MPhys Research Project

Machine Learning with Convolutional Neural Networks in Medical Diagnosis

Michael Smith, 200784771

Supervisors: Prof. Ben Varcoe, Mr. John Mooney

Assessor: Dr. Almut Beige

25th April 2017

Convolutional neural networks (CNNs), and their possible uses in electrocardiography and magnetocardiography are explored. A CNN that diagnoses myocardial infarction in electrocardiograms (ECGs) taken from the Physikalisch Technische Bundesanstalt (PTB) diagnostic database is described [1, 2]. This CNN has a diagnosis accuracy of 99.8% in unseen patients, on par with state of the art machine learning methods [3, 4]. CNNs that diagnose magnetocardiograms (MCGs) generated by a magnetocardiograph developed by Mooney et al are described [5]. With a best diagnostic accuracy of $(88 \pm 3)\%$, these CNNs outperform results obtained by Kangwanariyakul et al, Fenici et al, Tantimongcolwat et al, and Wilson [6–9]. Through a moving window method, the diagnostic powers of different segments of ECG and MCG are found. Due to the platform agnosticity afforded by CNN's automated feature extraction, the methods described here can be easily adapted to a wide range of vital signs, such as magnetoencephalography, electroencephalography, and ballistocardiography. The 3D CNN developed is highly portable; two MCG devices with different sensor arrangements can interpolate to an identically shaped output. Therefore, the CNN can be trained on one MCG device, and deployed on another dissimilar one.

Contents

1 Introduction

Machine learning is a subfield of artificial intelligence that concerns itself with making a computer learn without explicit programming. Since its origin in the 1950s, machine learning has rapidly become a part of everyday life [10,11]. Spam filters, face recognition, and Google's reverse image search are all examples of machine learning.

Figure 1: An example of the some handwritten digits found in the MNIST dataset [12]. Each image is 28×28 pixels and monochrome.

Most machine learning algorithms can be sorted into supervised and unsupervised learning methods. In supervised learning, the machine is given a set of data, known as the training set. This set of data is labelled with correct solutions. The algorithm can then adjust itself to best fit the data provided, and give a correct solution as often as possible. For example, hand written numbers, such as those in the MNIST digit database shown in figure 1, could be classified [13]. If the machine is fed many of these digits, each labelled with the correct classification, the algorithm will begin to learn which collections of pixels go together to make the appropriate digit. How this is achieved is determined by the machine learning method in place. Unsupervised learning has the machine sort through unlabelled data in order to find a descriptive function. Using the MNIST example, an unsupervised leaning algorithm would sort through an unlabelled set of images and return with a set of classifications. For example, without being told so, the computer could recognise that a five is different from a six.

1.1 Machine learning and medicine

Artificial intelligence, specifically machine learning, has proven to be a highly useful diagnostic technique [14–18], even outperforming medical professionals in some cases [19]. Medical diagnosis data are usually available already in many hospitals and medical departments in the form of medical records. This data can easily be repurposed for various supervised and unsupervised learning methods. A machine can learn whether a patient is healthy or not by looking at thousands of patient records and discovering patterns that a human cannot find, either due

Classifier	Positive Cases		Negative Cases	
	Reliable (%)	Errors (%)	Reliable (%)	Errors (%)
Physicians	73	3	46	8
Naive Bayes	89	5	83	1
Semi-naive Bayes	91	6	79	2
k-Nearest Neighbours	64	12	80	12
Neural Network	81	11	72	11

Table 1: A comparison of different machine learning techniques for the diagnosis of ishaemic heart disease. It can be seen that the naive Bayes, semi-naive Bayes and neural network algorithms outperformed a human physician in this specific case [20].

to a lack of time or a lack of perspective.

Many different methods of supervised machine learning have been developed and tested on medical data. Kukar's PhD thesis explores the diagnosis of ishaemic heart disease [20]. A comparison between a physician's diagnosis and several different machine learning methods' diagnoses show that a learning method can significantly outperform a human doctor in the diagnosis of disease, as shown in table 1.

1.2 The data

This study's purpose is to develop a machine learning algorithm that diagnoses heart disease. In order to choose which algorithm to use, the data sets that the learning method will be trained on are examined.

1.2.1 Electrocardiography (ECG) scans

An electrocardiogram records the electrical activity of the heart. In a standard ECG, 10 electrodes are placed on the skin, and dynamically measure small electrical charges caused by the depolarisation and repolarisation of various parts of the heart during a typical beat.

The ECG data used is taken from the PTB diagnostic ECG database [1, 2]. This database provides a standard 12 lead ECG, as well as a three lead Frank vectorcardiogram (VCG) [21]. A single 12 lead ECG scan can be represented as a two dimensional array with the first axis across the different leads, and the second in the time dimension. The three Frank leads could be similarly interpreted. Each index in this array contains a numerical value representing

the amplitude of the ECG output. Therefore, a scan can be interpreted as a heatmap image, which can be fed into a machine learning method. An example of this heatmap is shown in figure 9a.

1.2.2 Magnetocardiography (MCG) scans

Magnetocardiograms use sensitive magnetometers, such as the superconducting quantum interference device (SQUID), to measure the magnetic fields produced by electrical activity of the heart. Since these magnetic fields are caused by the same electrical charges that are picked up in ECG, the MCG and ECG are similarly shaped.

The MCG data was obtained through the use of a magnetocardiography (MCG) device as described in "A Portable Diagnostic Device for Cardiac Magnetic Field Mapping" [5]. This device consists of 15 hexagonally arranged SQUIDs, each outputting a channel of data, and was run a total of 400 times on a group comprised of 60 healthy and 60 unhealthy participants. The MCG device recorded each participant for 10 minutes, capturing around 600 heatbeats, which were averaged to create a single cardiac cycle of one second at 2 kHz, creating 2000 datapoints per channel [9].

This data has been interpreted in two different ways. The first is similar to the approach used for the ECG data; the output from each coil is stacked in an array with one dimension across coils, and one in time, resulting in an array with shape 15×2000. Using this interpretation results in a dataset of 400 heatmap images with 30000 data bins (pixels) each. Figure 12 shows an example.

A second way of interpreting the data takes into account the position of the 15 SQUID coils in the MCG device. The outputs of the coils are mapped onto a 2D plane; a magnetic field map of the patients' chests. A third dimension is then added to show how the magnetic field map changes in time. Still frames of this dataset are shown in figure 13. This interpretation results in a dataset of 400 arrays with shape $17 \times 19 \times 2000$, or 642000 data bins.

Similarly to the MNIST and ECG dataset, each bin in the 2D and 3D MCG interpretations holds a number that can be taken as an input to the machine learning method in use.

