University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Robottechnology and signalprocessing	Spring semester, 2020 Open
Author: Håvard Lapin	
Programme coordinator: Professor Kjersti Engan Supervisor(s): Øyvind Meinich-Bache Professor Kjersti Engan	
Title of master's thesis: Adapt and Generalize Deep Learning Methods for Activity Recognition on Newborn Resuscitation Videos	
Credits: 30	
Keywords: Deep Learning, Convolutional Neural Network, Object Detection, Activity Recognition, Image Processing, Newborn Resuscitation	Number of pages: 97 Stavanger, 14 of June 2020

Adapt and Generalize Deep Learning Methods for Activity Recognition on Newborn Resuscitation Videos

Håvard Lapin

Spring 2020

Abstract

Low- and middle-income countries have nearly 99 % of deaths of children under 28 days. Complications during childbirths, such as birth asphyxia, account for most of these deaths. To prevent this, strengthening the quality of resuscitation training and providing better quality health service are a major focus.

Major Projects like Safer Births has had a focus to establish new knowledge and develop new technology to reduce the number of deaths. They have collected videos of newborn resuscitation at Haydom Lutheran hospital in Tanzania. This data is useful for evaluating the official resuscitation guidelines by investigating the treatment process and compliance with guidelines. With this data a timeline over the activities like suction, ventilation and stimulation can be extracted. However, issues like low brightness, non-standard camera placement and low frame rate videos persist in these data.

The aims of this thesis is to improve upon previous work involving the Safer Births data and put all the previous work into a single pipeline. Earlier work has used object detection networks to locate and classify objects in the videos to post-process the videos to a format that the activity recognition network could predict on.

The Convolutional Neural Network (CNN) from earlier work like RetinaNet and Inception I3D was tested and had promising results, but still had some issues with detecting objects like the suction penguin and a large amount of False Positives. To improve object detection different kinds of pre-processing methods were applied. The methods to correct the lighting resulted in poorer results, but training a new model with gradient based input together with the original model resulted in a loss of False Positives among most classes. Combining previous binary activity recognition models resulted in a promising gain in suction, but a loss in the ventilation prediction. Adding a more efficient optical flow method for usage with activity recognition encountered issues with training data generation and resulted in worse performance than previous work. Creating new methods in python from previous work was completed with a new pipeline that could predict and generate a timeline, but has worse performance and needs more work.

Preface

I would like to thank the University of Stavanger, The Department of Electrical Engineering and Computer Science, Theodor Ivesdal and my supervisors Kjersti Engan and Øyvind Meinich-Bache for allowing me to continue with my thesis during the ongoing global pandemic which resulted in the closure of the University of Stavanger.

I would especially like to thank my supervisors, Kjersti Engan and Øyvind Meinich-Bache for offering valuable advice and encouragement during this global closure.

I would also like to thank Theodor Ivesdal for helping with remote access to the unix server at the University of Stavanger and quick response with technical difficulties.

Contents

Acronyms

AAP American Academy of Pediatrics. 6

ANN Artificial Neural Network. 9, 10

AP Average Precision. 32–34, 72, 75, 76

BMR Bag Mask Resuscitator. 42, 46, 48

CNN Convolutional Neural Network. i, 16, 17, 19, 23, 25, 57, 61, 69

FN False Negatives. 33, 89

FP False Positives. i, 32, 34, 72, 76, 77, 80, 87, 89

FPN Feature Pyramid Network. 21, 22

FPS Frames Per Second. 40, 46, 59

GGmGa Greyscale, Gradient magnitude, Gradient Angle. 46, 58, 64, 71, 72, 76, 77, 80, 87, 88

GGmI Greyscale, Gradient magnitude, Intensity. 46, 64, 71, 72, 76

HBB Helping Babies Breathe. 7

IoU Intersect over Union. 35, 58

mAP mean Average Precision. 34, 72, 76, 80

MSE Mean Square Error. 14, 15

NB Newborn area. 43, 61

OBJ Object area. 43, 48, 49, 61

OHEM Online Hard Example Mining. 21

PR Precision-Recall. 33

ReLu Rectified Linear Unit. 12, 22, 23

RGB Red Green Blue. 23, 63, 72, 77

RoI Region of Interest. 63

TN True Negatives. 33, 34

TNR True Negative Rate. 34

TP True Positives. 32, 33, 35, 72, 76, 77

TPR True Postive Rate. 33, 34

WHO World Health Organization. 6

1 Introduction

1.1 Motivation

Every year hundreds of thousands of newborn dies during the first 28 days of life[40]. Almost 99 % of these deaths occurs in low- and middle-income countries[26]. 25 % of those children dies due to birth asphyxia[42]. Strengthening the quality of the training and ensuring that quality health services are available are recommended[33]. While there is common guidelines for treating birth asphyxia available the interactions provided are not fully explored yet[31]. By studying the therapeutic activities like ventilation, suction and stimulating being performed and the duration of these the guidelines given could be evaluated or find out if they are followed. Videos of these therapeutic activities could be gathered and timelines over the activities could be extracted from these videos. Gathering of data is therefore crucial, yet offers two problems. One is the amount of manual labour required to label and verify that the data with timelines are correct, while the other is the privacy aspect. To try to handle both these problems machine learning and neural networks are being developed to generate data and statistics without the need for manual labour to the same degree.

1.2 Earlier major projects

A project that since 2013 has focused on establishing new knowledge and develop innovative product to help save newborns is Safer Births[4]. This project has collected a lot of data from Haydom Lutheran Hospital including video recordings of resuscitation, which is the data used in this master project and vitals.

1.3 Previous Work

Previous work on automatic analysis of videos of resuscitation in the Safer Births project has been done with using object detection to find relevant objects in the videos and activity recognition to find segments of therapeutic activities being performed in the videos, through the Safer Births project. Among these are the work performed by Øyvind Meinich-Bache[29] and Simon Lennart Austnes[3] that culminated with the article *"Activity Recognition from Newborn Resuscitation Videos"*[28]. Meinich-Bache et al. used RetinaNet an object detection network to predict the location of an object while using Inception I3D to determine if

an activity was being performed. With the activity recognition network both RGB input and optical flow input was used, but the optical flow method was very time consuming. The work done was promising, but still had some issues with detection and recognition. All of this earlier work was separate and did not have a continous pipeline.

1.4 Aims/Objectives

The objective of this thesis is to improve and adapt upon earlier work both in object detection and activity recognition, improve the existing optical flow calculation to better take use of the processing capacity available, and lastly put together all previous work into a single pipeline. With object detection there still exists some difficulties with recognising certain object as well as reducing false predictions. In activity recognition there is currently multiple binary models in use. Reducing these models to increase efficiency is one thing to look at. Previously a slow CPU based optical flow implementation were used. Cutting down the processing time for this would be wanted when new data from other hospital is added.

1.5 Thesis outline

Chapter 2: Medical background

This short chapter describes the medical background theory needed to better understand what is happening during resuscitation.

Chapter 3: Technical background

This chapter describes all technical theory behind the methods implemented.

Chapter 4: Data material

A chapter introducing where different data is coming from, what they contain and names for datasets used in this thesis.

Chapter 5: Methods

This chapter explains how the methods and shows how they are implemented to be used in experiments.

Chapter 6: Experiments

An introduction to each experiment conducted, what settings were used and which dataset were used.

Chapter 7: Results

The results from the experiments introduced in the previous chapter.

Chapter 8: Discussion

This chapter discusses the results from the previous chapter and why the result might have turned out the way they did.

Chapter 9: Conclusion

The conclusion of the thesis is presented here and further work.

2 Medical Background

2.1 Birth Asphyxia

Birth asphyxia is a condition where the newborn lack oxygen during birth and/or struggles to establish or sustain spontaneous respiration after birth which leads to lack of oxygen in various organs. This condition can lead to neuronal cell death and brain damage which can lead to death. Birth asphyxia is can be caused by a problem with blood to gas exchange which results in lack of oxygen in the blood and accumulates carbon dioxide. There is many causes of this condition, both maternal and fetal and some risk factors are: maternal age, multiple births, lack of regular check-ups during pregnancy, wrong orientation of the fetus and low birth weight[2]. According to World Health Organization (WHO) birth asphyxia causes 25 % of all neonatal[1] deaths[42]. In 2016 the number of neonatal deaths was 2.6 million were approximately 650 000 of these was caused by birth asphyxia[40].

2.2 Treatment

When birth asphyxia occurs the newborn will be in need of resuscitation. There is multiple steps of treatment when resuscitating a newborn. Often not all of them needed to be followed through. The American Academy of Pediatrics (AAP) treatment guide suggest these actions[24]:

Prevent Hyperthermia

First step is to dry the baby an wrap the newborn to prevent hyperthermia. Depending on the resources available wrap a warm blanket, using exothermic chemical mattresses or heating pads.

Suction

After the initial drying it is important to clear the airway by initially wiping the nose and mouth, but if this is still restricting the airways a suction device is needed to forcefully clear the airways. This need to be gentle not to harm the baby.

[1] neonatal period is the first 28 days of life[40]

Stimulation

During the drying and further resuscitating stimulating the baby by rubbing the back and/or flicking the feet is used to encourage the initial respiratory effort of the baby.

Ventilation

If the above treatment is not successful and the newborn is not breathing after one minute positive-pressure ventilation is needed with the help of a ventilator bag of some sort, self-inflating, flow-inflating or other similar device. The effectiveness of this can be seen by the colour, heart rate and muscle tone of the newborn.

Chest compression

As a last action chest compressions are used together with ventilation if the heart rate of the newborn is below 60 beats per minute 60 seconds after starting ventilation.

2.3 Helping Babies Breathe

Helping Babies Breathe (HBB) is a an evidence-based educational program focusing on the baby should breath well or be ventilated within the first minute of the birth, this time period is called "the golden minute". The HBB program was implemented in 8 hospitals in Tanzania and resulted in a reduction of early neonatal deaths with 47 % and fresh stillbirths by 24 % during the period from September 2009 to March 2012[32].

In cases where the baby is not breathing sufficiently at birth the action plan is shown in figure 1[14].

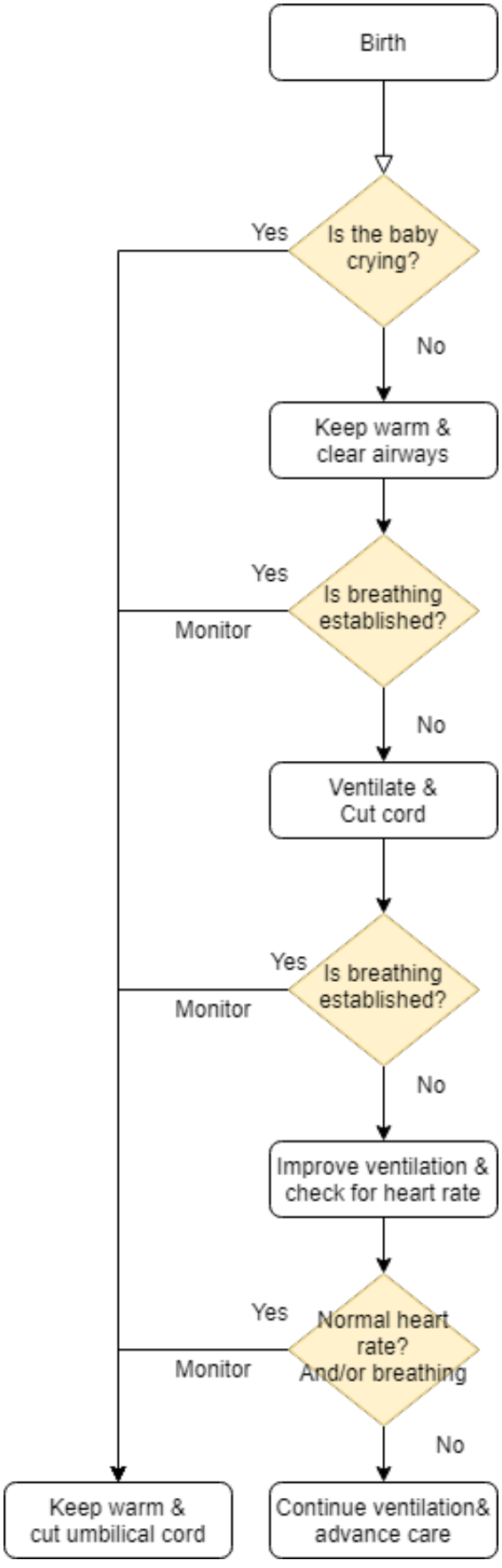

Figure 1: Flowchart over HBB guidelines for the treatment as described in AAP treatment plan[24]

3 Technical background

3.1 Artificial Neural Networks

Artificial Neural Network (ANN) is a biology inspired network trying to simulate the neural structure of the brain. It is a network which similarly to the brain attempts to learn from experience. ANN consists of units called neurons. Looking at the biological neuron in figure 2 it receives synaptic input via the dendrites from the axon where the sum of total inputs determines if the neuron will pursue an action. The dendrites is similar to a tree branching out and is connected to other neuron via their axon terminals through small protrusions called spines. These connections are called synapses. After the dendrites have completed processing the input the axon generates an action and transmits it to the terminal and to new neurons through the synapses[41].

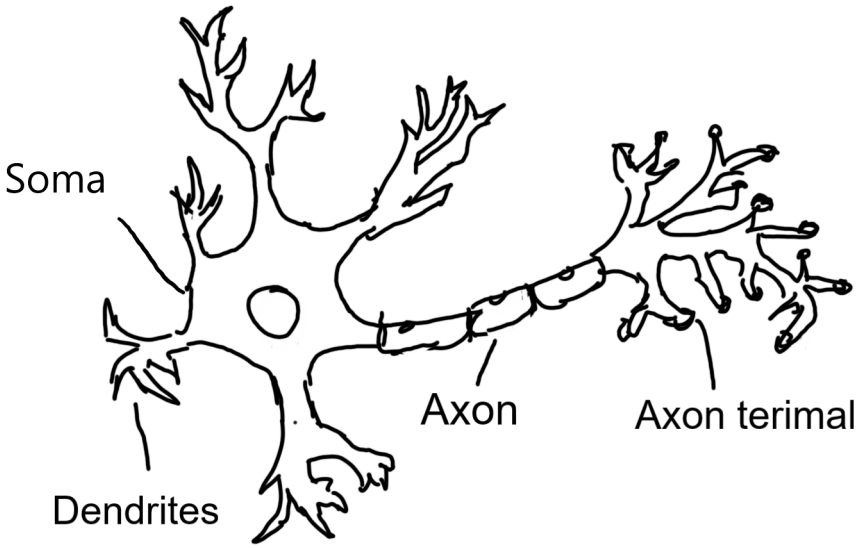

Figure 2: This image shows a biological neuron. The dendrites are treelike structures receiving synaptic inputs and sending them to be processed to the axon connected by the cell body (soma). After the signal has been processed it is sent to the axon terminal and connected to other neurons.

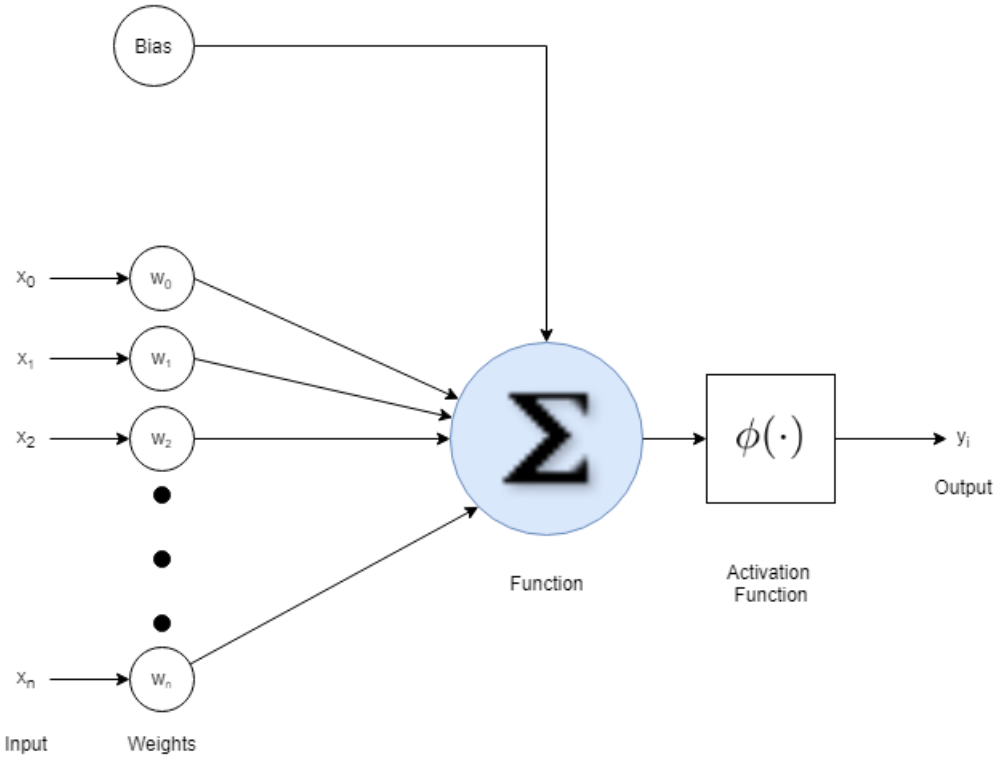

Figure 3: This image shows an artificial neuron called a perceptron. It has multiple inputs with weights per input. This is fed to an activation function $\phi(\cdot)$ which calculates the output y_i

ANN has multiple inputs like the dendrites which is connected to a weight or synapse. There is also the option for a bias value which allows for shifting activation functions up or down. In its simplest form the artificial neuron generates the sum of the input and weights and sends this to the activation function to generate a single output, this is depicted in figure 3. The simplest equation for this kind of artificial neuron can be written as:

$$y_i = \phi(x_0 w_0 + x_1 w_1 + ... + x_n w_n) = \phi(\sum_{i=0}^{N} x_i w_i) \tag{1}$$

This above function would then determine the output based on the function $\phi(\cdot)$[10].

3.1.1 Activation functions

Activation functions is the function that similarly to the dendrites and axons in biological neuron decides if an artificial neuron should be activated or not based on calculations with

the weighted sum and bias. Activation functions are non-linear since linearity would allow for following layers to be reduced to a single layer. There is a heap of different activation functions depending on the input, some different functions are shown in figure 4. Some used in Image Processing are:

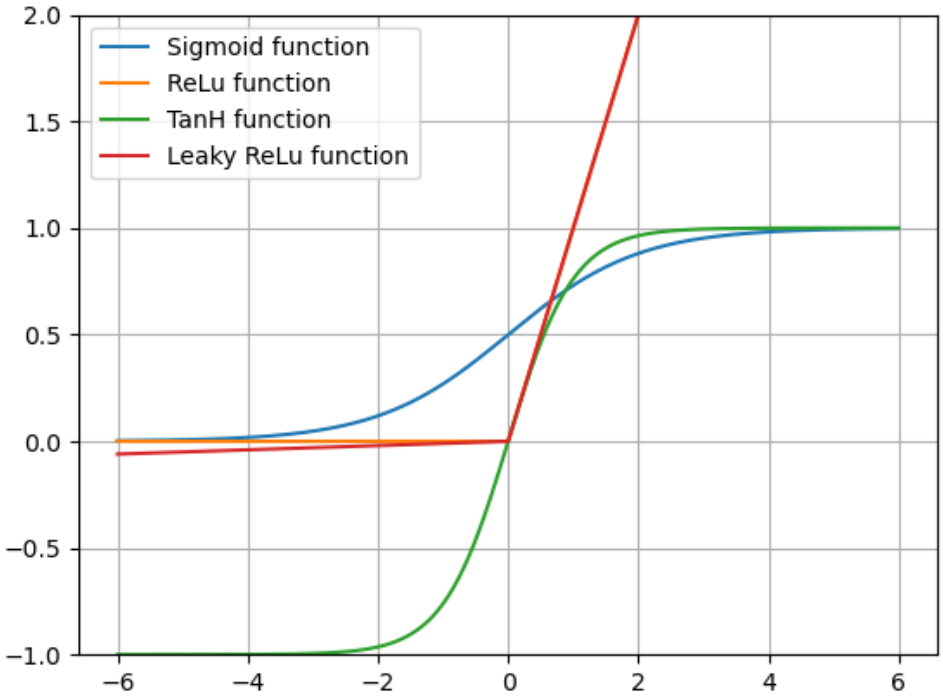

Figure 4: Comparison of three of the different activation Functions

Sigmoid

Sigmoid function ranges from 0 to 1 and is shaped into a characteristic 'S'-curve as seen in figure 4. The use of this function is when the wanted output is a probability. Drawbacks to this function is that the gradient can go to zero due to large negative inputs causing the gradient to vanish and the network to get stuck during training[23].

$$\Phi(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

TanH

The TanH is similar to the Sigmoid function, but ranges between [-1,1]. This function is shown in 4. Unlike the Sigmoid function large negative inputs is able to be mapped to a negative making it less prone to get stuck during training, but still maps values to a small interval[23].

$$\Phi(x) = tanhx = \frac{sinhx}{coshx} \tag{3}$$

Rectified Linear Unit (ReLu)

ReLu is a function which ranges from zero to infinity and is linear from zero to infinity, but due to the negative inputs being mapped to zero it is non-linear. Like the Sigmoid function ReLu has some issues with training and can kill neurons if the weights are updates in such a way they will never be reactivated. To fix this problem Leaky ReLu was introduced which has a smaller slope on the left-hand side of the y-axis and ranges from negative infinity to infinity[43]. Both figures can be seen in 4

$$\Phi(x) = \begin{cases} x & \text{if } x > 0, \\ ax & \text{otherwise,} \end{cases} \tag{4}$$

Softmax

Softmax is a regression form that normalises all output with a probability distribution which sums to one. This is often used as an output layer due to its distribution. All the output values ranges between [0,1] and can be used both by multi classification and binary classification. This function has a known use-case together with Cross entropy[2][13].

$$\Phi(x) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_i}} \tag{5}$$

[2]Cross entropy loss is described in section 3.1.4

3.1.2 Fully Connected Feedforward Network

A fully connected feedforward network as its name suggests is a network that the information moves forward, does not cycle or loop and every neuron in one layer is connected to another neuron in another layer[30]. In its simplest form (single-layer perceptron shown in figure 3) the outputs are connected directly to the inputs and uses output data from previous layer as input data in the next neuron. Multi-layer perceptron however is interconnected like figure 5 and usually contain at least one hidden layer. Based on the bias and weights different inputs triggers different nodes in the network.

