Accuracy Optimization of a Convolutional Neural Network for Real-Time Pulse Detection

Duncan Boyd, Bailey Boyd*
86101888
duncan@wapta.ca

Abstract

During CPR, automatically detecting the pulse can save precious seconds compared to manual palpation. While devices exist for pulse detection[2], there are limited options when it comes to rigourous detection algorithms. This paper will explore convolutional neural networks (CNNs) as an option for the real-time binary classification of a pulse reading. Using a pulse wave dataset, we swept several critical hyperparameters, optimizing for accuracy. Our final model has achieved an accuracy of 95%, indicating that the use of CNNs for pulse classification is viable.

1 Introduction

Cardiopulmonary resuscitation (CPR) is one of the key interventions in cases of cardiac arrest. One of the critical steps during CPR is checking for the return of spontaneous cardiac activity by manually palpating the patient's pulse. Pauses greater than 10 seconds, the current limit recommended by the Canadian Heart and Stroke Foundation, are associated with increased mortality and morbidity [1]. The purpose of this project is to explore a real time algorithm for pulse detection during CPR. Our goal is to develop an algorithm that is superior to manual palpation, which is approximately 90% accurate [2]. This project documents the development and tuning of a 1D CNN that can beat manual palpation in the detection of the pulse in real time.

2 Input Data

2.1 Dataset Description

The dataset used for this work was collected as part of UBC ECE Capstone Group LS-15's project. It is a series of four second segments, each a time series of pulse data described as sensor values (arbitrary units). Data is sampled at 60Hz, meaning each sample is 240 points in length. A typical sample might look like the plot as shown in figure 1.

*See appendix A. I like using 'we' on formal papers.

[2]For example, my capstone group's device. Note that although it was inspired by it, this work was NOT done as part of my capstone project, which used another algorithm.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

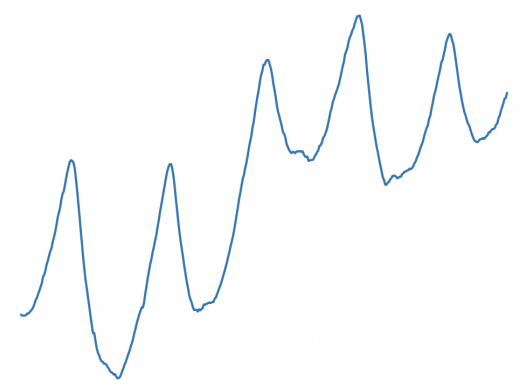


Figure 1: Typical pulse sample

Each data point (four second segment) is labelled by hand as either having, or not having a pulse. The sample in figure 1 would be labelled as having a pulse. Labelling was performed by non-experts, and as such is expected to have some error.

The dataset comprises of 701 data points, collected in a variety of test conditions. The distribution of the data is shown in figure 2. 61% of data has a pulse, while 39% does not.

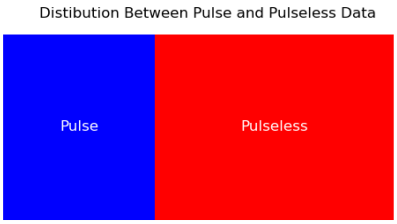


Figure 2: Dataset distribution

Although many samples such as figure 1 may have an easily visible pulse, some samples are more challenging, with spikes created by noise or weak pulse signals.

2.2 Preprocessing

As seen in figure 1, data often has high and low frequency noise that we would like to filter. After flooring the data by setting the lowest value to 0, we apply a band pass filter as seen in figure 3. This passes frequencies from 0.5Hz to 5Hz, which translates to 30 BPM and 300 BPM, considered the possible range for the human heart rate.

