Lund University

Faculty of Engineering

Centre for Mathematical Sciences

An Implementation Of A Rate Controller Using A Neural Network

Author
Martin Sundeqvist
martin.sundequist@gmail.com

Supervisor
Kalle Åström
karl.astrom@math.lth.se

Assistant Supervisor
Martin Ahrnbom
martin.ahrnbom@math.lth.se

Assistant Supervisor
Fredrik Pihl
Fredrik.Pihl@axis.com

Assistant Supervisor
Alexander Toresson
Alexander.Toresson@axis.com

Assistant Supervisor
Hampus Åström
Hampus.Astrom@axis.com

November 10, 2018

Abstract

The rate controller is a crucial component of a video encoder. The component regulates bit rate in a video, determining how much information is to be updated in a video frame by assigning an overall compression rate to said frame. At Axis Communications, a company specializing in network cameras, the component is implemented using traditional automatic control methods operating on a select few data parameters. This thesis project aims to implement the rate controller with a machine learning approach, using a neural network.

A key challenge is the task of devising a set of customized quality measurements suited for the purposes of surveillance video. Mainly, these measurements are based around object detection. The quality measurement deemed most promising for future use is based on the presence of false positives and negatives in object detection data. The measurements are used to generate a set of labels, each a theoretically optimal compression rate. The input features used by the network is video meta data, a set of parameters describing video content that is more low-dimensional than pixel data from the video.

An attempt at encoding videos with a rate controller implemented using a set of trained neural networks is finally carried out. The results indicate that the networks can be trained to adapt compression rate based on changes in activity within a video. However, the compression rates given by the network can change drastically from frame-to-frame. Also, the ability of the networks to adapt to smaller changes can not be reliably determined, requiring further testing.

Acknowledgements

I would like to thank my supervisors at Lund University and Axis Communications: Kalle Åström, Martin Ahrnbom, Hampus Åström, Alexander Toresson and Fredrik Pihl. I have been lucky to be surrounded with people that are so skilled in their respective fields and that so readily have taken time to provide guidance or help out in various practical ways whenever needed. This thesis project would not have been possible without everyone of you. Further, I would like to thank Axis Communications for providing the opportunity and the resources to carry out this thesis project.

Contents

1 Introduction

The end goal of designing video encoders for surveillance cameras is to keep what is important from going missing. A flaw in the video encoding software, a decision of removing the wrong information at the wrong time in the video can be the difference between seeing a perpetrator's face and seeing a blur of random pixels.

Axis Communications is a company specialized in the development of network cameras, and by extension the on-camera video encoders as well. Given the possible consequences of losing important information, engineers at Axis might be tempted to always encode video with the highest quality, updating every new video frame with as much information as possible. But sending and storing information requires infrastructure in the form of server halls and stable networks, both of which cost money for the customers.

This is the problem faced by video engineers at Axis: They must uphold a quality level referred to as **forensic detail**, where key elements such as faces or registration plates can be identified during important events. At the same time, they should use as few bits as possible to encode the video when there is no need to send information.

A key task in the video encoder is carried out by the **rate controller** component, which chooses an overall compression rate value for each video frame. This compression rate is later used in the encoding process to determine the specifics of how compression should be performed.

Currently, this task is implemented with automatic control techniques, where the rate controller chooses compression rates to regulate the bit rate in the video according to pre-determined heuristics. The data used to guide the decision is a set of parameters available on the camera at run-time, referred to as video meta data. These parameters contain information regarding the encoding of previous frames as well as data from the image processing utilities that runs on the camera. Recently, it has been suggested that a machine learning approach could be used to improve the rate controller. A few reasons for this include:

- The possibility of adapting compression rate based on custom quality measurements adapted for a surveillance video context rather than the video bit rate.
- The prospect of using the existing data parameters more efficiently.
- Using a classic control approach also entails that feature selection of parameters be done manually, so introducing new parameters for control is cumbersome. A machine learning approach could decrease the need for manual feature selection and allow the use of more data parameters when designing a rate controller model.

Exploring this possibility is the subject of this thesis project. The aim of this thesis is to develop an implementation of a rate controller using an artificial neural network, a machine learning algorithm that uses connected computational nodes to approximate complex functions.

Literature in the area of video compression using machine learning is limited and not adapted to the needs of the surveillance industry. Most current papers are directed towards general-purpose applications of encoding, solutions where images or video frames are compressed only with the general quality of the image or video in mind. Axis's customers have limited interest in aesthetically pleasing video with generally high bit rates. This means that the intended rate controller model needs custom quality measurements to guide its choice of compression rate, so that the resulting compression is more in line with the needs of the customers. Developing and evaluating these quality measurements is a key aspect of this thesis.

Networks used in image processing are often large and operate directly on image data, which is very high-dimensional. Due to the limited hardware capabilities of a surveillance camera, which is the intended platform of the rate controller, running large and complex networks on the camera is likely to cause issues. One way to keep down the scale of the implemented rate controller, is to operate on more low-dimensional input data, such as the video meta data. This serves as another interesting aspect of the thesis project, to find out how well the low-dimensional video meta data can be mapped to the compression rates generated with the custom quality measurement.

The experience gained during this thesis project helps expand the current body of knowledge regarding the use of machine learning in video compression, with the specific use case of the surveillance industry kept in mind.

2 Current literature

Researching the current literature does not yield any studies that are directly related to the concept of implementing a rate controller using a machine learning solution. Instead, most studies in the research area focus on experimental ways of performing the entire encoding process. Little relevant information is provided regarding the implementation of rate controllers, with most studies having a much wider scope. A few examples of existing literature is listed here to give an idea of the state of the research area, although the literature was not very pertinent to the project.

Researchers at Nanjing University have developed a video encoder implementation that uses machine learning (Chen et al. (2017)). Their solution is based on a convolutional neural network architecture trained on Twitters Image Dataset. **Structural similarity index** (a general image quality measurement which will also be studied in this paper) and bit rate are used as quality measurements in their study. The researchers evaluate the performance of the so called "DeepCoder" network and compare it to the performance of a famous video encoder called $x264$, concluding that their resulting performance is comparably efficient.

While the above solution is an example of supervised learning, another paper (Singh et al. (2009)) uses an unsupervised learning solution (see section 5.2.1 for information on supervised and unsupervised learning). Their solution utilizes self-organizing maps, a type of neural network architecture, that projects a high dimensional data space into a lower dimension data space. The researchers use this mapping to represent each video frame with a reduced set of features, saving the mapping pattern using a Hopfield network (a special type of neural networks used to memorize patterns). The paper does not include any results or data regarding the quality of video frames decoded from the pattern stored by the Hopfield network.

Google also provide a set of interesting papers (Toderici et al. (2015)) (Toderici et al. (2016)). While they focus on the area of image compression, the fundamental concepts are shared with the realm of video compression. The Google researchers use a set of recurrent networks as well as convolutional networks to achieve results that outperform the JPEG standard.

3 Requirements and limitations

The fundamental uniqueness of this work relies on the specific needs and challenges that surveillance video entails:

- Surveillance networks always deal with real-time compression, which means that there is no time to "wait" for processing to finish. When encountering a frame, a decision must be made instantaneously as to which compression rate to choose, or whether to drop the frame completely (dropping a frame means sending another copy of the last encoded frame).
- The necessity of running software on embedded systems. Running performance-demanding deep networks, requiring split-second handling of raw image data, as in the case of (Chen et al. (2017)) is likely not an option due to the limited storage and processing power of security cameras.
- Quality only needs to be held at an acceptable level and only for select areas of the video frame. As mentioned earlier, customers require **forensic detail**: When something important does occur, the video bit rate must be sufficiently high to allow recognition of important details (such as faces or license plates). The rest of the time, customers usually prefer the highest possible compression, as sending frames of large sizes requires costly investments in infrastructure.

Developing a full-fledged rate controller model using a machine learning approach is a far-reaching project. To limit the scope of this thesis, the intention is only to be an exploratory work. Emphasis is put on trying out and evaluating different approaches. The following limitations are applied to this project:

- The finalized implementation is not intended to run on-camera. This would in all likelihood be a time-consuming process and adapting an implemented network to the constraints imposed by an embedded environment would likely require highly specialized skills.
- The training data for the neural network will be generated from videos recorded with an Axis camera. Using a large array of different scenes entails more complexity in data. This project limits the data complexity by only using recorded sequences from one type of scene (a road outside Axis offices).

4 Problem formulation

The goal of this thesis can be concluded in the following points:

1. Find or develop custom quality measurements that can be used to discern the degree to which frames should be compressed.
2. Generate appropriate data sets for training, with labeled input examples that can be used to train and evaluate a neural network. The labels should be compression rate values generated based on some custom quality measurement.
3. Develop an implementation of a rate controller using a neural network that takes video meta data as input and returns compression rate values for video frames.
4. Evaluate the network for the implemented rate controller with regard to some performance measure, i.e. determine the degree to which the network learned to predict the label compression rates that were generated with a quality measurement.
5. Investigate to what degree the implemented rate controller is useful in a real-life setting. In other words try to evaluate the result when encoding a video with the implemented rate controller.
6. Investigate the feasibility of transferring the implementation to an embedded system.

5 Theory

5.1 Video Encoding

Video encoding is the process of taking a raw byte data stream representing a video and finding another, smaller representation for the video. This representation can either be **lossless** or **lossy**. In the former case the original video can be reconstructed completely using the representation, the latter sacrifices this reproducibility trying to achieve a representation that is hopefully much smaller in size. This thesis only deals with video encoding in the context of lossy compression.

5.1.1 H264

Software and hardware that encodes video are referred to as video encoders. How encoding is performed, the intended use case of the video and the system it is to be performed on is highly individual. But while Netflix and Axis have different intended purposes for video encoding, there are also many commonalities. The same broad concepts are used when encoding video with lossy compression. For this reason, common video encoding standards are used that provide a set of rules for the implementation of video encoders. A highly prominent standard in industry is the H264 standard. Inside the H264 standard there exists a set of **profiles**, each corresponding to some set of features in the standard.

All Axis cameras have a video encoder, many of which use the H264 standard. For this reason it is also the standard that will be used within the context of this thesis work. The H264 standard is complex, and as mentioned previously each profile holds a different set of features. A more detailed look into the subject is given for example by (Poynton (2003)), but some main concepts are:

- GOP (Group of Pictures)
- I-frames
- P-frames
- Macroblocks
- QP values (Quantization Parameter)

The last concept, the QP value, is to some extent interchangeable with the previously mentioned compression rate. It is the entity that the final implemented rate controller in this project is supposed to produce.

In H264, a compressed video is represented by a set amount of GOP:s, each GOP is in itself a set of frames. Usually, the number of frames in a GOP is approximately the same as the video fps. There are a number of different frame types that are defined by the H264 standard. The I-frame is a frame that

contains sufficient information to be rendered without any auxiliary information. As I-frames convey a lot of information they are usually the greatest size of all the frame types.

To avoid sending unnecessary information, a GOP only has one I-frame at the start of the GOP, followed by a sequence of other frame types of smaller sizes, called P (predicted) and B (bilinearly predicted) frames. For the sake of simplicity, only P-frames are considered here and during the thesis work.

The encoder uses a block-oriented motion-compensating algorithm to create a prediction of what the current frame should look like. What this means is that the encoder tries to match macroblocks ($16x16$ pixels in quadratic blocks that make up the video frame) in the current frame with a previous reference frame. This information is then used to generate a prediction of the current frame.

After the prediction phase, the encoder considers the predicted frame in relation to some previous frame called a reference frame. It outputs the difference between the predicted frame and this reference frame in a P-frame. Usually, the majority of the predicted frame will be approximately the same as the reference frame. As only the amount of information required to communicate the difference between the target frame and the reference frame is saved in the P-frame, this means that it is usually a lot smaller than an I-frame.

The encoder now moves on to the **transformation** phase, where it transforms the residual data into its frequency components. In H264, this transformation is done using a modified version of the Discrete Cosine Transform (Mandyam et al. (1997)). The transformation returns a set of coefficients that can be used by the decoder to reconstruct the frame with sinusoid bases for each macroblock.

After the transformation phase, the encoder outputs all the information for the video in a coded bit sequence.

5.1.2 Rate controller

This brings up the topic of the rate controller and QP values. The job of the rate controller is to decide an overall QP value for each frame. The QP value is defined by the H264 standard and ranges from 0 to 51. Compression with QP 0 is lossless, while a QP of 51 is the highest possible lossy compression. After the rate controller chooses such a value, the QP value is embedded in the coded bit sequence, to be used in the quantization process.

During quantization, the precision of the coefficients extracted with the modified DCT is reduced according to the QP value. The higher the QP value, the more of the transform coefficients are set to zero, meaning fewer frequency components are used to represent the frame block.

Currently, the rate controller regulates video bit rate with the QP value. Either the rate controller is instructed to choose QP values so that bit rate is kept constant, this is referred to as constant bit rate (**CBR**). Another option is to use variable bit rate (**VBR**), where QP is held constant, which can cause large changes in bit rate throughout the video. Maximum bit rate (**MBR**) establishes a top limit for the bit rate so that it does not go too high.

But in this project, the idea is to not regulate the bit rate, but rather choosing the QP value based on a custom quality measurement.

To be kept in mind when implementing the new rate controller is the importance of the first few frames of a GOP. When predicting a P-frame the encoder uses information from the last I-frame as well as all the P-frames that followed it. If a decision is made to get rid of a lot of frame data early on in a GOP and the need arises later on in the same GOP to improve quality, a large amount of macroblocks must be updated. For this reason, the selected QP values of the first few P-frames in a GOP are important.

Another important aspect of the current rate controller implementation is the limitation on frame-to-frame changes. The QP is only allowed to change to a certain extent between frames, as too large changes cause issues with video quality.

5.1.3 Colour Spaces and Y'CbCr

A few words regarding colour spaces are in order. Axis adheres to a set of different standards when developing the format of the recorded video and their representation of colour is a central issue. During various stages of the project it is a necessity to convert between different spaces to allow video data to be used for different applications.

Colour spaces are different types of notations for describing colours mathematically. Which colour space is used depends on the intended application. At Axis and in many other places working with video systems, a **Y'CbCr** colour space is used to represent a video signal.

The **RGB** colour space in which signals are represented in terms of their red, green and blue components can be considered the "basic" colour space (Tkalcic & Tasic (2003)). But when designing a video system, one needs to keep in mind that the human visual system is more attuned to detect differences in the intensity of light rather than differences in color. As long as there is full detail in terms of intensity, one can get rid of a lot of color information (by means of averaging or filtering out color signals) while retaining good video quality (Poynton (2003)).

The above is referred to as **luma and chroma subsampling** (luma referring to intensity and chroma to colour components) and it is where the **Y'CbCr** colour space becomes relevant. Rather than representing colour simply in terms of red, green and blue it represents colours with luma (light intensity) and chrominance (color information) components:

- **Y'**; The luma component
- **Cb**: A chrominance component, the blue colour component of the signal as it relates to the luma component.
- **Cr**: A chrominance component, the red component of the signal as it relates to the luma component.

When subsampling, a sampling scheme is used to determine the relation of luma to chroma components. At Axis the standard scheme is $4:2:0$. In such a scheme, all pixels get one luma component, but the entire frame samples only a fourth as many chroma components. This means that in a group of 4 pixels all pixels get their own Y' component but must share Cb/Cr components.

