Universitetet
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Automation and Signal Processing	Spring semester, 2019 Open
Author: Stian Døhlen Berntsen	[illegible] (signature of author)

Programme coordinator:

Professor Kjersti Engan

Supervisor(s):

Jarle Urdal and Professor Kjersti Engan

Title of master's thesis:

Safer Births – Using Deep Neural Networks on Fetal Heart Rate Signals

Credits: 30

Keywords: Deep Learning, Convolutional Neural Networks, Signal Processing, Fetal Heart Rate	Number of pages: 43 + supplemental material/other: 1 + attached files Stavanger, June / 2019 date/year

University of Stavanger

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Safer Births **- Using Deep Neural Networks on Fetal Heart Rate Signals**

Master's Thesis in Automation and Signal Processing
by

Stian Berntsen

Internal Supervisors

Jarle Urdal

Professor Kjersti Engan

June 2019

Abstract

Infant death is a big issue, especially in Africa and parts of Asia where between 24 and 30 [21] in every thousand do not survive the first month. In Europe this number is only 5.9 in every thousand. Reading fetal heart rate signals requires specialists and is time consuming and tedious work.

The objective of this thesis is to determine if deep neural networks can detect birth complications based on fetal heart rate signals collected by using a Moyo fetal heart rate monitor as a part of the Safer Births project.

The best method found in this study included augmenting the data to get similarly sized classes, creating spectrogram images, and using a convolutional neural network for classification. The final method produced an F1-score of 0.13 and detected 21.875% of the births were bag-mask ventilation was needed immediately after. The proposed methods tested in this study have not been able to detect birth complications accurately, due to insufficient amounts of data, and low-quality signals missing important features for detection.

Preface

This thesis was written at the Department of Electrical Engineering and Computer Science, University of Stavanger. The thesis was conducted during the spring semester of 2019, and has been challenging and educational, but also enjoyable.

I would like to thank my head supervisor Jarle Urdal for his dedicated guidance and useful inputs through the semester.

I would also like to thank my supervisor Prof. Kjersti Engan for her advice and feedback throughout the master period.

Lastly I want to thank Laerdal Global Health for giving me the opportunity to participate in this project.

Contents

Abbreviations

ANN	Artificial Neural Network
BMV	Bag-Mask Ventilation
BPM	Beats Per Minute
CL	Convolutional Layer
CNN	Convolutional Neural Network
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
FC	Fully Connected
FHR	Fetal Heart Rate
FN	False Negative
FNN	Feedforward Neural Netowrk
FP	False Positive
LSTM	Long Short-Term Memory
MSE	Mean Squared Error
NN	Neural Network
RNN	Recurrent Neural Network
STFT	Short Time Fourier Transform
TN	True Negative
TP	True Positive

Chapter 1

Introduction

1.1 Motivation

Every year 2.2 million [21] children are stillborn or do not survive their first day. Many of these infants could have had a chance of survival if essential care was available around labour, delivery, and immediately afterward. That means having a skilled, well-equipped birth attendant available to assist women and newborns during childbirth. It also means detecting problems and intervening during labour and in the postnatal period. It is estimated that skilled care during labour could reduce the number of stillbirths during labour by 45% and 43%[21] of newborn deaths.

In Europe, 5.9 infants in every thousand do not survive beyond 28 days, but in Africa and parts of Asia, that figure is four to five times higher[21], depending on country. Laerdal Medical, Helse Stavanger and the University of Stavanger are part of the Safer Births research project working to reduce the newborn mortality rate by providing training and making automated, easy to use equipment to decrease the workload on the midwives.

A challenge with fetal heart rate (FHR) monitoring in the delivery room is that interpretation of these signals requires experienced specialists, which in less developed parts of the world, are not as available. In this work, a study of deep neural networks (DNN) applied to the problem of detecting patterns in fetal heart rate, which can help identify birth complications, and alert medical professionals faster than available methods.

1.2 Objective

The goal of this work is to explore if deep neural models can be used to detect childbirth complications based on the fetal heart rate leading up to birth. The task is also to find if a specific architecture is better suited than others, and if different pre-processing methods can give better results. If complications can be detected with the help of the Moyo monitor, illustrated in figure 2.1, and DNN, this can be further developed into a birth assistance tool.

1.3 Related Work

In 2008, a method using recurrent neural networks (RNN) to detect and classify congestive heart failure, ventricular tachyarrhythmia, atrial fibrillation and regular heartbeat in electrocardiogram (ECG) beats was published in the article "*Combining recurrent neural networks with eigenvector methods for classification of ECG beats*"[19]. Using eigenvector methods to generate power spectral density estimates, and using this as inputs for a RNN. The method performed well on these types of signals, with a 98.06% accuracy and concluded that pre-processing was an essential factor.

Übeyli wrote another article in 2010 doing a similar experiment but using the Lyapunov exponents of the ECG signals as input for the neural network. This performed slightly worse than the previous experiment but still had a 94.72% accuracy in the 4 class problem [20].

A paper investigating the use of artificial neural networks (ANN) applied on signals acquired from fetal monitoring during labour was published in 2013[4]. A dataset of CTG signals was used for prediction of neonatal outcome. The final result was a classification accuracy of 63.89% for classes living or dead.

In this thesis, ECG signals for the fetus were not available. Instead, noisy Doppler signals were used. The study using CTG signals is similar to this study, but 63.89% is a number which leaves a lot of room for improvement. The ECG studies suggest that heart activity and the birth outcome is highly correlated and using DNN to detect birth complications have given good results.

1.4 Proposed Method Overview

Using a convolutional neural network (CNN) on spectrogram images to predict if bag-mask ventilation (BMV) is needed immediately after birth, is the proposed solution to the problem. Initially RNN was tested, but it was quickly discovered that the proposed method, illustrated in figure 1.1 performed better.

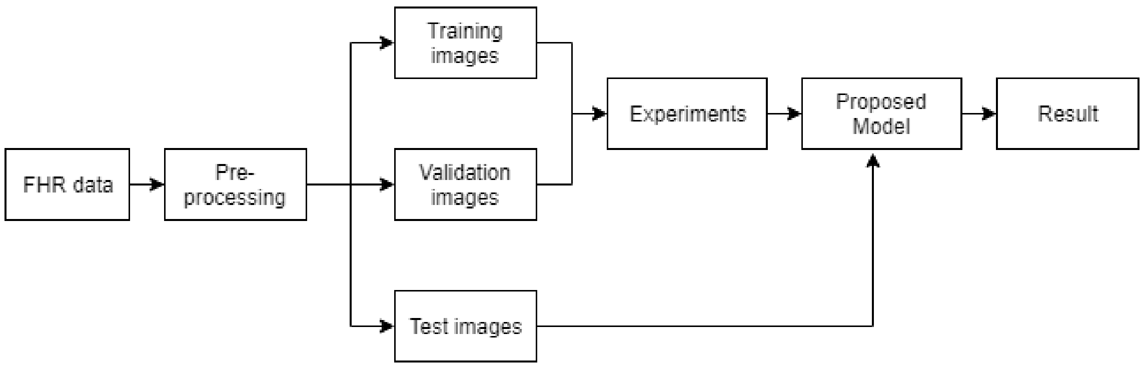

Figure 1.1: Overview of the proposed system.

1.5 Thesis outline

Chapter 2 - Data Material

This chapter describes the background of this dataset and how it was collected.

Chapter 3 - Background

This chapter describes the background for the thesis and theory behind the implemented methods in this thesis.

Chapter 4 - Method

This chapter explains how the methods where implemented and the classification experiments where set up.

Chapter 5 - Results

This chapter presents the results of the experiments described in the previous chapter.

Chapter 6 - Discussion

The results and experiments are discussed in this chapter.

Chapter 7 - Conclusion

The conclusion of the thesis are presented in this chapter.

Chapter 2

Background

This chapter contains the background information on the medical background, pre-processing, processing, and performance metrics used in this thesis. Neural networks are the main focus of this chapter because previous work indicates that deep neural models have given some performance on similar tasks before.

2.1 Medical Background

The medical background presents an overview over fetal heart rate and bag-mask ventilaton. The proposed method analyzes fetal heart rate signals to detect if bag-mask ventilation is needed immediately after birth.

2.1.1 Fetal Heart Rate

Normal FHR ranges are usually in the range of 110 to 160 beats per minute (BPM). Monitoring the FHR is usually done by a healthcare worker during late pregnancy and labour. There are 2 main methods to monitor this heart rate.

External FHR monitoring uses a device to listen to the fetus' heartbeat through the abdomen. Typically a Doppler ultrasound device is strapped on to reduce movement of the device itself, but it is affected by maternal or fetal movement.

Internal FHR monitoring consists of using a thin electrode put on the fetus' scalp. The electrode runs through the cervix and is connected to a monitor. This method is less sensitive to movement, which gives it better readings, but is more intrusive and can only be used if the amniotic sac has broken.

During labour, the FHR is usually classified into one of three categories. Category 1 is the normal or healthy category with the baseline of the FHR in the normal range. No special care is needed for cases in this category. Category 2 is the indeterminate category, typically seen by FHR deceleration between 2 and 10 minutes, recurrent variable decelerations with moderate variability or slightly higher or lower FHR than the baseline 110 to 160 BPM. If a FHR is classified as category 2, continuous surveillance is required, and a reevaluation as this can be indicative of something wrong. Category 3 is the abnormal category, indicated by a sinusoidal pattern or absent variability with recurrent late decelerations, recurrent variable decelerations, or bradycardia. Category 3 FHR requires immediate attention to solve the underlying cause of the abnormal FHR pattern. Depending on the situation, this may include provision of maternal oxygen, change in maternal position, treatment of maternal hypotension, or discontinuation of labour stimulation[10].

Laerdal Medical has developed an external FHR monitor called Moyo. The Moyo, see figure 2.1, is a small handheld Doppler based device designed for continuous FHR measurements and consists of a display unit and a sensor unit that can be strapped to the maternal abdomen for continuous monitoring. The sensor unit includes a 9-crystal pulsed wave Doppler ultrasound sensor with a frequency of 1 MHz to detect the FHR signal, which is logged at 2 Hz. It also includes a dry-electrode ECG sensor for the maternal heart rate and a three-axis accelerometer to capture maternal movements, but the focus in this thesis has been on the FHR signals.