As well as a standard ECG-like signal, the MCG device picked up noise generated by movement of the patients' beds. This noise, along with the standard signal, can be seen in figures 12, and 15. The bed noise is due to mechanical movement of the heart causing the bed to oscillate. Therefore, this noise is a type of ballistocardiogram (BCG), with the MCG device acting as a pickup.

Ballistocardiograms measure ballistic forces caused by blood flow into and out of the heart. This blood movement exerts a force on the body, causing minute movements that can be picked up with an accelerometer, or other sensitive instrument. In this case, the minute movements have been picked up by the MCG device.

Both the ECG and MCG data can be interpreted as an image. High performing image recognition research is dominated by deep neural networks in general, and deep convolutional neural networks (CNNs) in particular [22–25]. Also, since CNNs convolve over the entire image, found features would not need to be in identical places in different scans, making the method generic, and transferable. CNNs also extract features from given data automatically, bypassing the need for manual feature extraction. Therefore, deep CNNs (section 1.3.1) could be an appropriate method to analyse the given data.

1.3 Neural networks

Understanding how neural networks learn requires first understanding how the individual neurons learn. One such neuron is the perceptron, developed by Frank Rosenblatt in the 1950s [26]. This neuron takes a number of binary inputs x_j and multiplies them with a weight w_j[1]. If the sum of these weights multiplied by the inputs is larger than a threshold value, $-b$, then the perceptron "fires" or outputs 1. Algebraically the perceptron can be described by (1).

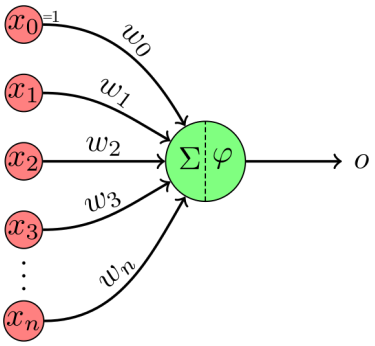


Figure 2: View of a single generic neuron with bias $x_0 = b = 1$, inputs $x_1, x_2, ..., x_n$, and weights $w_1, w_2, ..., w_n$. The output function is φ [27].

[1]that signifies the importance of the input; a larger weight means that the input is more important

$$output = \begin{cases} 1, & \text{if} \quad \sum_j w_j x_j + b > 0, \\ 0, & \text{if} \quad \sum_j w_j x_j + b \leq 0. \end{cases} \tag{1}$$

Many other artificial neurons are in use. A major differing factor is the output function, shown in figure 2 as $\varphi(\Sigma w_i x_i)$. Another neuron that has found significant use is the rectified linear unit (ReLU), which has an output function $\varphi(z) = \max(0, z)$. The ReLU's and perceptron's output functions are shown graphically in figure 3. ReLU neurons are used in some of the best performing neural networks [28, 29]. This is due to the increase in learning speed found with their use and their relative ease of computation, when compared to other neurons [30]. ReLU neurons are prone to "dying", or outputting 0, if they are fed a negative input. This is a solved with the use of a "leaky" ReLU, one that outputs a small value for inputs below 0.

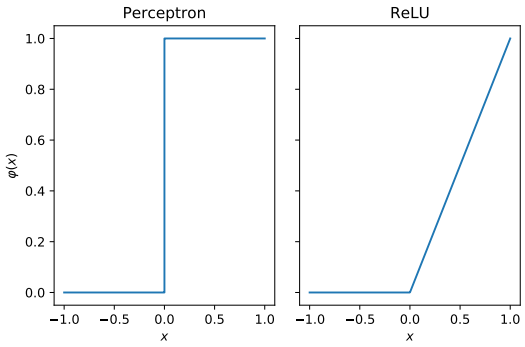


Figure 3: Output functions of the perceptron (left), and ReLU (right).

Grouping many artificial neurons together may result in something resembling figure 4. Figure 4 shows a three layer neural network, with one input layer, one hidden[2] layer, and one output layer.

Returning to the MNIST example data can further clarify how a neural network learns. Suppose that the goal is for the network to learn what a "5" is. A neuron can be assigned to each pixel on the 28×28 pixel grid, with its input being the numerical shade of the pixel. A completely white pixel would be a "0", a completely black pixel would be a "1", and shades of grey would be somewhere in between. As the neural network learns what is and is not a five from the training data, its weights and biases adjust in order to fit the given data. At this point the neural network will predict that the input is a five when it is presented with the image of the five. This adjustment of weights and biases is done by

[2]not input nor output

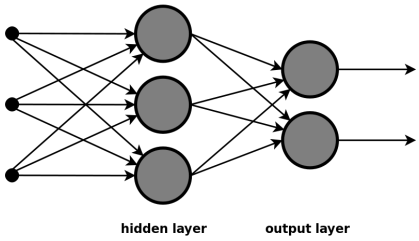


Figure 4: View of a simple shallow multilayered neural network [31]. A deep neural network would have multiple hidden layers.

defining a cost function, then attempting to minimise it. The cost function can be thought of as a function with a global minimum. Once the neural network's cost function is equal to this global minimum the network has found the optimal values of its weights and biases for classifying the training data.

One such cost function is the quadratic cost. Algebraically it is defined as

$$C(w,b) \equiv \frac{1}{2n} \sum_x ||y(x) - a||^2, \tag{2}$$

where w is the sum of all weights, b is the sum of all biases $y(x)$ is the desired output and a is the current output.

A second commonly used cost function is the cross-entropy cost. This is algebraically defined as

$$C(w,b) \equiv \frac{-1}{n} \sum_x \sum_j [y_j \ln a_j^L + (1-y_j) \ln(1-a_j^L)], \tag{3}$$

where n is the number of items in the training data, x is the training input index, j is the output neuron index, $y_1, y_2, y_3, ...$ are the desired outputs, and $a_1^L, a_2^L, a_3^L, ...$ are the actual outputs. The cross-entropy cost function is the preferred cost function to use in classification problems, such as medical diagnosis, where the patient is either "healthy" or "unhealthy" [32]. Such classification problems lend themselves to the use of a softmax layer as the final layer in the neural network. This softmax layer outputs a normalised probability distribution representing each desired outcome. There are of course other cost functions, all of which are minimised during training.

The *de facto* way of minimising the cost function is through stochastic gradient descent (SGD). This descent algorithm works by calculating the gradient of the cost function via a technique called backpropagation [33,34]. Once the gradient is found, the weights and biases of the network are adjusted to follow this gradient "down", and minimise the cost function. This process of finding the gradient and

adjusting the weights and biases continues until the learning rate slows to a stop. Since SGD can only follow the gradient of the cost function down, it often gets trapped in local minima. This can be solved with the implementation of batch training, where the gradient is calculated after a set number of scans have been processed. It can also be solved with the use of a momentum coefficient. The coefficient causes the optimiser to overshoot its target by an amount proportional to the calculated gradient. In an ideal case the optimiser would also overshoot local minima. The ADAM optimiser uses a momentum coefficient, and so would be a good choice of optimiser for the developed CNN [35].

