Compressed Convolutional Neural Networks for the Classification of Electrocardiogram Signals on Embedded Devices

Laurentiu Birton
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
l.birton@student.utwente.nl

ABSTRACT

To determine abnormal conditions of the cardiac muscles, medics usually rely upon electrocardiography, a process that maps the electrical activity of the heart onto an electrocardiogram. Classic methods used to read and classify ECG's can be sensitive to the difference between patients or require deep specific knowledge and can be time-consuming. As an alternative, Machine learning is often used nowadays to automatize the classification of ECG signals, mostly for the detection of cardiovascular diseases. However, this comes with a large computational cost, usually requiring large and powerful machines to perform the classification task. This can be very limiting, especially in situations where high mobility or low costs are desired. Since research into the application of efficient machine learning methods for use in low powered devices has greatly increased in the last years, here we show an investigation into the compression of Convolutional Neural Networks for use in embedded devices for the classification of ECG signals. We describe the data we used, the employed compression techniques, what type of models we have analysed and finally show the results of our experiments together with possible ways of improving the performance of these models in the future. By the end of this paper, we'll have shown how a CNN can be compressed to sizes of down to under 1 Megabyte and still obtain very good accuracy in heartbeat classification.

Keywords

electrocardiogram, classification, compression, convolutional neural network, embedded devices

1. INTRODUCTION

The field of Artificial Intelligence has seen a huge amount of progress in the last decade. Using machine learning, electronic devices are capable of achieving tasks previously considered close to impossible. A particular class of ML algorithms, called neural networks, have proved very successful in tasks like image classification, video processing or speech recognition [17]. Convolution Neural Networks(CNN) are one such type of algorithm, being capable of achieving an accuracy close to or better than humans in some image recognition tasks [11]. CNN's have also been successfully employed in healthcare, specifically for the task of biological signal analysis [2]. One such type of signal is Electrocardiogram data, which is used to detect possible heart conditions in patients. Due to the cost and size of usual ECG machines, as well as the expertise needed when analysing an ECG diagram, in this paper we take a look at the possibility of using CNN's on low powered devices, by analysing various successful CNN architectures, compressing them and seeing how well they perform on our device of choice.

CNN's can produce such results due to their feature extraction and codification methods, which capture high-level relationships in the spatial and temporal dimensions of the data [9]. There are 2 distinct layers through which input data, usually referred to as input volume due to the multi-dimensional aspect, goes through in a CNN, before being fed to a standard type of neural network. These are the *Convolution Layer*, used to map regions of the input volume to a single output, and the *Pooling Layer*, used to further reduce the size of the convoluted features, by sub-sampling the output from the previously mentioned layer. Together, these steps can be repeated any number of times, allowing CNNs to easily scale for large data instances [15].

In the last few years, besides artificial intelligence, mobile computing has also seen a hegemonic period. As a result, different techniques have been proposed to compress CNNs so that devices with limited computing capability(relative to modern desktop computers) would be able to at least perform the classification tasks. Methods for compression include *Parameter Quantization and Binarization*, *Parameter Pruning and Sharing* and *Knowledge Distillation* [19].

Bringing attention to a different field, we look at the challenges posed by the detection of cardiovascular diseases(CVD). Electrocardiography is a process through which the activity of the cardiac muscles is mapped onto a graph representing electric charge against time. Cell membranes of a heart constantly undergo voltage changes, which are caused by the movement of charged ions between and the inner and outer parts of the cell. Using electrodes, this change of voltage can be recorded, and, when plotted against time, a type of graph is created known as an electrocardiogram(ECG). These ECGs can either be classified by specialised software on ECG machines or be manually read by a specialist. Both of these solutions come with downsides, however, as there can be significant disparities between the signals acquired from different patients, and manual reading requires a piece of deep knowledge and is time consuming. [8]. In the last years, different machine

28^{th} *Twente Student Conference on IT* Febr. 2^{nd}, 2018, Enschede, The Netherlands.

learning techniques have been proposed to facilitate the classification of ECG signals. CNN's are a natural candidate to help with such a problem, due to their known high accuracy in signal and image processing. In their paper surveying the application of CNNs for different types of body anomalies detection, Sarvamangala and Kulkarni have reached an accuracy of 92 per cent when classifying ECG signals [16]. However, they point out one problem with this method, which was the long time to train the model. Indeed, model size and computational complexity can hamper the use of CNNs, especially in critical situations when real-time data analysis and high mobility are required [13]. Cardiovascular disease is the number one cause of deaths worldwide, with an estimated amount of 18.6 million people dying in 2019 due to CVDs [14]. The price of ECG machines is also relatively high, going up to hundreds of euros. As a result, these can be quite inaccessible to some medical facilities, especially in developing countries. This poses a huge problem, as early detection and intervention can be vital in preventing CVD deaths.

