novel dataset.

1 Introduction

In this paper, we will study and implement a deep learning network to classify heartbeats in an ECG dataset as: normal, arrhythmic (separated by type), or unknown [6]. The proposed network, called a *hybrid attention-based deep learning network* (HADLN), is unusual in that it utilizes several distinct neural network architectures: bidirectional long short-term memory (BLSTM), residual convolutions (ResNet), and attention. These architectures were combined into one network, tested on the PhysioNet 2017 challenge dataset, and measured using a variety of popular machine learning metrics. The overall architecture is claimed to have significant improvement over traditional machine learning methods typically used for ECG classification. In the following sections, we discuss our method to reproduce the proposed network and apply it to a new ECG dataset in order to validate its effectiveness.

2 Motivation

As the demand for high quality healthcare services increases, many countries are finding that they lack quality healthcare practitioners such as physicians [3]. This problem is amplified by the fact that most countries require nine to fifteen years of additional post-secondary education to become a licensed physician — a heavy burden both in terms of opportunity cost and commitment, even for the most dedicated students. Furthermore, the quality and cost of healthcare can vary greatly by country, geography, class, and a variety of other factors. For these reasons, we propose that developing suitable, cost effective, and accurate substitutes for physicians is in the best interest of the general population. A clear choice to fill this void in care is of course machine learning powered diagnosis and treatment planning in combination with cost effective self administered testing equipment. Our hope is that by continuing to research AI powered healthcare solutions, like the ECG classification implemented here, we can help to further propel the implementation and widespread adoption of low cost healthcare solutions for those who are in need.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

3 Problem Statement

The problem this paper will solve is the implementation and assessment of the HADLN network, in comparison to other networks, in accurately predicting the classification of an ECG as: normal, unknown, or as three prevalent types of heart arrhythmia. We will assess our HADLN network implementation on the MIT-BIH Arrythmia dataset. This dataset contains 48 half-hour excerpts of two-channel ambulatory ECG recordings, obtained from 47 subjects studied by the BIH Arrhythmia Laboratory between 1975 and 1979 [9]. These samples were later digitized and provided freely online and have since become a key dataset in ECG classification tasks.

4 Related Work

4.1 HADLN: Hybrid attention-based deep learning network for automated arrhythmia classification [6]

This paper describes the architecture, training, results, and intuition behind the HADLN network and is the work we plan to re-implement. This work describes previous state-of-the-art approaches on the PhysioNet ECG classification challenge which inform the ensemble of approaches utilized in HADLN. Additionally, this work explains both the mathematics and intuition behind residual convolutions, long short-term memory, and attention models.

4.2 Neural machine translation by jointly learning to align and translate [2]

A 2014 paper by Bahdanau et al. introducing the application of self-attention to language translation tasks. Although the application of this work is not directly related to ECG classification, the ideas presented within are pivotal both in terms of understanding and applying attention to deep learning models.

4.3 The application of medical artificial intelligence technology in rural areas of developing countries [3]

This paper examines the ability of artificial intelligence models to improve the efficiency and quality of work by physicians. Additionally, this work examines and finds that giving non-physician healthcare workers access to these tools would greatly improve outcomes, especially in rural areas of developing countries. Guo et al. end this paper by concluding it would be in the best interest of the general populations of developing countries to create comprehensive multi-level AI based medical networks.

4.4 Bidirectional Recurrent Neural Network and Convolutional Neural Network (BiRCNN) for ECG Beat Classification [11]

In this work, Xie et al. describe a new state-of-the-art model on the task of classifying the MIT-BIH ECG dataset. This work allowed us to gain insight into the most effective current classification models and is highly relevant both in terms of comparison and education.

5 Methodologies / System Architecture

Three separate neural networks must be implemented for the HADLN model. The first is a bidirectional long short-term memory network (BLSTM). These network architectures remember sequences of data and identify patterns in those sequences to output predictions. BLSTM's are utilized in speech recognition and natural language processing where the sequence of data is critical. In this implementation, the BLSTM runs in parallel to a residual convolutional network (ResNet) as both concurrently process the input. These networks, which are covered in more detail below, are unique in that they integrate features from earlier convolutions into later ones, helping to better extract relevant features and avoid vanishing gradients in deep networks [6]. The outputs of the BLSTM and ResNet networks are finally concatenated and enter an attention network which assists the model in only considering the relevant information from both earlier branches of the network. The full HADLN network is shown in Fig. 1.