Neural networks do not scale well to high dimensional datasets. The MNIST data neural network that has been used as an example has $28 \times 28 = 784$ pixel inputs, so that a fully connected neuron in the first hidden layer would have 784 weights. In the same way, a neuron in the first hidden layer of a neural network designed for the 2D data collected in "A Portable Diagnostic Device for Cardiac Magnetic Field Mapping" would have $2000 \times 15 = 30,000$ weights [5]. Hughes' phenomenon states that as the dimensionality of a neural network (or any machine learning method) increases, the predictive power of the network decreases [36]. This dimensionality issue can be reduced with the use of a covolutional neural network.

1.3.1 Convolutional neural networks

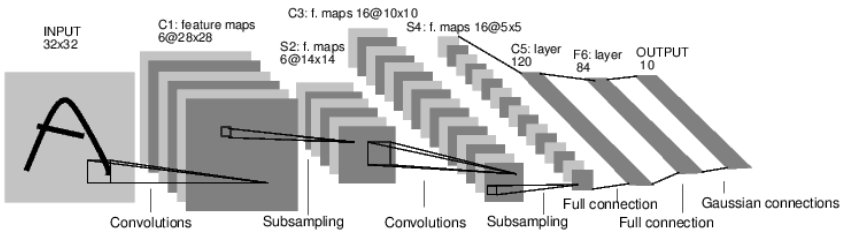


Figure 5: LeNet-5, an example of a convolutional neural network with convolution and pooling (subsampling) layers [24].

Unlike the neural network described previously, a CNN does not entirely consist of fully-connected layers, where each neuron is connected to every neuron in the next and previous layer. The two major layers used are the convolutional and pooling layers, as seen in figure 5.

A convolutional layer can be analogised as a "feature filter". Figure 6 shows a mathematical example of a this feature filter, or local receptive field, operating on an input layer. In figure 6 the local receptive field is represented by a matrix,

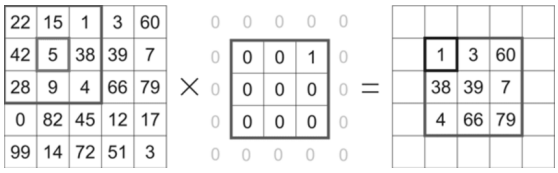


Figure 6: A convolution filter operates on an input data set [37].

whereas in the CNN the local receptive field is calculated via a set of weights. The field is then moved a certain number of neurons, the stride length, and a new output is calculated. This continues until the local receptive field reaches the final neuron of the input layer. As seen in figure 5 there can be many layers, or feature maps, reading the input. Each feature map has a unique bias and a set of weights representing the local receptive field. For example, if the local receptive field is made up of $5 \times 5 = 25$ neurons, then there would be a weight for each of those neurons.

A pooling, or subsampling layer is also used, often immediately after a convolutional layer. A pooling layer summarises the output of a previous layer. For example, a 2×2 max pooling layer would take a 2×2 group of neurons in the previous layer and output the maximum neuron value.

The final layer of a CNN is often a fully connected layer. This connects every neuron in the previous layer to the output neurons.

In LeNet-5 the output layer is a softmax layer, representing the probability of the input being any one of the ten digits found in the MNIST data set. For the data found in "A Portable Diagnostic Device for Cardiac Magnetic Field Mapping" the final layer will be a softmax layer comprised of two neurons, one for "healthy" and one for "unhealthy". The output of these layers can be interpreted as a "healthiness rating" of the heart. Since a softmax's layer's outputs always sum to 1, either neuron's output can be interpreted as the heart's relative "healthiness score".

1.3.2 The problem of overfitting

A model with enough free parameters can be used to fit nearly any data set that it is given. Such a model could learn the data set very well, but may not be very good at generalising to data that it has not been trained with. This is known as overfitting and is shown in figure 7. The CNN LeNet-5 has tens of thousands of free parameters in the form of its neurons' weights and biases. Therefore, overfitting can be a major problem when training deep CNNs.

Overfitting can be detected via a validation technique, where the network

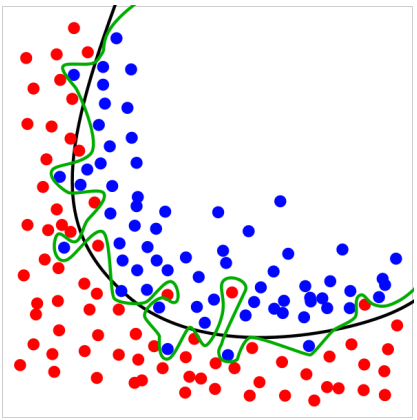


Figure 7: The green line is an example of an overfitted model which has learnt the noise present in the data. The black line shows a well-fitted model and is more likely to accurately classify incoming data [38].

is periodically given a set of validation data to classify. If the classification accuracy of the validation data remains static, and the cost function continues to decrease, it can be said that the network is now overfitting. It is learning noise in the training data.

Regularisation techniques are used to stem overfitting [34]. One such technique is L2-regularisation. In L2-regularisation a term is added to the cost function of the neural network. Equation 4 shows a L2-regularised cost C.

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2 \tag{4}$$

where C_0 is the unregularised cost function (such as the quadratic cost), λ is the regularisation parameter, n is the size of the training data set, and w is a weight. The added expression causes the network to favour small weights if λ is large, and a small cost function if λ is small. A CNN with small weights would have a lower complexity compared to a CNN with large weights. This results in a simpler function fitting the training data.

Dropout is another regularisation technique [39]. In a dropout layer a certain percentage of hidden neurons are ignored at random. The network then trains as normal. After a set amount of cycles a different set of neurons is ignored. This has the same effect as taking the average result from many similar neural networks, without the sizeable computational overhead.

2 Experimental methods

All CNNs were developed in TensorFlow and can be found on github[3].

2.1 A CNN to categorise MNIST data

To find a convolutional neural net (CNN) that works in principle, MNIST numbers were used as a preliminary dataset [13]. An example of the MNIST dataset is shown in figure 1. A 1D convolutional net was written using TensorFlow with two convolution-pooling layers and one 1024 neuron dense layer, along with an output layer. The first convolutional layer has 32 feature masks with shape 1×6, and the second has 64 feature masks with shape 1×6. Each convolutional layer is followed by a 2 wide max pooling layer. It uses ReLU neurons, and categorical cross-entropy loss with the ADAM optimiser at a 10^{-5} learning rate [35]. This neural net learns to 90% accuracy after 10 epochs of 10,000 letters.