Further, Axis follows a set of different standards for encoding analog to digital video provided by the **International Telecommunications Union**. These are referred to as **bt.601**, **bt. 709** and **bt. 2020**, each of which impose certain rules with regard to how colour is represented in the digital video signal. One of the main reasons this is important is the concept of colour range. Axis uses full range with 8 bit depth, this means that all components are represented with a $[0, 255]$ value range (ITU Radiocommunication Sector (2011)). However, many applications use the same standard but with a clamped range where each component has a reduced range:

- The luma component is in the range $[16, 235]$.
- Both chrominance components are in the range $[16, 240]$.

During the project it is key to preserve the full range as the original recorded video is re-encoded for data generation purposes. During certain stages of the project, it is also required that one converts to the RGB colour space to allow interfacing with the YOLO network. Maintaining colour range for these different applications is key to ensuring consistency for the thesis results.

5.1.4 Frame compression

Throughout this paper, the phrases **uncompressed frame** and **compressed frame** are commonly used. The uncompressed frame being referred to has in fact been compressed, but at such a low level of compression (QP 14 as overall compression rate) that it is indistinguishable from the raw video frame. When talking about compressed frames in the paper, what is referred to is the uncompressed frame that has been compressed again at some other QP value.

5.2 Machine Learning

Machine learning is essentially a mixed discipline with one foot in computer science and the other in applied statistics. It can be described as a way of using applied statistics, along with the power of computers to statistically estimate functions (Goodfellow et al. (2016)). A clear definition of what is meant by a learning algorithm is given by (Mitchell (1997)):

> A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in T, as measured by P, improves with experience E.

5.2.1 Experience

Machine learning problems can be separated into the two main groups of **supervised** and **unsupervised** learning. These are relevant to describe the type of experience the rate controller is expected to operate on when learning the task of predicting QP values.

Supervised learning problems are considered to be more traditional problems. With these types of problems, the algorithm is presented with input examples, each example some set of features. Each such input example has an attached **label**, that is to be considered the correct output the network should produce. The learning process is then reduced to having the algorithm systematically make educated guesses of what the corresponding label is for a given input and automatically adapting its decision-making based on the results. If designed correctly, the algorithm should produce better guesses as it gets more experience.

Unsupervised learning has no label for the input examples. Instead, the concept of learning here is to try to find structure, or some representation of the input examples that is useful. An example of this are the self-organized maps used to compress video in (Singh et al. (2009)).

In this thesis work, supervised learning is used. The input examples is a set of features that are assumed to be available for a future on-camera implementation. Features include **encoding data** for a set of previous frames and data from the **camera imaging pipeline**, software that handles image processing on the camera. As previously mentioned, all such input examples must be paired with a label. This is key to the thesis project: Finding custom quality measurements adapted to surveillance video and producing labels for the input examples based on those measurements.

5.2.2 Task and Performance Measure

The task T (section 5.2) in this thesis project is a regression task. In such tasks, the learning algorithm needs to output a function f: $\mathbb{R}^N \rightarrow \mathbb{R}$ (Goodfellow et al. (2016)). In other words, the neural networks designed in this project needs to learn a function that maps inputs from an N-dimensional space (the input data of a frame) to a numerical real value (a QP value).

In this project, a custom performance measure is used, named **residual interval ratio**.

The residuals for each predicted QP value is the difference between the predicted value and the QP value it tries to predict. Using residual interval ratio as performance measure, these residuals are studied to determine which ratio of predicted values fall within specific residual intervals ($\pm 1, 3, 5, 7, 10$). In other words, if a large ratio of values fall within ± 1 or ± 3 this suggests that the network manages to predict the theoretical QP values quite well. A small ratio of predicted value residuals within these intervals calls the network performance into question. Objectively determining an acceptable performance level with regard to this measure is difficult. As a guideline used to evaluate performance for this thesis project, **acceptable performance** is defined as the case where the majority of predicted values have residuals within the ± 5 interval.

5.2.3 Formatting data and generalization performance

Producing the data sets used for the supervised learning requires forethought. To properly train and evaluate a machine learning algorithm, in this case a neural network, the available data is split into three parts:

- A **training set**, this set is the only data that the network uses to directly modify its behaviour.
- A **validation set**, used to set and tune **hyper parameters**. The objective being to improve **generalization performance**, the network's ability to perform well for yet unseen data.
- A **test set**, which is completely isolated from the other sets during the training phase. Used to evaluate generalization performance after training.

The basic idea here is that the **training set** is what the network uses to improve itself, it is the process of trying to predict the correct output for input examples in the training set (and adapting behaviour based on the error obtained after prediction) that trains the network. Training takes place over a number of **epochs**: A set of training rounds in each of which the network predicts, evaluates performance and adapts its behaviour accordingly.

The validation set is not used by the network to directly change behaviour, rather it is used to improve the network performance on data outside the training set. This is done by tuning hyper parameters: Parameters set prior to the start of the training that have an impact on the results.

The objective of tuning the hyper parameters is the prevention of **overfitting**. Overfitting is something that occurs when the learning algorithm (in this case the neural network), adapts its behaviour too much with regard to specific attributes of the training set data. Over time, these adaptive changes can lead to the network having great performance for the training set, but poor performance for unseen data. The network is then said to have poor generalization performance.

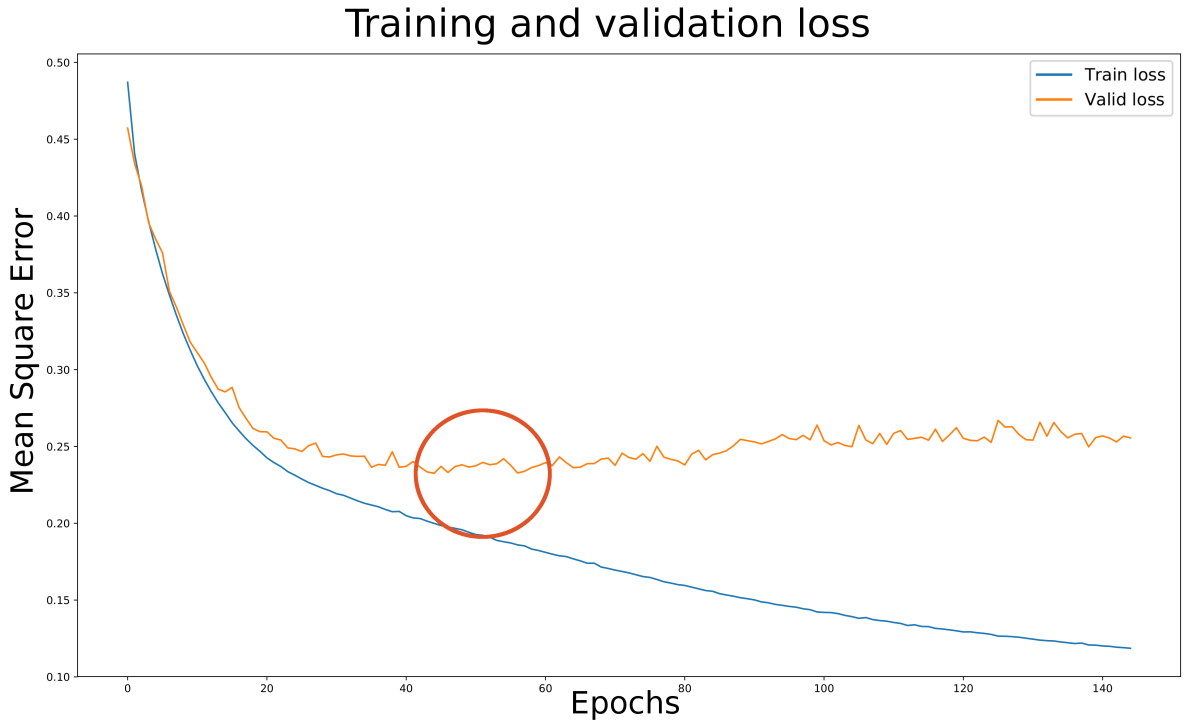


Figure 1: This figure illustrates overfitting. Note how for the epochs to the right of the red circle, the validation loss starts to diverge from the training loss. This indicates that the network has started adapting its behaviour based on specific attributes of the training data not present (or less common) in the validation set. As a result, training loss keeps improving, while the network validation loss starts to gradually worsen.

To identify when overfitting occurs, a common strategy is to observe **training loss** and **validation loss** concurrently. Here, **loss** refers to the output of a loss function (see section 5.3.1) with regard to the network prediction results in each epoch. If loss decreases over time it indicates that the model is improving itself with regard to prediction results as it gains more experience. Looking at figure 1, the overfitting starts to occur where the loss curves start diverging (indicated by the red circle). Various strategies can be adapted to prevent this, they are referred to with the umbrella term **regularization** (see sections 5.3.1, 5.4.2 and 5.4.3).

The goal of training is to achieve a high training set performance, but also good generalization performance. The test set is a final test to evaluate the generalization performance after training is done.

Some research has been made regarding what the difference in size for the sets should be (Guyon (1996)). However, for most practical applications it is usually sufficient to ensure that the training set is substantially larger ($70 - 80$ % of the total available data) than the other sets. A commonly used setup is a $70 : 20 : 10\%$ relationship for training, validation and test data.

5.3 Artificial Neural Networks

The structure and purpose of artificial neural networks can be widely different, but the general idea is to use computing units, or **neurons** to approximate some function f.

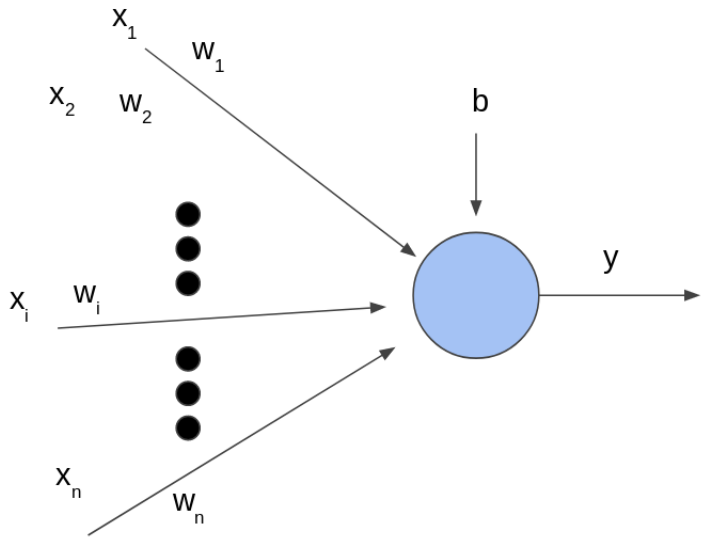


Figure 2: An example of a neuron in a neural network. The $\{x_i|i = 1..N\}$ correspond to inputs from other neurons, the $\{w_i|i = 1..N\}$ being scalars associated with a connection between the neuron and a corresponding input neuron. Output y is produced by adding together a weighted sum (equation 1) of the inputs and weights with a bias b and passing the result to an activation function 2.

$$s = \sum_{i=1}^{N} w_i x_i + b \tag{1}$$

$$y = \phi(s) = \phi(\sum_{i=1}^{N} w_i x_i + b) \tag{2}$$

$$\phi(s) = s^+ = max(0, s) \tag{3}$$

A neuron is a computing unit that takes a set of inputs $\{x_i|i = 1..N\}$, each with an associated **weight** $\{w_i|i = ..N\}$ (see figure 2), and computes a weighted sum of the inputs along with some **bias** b (see equation 1). This sum is then passed to an **activation function** (ϕ in equation 2), producing the network output and mapping it to a specific interval. In this project the rectified activation function is used for all neurons (equation 3).

The aforementioned weights are essentially what determines the behaviour of the network. Updating weights changes the weighted sums that are passed to the activation functions for each neuron, allowing the network to map complex patterns in input data to some given output. Changing these weights in a systematic way is the process of training, which is described in section 5.3.1.

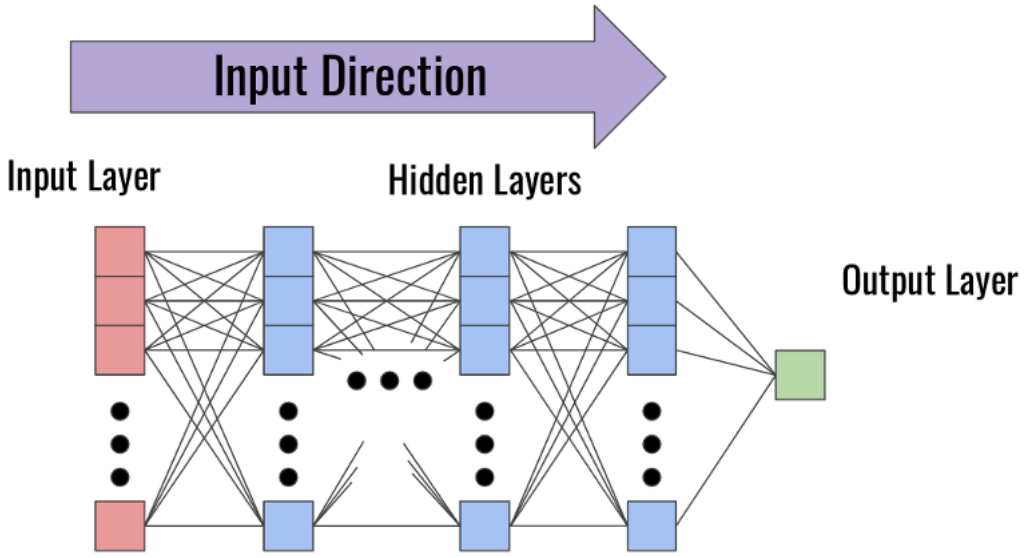


Figure 3: An overview of the network architecture for a multilayer perceptron. The input feature vector is applied to the input layer, which computes the input for the first hidden layer. Input moves in one direction, with each hidden layer's output acting as input for the next hidden layer. Network output is aggregated in the output layer. In the figure, the output layer is a single neuron, yielding one dimensional output (which is what will be used in this project). However, the output layer can have any shape.

This project will use the simplest type of network architecture, referred to as a **feed-forward network**, with stacked layers of neurons where outputs go from one layer to the next in one direction (see figure 3). Also referred to as a **multilayered perceptron** (MLP), these types of networks have:

- One input layer, the first layer, which processes the input feature vector.
- One or more hidden layers. An MLP can have any given number of hidden layers, each with some arbitrary set of neurons.
- One output layer, the final layer of the network. While the output can have any shape, in the case of this project it will be a real number, the predicted QP value.

The allure of neural networks is their ability to approximate a large number of different linear and non-linear functions. The fundamental assertion of the universal approximation theorem (Cybenko (1989)) is that even the simplest multilayer perceptron (with only one hidden layer having a finite set of neurons) can approximate a large set of continuous functions assuming "mild" assumptions on the activation function on compact subsets of $\mathbb{R}^N$.

In other words, by only having a single hidden layer in the feedforward network with a finite set of neurons, it is theoretically possible to approximate most conceivable functions. That is not to say that the approximation will be useful in terms of generalization performance. Also, as stated in the original paper, complicated functions might require a massive amount of neurons in the hidden layer.

5.3.1 Central concepts and the process of learning

While section 5.3 describes neural networks in general, it gives no idea of how the process of learning actually works. As mentioned in section 5.2.3, the training takes place during a set of the aforementioned epochs, or training rounds. For each such epoch, the network predicts the QP values for some subset of frame inputs in the training set. These predicted values are passed to a **loss function** along with their correct corresponding values.