In the light of this information, we can see the potential a low cost, highly mobile device would have in helping medics detect cardiovascular disease, especially in edge or in-field situations. Here we show an analysis of how successful Convolution Neural Networks algorithms, when compressed to run the inference task on low-powered devices, would be in detecting cardiac anomalies.

This leads to the following research question, which covers the two most important aspects we're going to research: accuracy and performance.

1.1 Research question

Main Question: How can a CNN be modelled and compressed so as to produce highly accurate ECG signal classifications on a low powered device?

This in turn will be answered by analysing the following sub-questions:

Sub-RQ1: How accurate is a compressed CNN running on an embedded device in classifying ECG signals?

Sub-RQ2: What are the computational resources required by a compressed CNN algorithm?

To carry out these questions, we analyse recent literature and evolution in the study of neural networks, focusing on the use of CNNs to classify ECG signals and the compression of CNNs for deployment on low powered devices. Then, using a modern desktop PC, we build and train various CNN algorithms, compress them, and finally deploy them onto a Raspberry Pi Zero to analyse their performance. As a source of training and classification data, we use the open-access MIT-BIH Arrhythmia Database [12], which contains ECG readings from various patients, courtesy of BIH Arrhythmia Laboratory.

The experiments will be carried out by building and training different CNN architectures on the desktop PC. Then we'll compress these through various methods, save the compressed models, and transfer them over to the Raspberry Pi. We will then analyse their performance using various metrics (accuracy, precision, recall) and size reduction. Finally, we draw conclusions from these experiments and propose possible improvements and future work on this topic.

2. RELATED WORK

Because of the relative novelty of both efficient neural networks and decently powerful embedded devices, research into this topic is still in its infancy and subject to rapid changes.

Y. Chen et al. give a description of the state of the art on the problem of deep learning algorithm compression for use on mobile and embedded devices. They outline the challenges that come with this topic and describe 7 techniques that can be used for compression. They group these methods into *pruning, quantization, model distillation, network design strategies, low-rank factorization, other and hybrid*. This will be used as a guiding point for our research, as we will both employ some of these techniques and further follow the topic to discover different or newer techniques than what was presented here. Since this is a survey on the topic of compression, it doesn't give us any information on how these methods actually perform on the ECD classification task [3].

In one of the earliest takes on the subject, researchers have shown in 2018 how CNNs can be used for real time myocardial infarction and the efficiency of this method when deployed on a low power Cortex-A9 processor [10]. They have shown that such devices are suitable for real time application, with their embedded implementation capable of processing each heartbeat in 26.75ms, or 1.07s for a 40 heartbeat segment. They use a custom multi-lead-CNN, which captures distinct features from each lead of the ECG machine. This work shows the possibility and potential of using CNNs for this task, but mainly focuses on the preprocessing and feature extraction part of the process and doesn't try to compare how different type of algorithms. The platform they used, Terasic DE1-SoC, while it can be considered low power, is still quite expensive. We propose carrying out our research on a much cheaper device.

In 2020, A. Faraone and R.D-Gonzalo were able to design a Convolution Recurrent NN that could run on an nRF52832 general purpose SoC and classify between normal heart rhythm, atrial fibrillation, noise and other rhythms [4]. They were able to achieve an accuracy of 85.7 per cent and an execution time of 94.8ms per window of data(160 ECG samples). They're program also occupied only 210KB of storage memory, while storing no more than 7KB of data in RAM at the same time. This improves on the previous paper by using a device with considerably less processing power, but still focuses on building its own custom algorithm and doesn't compare this against different solutions.

3. CNN'S, COMPRESSION AND DATA

3.1 Convolutional Neral Networks

Convolutional Neural Networks are a class of Deep Neural Networks specialized for the classification of image data, which has found successful use in various type of applications. These include image recognition, recommendation systems, medical imaging analysis, natural language processing and so forth.

In the medical field, CNN's have already been used to facilitate classification tasks usually performed by specialists or specialised devices. Such applications are bio signal analysis, seizure and cardiac arrest prediction, drug discovery or the analysis of electronic health records. ECG signals are part of the lattermost domain. [18]

CNN's are based on three main building blocks: convolution layers, pooling layers and a fully connected layer.