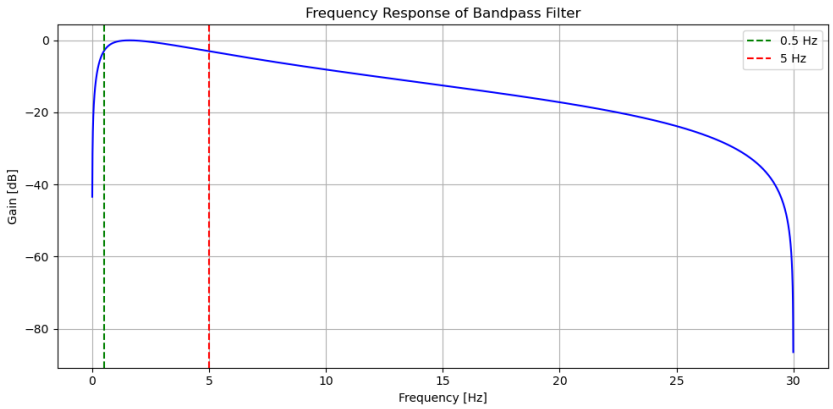


Figure 3: Frequency response of band pass filter

Data often has a strong baseline shift. To eliminate this, we fit a 4th degree polynomial to the data, and then remove that, treating it as a baseline. An example of a fitted polynomial can be found in figure 4.

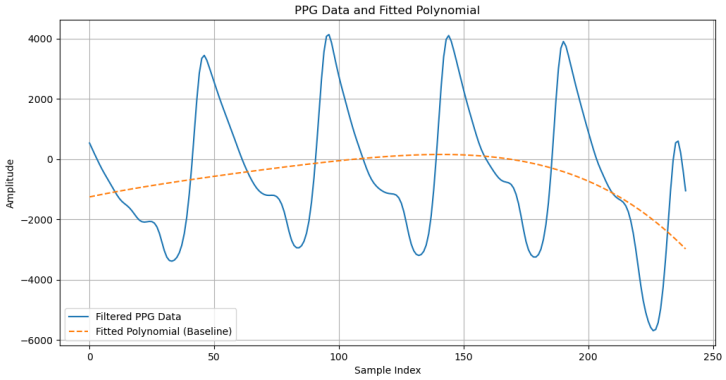


Figure 4: Polynomial fit to data sample

Finally, data is normalized. The full process makes our data more uniform, allowing for our CNN to focus on features that indicate pulse.

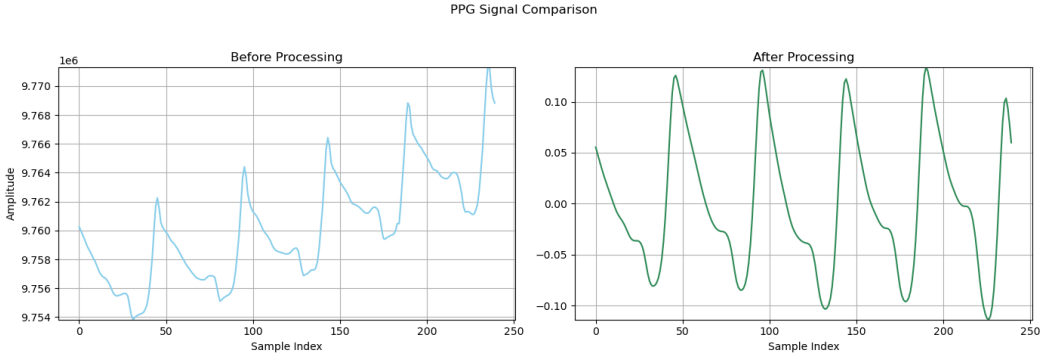


Figure 5: Before and after data preprocessing

Although this preprocessing can radically change data points, this results in clean data that performs well in our classification model. As a final step, when data is loaded for model training, it is shuffled randomly to mix up different sections of the dataset.

3 Model Structure

To classify four second signals into either a pulse or pulseless category, a 1D CNN was chosen. 1D CNNs are suitable for real time pulse detection because of their efficient feature extraction, resistance to noise, and potential for deployment on devices with limited hardware. These advantages come at the cost of needing a large, diverse dataset to avoid overfitting.

A typical 1D CNN structure was used, with convolutional layers followed by pooling layers followed by dense layers. A dropout layer was added to test the effectiveness of setting some inputs to zero. As seen in figure 6, the number of convolutional and max pooling layers is dependant on the hyperparameter N.