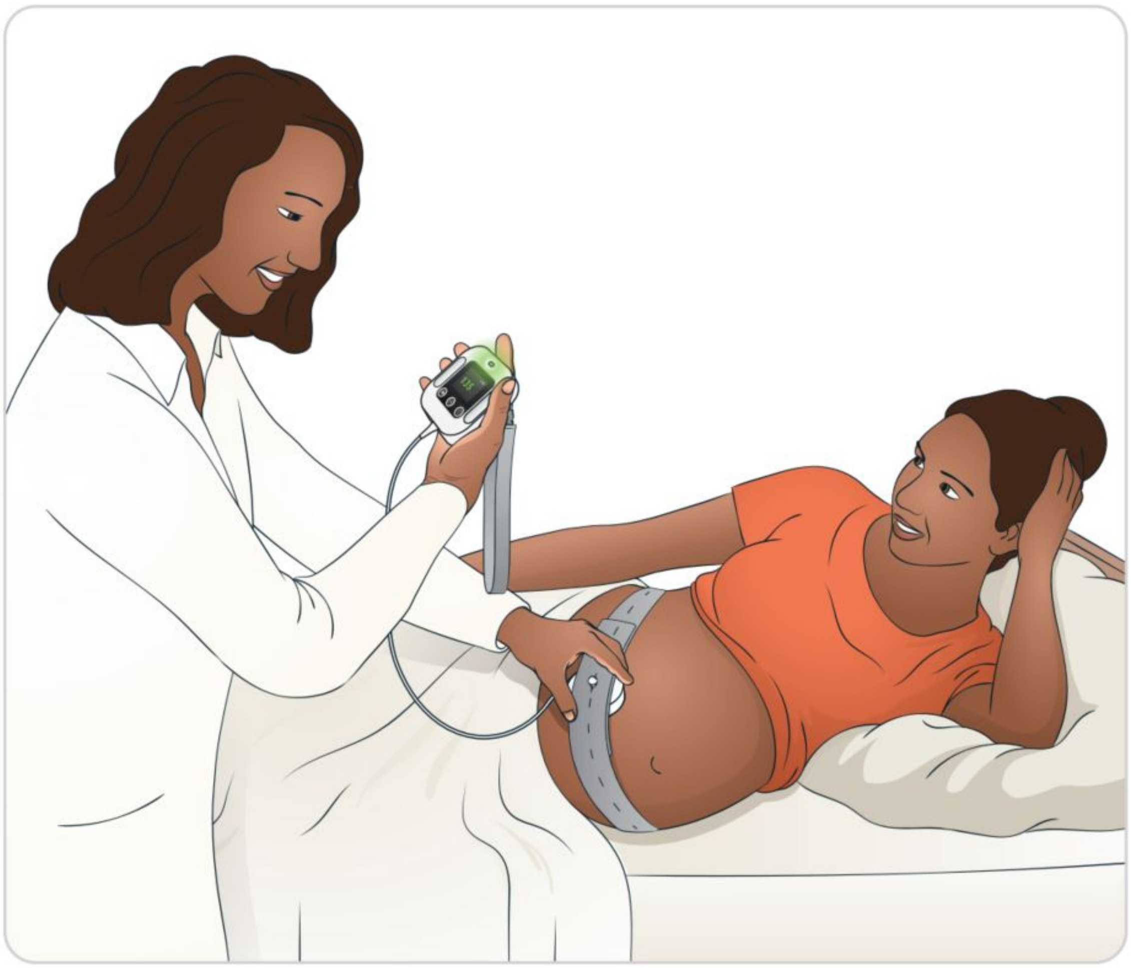

Figure 2.1: Moyo Fetal Heart Rate Monitor, Laerdal Global Health AS, Norway. Illustration reproduced with permission [1]

[1]Moyo Fetal Heart Rate, user guide", 20-08388/00026151 Rev E

2.1.2 Bag-Mask Ventilor

The bag-mask ventilator was invented in the year 1953[1] and is used as a manual resuscitator to provide ventilation to patients not breathing adequately. As seen in figure 2.2 the bag-mask ventilator consists of 3 parts; the bag, attached to a mask using a shutter valve. When the bag is squeezed, the device forces air into the lungs, and when released air gets sucked into the bag from the other end.

During regular breathing, the chest muscles pull the lungs outwards, creating a slight vacuum that fills the lungs with air. When using a ventilator, the lungs are force inflated with pressurized air. This can cause air to inflate the stomach, lung injury from over-stretching or lung injury from over-pressurization. Manual resuscitation is used on a sizable amount of newborns, using infant-sized ventilators to stimulate regular breathing.

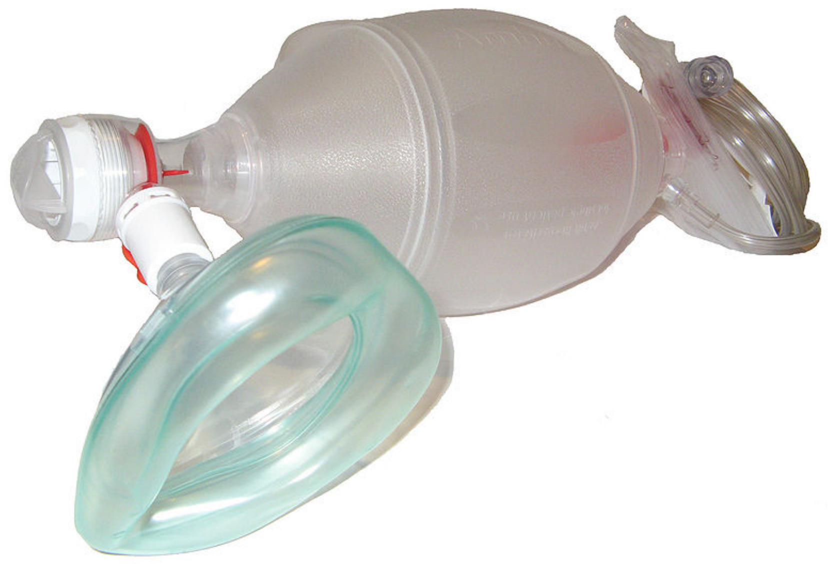

Figure 2.2: Bag-mask ventilator[2]

[2]Used with permission under the terms of the GNU Free Documentation License

2.2 Technical Background

The technical background presents the subjects used in thesis. Specifically the theory behind linear interpolation, spectrograms, artificial neural networks, and evaluation metrics.

2.2.1 Linear interpolation

Linear interpolation uses linear polynomials to construct new data points by using curve fitting. Given two known points given by the coordinates ($\mathrm{x_0}$, $\mathrm{y_0}$) and ($\mathrm{x_1}$, $\mathrm{y_1}$), linear interpolating gives a straight line between these. For a value x in the interval ($\mathrm{x_0}$, $\mathrm{x_1}$), the value y is given from the equation[16]:

$$y = \frac{y_0(x_1 - x) + y_1(x - x_0)}{x_1 - x_0} \tag{2.1}$$

2.2.2 Spectrogram

A spectrogram is a tool used for signal processing to represent a signal in the time-frequency domain. It is a visual representation of the intensity plot of a Short Time Fourier Transform (STFT) magnitude. STFT is the equivalent of doing a discrete Fourier transform (DFT) on segments of a signal. Equation 2.2 shows how to calculate the DFT, where $\mathrm{X_k}$ is segment k out of N total segments and $\mathrm{x_n}$ is a discrete impulse.

$$F(X_{\mathrm{k}}) = \sum_{n=0}^{N-1} x_{\mathrm{n}} e^{-\frac{2\pi i}{N} kn} \tag{2.2}$$

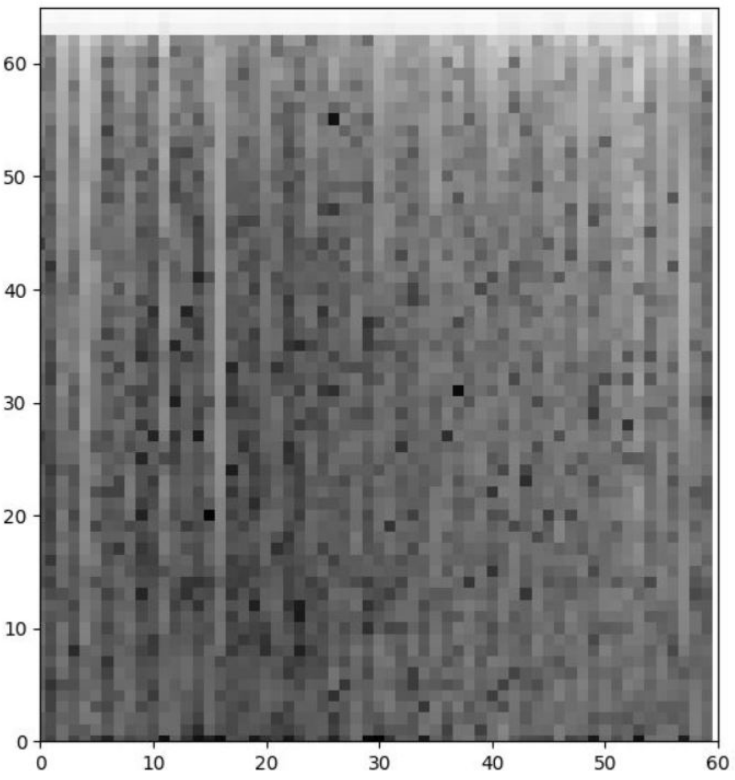

Figure 2.3: Example illustration showing a spectrogram using a grayscale heatmap. The darker areas represent higher intesity of the given fequency in the given segment.

Figure 2.3 is a typical example of a spectrogram represented as a grayscale heat map, with the x-axis representing segment number and the y-axis representing the frequency range. Spectrograms are often used in combination with NN for audio related tasks, for example music genre recognition[3].

2.2.3 Neural Networks

A neural network (NN), often called artificial neural network is a system with the ability to acquire knowledge by extracting patterns from raw data. The original idea of ANN was to mathematically model the neural networks seen in the brains of biological beings. The NN illustrated in figure 2.4 is built up by three different layers, input layer, hidden layer, and output layer. In order to be labelled as a DNN the network model has to consist of multiple interconnected hidden layers between the input and the output layer.

The input layer receives the data from the environment and transforms it into a readable pattern to the rest of the network. The hidden layer, or layers in the case of a DNN, process the input pattern before the information is transmitted to the output layer. The output layer processes the information gathered from the hidden layers and traditionally the outputs of the network corresponds to the predicted labels of the input.[9]

One of the issues with deep learning compared to more traditional methods is that it requires large amounts of training data and processing power, but with the advancements in processing power the last years and the growing popularity of the internet of things and big data this is becoming less of an issue.

In figure 2.4 the arrows between the fully connected layers represent outputs of the previous layers' activation functions. More about these concepts in the sections below.