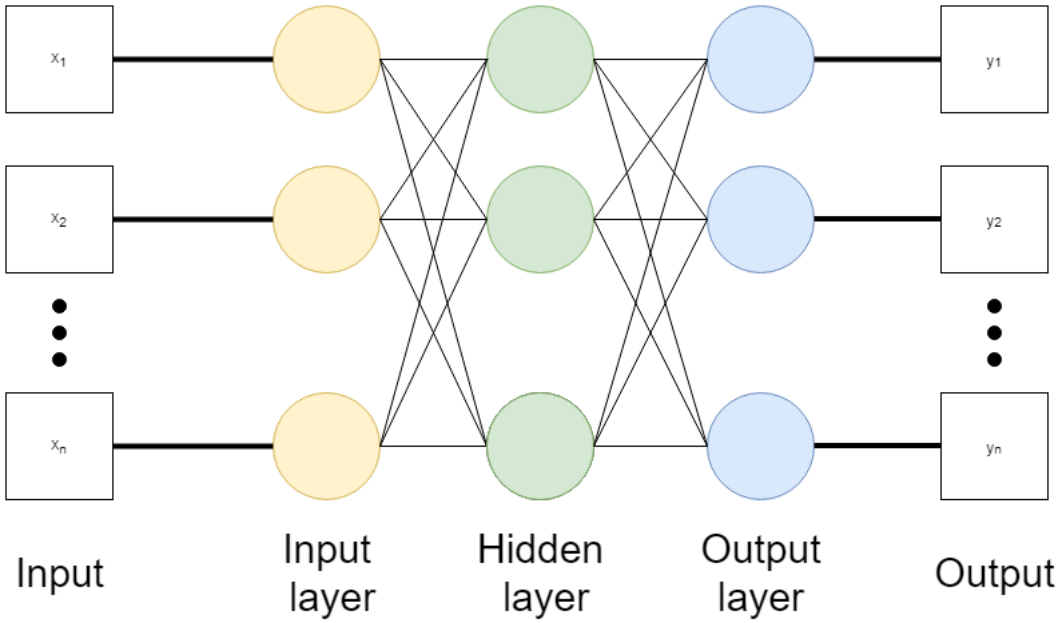

Figure 5: An example of a fully connected feedforward multi-layer perceptron network with a input layer, output layer and one hidden layer. Each of the nodes in the different layers contain perceptrons similar to figure 3.

3.1.3 Supervised Learning

A popular learning method is supervised learning. This is when the input data to the network is fed with knowledge of both the input data and its labels. The goal is to approximate the mapping between input data and labels to further predict on similar data. In an easier term: we know the answer and want the network to predict as close to correct answer as deemed satisfying[18].

3.1.4 Loss functions

Loss functions compares the prediction outputs and true labels that can be used to minimise with the help of backpropagation. The objective is to calculate the overall error of the model during a batch for the optimisation process to acquire a measurement for improvement. Some of the most common loss function are listed in the following.

L2

L2 has been a commonly used loss functions in image processing[44]. L2 is essentially Mean Square Error (MSE) or quadratic loss. It is the average of the squared difference of the predicted outputs and true labels. The true label is y_i, the prediction is $\hat{y}_i$ and N is the number of predictions.

$$L2(\hat{y}, y) = \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2 \tag{6}$$

Smooth L1

Smooth L1 loss function is a loss function that is less sensitive to outliers[3] than the popular L2 loss is.

$$Smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \quad \text{Where } x = (y_i - \hat{y}_i) \tag{7}$$

If the absolute value of x is lower than 1 it shares similarities with L2 loss function where it is a the squared difference between the calculated output and the actual output multiplied with a weight, but differs if the above statement is not true. However smooth L1 might need more tuning to the learning rates than other loss functions.[17].

Cross-entropy loss

Cross entropy or logistic loss is a method to calculate loss based on the sum of true labels y_i multiplied with the logarithm the prediction output $\hat{y}_i$[27].

$$CE(\hat{y}, y) = -\sum_{n=1}^{N} y_i log(\hat{y}_i) \tag{8}$$

[3]Outliers here mean data that strays significantly from the data.

Focal loss

A problem with certain one-stage object detection networks[4] are the scenarios where the imbalance between foreground and background classes causes issues. Focal loss was designed to address these issues. The focal loss is defined as:

$$FL(p_t) = -(1 - p_t)^{\gamma} log(p_t) \tag{9}$$

This loss function was designed by Facebook AI Research in 2017 and is derived from Cross Entropy on binary classification[27]. This can be shown in equations 10 and 11.

$$CE(\hat{y}, y) = \begin{cases} -log(\hat{y}) & \text{if y = 1} \\ -log(1 - \hat{y}) & \text{otherwise} \end{cases} \tag{10}$$

$$p_t = \begin{cases} \hat{y} & \text{if y = 1} \\ (1 - \hat{y}) & \text{otherwise} \end{cases} \tag{11}$$

3.1.5 Backpropagation

Backpropagation is adjusting the networks weight and biases to minimise the error between the prediction outputs $\hat{y}$ and the true labels y. It is implemented in neural networks to make supervised learning work. The output error is calculated with a loss function $J(\cdot)$. One of the simplest loss functions is Mean Square Error (MSE) (or L2 loss) shown in equation 12.

$$MSE = \frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 \tag{12}$$

To adjust the weights and biases with this function the partial derivative (gradient) is calculated with respect to the end layer and propagated backwards, which explains the name. Optimisation algorithms like Adam or gradient descent uses backpropagation and the resulting gradient to update the weights of the network if the activation functions are differentiable[25]. An important hyperparameter used in backpropagation is η also known as the learning rate. This parameter is multiplied with the gradient of the loss function to scale the effect of the function. A larger learning rate speeds up the training process, but might cause issues with convergence, while a small learning rate slows down the training

[4]One-stage object detection network is networks which first distinguishes the object from the background then localise the position of the object[7].

process, but ensure converges to a potentially local minimum value[18]. The formula for backpropagation is shown in equation 13 where w is the weights that are updated.

$$W_{k+1} = w_k - \eta \cdot \nabla J(w_k) \tag{13}$$

Performing backpropagation depends on the settings. A update can be performed for each training sample, per batch or when the entire data set has been fed through the network (epoch). RetinaNet for example uses upwards of 95-100 epochs to complete training, while simpler networks can require significantly less. To monitor the performance gain/loss of each epoch several kinds of loss are calculated based on classification on both validation and training sets. This can be calculated per batch to determine if the network is overfitted or require more training.

3.2 Convolutional Neural Network

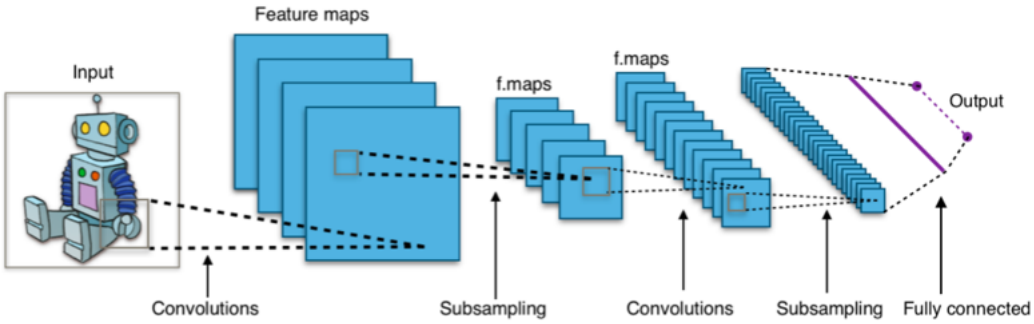

Figure 6: Typical CNN structure with a fully connected layer. the pooling layer is called subsampling here.

Figure created by Aphex34 under Creative Commons Attribution-Share Alike 4.0 license International[6]

Convolutional Neural Network (CNN) has been in use and development since the latter half of the 20th century. This network was a method to imitating how a biological neural network perceives visual data, but it is not the only usage. In 1980 Kunihiko Fukushima developed a system called "neocognitron" which introduced two of the basic CNN layers: convolutional and downsampling layers[16]. Though the neocognitron shares some properties with CNN a major difference is the training process where CNN uses backpropagation

[6]Link to the licence: https://creativecommons.org/licenses/by-sa/4.0/legalcode

for training while the former has had different kinds of supervised and unsupervised learning algorithms during its development and usage[16]. Common for Convolutional Neural Network is how they are constructed. These kinds of networks can consist of multiple layers. In some cases CNN consist of convolutional layers, downsampling layers, pooling layers and fully connected layers. Common for these kinds of CNN is the usage of backpropagation for training.

Convolutional layer

As the name indicates the convolutional layer uses a form of convolution or more precise using a filter kernel to perform sliding dot product to approximate convolution. The input to this layer is a tensor with the required dimensions of Height x Width x Number of channels. Next, a kernel/filter with different size depending on the desired output from the convolutional layer is applied onto the image using sliding dot product. The chosen kernel is multiplied with a segment of the image the same size as the kernel using dot product to create the convoluted output. This is repeated with shifting after each "convolution" with a set stride. An example on this would be a 5x5x3 image and a 3x3x3 kernel. Simplifying to focusing on one channel (5x5x1 and 3x3x1) for reference see figure 7. The dot product

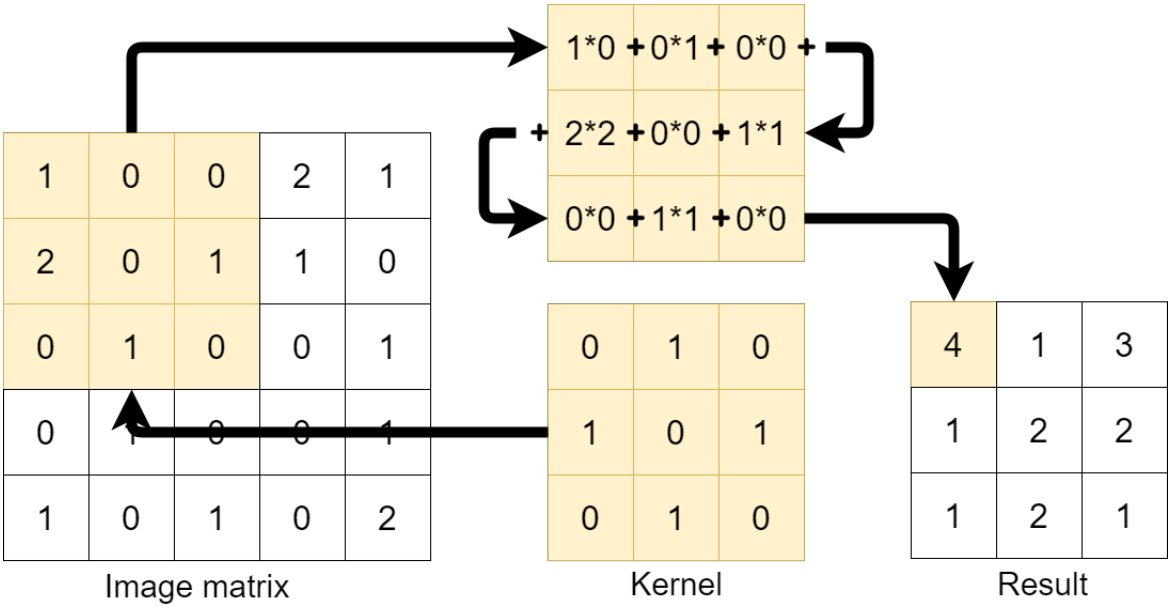

Figure 7: Example of sliding dot product with convolutional kernel filter. The highlighted part multiplied using dot product and the result is then showed in the highlighted part in the result matrix.

between the kernel and a highlighted 3x3 image segment is calculated, giving the result 4. Continuing with this operation with a stride one gives the result matrix and the output has been reduced to a 3x3 convoluted feature map. The benefit of using the convolutional layer is to extract features such as gradient angles, colours and gradients. There can be more than one convolutional layer where each feature is compacted into a smaller more feature rich understanding of the image. The convolutional layer also supports padding options as well to augment the output size of the convoluted feature matrix[18]. The figure 7 does not have padding and is using stride at one.

Pooling layer

The pooling layer's output can look similar to the convolutional layers, but it is used to reduce the spatial size of the convoluted feature matrix to reduce the required computational power needed to process the data. An added benefit of pooling is to extract the dominant feature of the matrix to furthermore improve the training of the network[36]. Two common pooling methods are: Max Pooling and Average Pooling. Max pooling extracts the dominant feature value from ie. a 3x3x1 matrix in a 5x5x1 convoluted feature matrix see figure 8 for reference. Average pooling on the other hand calculates the average from the 3x3x1

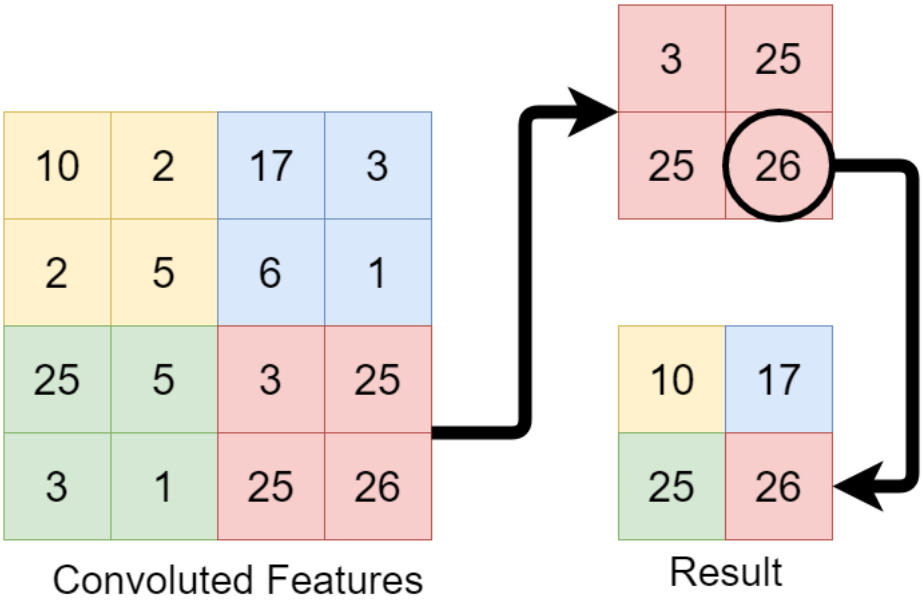

Figure 8: Max pooling on a feature matrix and its corresponding result. The colour corresponds to the segments calculated.

matrix (using the same example as last paragraph) in the 5x5x1 convoluted feature matrix, see figure 9. One benefit that max pooling has over average pooling is that it suppresses

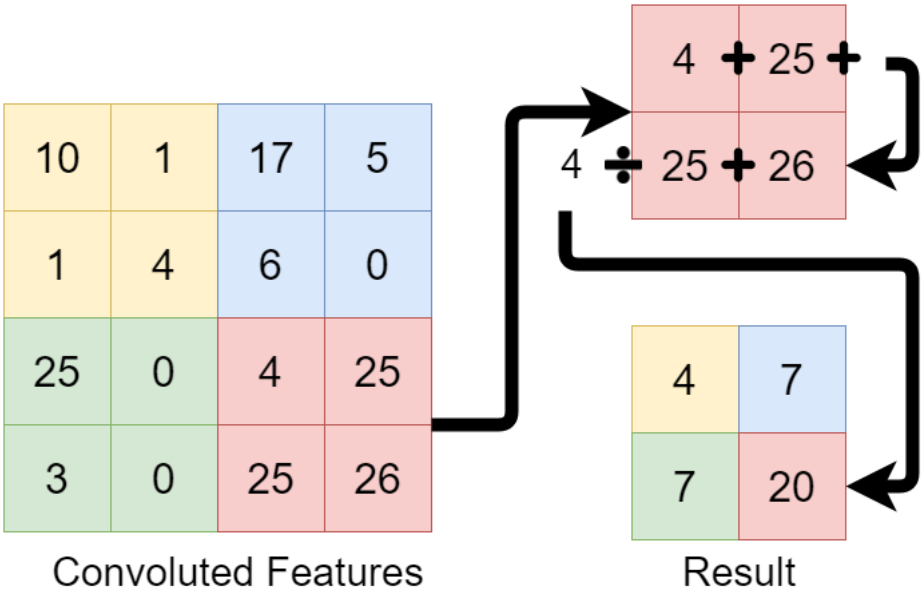

Figure 9: Average pooling on a feature matrix and corresponding result. It takes the input of a segment sums it and divides by the number of segments.

noise better than average pooling by discarding noisy values while average pooling averages them[36].

Fully connected layer

A fully connected layer is a method to flatten the output of a CNN and feed it to a regular feed-forward neural network where training has applied backpropagation for every iteration. This is useful to keep the previous features obtained with the earlier convolutional layers and pooling layers. After N number of epochs the model is now able to be classified with an activation function, see 3.1.1 for different kinds of activation functions.

3.3 Transfer Learning

With low amount of labelled data, or for low amount of data in general training a new model from scratch would usually perform worse than using pre-trained models. To improve the performance transfer learning is a an option where larger data centers already have pre-trained models that can be trained on new data[37]. Supervised training depends on enough

data with ground-truth labels. There is often not enough data with ground-truth labels, or it is to time consuming to label every image in a large data set.

With transfer learning the old network structure is kept with its corresponding weights, learning rates, etc., but the network is trained on new data to better recognise the newly introduced classes. The old neural network might be trained to recognise pictures of animals, cooking utensils or other everyday objects while in this case it needs to recognise medical equipment (and activities). Running predictions on this new network would result in very poor performance, but if the whole network is trained further with a new data set containing labelled training data and verification data the network would be able to recognise the new classes better than a new network could.

There are multiple options for transfer learning. One is to further train the whole network, i.e. all X layers of the network. Other methods is to further train just a couple of the last layers or only the output layer itself[39].

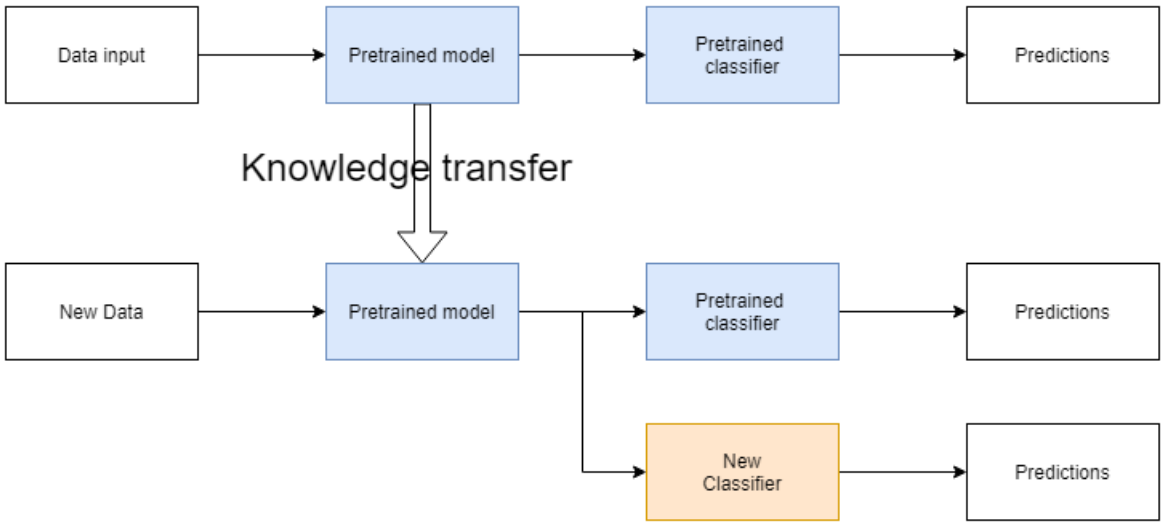

Figure 10: A visualisation on how transfer learning could look like. It is not necessary to introduce a new classifier with transfer learning, but a possibility.

3.4 Object and Activity Detection networks

Object detection networks predicts the certainty that there exist an object of interest in a region of an image. The output is usually a bounding box with a probability of the object being in the box. There is two different detectors for these kinds of networks: One-stage or two-stage detectors. One-stage detectors needs to enumerate several thousand object location candidates which makes them quite inefficient. Two-stage detector on the other hand uses sampling heuristics like fixed foreground-to-background or Online Hard Example Mining (OHEM) to narrow down the candidates for object location[27].