A convolution layer is used to extract features from an input set of data using a *kernel*. A kernel is a multi-dimensional data structure, usually a 2D or 3D matrix, depending on the input, which is multiplied in sections

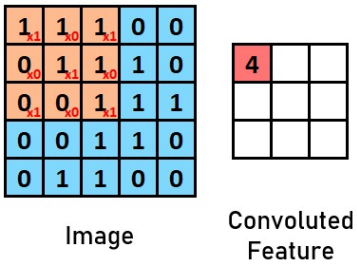


Figure 1: Example of a convolution: a 3x3 kernel is applied over a 5x5 input, producing an output structure with dimensions 3x3 and a value of 4 at the first index

against the input data. A kernel also has a *stride* value, which represents the number of positions over the input that separates consecutive operations. The kernel extracts relations between data points, creating feature crosses along the way. Multiple kernels (kernel with different values) can be applied during one convolution to obtain a higher dimensional output, in which each set of points along the new dimension represents a different type of high level feature. As an example, one kernel might record spatial correlations along the input image, while another can capture the edges of objects in the input data.

Pooling layers follow after convolutions and are used after to reduce in plane dimensionality of features. This is done both to reduce the number of parameters and decrease model size, and to maintain feature correlation in case of small changes in values. A pooling operation has a dimension, just like a kernel, which is the space over the input where the values are pooled. In that space, pooling is done in usually 2 ways: max pooling, where the largest value in the pooling space is used in the output, or average pooling, where the average of all the values is taken. Again, just like the kernel, pooling operations have a stride, which states how many positions should the pooling move between operations.

The output of the final convolution or pooling layer is the flattened and fed into a *fully-connected layer*. This is a typical neural network, where each input value is connected to output values with a set of weights. An FCL can have multiple layers containing nodes, in which each node represents an output value based on the input and the weights associated with it. The last layer of the FCL, also called the output layer, has as many nodes as there are classes used for classification, and it determines under which category the data falls into.

Anywhere in-between the previously mentioned layers an activation layer can be used. This makes the model perform better at predicting non-linear data. One such example is ReLu (Rectified Linear Unit) [1], which for each value X taken as input, it outputs max(0, X)

Furthermore, we will look at different CNN architectures and how they can be used to classify the recordings from the MIT-BIH Arrythmia Database.

3.2 CNN compression

Compression in the context of Convolutional Neural Nets

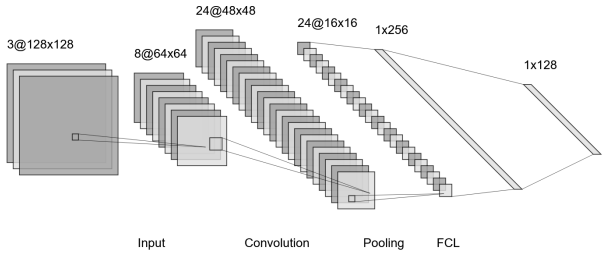


Figure 2: Typical CNN architecture

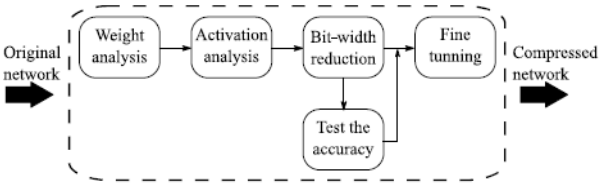


Figure 3: Example of quantization process. Here, Ristretto quantization is shown [13]

refers to the reduction in both size and complexity of the model, so the storage space and inference time is greatly reduced. In their survey paper on this topic, Pilipovic et. al group compression techniques in 3 main categories [13]:

- Precision Reduction
- Network pruning
- Design of compact neural networks

Precision reduction is done by changing the format in which values for weights and parameters is stored. Usually, these are stored in 32 bit floating point format. By applying quantization, these can be further reduced to a smaller value, like 16 or 8 bit, in either fixed or dynamic point format. For the purposes of this paper, we are using TensorFlow's 8 bit training aware quantization. Training aware means that after compressing the network, we re-fit the compressed model again on a subset of data. Another method of precision reduction employed here is weight clustering, where weights with values close to one another are given a similar value. These techniques help in both reducing model size and speeding up inference time

Network pruning is done by gradually setting non-salient weights to zero. In an iterative process, weight are set to 0 and the impact of this action on the model's loss is evaluated. In this way, weights that contribute little to the model's overall performance are nullified in a step-by-step process.

Compact neural nets are networks designed specifically for use in a low-performance environment, such as mobile phones or microcontrollers. They usually have small model sizes and result in faster inference times than a "traditional" model. In this paper we train and deploy some of this, specifically MobileNet and EfficientNetB0, analyse their performance compared to larger models that we compress and look and possible improvements.