An array of different loss functions exist, what they have in common is that they map the network performance for the training and validation set during each training round to a real number. In this thesis work, mean square error will be used as the loss function. Given a set of predicted values $\{\hat{y}_i | i = 1..N\}$ with associated correct values $\{y_i | i = 1..N\}$ the loss-function mean squared error is defined as:

$$MSE(\hat{y}, y) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4}$$

The loss function must be differentiable with regard to the weights of the network. The reason for this is that the way the network determines how the weights should be updated for a given neuron is by calculating the gradient with respect to the weights of the neuron.

After all neurons are associated with some error, an algorithm called **backpropagation** is used to calculate the gradient for each neuron in the network. These gradients are passed to an **optimization algorithm** that updates the weight given some heuristic. Commonly used, algorithms based on **gradient descent** choose to update weights so that one moves in the direction of the negative gradient of the loss function. The idea here is that by continuously moving in the direction of the negative gradient over a large set of epochs, one should end up in a minimum of the loss function.

The degree to which changes should be made to weights in each epoch is determined in part by the **learning rate**. The learning rate parameter determines how long of a step the optimizing algorithm should take in the direction of the negative gradient direction, hence determining how much the weights should be updated. In short, the following tends to hold true:

- Larger learning rates means a shorter training time. Further, they usually prevent the network from getting stuck inside of a local minimum of the loss function rather than finding the global minimum. However, if the training phase does not last long enough, there exists a risk that the global minimum can be overshot, so that the training results do not converge.
- Smaller learning rates means a longer training time. Given that a small learning rate is used, the risk of overshooting a minima is reduced. However, small updates to weights also means a risk of getting stuck in local minima.

Regularization is another important concept in machine learning terminology. It is a set of possible measures one can take to reduce overfitting (section 5.2.3). One type of regularization can be done by introducing an extra term in the loss function that is some constant λ, multiplied by some norm of the weights. Different types of norms can be used, but the general idea is that as long as weights are close to zero if will have little effect on the norm, while greater values will have a very strong effect. The loss function will thus produce much higher values when the network tries to select more complex models (where certain weights have very different values as compared to others). This improves generalization as it inhibits specialization to the training set. In this thesis project, the L2 norm is used, meaning a term that is the sum of squared weights multiplied by some constant is added to the loss function. As mentioned earlier, the mean square error will be used as loss function in the network. With L2 regularization, the regularized loss function $loss_{regl}$ has the following appearance:

$$loss_{regl} = MSE + \lambda \cdot \sum_{i=1}^{N} w_i^2 \tag{5}$$

The λ parameter is one of the hyper parameters (section 5.2.3) that is varied during training to achieve different levels of regularization.

5.4 Training Strategies

5.4.1 Formatting data

It is often recommended to standardize or normalize data that is to be used as input in neural networks. In this project input and output values for a given model were standardized in the following way. Given an array of N values $\mathbf{x} = x_1, x_2, ..., x_N$, the standardization for x_i is given by:

$$x_{i,standardized} = \frac{x_i - min(\mathbf{x})}{max(\mathbf{x}) - min(\mathbf{x})} \tag{6}$$

For each feature in the input, the maximum and minimum values correspond to the maximum and minimum of the training set used. For the output QP value, the standardization is made using 14 as minimum and 51 as maximum (51 is the highest possible QP value, 14 is the lowest value used when generating optimal QP labels, see section 5.5.2).

5.4.2 Data Augmentation

Adjusting the network architecture, tuning hyper parameters and applying various regularization strategies can have a large effect on the training results and generalization performance. However, the access to a sufficiently large and varied data set is often the critical factor for achieving a good model. This echoes the words of (Domingos (2012)):

> As a rule of thumb, a dumb algorithm with lots and lots of data beats a clever one with modest amounts of it.

But access to new input data is often limited or resource-heavy to generate. Perhaps there is even a large amount of data, but the inherent variance is insufficient to provide the algorithm with sufficiently different examples. In this case, data augmentation can be an efficient way of artificially extending the amount of available data to improve network performance.

The general idea of data augmentation is to reuse the existing data, but manipulate it in different ways so that it becomes markedly different from the existing data. In many machine learning applications in the video- or imaging field, the input data are the images in themselves. In such cases, a classic way of augmenting the data is by rotating, cropping or flipping the images (Perez & Wang (2017)).

In this thesis paper, the input data has a very different format (video meta data for various time steps organized in a one-dimensional array). One way of augmenting such data is by simply adding gaussian noise to each feature. Conceivably, it is desired that the network should select the same output for an input example if only small random noise has been appended to the example. A possible issue with this strategy however, is the fact that in the case of image data augmentation the image output should usually be the same after transformations are applied (for example a rotated image of a dog is still an image of a dog). However, adding gaussian noise to input data is by nature a random transformation, and it is not always certain that one would like to map the noisy inputs to the corresponding output.

In practical terms, the way that data augmentation is performed in the project is by adding random samples from a gaussian function to each input feature:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{7}$$

where the expected value μ is set to $\mu = 0$ and the standard deviation σ is set to $\sigma = 0.4$. Samples are generated with a fixed random seed.

5.4.3 Dropout Layer

Dropout layers are a common regularization strategy, capable of improving network performance in many different machine learning applications (Srivastava et al. (2014)). Dropout layers are based around the concept of randomly inactivating ("dropping") network nodes during training. For each epoch, some ratio of randomly selected network nodes in one or more layers is dropped. This prevents the associated weights from affecting the output value or prediction loss, as well as updating themselves.

Similar to the above mentioned use of a regularization term, one or more dropout layers inhibits the network in terms of making large updates to specific layer weights early on in the training session. The intention is to end up with a more robust network that has better generalization performance (section 5.2.3). **Dropout rate** refers to the ratio of nodes in a layer that are dropped during one epoch.

5.5 Video Quality Measurements

This thesis project uses supervised learning as the principle for teaching the rate controller how to select QP values. This means that the neural network needs access to labeled input examples to learn (section 5.3.1). As discussed in section 5.2.1, each input is a set of video meta data parameters for the current frame as well as a subset of the previous frames.

5.5.1 Quality measurements

In this thesis paper, **quality measurements** will be mentioned throughout the text. A quality measurement is defined as a metric according to which a (theoretical) quality can be attributed to a compressed frame.

These quality measurements are intended to inform the decision of the rate controller so that it does not choose a compression QP value that results in poor quality with regard to the measurement.

5.5.2 Optimal QP

Having defined some quality measurement, what remains to do is to train the network to uphold the related quality when choosing QP values. To enable the network to do this, QP value labels must be chosen for each input example that encourage this behaviour while being mindful of video bit rate. For this reason, each quality measurement will have an attached **quality cost function**. This function is designed to create optimal QP values to be used as labels for the input examples corresponding to a set of frames. The goal being that a frame compressed with its optimal QP value should strike a balance between upholding quality as defined by the quality measurement and minimizing video frame size.

Thus, the term **optimal QP** when used in this text does not refer to an objectively optimal choice of compression rate. Rather it is interchangeable with the input QP label, a QP value that is (theoretically) the optimal compression rate with regard to a quality measurement and its associated quality cost function. The process of finding a set of optimal QP values can be summed up in the following steps:

- Decide on a quality measurement.
- Design a quality cost function that can be used to select QP values based on the definition of the quality measurement.
- Extract data for a set of frames that is required for the quality cost function and generate the optimal QP values to be used as labels for the corresponding input examples.

A few important criteria must also apply to the optimal QP values:

- The network must be able to train for the optimal QP values, i.e. network performance (section 5.2.2) must improve with experience.
- The optimal QP values must be chosen in such a way that the network can generalize the results (section 5.2.3).
- The prediction strategy learned by the rate controller must correspond to something reasonable in the real world. If after training, the rate controller compresses a test video with erratic, random compression this demonstrates poor real-life performance.

5.5.3 Quality Cost Functions

As stated in section 5.5.2, each quality measurement needs an associated quality cost function. For a given compressed frame, a quality cost function needs access to the following:

- The **frame size** of the compressed frame.
- Data that can be used to determine **quality measurement**.

The idea of a quality cost function is to generate costs for each compressed version of a frame and then choosing the QP with the lowest cost as the optimal QP. They way these costs are generated is by weighing the frame size of a compressed frame against a **penalty** that has been attributed with regard to the used quality measurement.

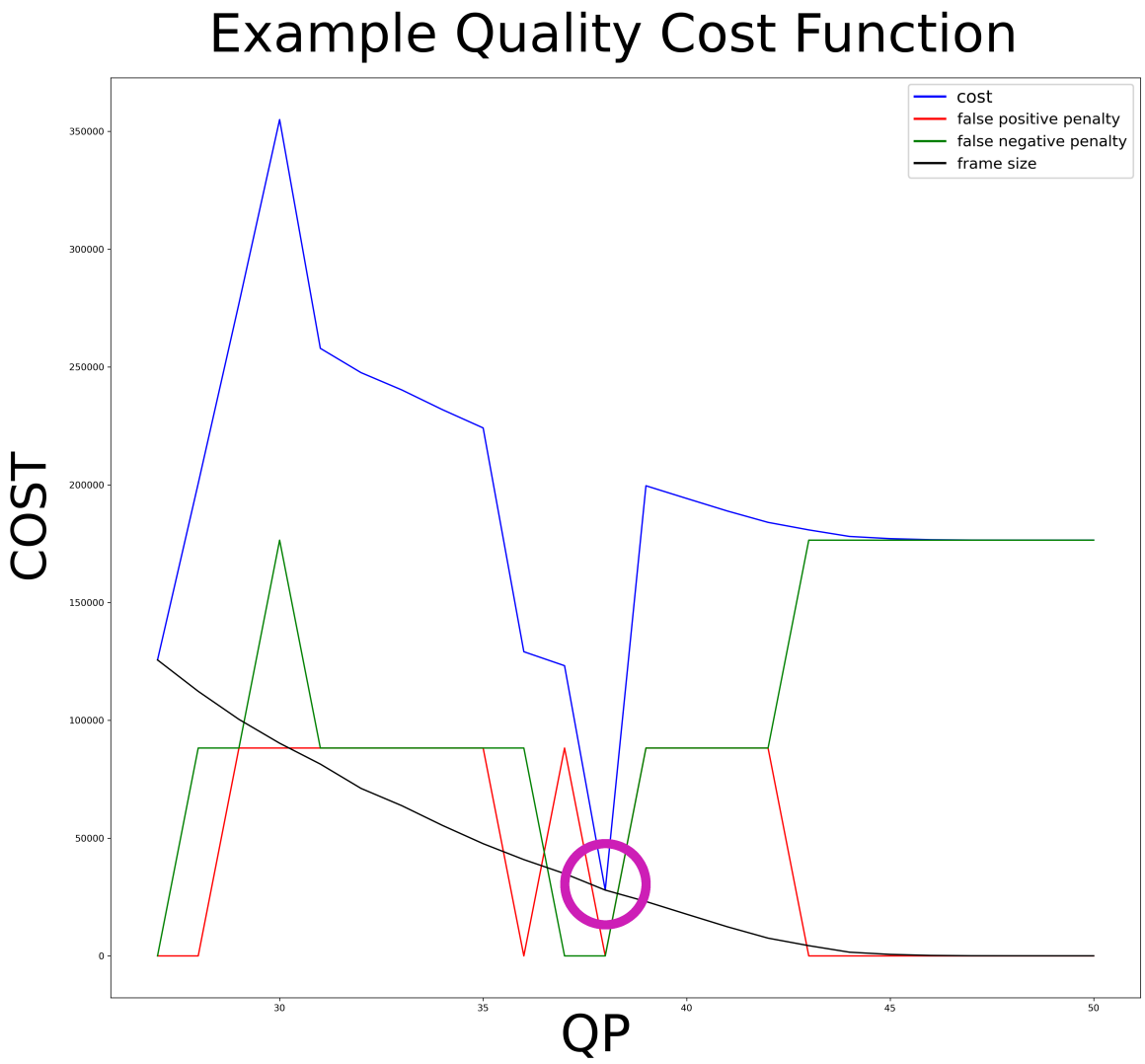


Figure 4: Example of costs produced by the quality cost function defined in equation 13 along with constituent components of the cost for all compressed versions of a sample frame. In this case, the quality measurement penalty is based on the presence of false positives or negatives (with regard to object detection) in a compressed frame (see section 5.7.4). Note that at each point, the cost curve is the sum of the frame size and the penalties attributed due to the number of false positives and negatives individually. The purple circle indicates the lowest cost, the corresponding QP is chosen as the label for the input example corresponding to the video frame.

Figure 4 shows an example of a quality cost function generated for all compressed versions of a sample frame using the quality cost function in equation 13. As is evident from the plot, the frame size is monotonically decreasing, while the penalties vary with QP. Over time, more and more of the cost consists of penalties and the frame size influence becomes unnoticeable.

5.6 SSIM

Traditionally, mean square error is a popular signal fidelity measure in the realm of signal processing, image processing being no exception. Assuming that $\mathbf{x}$ and $\mathbf{y}$ are two finite-length, discrete signals, both with N samples and indexed so that x_i and y_i are the i:th values of the respective signals, then the MSE between the two signals is defined as:

$$MSE(x, y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2 \tag{8}$$

In later years, the utility of the MSE has been questioned with regard to the realm of image processing and the SSIM (structure similarity) index has been suggested as an improved or complementary measurement. In (Wang & Bovik (2009)) auhors Zhou Wang and Alan C. Bovik, the creators of the SSIM index, credit the new measurement with having an improved understanding of the fact that natural images usually have large structural similarities. To a greater degree SSIM identifies that image distortions causing "damage" to these similarites have a high impact on signal fidelity not identified by MSE. The SSIM for two signals (in this case video frames) $\mathbf{x}$ and $\mathbf{y}$ can be calculated by calculating the local SSIM at local patches of the image and averaging those calculated values.

This local SSIM considers the similarity for three elements of each image patch:

- $l(\mathbf{x}, \mathbf{y})$ luminance (brightness) similarity
- $c(\mathbf{x}, \mathbf{y})$ contrast similarity
- $s(\mathbf{x}, \mathbf{y})$ structure similarity

Given these elemental similarities the local SSIM is calculated as:

$$SSIM(\mathbf{x}, \mathbf{y}) = l(\mathbf{x}, \mathbf{y}) \cdot c(\mathbf{x}, \mathbf{y}) \cdot s(\mathbf{x}, \mathbf{y}) = \left(\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}\right) \cdot \left(\frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}\right) \cdot \left(\frac{\sigma_{xy} + C_3}{\sigma_x\sigma y + C_3}\right) \tag{9}$$

Where for x and y respectively:

- μ is the local sample mean.
- σ is the standard deviation of the local samples.
- σ_{xy} is the sample cross correlation for the images after their means have been removed.

5.6.1 SSIM cutoff

A basic quality measurement (see section 5.5.1) developed during the thesis project that determines quality based on SSIM data is the **SSIM cutoff**. The idea of this quality measurement is that the network should be trained to choose QP values that prevents the SSIM index of the resulting compressed frame from falling below a certain value, the **cutoff**. For all compressed frames with SSIM index above the cutoff value, quality is considered as equal. However, for those compressed frames with an SSIM index below the cutoff value, quality is considered to be worse the larger the difference between the SSIM index and the cutoff.

5.6.2 SSIM cutoff quality: Quality cost function

A quality cost function based on the SSIM cutoff quality presented in section 5.6.1 is presented here. Assume that for some frame F, there exists a number N of compressed versions of the frame f_i, each with an associated QP value.

Each f_i can be associated with some quality cost c_i, a scalar value that indicates the fitness of the QP value. The optimal QP is defined in this context as the QP value of the f_i with the smallest associated cost c_i.