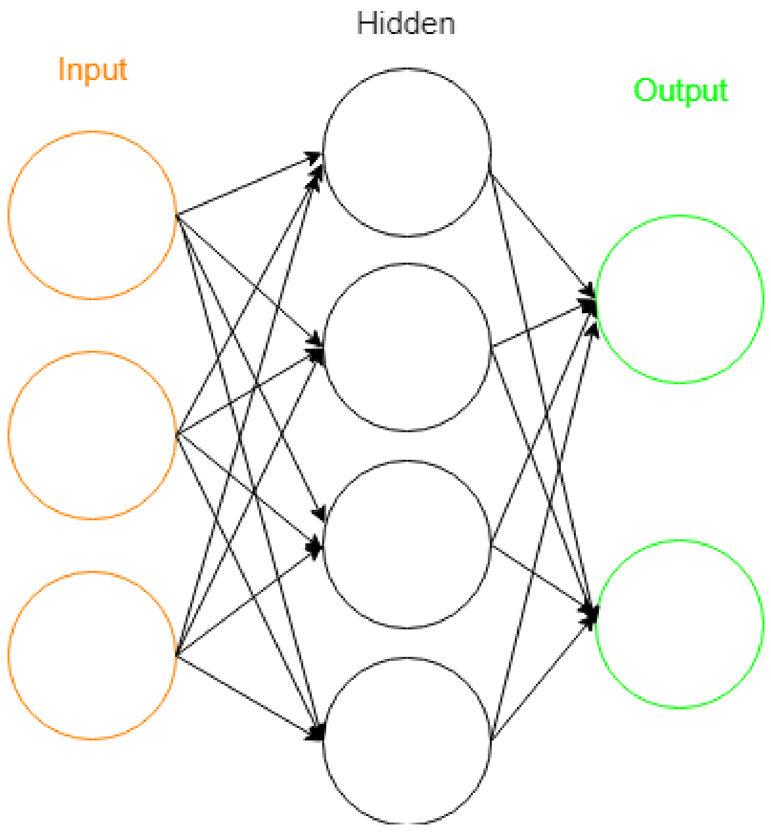

Figure 2.4: A fully connected neural network

2.2.4 Artificial Neuron

Artificial neurons are mathematical models of biological neurons and are the primary building block of neural networks. The neuron receives an input from the raw inputs or the output of neurons in a previous layer, as can be seen in figure 2.5. Every input link to the neuron has a modifiable weight associated with it. The goal of the weights is to bring the output as close as possible to the desired output value given a specific input. The sum of the weighted inputs is then fed into the activation function.

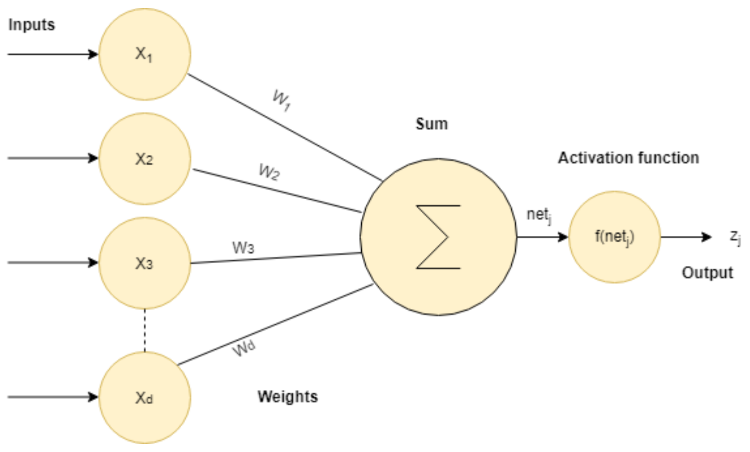

Figure 2.5: Illustration of an artificial neuron

In equation 2.3 $\mathrm{net_j}$ is the output of the neuron before the activation function, $\mathrm{a_i}$ is the inputs and d is the number of outputs of the previous layer, and $\mathrm{w_{ji}}$ are the weights and biases.

$$net_j = \sum_{i=1}^{d} a_i w_{ji} + w_{j0} = \sum_{i=0}^{d} a_i w_{ji} \tag{2.3}$$

The activation function $f(\cdot)$ generates the non-linearity for the outputs $\mathrm{z_j}$ as shown in equation 2.4. Activation functions are further discussed in subsection 2.2.5

$$z_j = f(net_j) \tag{2.4}$$

2.2.5 Activation function

The activation function defines the output of the neurons, given input and weights, and it ensures non-linearity in the perceptrons. Another property of the activation function is that it needs to be differentiable for the gradient-based optimization methods. Some of the most popular activation functions are:

Rectified linear unit - ReLU

ReLU is currently the most popular activation function[14]. As can be seen from equation 2.5 it returns 0 for all inputs $\mathrm{net_j}$ smaller or equal to 0, and returns the input for $\mathrm{net_j}$ greater than 0. It is computational inexpensive as there is no complicated math, which means the model takes less time to run. It is this simplicity and the effectiveness that has made ReLU so popular.

$$f(net_{\mathrm{j}}) = \begin{cases} 0 \ for \ net_{\mathrm{j}} < 0 \\ net_{\mathrm{j}} \ for \ net_{\mathrm{j}} \geq 0 \end{cases} \tag{2.5}$$

Softmax function

The softmax activation function is often used in the final output layer for classification problems. It assigns each element in the output a value between 0 and 1, and the sum of these values is equal to 1. These values can be interpreted as the probability of each class being correct.

$$f(net_{\mathrm{j}}) = \frac{e^{net_{\mathrm{j}}}}{\sum_{j=1}^{K} net_{\mathrm{j}}} \tag{2.6}$$

2.2.6 Fully connected layer

Fully connected (FC) layers connect every neuron in one layer to every neuron in the next layer. Fully connected means that the $\mathrm{net}_{\mathrm{j}}$ variable of all neurons in a layer is a function of all the outputs of the previous layer.

2.2.7 Convolutional Neural Network

Convolutional neural networks are a class of FNN, for multidimensional data, like images. Images are typically 2D or 3D arrays with the dimensions representing the height, width, and depth in case of color images. The images used in this thesis are grayscale images, meaning they only have two dimensions.

It is not always useful for all the neurons to consider all the inputs from the previous layer, for example, when using local patterns like colors or edges in an image. The neurons in a convolutional layer (CL) consider only a small neighbourhood of the neurons in the previous layer. This small neighbourhood is called the receptive field and is often much smaller than the total input to the layer. A CL works by using a filter kernel which is then convolved with the input images. The goal of this is to reduce the input into a form that is easier to process, without losing features. Combining this with parameter sharing leads to a lot fewer parameters than a FC layer, thus significantly increasing efficiency in terms of memory usage [15]. This also makes the network more scalable for big datasets, compared to only using FC layers.

Several parameters need to be determined for the convolutional layer: kernel size, the number of filters, stride, and zero padding. Kernel size, is the size of the filter kernel, usually a square with size 3x3, but other sizes are also used. The number of filters decides the number of feature maps that are created. Stride determines how much the filter is moved, stride 1 meaning the filter is moved 1 pixel at a time. Zero padding being a binary parameter deciding to zero pad the border of the input or not.

The output size of the layer is dependent on all these parameters, and the input size as can be seen in equation 2.7.

$$Output\ size = \frac{Input\ size - filtersize + 2\ x\ padding}{stride} + 1 \tag{2.7}$$

All of the operations above needs to be discrete because both the input image and filter kernel are discrete. The output size is the size of a matrix and needs to be an integer.

2.2.8 Loss function

The neural networks' loss function is used to evaluate the set of weights and biases for each iteration. It measures the performance of the output against the desired output or label and returns a single number, loss, which is then used in the backpropagation to update the weights. Some typical loss functions are given by equation 2.8 and 2.2.8. Where $\mathrm{y_i}$ is the actual value, y_i^p is the predicted value, and n is the number of predictions.

Mean square error loss

Mean squared error (MSE) measures the average squared error, or average squared difference between the predicted outcome and the actual outcome. The best MSE score possible is 0, meaning there is no difference between the predicted outcome and the actual outcome. Higher error meaning the predicted outcome is further from the actual outcome.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - y_i^p)^2}{n} \tag{2.8}$$

Cross Entropy Loss

Cross entropy loss can be read as the probability of the output being the correctly classified.

$$CrossEntropy = -\frac{\sum_{i=1}^{n}(y_i * log(y_i^p) + (1 - y_i) * log(1 - y_i^p)}{n}$$

(2.9)

2.2.9 Backpropagation

The backward propagation of errors, often shortened to backpropagation, uses optimization algorithms to train a NN efficiently. Optimization algorithms or optimizers are the functions calculating the new weights and biases to minimize or maximize the error function. Typically the job of the optimizer is to minimize the loss of the model. There are two main categories of optimizers, first-order optimization algorithms, and second-order optimization algorithms.

First-order optimizers minimize or maximize the loss function by using the gradient values with respect to the parameters. The derivatives show whether a function is increasing or decreasing at one specific point.

Second-order optimizers use the second-order partial derivatives called a Hessian matrix to minimize or maximize the loss. Since second-order derivatives need more processing to compute, this category is not much used. Even if it is not much used, second-order optimizers have the advantage that it accounts for the curvature of surface and give faster results in terms of the number of steps. In terms of computational complexity and memory requirements, it is slower.

There are different optimization methods for doing backpropagation. Stochastic gradient descent is the simplest one. Signals in the dataset are presented randomly, and the weights are updated after each one. In batch backpropagation the weights are updated once after each epoch, meaning all the patterns in the dataset have been presented once. This is usually the best training method, but if the dataset is large, it is much more practical to use smaller batches, since it takes longer to converge. With mini-batch backpropagation, the weights are updated after each mini-batch has been presented. The loss is calculated as in equation 2.10.

$$J(w) = \frac{1}{bd} \sum_{k}^{bd} L(t_{\mathrm{j}}, z_{\mathrm{j}}) \tag{2.10}$$

Where b is the batch size, d is the dimension of the output and L is the loss function with target value $\mathrm{t_j}$ and the produced output of the network $\mathrm{z_j}$. The loss can also be calculated as a sum of losses, but this is less robust for noisy data[13].

2.2.10 Overfitting

One of the biggest issues in data science is overfitting. Overfitting or overtraining means a model fits too good to the training data and therefore does not generalize well. The training data may contain noise, which the model learns, or is not a representative selection of the dataset. An example of overtraining is shown in figure 2.6. The upper graph shows the training loss, shown in red, going downwards and reaching almost zero, while the validation loss, shown in blue, doesn't change much. In the lower graph, the training accuracy goes up to almost 100%, while the validation accuracy goes down, then stabilizes between 50% and 60%. Due to the unbalanced nature of the dataset, with 90% of the observations belonging in class 0, the system starts off with guessing all observations to be class 0. Thus resulting in an accuracy of 90%.