3.3 Dataset

Our dataset of choice for this project is the MIT-BIH Arrhythmia Database. This is a repository containing 48 half-hour ECG recordings, each belonging to a different patient, obtained by the Beth Israel Hospital Arrhythmia Laboratory between 1975 and 1979. These were chosen

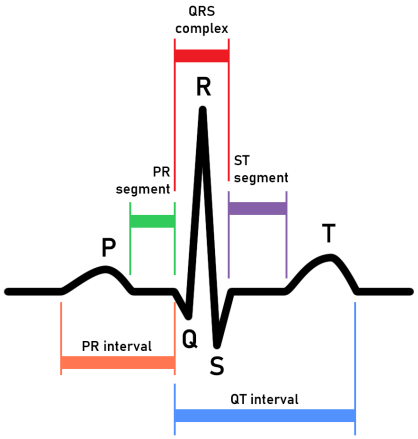


Figure 4: QRS comlex pattern

Label	Description
N	Normal beat
L	Left bundle branch block
R	Right bundle branch block beat
B	Bundle branch block beat
A	Atrial Premature Beat
J	Nodal premature beat
S	Supraventricular premature or ectopic beat
V	Premature Ventricular Contraction
r	R-on-T premature ventricular contraction
F	Fusion of ventricular and normal beat
e	Atrial escape beat
j	Nodal escape beat
n	Supraventricular escape beat
E	Ventricular escape beat
/	Paced beat
f	Fusion of paced and normal beat
Q	Unclassifiable beat
?	Beat not classified during learning

Table 1: Beats in the MIT-BIH database

from a set of 4000 holster recordings, and can be divided in two groups, based on how they were chosen. 23 recordings were selected randomly from this set and are supposed to represent common waveforms and artefacts that an ECG device might encounter with typical use. The 40 minutes excerpts from each recording were also selected randomly. The other 25 records were picked to include signals with features (like rhythm, QRS morphology and signal quality) that are expected to pose difficulties for arrhythmia detectors. These are described as "complex ventricular, junctional, and supraventricular arrhythmias and conduction abnormalities" by the MIT-BIH Arrhythmia Database directory.

The two signals in each recording come two lead configurations, modified by placing the electrodes on the patients' chest. The upper signal is a modified Limb II Lead, while the lower one is a modified V1. However, this is not the case for all recordings, with some of them using different lead configurations (like V2 or V5). One recording also has the lower and upper signals inverted. The argument for this is that arrhythmia detectors should be able to deal with such situations.

The recordings were originally stored on tape, being digitized afterwards with an Analog-to-digital converter. The analogue signal was filtered for saturation and anti-aliasing and were sampled at a rate of 360Hz. The digital signal originally used 11-bit values to represent a signal sample, so sample values ranged from 0 to 2047, with 1024 representing 0 volts. These were further reduced to 8-bit first difference format values, due to storage limitations.

The records were then labelled, with a label applied for each heartbeat. This was first done with a QRS detector, which marked each beat as normal. Then, for each record, two paper sheets were printed containing the entire 30-minute excerpt, which added labels where these were missing and correctly labelled abnormal beats. They also labelled the heart rhythm and signal quality from the recordings. In total, the database contains 109,000 labelled beats, some of which were revised throughout the years [12].

Heart signals are modelled using the QRS complex, which represents the electrical patterns exhibited by a heart during a heartbeat. It is composed of 4 sub-waves present in the entire model, which together form 5 different segments. For a healthy heart, the wave pattern represents the following:

P-Wave: atrial depolarization, as electric charge "passes" through the atria (the part of the heart though which blood flows inside). Characterized by a small uniform transition.

Q-Wave: downward charge following the P wave. Represents depolarization in the interventricular septum, the "wall" separating the ventricles.

R-Wave: immediately following the Q-wave, represents the charge passing though the ventricular walls. Due to the thickness of these walls, more charge is required, resulting in the spike nature of this wave.

S-wave: depolarization in the Purkinje fibres, which are located inside the ventricular walls. This immediately follows the R-Wave, and the dip comes from the charge being moved out of the ventricular walls.

T-Wave: before the cycle repeats, both ventricles are repolarized. This results in this wave appearing at the end of a heartbeat.

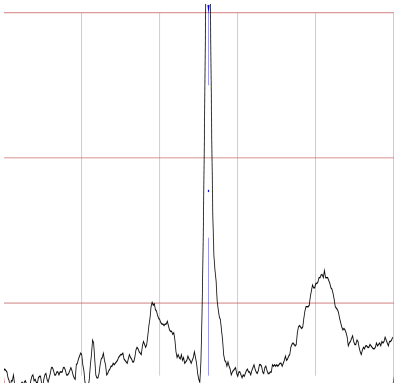
There are multiple types of abnormal heartbeats labelled in the MIT-BIH database. The second example in fig. 5 is a premature ventricular contraction, which happens when a heartbeat is initiated by the Purkinje fibres in the ventricles, rather than the sinoatrial node in the right atrium. As it can be seen, it is characterized by a sudden discharge of the fibres, followed by a surge in the ventricular walls.