3.4.1 RetinanetV2

RetinaNet is like other recent convolutional neural networks a one-stage detector network. Unlike other one-stage network RetinaNet matched state-of-the-art two-stage methods such as Mask R-CNN and COCO AP. As mentioned at the start of section 3.4 two-stage detectors uses heuristic sampling methods, these exists for one-stage detectors as well, but RetinaNet uses a new loss function to deal with issues like unbalanced classes in the data set. It is a dynamically scaled cross entropy loss function where the scaling factor decays towards zero as the confidence in correct classification rises.

RetinaNet is as mentioned earlier a relative simple one-stage detector with one backbone network and subnetworks that are task-specific. One of the subnetworks performs convolutional bounding box regression while the other performs convolution object classification on the backbone network.

The backbone that has been used is the Feature Pyramid Network (FPN) which is a network that augments a normal convolutional network with a path from top to bottom that constructs a multi-scale feature pyramid. Each level of this pyramid can be used for object detection. This improves the multi-scale prediction for the network. The pyramid is constructed with different levels, P_3 to P_7 with 256 channel and 2^l resolution where l is the pyramid level.

Each of the pyramid levels contains translation-invariant anchor boxes with areas from 32^2 to 512^2 with different aspect ration: 1:1, 1:2, 1:1.

One of the two subnetworks is a classification subnetwork which predicts the probability of an object for each anchor and object class. It is a simple subnetwork which takes in

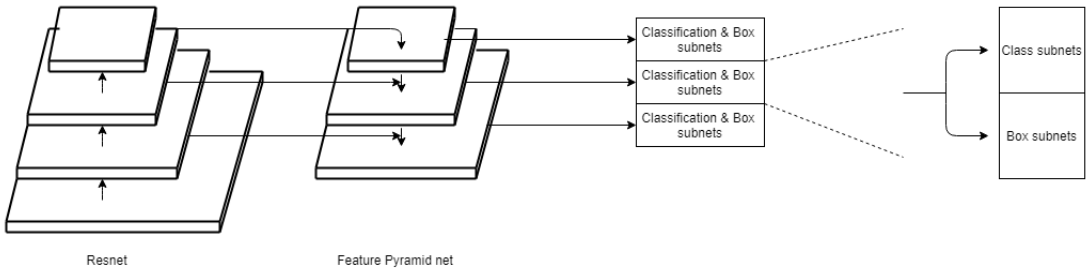

Figure 11: Figure showing the image being separated into a feature pyramid where each layer is processed with both subnets, classification and regression box

a feature map with the 256 channels from a pyramid level, then it applies four 3 x 3 convolutional layers with 256 filters and followed up by a ReLu activation layer and another 3 x 3 convolutional layer with filters denoted by the number of anchors and object classes.

The other subnetwork is the box regression subnetwork. It is a small fully connected network which is connected to each pyramid's mid level. The purpose of this subnetwork is to regress the offset from each of the anchor boxes to a ground-truth object that is close to its proximity. It shares a lot of similarities with the classification subnetwork except for its termination which is a variable number of output defined by the amount of spatial allocation[27].

RetinaNet utilising ResNet-50 architecture with weights trained on ImageNet1k and using a FPN.

3.4.2 Inception I3D

Inception I3D is a so called "inflated" convolutional neural network pre-trained on the ImageNet data set. It ranked as number one in the VU 2017 Charades Challenges[1].

The term inflated bears the meaning that the network originally was a 2D neural network used for image classification network turned into a 3D network and the square filters and pooling kernels were made cubic.

While converting the filters and kernels from square to cubic inflates a pre-trained network bootstrapping the parameters are preferred. Bootstrapping the parameters from the pre-trained 2D network retains important information that can improve performance. The method used for I3D to keep the pre-trained weights was to create a video sequence of the images from ImageNet dataset. These video sequences were just repeated images, but

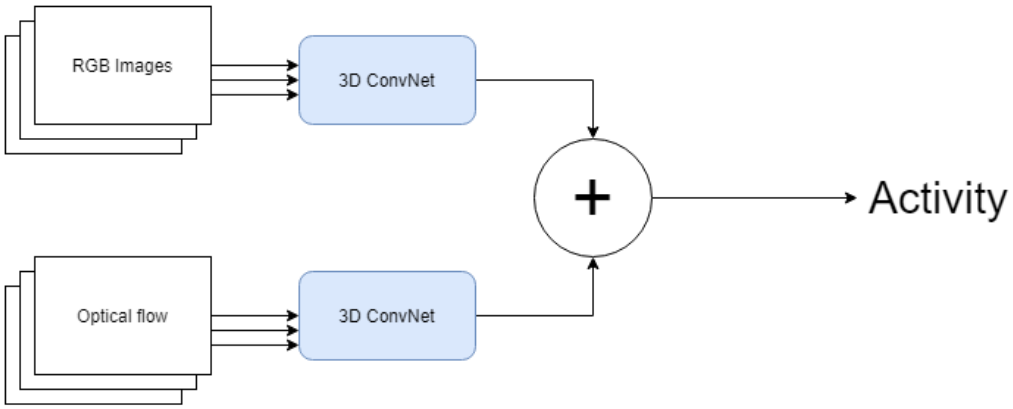

Figure 12: A model for two-stream 3D convolution networks. Multiple images are sent into the 3D convolution network and predictions are concatenated into one output.

by repeating the images N times the weights and parameters from the 2D networks filters could be repeated N times along the new time dimension and normalising them by N. By doing this the convolutional layers should have the same response as in the 2D network[8].

The Inception-v1 convolutional neural network is using two 3D streams for input, RGB and optical flow, this is shown in figure 12. Essentially it is two models, one RGB model and one optical flow model. Each convolutional layer is followed up by a batch normalisation and the activation function used for all parts except the output layer is the Rectified Linear Unit (ReLu) activation function. For the final output the two models are summed and averaged.

The training input of this network has a data augmentation process where the smallest side of the video input were resized to 256 pixels, then crop a randomly chosen 224 x 224 area of the image. For the shorter video clips they were looped, and during this training videos were randomly flipped left to right.

The input used for inference however uses 224 x 224 pixel crops with a center focus. Finally the average of the two predictions from both two models was calculated[8].

3.5 Optical flow

3.5.1 FlowNet2

The original FlowNet was a radical shift away from the traditional methods used to estimate optical flow. This estimation uses a CNN architecture to learn how to estimate optical flow

from data on GPU. The method was a unique idea that had not been used in any previous established work. The method was still sub-par compared to existing methods, but was a promising first implementation. What it did was to resolve smaller displacements and noisy artifacts and increased the performance of action recognition and motion segmentation. This leading to FlowNet 2.0 becoming close to state-of-the-art[21].

FlowNet 2.0 is an evolution of the original FlowNet with modifications. The influence of chosen data set schedules was evaluated, and multiple data sets were added which increased the performance. Alongside the new datasets warping operations was also introduced. This network is also multiple FlowNet networks stacked where the depth, stack and size of components vary resulting in a controllable network where the trade-off between accuracy and computational resources can be chosen. All this computation can be done with different levels of frame rates ranging from 8-140 fps, and can estimate large and small displacements with high level of details.

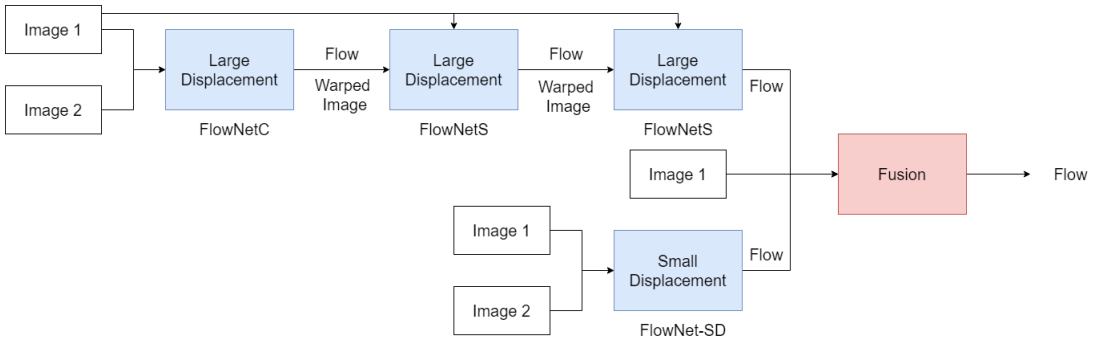

Figure 13: Flownet2 architecture. Multiple networks are estimating the large displacement based on flow and warped images from the previous model and image 1. The smaller displacement is however estimated with a new model trained for smaller displacements. These estimations are fused and flow output are generated

FlowNet 2.0 improved the result with up to 30 % compared to Flownet 1.0 by using the FlowNetC architecture, which includes explicit correlation of feature maps[21]. The original FlowNet was trained on the FlyingChairs dataset which is 22000 images of chairs on random backgrounds and transformed images. This time with FlowNet 2 they got better results with first training on the aforementioned data set FlyingChairs then training on FlyingThings3D a 3D version of the same set. This new training and a new schedule period with smaller learning rate improved the results by a significant amount. Another

method used to improve the end result was stacking of the networks where the first network gets two images as input while the subsequent networks gets the images and the previous flow estimation. Images along with the previous estimation is used to warp the second input image to assess the previous error, and this is then used as input. The last improvement is to handle FlowNets limitations towards estimating smaller displacements. This is solved by creating a new data set similar to UCF101[7] with displacements smaller than 1 pixel. The new data set was based on Chairs, but created to match the characteristics of UCF101.

Overall the FlowNet 2.0 performs comparable to DeepFlow, FlowFields and PCA- Layers, but at a significant lower runtime per frame, in the 100 range of ms compared to 3000 - 30000 range of ms of CPU based methods like DeepFlow, FlowFields and PCA- Layers[21].

3.6 Network backbones

A lot of CNN has a base for extracting features that is based on other work. This is called network backbone. The neural network used in thesis' backbone will be presented in the following section.

3.6.1 RetinaNet backbone - ResNet-50

ResNet is a backbone created by researchers at Microsoft Research and introduced something called residual learning. The network backbone is has many variants based on the number of weight layers. These layers ranges from 18 weight layers up to 152 weight layers. Increasing the depth of the network usually makes the network harder to train, this is why Microsoft Research implemented a residual learning framework to ease the training. When training deeper neural networks and they start to converge a degradation problem was exposed by this team. If the depth was increased the accuracy got saturated and it started to degrade. Residual networks introduces so-called shortcut connections through a single or multiple layers connecting the input to the output of the residual block. With this in place the residual function could be reformulated from approximating $\mathcal{H}(x)$ the underlying mapping to the residual function $\mathcal{F}(x) = \mathcal{H}(x) - x$ to $\mathcal{H}(x) = \mathcal{F}(x) + x$ where $\mathcal{F}(x)$ is the residual function and x is the input[19].

The overall architecture of ResNet-50 is shown in table 1.

[7]UCF101 is a data set with very small displacement between images

Layer name		Output size
conv1	7x7, 64, stride=2	112x112
conv2_x	3x3 max pool, stride=2 $\begin{bmatrix} \text{1x1 , 64} \\ \text{3x3, 64} \\ \text{1x1, 256} \end{bmatrix}$ x3	56x56
conv3_x	$\begin{bmatrix} \text{1x1 , 128} \\ \text{3x3, 128} \\ \text{1x1, 512} \end{bmatrix}$ x4	28x28
conv4_x	$\begin{bmatrix} \text{1x1 , 256} \\ \text{3x3, 256} \\ \text{1x1, 1024} \end{bmatrix}$ x6	14x14
conv5_x	$\begin{bmatrix} \text{1x1 , 512} \\ \text{3x3, 512} \\ \text{1x1, 2048} \end{bmatrix}$ x3	7x7
	Average pool, fc 1000, softmax	1x1

Table 1: A look at the ResNet-50 arcitechture. It is a residual network with the brackets containing the residual building blocks. As the output size downsamples the filter size doubles. The end of the network is using softmax with a 1000-way fully connected network layer.

3.6.2 I3D backbone - Inception-v1

I3D leverages the ImageNet architecture designs and parameters, but can be seen as a evolution moving from the 2D based ImageNet over to the 3D Inception I3D architecture with its two-stream approach. The specified back bone in Inception I3D is the Inception-v1 classification architecture with batch normalization[22].

As explained in the previous section about ResNet-50 and RetinaNet some complications occur with training deeper neural network. Inception-v1 handled this with a different method instead of lowering the learning rate. The input layers were normalised on each mini-batch. Utilising normalisation on each mini-batch the learning parameter could be raised again and the careful initialisation parameters were relaxed. The process of normalisation on each mini is to normalise the mean to zero and the variance to one on each scalar feature independently[22].

$$\hat{x} = \frac{x - E[x]}{\sqrt{(Var[x])}} \tag{14}$$

Overall this improved the ImageNet classification performance to exceed the human accuracy[22].

In table 2 the Inception I3D architecture can be seen. It is similar to the Inception-v1 architecture, but has an extra dimension and a convolution layer as output instead of an average layer.

Layer number	
1	7x7x7 conv, stride=2
2	1x3x3 Max pool, stride=1,2,2
3	1x1x1 conv, stride=1
4	3x3x3 conv stride=1
5	1x3x3 Max pool stride =1,2,2
6 7	Inception Module
8	3x3x3 Max pool stride=2
9 . . 13	Inception Module
14	2x2x2 Max pool stride=2
15 16	Inception Module
17	2x7x7 Avg pool
18	1x1x1 conv

Table 2: A look at the architecture of the inflated Inception V1 network. It is a fully convolutional network with no fully connected layers. The inception module can be seen in figure 14.

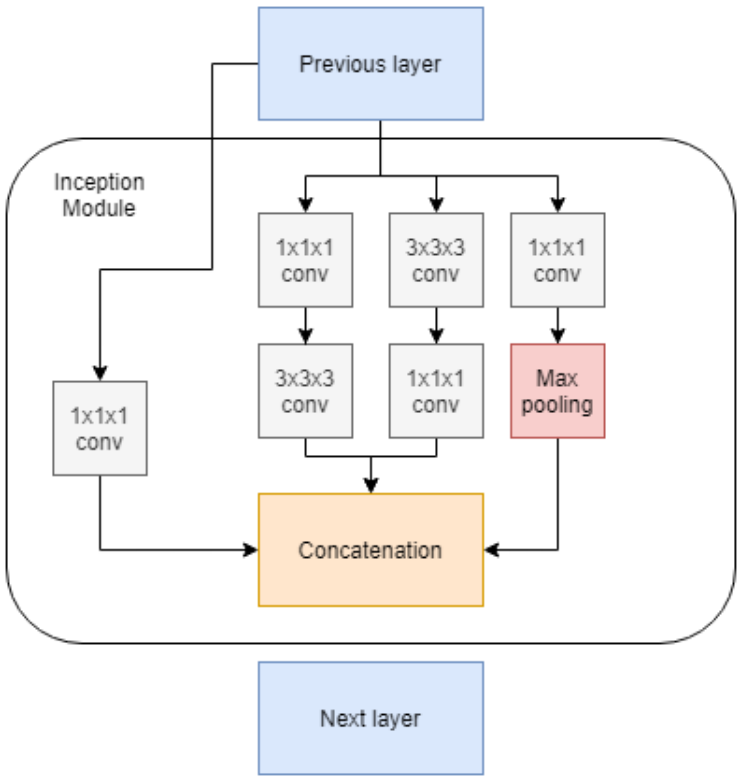

Figure 14: Overview of the inception modules that is used in combination with the convolution and pooling layer. It consist of concatenated smaller convolution networks.

3.6.3 FlowNet2 backbone - FlowNet

FlowNet2's backbone is based on the new FlowNet's architecture. There were created two different architectures, FlowNetSimple and FlowNetCorr. These two network architectures share layers from conv4 layer till output, but differs from start[12].See table 3 for reference.

Layer name	FlowNetS	FlowNetC	Output size
conv1	7x7, 6	$\left[\text{7x7, 3}\right]$ x 2	512x384
conv2	5x5, 64	$\left[\text{5x5, 64}\right]$ x 2	256x192
conv3	5x5, 128	$\left[\text{5x5, 64}\right]$ x 2	128x96
conv_{redir}	NaN	$\left[\text{1x1, 256}\right]$ x 2 corr	128x96
conv3_1	3x3, 256	3x3, 441/32[8]	64x48
conv4	3x3, 256	3x3, 256	64x48
conv4_1	3x3, 512	3x3, 512	32x24
conv5	3x3, 512	3x3, 512	32x24
conv5_1	3x3, 512	3x3, 512	16x12
conv6	3x3, 512	3x3, 512	16x12
	1024	1024	8x6
Refinement			128x96

Table 3: The two FlowNet archictetures. The last layer is a refinementprocess that uses convolution layer with a 5x5 size filter kernel and decreasing depth level down to 64 to increase the resolution.

These two architectures are used in FlowNet2 albeit with a modified training scheduler and since there were some issues with the smaller displacement a new similar architecture was used with a new ensembling architecture shown in table4

[8]The first filter with 441 in depth has is used with the one of the previous two blocks, while the other with 32 depth is the other block

Layer name		Output size
conv0	3x3, 6/64, stride=1	512x384
conv1	3x3, 64/64, stride	256x192
conv1_1	3x3, 64/128, stride=1	256x192
conv2	3x3, 64/128, stride=2	128x96
conv2_1	3x3, 128/128, stride=1	128x96
pr2+loss2	3x3, 128/2, stride=1	128x96
upconv1	4x4, 128/32, stride=2	256x192
rconv1	3x3, 162/32, stride=1	256x192
pr1 +loss1	3x3, 32/2, stride=1	256x192
upconv0	4x4, 32/16, stride=2	512x384
4conv0	3x3, 82/16, stride=1	512x384
pr0+loss0	3x3, 16/2, stride=1	512x384

Table 4: The ensembling of the two network through the FlowNet2 Fusion network. Downsampling the image down to 128x96 by using the last output as its input on the next layer. However the middle layer prx+lossx is only used in the rconvx layers for reconstruction.

3.7 Performance metrics

Depending on the dataset and distribution there exists different metrics to quantify the performance of a neural network. A single metric might not properly quantify the performance of the network so in this study multiple different metrics will be used:

3.7.1 True and false positives

When quantifying the performance of a machine learning algorithms and/or neural networks the prediction can be categorised into two different outputs: True Positives (TP) and False Positives (FP). TP is when the prediction from the model lines up with the true class of the object. FP on the other hand is when the model predicts a positive while the true condition is negative. In object and activity detection TP has to line up with the location of the ground-truth box (and performed activity for activity detection network) to be correctly classified as a TP. If there is no overlap, or the overlap is not large enough the prediction will be classified as a FP.

With object detection and activity recognition networks a threshold is often used. This is because of a significant amount of False Positives with low probability that can skew the overall quantified performance of the network. This threshold is therefore used to decrease the amount of False Positives at the cost of negatively affecting the Recall metric[11].

3.7.2 Precision-Recall

Precision and Recall is two performance metrics that can be used individually or can be used together to calculate the Average Precision (AP).

Precision

Precision is the True Positives (TP) over the sum of all positive predictions. It measures the proportion that is classified correctly when the network makes a prediction[11]. This is used to calculate the networks ability to predict the activity/object, however does not take into account for lost predictions, ie. predictions not detected by the network:

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

Recall

Recall is the TP over the sum of TP and FN. This is often called True Postive Rate (TPR) and measures the proportion of TP that were identified[11]:

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

Neither Precision or Recall takes True Negatives into consideration and isolated can give skewed results.

Average Precision

To further quantify the performance a combination of the two aforementioned metrics are used together to calculate the Average Precision. This metric is given in the Pascal VOC challenge 2012 development kit[15]. Average Precision is calculated with the help of the Precision-Recall (PR)-curve which is the relationship between the Precision and Recall metrics. The goal is to be in the upper right corner of the graph, see figure 3.7.2 for an example on how this curve can look like[11].