Overfitting can happen due to insufficient amount of training data, training for too long or noisy signals, causing the model to specialize too much, or learning features that are not part of the actual signal.

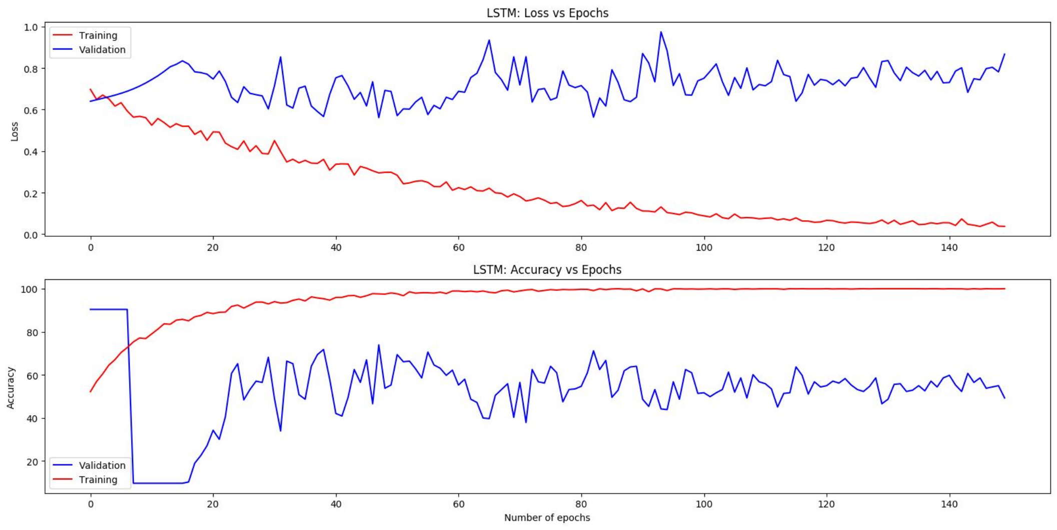

Figure 2.6: Upper graph: training and validation loss when overfitting. Lower graph: training and validation accuracy when overfitting. Validation is shown in blue and training is shown in red.

2.2.11 Batch normalization

When training DNN, the distribution of each layer's inputs changes, as previous layers change. This can slow down training by requiring lower learning rates and saturating activation functions. Batch normalization can counteract this by giving each mini-batch mean 0 and variance 1. Batch normalization uses the mean, $E(x)$, variance, $Var(x)$, and a constant e, added to ensure numerical stability, in each mini-batch in the training data to normalizes each input, x_i in the mini-batch using the equation below[8]:

$$\hat{x}_\mathrm{i} = \frac{x_\mathrm{i} \ - \ E(x_\mathrm{i})}{Var(x_\mathrm{i}) + e} \tag{2.11}$$

Only normalizing the inputs of a layer may change what the layer can represent. To counteract this the parameters γ_i and β_i, which scale and shift the normalized value are introduced. These parameters are learned along the model parameters, and restore the representation power of the network. Equation 2.12 shows how the normalized input $\hat{x}_\mathrm{i}$ is scaled and shifted. It also shows that by setting γ_i and β_i to $Var(x_i)$ and $E(x_i)$ respectively, and neglecting e, the transform can represent the identity transform.

$$y_\mathrm{i} = \gamma_\mathrm{i}\hat{x}_\mathrm{i} + \beta_\mathrm{i} \tag{2.12}$$

The batch normalization itself doesn't increase the training speed of the network, but it allows for increased learning rate. When using batch normalization, one input does not give a deterministic value, since it is dependent on all the data in the mini-batch. This is shown to reduce overfitting and reduces the need for dropout when training a network.[8]

2.2.12 Pooling Layers

One of the limitations of the convolutional layer is that the feature maps output the precise position of features in the input. This means that small movements of features in the input results in a different feature map, which can happen when cropping, rotating, or shifting the input image. Pooling layers address this problem by reducing the spatial dimensions of the feature maps.

A pooling layer can perform various operations, some of the more popular being max pooling and average pooling. The size of a pooling operation or filter is smaller than the size of the feature maps, and it is almost always 2x2. Figure 2.7 shows a max-pooling operation with kernel size 2x2 and stride of 2. The size of the input is reduced from a 4x4 matrix to a 2x2 matrix. The primary purposes of max-pooling are, first of all, reducing the size of the input, which reduces the number of parameters and weights, lessening the computational cost. The second is to reduce overfitting, by making the model converge faster, which also improves generalization performance [12].

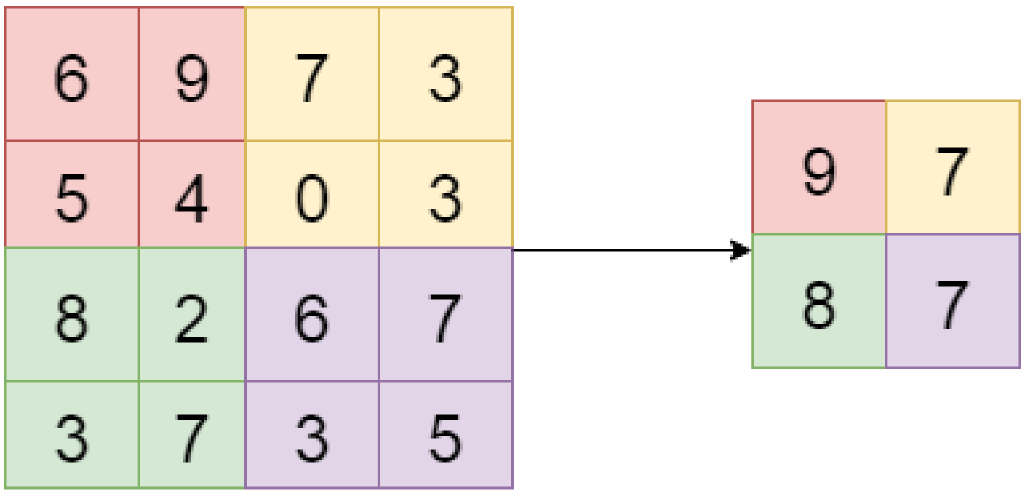

Figure 2.7: Max pooling filter with size 2x2 and stride 2. Left is the input matrix, and right is the downsampled output matrix

2.2.13 Dropout

A common approach to avoid overfitting is to use a method known as dropout. This refers to leaving out some of the neurons in the NN when doing a forward or backward computation. At each stage of training, each node has a 1-p probability of being dropped or p probability of not being dropped. This technique is used to help prevent overfitting by not letting the system settle into an overfitted state. Too much dropout can affect the performance of the network, and training can also require more epochs. The goal is to find a dropout probability that prevents overfitting without destroying the model[18].

When validating or testing dropout works differently than when training, as depicted in figure 2.8. Instead of randomly dropping neurons with probability 1-p, every neuron is weighted with p.

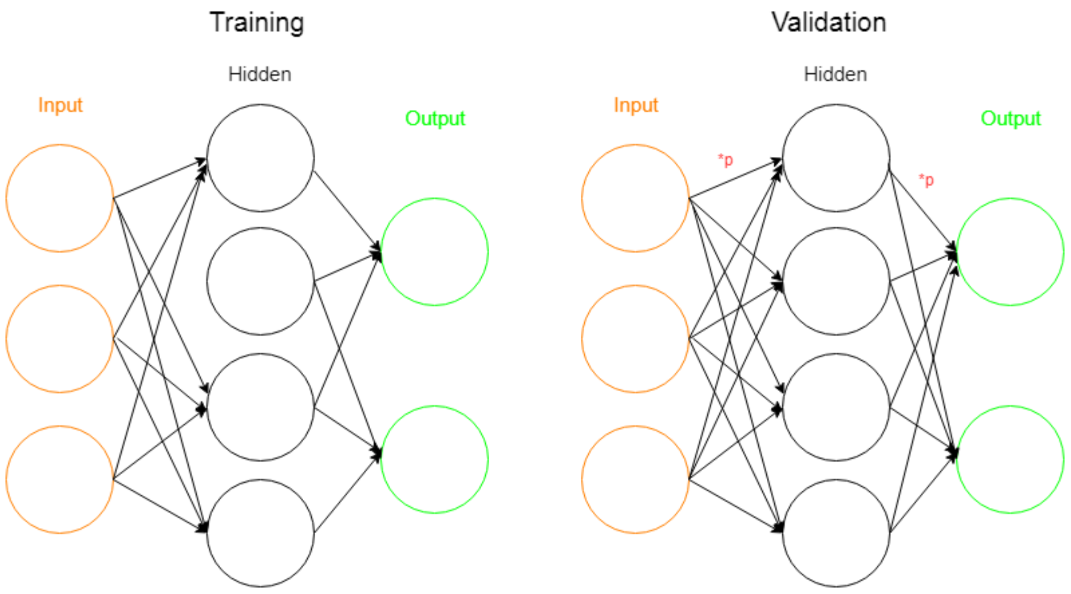

Figure 2.8: Dropout during training and validation

2.2.14 Transfer Learning

One of the big challenges in deep learning, especially on complex tasks, is that they require a lot of labeled data. Getting a lot of labeled data can be difficult, considering the time it requires to create. In machine learning, transfer learning is the concept of using a model trained on one dataset as a starting point for a second dataset. Using a pre-trained NN can help reduce the need for labeled data. The pre-trained models are usually trained on massive datasets. A well known pre-trained model is the VGG-16[17], which is trained on 1.3 million images and used to distinguish between 1000 classes.

The idea is training a model for a task on a lot of data, and using the same model with a new classification layer can give good results, similarly to how humans can transform something they have learned in one task, and use it for another. For example, know statistics and can use this to learn machine learning quicker.

Transfer learning has been shown to work well on computer vision based tasks, outperforming state of the art models on image classification tasks[17], such as the ImageNet challenge. Some popular pre-trained models for computer vision tasks are VGG-16, VGG-19, Inception V3, XCeption, ResNet-50.

2.2.15 Feedforward Neural Network

The feedforward neural network (FNN) is the most straightforward class of NN. In a FNN, all the information moves in one direction, meaning there are no feedback connections. The network illustrated in figure 2.4 is called a multi-layer perceptron, meaning there is more than one computational unit. A single-layer perceptron has no hidden layers, meaning the weighted inputs are fed directly to the outputs of the network. The sum of the weighted inputs are calculated in each node, and the node is activated by the activation function if the sum satisfies the requirements.