2.2 Categorising ECG scans

The CNN developed from the MNIST data does not work well with the MCG data, and so an intermediate dataset was used to develop a better performing network. This dataset is the PTB diagnostic ECG database [1, 2]. 39 healthy patients and 33 patients diagnosed with myocardial infarction were selected to train the CNN, and 41 different healthy patients and 33 different unhealthy patients are used to test the CNN's diagnostic capabilities. The data scraped from the PTB database is the output of a standard 12 lead ECG, plus three Frank VCG leads that can be derived from the standard ECG data [21, 40]. This data is continuous and so a method of extracting a single "heartbeat" is required. To do this the continuous data is differentiated and the absolute values are taken. Taking the derivative centres the scan on a baseline and negates any random walking present in the raw data. Taking the absolute value allows negative gradients in the QRS complex to be found by a peak finder. A simple peak finder is then used to find the locations of the QRS complexes, and this is taken as the centre of the beat. Finally, the ECG data is returned to its initial state, with each heartbeat cut out of the continuous data. The heartbeats are then compressed into 20,000 scans with the use of averaging. The scans from healthy controls are labelled "0", and those from patients diagnosed with myocardial infarction labelled "1".

The diagnostic power of different parts of the ECG was explored, along with the diagnostic power of individual leads, concatenated VCG, concatenated

[3] `https://github.com/Smith42/neuralnet-mcg`

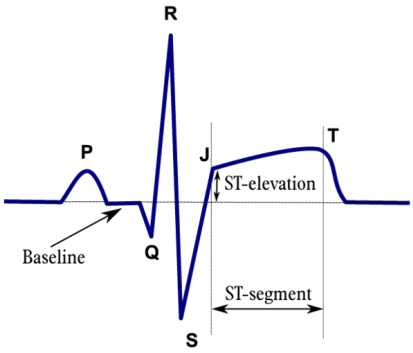


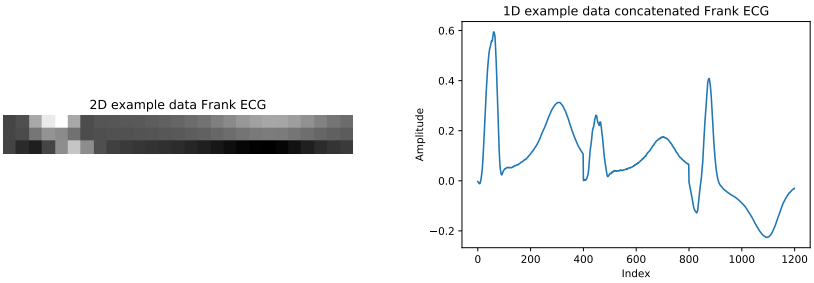
Figure 8: An example of a labelled ECG output [41]. This ECG has an ST elevation, symptomatic of myocardial infarction [42].

standard ECG leads, and 2D arrays of these leads.

To find the diagnostic power of different parts of the ECG, a moving window is applied to the set of training, testing, and validation scans. This window has a width of 200 indices in time. The window selects a segment of the ECGs to feed into the CNN. after the CNN is trained, tested, and validated with a window, the window is moved 25 indices further along in time. This process is repeated until the window reaches the end of the ECG scans. The results from this experiment are shown in figure 14.

It was found that the ST-section of the ECG is most diagnostic, and so this section was used to train the CNN. A CNN that categorises unseen data to an accuracy of more than 99% was heuristically developed for this data. This CNN is composed of two convolution-pooling layers, followed by two 1024 neuron dense layers, and an output layer. The first convolution has 32 feature maps of shape 1×5, and the second has 64 feature maps of shape 1×5. Each convolution layer is followed by a 2 wide max pooling layer. Leaky ReLU neurons are used, with categorical cross-entropy loss, and the ADAM optimiser at a learning rate of 10^{-5}. To combat overfitting, the dense layers are regularised with an L2 regularisation method at a weight decay of 10^{-3}, and are subject to dropout at a rate of 0.5. If the ECG leads are arranged in a 2D array, shown in figure 9, the convolution layer feature maps have shape 5×5, and the pooling have shape 2×2. The results from this CNN are shown in table 2.

Once a CNN with a good unseen accuracy rate was found for categorising the ECG scans, the 1D CNN was adapted for the MCG data.



(a) (b)

Figure 9: Example inputs for the 2D (figure 9a) and 1D (figure 9b) CNNs developed for ECG. This example is of the three Frank VCG leads "vx", "vy", and "vz". In the 2D example the Frank leads are stacked in the y-axis, with their indexes in the x-axis and their amplitudes shown in the heatmap. The 2D example has been subsampled at every 15th point in time for a better visualisation. The 1D example has the Frank leads concatenated, so that they make one long input. The QRS complex can be seen on the far left, and the ST-segment can be seen on the right in each Frank lead.

2.3 Categorising MCG scans

The MCG data contains far fewer patients than the ECG data. Therefore a k-fold cross validation method is used, so that a more representative accuracy, specificity, and sensitivity is obtained.

2.3.1 K-fold cross validation

In small datasets, it is imperative that the majority of the data is used for training, so that the model generated is as general as possible. However, maximising the data used for training means minimising the data held back for testing the trained model. The smaller a testing set is, the less likely it is to be representative of the data as a whole. K-fold cross validation is a validation method that maximises the amount of data available for training a model, while not impacting the data available for testing the completed model.

In a k-fold cross validation, a dataset is split into k chunks. The model is then trained on $k-1$ of these chunks, and the final chunk is held back, for testing the trained model. This training is run k times, with the testing chunk being different each time. The performance of the models found from the testing chunks can then be averaged so that a more representative value is found. Figure 10 shows an example of k-fold cross validation (where $k = 4$) carried out.

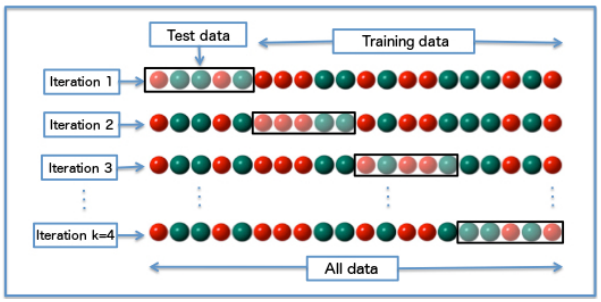


Figure 10: An example of a 4 fold cross validation method being carried out on a dataset. Grouped sections are held back from training, and are only used for testing the data after training has been completed [43].

2.3.2 The receiver operating characteristic (ROC) curve

In the case of heart disease, a heart's healthiness is a continuous variable, and classification requires an arbitrary cut off point between hearts labelled as healthy (0), and hearts labelled as unhealthy (1). In the accuracy metric, the closeness of the patients to this cut off is lost. However, the confidence the algorithm has in its classifications can be represented in a receiver operating characteristic (ROC) curve, and the area under the ROC curve (AUC). In the medical case, where it is preferable to have a high sensitivity and low specificity, it is possible to see what loss in specificity is required for a 100% sensitivity by looking at the corresponding ROC curve.