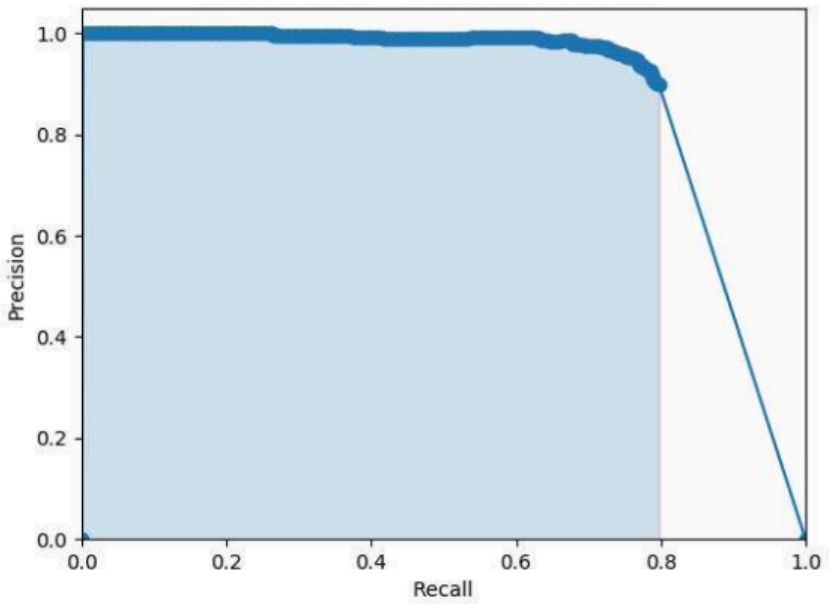

Figure 15: Example of the Precision-Recall curve

Mean Average Precision

mean Average Precision (mAP) is used when there is a multi-class detection and is simply the mean of the AP:

$$mAP = \frac{1}{N} \sum_{n=1}^{N} AP(n) \tag{17}$$

3.7.3 Accuracy

Regarding activity recognition there is need of another way to quantitatively measure the performance. Based on previous work[28] accuracy was chosen as one of the metrics for performance evaluation. However imbalance in the dataset might occur therefore another metric is used as well. Balanced accuracy which is based on the average between True Postive Rate and True Negative Rate (TNR). The TNR formula is the sum of the TN over the sum of TN and FP[6].

$$TNR = \frac{TN}{TN + FP} \tag{18}$$

Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{19}$$

Balanced Accuracy

$$BalancedAccuracy = \frac{TPR + TNR}{2} \tag{20}$$

3.7.4 Intersect over Union

The Intersect over Union (IoU) or commonly called Jaccard index is a statistical metric to determine the similarity of two sample sets. It is used to determine the Ground-truth bounding box against the Predicted bounding box in object detection[38]. In the VOC2012 challenge a valid TP is determined as a 50 % overlap between ground-truth bounding box and predicted bounding box[15]. Mathematically the formula for IoU is the intersect of two rectangles divided by the union.

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{21}$$

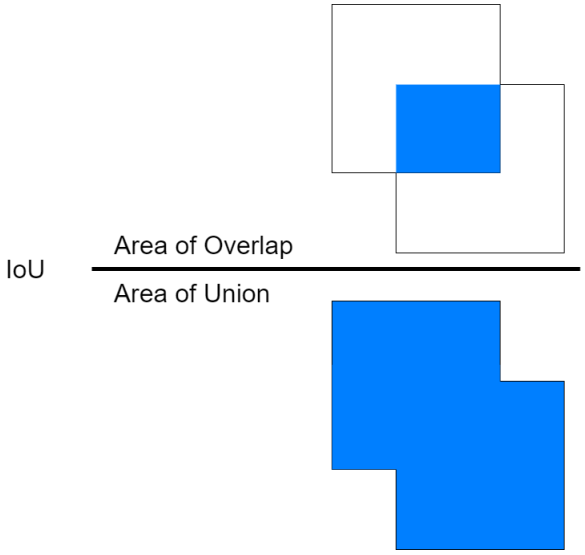

Figure 16: Visualisation of Intersect over Union

3.8 Data augmentation

With Artificial neural network it exists multiple methods of augmenting the data in ways for the network to think the input is different enough from a previous to learn new features or easier to differentiate objects. These methods could be rather simple consisting of blurring, rotating and flipping the image, or most often a combination.

3.9 Pre-Processing

Pre-processing is often needed to enhance the input images/video. Some options are to even the brightness, extracting images features and for video inserting constructed missing images.

An issue with a lot of data is the uneven spread of illumination where certain features of the image might be indistinguishable because of the poor illumination range. A couple of methods to counteract this is histogram equalisation and homomorphic filtering.

3.9.1 Histogram Equalisation

The process of histogram equalisation is to increase the contrast by spreading the grey values over a larger area. Ideally the new histogram should be flat, but this is often not possible. This process increases the contrast and spreading the histogram over a larger scope as shown in figure 17.

The calculations for histogram equalisation can be seen in equation 22 where g_{new} is the new grey value, $p_{old}(g)$ is old image's histogram value, g_{min} is the minimum grey value, g_{old} is the old grey value and G is the grey value range often 256[34].

$$g_{new} = [G \sum_{g=g_{min}}^{g_{old}} p_{old}(g) - 1] \tag{22}$$

3.9.2 Homomorphic Filtering

Another method to pre-process illumination is homomorphic filtering. This process enhances the higher frequencies while suppressing lower frequencies in an image. A stark difference between homomorphic filtering and histogram equalisation is that homomorphic filtering sharpens details and edges while histogram equalisation has a tendency to soften them. Homomorphic filtering separates the multiplicative reflectance function into additive

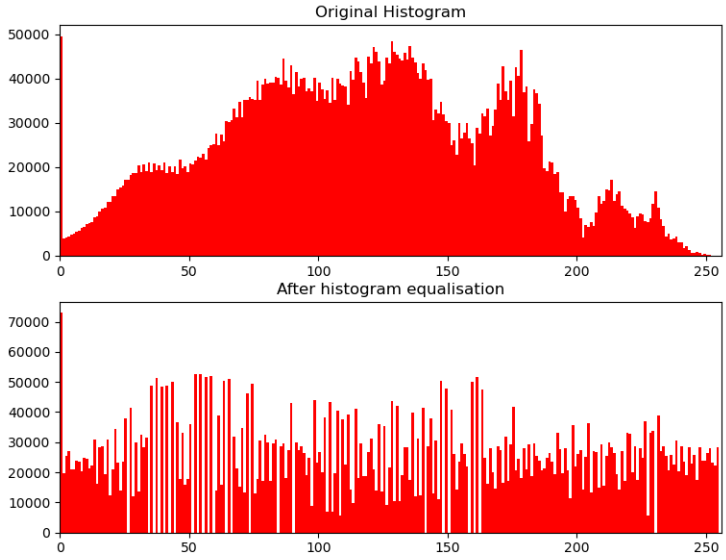

Figure 17: Example of different the grey values in a histogram before and after histogram equalisation. Ideally the lower graph would be a flat line.

by using the logarithm of the image as shown in equation 23 where m is the image, i is illuminance and r is reflectance.

$$m(m,n) = i(m,n) \cdot r(m,n) \tag{23}$$

$$ln\ m(m,n) = ln\ i(m,n) + ln\ r(m,n) \tag{24}$$

$$\mathcal{F}(m(m,n)) = (\mathcal{M}(u,v)) \tag{25}$$

$$\mathcal{G} \cdot \mathcal{M}(u,v) = \mathcal{N}(u,v) \tag{26}$$

$$\mathcal{F}^{-1}(\mathcal{N}(u,v)) = n(m,n) \tag{27}$$

$$\hat{m}(m,n) = e^{n(m,n)} \tag{28}$$

With this done a filter can be applied in the frequency plane and using the exponential function the image can be turned back to an augmented image $\hat{m}$[34].

3.9.3 Edge Detection

Another method to pre-process the image is edge detection where the image is convoluted with one or more filter kernels (same operation as described in section 3.2). Edge detection is used to identify where the image brightness changes sharply either by using the first order, second order derivative or other methods[34].

Sobel Edge detection

The Sobel edge detection method is a gradient based edge detection where two masks are used:

$$G_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \; G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{29}$$

These two masks in equation 29 when applied to a image creates the gradient in x and y orientation. For usage the magnitude of them or angle are needed:

$$G_m = \sqrt{G_x^2 + G_y^2}, \; G_a = tan^{-1}\frac{G_y}{G_x} \tag{30}$$

3.9.4 Video frame interpolation

In some videos the frame rate is unstable or at a lower than required rate. To create new or missing frames a form for interpolation is needed. A method that need low amount of processing power is frame averaging. Frame averaging calculates the distance between frame 1 and frame 2 from the missing frame and creates a weight based on the distance[9]. In equation 31 F_1 and F_2 are the frames, F_{int} is the interpolated frame and d_1 and d_2 are the distances from the existing frames to where the new frame is supposed to be. For example in a 30 Frame per Second video to 60 conversion a new interpolated frame needs to have distance 1 from F_1 and F_2 making both weights 0.5.

$$F_{int} = \frac{d_1}{d_1 + d_2}F_1 + \frac{d_2}{d_1 + d_2}F_2 \tag{31}$$

4 Data material

As mentioned in the introduction data has been collected through the Safer Births project[4].The data material collected are video recordings recorded over the resuscitation-tables while resuscitation is being performed. Most of the data used for this work is from Haydom Tanzania, while new unlabelled data has been provided from Nepal.

4.1 Safer Births

Safer births is development and research project that since 2013 has had a goal to estabilsh knowledge and develop products to reduce neonatal deaths[4]. Through the years they have gathered a large amount of videos from newborn resuscitation. The project is a co-operations between multiple partners such as Laerdal Gloabl Health, Helse Stavanger, Univeristy of Stavanger and Muhimbili and Haydom hospitals located in Tanzania. In these videos different therapeutic activities described in section 2.2 are being performed. These videos depicts Suction, Ventilation and Stimulation activities and these activities are wanted for creating a timeline to evaluate guidelines and if the guidelines are followed.

4.2 Project data

The data used is videos acquired from Haydom in Tanzia in cooperation with the Safer Birth project. Still images from the videos can be seen in the figures 18 - 20. All of these shots differs in camera placement and lighting.

Figure 18: Example on ventilation being performed. Notice the ghosting effect on one of the hands.

In total there are about 500 videos depicting resuscitation of newborn. While there is enough of data to use for image detection and activity recognition a lot of the data acquired is varying in quality and there it lacks a standard configuration (camera placement, resolution, frame rate, etc.) as can be seen in the previously mentioned figure 18 - 20. The resolution varies from 1024x768, 1280x1024 and 1280x720 which in general causes little problems except for more static areas with higher resolution. The problem with this data however is the unstable frame rate. The videos are encoded in 30 or 15 Frames Per Second (FPS), but a substantial amount of the videos have unstable frame rate varying from 2 to 30 fps, and in some cases as mentioned in the article by Meinich-Bache et al[28] the average frame rate is under 5 fps in a large amount of the videos. The unstable frame rate generates motion blur which obfuscates large amount of the frame data which otherwise could have been used for object detection or activity recognition. The positives with this data set is that a large amount of it is labelled data. Training, validating and testing therefore is

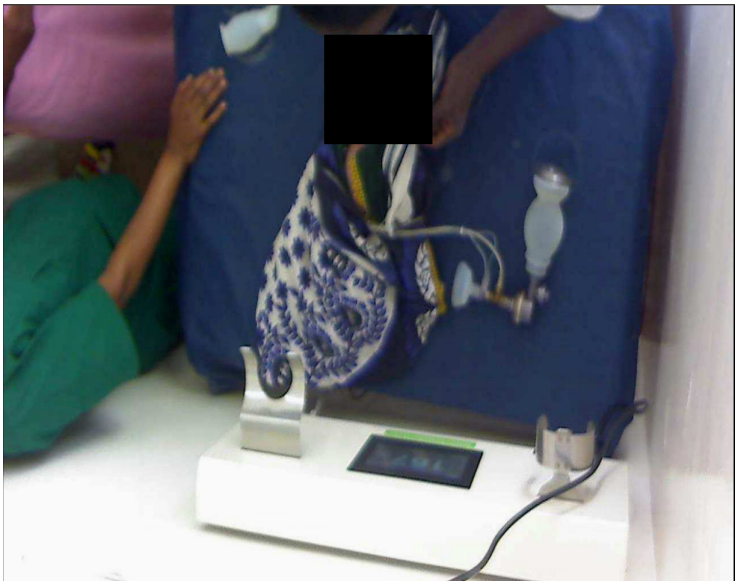

Figure 19: Example of a wrapped newborn after activities has been performed. Very awkward camera angle, but the medical equipment is easy to spot in this image.

Figure 20: Another resuscitation table. It exists heavy motion blur around the person similar to the hand in figure 18.

less time consuming since there is no need for manually labelling data. Overall 76 videos out of the 500 are used in object detection and in activity recognition with the usage of post-processing method such as cropping and making short video clips.

4.2.1 Data

Object set

The activities Stimulation, Ventilation and Suction that is wanted for the timeline all relies on different objects. Stimulation relies on hand motion, Ventilation relies on a Bag Mask Resuscitator (BMR) while Suction relies on a Suction Penguin. Therefore the objects used performing these activities are of interest. Along with these a fourth object the green heart rate monitor which provides useful vital data.

In the object detection training set 12326 of the images are real objects selected from the videos used as data. The last 5000 images in the training data set is synthetic images created by capturing images of the four objects in front of a blue screen. Then using the cropped images on randomised backgrounds. This is to force the network to recognise the objects in more difficult environments and providing more data in general. This has created some issues where there is some leftovers from the blue screen around the objects and that the semi-transparent objects is tinted blue. These two issues can in some cases create unnatural looking images which can trick the neural network.

	Num of images	Resolution
Training set	17326	1024x768, 1024x1280 & 1280x720
Validation set	926	1024x768, 1024x1280 & 1280x720
Test set	996	1024x768, 1024x1280 & 1280x720

Table 5: Information on the Object detection sets

In total there is 4371 objects to detect in the testset consisting of 996 images. The distribution of the different objects are shown in figure 21.

Activity set

Each video is cut into 3 second clip containing 45 frames per input channel and the objects are cropped down to 256x256. There is three different channels: RGB, optical flow x direction and optical flow y direction. Each clip is a fixed resolution 256x256 pixels.

Furthermore all validation and training sets includes their respective class as true labels as well as other classes for indicating that an activity is not being performed. The detailed data is shown in table 7 where all classes except for Stimulation have two sub-categories:

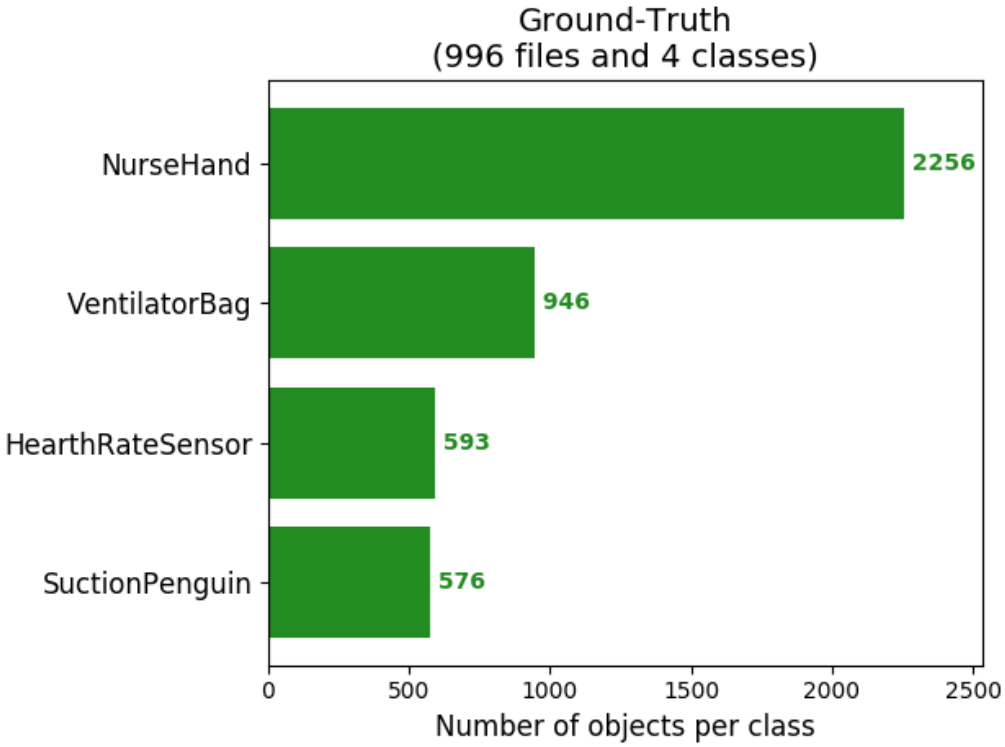

Figure 21: The number of objects to detect per class in the Ground-Truth information for the object detection testset.

Class	Training	Validation	Test
Stimulation	16075	7013	8536
Suction	24493	9972	8536
Ventilation	19505	8355	8536

Table 6: Information on amount of training data used to train these activities. The different activities are described in section 2.2.

Newborn area (NB) or Object area (OBJ).

Not all of these classes are used for training every model, but a mix is used to tell the networks that another activity is being performed. The "Not" classes are clips from before and/or after an activity is being performed. They are chosen to be similar to the action being performed, but contains no active activiy. For example a nurse holding the ventilator up to the newborn, but there is no hand-motion indicating an activity being performed.

Class		Training	Validation
Stimulation	NB	8518	3788
Suction	NB	6229	2703
	OBJ	5941	2452
Ventilation	NB	4990	2140
	OBJ	4724	2024
Attaching ECG	NB	1428	NaN
	OBJ	531	832
Removing ECG	NB	392	NaN
	OBJ	169	243
NotVentilation	NB	1875	1286
	OBJ	1878	1277
NotSuction	NB	2984	806
	OBJ	2999	802

Table 7: Detailed information on each class that is used as datasets

4.2.2 Datasets

	Training	Val	Notes
Object_FullAugSynth	17326	926	Training set consists of 5000 synthetic images. All images is in RGB
Object_FullAugSynth_GGmGa	17326	926	Same as last set, but all images are pre-processed with grayscale, gradient magnitude and gradient angle
Object_FullAugSynth_GGmI	17326	926	These images are pre-processed with greyscale, gradient magnitude and illumination
Activity_Aug_Suction	24493	9972	Clip length of 45 frames with two classes performing activity or not
Activity_Aug_Stimulation	15742	7013	Clip length of 45 frames with two classes performing activity or not
Activity_Aug_Ventilation	19505	8355	Clip length of 45 frames with two classes performing activity or not
Activity_Aug_Multiclass	27631	13404	Clip length of 45 frames with three classes performing the activites Suction, Ventilation or neither
Activity_Flow_Aug_Suction	24493	9972	Clip length of 45 frames with two classes performing activity or not
Activity_Flow_Aug_Ventilation	19505	8355	Clip length of 45 frames with two classes performing activity or not
Activity_Flow_Aug_Stimulation	15742	7013	Clip length of 45 frames with two classes performing activity or not
Activity_Flow_Aug_Stimulation_FlowNet	15732	6674	Clip length of 45 frames, new data generated from previous training labels
Activity_Flow_Aug_Ventilation_FlowNet	17798	7626	Clip length of 43 frames. It is a optical flow dataset converted using Activity_Aug_Ventilation

Table 8: Content information among the different named sets. Aug bears the meaning that the set has augmented images while Synth means there is synthetic data.

Class	Number of content	Notes
Object_FullAugSynth_TestSet	996 images	Contains real images in RGB.
Object_FullAugSynth_TestSet$_{G}GmGa$	996 images	Same basis as last, but pre-processed with GGmGa.
Object_FullAugSynth_TestSet$_{G}GmI$	996 images	Same basis as last, but pre-processed with GGmI.
Suction Penguin_Testset	8536 video clips	Clip length of 45 frames, exists two one set in RGB and one in flow
Bag Mask Resuscitator_Testset	8536 video clips	Clip length of 45 frames, exists two one set in RGB and one in flow
Newborn_Testset	8536 video clips	Clip length of 45 frames, exists two one set in RGB and one in flow
Bag Mask Resuscitator_Testset_Flownet	8536 video clips	Clip length of 45 frames, exists two one set in RGB and one in flow
Newborn_Testset_Flownet	8536 video clips	Clip length of 45 frames, exists two one set in RGB and one in flow

Table 9: Overview of the different test sets.

4.3 Nepal

The data from Nepal is separated into two different sets: High Quality and Low Quality. The low quality dataset is older videos captured and anonymized with a low resolution camera. The resolution of these videos are in the range of 480x240 with missing pixels because of said anonymizing covering a rather large amount of the viewpoint. This data is also new and lacking labelling, but it has a stable 24 FPS which is helpful for activity recognition.

The other dataset from Nepal is a higher quality one. The resolution here is 1920x1080 and the frame rate is a stable 30 FPS. However like the previous data this has some issues like changing camera positions and like the lower quality Nepal data it has black areas for anonymizing.

4.4 Classes of Interest

In general there is different classes that might be of interest. For therapeutic activities four classes are interesting: Nurse Hands, Ventilator Bag, Heart Rate Sensor and Suction Penguin. The different classes that are used is the following:

Nurse Hands

As the name applies and figure 22 this class is nurse hands, both gloved and bare. This is

Figure 22: While nurse hands are not used to make a timeline, they are used to approximate the region of interest.

detected by the object detection network and used to approximate the newborn region in the activity recognition network.

Ventilator Bag

The Ventilator Bag or Bag Mask Resuscitator (BMR) is used for ventilation (see section

Figure 23: Ventilator bag in the holder and laying on the bed

2.2 for information) and is a semi-translucent multi-part device. This is captured by object detection and used for the OBJ-region later in activity recognition. The usage of this device in activity recognition is a distinct pumping activity on the newborn's mouth that is quite unique compared to other activities that can occur on the resuscitation-table.

Heart Rate Sensor

The green Heart Rate Sensor is used to detect vitals from the newborn and send these

Figure 24: Heart rate sensor with its distinct green colour

signals to monitoring device and save the data. Both the shape and colour is unique and provides unique features to the object detection network. In activity recognition it is another

OBJ-region and provides attaching and removing data.

Suction Penguin

Suction Penguin is used for clearing the airways and is another semi-translucent object. As

Figure 25: Suction penguin which is due to its irregular shape and being transparent is difficult to detect

the name implies it can look similar to a penguin. This object's semi-translucent body offers a lot of different scenarios where the colour is different depending on where it is positioned. In activity recognition the activity involves squeezing it in one of the correct position (nose or mouth).