2.2.16 Recurrent Neural Network

A RNN is a class of ANN where connections between nodes form a directed graph along a sequence. Unlike feedforward neural networks, RNN can use their memory to process sequences of inputs, for example, a video instead of an image. Like a CNN, the outputs of a RNN are influenced by weights, but it also has a hidden state or a context state based on prior inputs. This means an input could produce different outputs based on earlier inputs in the series, which makes RNN applicable to tasks such as handwriting recognition[7], speech recognition[6] or time-series analysis[2].

2.2.17 Long Short-Term Memory Neural Network

Long Short Term Memory (LSTM) is a type of RNN that models long range dependencies better than conventional RNNs. It has been shown [5] to work well for classifying and making predictions using time series data as input.

2.3 Evaluation metrics

2.3.1 Confusion matrix

The confusion matrix is tool for visualizing the performance in machine learning and statistical classification. It shows the results for each class in a table layout, and these results can be used to calculate most other evaluation metrics used. A confusion matrix for a binary problem is shown below:

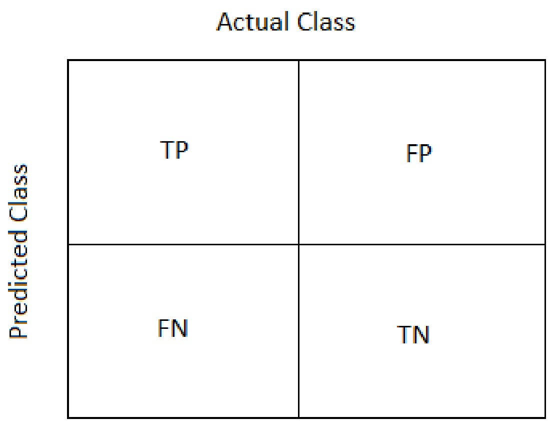

Figure 2.9: Confusion Matrix

True Positive (TP) is the number of correctly classified positive results. False Positive (FP) is the number of incorrectly classified negative results, also called a type 1 error. False Negative (FN) is the number of incorrectly classified positive results, also called type 2 error. True Negative (TN) are the correctly classified negative results,

2.3.2 Accuracy

Accuracy is the most commonly used evaluation metric when it comes to the performance of neural networks, and indicates what percentage of classifications that were correct. Calculated by using the formula:

$$Accuracy\ \% = \frac{TP + TN}{TP + FP + FN + TN}\ x\ 100 \tag{2.13}$$

With unbalanced datasets, this is not the best evaluation metric. Using the example of the dataset in this thesis where 91.25 % of the data is labeled class 0. If all data tested get classified to class 0, the network will get a 91.25 % accuracy, but it doesn't find any of the births were BMV is required. This is called the accuracy paradox.

2.3.3 Precision

When looking at one class' predictions, precision is the fraction of correctly classified instances. The Precision is calculated using the formula below:

$$Precision\ = \frac{TP}{TP + FP} \tag{2.14}$$

2.3.4 Recall

Recall, also known as sensitivity or true positive rate indicates the proportion of actual positives correctly classified. This can be a very relevant metric when a dataset is unbalanced. An example of this being the data used in this thesis, where the objective is to detect if BMV is required. The Recall is calculated using the formula below:

$$Recall = \frac{TP}{TP + FN} \tag{2.15}$$

2.3.5 Specificity

Specificity, also known as true negative rate indicates the proportion of actual negatives correctly classified. Used in the same way as recall, focusing on the opposite class in a binary classification problem. Specificity is calculated by using the formula below:

$$Specificity = \frac{TN}{TN + FP} \tag{2.16}$$

2.3.6 F1-Score

F1 score is a measure of the performance of a neural networks on a binary problem. It is a harmonic mean between the recall and the precision of a network, where the best score is 1, and the lowest score is 0. F1 score formula shown below:

$$F1 = (\frac{recall^{-1} + precision^{-1}}{2})^{-1} = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.17}$$

F1-score is a class of F-measure or F$_\beta$, the full formula is as follows:

$$F_\beta = \frac{(\beta^2 + 1) * Precision * Recall}{\beta^2 * Precision + Recall}, (0 \leq \beta \leq \infty) \tag{2.18}$$

β controls the balance between the precision and the recall. If β is larger than 1 the recall is weighted more, and if β is smaller than 1, precision is weighted more. If β equal 0, the F-score is equal to the precision.

Chapter 3

Data material

In this chapter, the data used in this study is presented.

3.1 Collection

The dataset used in this thesis is collected by the Safer Births Research Project, which is a research collaboration with partners including but not limited to, University of Stavanger, Laerdal Global health, and partner hospitals in Tanzania. The data was collected between October 2015 and June 2018 and were anonymized before transferred to research.

The project was ethically approved prior to implementation by the National Institute for Medical Research in Tanzania and the Regional Committee for Medical and Health Research Ethics in Norway (2013/110/REK vest).

3.2 Dataset

The dataset used in this study is collected from 3111 births. In addition to the FHR signals, it contained additional clinical information describing the labour annotated by designated research assistants present in the labour ward. This information includes, but is not limited to, outcome after 30 minutes and outcome after 24 hours.

The measured FHR is prone to disturbances creating noise in the signal. To get good measurements, the probe needs to placed correctly, but too much movement from either the fetus or the mother will introduce noise in the signal. To counteract this, a pre-processing was done earlier in the safer births project, to identify which areas are likely to contain incorrect data.

Some of the challenges with the dataset was that the FHR signals had various lengths, and some contained very little actual data. In this thesis, signals missing more than 50% of data were not considered usable and were together with data missing relevant labels removed from the dataset used for solving the problem. Table 3.1 gives an overview over the dataset. The starting and stopping points before births were also varying.

	no BMV	BMV	Too short	Missing label or less than 50% data	Total
# of births	1665	164	324	958	3111

Table 3.1: Overview of the dataset. The bordered data were considered usable.

The final usable dataset consists of 1829 births, where 1665, 91.03%, of them were normal, and 164, 8.97% needed bag-mask ventilation immediately after birth. The reason for the unbalancedness is that all the data is collected from births considered low risk, leading to fewer cases where immediate resuscitation is needed. In figure 3.1 and 3.2 you can see examples of usable signals.

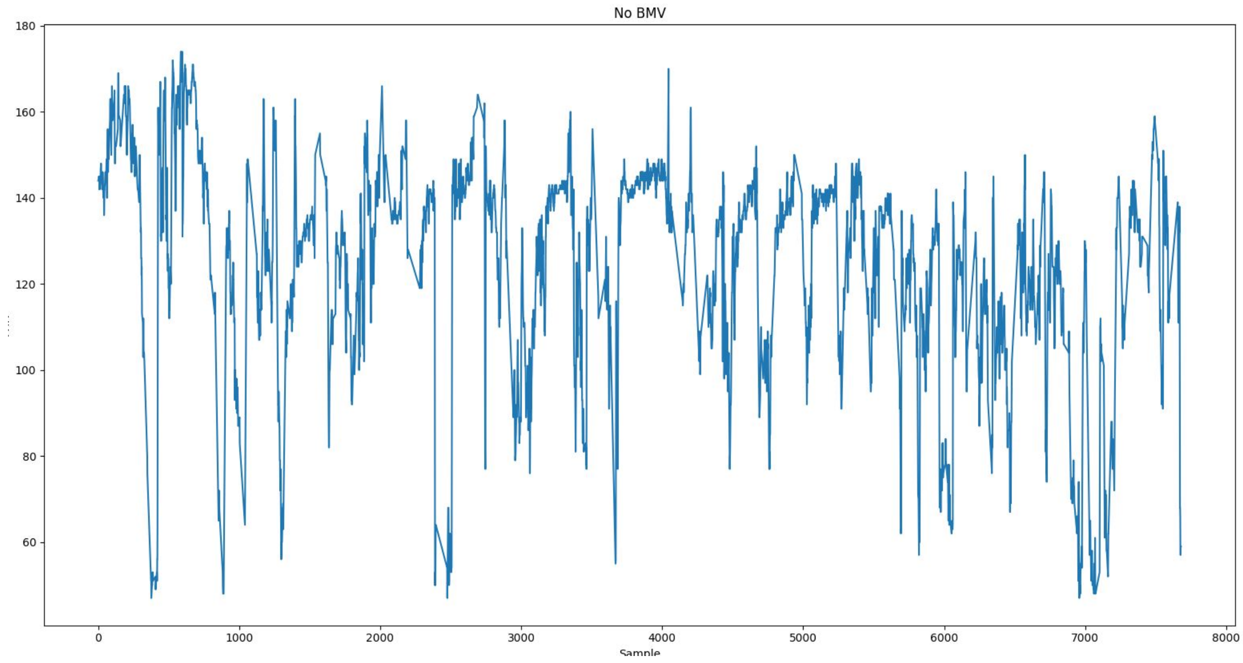

Figure 3.1: Example of a signal where BMV was not required

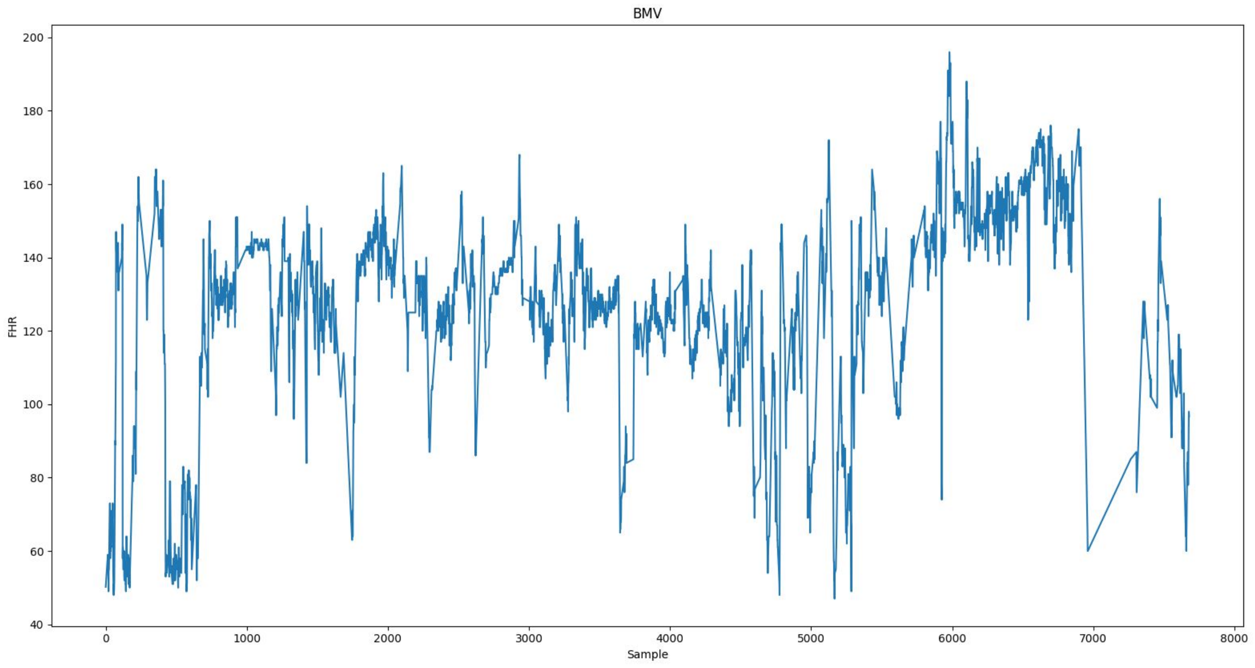

Figure 3.2: Example of a signal where BMV was required

3.3 Data Augmentation

One of the most frequent problem in the field of machine learning is the lack of data or unbalanced classes. In this thesis, the total amount of data after pre-processing was 1829 signals, and 91% of these were class 0, and 9% were class 1. One way of dealing with this problem is data augmentation, which means increasing the number of data points.