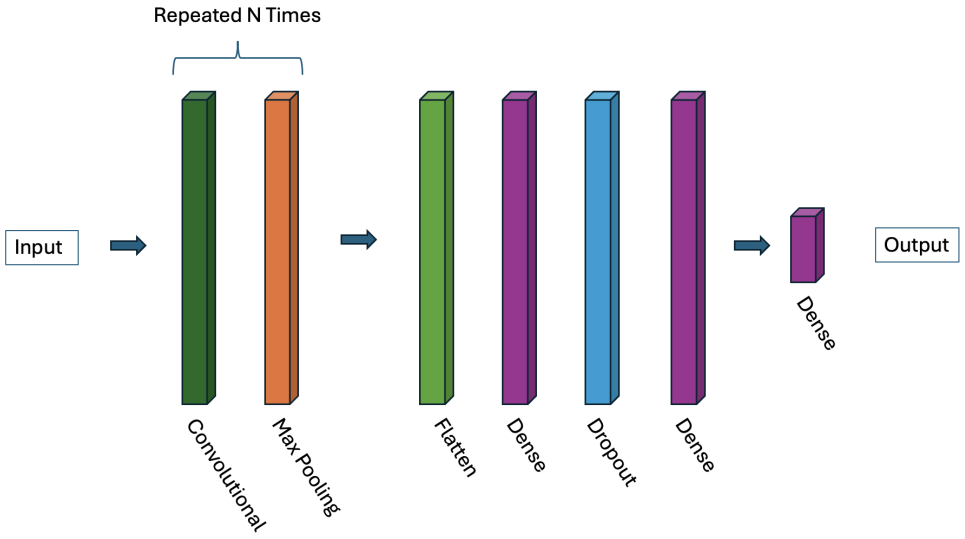


Figure 6: Structure of 1D CNN

4 Model Training

Training was performed using a desktop PC equiped with a Nvidia 1660 Super GPU. Because of limited compatibility with software designed to take advantage of this hardware, Tensorflow 2.0 was used. Several steps were taken to train and evaluate this model accurately, especially considering the small dataset.

4.1 Data Augmentation

The dataset used is collected on few people, and includes only heartrates within a narrow range of possible values. To compensate for this, data augementation was used. The total size of the training dataset was doubled, raising it from 630 data points to 1260. For each sample, a variation was created with an amplitude and a timescale scaled by a random factor.

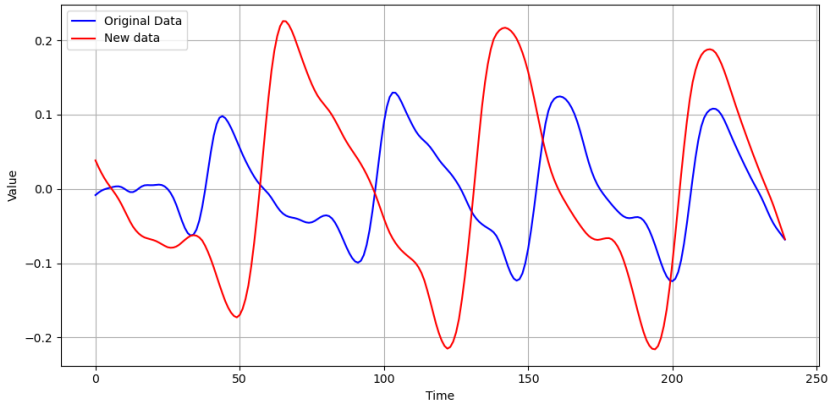


Figure 7: Example of augmented data

4.2 Data Splitting

Data is split into a training, test, and validation set. This results in a training set of 1008 points (augmented), a validation set of 252 points (augemented), and a test set of 71 points (not augmented). To mitigate large variations in results due to the small test set, 10 fold training was used. This means that the data was seperated into 10 random train test splits, then trained on each of these models.

4.3 Evaluation

For each set of hyperparameters that we want to test, 10 models are trained using 10 different data splits. The accuracies of these 10 models are averaged to get the accuracy for a given set of hyperparameters.

Models are trained with the Adam optimizer and binary crossentropy loss, common choices for this type of model. Models are trained for a maximum of 200 epochs, but training will be stopped early if no improvements are seen after 5 epochs.