5 Methods

5.1 Earlier work

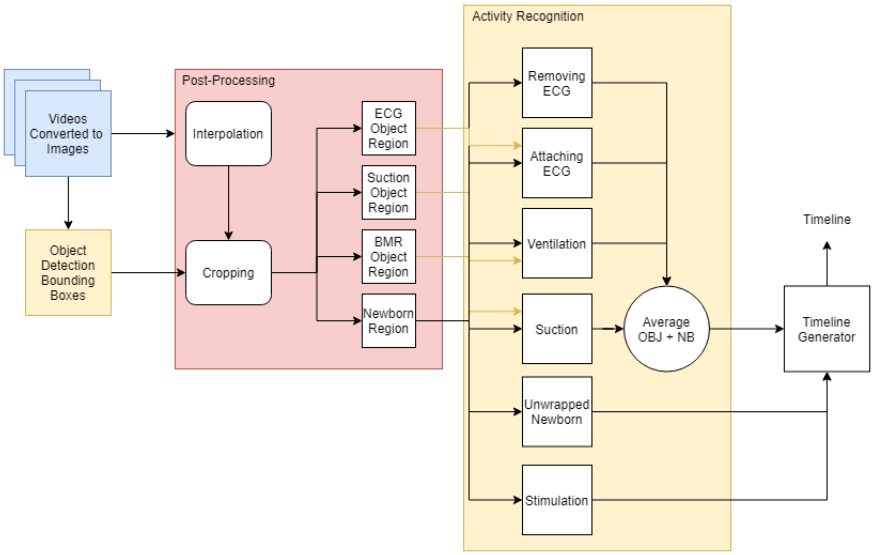

Figure 26: Overview of the old pipeline used in earlier work. All blocks are separate implementations and needs to manually be linked to the next in line. All post-processing is done in Matlab.

The basis of this project started as a task in a PHD and is described in multiple articles[29][28] and work were committed in a previous master thesis[3]. With a given video of newborn resuscitation this pipeline should be able to recognise the objects and detect if there was any resuscitation activity going on. The overview of the previous methods is shown in figure 26. In the previous work the proposed solution was to run object detection to find the region of the different objects of interest, then do some post-processing to convert them into a format that the activity recognition could handle. This was a 3 second overlapping video that was fed to one of several activity recognition models. This resulted in a sequence of probability of the activity happening in that 3 second windows. The following result was then averaged if there were multiple models predicting the same class, and the final output could be put together as a timeline. The previous master thesis[3] focused on improving the object detection part of this network and chose to use RetinaNet since that performed best out of the tested networks. Based on his and Øyvind Meinich-Baches's work both of their results RetinaNet and Inception I3D were worked further on in this thesis.

5.2 Method overview

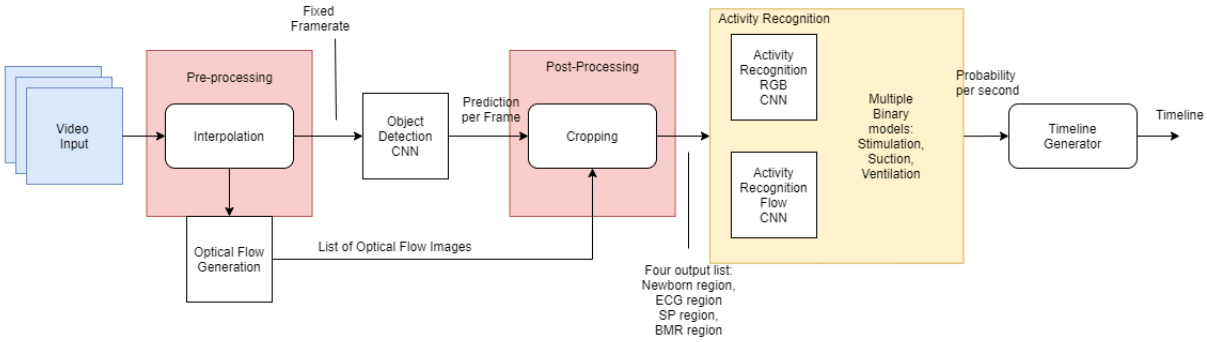

Figure 27: Flowchart on the complete method overview. Videos are used as input for pre-processing which interpolates and calculates optical flow. Object detection is used on the interpolated video and both RGB and optical flow networks are cropped in the post-processing. Finally both crop output are sent to the multiple activity recognition models and a timeline is generated based on the predicted output.

In this chapter, the methods for automated video analysis developed in this master thesis work are presented.

5.3 Pre-processing

A common issue that occurs in the video material from Safer Births is the inconsistency of the camera placement and irregular lighting. This causes the relevant objects and activities to be less recognisable and might even cause objects to blend in with the environment. The uneven lighting with sub-optimal camera placement is shown in figure 28. In regards to object detection there is another issue. Some of the objects have pretty poor detection rate. Synthetic data was added to attempt to expand the training data, but there were some issues with this new data as well. The synthetic data was captured in front of a blue screen resulting in the semi-transparent objects like suction penguin and ventilator bag getting a blue tint. The problem with this discolouration is that it created a distinct difference compared to the real world examples. Also the edge of the objects were imperfectly cut and a slight border around the objects were visible and creating extra features on the object that often does not occur in the real data-set. This can be seen when comparing synthetic and real examples in figure 29. In prior work there has been attempts to solve these issues,

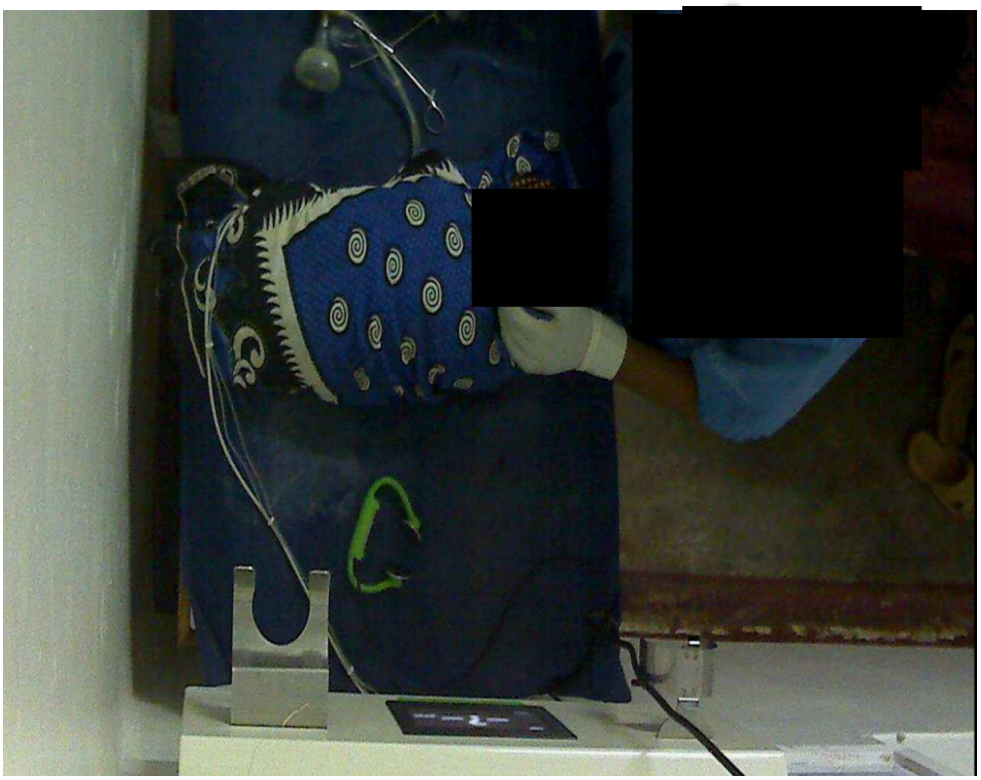

Figure 28: This figure is showing an example of a very poorly lit environment where objects might be hard for the network to recognise correctly.

Figure 29: The object on the left is captured in front of a blue screen then cropped. Along the edge there is a slight blue border and the object is tinted blue. The right image is showing the same object in a real world example.

but the issue still exists in some degree. A couple of pre-processing methods have been chosen to try to rely more on the shape than the colour of the object.

5.3.1 Lighting correction

Two methods have been chosen to try to improve the lighting in the images: histogram equalisation (3.9.1) and homomorphic filtering (3.9.2).

Figure 30: Left image is an unaltered image, the one in the middle has been through histogram equalisation while the right one has been filtered by the homomorphic filter.

Histogram equalisation

Histogram equalisation is pretty simple to apply to an image. The method is applied per channel. Starting by splitting the channels, flatten them and calculating the histogram per channel. Using the cumulative sum to calculate the cumulative distribution function and using this to find the equalised value. The result is shown in the figure 30. The output

brightens up the image, but have a tendency to make them softer and blow up highlights if the image already is bright.

Homomorphic filtering

The homomorphic filtering is done differently. This is also a per channel operation. Like the formula in section 3.9.2 each channel has been altered with the logarithmic formula, then Fourier transformed and applied a filter before inverse transformed and converted to an image again with the exponential function. The result is shown in figure 30. The image has preserved most of the edges and evened out the brightness, however at a cost of overall contrast.

Comparing both methods in figure 30 we can see that the histogram image has higher contrast, but loses its details while being softer to look at. It is also struggling more with the highlights being blown out. The homomorphic filter has a much brighter output, with much sharper edges preventing the highlights to affect the nearby objects to a lesser degree.

5.3.2 Edge detection

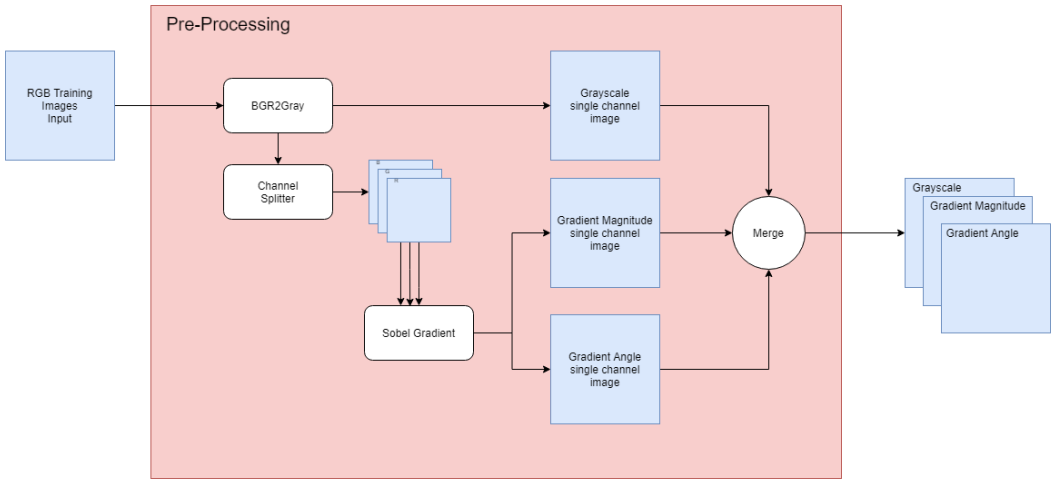

Figure 31: Overview of the image pre-processing procedure. A list of RGB image is taken as input and processed as individual channels. The end result is a three channel modified image.

The edge detection method used is the Sobel edge detection as described in section 3.9.3. Images from training-, validation- and test-set has been transformed by using the

Sobel filter kernels from OpenCV[5], the process is shown in figure 31.This is an alternative pre-processing method not shown in figure 27. Using the filter kernel two images are created, one gradient in x-direction and one in y-direction. With these two calculations performed they can further be used to create a gradient magnitude channel and a gradient angle channel as can be seen in figure 32.

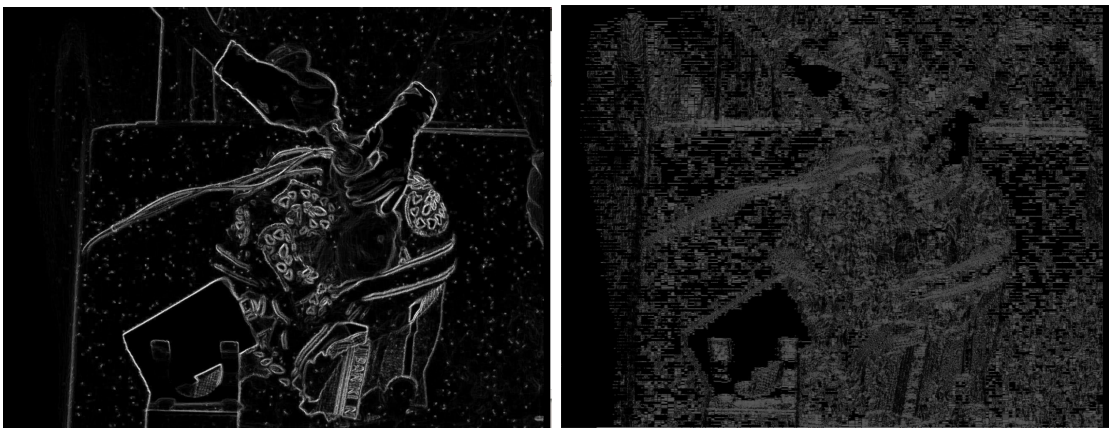

Figure 32: Output generated by using Sobel operator from OpenCV. The left is the gradient magnitude and on the right the gradient angles are shown.

5.3.3 Interpolation

The videos from Tanzania has an issue where the frame rate is unsteady and frame time varies frame to frame. As mentioned in the data material section 4 most of the videos are encoded to 15 frames per second or 30 frames per second, but in reality varies from 2-30 fps[9]. The original interpolation method by Meinich-Bache et al.[28] were written in Matlab, while the new interpolation method that is based on previous work is written in Python, for function reference see figure 33. This new method captures the actual frame time and has a desired frame time configured as default to 15 fps. Since the videos are encoded to 30 fps the interpolation method first interpolates up to 30 fps, then discards every other frame. By using the frame timestamp metadata the difference between the two last frames are calculated, and if it is not 33.33 ms new frames are created by inserting merged frames from the two frames. A weight is used to decide which of the two frames are most important in creating the next frame or frames. For example the distance between

[9]Videos with sub 5 fps were not used for training

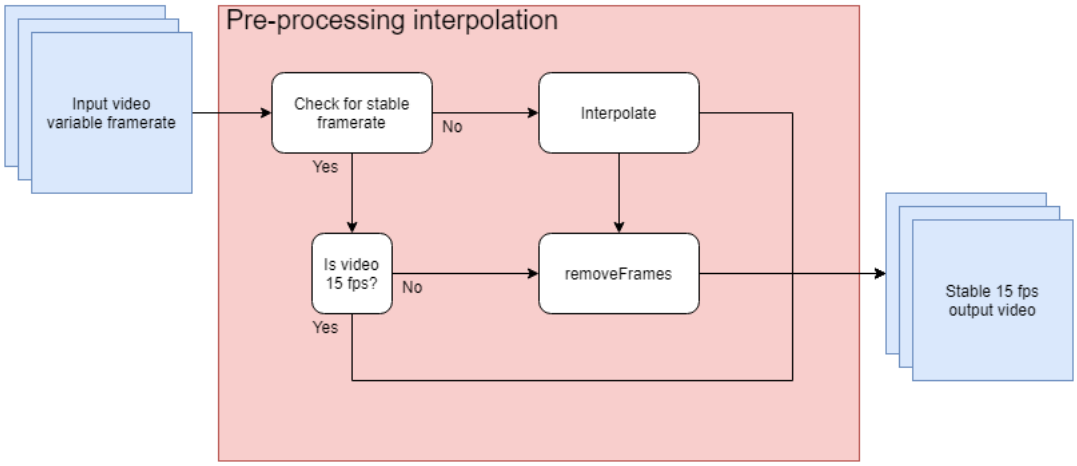

Figure 33: The interpolation block. This implementation consists of three functions. The first checks if there is need of implementation, one that interpolates and one that discards extra frames. Due note that the 30 to 15 fps converter only works if there is a stable frame rate

two frames are 133.33 ms, this means that it is missing three frames. In this case the first image is weighing most in the first interpolated image, then equally weighing and the last interpolated image is weighing the last image most. This is done with all the images in the video. Once this is done every other frame is discarded and a new 15 fps video is created. The downside to this method is that the new interpolated images contains large amount of motion blur as can be seen in figure 34.

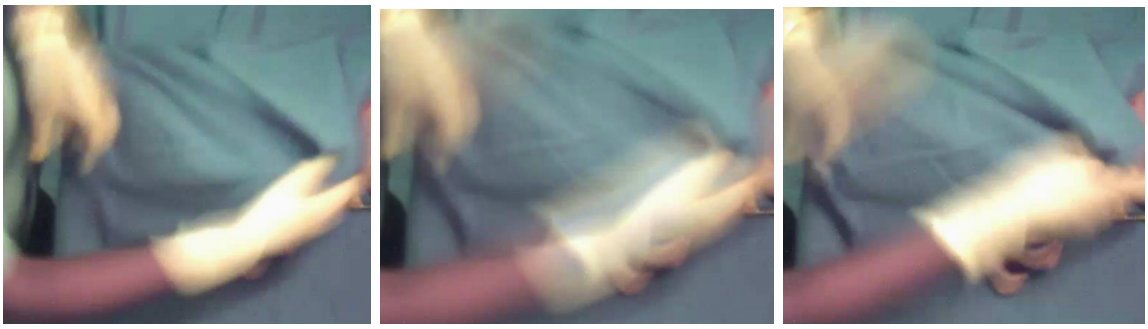

Figure 34: The middle image is the new one created as a new image in-between the two next to it. Notice that the already poor motion blur is getting worse.

5.4 Optical flow

The optical flow method FlowNet2 with the FlowNet backbones described in section 3.6.3 takes two images as input as can be seen in figure 35. Through the usage of multiple CNN it generates something called Middlebury flow which is encoded as byte with the first 0-3 bytes being the magic number 202021.25. This number is used to check if the flo file is valid. Byte 4-7 is width, 8-11 is height, and 12 to the end is the data in vector format where flow can be separated in x or y direction[35]. This network is used for inference only, no further training on this network, but the output was used to generate training data.

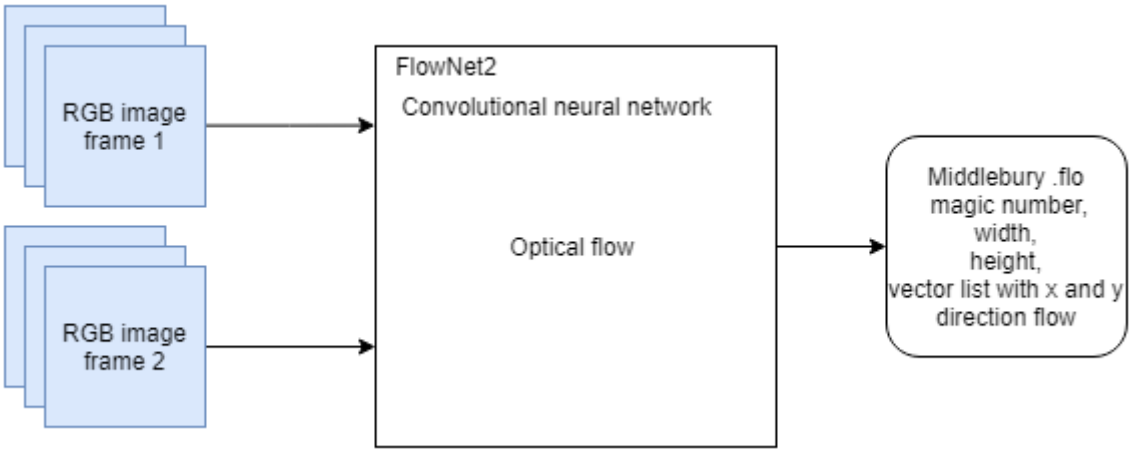

Figure 35: Overview of the input and output of the optical flow method used.

5.5 Object Detection

The object detection model is based on RetinaNet, a CNN with the network backbone ResNet-50. This object detection network was used in previous work by Simon Lennart Austnes[3]. The network was further trained with new data-sets. The input is a three channel image and the output is a bounding box[10], a class and the certainty. This is shown in figure 36. With this system only RetinaNet is tested, but two methods were used. One relying solely on a single three channel input, and the usage of two models relying on different three channel inputs in an ensemble.

[10] A bounding box is two coordinates that marks the coordinates for the lower left and upper right corner of the box.