Considering the amount of data available, under-sampling the largest class was ruled out and over-sampling the minority class was chosen as an experiment. Only the last 7680 samples were initially used for the classifier, which is only a segment of the original signals. The over-sampling consisted of shifting these 7680 samples 3 minutes backward until there were created 10 spectrogram images for each birth.

Chapter 4

Proposed method

In this chapter each part of the proposed system are presented. First the pre-processing of the input signals is explained. Then the neural networks used and the experiments on the proposed system are explained. An overview of the system is illustrated in figure 4.1.

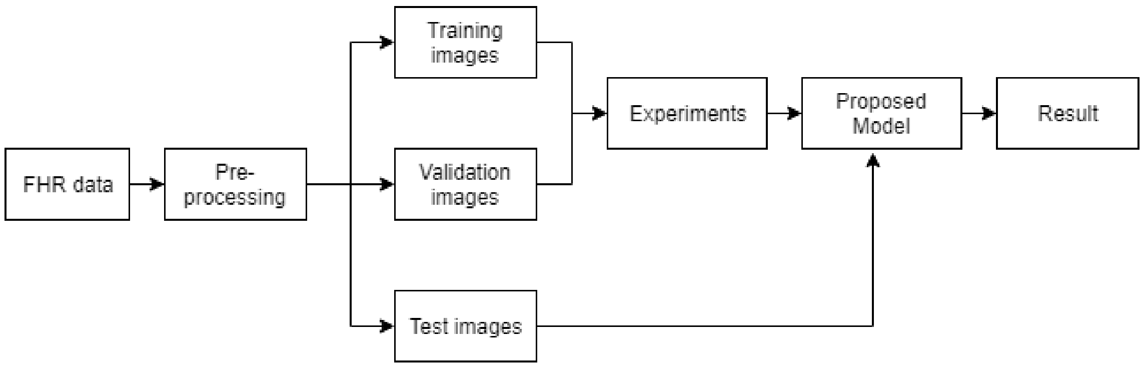

Figure 4.1: System overview

4.1 Pre-processing

This section explains the implementation of the pre-processing methods introduced in section 2, illustrated in figure 4.3.

The FHR signals used in this study are of varying length and quality. For the dataset to be usable, all of the data needs to go through similar pre-processing steps. The first part of pre-processing was removing data considered too noisy to be used, meaning signals containing less than 50% of actual data, not including when the Moyo device was turned off.

The labeled noise masks in the signal was removed, as to not make the classifier learn features created by something other than the FHR. This left the signal with a lot of blank samples, as shown in the second image on the figure. These were linearly interpolated over to make the samples more in the range of the actual heart rate. The result of this can be seen in the third image.

As seen in figure 4.2, there were large gaps in most of the signals, created by the Moyo device being turned off. Considering this, the varying lengths of the signals and a theory that most of the information is in the last part of the signal, only the last 7680 samples, which is slightly more than an hour, was used from each one.

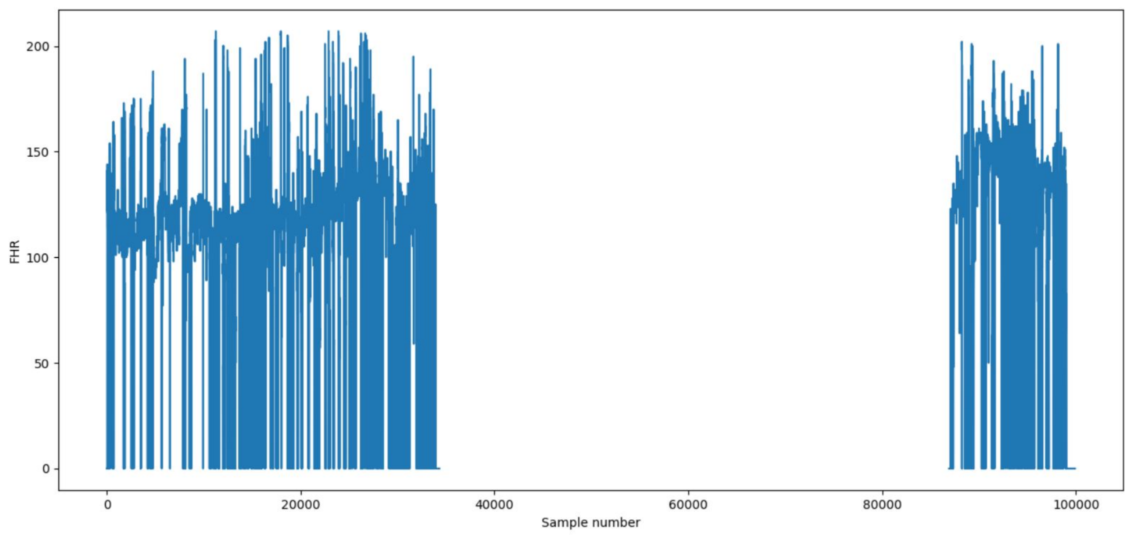

Figure 4.2: Example signal

To increase the distinctiveness of the features in the signals and to be able to use a more traditional method for classification, the signals were transformed into 65px x 60px spectrogram representations using 60, 128 samples long, non-overlapping segments. The result of this can be seen in the bottom image of the pre-processing figure 4.3.

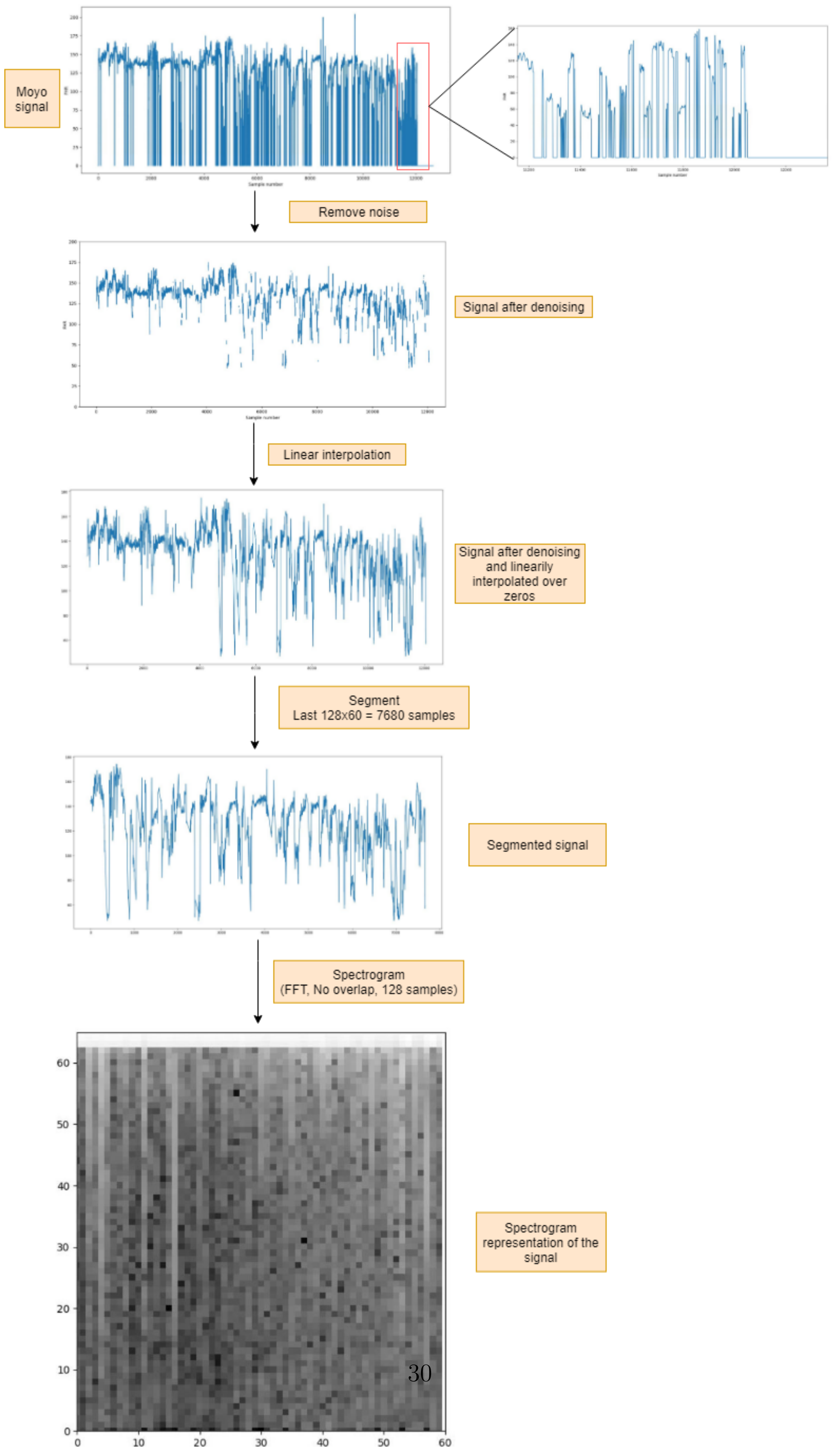
Moyo signal
Remove noise
Signal after denoising
Linear interpolation
Signal after denoising and linearily interpolated over zeros
Segment
Last 128x60 = 7680 samples
Segmented signal
Spectrogram
(FFT, No overlap, 128 samples)
Spectrogram representation of the signal

Figure 4.3: Pre-processing overview

4.1.1 Neural network

Several networks were tested, including RNN, LSTM, and different CNNs. The two NN giving the best initial results were one consisting of 7 CL, 2 max-pooling layers and a fully connected linear layer, using batch normalization, and ReLU as the activation function. The other was a pre-trained VGG-16 model, where the classifier was trained on the FHR data, and the feature extraction weights were frozen. The loss was calculated using cross entropy loss, as this seemed to give the best results.