2.3.3 1D and 2D CNNs for MCGs

Since the model described in section 2.2 works very well with 1D ECG data, scans from individual MCG coils were normalised, and fed into a similar CNN. Some examples of this data shown in figure 11.

As with the ECG CNN, the hyperparameters are refined heuristically, finding that a learning rate of 10^{-4} works best.

This neural net can be trained over a large number of epochs, with a validation set of 20% of the training data, to find the point where it has completely learnt the model. The validation accuracy, and therefore learning, stops increasing at around the 25 epoch mark.

After 25 epochs of training this data arrangement achieves an accuracy of around 81% on validation data, with a specificity of 77%, and a sensitivity of 84%, averaged over a ten fold cross validation. Full results are in table 3.

The 1D method ignores positional data on the coils, so a different arrangement

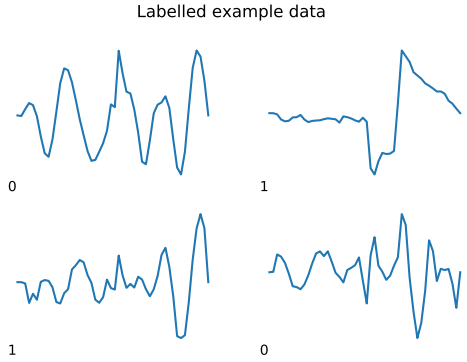


Figure 11: Example of data fed into the 1D CNN. The QRS complex can be seen in the centre of the scan. Noise from the movement from the metal bed can be seen on the right of the QRS complex. Healthy controls are labelled as zero, patients are labelled as one.

of the MCG scans was examined (figure 12). This new arrangement is similar to the 2D ECG arrangement in figure 9, and gives the CNN information about the positions of the individual coils, albeit in an arbitrary way.

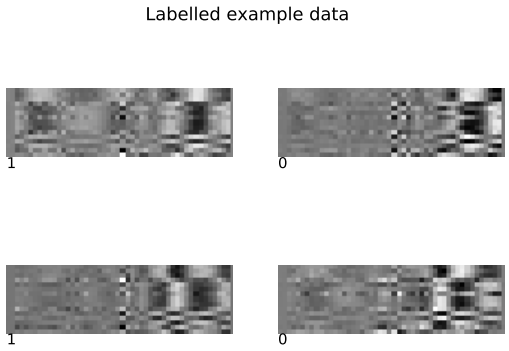


Figure 12: Example of data fed into the 2D CNN. The QRS complex can be seen in the centre of the image, and the bed noise can be seen on its right. Healthy controls are labelled as zero, patients are labelled as one.

The CNN used for the 1D data was reused for the 2D data. The convolution layers have a mask of shape 15×5, and the pooling layers have a mask of shape 2×2. As in the 1D CNN, this neural net is trained to find the point where it had completely learnt the model. This point is around the 40 epoch mark.

After 40 epochs of training this CNN achieves an accuracy of around 88% on validation data, averaged over a 10 fold cross validation. Full results are in table 3.

2.3.4 3D CNN for MCGs

Up to this point, the MCG data has been interpreted in the same way as ECG scans. However, the data drawn from the MCG device can also be interpreted as a moving two-dimensional heatmap of the patients' chest. Example still frames of this are shown in figure 13.

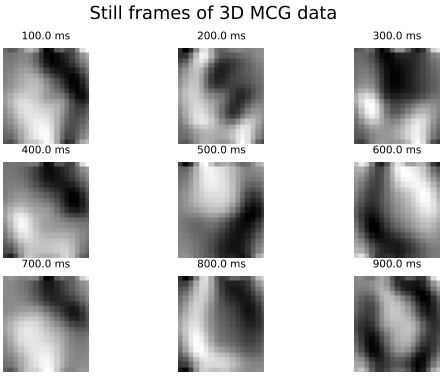


Figure 13: Still frames of data fed into the 3D CNN. Time axis is above each frame.

With these two spacial dimensions, and a third dimension in time, a 3D convolution can be used to diagnose the patient.

As with the ECG data, different sections of the MCG are tested for their diagnostic power. The results of this are shown in figure 15. The neural net used is similar to that used for the 1D MCG data, with the convolution layers having a mask of size $5 \times 5 \times 5$, and the pooling layers having a mask of $2 \times 2 \times 2$. Due to computational limitations, the 3D MCG data had to be subsampled every fourth unit in time, and only the most diagnostic part of the scans were used (400 ms and above, as shown in figure 15).

Again, as in the 1D and 2D CNN, the neural network is trained to find the point where it has completely learnt the model. This point is around the 60th epoch. A 10 fold cross validation is carried out on the 3D data, resulting in an accuracy of around 88%. Again, full results are found in table 3.

3 Results

The main findings are presented here.

3.1 ECG data

Table 2 summarises the ECG CNN results, and figure 14 visualises the diagnosticity of different sections of an ECG scan.

	Test data			Validation data		
Input data	Spec	Sens	AUC	Spec	Sens	AUC
Frank concatenated	99.9	100	1.00	99.9	99.6	1.00
2D Frank	99.7	100	1.00	99.3	99.0	1.00
Standard concatenated	99.9	100	1.00	100.0	96.4	0.999
2D standard	99.9	99.6	1.00	99.9	92.4	0.970

Table 2: Performance of ECG scan CNNs at one epoch of 16,000 scans. 4,000 scans from the training set are kept back for testing. 20,000 scans of patients the neural network has not seen are used for validation. Spec is specificity, and Sens is sensitivity. Both specificity and sensitivity are in percentages. AUC is the area under the receiver operating characteristic curve. All scans are of the ST-segment of their respective leads. The means of three runs are shown.

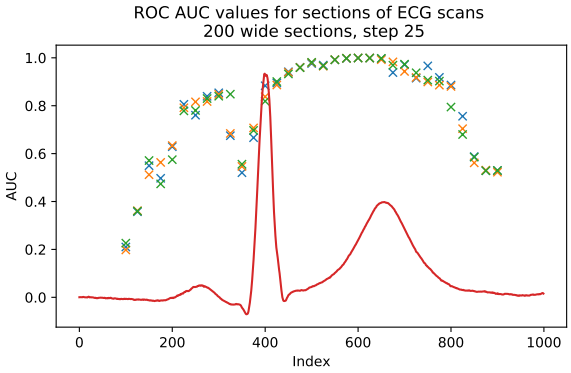


Figure 14: ROC AUC values for sections of Frank lead "vx" superimposed over an example ECG "vx" scan. Three repeats were carried out on a dataset of 20000 scans. The AUC values were calculated from an unseen dataset of 20000 scans. It can be seen that the ST-segment carries the majority of the diagnosticity of the scan.