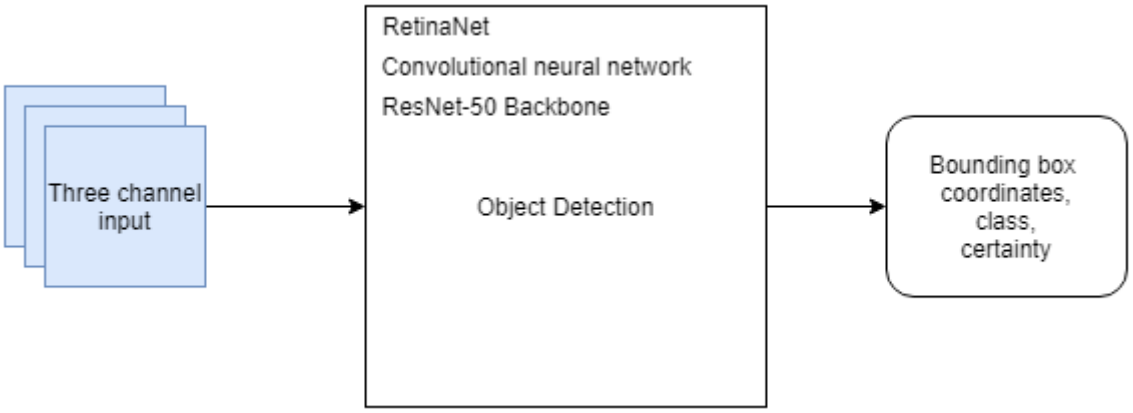

Figure 36: Overview of the Object detection input and output from figure 27

5.5.1 Ensembling networks

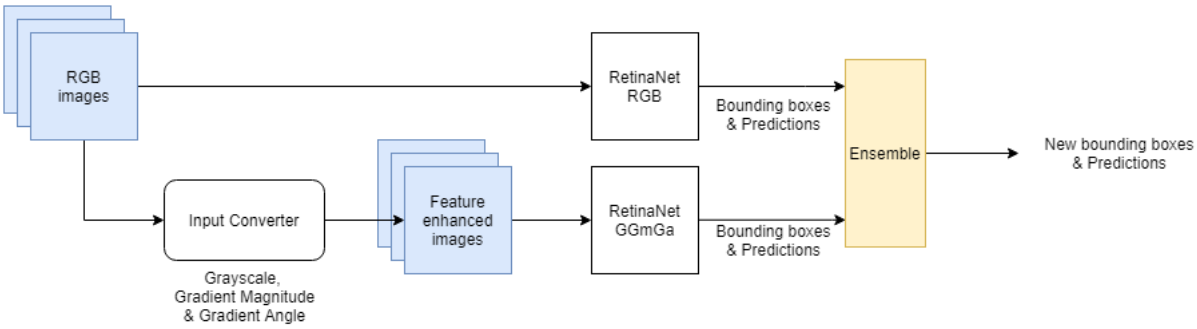

Figure 37: Overview on how the ensemble method works between these two networks. One of the networks uses full RGB images as input, while the other converts them into a different 3-channel image. The input converter block is shown in figure 31.

The usage of multiple RetinaNet models could benefit the system. Therefore two different RetinaNet models were trained and used in conjunction. This method takes the output from both networks which is a bounding box, a class and a prediction. This ensemble part uses a threshold system were if the RGB prediction is above a certain threshold it counts as a valid prediction, then the Greyscale, Gradient magnitude, Gradient Angle (GGmGa) network check for the same prediction with IoU. The process is shown in figure 37 and could be an extra option for the pre-processing block if needed. If the IoU is over 50 % it counts as an already chosen prediction and is discarded. The opposite where the GGmGa model predicts is also possible. If neither models have a strong enough prediction there is another

threshold that checks if the combined output of both network is larger than the average of both models thresholds. If this is true a function uses the two bounding boxes to create a larger bounding box based on the both models output.

5.6 Post-processing

The major focus of the post-processing is to crop down the larger video images down to smaller frames focusing on the region of interest. There is two different kinds of cropping methods in use here. One focusing on the object region, and one focusing on the newborn region as can be seen in figure 38.

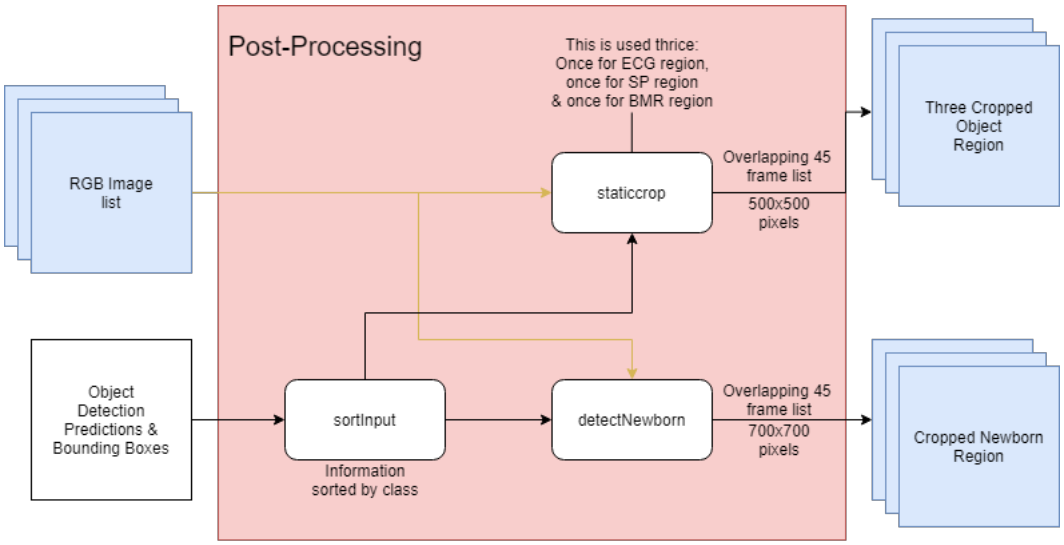

Figure 38: The post-processing implementation sorts the object detection into classes and crops depending on the class. In total there is four different crops that is sent to multiple activity recognition models.

5.6.1 Object Cropping

After the object detection is completed and have returned the bounding boxes and predictions the images goes through a new process were the output is cropped down to 500x500 pixels. This method is based on the previous work done by Meinich-Bache et. al[28]. The cropping looks at a three second interval or 45 FPS. Comparing the whole three second and filters out random spikes that lasts less than three frames. Then it looks at the whole sequence and chooses the largest and smallest values to maximise the region. If the region

is smaller than the predefined sizes the frame is padded with the missing pixels. However if the frame is to close to the border of the frame the crop is shifted in the opposite direction of the border to prevent creating a black bar on the image. In figure 39 the blue bounding box is larger than the object because of the added width and height to make the region 500x500.

Figure 39: Original image on the left is with bounding boxes applied while the image to the right have been cropped based on the boundary boxes. Due note that the original image has been reduced in size to emphasise on the blue bounding box region that has been created.

The figure 39 shows how the crop looks after it has been post-processed. The post-processed image focuses more clearly on the object and discards most of the background where not activity can be performed.

5.6.2 Newborn Cropping

The newborn cropping method does not focus on the same object detection results as the object cropping. This method creates a heat map of the nurse hands and based on the heat map decides the region of the newborn. As with the previous method this is also a continuation of Meinich-Bache et al. work[28]. For every frame with one or more nurse hands detection the bounding box from the prediction is used to crop out the section with the object. This is used to create a binary image that is added to an accumulated image. Adding more and more binary image creates different regions that have more hands which with prior knowledge often indicates that this region contains the newborn. After a number

of frames the heat map now contains enough data to find coordinates used to create a crop. A new crop is proposed if the region is larger than the previous one and at least 45 frames have passed. If the function proposes a new region, the frame coordinates around the area is padded with the missing length to be 700x700, and the same method as above is used if it needs to be shifted because of an image border. Figure 40 shows the heat map and the resulting crop.

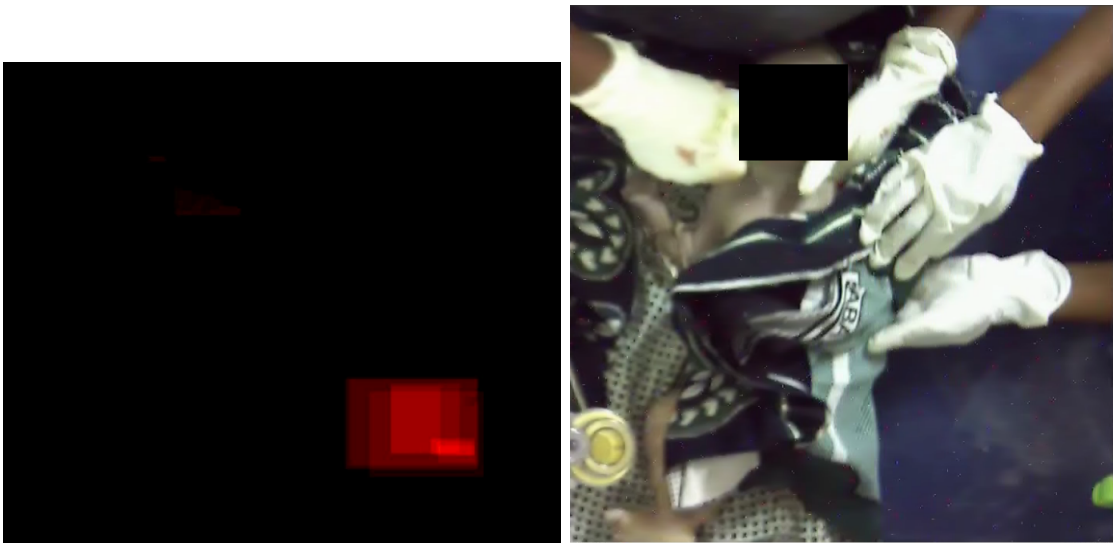

Figure 40: Heat map is on the left and to the right is the final region based on the heat map and padding.

5.7 Activity Recognition

The method used for activity recognition were used with the Convolutional Neural Network Inception I3D that has been used in previous work[28]. Both of the models were further trained with training-data described in section 4. It can use two inputs. One RGB input and one optical flow. With the help of post-processing the input video is cut down in size to either an Object area (OBJ) or a Newborn area (NB). This results in the blocks in figures 41 and 42 called Object Region and Newborn Region. With the original method the activity recognition uses four inputs where four of the activity models are used twice, once for NB region and one for OBJ region hence the two outputs. All of the output is on the form: [second, activity probability] for this method and for the models that is used twice the average of the prediction is calculated.

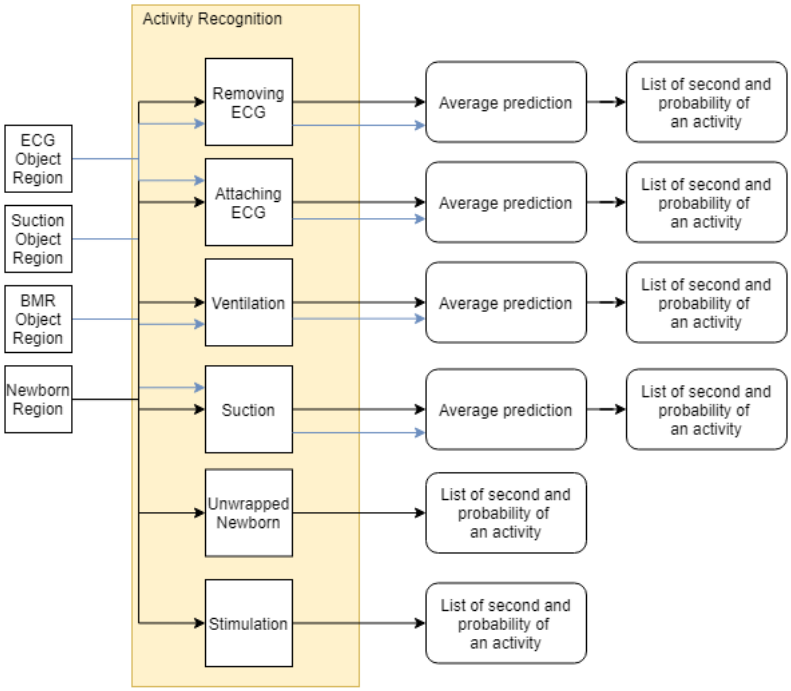

Figure 41: Overview of the original Activity model block. Four of the activities is used twice with one OBJ region and one NB region.

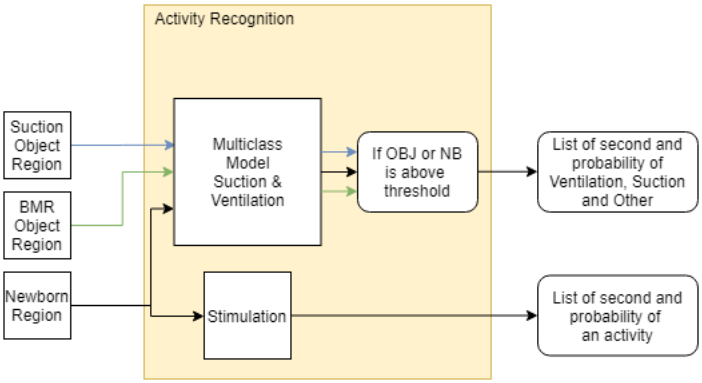

Figure 42: The new method to simplify activity recognition. This method uses a multi-class model for ventilation and suction (in this experimental function ECG classes is not considered), while keeping Stimulation as a binary model. Along with the reduction in models a new thresholding method is applied that instead of averages the OBJ and NB output uses a highest probability (voting) method along with requirement to be over a certain threshold.

The other method for the activity recognition is shown in figure 42. This method uses the same network as the previous, but this time only two models are used. One multi-class detection which takes the same input and is used three times, once for every region inputted to it. Three output is generated from this activity model. Then the OBJ region prediction is sent to a voting with threshold combination along with the newborn region prediction and checks which region returned the highest certainty and if it is above a threshold. This is only performed with the corresponding activity, so if the Ventilation OBJ region have any Suction prediction these are not compared with the Suction predictions in the Newborn region or Suction region. The final outputs to this method is two lists of seconds and activity probability. One is on the same form as the first method while the other is on the form [second, Ventilation probability, Suction probability, other probability] where the "other" probability is activities that is neither Suction or Ventilation and has no further usage.

5.8 Proposed new pipeline

The input data will need to go through a pre-processing phase where features are extracted and/or data is normalised with the help of multiple methods such as histogram equalisation, homomorphic filtering and edge detection. If needed the data will be interpolated up to a stable frame rate. After the interpolation a thread needs to run optical flow calculations separate from the object detection, and it preferably needs to be completed prior to activity recognition. Using RetinaNet for object detection to find the Region of Interest. The predictions RetinaNet outputted will need to be translated and the coordinates from the object detection is used to crop videos into separate smaller crops improving the activity recognition.

Using both RGB and optical flow, the probability of an activity being performed during a three second interval will be the output of the activity recognition network. Finally a final phase will calculate if an activity is being performed if one or both of the regions (NB or OBJ) is above a sat threshold. Afterwards a text file with the activity, time started, time stopped, duration, threshold values and real values will be created.

5.9 Implementation

Common for all of the implementation is that it is written in Python 3. The different libraries and models that are not self designed can be found in table 10.

Function/library name	link
Nvidia's Flownet2-pytorch	https://github.com/NVIDIA/flownet2-pytorch
Keras-Retinanet	https://github.com/fizyr/keras-retinanet
I3D-Tensorflow	https://github.com/LossNAN/I3D-Tensorflow
OpenCV	https://opencv.org/

Table 10: Links to libraries and network used in this thesis

5.9.1 Pre-processing

Image pre-processing

In total there were two implementations in regards to pre-processing the images: colour pre-proceesing with homomorphic filtering and histogram equalisation and colour pre-processing where edge detection was performed. The colour pre-processing implementation consisted of splitting the image into three separate channels (RGB) and performing the processing on a per channel basis. The code used for histogram equalisation was found on TowardsDatasciene[11]and modified to fit in this usage. Homomoprhic Filtering on the other hand uses a heavily modified code from stackexchange[12]where the code is applied per channel.

The colour pre-processed images were done with OpenCv and uses the BGR2GRAY to convert the image to greyscale. The cv2.Sobel function were used to create gradient in x and y direction and the weighted sum of these were used to calculate the gradiant magnitude. For the gradient angle the arcus tangens were used with the gradient y as numerator and gradient x as denominator. The intensity on the other hand was the value channel from the image converted to the HSV colour space with cv2.RGB2HSV. These four channels were then merged to two different image types: Greyscale, Gradient magnitude, Gradient Angle (GGmGa) and Greyscale, Gradient magnitude, Intensity (GGmI).

Interpolation

The interpolation method is as all the other implemented with Python 3. First it checks for uneven frame times, if the frame times frame-to-frame varies, the video needs to be

[12]https://towardsdatascience.com/histogram-equalization-a-simple-way-to-improve-the-contrast-of-your-image-bcd66596d815 and https://dsp.stackexchange.com/questions/42476/homomorphic-filter-python-overflow

interpolated[13]. The implementation takes a video loops through the video with the use of OpenCV's VideoReader function. As explained earlier the method of interpolation is interpolating up to 30 fps and discards every other frame. This is implemented by reading the timestamp of every frame by using the frames metadata and uses OpenCV's AddWeighted function to create the interpolated images.

5.9.2 FlowNet2

The FlowNet2 implementation is used with a pre-trained Caffe model and running inference on the data from section 4. The models used are Caffe pre-trained models and are allowed to use for research by the author[20].

5.9.3 RetinaNet

RetinaNet is using the code from table 10. It is trained, validated and tested on data from table 4. However in the new pipeline implementation a new input was created to let a list of images from memory being implemented into the github code. This new input takes an array of images as input and perform prediction as normal.

5.9.4 Inception I3D

Training and testing Inception I3D is performed as the Readme in the github code in table 4. As with RetinaNet custom input functions were created to allow arrays of images as input instead of image folders. The modified input function can take both RGB and flow images and is augmented and resized with the standard input function.

5.9.5 Post-processing

Cropping object

The cropping implementation takes the previous array of interpolated images and the output from RetinaNet which is a bounding box and a probability. Since the bounding box usually is quite small both sides are padded to make a 500x500 images. The output from RetinaNet can have minor false positives where a couple of frames where the object position

[13]This occurs in basically every video from the given data set.

suddenly "jumps" to another place on the image. To circumvent this the array of images are separated into 45 frames which have a 2 second overlap (one with the previous section and one with future section) and the minimum x and y and maximum x and y is found. However to assure that the maximum or minimum is not a false positive at least three frames in a sequence needs to move more than the threshold of 50 pixels in any direction. The new output is a longer array consisting of 45 frames with a static border made from the minimum and maximum along with padding extra width and height to make a 500x500 crop of the image. This implementation is for objects only. For newborn another implementation was needed.

Cropping newborn

To crop the newborn region of interest only the nurse hands bounding boxes were necessary. As with the object cropping implementation this implementation needs the interpolated images along with the nurse hands predictions. This cropping implementation is designed not to crop a region before there is enough predictions and frames to create a large enough crop of the region. To find this region the bounding boxes are added onto a black image the size of the original video and after a certain amount of frames the new blocks are cut out with the help of OpenCV's findCountors function. The reason behind the usage of this function is that it generates coordinates of the contours as as rectangle even if the contour is an irregular polygon. A small function is added to determine if the right contour is chosen since the function returns an array of contours[14]. If the length of the contour array is larger than 0 it calculates the area of the contour and if it is above 25000 pixels the midsection of the area is calculated. This midsection is used along with a padding function to create a 700x700 size crop where most of the hand activity is contained. A check is implemented to not allow smaller crops to be used once a crop has been chosen and the contours are checked once every 45 frames to save on processing power. The array of cropped images is then loaded to memory and sent along the other crops to the activity recognition network.

[14]This happens when the network either makes a false prediction or hands appear in the frame disconnected from the main area.

5.9.6 New pipeline

The new pipeline implementation is a multi-step implementation with all of the previously mentioned implementation along with modification to most of the neural networks. The following steps are how the implementation works:

Step 1: Check to determine if interpolation is needed
As mentioned in interpolation implementation the first step is to check if the interpolation is needed.

Step 2a: Interpolate
Run the interpolation implementation on the input video and if flow is selected to run the images are saved in a temporary folder.

Step 2b: Run FlowNet2 inference
The newly created temp folder containing all the images from the interpolated implementation is used to generate optical flow with.

Step 3: Object Detection
The interpolated RGB array of images is sent to RetinaNet to be predicted on. This method saves all the object detection in a text file.

Step 4: Sorting the object detection files
Now all the detections are sorted into separate dictionaries with frame number as keys and the bounding boxes along with possibility set to value.

Step 5: Multi-cropping RGB and Flow
A for loop of all the different crop regions needed is then used to run through the cropping implementation for every object selected both RGB and flow (if selected) and once more for the newborn region.

Step 6: Activity Recognition
The next to last step is to predict on all the newly cropped image arrays and writing them into separated text files.

Step 7: Combining output

Since all activities except Stimulation and Unwrapped Newborn requires two regions (object and newborn). these outputs are combined with using a form of voting ensemble where the highest is chosen as an activity or if the summation of both is satisfying a threshold sat to 0.5. The new outputs are six predictions, one for each of the classes.

Step 7.5: Timeline Generation

The timeline generation implementation takes the activity recognition output as a input. Per class it determines that if an activity is above a certain threshold that an activity is being performed that second. With this information an activity is created in the timeline and the start time is set and when the activity is under the threshold a stop time is set. The duration is then calculated and the function is looking for the next time the activity is over the threshold. After one activity is determined the next in line is chosen.

5.10 Generating training data

The data used is described in section 4. Data used in the object detection part was provided by Øyvind Meinich-Bache and Simon Lennart Austnes[28]. Activity recognition data was also provided by Øyvind Meinich-Bache with the exception of optical flow that needed to be created. The process of creating optical flow was to use the interpolated videos that were provided and run inference with FlowNet2. FlowNet2 returns something called Middlebury .flo format which is a byte format that contains a magic number (to check if the bytes are valid), the width, the height and the data. The data contains a vector with x and y orientation were this data is separated into two images. With these optical flow x direction and optical flow y direction files a Matlab script could be used to create short snippets of the actions based on annotations files. After this generation the clips were cut into 45 frame clips and augmentations were applied afterwards. As previously mentioned this implementation differs from Øyvind Meinich-Bache's implementation performed during previous work with the augmentation being the last step instead of optical flow being the last step. Overall this process took up to a 6 days to finish.