This c_i is calculated using the associated frame size s_i and the SSIM index u_i for f_i. $\mathbf{p}$ is the penalty value, chosen as a very big constant as to have a significant impact on the total cost when weighed together with the frame size. p_i is the specific penalty for a compressed frame, defined in such a way that a penalty is invoked only when the associated SSIM index u_i is smaller than some cutoff value A.

$$p_i = \begin{cases} (A - u_i) \cdot \mathbf{p} & \text{if } A > u_i \\ 0 & \text{otherwise} \end{cases}$$

The cost c_i of a compressed frame f_i is defined as:

$$c_i = s_i + p_i \tag{10}$$

5.7 Object Detection

An object detection problem consists of identifying a set of objects in an image or video frame. Usually, this means that an algorithm needs to both be able to identify the type or label for an object, as well as the position and area the object inhabits in the image.

Much effort is put into the research area of object detection and neural networks are commonly used. The most prominent networks are based on convolutional networks, such as the Single Shot MultiBox Detector (SSD) (Liu et al. (2015)) and Region-based Convolutional Neural Networks (R-CNN) (Girshick et al. (2013)). For this thesis, another successful object detection network is used called the "You Only Look Once" (Redmon et al. (2015)) network, or YOLO for short. It is usually praised for its speed, although it can suffer in terms of accuracy with regard to other networks. Yolo was updated to version 3 (Redmon & Farhadi (2018)) in april of 2018, this is the version that is used in this thesis.

The YOLO detection network is used during the data generation phase to output object detection data, for a given frame this data is a list of detected objects with the following:

- A label. This indicates the object type (such as "car" or "truck").
- A confidence score. A value $0 < C < 1$ that indicates the certainty of the YOLO network that the detection is correct.
- Bounding box data. Cartesian coordinates for the box mid point along with width and height parameters, that allows one to draw a box around the detected object.

YOLO data is generated for all input frames, both the uncompressed version as well as all compressed versions.

5.7.1 Total detection difference

A quality measurement (see section 5.5.1) developed during the thesis project that determines quality based on the object detection data is the **total detection difference**. The total detection difference refers to the difference in number of detected objects between an uncompressed frame and a compressed version of it (see 5.1.4). For compressed frames where the total detection difference is zero the quality is considered equal. However, if the total detection difference is not equal to zero, quality is considered as worse the larger the size of the total detection difference.

5.7.2 Total detection difference: Quality cost function

This quality cost function is based on the total detection difference quality measurement presented in section 5.7.1. Assume that for some frame F, there exists a number N of compressed versions of the frame f_i, each with an associated QP value.

Each f_i can be associated with some quality cost c_i, a scalar value that indicates the fitness of the QP value. The optimal QP is defined in this context as the QP value of the f_i with the smallest associated cost c_i.

This c_i is calculated using the associated frame size s_i and number of detected objects d_i for the corresponding frame f_i. The number of detected objects for the uncompressed frame is defined as D. m is a penalty modifier, some scalar value $0 \leq m \leq 1$, used to determine how much of the total penalty should be applied when the total detection difference indicates one should be applied to the cost. p is the penalty value, chosen as a very big constant as to have a significant impact on the total cost when weighed against the frame size.

$$c_i = s_i + m \cdot |D - d_i| \cdot p \tag{11}$$

5.7.3 Matching

For some quality cost functions it is relevant to know to what degree objects detected in one frame can be detected again in another compressed version of the same frame. YOLO has no built in support for this, but provides bounding box coordinates for all object detections (see section 5.7). By comparing bounding boxes, one can determine which objects in the uncompressed frame have a matching object in some compressed version of the same frame.

When trying to find matching objects for a frame in a compressed version of that frame, the first step is to find the possible match candidates for the detected objects in the original frame. All possible

candidates found in the compressed frame are determined by checking if the bounding box midpoint coordinates of the boxes in the compressed frame can be found in the bounding box of a detected object in the uncompressed frame. Once all candidates have been identified, the one with the highest **match score** is chosen. This match score is the **intersection over union**, where the area of overlap for the bounding boxes is divided by the total area of the boxes:

$$\text{Match Score} = \frac{\text{Overlapping area of bounding boxes}}{\text{Total area of bounding boxes}} \tag{12}$$

5.7.4 Detection error data

An **ideal** definition of false positives and negatives with regard to object detection, with the specific case of YOLO (see section 5.7O) as detection algorithm is given by:

- **False positives**: Detected objects given by YOLO that are incorrect. This includes:
  - Detected objects which are not there, in other words if a bounding box has been drawn around an area that does not contain an object.
  - Detected objects with false labels. For example a car that has been labeled as a person.
- **False negatives**: Objects that YOLO has been trained to detect but which are not detected in the image or video frame.

Finding false positives and negatives according to this definition is in essence only something a human can do. In the context of this project, they will be defined in a more narrow sense, where the YOLO results in the uncompressed frame are considered to be **ground truth**. In other words the detected objects and bounding boxes for the uncompressed frame is the standard that the YOLO results for each compressed version of the frame should try to live up to. The compressed frame's ability to live up to this standard is given by the matching results (see section 5.7.3). The following **project-specific** definition will be used for false positives and negatives going forward:

- **False positive**: Objects that are detected in the compressed frame that have no corresponding matching object in the uncompressed frame.
- **False negative**: Objects that are detected in the uncompressed frame which have no matching object in the compressed frame.

5.7.5 Smoothing of detection error data

At one stage of the project, an effort is made to apply smoothing to the detection error data. The smoothing is an effort to reduce the impact of noise inherent in the data. This smoothing is conducted by convolving the data with a window employing a gaussian function (see equation 7), using $\mu = 0, \sigma = 2$ and window length $w = 9$. The preprocessed data is then used as input to the cost function in equation 13.

5.7.6 Detection error

A quality measurement (see section 5.5.1) developed during the thesis project that determines quality based on false positive / negative data (section 5.7.4) is **detection error**. The idea of this quality measurement is that quality is determined by the presence or absence of false positives and negatives with regard to object detection. All compressed frames with no such detection errors are considered to have the same quality. However, for those compressed frames where detection errors are present, the quality is considered worse the larger the number of detection errors.

5.7.7 Detection error quality: Quality cost function

A quality cost function based on the detection error quality measurement (see section 5.7.6) is presented here. Assume that for some frame F, there exists a number N of compressed versions of the frame f_i, each with an associated QP value.

Each f_i can be associated with some quality cost c_i, a scalar value that indicates the fitness of the QP value. The optimal QP is defined in this context as the QP value of the f_i with the smallest associated cost c_i.

This c_i is calculated using the associated frame size s_i, number of false negatives fn_i and false positives fp_i for the corresponding frame f_i. m is a penalty modifier $0 \leq m \leq 1$, the purpose of which it is to scale the impact of the penalty applied due to false positives / negatives in the compressed frame. p is the penalty value, chosen as a very big constant as to have a significant impact on the total cost when weighed together with the frame size.

$$c_i = s_i + m \cdot p \cdot (fp_i + fn_i) \tag{13}$$

6 Methodology

6.1 Equipment

Axis provides all equipment required to perform the required steps in the project. A stationary computer is used for most of the computing tasks, the main specifications are:

- **CPU**: Intel Xeon, 3.60 GHz
- **GPU**: GeForce GTX 1080 Ti
- **RAM**: 16 Gb

Further, an Artpec-7 camera outfitted with a special software patch was provided by Axis to be used when recording video.

6.2 Data generation

6.2.1 Recording video

All data used during training is obtained by first recording videos and then processing the videos. Videos are recorded using an Axis Artpec-7 camera. The camera has a special software patch that enables it to save data from the image processing software running on the camera (the **imaging pipeline**). A recording resolution of $1280x720$ is used.

Recorded videos are usually about an hour long. Each such video is split into approximately five minute long sequences. This is also were the splitting of the data into training, validation and test set takes place (section 5.2.3). Each five minute sequence is randomly assigned to one of the three set types, with the approximate $70 : 20 : 10\%$ split described in section 5.2.3.

As described in 5.1.2 the first few frames of each GOP (see theory for H264 in 5.1.1) are important. To reduce the amount of data generation, only the five first frames of each GOP in each video are chosen to be used as input examples.

6.2.2 Input data

As mentioned when describing the type of experience the network is to learn from (see section 5.2.1), it is decided that the input data for a given frame consist of:

- Encoding data.
- Camera imaging pipeline data.

An input feature vector for some frame would have **encoding** data for its history, in other words the frames that came before it. For each previous frame included, this data includes:

- Frame size (bytes).
- Compression rate values (QP, see section 5.1.1).
- Macroblock (section 5.1.1) data, that indicates how macroblocks were updated during encoding.

The same frame would have **imaging pipeline data** both for the current frame and the previous frames. For each frame, this data includes:

- Signal-to-noise ratio data from camera sensors.

- Parameters related to automatic white balance algorithm.
- Light intensity histogram values for pixel data.

The imaging pipeline data is easily accessible due to the aforementioned camera software patch. However, in practise the light intensity histogram data is not included in training. There are a lot of data points for the histogram. If this data is available for the current frame as well as multiple previous frames, this would warp the ratio of input features versus available training data, which in all probability would cause bad training results.

To obtain encoding data for the input examples, a **dummy encoding history** is produced for each video. This means that QP values are generated for frames in all videos that are semi-randomly assigned. For each GOP a random QP value is selected for the first frame. The following frames in the GOP are then given random values close to that first QP value. This is done to ensure a range in the compression rates for the fictional history. The video is then encoded with the QP values in the dummy history and the encoding data for each frame is saved.

6.2.3 QP output

Section 5.5.2 describes the reasoning for generating QP value labels for which to train. In short, quality measurements with associated quality cost functions (section 5.5.3) are used to generate optimal QP value labels for videos. The quality cost function needs frame sizes and data related to the quality measurement for each compressed version of a frame to calculate the corresponding cost.

For each frame slated to be used as a network input example, the frame is re-encoded with a range of QP values (± 20 QP values from the QP in the dummy history for the frame). For each such compressed version of the frame, the following data is generated to be used in the quality cost functions:

- Frame size.
- SSIM index, see section 9.
- YOLO data, as seen in section 5.7.
- False positive / negative data, see section 5.7.4.

6.3 Experiments

For clarity, the main experiments performed during the thesis project are outlined here.

6.3.1 Main Experiments

- **SSIM cutoff training** This is a pilot experiment where SSIM cutoff (section 5.6.1) is used as quality measurement and its associated quality cost function (equation 10) is used to generate a set of optimal QP values as input labels. The purpose of the experiment is to determine the feasibility of using video meta data to train for the QP value labels (section 5.5.2).
- **Total detection difference training** In this experiment, the idea is to use the total detection difference (section 5.7.1) as quality measurement. The purpose of the experiment is to determine network performance when training for the QP labels (section 5.5.2) obtained with the quality cost function (equation 11) of this quality measurement.
- **Detection error training** For this experiment, detection error (section 5.7.6) is used as quality measurement. The objective of this experiment is to determine network performance when training for the optimal QP value labels obtained with this quality measurement. Further, the penalty modifier in the quality cost function (equation 13) is adjusted to determine its effect on network performance.
- **Improving detection error training** This experiment looks at different ways of improving performance for networks using detection error as quality measurement (section 5.7.6). Smoothing of the false positive / negative data prior to using the quality cost function is attempted to reduce the influence of noise (see section 5.7.5). Further, the effects of regularization strategies such as data augmentation (section 5.4.2) and dropout layers (section 5.4.3) on generalization performance are studied.

- **Video encoding test** In this experiment, an experimental rate controller is implemented that uses the trained networks to encode videos. The intention is to look at the QP values chosen by a few trained networks and determine if they managed to adapt their output to video content. As part of the experiment, some degree of manual regularization of QP value is added to the rate controller, so that QP values can only change to a certain extent from frame-to-frame.
- **Data examination** The results from some of the experiments yields very high performance for the test set as compared to other set types. In this experiment, the data in each set-type and the corresponding performance is evaluated. The objective being to determine if there might be an issue with how data is split into training, validation and test sets (see section 6.2.1).

6.3.2 Box plots

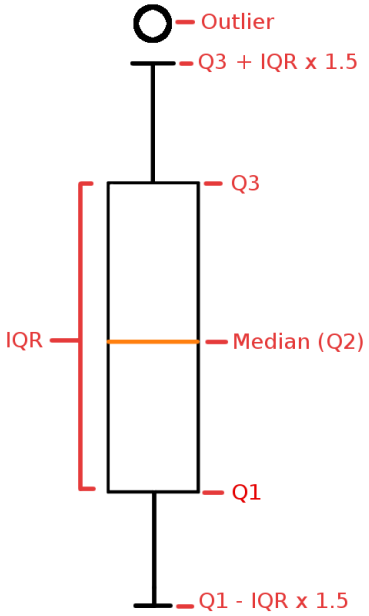


Figure 5: An example of a box plot as it appears in this report.

Box plots are used throughout the report to explain how values are distributed, figure 5 shows an outline of one such box plot. $Q1$, $Q2$ and $Q3$ refer to the distribution quartiles:

- $Q1$ and $Q3$ marks off the lowest 25% and highest 25% of values in the distribution respectively.
- $Q2$, or the median, marks off the highest 50% from the lowest 50% of the values respectively.
- The **interquartile range** (IQR) interval contains 50% of the values, which follows from the fact that $IQR = Q3 - Q1$. It is often used as a measure of the variability of the data.
- The notches attached to the thin lines below $Q1$ and above $Q3$ are the box plot whiskers. Whiskers can be determined in different ways, but the purpose is always to delineate the values that appear to belong in the distribution from outlier data points that are markedly different from the others. In the box plots for this thesis project, the circles are depicted as circles. In this report, the whiskers are always offset by a distance of $IQR \cdot 1.5$ from $Q1$ and $Q3$ (figure 5 is not to scale).

7 Training

Keras, with a Tensorflow backend is the software used to create the network models used in this project.

As mentioned earlier, all input features and the QP value labels are standardized prior to training according to equation 6. The standardized input vector for a given frame has 994 features, which include video meta data for the current frame and five previous frames (see section 6.2.2). Some of the available camera imaging pipeline data is not included during training, this is to prevent warping the relationship

between the number of input features for an input example and the number of available input examples (see section 6.2.2).

A multilayered perceptron architecture with 6 layers using the following architecture was employed for all trained networks (with the exception of a network trained with dropout layers):

- **Input Layer**: 994 nodes, rectified activation function.
- 4 **Hidden Layers**: 994 nodes, rectified activation functions.
- **Output Layer**: 1 output node, rectified activation function.

7.1 Trained networks

Below follows a list of networks which are considered to have interesting results or which are otherwise specifically mentioned in the text:

- **SSIM cutoff network**: The only network from the SSIM cutoff training experiment (section 6.3.1) mentioned in this report, all results in the results section 8.1 are given by this network.
- **Total detection difference network**: The only network from the total detection difference experiment (section 6.3.1) mentioned in this report, all results in the results section 8.2 are given by this network.
- **Detection error network 1**: One of the networks from the detection error training experiment (section 6.3.1), trained using QP value labels given by the quality cost function in equation 13 with $m = 0.2$.
- **Detection error network 2**: One of the networks from the detection error training experiment (section 6.3.1), trained using QP value labels given by the quality cost function in equation 13 with $m = 0.7$.
- **Smoothing detection error network**: A network from the improved detection error training experiment (section 6.3.1), trained using the same configuration as with detection error 1 but with a dropout layer (section 5.4.3). However, the false positive / negative data is smoothed according to the description in section 5.7.5 prior to being passed to the quality cost function.
- **Dropout detection error network**: A network from the improved detection error training experiment (section 6.3.1), trained using the same configuration as with detection error 1 but with a dropout layer (section 5.4.3) after the input layer as well as after all hidden layers. The dropout rate is $r = 0.2$ for all dropout layers.
- **Data augmented detection error network**: A network from the improved detection error training experiment (section 6.3.1), trained using the same configuration as with detection error network 1 but with data augmentation as described in section 5.4.2.