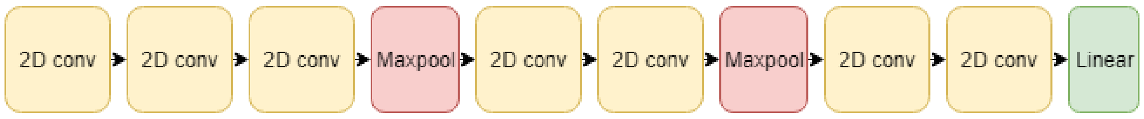

Figure 4.4: Convolutional neural network with 7 convolutional layers and 1 fully connected layer

The training logic for the NN is illustrated In figure 4.5. The dataset and the model are loaded into memory and initialized. After each mini-batch, the weights are updated, and if it was the last batch of the epoch, it goes through validation. In the validation step, the performance is evaluated based on unseen data. The model parameters are saved if there have been enough epochs, if the specificity improves or if the F1-score improves over the best previous set of parameters. The check for overtraining was done manually, and stopping of the training was done by setting a max amount of epochs. Only the best performing set of parameters for each dataset was used in the experiment phase of the thesis.

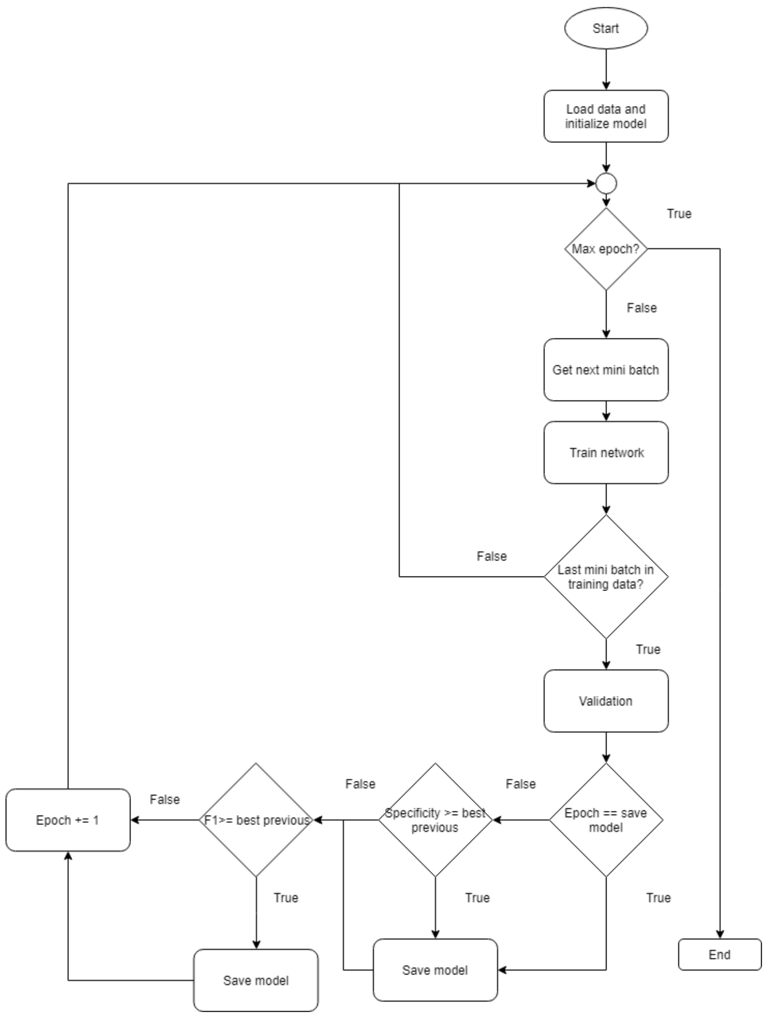

Figure 4.5: Flowchart visualizing the process of training the neural network

4.2 Transfer Learning

Due to the small amount of data available, transfer learning was implemented. Leveraging the weights of a pre-trained network and transfer some of the knowledge from previously learned domains to solve this task. The architecture and weights from the popular network VGG-16 were experimented on in this thesis.

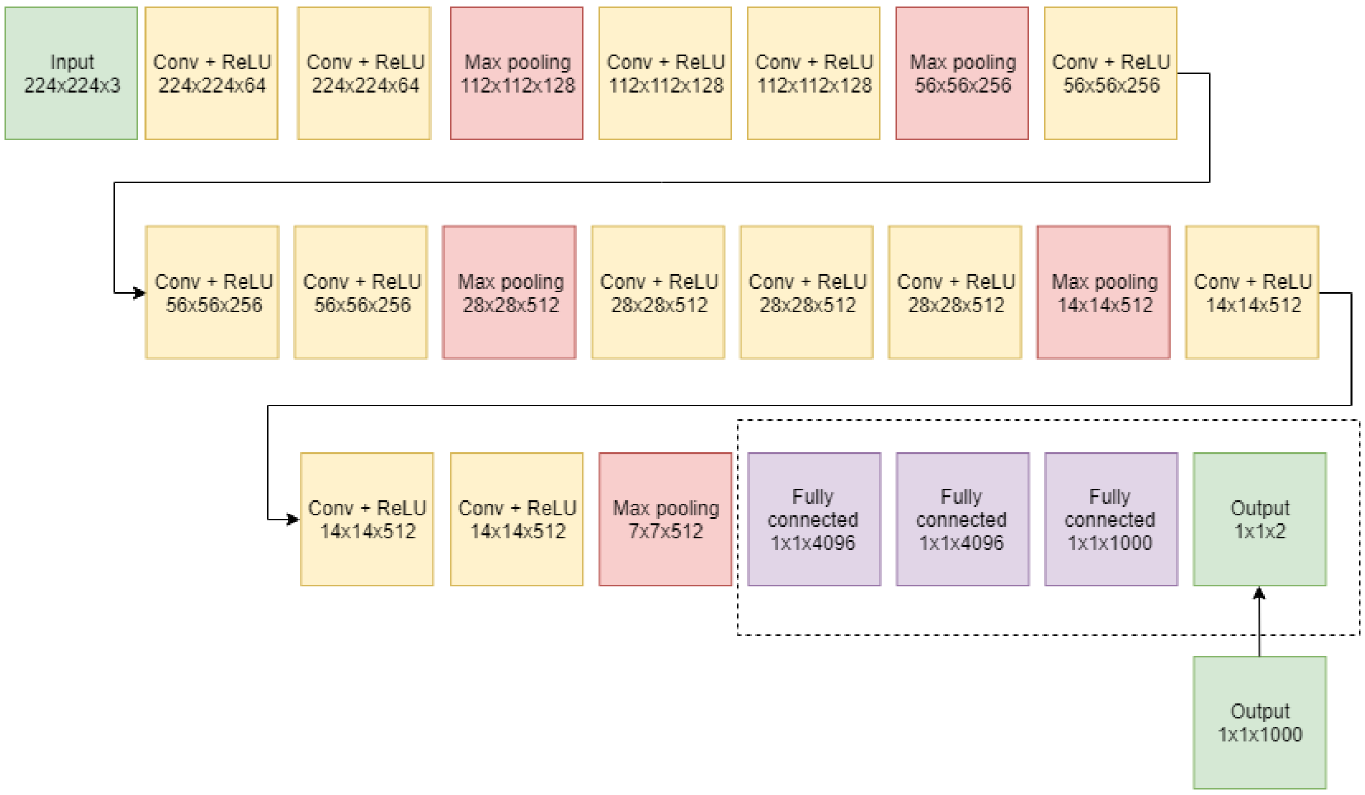

Figure 4.6: VGG-16 architecture with modified output layer.

As can be seen from figure 4.6, the VGG-16 network contains 13 convolutional layers and 3 fully connected layers. The weights of the 13 convolutional layers, the feature extraction part, are frozen, and only the marked fully connected layers, called the classifier are trained on the FHR signals. The output layer is also changed from predicting 1000 classes to predicting 2 classes.

4.2.1 Input Transformation

The VGG-16 network requires a 224px x 224px sized images as input. The original spectrogram images were 65px x 60px and needed to be resized for the transfer learning experiment. The torchvision library has several built-in transforms, including resizing of images, which uses a triangle filter for upscaling, and this was used to reshape the input images.

4.3 Experiments

The classification problem was a binary problem. The need of BMV or no need for BMV. The dataset was split into a training set, 60%, a validation set, 20%, and a test set, 20%

randomly after initial pre-processing of the data. These were the same for all networks tested. The validation set was used to decide the best performing parameters and architecture for the models, testing how well the trained model performed on unseen data. The test set was used for the final experiments giving the results presented in this thesis. Considering the imbalance between the classes in the dataset, accuracy was ruled out for performance evaluation. Therefore F1-score was used to decide which architecture and hyperparameters performed best for each experiment.

The initial pre-processing of the FHR signals was the same for all experiments. Removing the labeled noise, linearly interpolating over missing data in the signal, segmenting and transforming into 65px x 60px 0-1 normalized spectrogram images. The same dataset was used for all the experiments. The optimal hyper-parameters were searched for in each variation and model individually. The loss was calculated using cross entropy loss, as this seemed to give the best results initially. All the neural networks were implemented using python 3.7 and the package pytorch 1.1.

4.3.1 Pre-experiments

Rnn and LSTM-NN were tested with different number of layers, learning rates and layer dimensions. These experiments did not give any real results and it was quickly discovered that spectrogram images and CNN performed better, and these ideas were abandoned.

4.3.2 Experiment 1 - Baseline

In experiment 1, the spectrogram images were used directly, without more pre- or post-processing. There were in total 1829 usable signals, 1665 of these were class 0, no need for bag-mask ventilation, and 164 were class 1, need bag-mask ventilation. To counteract the unbalancedness of the training data, a weighted loss function was used during the training of the model. Weighing a misclassification of a signal belonging to class 1 as heavily as 11 misclassifications of a class 0 signal gave the best results on the validation set. There were a total of 1163 training signals, 1063, 91.4% of these were class 0, and 100, 8.6% were class 1. The results can be seen in chapter 5.