5 Hyperparameter Sweeps

Four hyperparameters were targetted for sweeping, convolutional layers, dense points, dropout rate, and L2 rate. See the sections below for more details about these parameters.

When sweeping a parameter, the remaining parameters were set to the values in table 1. These default values were established during preliminary sweeps. Sweeps were evaluated based on the accuracy of each model, as determined in section 4 above.

Table 1: Default parameters

Parameter	Value
Convolutional Layers	3
Dense Points	128
Dropout Rate	0.2
L2 Rate	0.06

5.1 Convolutional Layers

The number of convolutional layers is varied by adding N convolutional layers to the model, each followed by a max pooling layer (see figure 6). The number of filters of each subsequent convolutional layer is doubled, starting from 64 filters. This means that a model with N = 3 would have three convolutional layers with 64, 128, and 256 filters each.

The number of convolutional layers was swept from N = 1 to N = 6. The results of this sweep showed that having more layers was generally better. The best performing model used four layers for an accuracy of 93%.

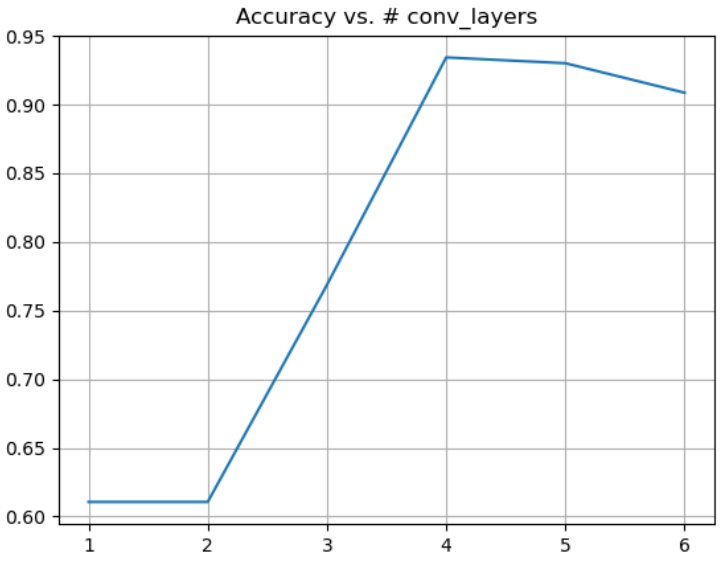


Figure 8: Convolutional Layers Sweep Results

5.2 Dense Points

The number of dense points in the first two dense layers was swept. A ratio of 2:1 was maintained between the first and second layer while the number of dense points was swept from 32 to 2042, incrementing in powers of two. This means when testing 256 points, the second dense layer has 256 points while the first has 512 points.

Surprisingly, the results of this sweep showed that having fewer dense points was generally better. This may be due to overfitting with more dense points. The best performing model used 32 and 64 dense points for an accuracy of 92%.

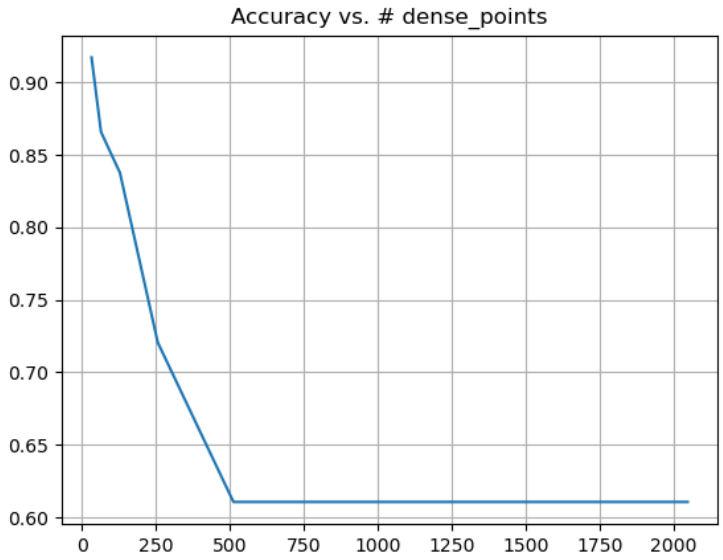


Figure 9: Dense Points Sweep Results

5.3 Dropout Rate

Dropout rate was varied in the dropout layer (see figure 6). The rate was varied from 0 to 0.5 with 0.01 increments. The results have little trend for this parameter, with smaller variance in accuracy compared to previous parameters. The best dropout rate was found to be 0.21, with an accuracy of 92%.