7.2 SSIM cutoff training

7.2.1 Introduction and purpose

In this experiment, the SSIM cutoff quality (section 5.6.1) is used as quality measurement. With the associated quality cost function (section 10), a set of optimal QP values (section 5.5.2) are generated for training.

As mentioned in section 6.3.1, the purpose of the experiment is to evaluate the degree to which one can train using video meta data and these optimal QP values. The expected outcome is that training and validation loss (see section 5.2.3) should both decrease over time. An acceptable performance with regard to residual interval ratio (as defined in section 5.2.2) is also expected for the training, validation and test sets.

7.2.2 Execution

To generate the QP value labels for training, a cutoff value is chosen as $A = 0.982$ for the quality cost function in equation 10. The penalty p in the quality cost function is set to the maximum frame size detected in all the training set examples.

The number of training epochs, learning rate and the L2 regularization hyper parameters (section 5.3.1) are varied during training.

7.3 Total detection difference

7.3.1 Introduction and purpose

For this experiment, the total detection difference (section 5.7.1) is used as quality measurement. Optimal QP values (section 5.5.2) are generated using the associated quality cost function in equation 11.

The objective of this experiment is to evaluate the ability of a network to train using these optimal QP values. A decrease in training and validation loss over time (section 5.2.3) is expected. It is also expected that an acceptable performance with regard to residual interval ratio performance (as defined in section 5.2.2) is attained for the training, validation, and test sets.

7.3.2 Execution

The optimal QP values for training is generated using the quality cost function, where:

- $m = 0.7$
- p is set to the maximum frame size detected in all the training data.

Different settings for training time, learning rate and the L2 regularization hyper parameters are used throughout the training process in an attempt to find the best training configuration (section 5.3.1).

7.3.3 Analysis

The YOLO data is analyzed with the total detection difference quality measurement kept in mind.

The bounding boxes for each compressed version of a sample frame are studied. Further, an individual plot for this sample frame is studied that shows how the number of detected objects varies for different compressed versions of the sample frame. The purpose here is to determine if a significantly different amount of detected objects is seen for the higher QP values. It is expected that the size of the total detection difference is larger for compressed frames with high QP values.

A set of box plots for a sample data set (the data set containing the above sample frame) is produced to study how the total detection difference (section 5.1.4) relates to QP value. The purpose is to determine if there is in general a greater total detection difference for compressed frames with higher QP values as compared to lower QP values. Seeing such a trend in the box plot is the expected outcome.

Another set of box plots is also produced for the same data set, but with detected objects that are filtered by confidence degree (all objects with $< 90\%$ YOLO confidence are removed). This is to determine if confidence filtering has an impact on total detection difference. Larger total detection differences are expected for the higher QP values when using confidence filtering, as compared to the unfiltered box plot.

7.4 Detection error training

7.4.1 Introduction and purpose

For this experiment, detection error (see section 5.7.6) is used to generate optimal QP values (section 5.5.2) using the associated quality cost function in equation 13. This quality measurement is an attempt at creating a more fine-grained version of the total detection difference quality measurement (section 5.7.1).

As described in section 6.3.1, the objective of this experiment is to evaluate the ability of a network to train for QP value labels generated based on the detection error quality measurement. Two different network configurations are evaluated. Each is defined by the value of the penalty modifier in the quality cost function (section 5.7.7). After training, the networks are both expected to have acceptable residual interval ratio performance (see definition of acceptable performance in section 5.2.2) for training, validation and test sets respectively.

7.4.2 Execution

QP value labels are generated using the quality cost function given by equation 13 for two different network configurations (detection error networks 1 and 2 in section 7.1):

- Penalty modifiers are set to $m = 0.2$ (network 1) and $m = 0.7$ (network 2).

- Penalty value p is set to the maximum frame size detected in all the training data for both configurations.

The number of training epochs, the learning rate and L2 regulation hyper parameters are varied throughout the training process to try and find good training setups for both network configurations (section 5.3.1).

7.5 Analysis

Histograms for the QP value labels (see section 5.5.2) that are generated using the quality cost function in equation 13 with different penalty modifiers $m = 0.2$ and $m = 0.7$ are created. The purpose of the histograms is to help determine the effect on the distribution for the QP labels when changing the penalty modifier. It is expected that when using the larger m value to generate QP values, smaller QP value labels are chosen to a greater extent.

Further, the relationship between the number of false positives / negatives and the amount of detected objects in the uncompressed frame is investigated. For a sample data set, the 100 frames with the largest and smallest amount of detected objects in the undetected frame are chosen. For both the large and small group, box plots showing the distribution of false positives / negatives at each QP are plotted. It is expected that the box plot distributions show that false negatives are more common when there are more detected objects in the uncompressed frame. More false positives are expected when the uncompressed frame has fewer detected objects.

The effects of first applying smoothing (section 5.7.5) to the false positive / negative data prior to use in the quality cost function is examined. The data has a lot of noise, and the smoothing is an attempt at reducing noise influence. An analysis is performed by outputting a set of quality cost function plots and trying to determine if there is some case where application of smoothing has caused a significant difference in the choice of QP label. It is expected that some case where the output label has changed significantly after smoothing has been applied to the data can be found.

7.6 Improving detection error training

7.6.1 Introduction and purpose

In this experiment, a few attempts are made at improving performance for networks based on detection error as quality measurement (section 5.7.6). Training set performance is high for both detection error networks 1 and 2 (see tables 3 and 4), but generalization performance is poor when looking at performance for the validation and test sets. Improving generalization performance is the main purpose of this experiment.

Although detection errors generally correlate with high QP (see figures 15-18) they also sometimes show up in strange places, seemingly due to random changes induced by compression (noise). The noise in the data causes penalties to be applied in strange places in the quality cost function (equation 13), which in turn can result in irrational choices of optimal QP value for the labels (section 5.5.2). Figure 19 shows the effect on the quality cost function results after applying smoothing to detection error data (as seen in section 5.7.5). A new, lower QP value is chosen as label after penalty peaks seemingly related to noise in the data are attenuated by the gaussian filter. If optimal QP values are generated using the quality cost function in equation 13 with smoothed detection error data, it is believed that that the optimal QP values will be chosen less randomly (i.e. less due to noise). Training for such optimal QP values is expected to lead to better generalization performance, as the network will be less prone to adapt to the noisy aberrations in the training data. Residual interval ratio performance (section 5.2.2) for the validation set is expected to improve.

Regularization strategies, such as data augmentation (section 5.4.2) and dropout layers in the network (section 5.4.3) are also applied in the hopes of improving generalization performance. A clear improvement in validation set performance for the residual interval ratio is expected for both cases.

7.6.2 Execution

For all described improvement approaches (smoothing, data augmentation and dropout), the same quality cost function is used (see equation 13). The same choice of penalty modifier ($m = 0.2$) and penalty value (p is set to largest frame size found in training set) is used.

Smoothing is applied to the detection error data (see section 5.7.5) prior to using it in the quality cost function (equation 13) to generate optimal QP values.

To apply dropout, the same network architecture as described in section 7 is used, but with added dropout layers (see "Dropout detection error network" in section 7.1). The dropout rate for each layer is 0.2, in other words 20% of all nodes in each dropout layer are deactivated each training epoch.

Data augmentation is carried out by applying gaussian noise to each input example as described in section 5.4.2.

7.7 Video encoding test

7.7.1 Introduction and purpose

As mentioned in the problem formulation (section 4), one of the main goals of the project is to evaluate the real-life performance of the trained rate controller networks. To do this, a simple implementation of a rate controller is realized.

From both the training and the test set, two videos are taken that reflect an **active** and **calm** scene respectively. Here, these terms are rather loosely defined. Active scenes are chosen so that they:

- Contain many objects (cars, people among others).
- Have consistent movement, in the sense that multiple objects tend to be on the move simultaneously at any given time.

Conversely, calm scenes feature:

- Few objects.
- Little to no moving objects throughout the sequence.

For each set, the active and calm sequence are spliced together to one video. The training set video is referred to as **AC1** ("Active-to-Calm 1") and the test set video as **AC2**. The expected result when the rate controller chooses compression rate for these videos, is that one should be able to detect a clear change in encoding QP values when the scene changes. It is also expected that the rate controller chooses QP values so that the active sequence has lower compression.

If an adaptation in encoding QP can be seen, it is also of interest to determine the stability and robustness of the implemented rate controller. Stability is here defined as a low local variance in QP in a short time interval. If QP changes by too much from frame-to-frame (for example from QP 20 to 50) this will have large implications for the quality of the video. Usually, Axis engineers try to prevent the QP from increasing by more than four or decreasing by more than six from frame-to-frame. Here, the expected and desired result is that QP does not change more than $\pm 4 - 5$ values from frame to frame. Regarding robustness, it is used interchangeably with generalization performance (section 5.2.3) in this context. The network is expected to perform in the same manner for the AC2 video from the test set as it does for video AC1.

7.7.2 Execution

A rate controller is implemented in different incarnations using the network based on detection errors (section 7.1, as discussed in section 9.2.2 it is believed that the detection error quality measurement (section 5.7.6) allows a more accurate basis for decision-making than the total detection error quality measurement (section 5.7.1).

Throughout the experiment, **manual regulation** is introduced, that prevents the QP value from decreasing more than 3 or increase more than 2 from the previous QP value.

7.8 Data Examination

7.8.1 Introduction and purpose

Test set performance is unexpectedly high for the results in general with regard to the residual interval ratio performance (section 5.2.2, tables 1-7). In the case of the total detection difference network (table 2) and the various detection error networks (tables 3-7), the test set performance is higher than the validation set performance. The performance in terms of residual interval ratio for the SSIM cutoff network (section 7.1 and table 1) has a better test set performance than validation and training set performance.

As mentioned in section 6.2.1, each recorded video is split into smaller sequences and randomly assigned as training, validation or test set data. The above results indicate that something might be off with the way the data has been split up into the different sets. At the same time, the test set performance could also just be the result of statistical variance. The goal here is to see if the distribution of the performance results of each smaller sequence seems to be very different for the different set types. Such a result would suggest that there might be an issue with the current selection process.

7.8.2 Comparison

The individual network performance results for the smaller sequences in a given set type are assumed to be normally distributed. For this analysis, mean square error (equation 4) will be used to measure performance, as it will enable a clearer comparison for this context.

For each set type, the mean square error is calculated for the associated smaller sequences. The **mean of the performance results** is the mean of all the mean square error results from the smaller sequences. Let:

- μ_a be the mean of the mean square errors for one set type.
- μ_b and μ_c be the corresponding means for the two other set types b and c.
- σ_b and σ_c be the standard deviations for the other set types.

Then the expected result, which indicates that the distribution of the set type a and the other set types are similar, is that:

- $\mu_b - \sigma_b \leq \mu_a \leq \mu_b + \sigma_b$
- $\mu_c - \sigma_c \leq \mu_a \leq \mu_c + \sigma_c$

The above interval for a set type is referred to its **established boundaries**. If the expected result is not obtained, this could indicate that the set types are not sampling video sequences from the same underlying distribution, in which case it is of interest to re-evaluate the selection process.

8 Results

8.1 SSIM cutoff training

8.1.1 Residual Interval Ratios (RIR)

QP interval	Training	Validation	Test
±1	0.5069	0.4844	0.5538
±3	0.9394	0.9269	0.9563
±5	0.9889	0.9878	0.9947
±7	0.9951	0.9938	0.9992
±10	0.9994	0.9980	0.9996

Table 1: Residual interval ratio (section 5.2.2) performance for the SSIM cutoff network (see section 7.1) after training.

The performance after training for the SSIM cutoff network (see section 7.1), according to the residual interval ratio (section 5.2.2) is seen in table 1. The training session for these results is conducted with learning rate 0.00001 and L2 regularization parameter set to 0.0001 (see section 5.3.1). According to the definition in section 5.2.2, the network has acceptable performance for training, validation and test sets, which is also the expected result (section 7.2.1). The fact that test set performance is higher than training and validation set performance is however a strange result. See section 7.8 for an investigation of this issue.

8.1.2 Loss curves

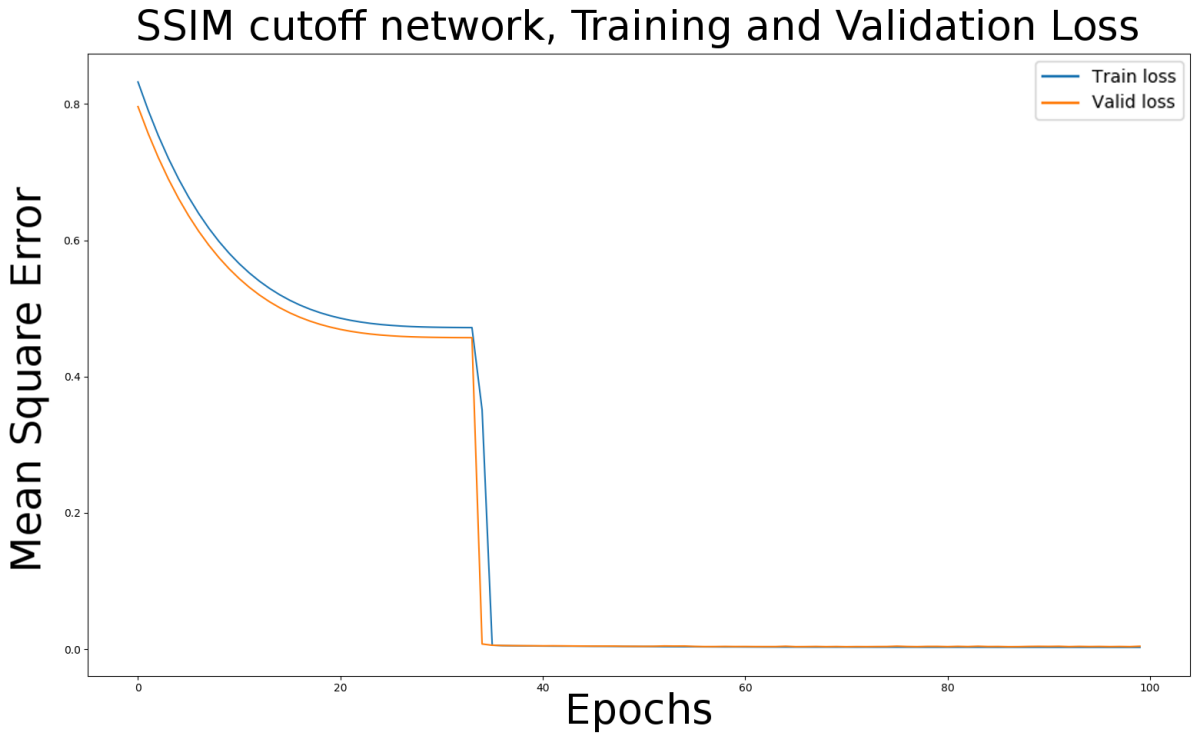


Figure 6: The training and validation loss plots (section 5.2.3) for the SSIM cutoff network (see section 7.1).

Figure 6 shows the training and validation loss (section 5.2.3) obtained while training the SSIM cutoff network. Both the training and validation loss clearly decrease with more experience, which is expected (section 7.2.1). A point of interest is the fact that validation loss is smaller than training loss until about epoch 40, after which the losses seem to converge, no clear explanation as to why can be determined at this point.