4.3.3 Experiment 2 - Augmented data

In experiment 2, the signals from class 1 were augmented to balance the classes. The augmentation process consisted of using the same signals with various shifts in time. Each shift was 360 samples, corresponding to 3 minutes. This was done 9 times for each birth in class 1, meaning a minimum signal length of 10920 samples was required. The total number of images in the training set was now 1869, distributed 959 in class 0 and 910 in class 1. The total number of signals in class 1 has been increased dramatically, but the number of patients was reduced from 100 to 91 because of the new signal length requirement.

4.3.4 Experiment 3 - Pre- and post-processing

The third experiment used the same model, which was trained for experiment 2. The difference is in the pre-processing and post-processing. Instead of only class 1 being augmented, all signals were augmented to make 10 spectrogram images per patient. The validation set for experiment 3 contain 2800 images, 2500, 89.3 % being class 0 and 300, 10.7 % being class 1.

All spectrogram images were first classified individually, similar to what was done in the previous experiments, then all the classifications for one birth were combined and thresholded. The threshold giving best performance on the validation set was chosen.

4.3.5 Experiment 4 - Transfer learning

The last experiment was repeating all the previous experiments, but instead of using randomly initialized weights, a pre-trained VGG-16 model was used for initializing the model. The feature detection part of the model was frozen and only the classifier was trained, using the FHR data. The output layer was changed to fit a 2 class problem.

Chapter 5

Results

This chapter consists of the evaluation of the experiments presented in section 4.3. Learning rates from 0.0001 to 0.05, dropout between 0 and 0.2 and different weights were tested. The results presented use the best performing set for each of the experiments.

5.1 Results and Analysis

All the models were trained on the training set, and the hyper-parameters were chosen based on the performance on the validation set. Each model's best F1-score for each dataset is presented in table 5.1.

	8 layers	Experiment 4
Experiment 1	0.2135	0.2368
Experiment 2	**0.2772**	0.2752
Experiment 3*	0.2111	0.2532

Table 5.1: F1 score during validation. *Thresholded at 0.2 for 8 layered model and at 0.5 for the Pre-trained model

The best performing method for the validation set was the 8 layered model trained on the more balanced training data with the class 1 data augmented to give 10 images per birth. More parameters in table below:

Parameter	Value
Train set	1869 images
Validation set	333 images
Epochs	33
Learning rate	0.003
Activation function	ReLU
Dropout	0.1
Mini-batch size	333

Table 5.2: Parameters of the model used in experiment 2

		Predicted	
		C0	C1
Actual	C0	246	55
	C1	18	14

Table 5.3: Confusion matrix of the validation on the 8 layered model

As table 5.3 shows, the network detected 43.75 % of the births were BMV was needed, with a 78.08% overall accuracy.

The pre-trained VGG-16 also performed best on the balanced training data, giving nearly the same f1-score. Parameters in the table below:

Parameter	Value
Train set	1869 images
Validation set	333 images
Epochs	3
Learning rate	0.0023
Mini-batch size	111

Table 5.4: Caption

		Predicted	
		C0	C1
Actual	C0	239	62
	C1	17	15

Table 5.5: Confusion matrix of the validation on the pre-trained VGG-16 model

The mini-batch size needed to be reduced because of memory limitations on the GPU available. The pre-trained models needed less time to reach the best result, and used a lower learning rates. This model detected 46.876 % of the birth complications, with a 75.68 % overall accuracy.

5.2 Verification of the result

The best performing network from validation was further evaluated. In the previous experiments, the model parameters were chosen according to how well it performed on the validation data. To verify the results, the model was tested on the test set. Only the best performing model on the validation set was tested this way. The verification results are shown in the figures below.

		Predicted	
		C0	C1
Actual	C0	234	67
	C1	25	7

Table 5.6: Confusion matrix of the final test. F1-score for class 1 is 0.13207

The verification results show that the model performs less than half as good on the unseen data. This may be because the test set or the validation set is not representative of the whole dataset. The more likely reason is that the model was fitted to well for the validation data.

Chapter 6

Discussion

6.1 Discussion

6.1.1 Dataset

The main challenge in this thesis was the dataset. There were only 3111 samples, and more than 91 % of those were labeled no BMV needed, and less than 9% were samples where BMV was used, which is the class most important to detect. Many of the samples also contained a lot of noise, in the form of high peaks caused by the movement of the Moyo device and zeros in the signal caused by the Moyo device not being able to measure the heart rate signals. It was decided that signals containing less than 50% actual data were considered unusable. The signals also needed more than 7680 samples, labels with BMV status, leaving only 1829 usable signals, of these only 164 required BMV immediately after birth. Having this amount of available data can make it difficult for a neural network to detect enough differences in the classes to make a good classifier that generalizes well.

6.1.2 Pre-processing

The goal of the pre-processing was to optimize the data for the neural network. There were gaps and noise in most of the signals, and these were linearly interpolated over, to make the samples in the range of an actual heart rate. This made the networks perform better, but can make the diversity between the signals smaller, meaning it could be harder to differentiate between the classes, even if it makes all the detectable features in a realistic range.

The signals were transformed into spectrogram images, which could be used as input for CNN. The spectrogram images gave the best results and became part of the final solution.

6.1.3 Data augmentation

Early experiments tried to distinguish between newborns defined as normal 24 hours after birth from those which had died before this point in time, which contained 0.71% of class death. Changing this to classifying normal and BMV, which had 8.97 % of class BMV gave better results. The increased performance could be because the classes were easier to distinguish from each other, or more likely, the classes were more balanced.

Considering the amount of data available, under-sampling the largest class was ruled out and over-sampling the minority class was chosen as an experiment. Data augmentation to balance the training data gave the best results on the validation experiments conducted in this study. Reusing the same signals, mostly overlapping, could affect the final verification results and make the model less generalized. This may be one of the reasons the final model had less than half the f1-score on the test set, compared to the tests on the validation set.

6.1.4 Neural Network

Different architectures were tested, different numbers of CL, dropout layers and values, pooling layers, with very little change in results. Increasing the number of CL increased time per epoch, with no increased accuracy, decreasing the number of CL also decreased the accuracy. Changing other parameters also gave very little change in the final results. A lot more parameters could have been tested, but the results of the ones tested indicated that the main problem was not with the network architecture, but something else.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The objective of this thesis was to explore deep neural models' ability to detect birth complications in fetal heart rate signals collected with the Moyo device. Several methods were tested, but the final developed method used a CNN to predict the outcome of a birth, based on spectrogram representations of the signals. Tested methods have not been able to predict outcomes accurately, with a test accuracy on the BMV class of 21.875%. The validation curve had very high variance, and the network overfitted fast. Data augmentation made minor improvements on the validation model, but the final tests performed less than half as good. The reason the proposed method is not producing is due to the insufficient amounts of data, and low-quality signals missing important features for detection.

7.2 Future Work

In future work, it could be investigated if training a CNN on a bigger, more balanced, and higher quality dataset would yield better results, especially important for the smaller class. LSTM networks have shown to produce good results on time-series data before[11], and could also give good results in this with a better dataset.

There is also different pre-processing methods that could be tested. Different segments of the FHR signal could be used, also shorter or longer segments. The noise and zeros in the signal were interpolated over, this could possibly be done using other methods.

Bibliography

[1] Ambu. *Ambu's History.* https://web.archive.org/web/20110427094405/http://www.ambu.co.uk/UK/About_Ambu_Ltd/Ambu%C2%B4s_History.aspx, year = 2011, note = Accessed: June 6, 2019.

[2] Jerome T Connor, R Douglas Martin, and Les E Atlas. "Recurrent neural networks and robust time series prediction". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 240–254.

[3] Yandre MG Costa, Luiz S Oliveira, and Carlos N Silla Jr. "An evaluation of convolutional neural networks for music classification using spectrograms". In: *Applied soft computing* 52 (2017), pp. 28–38.

[4] Antoniya Georgieva et al. "Artificial neural networks applied to fetal monitoring in labour". In: *Neural Computing and Applications* 22.1 (2013), pp. 85–93.

[5] Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. "Applying LSTM to time series predictable through time-window approaches". In: *Neural Nets WIRN Vietri-01.* Springer, 2002, pp. 193–200.

[6] Alex Graves and Navdeep Jaitly. "Towards end-to-end speech recognition with recurrent neural networks". In: *International conference on machine learning.* 2014, pp. 1764–1772.

[7] Alex Graves and Jürgen Schmidhuber. "Offline handwriting recognition with multidimensional recurrent neural networks". In: *Advances in neural information processing systems.* 2009, pp. 545–552.

[8] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[10] George A Macones et al. "The 2008 National Institute of Child Health and Human Development workshop report on electronic fetal monitoring: update on definitions, interpretation, and research guidelines". In: *Journal of Obstetric, Gynecologic, & Neonatal Nursing* 37.5 (2008), pp. 510–515.

[11] Pankaj Malhotra et al. "Long short term memory networks for anomaly detection in time series". In: *Proceedings.* Presses universitaires de Louvain. 2015, p. 89.

[12] Jawad Nagi et al. "Max-pooling convolutional neural networks for vision-based hand gesture recognition". In: *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA).* IEEE. 2011, pp. 342–347.

[13] Adrien Payan and Giovanni Montana. "Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks". In: *arXiv preprint arXiv:1502.02506* (2015).

[14] Prajit Ramachandran, Barret Zoph, and Quoc V Le. "Searching for activation functions". In: *arXiv preprint arXiv:1710.05941* (2017).

[15] M.K. Samarin. *Convolutional Neural Networks for Visual Recognition.* http://cs231n.github.io/convolutional-networks. Accessed: May 7, 2019. 2019.

[16] M.K. Samarin. *Linear interpolation. Encyclopedia of Mathematics.* `https://www.encyclopediaofma`
`org/index.php/Linear_interpolation`. Accessed: May 15, 2019. 2019.

[17] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[18] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[19] Elif Derya Übeyli. "Combining recurrent neural networks with eigenvector methods for classification of ECG beats". In: *Digital Signal Processing* 19.2 (2009), pp. 320–329.

[20] Elif Derya Übeyli. "Recurrent neural networks employing Lyapunov exponents for analysis of ECG signals". In: *Expert Systems with Applications* 37.2 (2010), pp. 1192–1199.

[21] Simon et al. Wright. *Ending Newborn Deaths*. 2014.

Appendix

Program Files

Attached in PyFiles.7z are the python scripts. They can be described by:

- CNN_main.py

  - Python script for loading the dataset and training a CNN.
- VGG16_main.py

  - Python script for loading the dataset and training the last layers of a pre-trained VGG-16 NN.
- PreProcess.py

  - Python script for loading the .mat files and doing initial pre-processing.
- Spektrogram1.py - Python script for leading the pre-processed signals and creating spectrogram image datasets.
- Testings.py - Python script for doing the final test of the model.