Another example of an abnormal beat (fig. 5, (c)), a fusion beat occurs when a charge from multiple sources is collected in the same region at the same time. This can happen in the atrial chambers (atrial fusion beats) or ventricular chambers (ventricular fusion beat).

The following is a table of valid beat types present in the database (valid means the recording wasn't noisy or a measuring artefact):

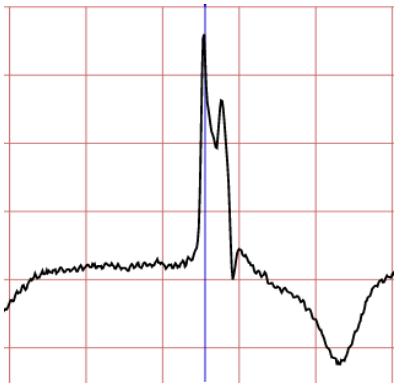
4. TOOLS AND METHODOLOGY

In this section, we will go over the tools and procedures used in carrying out this research. We first describe the hardware used here, then move on to the preprocessing and extraction of features from the MIT-BIH dataset. Afterwards, we describe the CNN architectures we chose for compression and give an account of the hyperparameters



(a) Example of normal heartbeat from the MIT-BIH database

(b) Premature ventricular contraction (subject 200, peak time=1.905s)



(c) F-beat (Subject 217, peak time = 1.27.697m)

Figure 5: Heartbeat examples from the MIT-BIH database

used for these. Finally, we also describe the metrics we use for measurements and give a detailed description of the procedures through which these measurements were carried out.

4.1 Tools

The entire programming process (building the model, training, compression, testing) was done using the TensorFlow library in Python3. Two devices were used along the way: a desktop computer for building, training and compressing the models and a Raspberry Pi Zero for deployment and testing of compressed models. The Raspberry Pi Zero was chosen since it is a low cost, a low power hardware platform that suitably represents the type of devices we target in this paper.

The specifications of the desktop PC are as follows:

- Operating System: Windows 10 Professional
- CPU: Intel Core i5-7600K CPU @ 3.80GHz 3.8GHz
- Memory: 16GB RAM DDR4
- GPU: NVIDIA GeForce GTX 1070 Ti
- Video Memory: 16GB Ram GDDR5
- Nvidia Cuda version: 11.3
- Nvidia driver version: 27.21.14.6589

The Raspberry Pi Zero has the following specification:

- Raspberry Pi OS Lite
- CPU: 1-GHZ, Broadcom BCM2835
- Memory: 512MB RAM
- SD Card Size: 16GB

To transfer data between the PC and the Pi, the Pi was connected to the same wireless network as the PC and an SSH connection was made from the desktop. Most of the coding was done on PC, only the necessary files being moved over to the Pi for testing (heartbeat data, models, python scripts).

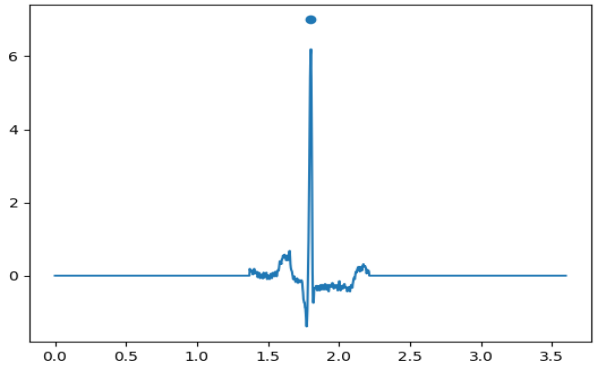


Figure 6: A single heartbeat captured in a 3.6s time window

4.2 Methodology

Preprocessing

Data is read from the MIT-BIH database files one subject at a time. Before any other operations can be done, the values normalized by removing the mean and are scaled in units of standard variance. That is, for each data point X, the normalized value is $f(X) = (X-u) / d$, where u is the mean and d is the standard variance of all data points values of given a subject. Subsequently, we build the actual input data for our models, by simulating an image from the given sample values around each heartbeat present in the dataset. These steps are detailed in the following section.