8.2 Total detection difference training

8.2.1 Residual Interval Ratios (RIR)

QP interval	Training	Validation	Test
±1	0.5285	0.1330	0.1741
±3	0.9353	0.3281	0.3969
±5	0.9949	0.4578	0.5232
±7	0.9998	0.5694	0.6162
±10	1	0.7003	0.7331

Table 2: Residual interval ratio performance (section 5.2.2) for the total detection difference network (see section 7.1) after training.

The residual interval ratio performance (section 5.2.2) after training for the total detection difference network (see section 7.1) is seen in table 2. The training is conducted with learning rate 0.00001 and L2 regularization term parameter set to 0.0001 (see section 5.3.1). There is a large difference in terms of the performance for the training set as compared to the validation and test set performance. This indicates poor generalization performance (see section 5.2.3). According to the definition in section 5.2.2, acceptable performance is attained for both the training and test set, but not for the validation set. The

expected result of having acceptable performance for all sets is thus not obtained (section 7.3.1). Test set performance is higher than validation set performance which is a bit odd. Another experiment (section 7.8) takes a closer look at this issue.

8.2.2 Loss curves

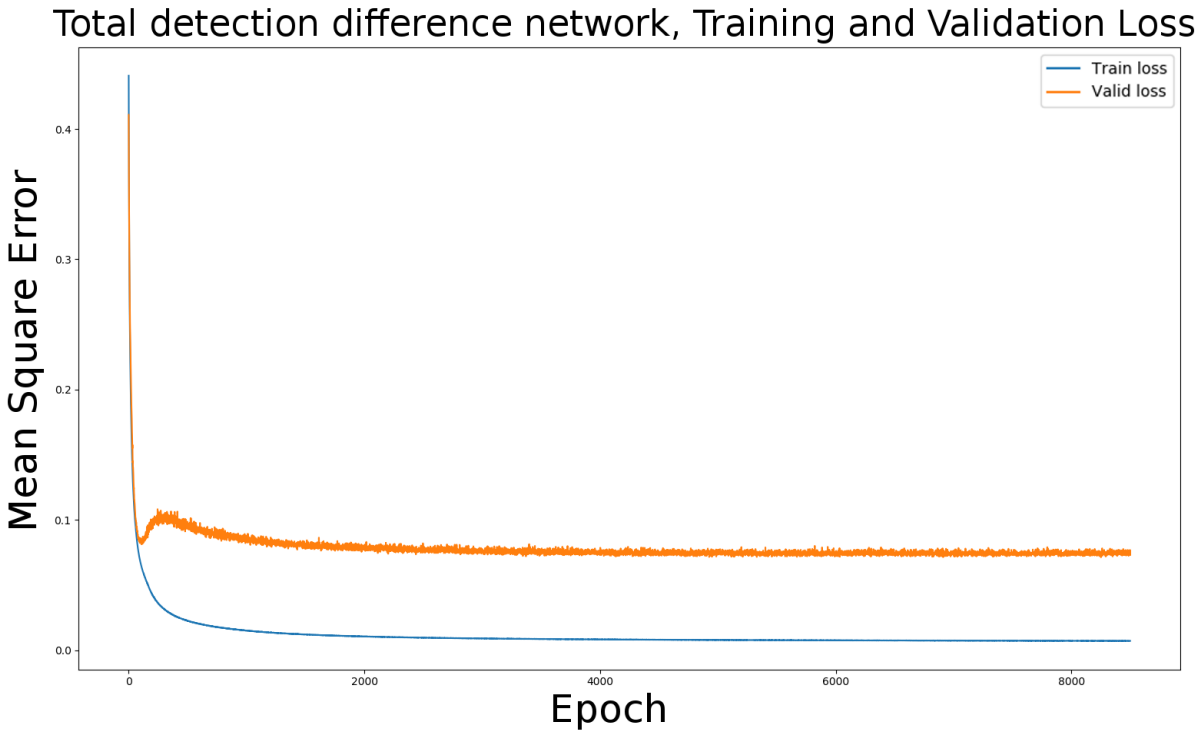
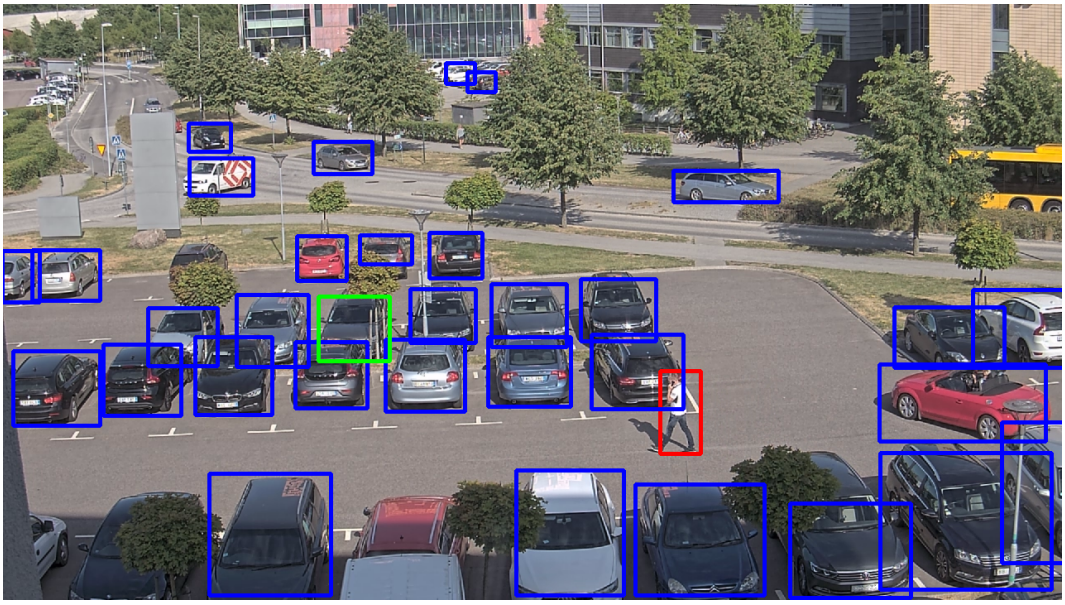


Figure 7: The training and validation loss curves for the total detection difference network (section 7.1).

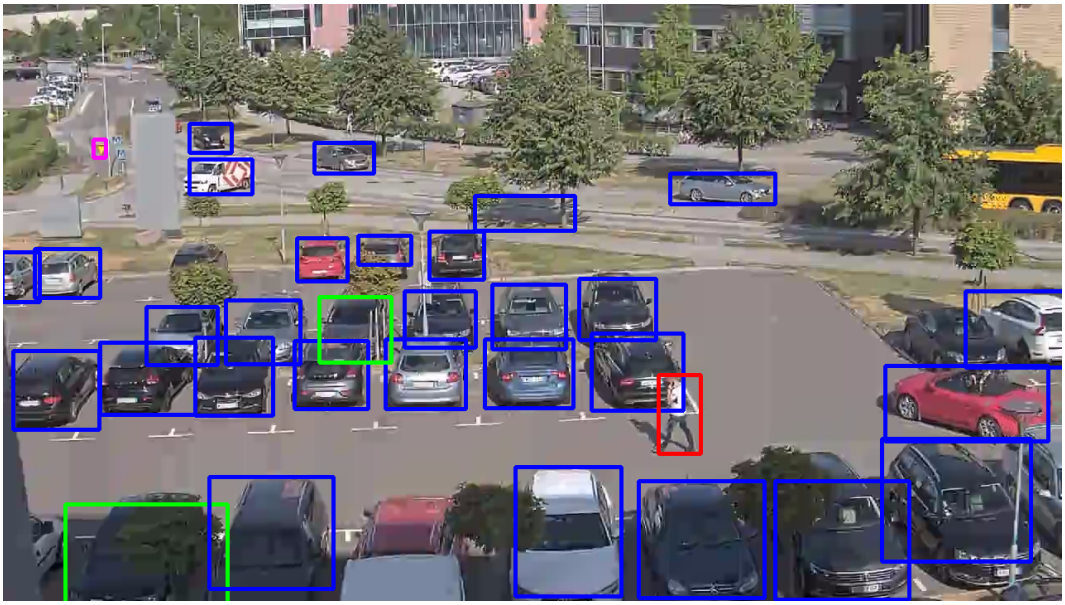
Figure 7 demonstrates the training and validation loss (section 5.2.3) obtained when training the total detection difference network (section 7.1). Both decrease over time, which is in line with what is expected (section 7.3.1). However, after a transient period both losses start decreasing very slowly and appear to have a seemingly constant and rather large distance in between them. This indicates generalization performance is quite poor (section 5.2.3) which is corroborated by the residual interval ratio performance results (section 5.2.2 and table 2).

8.2.3 Bounding box comparison

Comparison of YOLO bounding boxes in uncompressed and high compression version of a sample frame



(a) No compression



(b) QP 50

Figure 8: The two figures demonstrate bounding boxes generated by YOLO (see section 5.7) for two versions of the same sample frame. The top image shows an uncompressed version, on the bottom is a compressed version with the highest available compression at QP 50 (see section 5.1.4). Each bounding box color in the frames corresponds to one type of object label, as given by YOLO:

- **Blue**: Car
- **Green**: Truck
- **Red**: Person
- **Purple**: Street sign

Absent object detections in the compressed version of the sample frame

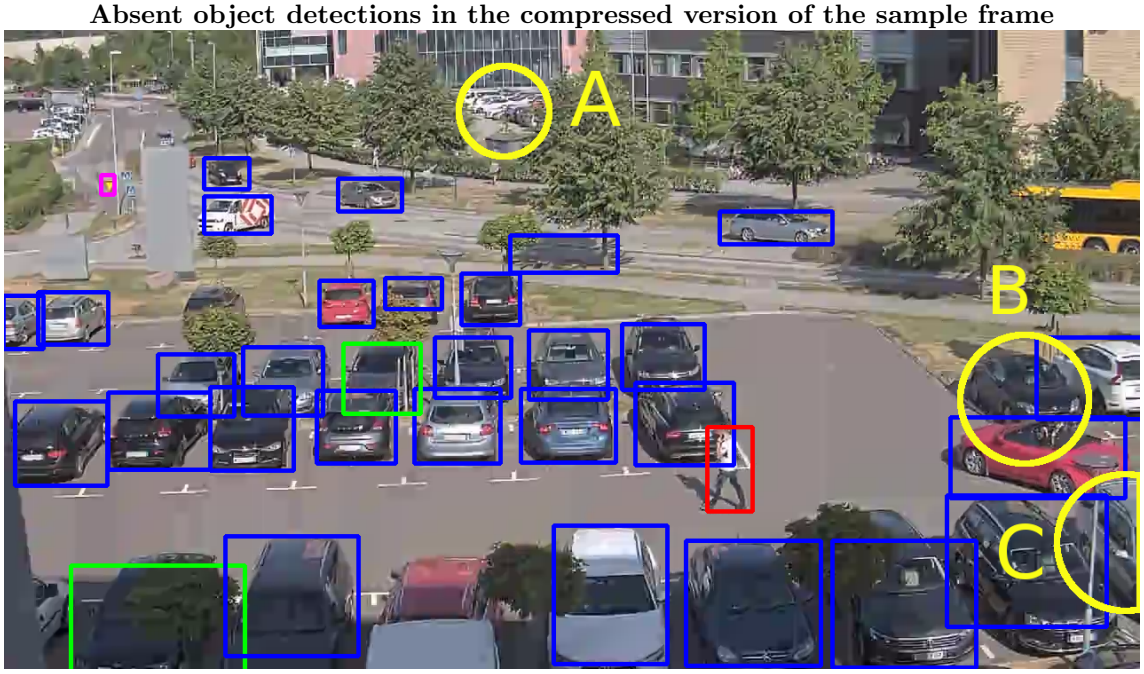


Figure 9: This figure shows the sample frame compressed at QP 50 from figure 8 with associated bounding boxes. The yellow circles mark areas where objects are detected in the uncompressed frame (section 5.1.4) but are not detected in this compressed version.

- **A**: Two cars are detected here in the uncompressed frame.
- **B**: A car is detected here in the uncompressed frame.
- **C**: A car is detected here in the uncompressed frame.

Section 5.7.4 demonstrates the ideal definition of false positives and negatives as well as the more narrow, project-specific definition used in this thesis project. According to the latter, where the uncompressed frame detection results are used as ground truth, examples (A), (B) and (C) are all examples of false negatives. In other words, the most compressed version of the sample frame has four false negatives in total with regard to the uncompressed frame.

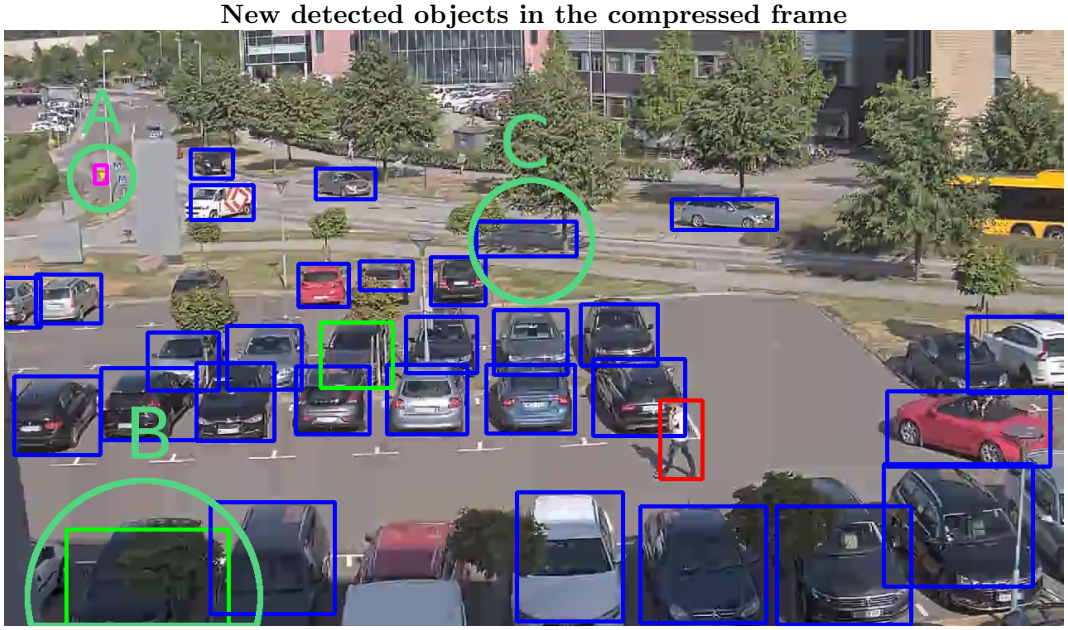


Figure 10: This figure shows the highly compressed frame (QP 50) from figure 8 with bounding boxes. The green circles show detected objects for the compressed frame that were not detected in the uncompressed frame:

- **A**: The detected object is a street sign, the object has the accurate label.
- **B**: This car has been detected, but as can be seen from the green bounding box, the wrong label (truck) has been applied to it.
- **C**: The detected object here is the shadow of a tree, which has been labeled as a car

Section 5.7.4 demonstrates an ideal, as well as a more narrow, project-specific definition of false positives and negatives in the context of object detection. (A) would not be a false positive according to the ideal definition, as it is a valid detection. However, the project-specific definition puts it in the same category as (B) and (C), as a false positive. As a result, this compressed version of the sample frame has a total of three false positives.