3.2 MCG data

Table 3 summarises the MCG CNN results. Figure 15 visualises the diagnosticity of different segments of a MCG scan. Figure 16 shows the ROC curves for the 1D, 2D, and 3D MCG CNNs.

CNN	Accuracy (%)	Specificity (%)	Sensitivity (%)	ROC AUC
1D	81 ± 3	77 ± 2	84 ± 3	0.88 ± 0.01
2D	88 ± 3	87 ± 3	89 ± 3	0.93 ± 0.02
3D	88 ± 3	86 ± 4	90 ± 3	0.94 ± 0.02

Table 3: Comparison of diagnostic power of the three different CNN techniques used.

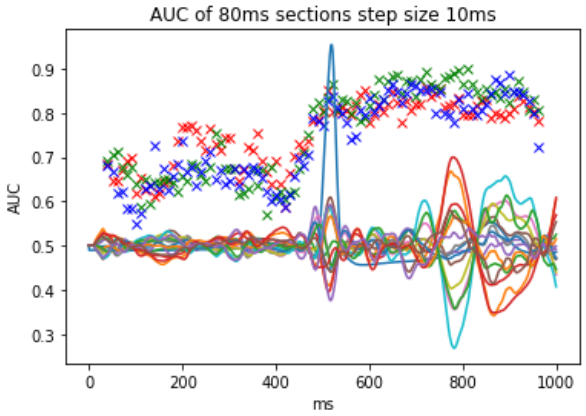


Figure 15: ROC AUC values for sections in the 3D MCG data superimposed on all 15 MCG coil scans. 4 fold cross validation was carried out and is represented by different coloured crosses. The subsampling is every 2 in time. It can be seen that the majority of the diagnosticity is contained in the region past the QRS complex.

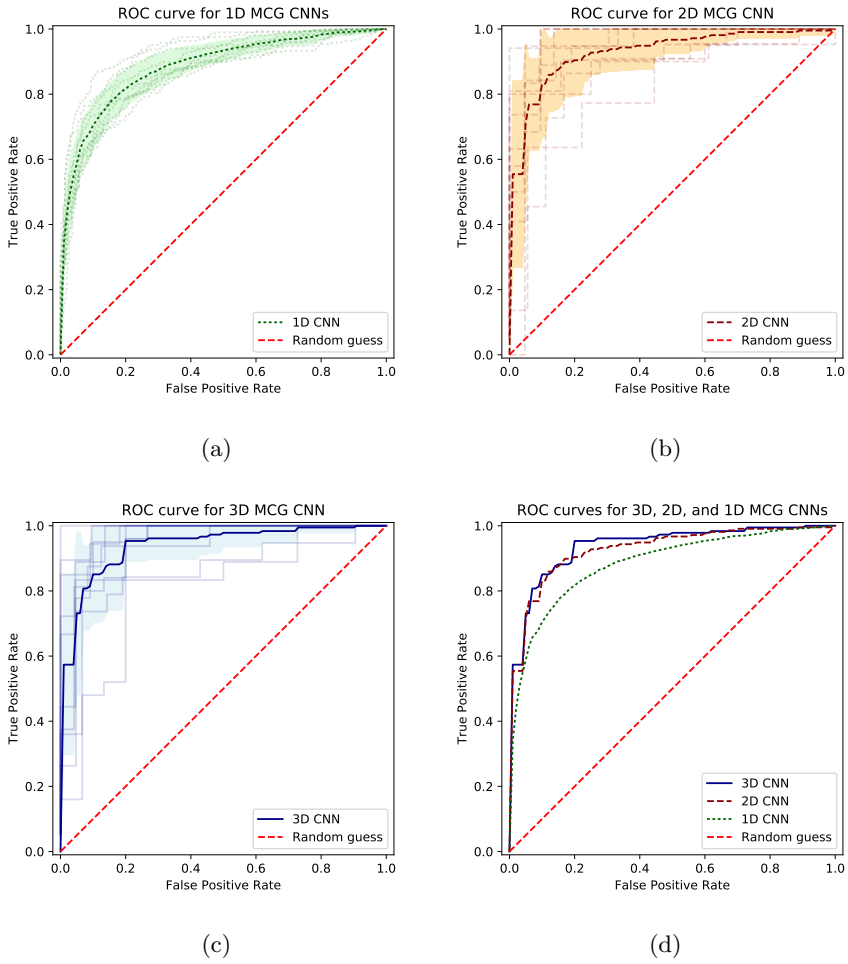


Figure 16: ROC curves for the MCG 1D (figure 16a), 2D (figure 16b), and 3D (figure 16c) CNNs. Darker lines are mean ROC curves. Shaded areas are the standard errors. Lighter lines are ROC curves from individual folds. All three CNNs were trained in a 10-fold cross validation. The means of all three ROC curves can be seen in figure 16d.

4 Discussion

4.1 ECG results

The CNN developed for ECGs had an accuracy of close to 100% on the three VCG Frank leads. This accuracy is on par, and in most cases surpasses, state of the art machine learning methods [3, 4]. A major advantage of the method described in this paper is the lack of a need for manual feature extraction, making the method highly portable. This allows the method to be used with many similar signals, such as the MCG, without manually redefining features.

Worse average performance of the 12 lead ECG, compared to the 3 lead frank VCG could be due to incomplete hyperparameter optimisation, as the hyperparameters were shared between both data representations. Training the 12-lead ECG CNN for longer may also result in a better average accuracy. Increasing the number of features in each convolutional layer, taking into account the added complexity of the 12 lead ECG, may also achieve better results.

In the case of myocardial infarction, the majority of the diagnosticity of an ECG scan was found to be contained within the ST segment (figure 14). This is in agreement with the literature [44]. As with the general CNN, the technique for finding the most diagnostic parts of a scan is highly portable.

4.2 MCG results

The MCG CNN's accuracies of 88% for the 2D and 3D data outperforms those found by Kangwanariyakul et al in 2010 [6]. Their backpropagation neural network (BPNN), Bayesian neural network (BNN), and probabalistic neural network (PNN) had accuracies of 78.4%, 78.4%, and 70.6% respectively. Their best performing non-neural network algorithm (a linear support vector machine (SVM)) had an accuracy of 74.5%. Fenici et al's direct kernel partial least squares (DK-PLS) had an accuracy of 80% [7]. Tantimongcolwat et al's BNN and direct kernel self organising map (DK-SOM) had accuracies of 74.5% and 80.4% respectively [8]. A more representative comparison can be made with Wilson's SVM, since it was trained and tested on identical data [9]. The SVM achieved an accuracy of 79.3%, which the CNNs described in this paper also outperform.