Data structuring for classfication

Data in the MIT-BIH is represented by a continuous set of values. As such, we have used the signal values present in the database to build an "image" of each heartbeat, which can be fed to many pre-existing models, that have already proved highly performant. In the MIT-BIH database, we are provided with the location for each R peak of a heartbeat, where a label was manually applied during the databases' digitalization. Since the electrical signal was sampled at a rate of 360Hz, implying 360 values for a second of recording, we can create a time window around each R peak to capture an individual heartbeat. We have chosen a time window of 1.8s before and after each peak, from which we capture every value and associate this value set with the label at the peak. This results in a 3.6s time window, in which 360 values will be captured. This was chosen so the

Class label	MIT-BIH Label	Description
0	N	Normal beat
1	V	Premature Ventricular Contraction
2	F	Fusion of ventricular and normal beat
3	/	Paced beat
4	f	Fusion of paced and normal beat
5	*rest*	Other beat types

Table 2: Beat labels used during classification

"image" build-out of the signal can be translated to a matrix of shape 36x36. Since most CNN's also expect a third dimension, usually associated with colour, we also reuse each value in the image matrix 3 times, giving us a final shape of 36x36x3, like a small image. Here each "pixel" is a sample value given in the MIT-BIH database, and each 36x36x3 matrix represents a heartbeat associated with a label. Given the variability in heart rate, and the fact that usually and hopefully more than one heartbeat takes place every 3.6s, it is possible for each image to capture more than 1 heartbeat. To deal with this, we have decided to capture only the values around each label position, relative previous and future labels. Since a label in the dataset is exactly at the peak of a heartbeat, we take half distance (in value indexes) between consecutive beats as the area which we consider to be part of a single heartbeat. In this way, we can accurately capture the P-T-R interval for all the heartbeats. The values outside the P-T-R range are all normalized to 0.

Data segmentation

75 percent of the labelled data was used for training, picked at random from the complete set, was used for training, with the rest used for testing. Then, from the training data, 25 percent of it is used for validation during training. This is necessary to avoid overfitting, a common problem in machine learning where the model is biased towards the data it was trained with, resulting in erroneous predictions for values outside the training set. It is a best practice to keep training and validation data separate from testing data, as the first should be used during the training phase, and the latter while evaluating the performance of a completely trained model.

Classification categories

Although there are many types of heartbeats recorded in the MIT-BIH dataset, they do not appear with the same frequency. Many of them occur only a few times. For example, there are only 106 ventricular escape beats (labelled as E, see table). As such, we have picked for classification only those types of beats that occur at least 1000 times. The heartbeats that do not fall into this category were classified as "other". As a consequence, the models we train are able to classify the following categories:

Models

Here we describe the models we have trained and compressed for this project. They were chosen based on their performance in image classification and versatility for use in other tasks, and by looking at their ImageNET accuracy.

ResNET [5]: here we use ResNET50 and ResNET 101. We chose these particular versions due to their smaller "out of the box" model size compared to the 152 and Inception variants, as well as higher accuracies than their "v2" counterparts. These obtained scores of 95.3 and 95 percent Top 5 accuracy on the ImageNET dataset.

The distinguishing feature of ResNET is the use of *residual learning*, where successive layers are made to fit the residual functions of previous layers plus the desired function. This is done to avoid the vanishing gradient problem, where due to continuous multiplications in large networks, the gradient value gets infinitely smaller.

DenseNet [7]: another type of deep network with a large amount of layers, it proposes a different solution to solve the vanishing gradient: the output of each layer is given as input to all subsequent layers. As such, each layer receives information from all previous layers. Here, we specifically use DenseNet 201, which has achieved 93.66 percent top 5 accuracy on the ImageNet dataset.

MobileNet [6]: build specifically for use on mobile and embedded devices, MobileNet uses depth wise separable convolutions to reduce feature processing, and two hyperparameters, for dimensionality reduction of input/output of each layer and image representation, respectively. We chose this to compare against the compressed model and see if an architecture like this, which was purposefully built for low power devices, can be further compressed

Hyperparameters and metrics

For every model, we have used the following parameters, as they have proven the most efficient, both in speed and classification performance:

- Batch size of 31. As there would be 82987 heartbeats in the training dataset, our batch size choice was quite limited (1, 31, 2677, 82987). We have found 31 to perform well and provide training in a decent amount of time.
- Final activation function: softmax, usually used for categorical classification
- Training epochs: 10 to 20, depending on the model
- Loss function: Categorical Crossentropy
- Optimizer: Adam. Used to provide for flexibility to the descent gradient of the model, by basing it on statistical features of previous gradients.