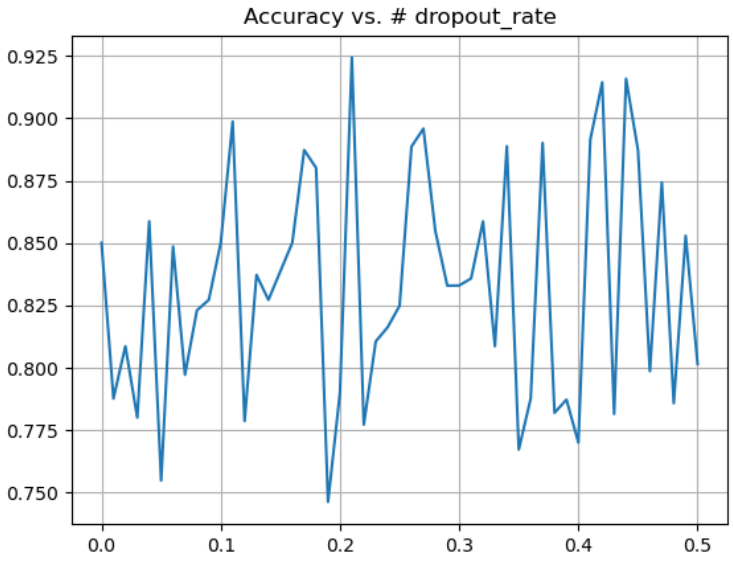


Figure 10: Dropout Rate Sweep Results

5.4 L2 Rate

L2 decay rate was used in the first two dropout layers. The rate was varied from 0 to 0.1 with increments of 0.005. It seems that a lower L2 rate leads to higher and more consistent accuracy. The best L2 rate was found to be 0.0, with an accuracy of 92%.

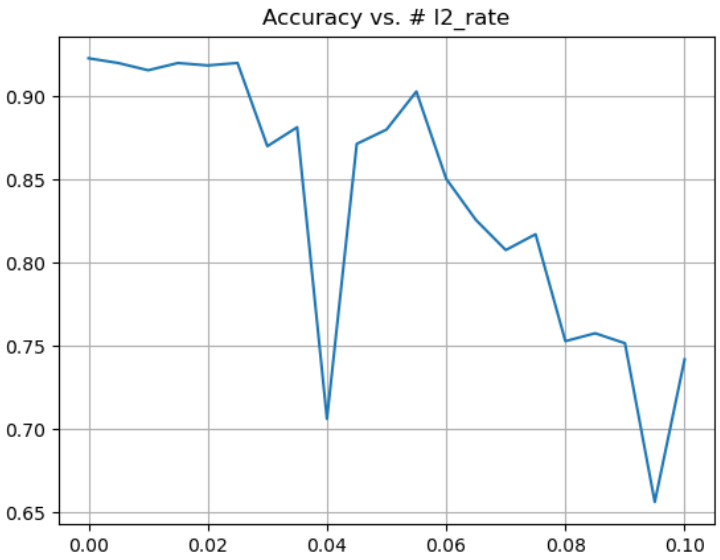


Figure 11: L2 Rate Sweep Results

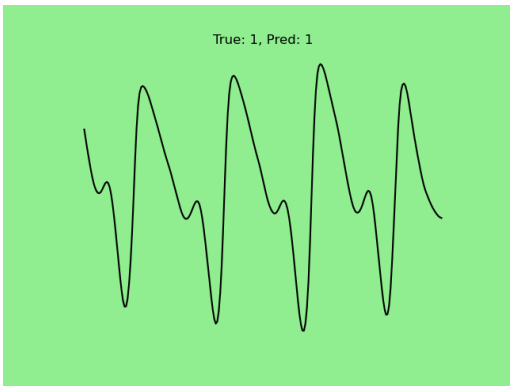
6 Results

The hyperparameters as determined by the hyperparameter sweeps are shown in the table below.