6 Experiments

To test out the different improvements and new implementations different experiments have been performed. The different experiments are:

Number	Name	Purpose
Ex 1	Lighting correction	The purpose is to try to improve the overall lighting of the image
Ex 2	Supplementing pre-processed images	Trying to rely less on colour and adding gradient pre-processed input images along RGB
Ex 3	Comparison of pre-processed input	Trying to rely less on colour and training new models with only gradient pre-processed input images
Ex 4	Comparison between RGB model and ensemble	Comparing the RGB model against gradient pre-processed and RGB model in an ensemble
Ex 5	Adapting binary activity recognition models to multi-class	Combining some of the activities into a single model.
Ex 6	FlowNet2 model	Using FlowNet2 CNN to create optical flow input data and training new models with the new data.
Ex 7	Implementing previous work into a single pipeline	Writing methods from previous used work in Python and combining into a single pipeline.

Table 11: Overview of the different experiments performed, and short description.

6.1 Training models

The models trained in this thesis were all trained on a Unix based server running three Nvidia Tesla V100 GPUs with 32 GB of RAM and one Nvidia Tesla P100 with 16 GB of RAM. The following parameters were used when training if nothing else were specified in the experiments section.

Object detection models

Parameter name	Values
Learning rate	0.00001
Epochs	100
Steps per epoch	4329
Batch size	4
Image minimum side	1024
Image maximum side	1280

Table 12: Parameters used in the training of object detection models

Activity recognition models

Parameter name	Values
Learning rate	0.0001
Max steps	15000
Batch size	6
Number of frames per clip	45
Crop size	224
Classics	2/3

Table 13: Parameters used in the training of activity recognition models

6.2 Pre-processing image data

6.2.1 Experiment 1: Lighting correction

Experiment 1 is to find out if lighting correction in the pre-processing stage can affect the performance of the object detection network. Three different RetinaNet object detection models were trained: one with unaltered images, one with images pre-processed with histogram equalisation and one with images pre-processed with homomorphic filtering (both methods were explained in section 5.3.1). The training, validation and test data were pre-processed with the corresponding method and trained using the weights from ResNet50 from ImageNet, with the equipment defined in section 6.1. The data sets used is the Object_FullAugSynth where one is not pre-processed, one pre-processed with histogram equalisation and one with homomorphic filtering, and the parameters used are listed in table 12.

Colour pre-processing

Experiment 2-4 was an attempt to improve the accuracy of harder to find objects such as suction penguin and improve the networks ability to recognise the shapes of objects and not solely rely on colour (for example the bright green heart rate sensor shown in figure 24 back in section 3.8). The same data sets as experiment 1 Object_FullAugSynth was used for the RGB, but here other pre-processes are performed. There is in total four different parts of this category of experiments looking at different aspects. One checking if supplementing pre-processed images can help performance, one with discarding colour images and training with pre-processed images, one looking at new thresholds for the newly created models, and one looking at the combination of the RGB model and one model with pre-processed image input.

6.2.2 Experiment 2: Supplementing pre-processed images

Two new object detection models were trained with converting 75 % of the synthetic training data and 25 % of the real training data into two new image-types with the channels: greyscale, gradient magnitude and gradient angle called Greyscale, Gradient magnitude, Gradient Angle (GGmGa), and greyscale, gradient magnitude and illumination, called Greyscale, Gradient magnitude, Intensity (GGmI). None of the training parameters were

changed from the RGB model in experiment 2. All three models were to be compared with mAP, TP and FP.

6.2.3 Experiment 3: Comparison of pre-processed input

To check if the pre-processing method could provide benefits to the performance of the object detection model, all of the data sets (training, validation and test set) were converted into the the new pre-processed image-types, GGmGa and GGmI. The new sets are called Object_FullAugSynth_GGmGa and Object_FullAugSynth_GGmI. This was done to check if the models performed better on pre-processed image-types than the RGB model did on its test set. Experiment 3 relies less on AP and uses the overall amount of TP and FP together with the mAP as performance metrics.

6.2.4 Experiment 4: Comparison between original model and ensemble

By having multiple object detection models the option to use multiple models as a single network output is possible. To try to improve upon performance an experiment was to ensemble the RGB model and the GGmGa model to test if the overall output could be better with two parallel models versus one. The performance metrics used were the TP and the FP together with overall mAP. With experiment 4 new thresholds had to be created for the GGmGa model. These were created by running predictions on the validation set. The distribution of FP and TP were plotted and a new threshold were used from these results.

6.3 Experiment 5: Adapting binary activity recognition models to multi-class

The previous work and experiments performed by Meinich-Bache et al.[28] in activity recognition focused on multiple binary activity recognition models to predict if there were an activity. While the results were acceptable the process of getting these results were time consuming. Two of the activities, ventilation and suction which cannot be performed at the same time were chosen to be combined to a single model. The new model were trained as a three class model with a ventilation class, a suction class and an "other" class. The other class contains the NotSuction and NotVentilation that was shown in table 7 as well as other activities. Since there were some overlap between Suction and NotVentilation, and Ventilation and NotSuction about 4000 activities were removed. The overall training data

used in this was reduced from 28037[15] down to 24838 while the validation set was reduced from 12032 down to 11432. However due to low amount of "other" activity about 3000 new training examples from Stimulation, Attaching ECG and Removing ECG was added with this label for a new total of 27631 videos in the training set and 13404 in the validation set, making the new data set Activity_Aug_Multiclass. The parameters used for experiment 6 is shown in table 13.

6.4 Experiment 6: FlowNet2 model

Experiment 6 is a comparison between the old Denseflow optical flow trained activity recognition model against the FlowNet2 trained activity recognition model, which uses GPUs to accelerate the speed and improves the overall quality of the calculations. In total the different data sets, training, validation and test set is generated from 76 different videos. In experiment 6 all of the aforementioned videos are converted into optical flow then cropped into smaller clips and augmented with blurring, cropping, rotation, flipping and adding noise. The predictions is then compared to the performance of the models from Meinich-Bache et al.[28]. One distinction that is worth mentioning is that the data from the article[28] where first generated into smaller augmented clips then calculating optical flow on the augmented images while now the augmentation happens as a last step, after optical flow has been calculated. The new Stimulation data set is generated with converting the 76 videos to optical flow, then shorting these down to three seconds overlapping clips of each activity. However due to a mismatch with the newly generated data where the new data contains more short clips the training and validation lists created by Meinich-Bache et al. Could not be used. Therefore this new data set might contain a significant amount of training and validation data with wrong labelling or the activity is behind objects or nurses. The Ventilation model on the other hand is the optical flow converted images from the Activity_Aug_Ventilation data set. Due to these difficulties only Activity_Flow_Aug_Stimulation_FlowNet uses the newly generated data while Activity_Flow_Aug_Ventilation_FlowNet was converted using Activity_Aug_Ventilation. The Suction model was not trained due to time constraints however.

[15]This was the sum of the Suction and Ventilation training set

6.5 Experiment 7: Implementing previous work into a single pipeline

Earlier work (see 5.1 for more in-depth) has focused on prototyping both in Matlab and Python, and improving results. Therefore a pipeline written entirely in Python was to be developed. The new interpolation method is used at the beginning of the pipeline, and the images is loaded into memory afterwards. Object detection is performed with the same network as previous work, but is now performed on the interpolated input sequence then based on the object detection result the new post-processing crops and divides the images into cropped sections that are sent to the activity recognition networks. Overall the system structure is similar to earlier work, but each of the networks have gotten a new data input function. All twenty videos from the test set were used, but due to some difficulties with the server some of the videos could not be read correctly and might cause issues with activity recognition. The result were then compared to the 8536 video clips in the test sets Suction Penguin_Testset, Bag Mask Resuscitator_Testset and Newborn_Testset. In experiment 7 the activity recognition performance between the old pipeline and the new pipeline is compared.

7 Results

7.1 Pre-processing image data

7.1.1 Experiment 1: Lighting correction

To evaluate if lighting correction pre-processing has any effect on object detection performance two new models were trained as explained in section 6.2.1. The results can be seen in table 14.

	No pre-processing	Histogram equalisation	Homomorphic filter
HeartRateSensor AP	0.79	**0.80**	0.78
NurseHand AP	**0.73**	0.70	0.70
VentilatorBag AP	**0.68**	0.66	0.68
SuctionPenguin AP	**0.45**	0.39	0.42
mAP	**65.97** %	63.78 %	64.59 %

Table 14: The difference between the three different models, with or without pre-processed input. Most of the AP is in the same region, but Suction Penguin suffers more with the new pre-processing methods.

The results from the new models are in the same range as the original model that has not been pre-processed or slightly lower performance.

Colour pre-processing

7.1.2 Experiment 2: Supplementing pre-processed images

Trying to supplement the RGB object detection model with pre-processed images. The two new models uses 75 % pre-processed synthetic training data and 25 % pre-processed real training data. The performance of the different models are shown in table 15.

	No pre-processing	GGmI	GGmGa
HeartRateSensor AP	**0.79**	0.69	0.70
NurseHand AP	**0.73**	0.73	0.69
VentilatorBag AP	**0.68**	0.67	0.61
SuctionPenguin AP	**0.45**	0.44	0.40
mAP	**65.97** %	63.07 %	60.22 %

Table 15: AP and mAP result from test set with the three different models. mAP and AP shows the model with no supplemented pre-processed images performing much better than the two others.

	No pre-processing		GGmI		GGmGa	
	Tp	Fp	Tp	Fp	Tp	Fp
NurseHand	**1750**	273	1724	238	1621	**184**
VentilatorBag	**680**	169	669	180	627	**122**
HeartRateSensor	**473**	54	419	79	421	**33**
SuctionPenguin	**266**	63	261	56	236	**36**
Total	**3169**	559	3073	553	2905	**375**

Table 16: The number of false and true positives per class and total for all four classes. While the model with no pre-processing made the most true predictions the overall sum of false predictions are significantly lower on the model supplemented with GGmGa pre-processed images.

At a glance the mAP indicates that the original method without using colour and shape pre-processed images produces the better model. However looking at the TP and FP for

each class the shows some minor reduction in FP and some increases showing that there is potential for improvements with modifying the colour channels.
Overall there is not much difference between the models, but there is some minor improvements in overall FP/TP.

7.1.3 Experiment 3: Comparison of pre-processed input

A comparison between the performance of the new object detection models trained with pre-processed training, validation and test sets and the original RGB trained model is shown in table 17.

	RGB		GGmGa	
	TP	FP	TP	FP
NurseHand	**1750**	**273**	1551	277
VentilatorBag	**680**	169	646	**97**
HeartRateSensor	**473**	54	299	**37**
SuctionPenguin	**266**	**63**	203	66
mAP	65.97%		53.79%	

Table 17: True and False Positives from the models on their respective test sets. The RGB model is performing better in most of the cases, except for a false predictions where in one class the other model has an advantage with fp/tp ratio.

In general this proposed method is weaker than using full coloured RGB images. The newly trained model manages to predict less correct classes and retains similar amount of false positives. Since there is a reduction in false positives in some of the classes a new threshold might be necessary to further evaluate the performance. To find the distributions both models have predicted on their validation sets with no thresholds. In the following figures 43 - 46 the false and true positives per class are shown in a histogram compared against the RGB model.

In general the classes where the pre-processed model has been applied had a lower amount of FP the distribution of FP is in general separated from the TP or more dense in the lower probability ranges.

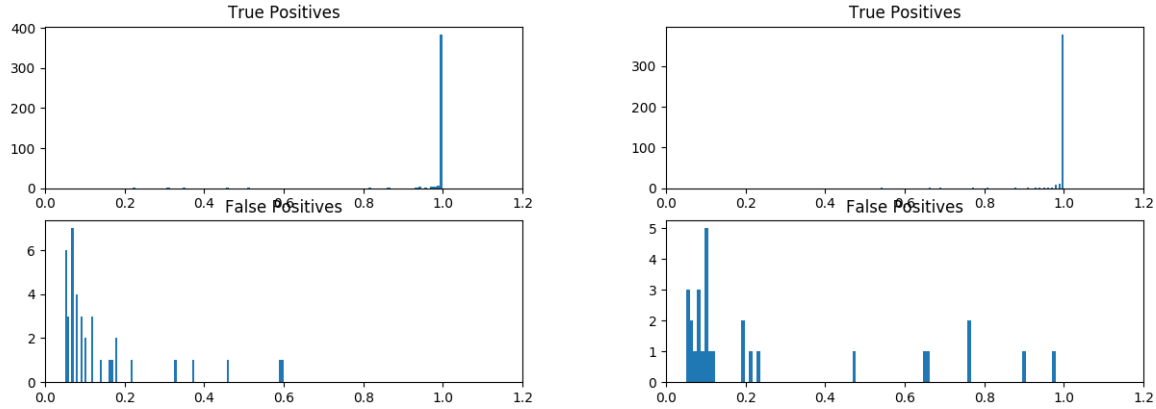

Figure 43: The distribution of true and false positives on the HeartRate class, pre-processed model on the left and RGB model on the right.

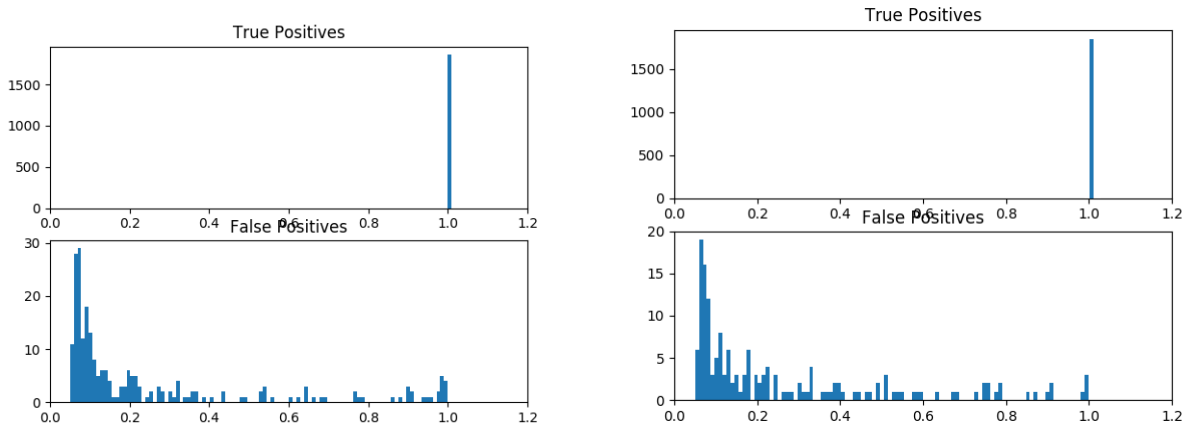

Figure 44: The distribution of true and false positives on the NurseHand class, pre-processed model on the left and RGB model on the right.

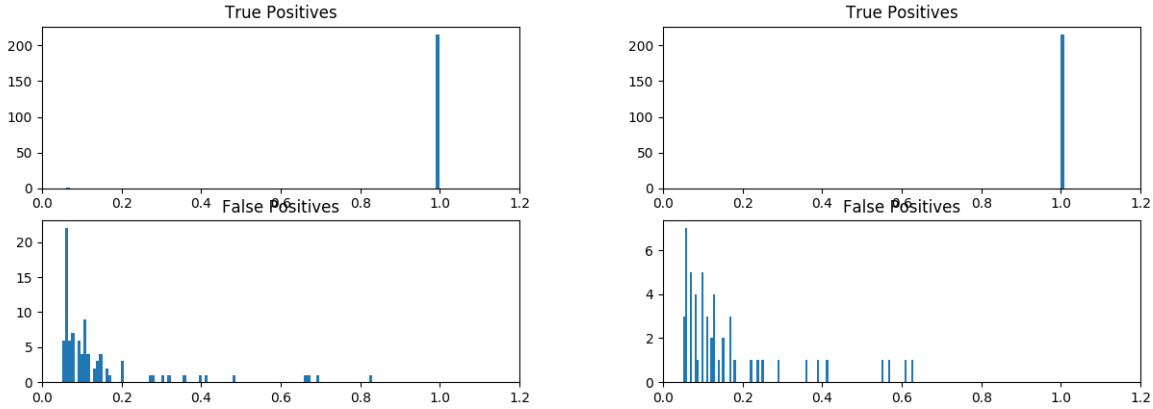

Figure 45: The distribution of true and false positives on the SuctionPenguin class, pre-processed model on the left and RGB model on the right.

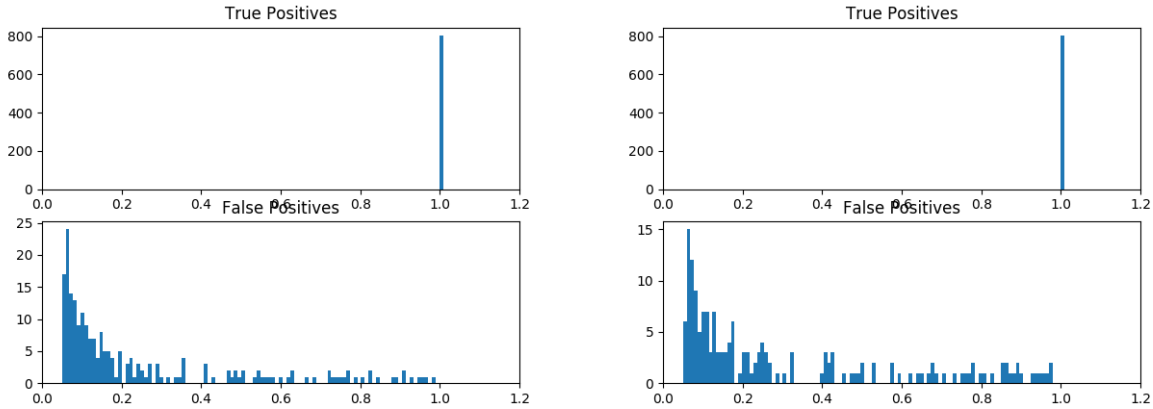

Figure 46: The distribution of true and false positives on the VentilatorBag class, pre-processed model on the left and RGB model on the right.

7.1.4 Experiment 4: Comparison between original model and ensemble

To try to increase performance two object detection models were being used in an ensemble. The models that were being used was the RGB model and the GGmGa model. This ensemble uses new threshold and these thresholds are based on the results from figures 43 - 46. The thresholds in the RGB model is the same as last attempt and is reused from an earlier thesis[3].

	RGB	GGmGa	RGB+GGmGa
NurseHand	0.5	0.99	0.745
VentilatorBag	0.5	0.9	0.7
HeartRateSensor	0.4	0.6	0.5
SuctionPenguin	0.55	0.85	0.7

Table 18: Thresholds used for the different model predictions and another one in case the average of of both thresholds is reached.

	RGB		RGB + GGmGa	
	TP	FP	TP	FP
NurseHand	1750	**273**	**1772**	348
VentilatorBag	**680**	169	676	**91**
HeartRateSensor	473	54	**476**	**50**
SuctionPenguin	266	**63**	**286**	68
mAP	65.97%		65.38%	

Table 19: A comparison of the numbers of total TP and FP per detected per class and the mAP divided into RGB model and the ensemble model RGB + GGmGa.

Overall there seems to be an improvement potential with ensembling two different models with different data set, but there is some inconsistency with what each model is able to correctly assess and overall this ensembling will not be used further. Even so the solution to increase the detection rate of the objects while keeping FP low can be ensembling and develop a more dynamic thresholding method, but due to time constraint this will not be followed up.

7.2 Experiment 5: Adapting binary activity recognition models to multi-class

This new activity recognition model as mentioned in section 6.3 is a combination of Ventilation and Suction activity with a third "other" class that predicts if neither of the activities can be performed. The overall performance of this new model is shown in table 20. The threshold used for this experiment was 0.5 for Ventilation and 0.8 in Suction due to previously poor performance. The thresholding system is a voting system with a static threshold. This means that for the class to predict it needs to be most probable and be over a certain threshold.

	Precision		Recall		Accuracy		Balanced Accuracy	
	Binary	Multi-class	Binary	Multi-class	Binary	Multi-class	Binary	Multi-class
Ventilation NB	0.8059	**0.8082**	**0.7174**	0.6672	**0.9373**	0.7863	**0.8502**	0.6089
Suction NB	0.2972	**0.5021**	0.4544	**0.4903**	0.8624	**0.9155**	0.6262	**0.7284**
Ventilation OBJ	**0.8262**	0.8171	**0.8332**	0.7796	**0.9529**	0.9456	**0.8529**	0.8518
Suction OBJ	0.4710	**0.6307**	**0.4820**	0.4365	0.9101	**0.9305**	0.7130	**0.7954**

Table 20: The performance difference between the Multi-class model and the binary model using the RGB models. Divided into OBJ and NB section.

Overall the common multi class model performs better in some aspects and worse in other.