Additionally, the following metrics have been used to assess the performance of each model, along with their size reduction after compression:

- Categorical Accuracy. The number of correct predictions per overall predictions.
- Precision. The number of correct positive predictions per all positive predictions. Shows how likely the model is to correctly classify a positive instance.
- Recall. The number of correct positive predictions per all actual positive entries. This shows how likely the model is to classify an instance as positive.
- Compression rate (compressed model size / original model size)

5. RESULTS

Uncompressed results

Before diving into the results of the compressed models, we show how these perform in their original state. For

Model	Accuracy	Precision	Recall	Size
ResNet50	0.958	0.961	0.957	274
ResNet101	0.956	0.959	0.951	497
DenseNet	0.944	0.951	0.939	226
MobileNet	0.934	0.936	0.931	30
EfficientNetB0	0.959	0.960	0.957	52

Table 3: Uncompressed models results

Model	Accuracy	Precision	Recall	Size
ResNet50	0.905	0.908	0.903	33/274
ResNet101	0.904	0.907	0.901	44/497
DenseNet	0.906	0.910	0.903	18/226
MobileNet	0.895	0.897	0.894	2.59/30
EfficientNetB0	0.918	0.919	0.916	4.67/52

Table 4: Quantized models results

each model, the results are measured using the metrics described above, together with their size in megabytes.

Here we see all the models performing very well doing heartbeat classification. We also see that increased architecture complexity doesn't also mean increased performance. ResNET 101 performs worse than its smaller ResNEt 50 counterpart, while EfficientNet gives us the best results. This can be caused by feature dilution in larger models given our type of input. As previously mentioned, the actual input is time-dependent and values outside the time-range of a heartbeat are set to 0. As a consequence, larger networks infer relational features between "useful" values and outside-beat values that can result in decreased prediction accuracy compared to smaller architectures, such as EfficientNetB0.

Compressed results

In the following sections, we take a look at the results of the compressed models, based on the used compression methods, and provide a discussion in which we express our findings.

For each model, they were compressed using quantization (training aware, i.e. model is fit after being quantized) and pruning, plus a combination of pruning and clustering. For each of these different compression methods, we have obtained the following results (where the values in the size column represent the space occupied on disk in Megabytes after and before compression):

Quantization

Pruning

Here, the first value in the size column represents the model size after ZIP archiving of the pruned model. Archiving is necessary to take full advantage of pruning, as the weights set to 0 allow archiving algorithms to efficiently compress the model file.

Quantization + pruning (Q+P)

Model	Accuracy	Precision	Recall	Size
ResNet50	0.908	0.911	0.905	25/274
ResNet101	0.903	0.907	0.902	37/497
DenseNet	0.910	0.912	0.906	15/226
MobileNet	0.900	0.899	0.897	1.88/30
EfficientNetB0	0.914	0.915	0.916	3.79/52

Table 5: Pruned models results

Model	Accuracy	Precision	Recall	Size
ResNet50	0.895	0.897	0.890	6.5/274
ResNet101	0.890	0.891	0.889	9.2/497
DenseNet	0.901	0.903	0.899	4.2/226
MobileNet	0.895	0.892	0.891	0.98/30
EfficientNetB0	0.903	0.905	0.905	1.21/52

Table 6: Q+P models results

Similar to the previous section, the size after compression here refers to the size of the model archive.

Discussion

From these results, we can draw the following conclusions. First, EfficientNet seems to perform the best for the classification task of arrhythmia. This was constant along with compressed models of each type, as well as for uncompressed measurements. As elaborated before, this is probably due to reduced feature complexity compared to the other models, as our input image only has a fraction of values providing actual useful information, so fewer relationships are made between empty pixels and useful data. This could also be seen in MobileNet, which performs better than the deeper models like DenseNet and ResNet. Second, by far the model with the most potential for compression is MobileNetV2. When processed using both quantization and pruning, it provides a model archive of under 1MB, while still performing really well in the classification task. The accuracy reduction (together with other metrics) is relatively small. When taken into consideration together with the size reduction, we can say it is worth having the models perform just slightly worse, while we benefit from great savings in storage space. On a similar note, we see how separate, quantization and pruning alone perform just a bit better than Q+P, while again, in our opinion, the reduction in size offsets this cost.

Overall, we have seen the Convolutional Neural Networks can successfully be compressed to small sizes and deployed on a low-powered, cheap device to perform accurate classification of the heartbeat signal. Different compression methods can successfully be used together to retain high accuracies while providing greatly reduced model sizes. Furthermore, we have seen when these methods are applied to compact architectures, like MobileNet and EfficientNet (compact at least in comparison with DenseNet and ResNet), we get can get even better results compared to using compressed complex architectures.

6. CONCLUSION AND FUTURE WORK

In this paper, we have shown how Convolutional Neural Networks can be compressed and successfully deployed on embedded devices to classify different types of heartbeats.

We analysed the MIT-BIH dataset to determine how data is represented and labelled and chose to classify categories based on the frequency of different types of heartbeats. After pre-processing the data and converting it into a representation that CNN's can take as input, we moved on towards model building and compression. For this, we trained 5 different established architectures, compressed them using quantization, pruning and weight clustering.