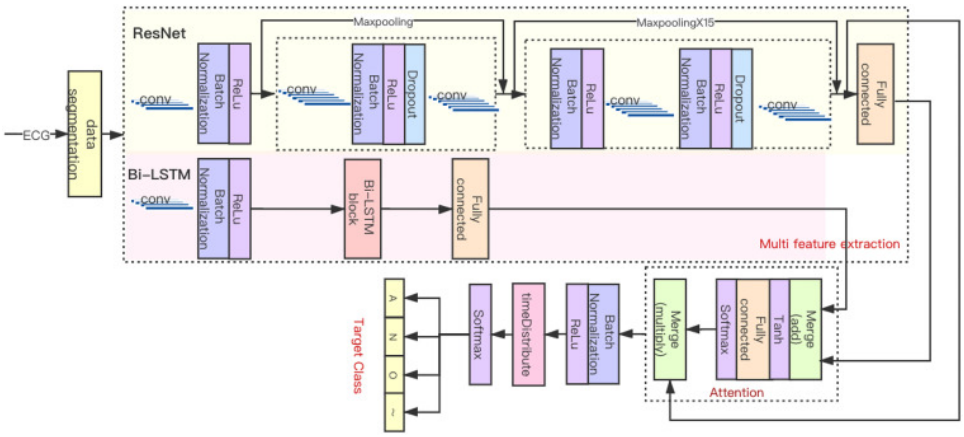

Figure 1: HADLN neural network architecture [6].

5.1 Recurrent Neural Networks

Bidirectional long short-term memory (BLSTM) networks are a subset of long short-term memory (LSTM) networks which are themselves a subset of recurrent neural networks (RNN). RNNs specialize in working with sequences of data and do this by creating long chains of recurrent cells and linking those chains together by a hidden state. A recurrent cell is a simple block that takes in an input and a hidden state, passes those through a fully connected layer and an activation function, then outputs both a hidden state and a cell output. Thus, RNN cells can be linked together by wiring their hidden states together and bootstrapping a random initial hidden state to the first cell. These cells are typically created in a one to one correspondence with the input, so if the input sequence is 187 units, the RNN would have 187 recurrent cells and the output would be the output of all the recurrent cells. Unfortunately, RNNs suffer from a problem with vanishing gradients [7]. Vanishing gradients occur when the gradient update to a parameter in the network is extremely small, causing parameters earlier in the model to receive even smaller gradient updates which halts learning. Because RNNs are long connected chains, small gradients early in the back propagation process lead to extremely slow learning. Furthermore, because RNN cells are only connected to their nearest neighbors, they often have trouble getting information from and passing gradient updates to RNN cells that are far away.

5.2 Long Short-Term Memory Networks

As explained above, LSTMs are a subset of RNNs which focus on giving each cell a "memory" as well as solving the vanishing gradient problem [5]. Recall that each RNN has a small linear layer inside the network that makes predictions and updates the hidden state for the next RNN cell. LSTMs follow the same format but have four linear layers inside each cell: a prediction network, forgetting network, selection network, and an ignoring network. Additionally, LSTM cells have their own "cell state" which acts as a memory unit across time and which a modified version of is passed to each cell. The prediction network inside the LSTM cell works in the same manner as the RNN cell prediction network, taking an input and a hidden state and outputting both an output and hidden state. This output state is then multiplied pointwise with the output of the ignoring network, added pointwise to the cell state, and subsequently multiplied pointwise by the output of the selection network, finally forming the complete output of the cell [5]. While this may seem like many steps, each has a specific job inside the LSTM cell. The ignoring network decides which input information is or is not relevant. The forgetting network determines what the cell state should remember between time steps, which is crucial with a limited cell memory. Finally, the selection network learns to select the best outputs to the network when fed in the pointwise multiplication of the prediction network and cell state memory. Thus, each LSTM cell takes in a hidden state, cell state, and input and outputs a new hidden state, cell state, and output. Notably however, the HADLN network uses a *bidirectional* LSTM. This is a simple modification to the LSTM concept which creates two LSTM networks where one processes the input sequence in normal order and the second processes the sequence in reverse order. These two LSTM networks, both forwards and backwards, then have their outputs concatenated to give the