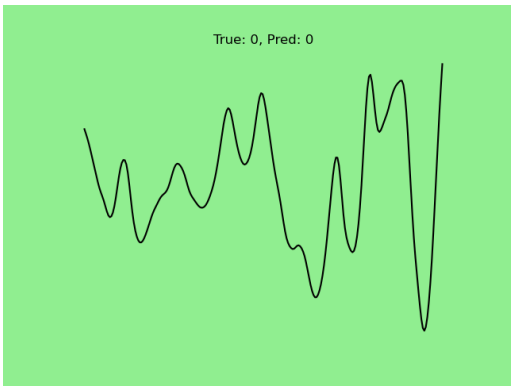
Table 2: Optimal parameters

Parameter	Value
Convolutional Layers	4
Dense Points	32
Dropout Rate	0.21
L2 Rate	0.0

After training a model with these parameters, the final model has an accuracy of 95%. This exceeds any of the accuracies achieved on the parameter sweeps, indicating that this combination is effective. Some examples of pulse data and the predictions of this model are displayed in the figures below.



(a) Pulse Data



(b) Pulseless Data

Figure 12: Correct predictions

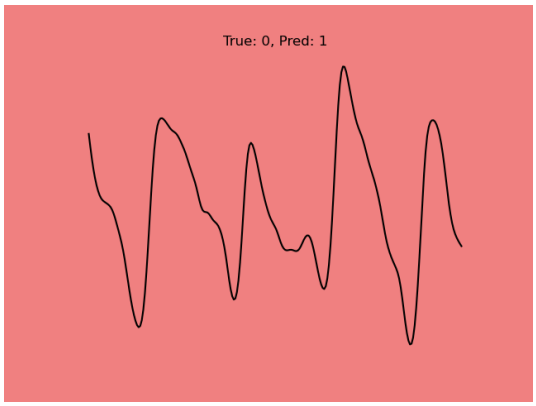


Figure 13: Incorrect prediction

As seen from figure 13, the data sample may actually have been labelled incorrectly, but identified correctly by this machine learning model. Many incorrect guesses seem to be on samples that, from a human perspective, may or may not have a pulse.

7 Discussion

These results seem extremely promising, with an accuracy that far exceeds many conventional pulse detection methods . This seems to be a suitable model to implement in a setting where real-time pulse detection is needed and limited data is available.

There are however some limitations that need to be investigated:

- This model is likely overfit, as the dataset it is trained on includes data from few people and at only a narrow range of possible heart rates. Training and testing this archetecture on a larger, more diverse dataset would greatly improve its practical use.
- If the hyperparameters are interdependant, the hyperparameters could be further refined with more sweeps. Although gains may be incrementally smaller, this could likely improve the accuracy somewhat.
- Further hyperparameters may have an effect on the model and should be investigated. Optimization of this model still has a lot of potential, as only four parameters were investigated.

Despite these limitations, this model has yielded extremely impressive results, especially considering the small and somewhat imperfect dataset. There is much potential in using 1D CNNs for pulse detection.

References

[1] P. A. Meaney et al. “Cardiopulmonary Resuscitation Quality: Improving Cardiac Resuscitation Outcomes Both Inside and Outside the Hospital,” Circulation, vol. 128, no. 4, pp. 417–435 Jul. 2013, doi: 10.1161/CIR.0b013e31829d8654.

[2] L. M. Cunningham, A. Mattu, R. E. O’Connor, and W. J. Brady, “Cardiopulmonary resuscitation for cardiac arrest: the importance of uninterrupted chest compressions in cardiac arrest resuscitation,” The American Journal of Emergency Medicine, vol. 30, no. 8, pp. 1630–1638, Oct. 2012, doi: 10.1016/j.ajem.2012.02.015.

A Co-author

Figure 14: Co-author diligently reviewing results

B GitHub Repository

Link to GitHub repository with code and results