To answer our main research question, of how can CNN's be compressed to produce accurate classification results on embedded devices, we have answered two sub-questions as follows.

For Sub-RQ1 we have determined that compressed models, close to or even under 1MB can perform ECG signal

classification on an embedded device, here represented by a Raspberry Pi Zero, with around 90 per cent accuracy.

Furthermore, we have shown that this can be done on a very cheap, low-powered device, which at the time of writing costs about 10 euros (in the Netherlands) and is powered by a Broadcom BCM2835 system-on-chip, with a processing speed of 1GHz and a pool of only 512MB of RAM.

Based on the results of the project, we can say that CNN's can successfully be used on cheap, low powered devices for electrocardiogram classification, to classify abnormal heartbeats with around 90 per cent accuracy. This answers our main research question and proves that machine learning on embedded devices can be highly useful in the healthcare domain. Deployment of such systems could greatly reduce costs and the human expertise required to diagnose patients suffering from cardiac issues.

Future improvements over the methods provided here could be done in 2 major areas. First is network architecture design, where a network could be designed specifically for this purpose. Second, more complex compression techniques could be applied. Quantization and pruning could be done at the layer level. Then it could be decided which layers to compress individually to optimize accuracy and model size. Finally, methods from transfer learning, like model distillation, or novel methods like low-rank factorization could be researched in association with those already presented here, for further compression possibilities. Other possible aspects that could be researched include the computation time and viability for real-time detection of different compress models. Based on model size and complexity, inference time could be analysed and a comparison between different models could show which are more suited for time-sensitive critical situations when medics would have no time to wait for their device to process the results.

7. REFERENCES

[1] A. F. M. Agarap. Deep learning using rectified linear units (relu). *arXiv*, 2, 2019.

[2] D. Belo, J. Rodrigues, J. R. Vaz, P. Pezarat-Correia, and H. Gamboa. Biosignals learning and synthesis using deep neural networks. *BioMedical Engineering OnLine*, 16(115), 2017.

[3] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang. Deep learning on mobile and embedded devices:state-of-the-art, challenges, and future directions. *ACM Computing Surveys*, 53(4):84:1–84:37, 2020.

[4] A. Faraone and R. Delgado-Gonzalo. Convolutional-recurrent neural networks on low-power wearable platforms for cardiac arrhythmia detection. *IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2020.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Microsoft Research*, 2015.

[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile visionapplications. *arXiv:1704.04861v1*, 2017.

[7] G. Huang, Z. Liu, S. Ren, and L. van der Maaten. Densely connected convolutional networks. *arXiv:1608.06993v5*, 2016.

[8] S. Kaplan, B. Alper, K. U. E. S. Gunal, S. E. S. Gunal, and M. B. Gulmezoglu. A survey on ecg analysis. *Biomedical Signal Processing and Control*, 43:2016–235, 2018.

[9] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53:5455–5516, 2020.

[10] W. Liu, M. Zhang, Y. Zhang, Y. L. Q. Huang, S. Chang, H. Wang, and J. He. Real-time multilead convolutional neural network for myocardial infarction detection. *IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS*, 22(5):1434–1444, 2018.

[11] D. Mishkin, N. Sergievskiy, and J. Matas. Systematic evaluation of cnnadvances on the imagenet. *arXiv:1606.02228*, 2016.

[12] G. B. Moody and R. G. Mark. The impact of the mit-biharrhythmia database. *IEEE ENGINEERING IN MEDICINE AND BIOLOGY*, 20(3):45–50, 2001.

[13] R. Pilipović, P. Bulić, and V. Risojević. Compression of convolutional neural networks: A short survey. *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2018.

[14] G. Roth and G. Mensah. Global burden of cardiovascular diseases and risk factors, 1990–2019. *Journal of the American College of Cardiology*, 76(5):2982—-3021, 2020.

[15] G. Sarker. A survey on convolution neural networks. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, pages 4–24, 2021.

[16] D. R. Sarvamangala and R. V. Kulkarni. Convolutional neural networks in medical image understanding: a survey. *Evolutionary Intelligence*, 2020.

[17] Z. Wu, ShiruiPan, F. Chen, G. Long, and P. S. Y. Chengqi Zhang. A comprehensive survey on graph neural networks. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 32(1):4–24, 2021.

[18] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi. Convolutional neural networks: an overviewand application in radiology. *Insights into imaging*, 9:611–629, 2018.

[19] Y. Zhang, W. Ding, and C. Liu. Summary of convolutional neural network compression technology. *2019 IEEE International Conference on Unmanned Systems (ICUS)*, 2019.