Figure 15 shows that the most diagnostic parts of the MCG scans are past the QRS complex. This is largely in agreement with figure 14. However, the AUCs calculated before the MCG QRS complex are relatively lower than those in the MCG figure. This is likely due to the environmental noise in the MCGs drowning out the P wave. The T wave also cannot be seen in figure 15, as it

is drowned out by vibrations caused by the patients' hearts moving the beds. It is therefore surprising that the majority of the diagnosticity is contained in the section following the QRS complex. This could be due to remnants of ST-elevation that are not visible, but are being picked up by the CNN. It could also be due to a high diagnosticity being contained in the BCG-like signal caused by the movement of the bed. It would be interesting to isolate the bed noise to find how diagnostic the BCG-like signal is. However, this cannot be done with the data at hand. A device that only outputs a BCG signal[4] could be fed into the 1D CNN to test this.

Figure 16 shows the ROC curves for the 1D, 2D, and 3D MCG CNNs. It can be seen that the 1D ROC AUC is less than that of the 2D and 3D ROC AUCs. This is due to the lack of positional data contained in the 1D MCGs. There is a lower standard error in the 1D data, because it allows a higher number of samples to be processed, as each coil is processed individually. This reduces the variance of the k-folds, as more samples are present to be tested in each k-fold. The 2D and 3D ROCs are very similar, and are well within each other's uncertainties. Both the 2D and 3D CNNs performed similarly, suggesting that the 2D arrangement is not as arbitrary as it first seems. It has to be remembered that, due to computational restrictions, the 3D CNN is subsampled every fourth point in time, and is only taken from 400 ms onwards, whereas the 2D CNN was not altered. This may have affected the results, and the 3D CNN may perform better with the unabridged data.

4.3 Future work

Due to the nature of manual optimisation, it is unlikely that the optimum hyperparameters have been found for the CNNs described in this paper. Further optimisation may be achieved by using an automated hyperparameter search, which would be possible with more time, or more computational power [46, 47].

It would be interesting to train the 3D MCG CNN on data with a higher spatial resolution, to see if a higher accuracy can be eked from this model.

A major advantage of the 3D CNN developed here is its portability between different MCG machines. It could be trained on a different MCG machine to the testing machine, as long as the output "video" has the same shape. It could also be trained on other 3D data, or even adapted to 4D data such as moving 3D nuclear magnetic resonance (NMR) scans, although in these cases the weights and biases would not be transferable.

[4] such as that proposed by Shin and Park [45]

The machine learning methods described in this paper are all supervised learning algorithms. Running an unsupervised learning algorithm, such as a generative adversarial network or a T-SNE, may provide an unique insight into the data [48].

The reason why the CNNs arrive at the filters that they find is rather opaque. It may be insightful to try some manually defined filters (such as edge detection) to see how well they perform. Using larger filters may also make this process more transparent, as a pattern in the filter is more discernible.

Overfitting could be occurring in the trained CNNs. It is difficult to see if the CNNs are overfitting when the pool of patients is so low. Adding more patients to the data, or artificially expanding the data would help with this [34].

5 Conclusion

A CNN that diagnoses myocardial infarction in electrocardiograms (ECGs) taken from the PTB diagnostic database has been developed, and has a diagnosis accuracy of 99.8% in unseen patients, on par with state of the art machine learning methods [3, 4]. This CNN has been adapted to diagnose magnetocardiograms (MCGs) generated by a magnetocardiograph developed by Mooney et al [5]. With a best diagnostic accuracy of $(88 \pm 3)\%$, the MCG CNNs outperform results obtained by Kangwanariyakul et al, Fenici et al, Tantimongcolwat et al, and Wilson [6–9]. Through a moving window method, the diagnostic powers of different segments of ECG and MCG have been found. Due to the platform agnosticity afforded by the lack of a need of manual feature extraction, the methods described here can be easily adapted to a wide range of vital signs, such as magnetoencephalography, electroencephalography, and ballistocardiography. The 3D CNN developed is highly portable; two MCG devices with different sensor arrangements can interpolate to an identically shaped output. Therefore, the CNN can even be trained on one MCG device, and deployed on another dissimilar one.

6 Acknowledgements

I would like to thank the Experimental Quantum Information group at the University of Leeds. I would also like to thank my supervisors, Prof. Ben Varcoe, and Mr. John Mooney, for their guidance and expertise (and use of John's GPU!).

References

[1] Bousseljot R D, Kreiseler D and Schnabel A 2004 The ptb diagnostic ecg database URL https://doi.org/10.13026/C28C71

[2] Bousseljot R, Kreiseler D and Schnabel A 2009 *Biomedizinische Technik/Biomedical Engineering* 317–318 URL https://doi.org/10.1515%2Fbmte.1995.40.s1.317

[3] Salem A B M, Revett K and El-Dahshan E S A 2009 *2009 International Multiconference on Computer Science and Information Technology* (IEEE) URL https://doi.org/10.1109%2Fimcsit.2009.5352689

[4] Jambukia S H, Dabhi V K and Prajapati H B 2015 *2015 International Conference on Advances in Computer Engineering and Applications* (IEEE) URL https://doi.org/10.1109%2Ficacea.2015.7164783

[5] Mooney J W, Ghasemi-Roudsari S, Banham E R, Symonds C, Pawlowski N and Varcoe B T H 2017 *Biomedical Physics & Engineering Express* **3** 015008 URL https://doi.org/10.1088%2F2057-1976%2F3%2F1%2F015008

[6] Kangwanariyakul Y, Naenna T, Nantasenamat C and Tantimongcolwat T 2010 Data mining of magnetocardiograms for prediction of ischemic heart disease URL https://doi.org/10.17877/DE290R-15805

[7] Fenici R, Brisinda D, Meloni A M, Sternickel K and Fenici P 2005 *Functional Imaging and Modeling of the Heart* (Springer Berlin Heidelberg) pp 143–152 URL https://doi.org/10.1007%2F11494621_15

[8] Tantimongcolwat T, Naenna T, Isarankura-Na-Ayudhya C, Embrechts M J and Prachayasittikul V 2008 *Computers in Biology and Medicine* **38** 817–825 URL https://doi.org/10.1016%2Fj.compbiomed.2008.04.009

[9] Wilson D 2016 *Optimising a Support Vector Classifier for use with a Magnetocardiogram* Master's thesis University of Leeds

[10] Turing A M 1950 *Mind* **59** 433–460

[11] McCarthy J and Feigenbaum E 1990 *AI Magazine* **11**(3) 10–11

[12] Alphabet Inc Example mnist images https://www.tensorflow.org/versions/r0.8/tutorials/mnist/download/index.html accessed 2016-11-08