Figure 8 provides an example of YOLO network (see section 5.7) results for an uncompressed and compressed version of a sample frame (section 5.1.4). Aside from the presence of image artifacts in the compressed frame, there is also a noticeable difference in the YOLO results. Figures 9 and 10 provide a more detailed understanding, with circles indicating the difference in the results. YOLO has managed to detect several objects in the uncompressed frame that are not detected in the compressed frame, see figure 9. According to the project-specific definition of false positives and negatives in object detection (section 5.7.4) these are false negatives. At the same time, several other objects are detected in the compressed frame which are not detected in the uncompressed frame, see figure 10. Based on the project-specific definition, these are to be referred to as false positives. According to the ideal definition, only (B) and (C) in the figure are real false positives, as (A) is actually a valid detection. (B) has the wrong label (truck rather than car), while (C) should not be detected at all (it is just a tree shadow).

To be taken into account is that the total detection difference between the uncompressed frame and the most compressed frame is only one. Consider the detection error quality measurement and its associated quality cost function (section 5.7.1 and equation 11). According to the latter, individual penalties would be applied to the cost of this compressed frame for each of the four false negatives and three false positives. However, when using the total detection difference and its associated quality cost function (section 5.7.4 and equation 13) only one penalty would be applied, as the total detection difference is one. Based on this, the total detection difference quality measurement would suggest that quality is to be considered almost the same in terms of YOLO performance for the uncompressed frame and the most compressed frame, which is obviously not the case. Based on the YOLO output of this particular sample frame, it seems that the detection error quality cost function offers a more accurate, nuanced reflection of the change in YOLO performance due to compression.

8.2.4 Number of detected objects by label

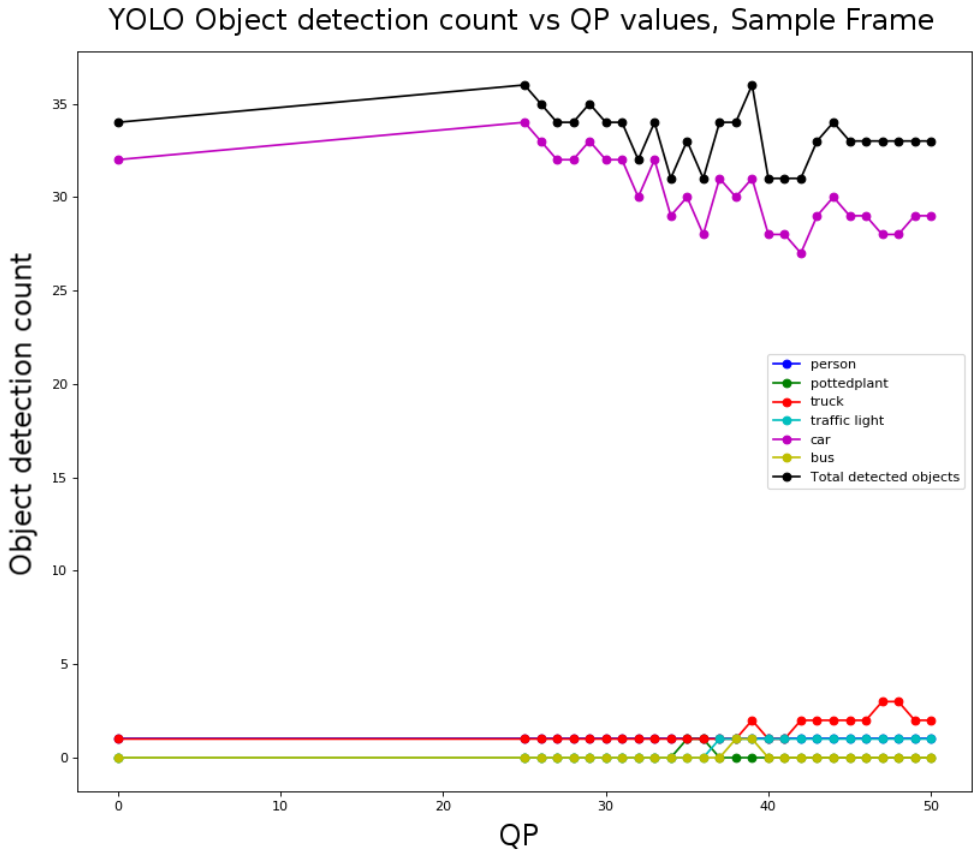


Figure 11: This plot shows the number of detected objects by label as well as the total number of detected objects as given by the YOLO network (see section 5.7). Each QP on the x-axis corresponds to the QP of a compressed version of the sample frame referred to in figures 8-10. QP 0 refers to the results for the uncompressed version of the sample frame (section 5.1.4).

The sample frame object detection results at a wider array of compression rates is seen in figure 11. Note that there is quite a lot of variance in the number of detected objects. There is no consistent trend in regard to whether the number of objects decrease or increase with growing QP. The total detection difference (see section 5.7.1) between the uncompressed frame and the most compressed frames (QP $45 - 50$) is only 1. At the same time a few of the compressed frames with lower QP have a greater total detection difference (section 5.7.1). This goes against what is expected in section 7.3.3, where it is assumed that higher QP values would have a greater total detection difference. With regard to total detection difference as quality measurement, it also seems a bit strange. For this particular case, a frame compressed at QP 33 is attributed with a lower quality than a frame compressed at QP 50 according to the total detection difference quality measurement, despite the fact that QP 50 results in a much heavier compression.

8.2.5 Total detection difference box plots

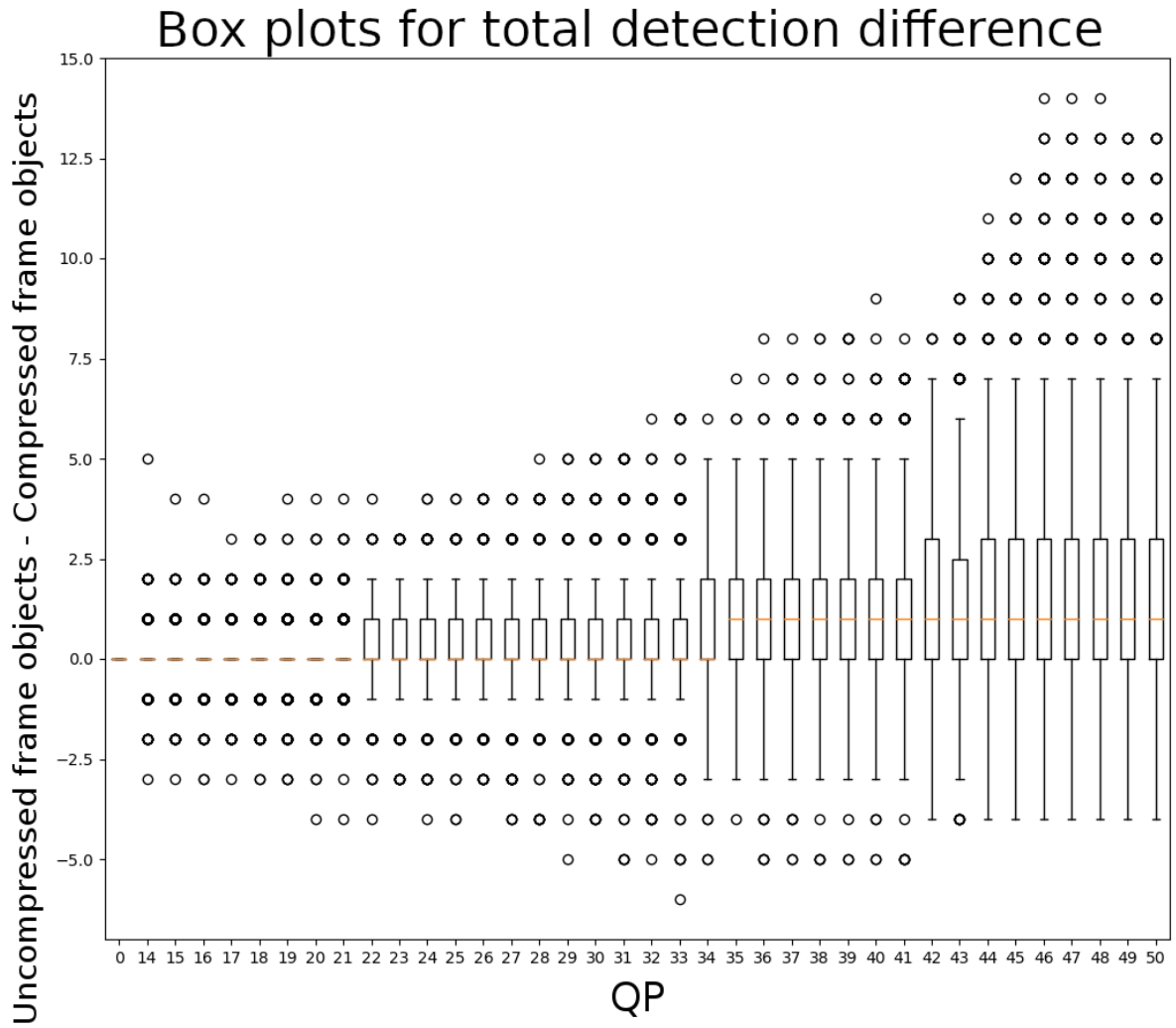


Figure 12: This figure illustrates the distribution for the total detection difference (section 5.7.1) at different compression rates for a sample data set. The distributions are illustrated with box plots (see section 5).

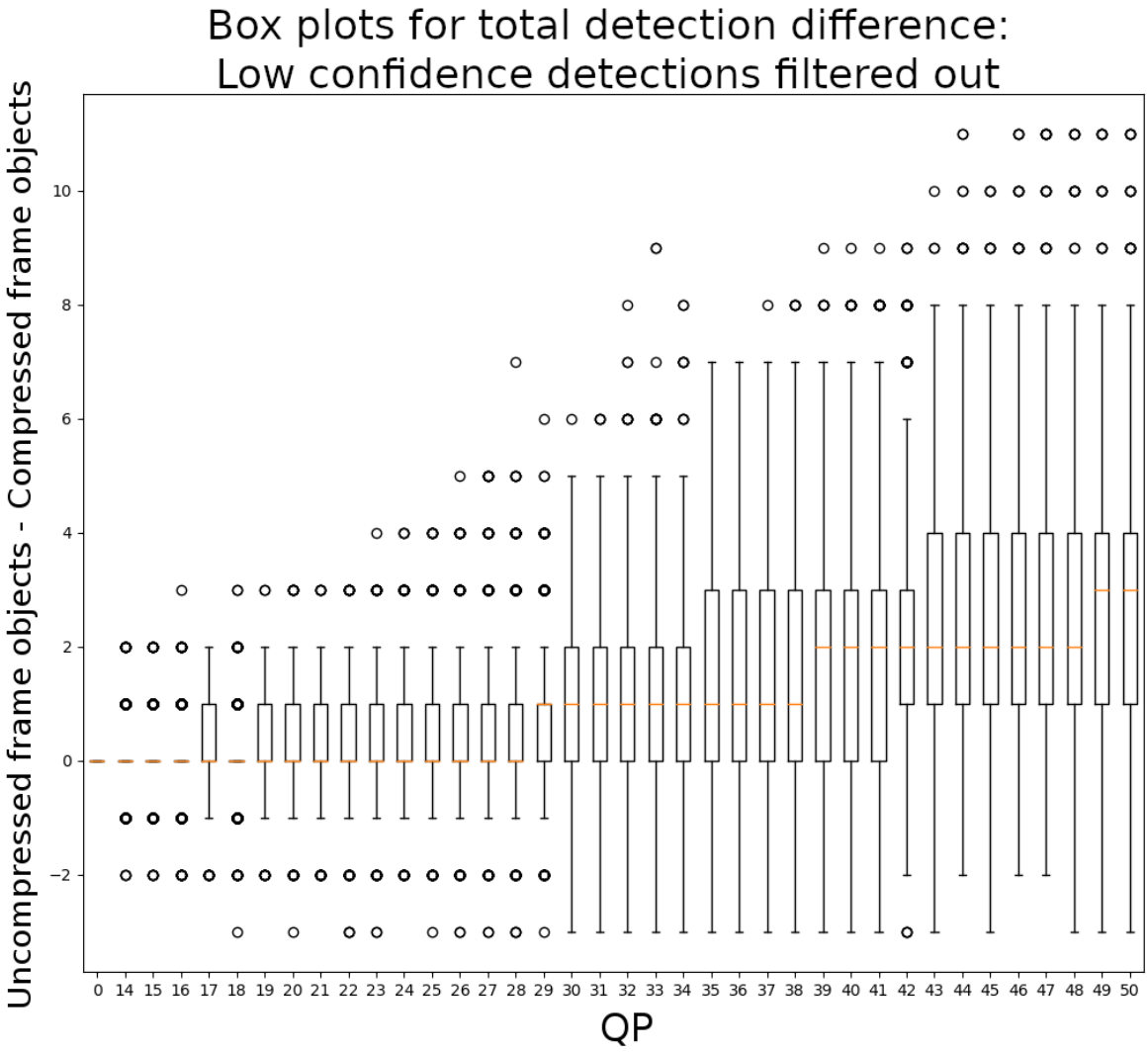


Figure 13: This figure, like figure 12, illustrates the distribution for the total detection difference. However, in this case all detected objects with a YOLO confidence < 90 % are filtered out to test the impact of object detection confidence on the distributions.

Figures 12 and 13 show distributions for the total detection difference 5.7.1 in two different cases. Figure 12 takes all objects detected by YOLO into account. The latter, figure 13, filters out the detected objects that had a YOLO confidence score smaller than 90 % (section 5.7).

The box plots in these figures aim to illustrate the distribution of total detection difference. They are also intended to be used to determine if compressed frames tend to have more or less detected objects than the uncompressed frames. Each box plot in these figures demonstrates the distribution for the total detection difference for all frames compressed at a given QP in a sample data set. The sample data set is the video sequence of the sample frame from figures 8-11 in section 8.2.3 and 8.2.4. Results in sections 8.2.3 and 8.2.4 do not suggest that higher QP values have a significantly different total detection difference. Those results are not in line with the expected outcome (see section 7.3.3).

However, the box plots in figures 12 and 13 do suggest that the expected outcome does hold true for the general case, as the higher QP values have larger sizes for total detection difference for both figures. For example, looking at figure 12, this is evidenced by the larger area taken up by the interquartile range (IQR) and box plot whiskers (see section 6.3.2 for box plot explanation) for QP values > 34. Figure 13 also shows a larger total detection difference for frames compressed with higher QP. As seen in the latter case, the trend seems even more prominent when lower confidence detections are filtered out: The box plot whiskers are extended further and the IQR takes up more area, the median level also moves further away from zero as compared to figure 12. This indicates that the total detection difference is generally larger after all low-confidence object detections are removed, which is also the expected result.

For both figures the IQR and the median for the box plots are always ≥ 0, which means it can be

determined that positive differences are by far most common. A positive difference means the compressed frame has fewer detected objects. These results suggest that compressed versions of frames tend to have fewer detected objects than the uncompressed frame and that higher compression entails fewer detected objects. A comparison of figure 12 and 13 in this regard also suggests that confidence filtering leads to a smaller number of detected objects for the higher compression versions of the uncompressed frame.

8.3 Detection error training

8.3.1 Residual Interval Ratios (RIR)

QP interval	Training	Validation	Test
±1	0.8006	0.2174	0.2733
±3	0.0.9915	0.3682	0.4207
±5	0.9997	0.4957	0.5463
±7	0.9999	0.6038	0.6469
±10	1	0.7320	0.7619

Table 3: Residual interval ratio performance (section 5.2.2) for detection error network 1 (see section 7.1) after training.