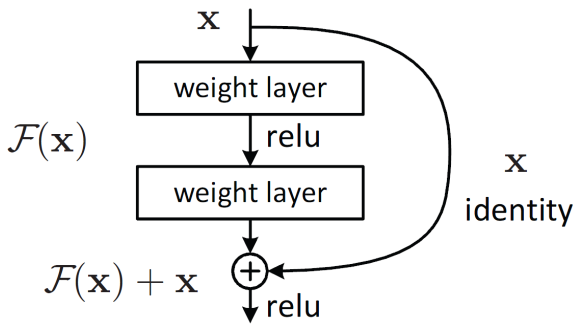

Figure 2: Residual convolution where weight layers are convolutions and the identity x is a skip connection incorporating information from earlier in the network to a later part [4].

network a better understanding of the data. This bidirectional processing has been shown to be more effective than standard LSTM models [10].

5.3 Residual Convolutional Networks

Residual convolutional networks are a method of dealing with the vanishing gradient problem that often occurs in deep convolutional networks. As explained in the original ResNet paper, one would expect network performance to improve as the number of convolutional layers increases, but because gradients tend to diminish between layers, normal convolutional network performance actually decreases after a certain depth [4]. ResNets propose solving this issue by introducing *skip connections* to the network. Shown succinctly in Fig. 2, residual convolutions form blocks where each block takes an input, *X*, applies two to three convolutions and non-linearities to X and finally adds the new and original X together to form the output of the block. This architecture has one key benefit over a more standard chain of convolutions — layers early in the chain are connected to layers late in the chain with less intermediaries between them, allowing larger gradients to propagate and giving later layers more information from earlier in the network.

5.4 Self-Attention

Self-attention, first proposed by Bahdanau et al. in 2014, provides a mechanism to take an input sequence and re-weigh each element within that sequence such that the resulting output of the re-weighing process is another sequence where each element of the sequence is contextualized in terms of the other elements in the sequence [2]. Following this re-contextualization, subsequent layers of the network are able to learn more effectively. Mathematically, these operations are explainable by the following (or shown in Fig. 3): start with an input sequence of length n, called V, where each element $v_i \in V$, is a feature vector with d dimensions. For each v_i, we plan to contextualize it in terms of every $v \in V$, including itself. To do this take each $v \cdot K$ for each $v \in V$, where K is a learned key matrix, call this output matrix VM, then take $vm \cdot v_i \cdot Q$ for each $vm \in VM$, the output of this operation will be an n length list of scores. We then take the sigmoid of this list and the resulting list, W, is a list of weights by which to scale the elements of V. Then multiply each $w \in W$ by V, which is a learned value matrix, multiply each of those resulting values by their respective indices of V, then finally sum all resulting elements to find the self-attention output v_i^* of v_i.

5.5 Focal Loss

Categorical cross-entropy loss (CCEL) is a standard loss function utilized in deep learning classification tasks. The function: $L = -\sum_{i=1}^{N} y_i * log(\hat{y_i})$, takes as input one-hot encoded labels in the form of ground truth, y, and predictions, $\hat{y}$, as per-class probability values between 0 and 1. Thus, the output of CCEL is minimized when the predicted true class is at 1 and maximized when the the predicted true class is at 0. This approach works well for standard, well balanced datasets with low intra-class variance but poorly with severely unbalanced or high variance data, even with weighting applied [8]. Due to these inadequacies and the inbalance in the MIT-BIH dataset discussed in section 5.6, we instead chose to utilize weighted focal loss (WFL), a variant