7.3 Experiment 6: FlowNet2 model

The interpolated data from previous work were converted into flow with the new method and then predicted on the converted test set. To get the comparison environment as equal the optical flow activity recognition models were trained as binary models in Stimulation and Ventilation similar to the previous activity recognition models trained with Denseflow data. The Stimulation FlowNet2 model uses the data set Activity_Flow_Aug_Stimulation while the Ventilation FlowNet2 model uses Activity_Flow_Aug_Ventilation.

Figure 47 shows the difference between FlowNet trained predictions and Denseflow trained predictions while figure 47 compares the output of the different functions.

	Precision		Recall		Accuracy		Balanced Accuracy	
	Denseflow	FlowNet2	Denseflow	FlowNet2	Denseflow	FlowNet2	Denseflow	FlowNet2
Ventilation OBJ	**0.6495**	0.4876	**0.7728**	0.7515	**0.9113**	0.8571	**0.7664**	0.6834
Ventilation NB	**0.6729**	0.3515	**0.7055**	0.6587	**0.9123**	0.7857	**0.7823**	0.6180
Stimulation	**0.6744**	0.6484	**0.7973**	0.5940	**0.8916**	0.8657	0.7548	**0.7610**

Table 21: The overall performance of the activity recognition models using flow. Denseflow is the old method while FlowNet2 is the new one.

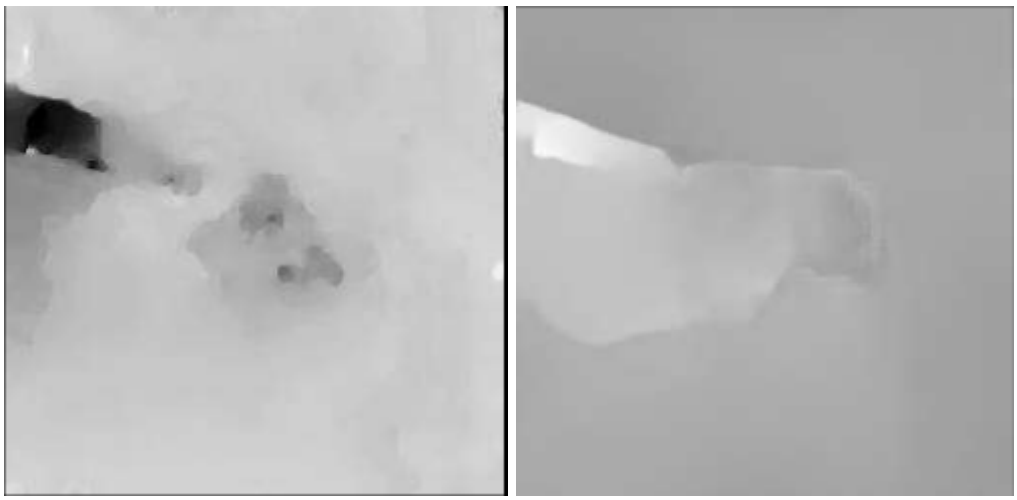

Figure 47: Comparison between Denseflow (left) and FlowNet2 (right) generated on hand movement. The right hand side shows a much cleaner image with less noise and the hand is much more visible in this image.

7.4 Experiment 7: Implementing previous work into a single pipeline

The results shown in table 22 is showing the performance of the activity recognition after the pipeline has processed the test-videos. The old pipeline column is using the models from Meinich-Bache et al.[28] and is predicted on the equivalent region.

The performance from the new pipeline are substantially worse in almost all metrics. Looking at figures 48 and 49 indicates that the network struggles to predict.

Some of the cropped images from the pipeline are shown in figures 50-52 for a reference on how the final output looks like.

	Precision		Recall		Accuracy		Balanced Accuracy	
	Old Pipeline	New Pipeline	Old Pipeline	New Pipeline	Old Pipeline	New Pipeline	Old Pipeline	New Pipeline
Ventilation NB	0.8059	**0.8908**	**0.7174**	0.2638	**0.9373**	0.8942	0.8059	**0.9251**
Ventilation OBJ	**0.8262**	0.7803	**0.8332**	0.1149	**0.9529**	0.8737	0.8529	**0.8811**
Suction NB	0.2972	**0.3563**	**0.4544**	0.1285	0.8626	**0.9064**	0.6262	**0.6722**
Suction OBJ	0.4710	**0.6387**	**0.4820**	0.1050	0.9101	**0.9190**	0.7130	**0.8145**
Stimulation	**0.7548**	0.6627	**0.7313**	0.5667	**0.9066**	0.8669	0.7548	**0.7711**

Table 22: The overall performance of the activity recognition comparing the new pipeline against the old one.

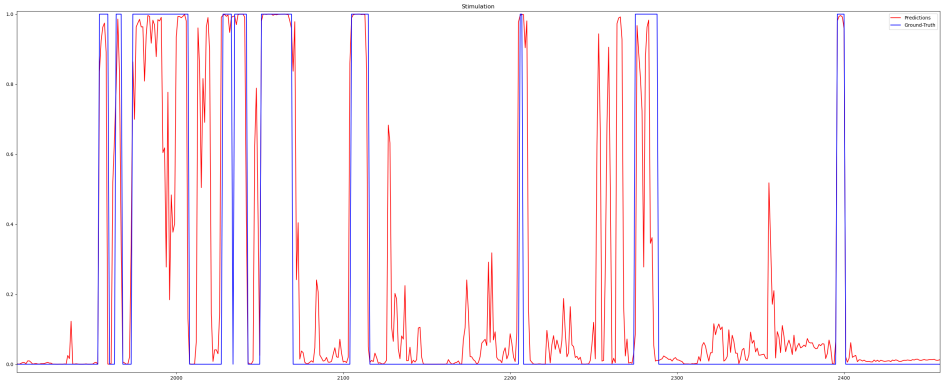

Figure 48: This shows the Stimulation predictions from the pipeline versus the ground-truth.

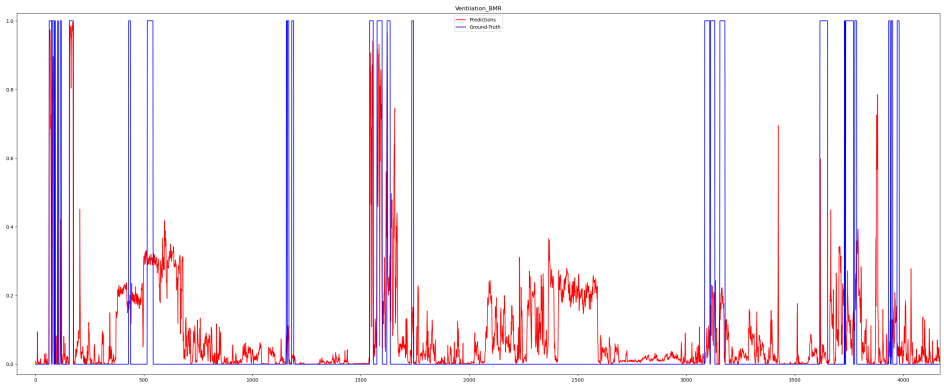

Figure 49: This shows the Ventilation predictions on the OBJ area. This graph is showing a longer timeframe than in figure 48, and shows some poor predictions on this region.

Figure 50: The newborn crop region. The whole newborn is showing, but in this example is located a bit to far on the left side of the image.

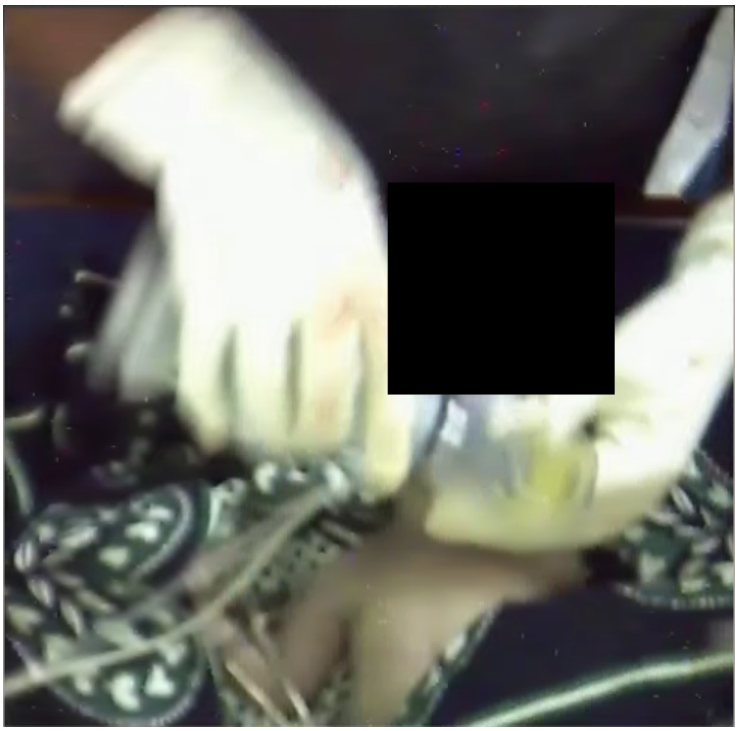

Figure 51: The Ventilator Bag crop region. The Ventilator bag is in the middle of the image and it should not be hard for the network to recognise any activities being performed.

Figure 52: The Suction Penguin crop region. Unfortunately the object is in the upper area of the image, but the whole object, the newborn and the hands are visible, and like the previous image should not offer difficulties for the network.

8 Discussion

8.1 Pre-processing

8.1.1 Experiment 1: Lighting correction

The overall performance of the different models are in the same range as the RGB model. Overall the performance impact is slightly negative in most cases except for heart rate sensor in the data set pre-processed by histogram equalisation. Looking closer on the actual images used for training the reason why they perform worse might be because some of the objects disappears when the lightning is adjusted. Another potential issue might be the thresholds used, it might be too low and allowing more FP than necessary. With more time editing the training data and evaluating threshold could improve the performance.

Looking at an example in figure 53 homomorphic and the histogram equalisation blows up certain highlights in the image making some objects blend in with the environment.

Figure 53: Comparison between the original image (left) and pre-processed image (right) by using histogram equalisation. The adjustments to the brightness makes the Ventilator Bag blend into to the hand making it hard to determine where the hand and Ventilator Bag starts.

8.1.2 Experiment 2-4: Colour pre-processing

Supplementing the network with pre-processed images did not make a huge difference, most of the results were around the margin of error, however the GGmGa trained model that performed worse in number of predictions actually managed to cut a large amount of false

predictions. The overall fp/tp ratio went down from 0.1764 to 0.1291. However this did not give any clear answers so the next part of the experiment was to look at a comparison between the new model with the least amount of false predictions and the regular, but this time the new GGmGa trained model were fully trained with pre-processed training, validation and test data. The overall performance decreased significantly compared to the regular one, but again it kept a lower amount of false positives, and when looking at the distribution of the false positives and true positives in the figures in section 7.1.3 the false positives is quite separated compared to the true positives, except for nurse hands. This could mean that it is possible to get a lower fp/tp ratio. Combining this with the regular model in an ensemble as shown in table 19 reduced a significant amount of false positives while retaining the high number of true positives. The only exception was the nurse hands that got significantly worse and would be advisable to discard most of the predictions from the GGmGa model. With a better optimised model the ensemble of the RGB and a pre-processed could improve the overall performance of the model even further.

8.2 Experiment 5: Adapting binary activity recognition models to multi-class

The overall performance of this new multi-class model is showing potential, but the gain in the Suction class is lost in the loss in the Ventilation class. This gain and loss can be contributed to the thresholds used or the training data. Since there is a positive effect on the Suction activity there seems to be an indication that reevaluating the training data could have something to do with the performance. This is especially true since the reasoning to add extra data to the "other" class was to improve overall performance. With more time a better threshold along with a rework of the training data could improve the performance even further.

8.3 Experiment 6: FlowNet2 model

Due to the issues with training data generation in the Stimulation model the results were poorer than expected. The precision is close, but the loss in recall is substantial. There is no clear reason for the new optical flow method to provide a poorer result when it outputs a cleaner output with higher contrast compared with Denseflow. The Ventilation activity model were trained using optical flow data created from already cropped and augmented

short clips. This calculation resulted in less than 45 images per clips that might have caused some issues with the network. This model also performed worse than its Denseflow counterpart with overall worse performance. Unfortunately due to limited time Ventilation (and Suction) were not possible to train and test with the same data as the Stimulation model.

With proper training data the better performance might be achievable, but due to time constraints and the mismatch in training data generation and labelled data the performance of the FlowNet2 models dropped significantly. This drop in performance could also be caused by the Inception I3D models being pre-trained on a completely different flow method that could not take usage of the new flow data with such a low amount of training data.

8.4 Experiment 7: Implementing previous work into a single pipeline

Looking at the results from the new pipeline most results show a significant loss in performance. As explained in the experiment part there were some late stage issues were certain videos would not load correctly into memory. An attempt were made to manage to load these, but at the cost of sending the full frame instead of a cropped region if the error was encountered. Looking at some of the crops from the object areas in figure 51 the whole object is visible and the network should not have any issues recognising the activity. The newborn region on the other hand might struggle with Ventilation and Suction if the rest of the 19 videos have the same positioning as figure 50, where the newborn is positioned at the left border. If that holds true the ventilator bag and suction penguin would be partially hidden by the crop. When looking at the results using a static threshold at 0.5 it seems to allow more FN than FP resulting in a very high precision, but incredible low recall. With more time a per activity threshold could be used for the new pipeline that most likely would improve the precision to recall ratio, but even with a better ratio the performance would still be worse than the old models. Looking closer at the predictions in figures 48 and 49 it shows that the Ventilation network is struggling more than the Stimulation network, where in some cases the Ventilation network is not predicting at all for 100s of seconds. This could mean there is something wrong with the cropping implementation as well.

Even though the new pipeline tried to perform the same post-processing as the earlier models it might have changed the input image just enough for the network not to recognise the new activities, and generating new training data with the new pipeline could also im-

prove the overall performance of the system. However due to time constraints and focusing on fixing the bugs this was not possible, but there is a possibility in future work to train new training data, add new thresholds or figure out why there is issues with certain videos in the test set and them not lining up perfectly with the ground-truth data. Fixing these issues could improve the system significantly.

9 Conclusion

While most of the pre-processing methods and implementation did not see much changes to the overall performance of the object detection network ensembling two different input models did improve most of the classes. It is still in need of fine-tuning considering the worse performance of the hands, but the overall performance is promising. Converting the activity models Ventilation and Suction to a new multi-class model also showed some promising results. Suction did get a slight improvement in performance while Ventilation decreased in performance. Overall this new model depends on training data and with more time and manual verification with adding better examples of the Ventilation could prevent this loss in performance. While the loss in performance is not staggering with FlowNet2 there still is substantial loss. However this one is most likely due to the issue with having to use poor training data or that the FlowNet2 cannot be utilised properly with the pre-trained Inception I3D models. Normally converting the training set to the new flow implementation would allow usage of the previous training lists, but this did not work out. However this data is much faster to calculate and does create cleaner looking optical flow calculations which can be used at a later stage to train new activity recognition models with optical flow. With the new pipeline method it is clear that there are some errors present in this implementation. The low recall indicates that the activity recognition part seldom predicts an activity. Looking at the cropped images there is not an obvious reason for the large loss in performance. It is suspected that there might be some issues with the order of the test set used not matching completely with the ground truth, but due to time constraints these errors were not found.

10 Bibliography

References

[1] The Allen Institute for Artificial Intelligence. *Charades Challenge*. 2017. URL: http://vuchallenge.org/charades.html (visited on 03/17/2020).

[2] Hafiz Muhammad Aslam et al. "Risk factors of birth asphyxia". In: *Italian journal of pediatrics* 40.1 (2014), p. 94.

[3] Simon Lennart Austnes. "Optimization of Object Detection in Newborn Resuscitation Video". MA thesis. University of Stavanger, 2019.

[4] Safer Births. *Safer Births — About*. URL: http://www.saferbirths.com/about/ (visited on 06/09/2020).

[5] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[6] Kay Henning Brodersen et al. "The balanced accuracy and its posterior distribution". In: *2010 20th International Conference on Pattern Recognition*. IEEE. 2010, pp. 3121–3124.

[7] Zhaowei Cai and Nuno Vasconcelos. "Cascade r-cnn: Delving into high quality object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6154–6162.

[8] Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[9] Roberto Castagno, Petri Haavisto, and Giovanni Ramponi. "A method for motion adaptive frame rate up-conversion". In: *IEEE Transactions on circuits and Systems for Video Technology* 6.5 (1996), pp. 436–446.

[10] Nagesh Singh Chauhan. *Introduction to Artificial Neural Networks(ANN)*. Medium. Library Catalog: towardsdatascience.com. Oct. 10, 2019. URL: https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9 (visited on 04/30/2020).

[11] Jesse Davis and Mark Goadrich. “The relationship between Precision-Recall and ROC curves”. In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 233–240.

[12] Alexey Dosovitskiy et al. “Flownet: Learning optical flow with convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2758–2766.

[13] Rob A Dunne and Norm A Campbell. “On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function”. In: *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*. Vol. 181. Citeseer. 1997, p. 185.

[14] IL: American Academy of Pediatrics Elk Grove Village. “Guide for Implementation of Helping Babies Breathe® (HBB): Strengthening neonatal resuscitation in sustainable programs of essential newborn care.” In: (2011).

[15] Mark Everingham and John Winn. “The PASCAL visual object classes challenge 2012 (VOC2012) development kit”. In: *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep* (2011).

[16] Kunihiko Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[17] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[19] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[20] E. Ilg et al. “FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017. URL: `http://lmb.informatik.uni-freiburg.de//Publications/2017/IMKDB17`.

[21] Eddy Ilg et al. “FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks”. In: *arXiv:1612.01925 [cs]* (Dec. 2016). arXiv: 1612.01925 version: 1. URL: http://arxiv.org/abs/1612.01925 (visited on 02/18/2020).

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

[23] Bekir Karlik and A Vehbi Olgac. “Performance analysis of various activation functions in generalized MLP architectures of neural networks”. In: *International Journal of Artificial Intelligence and Expert Systems* 1.4 (2011), pp. 111–122.

[24] William J Keenan et al. “Delivery and Immediate Neonatal Care”. In: *Pediatric Education in Disasters Manual. American Academy of Pediatrics, Buenos Aires, Argentina* (2009), pp. 217–237.

[25] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[26] Joy E Lawn et al. “4 million neonatal deaths: when? Where? Why?” In: *The lancet* 365.9462 (2005), pp. 891–900.

[27] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[28] Ø. Meinich-Bache et al. “Activity Recognition from Newborn Resuscitation Videos”. In: *IEEE Journal of Biomedical and Health Informatics* (2020), pp. 1–1.

[29] Ø. Meinich-Bache et al. “Object Detection During Newborn Resuscitation Activities”. In: *IEEE Journal of Biomedical and Health Informatics* 24.3 (2020), pp. 796–803.

[30] David J Montana and Lawrence Davis. “Training Feedforward Neural Networks Using Genetic Algorithms.” In: *IJCAI*. Vol. 89. 1989, pp. 762–767.

[31] Sarah G Moxon et al. “Count every newborn; a measurement improvement roadmap for coverage data”. In: *BMC pregnancy and childbirth* 15.2 (2015), S8.

[32] G. Msemo et al. “Newborn Mortality and Fresh Stillbirth Rates in Tanzania After Helping Babies Breathe Training”. In: *PEDIATRICS* 131.2 (Feb. 1, 2013), e353–e360. ISSN: 0031-4005, 1098-4275. DOI: 10.1542/peds.2012-1795. URL: http://

pediatrics.aappublications.org/cgi/doi/10.1542/peds.2012-1795 (visited on 05/04/2020).

[33] World Health Organization. *Newborns: reducing mortality*. 2019. URL: https://www.who.int/news-room/fact-sheets/detail/newborns-reducing-mortality (visited on 04/27/2020).

[34] Maria MP Petrou and Costas Petrou. *Image processing: the fundamentals*. John Wiley & Sons, 2010.

[35] Daniel Scharstein. *Middlebury flow*. 2007. URL: http://vision.middlebury.edu/flow/code/flow-code/README.txt (visited on 06/12/2020).

[36] Dominik Scherer, Andreas Müller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition". In: *International conference on artificial neural networks*. Springer. 2010, pp. 92–101.

[37] Hoo-Chang Shin et al. "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning". In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1285–1298.

[38] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[39] Lisa Torrey and Jude Shavlik. "Transfer learning". In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI Global, 2010, pp. 242–264.

[40] *WHO — Neonatal mortality*. WHO. Library Catalog: www.who.int Publisher: World Health Organization. URL: http://www.who.int/gho/child_health/mortality/neonatal_text/en/ (visited on 05/04/2020).

[41] Dr Alan Woodruff. *What is a neuron?* Library Catalog: qbi.uq.edu.au. Nov. 22, 2016. URL: https://qbi.uq.edu.au/brain/brain-anatomy/what-neuron (visited on 04/30/2020).

[42] World Health Organization. *Guidelines on basic newborn resuscitation*. OCLC: 851144994. 2012. ISBN: 978-92-4-150369-3. URL: http://www.ncbi.nlm.nih.gov/books/NBK137872/ (visited on 05/04/2020).

[43] Bing Xu et al. “Empirical evaluation of rectified activations in convolutional network”. In: *arXiv preprint arXiv:1505.00853* (2015).

[44] H. Zhao et al. “Loss Functions for Image Restoration With Neural Networks”. In: *IEEE Transactions on Computational Imaging* 3.1 (2017), pp. 47–57.