[13] LeCun Y, Cortes C and Burges C J C The mnist database of handwritten digits http://yann.lecun.com/exdb/mnist/ accessed 2016-11-08

[14] Polat K and Güneş S 2007 *Digital Signal Processing* **17** 694–701 URL https://doi.org/10.1016%2Fj.dsp.2006.10.008

[15] Medjahed S, Saadi T and Benyettou A 2015 *International Journal of Intelligent Systems and Applications* **7** 1–7 URL https://doi.org/10.5815%2Fijisa.2015.05.01

[16] Huertas-Fernández I, García-Gómez F J, García-Solís D, Benítez-Rivero S, Marín-Oyaga V A, Jesús S, Cáceres-Redondo M T, Lojo J A, Martín-Rodríguez J F, Carrillo F and Mir P 2014 *European Journal of Nuclear Medicine and Molecular Imaging* **42** 112–119 URL https://doi.org/10.1007%2Fs00259-014-2882-8

[17] Salvatore C, Cerasa A, Battista P, Gilardi M C, Quattrone A and Castiglioni I 2015 *Frontiers in Neuroscience* **9** URL https://doi.org/10.3389%2Ffnins.2015.00307

[18] Dreiseitl S, Ohno-Machado L, Kittler H, Vinterbo S, Billhardt H and Binder M 2001 *Journal of Biomedical Informatics* **34** 28–36 URL https://doi.org/10.1006%2Fjbin.2001.1004

[19] Kononenko I 2001 *Artificial Intelligence in Medicine* **23** 89–109 URL https://doi.org/10.1016%2Fs0933-3657%2801%2900077-x

[20] Kukar M 2001 *Estimating Classifications' Reliability* Ph.D. thesis University of Ljubljana

[21] Frank E 1956 *Circulation* **13** 737–749 URL https://doi.org/10.1161%2F01.cir.13.5.737

[22] Wan L, Zeiler M, Zhang S, LeCun Y and Fergus R Regularization of neural networks using dropconnect http://cs.nyu.edu/ wanli/dropc/ accessed 2016-11-12

[23] Ciresan D, Meier U and Schmidhuber J 2012 *2012 IEEE Conference on Computer Vision and Pattern Recognition* (Institute of Electrical and Electronics Engineers (IEEE)) URL https://doi.org/10.1109%2Fcvpr.2012.6248110

[24] Lecun Y, Bottou L, Bengio Y and Haffner P 1998 *Proceedings of the IEEE* **86** 2278–2324 URL https://doi.org/10.1109%2F5.726791

[25] Benenson R Who is the best at x ? https://rodrigob.github.io/are_we_there_yet/build/#datasets accessed 2016-11-12

[26] Tappert C Rosenblatt's contributions http://csis.pace.edu/~ctappert/srd2011/rosenblatt-contributions.htm accessed 2016-11-10

[27] Thoma M Perceptron unit https://commons.wikimedia.org/wiki/File:Perceptron-unit.svg accessed 2016-11-10

[28] Maas A L, Hannun A Y and Ng A Y 2013 *Proc. ICML* vol 30

[29] Zeiler M, Ranzato M, Monga R, Mao M, Yang K, Le Q, Nguyen P, Senior A, Vanhoucke V, Dean J and Hinton G 2013 *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (Institute of Electrical and Electronics Engineers (IEEE)) URL https://doi.org/10.1109%2Ficassp.2013.6638312

[30] Krizhevsky A, Sutskever I and Hinton G E 2012 *Advances in Neural Information Processing Systems 25* ed Pereira F, Burges C J C, Bottou L and Weinberger K Q (Curran Associates, Inc.) pp 1097–1105 URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[31] Offnfopt (wikimedia user) Image of a multilayered neural network https://commons.wikimedia.org/wiki/File:Multi-Layer_Neural_Network-Vector.svg accessed 2016-11-08

[32] Karpathy A Cs231n convolutional neural networks for visual recognition – linear classification https://cs231n.github.io/linear-classify/ accessed 2016-11-14

[33] Rumelhart D E, Hinton G E and Williams R J 1986 *Nature* **323** 533–536 URL https://doi.org/10.1038%2F323533a0

[34] Nielson M Neural networks and deep learning http://neuralnetworksanddeeplearning.com accessed 2016-11-12

[35] Kingma D P and Ba J 2014 Adam: A method for stochastic optimization (*Preprint* arXiv:1412.6980)

[36] Hughes G 1968 *IEEE Transactions on Information Theory* **14** 55–63 URL https://doi.org/10.1109%2Ftit.1968.1054102

[37] Wang H and Raj B 2015 A survey: Time travel in deep learning space: An introduction to deep learning models and how deep learning models evolved from the initial ideas (*Preprint* arXiv:1510.04781)

[38] Chabacano (wikimedia user) Overfitting.svg https://commons.wikimedia.org/wiki/File:Overfitting.svg accessed 2016-11-14

[39] Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 *Journal of Machine Learning Research* **15** 1929–1958 URL http://jmlr.org/papers/v15/srivastava14a.html

[40] Maheshwari S, Acharyya A, Schiariti M and Puddu P E 2016 *Journal of Electrocardiology* **49** 231–242 URL https://doi.org/10.1016%2Fj.jelectrocard.2015.12.008

[41] Kreuger R 2007 Stelevatie_en.png http://en.ecgpedia.org/index.php?title=File:Stelevatie_en.png accessed 2017-04-01

[42] Ovid Technologies (Wolters Kluwer Health) 2005 *Circulation* **112** IV–89–IV–110 URL https://doi.org/10.1161%2Fcirculationaha.105.166561

[43] Flöck F 2016 K-fold_cross_validation_en.jpg https://commons.wikimedia.org/wiki/File:K-fold_cross_validation_EN.jpg?uselang=en-gb accessed 2017-04-03

[44] Steg P G, James S K, Atar D, Badano L P, Lundqvist C B *et al.* 2012 *European Heart Journal* **33** 2569–2619 URL https://doi.org/10.1093%2Feurheartj%2Fehs215

[45] Shin J H and Park K S 2012 *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (IEEE) URL https://doi.org/10.1109%2Fembc.2012.6346792

[46] Domhan T, Springenberg J T and Hutter F 2015 *Proceedings of the 24th International Conference on Artificial Intelligence* IJCAI'15 (AAAI Press) pp 3460–3468 ISBN 978-1-57735-738-4 URL http://dl.acm.org/citation.cfm?id=2832581.2832731

[47] Claesen M and Moor B D 2015 Hyperparameter search in machine learning (*Preprint* arXiv:1502.02127)

[48] Goodfellow I J, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 Generative adversarial networks (*Preprint* arXiv:1406.2661)