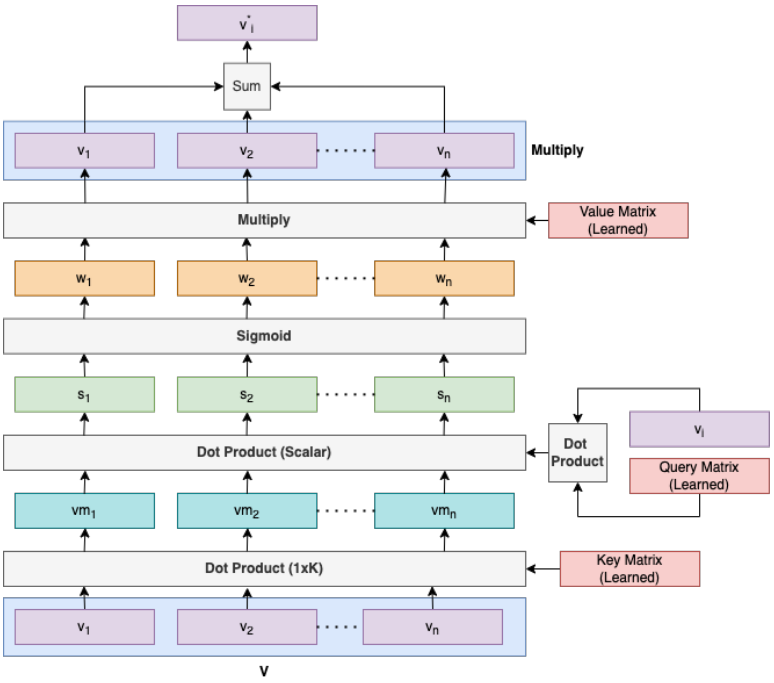

Figure 3: Self-attention re-weighing process of converting feature vector v_i to v_i^*.

of CCEL introduced in 2017 that focuses on unbalanced high-variance datasets. WFL is given by: $L = -\sum_{i=1}^{N} y_i * w_i * (1 - \hat{y_i})^\gamma * log(\hat{y_i})$, where γ and w are tunable hyperparameters. The addition of the product w_i serves as a weighing parameter, thus increasing the loss of infrequently occurring classes and vice versa. Similarly, $(1 - \hat{y_i})^\gamma$ serves as a scaling parameter, where classes that are difficult to classify have their loss scaled up, and those that are easy to classify have a scale that decays to 0 [8]. The inclusion of WFL is an improvement over the original HADLN model which utilizes the standard CCEL function.

5.6 Dataset

To train and evaluate the model we utilized the MIT-BIH Heart Arrhythmia Database [9], a commonly used dataset in training ECG classification models. As stated above, this dataset contains half-hour excerpts of two-channel ambulatory ECG recordings. These readings are assigned one of five labels: normal (N), supraventricular premature (SP), premature ventricular contraction (PVC), fusion of ventricular and normal beat (FVN), and unclassifiable beats (U). Note that SP, PVC, and FVN are all classes of arrhythmia. In addition, this dataset is notoriously unbalanced with roughly: 83% N, 2.5% SP, 6.6% PVC, 0.7% FVN, and 7.2% U samples. Our specific dataset was taken from Kaggle where it was pre-processed by converting the ECG frequency to 125Hz for all samples and then broken into sequences of 187 ECG measurements where each sequence contains an individual heart beat and is zero padded if necessary [1]. This resampling of the data created 109,444 samples, of which 87,553 are used for training and 21,891 for testing model accuracy. To further prepare the data for input into our networks we:

- Took the data, X, calculated the mean, μ, and standard deviation, σ.
- Normalized the data such that the new dataset became, Z, where $Z = \frac{X-\mu}{\sigma}$
- Converted the labels of the data to one-hot encoded vectors. Thus if the label was originally 0, which corresponds with a normal beat, then new label becomes [1,0,0,0,0]. Likewise, a label of 2 would correspond to [0,0,1,0,0]. This new labeling of data allowed for easier model implementation within PyTorch.

6 Evaluation

6.1 Evaluation Metrics

Results were evaluated utilizing a combination of standard machine learning evaluation metrics. Specifically, these metrics include the weighted forms of: F1 score, precision, recall, and average accuracy. These metrics were chosen because they are both present in the HADLN paper and are interpretable by most machine learning practitioners. The formulas for these metrics are given below:

- Note that: TP = true positive classification, FP = false positive classification, and FN = false negative classification.
- $precision = \frac{TP}{TP+FP}$
- $recall = \frac{TP}{TP+FN}$
- $F1\ score = \frac{2*precision*recall}{precision+recall}$

Thus, to compute the weighted form of each of these equations, take a weighted average of each metric over each class where the weighting is decided by the relative weight of that class within the training dataset.

6.2 Comparison Architectures

In order to accurately access our implementation of the HADLN model it was necessary to compare it with a robust set of models trained on the same dataset. For the purposes of this experiment we compared our implementation with the following model architectures:

Fully Connected (FC)
This model is a simple fully connected network intended to provide baseline results on the classification task. This model consists of: Two fully connected layers with 1,000 neurons each and rectified linear unit (ReLU) activations between followed by a final fully connected classification layer with softmax applied to ensure outputs sum to one.

Convolutional Bidirectional LSTM (CBLSTM)
The CBLSTM model consists of a single convolutional layer followed by a batch normalization, ReLU activation, BLSTM network, and finally a fully connected classification layer with softmax output. As this model contained many of the key networks used within HADLN, it served both as a more advanced comparison network and a half-way point to implementing the full HADLN model.

Original HADLN
We were unable to find the original code for the HADLN model, thus any comparisons made from our model to the original are purely for the readers interest. This model was trained and tested on the PhysioNet challenge dataset, a different ECG classification dataset. However, because this is the model we were implementing we felt if our evaluation metrics were similar to those published in the original paper then our implementation was successful.

Bidirectional Recurrent Neural Network And Convolutional Neural Network (BiRCNN) [11]
This model achieves state-of-the-art classification performance on the MIT-BIH dataset as of December 2020 and thus serves as an effective top-level benchmark of the applicability of the HADLN model to a variety of datasets outside of the PhysioNet challenge [11]. Composed of many of the same parts as HADLN, the BiRCNN model takes an input sequence and computes two parallel convolutional blocks followed by bidirectional RNN networks and finally pointwise adds these together with a third heart rate variance network also composed of an RNN model. These cumulative outputs are then fend into a fully connected classification layer with softmax applied to the output. Unfortunately, the authors of BiRCNN pre-processed their data with sequences of 361 instead of 187, thus any comparisons cannot be considered to be in one-to-one correspondence but are still informative.

6.3 Training Parameters

Each model was implemented in PyTorch utilizing the Adam optimizer, a learning rate of 0.001, batch size of 32, maximum training epoch count of 50, weighted focal loss with a γ parameter of two and relative class weights given in section 5.5, as well as a reduction of learning rate on validation

plateau of 5 epochs by 0.5. We acknowledge that model performance may be improved by further training, although due to computational limitations this was infeasible for our group.

6.4 Design Adaptations

Although our goal was to stay as close to the original HADLN implementation as possible, we made several reasonable adaptations to the original model which we feel were beneficial. Most notably, our implementation of ResNet utilizes stride size two convolutions for layer downsampling as opposed to max pooling which the original authors used. In making this change, we felt it was more logical to follow the authors of ResNet, who prefer increased stride over max pooling, due to the ability of strided convolutions to learn max pooling or other useful features. Additionally, as described in section 5.5, we chose weighted focal loss over categorical cross-entropy loss in order to account for the extreme imbalance and intra-class variance within our dataset. Finally, we made slight changes to the kernel size of our ResNet layers, from the original 16 to 17, to account our sequence size.

6.5 Tuning

Over the course of training we experimented with a variety of configurations in terms of: dropout percentage, learning rate, weighted loss, focal loss γ values, learning rate drop off, early stopping, and weight decay. We quickly found that standard categorical cross-entropy loss (CCEL) led to models predicting only normal heart beats, as they make up 83% of the training data. Weighted CCEL helped but still lead to low performance on non-normal classes, especially PVC beats which only make up 0.7% of training data. After implementing weighted focal loss and experimenting, we found the relatively high γ value of 2 helped significantly with this issue. Finally, we ran a variety of short comparisons utilizing different learning rates, weight decays, and dropouts to ascertain which values led to more rapid convergence.

6.6 Results

As shown in Table 1, our HADLN implementation performed reasonably well in classification of the MIT-BIH dataset, performing better in all metrics than both the original HADLN implementation and the self-implemented comparison models we created. Despite this, the model performed significantly worse in accuracy than the state-of-the-art BiRCNN model and surprisingly, only slightly better than the extremely simple comparison networks.

Comparisons				
Model	**Precision**	**Recall**	**F1 Score**	**Accuracy**
Fully Connected	88.80%	89.38%	89.08%	88.41%
Convolutional BLSTM	91.61%	91.99%	91.79%	92.98%
HADLN (Original)*	86.60%	85.90%	88.00%	86.70%
HADLN (Ours)	92.23%	93.97%	92.86%	94.00%
BiRCNN (SOTA)**	N/A	N/A	N/A	99.60%

Table 1: Comparisons between each network on classification of the MIT-BIH test dataset.

*The original HADLN network utilized a different testing and training dataset, thus comparisons cannot be considered in one-to-one correspondence. Performance of the original HADLN network was only included to display similar results.
**The BiRCNN network utilized the MIT-BIH dataset but segmented the data into sequences of 361 instead of 187, thus comparisons cannot be considered in one-to-one correspondence. Additionally, precision, recall, and F1 scores were not published for this model.

7 Conclusions

7.1 Evaluation

As shown above, our HADLN implementation performed better than all but the state-of-the-art model in our comparison testing. Despite this, we feel that given the complexity of the HADLN network and thus much higher training times, it did not perform sufficiently better than its competitors, especially in comparison to the jump in performance attained by BiRCNN. For reference, HADLN took roughly 16 hours to train while the CBLSTM model, which achieved only 1.02% less accuracy, took under two hours. We feel that this drop-off in gains could be due to a variety of factors including: not enough training samples to adequately exploit HADLN's relatively deep architecture, too low of a learning rate, a relative failure in effectiveness of the HADLN model, or too few computing resources to adequately train the model. Despite the potential issues with this architecture, it is promising to see that ECG classification can be done with such a high degree of accuracy.

7.2 Lessons Learned

In implementing the HADLN network we quickly found how important it is to have an in-depth understanding of the model one is implementing. Without fully understanding the intuition behind LSTMs, ResNets, or attention mechanisms they can be difficult or impossible to implement and even harder to debug. After learning the importance of this, development speed hastened and project quality increased significantly. We also learned the importance of descriptive documentation in published research. Due to omissions of key architecture details in the HADLN paper including a poor explanation of how attention was utilized in the model and a lack of detail about ResNet layer parameters, we had difficulty implementing the exact model described.

7.3 Overview

In this work we implemented, analyzed, and reviewed the HADLN architecture on the task of ECG classification utilizing the MIT-BIH dataset. On this dataset we found that the HADLN model performed better than reported in the original HADLN paper, but significantly worse than the state-of-the-art and only slightly better than other simpler models classifying the same data. Due to the slight performance boost and significantly more complex architecture, we propose that there are likely models better suited to the task of ECG classification, such as the BiRCNN model.

References

[1] Ecg heartbeat categorization dataset (mit-bih). `https://www.kaggle.com/shayanfazeli/heartbeat`. Accessed: 2021-12-05.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Jonathan Guo and Bin Li. The application of medical artificial intelligence technology in rural areas of developing countries. *Health equity*, 2(1):174–181, 2018.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] Mingfeng Jiang, Jiayan Gu, Yang Li, Bo Wei, Jucheng Zhang, Zhikang Wang, and Ling Xia. Hadln: Hybrid attention-based deep learning network for automated arrhythmia classification. *Frontiers in Physiology*, 12, 2021.

[7] Phong Le and Willem Zuidema. Quantifying the vanishing gradient and long distance dependency problem in recursive neural networks and recursive lstms. *arXiv preprint arXiv:1603.00423*, 2016.

[8] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[9] George B Moody and Roger G Mark. The mit-bih arrhythmia database on cd-rom and software for use with it. In *[1990] Proceedings Computers in Cardiology*, pages 185–188. IEEE, 1990.

[10] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[11] Pengwei Xie, Guijin Wang, Chenshuang Zhang, Ming Chen, Huazhong Yang, Tingting Lv, Zhenhua Sang, and Ping Zhang. Bidirectional recurrent neural network and convolutional neural network (bircnn) for ecg beat classification. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2555–2558. IEEE, 2018.