Table 3 shows the residual interval ratio performance results (section 5.2.2) after training for detection error network 1 (see section 7.1). The learning rate is set to 0.00001 and L2 regularization parameter is set to 0.0001 during training (see section 5.3.1). Training set and test set performance lives up to the level required for acceptable performance (as seen in section 5.2.2). Performance for the validation set is however just below the mark at ≈ 49 % in the ±5 interval. Generalization performance is clearly poor as validation and test set performance is much worse than training set performance. The fact that test set performance is better than validation set performance is a strange result and one that is investigated further (section 7.8).

QP interval	Training	Validation	Test
±1	0.8243	0.1935	0.2546
±3	0.9956	0.3492	0.4043
±5	0.9998	0.4813	0.5311
±7	1	0.5996	0.6304
±10	1	0.7274	0.7376

Table 4: Residual interval ratio performance (section 5.2.2) for detection error network 2 after training (see section 7.1).

The residual interval ratio performance (section 5.2.2) for detection error network 2 (see section 7.1) is seen in table 4. Training is conducted with learning rate 0.00001 and L2 regularization parameter set to 0.0001 (see section 5.3.1). The results for the network are very similar to those seen in table 3 for detection error network 1. The network performs very well for the training set, but generalization performance is poor with regard to validation and test set performance. The test set performance is above the level required for acceptable performance (section 5.2.2), however performance for the validation set is just below the mark ($\approx 48\%$ inside the ±5 interval) (see section 7.8 for investigation into why test performance is better here).

8.3.2 Optimal QP distribution

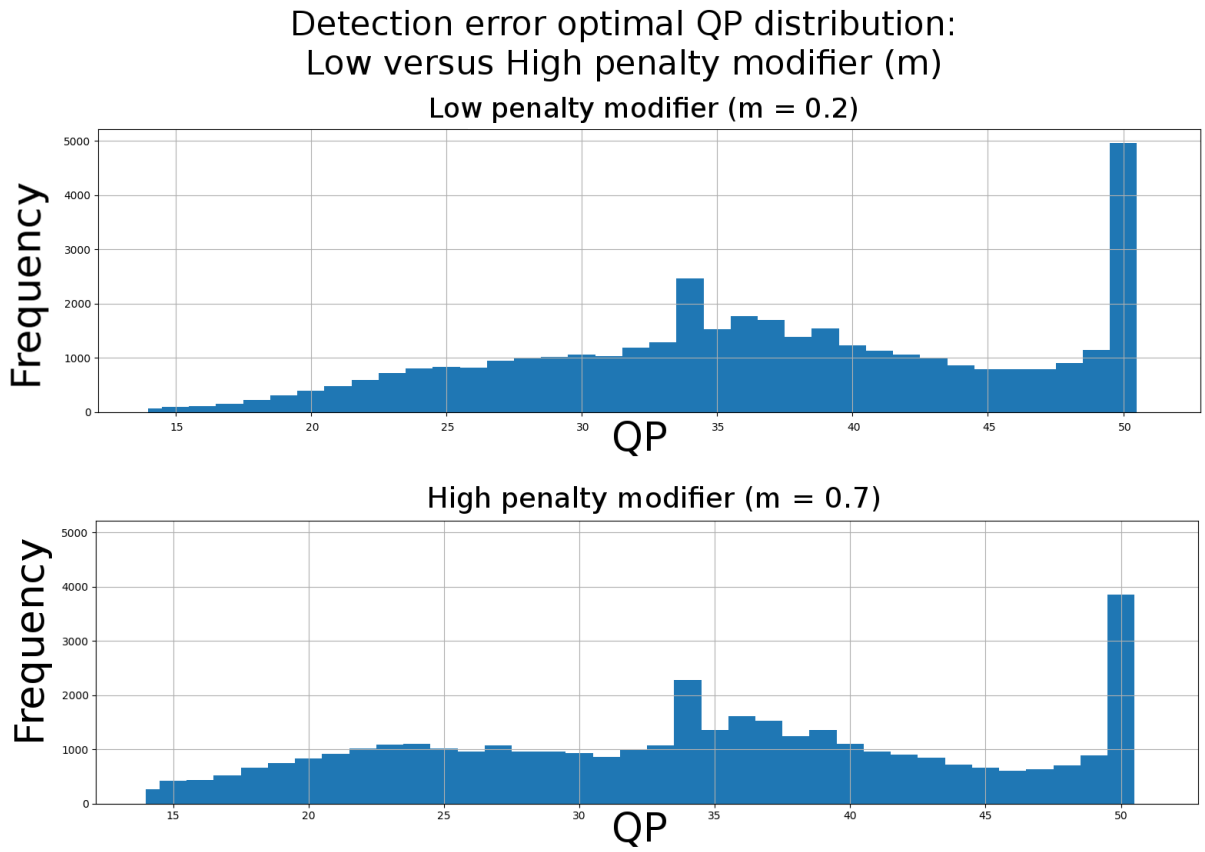


Figure 14: The figure shows two histograms, each displaying the distribution of QP value labels (section 5.5.2) provided by the quality cost function in equation 13 for a sample data set. The penalty modifier (m in the quality cost function definition) is set to a different value for each histogram.

Figure 14 shows optimal QP output (section 5.5.2) from the quality cost function in equation 13. The top histogram has a smaller penalty modifier value m set in the quality cost function (section 5.7.7) than the bottom histogram. A smaller value for the penalty modifier m reduces the size of each individual detection error penalty, while a larger m increases penalty size. As mentioned in section 7.5, it is expected that lower QP values are generally chosen to a greater extent when m is large. The reason for this is that compression with high QP values is expected to result in more detection errors, imposing larger penalties should therefore inflate the associated cost of high QP values (see equation 13). This is confirmed in the histograms: When comparing the histogram of the higher m ($m = 0.7$) to the histogram of $m = 0.2$, the high peak at QP 50 is attenuated and the left flank of the distribution grows slightly bigger. In other words, the distribution of the QP values has changed with the higher m so that lower QP values are chosen to a greater extent.

8.3.3 False positives / negatives versus uncompressed frame detected objects

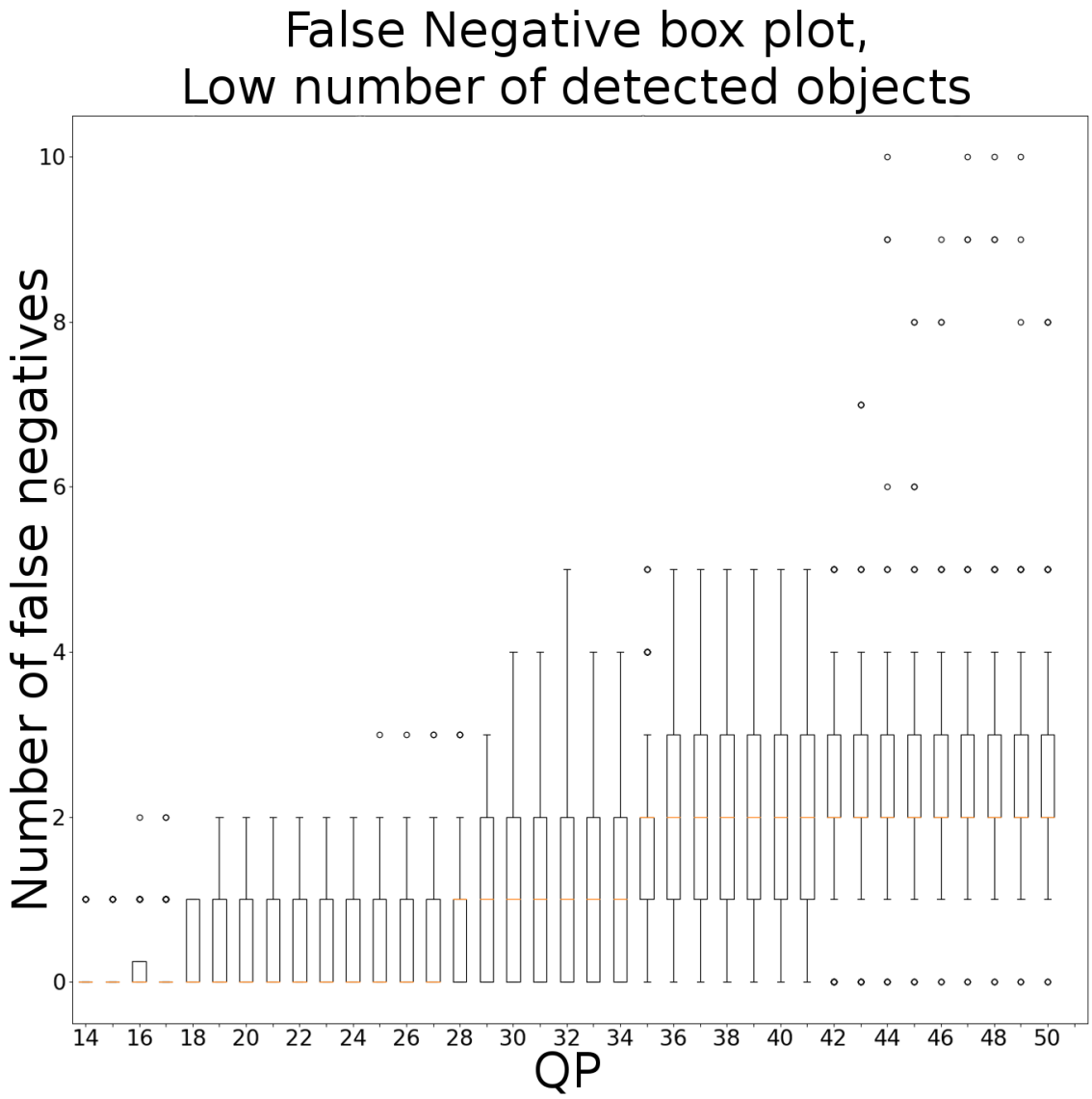


Figure 15: This box plot shows the distribution of false negatives (see section 5.7.4) for frames compressed with different QP values in a sample data set. In this figure, only the 100 frames in the data set with the fewest detected objects in the uncompressed frame are included. For the included frames, the lowest amount of detected objects in the uncompressed frame is 27 while the highest amount is 30.

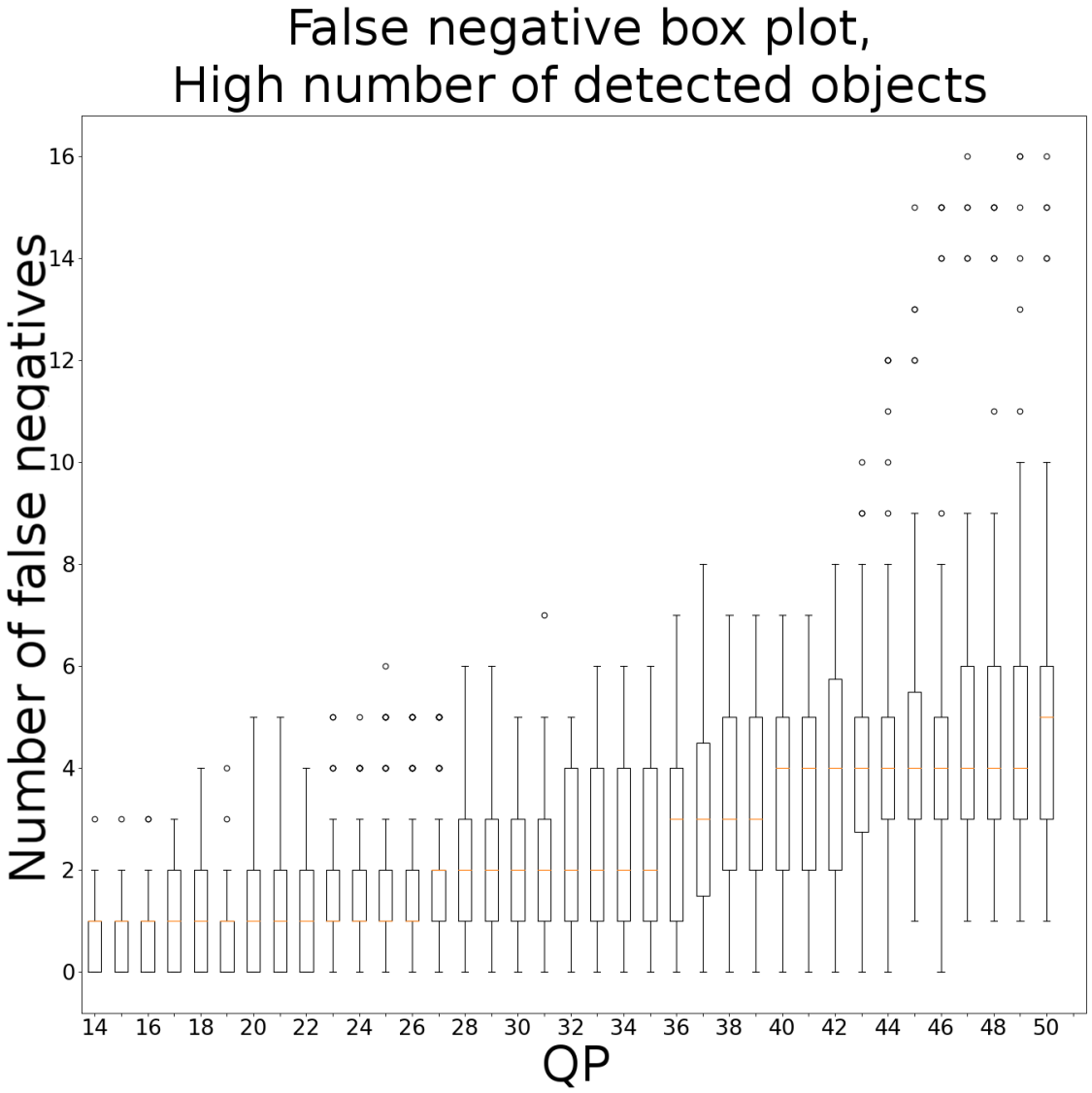


Figure 16: This box plot shows the distribution of false negatives (see section 5.7.4) for frames compressed with different QP values in a sample data set. Only the 100 frames in the data set with the most detected objects in the uncompressed frame are included. For the included frames, the lowest amount of detected objects is 35 and the highest amount is 38.

Figure 15 and 16 provide information about false negatives. They show the distribution of false negatives (section 5.7.4) for frames compressed with different QP values in a sample data set using box plots. The purpose of the figures is to determine if there is a relationship between the number of detected objects in the uncompressed frame (see section 5.1.4) and the number of false negatives (see section 7.5).

Figure 15 only includes the 100 frames in the data set with the fewest detected objects in the uncompressed frame. For the frames considered here the lowest number of detected objects in the uncompressed version of the frame is 27, the highest number being 30. Figure 16 includes instead the 100 frames in the data set with the most detected objects in the uncompressed frame. For this figure the lowest number is 35, while the highest is 38.

Both figures clearly demonstrate that the number of false negatives is greater for high QP values, as indicated by the the fact that both the median and the interquartile range (IQR) are shifted upwards as QP increases (see section 5 for explanation of box plot).

Figure 16 demonstrates a higher number of false negatives, as evidenced by the median and size of the IQR. For example, the highest QP values in this plot have a median false negative of ≈ 2 while in figure 16 it is ≈ 4. It appears then that the number of false negatives do increase with the number of detected objects in the uncompressed frame. The two main results obtained here is that false negatives increase with QP and that false negatives are more common when the uncompressed frame has a greater

number of detected objects.

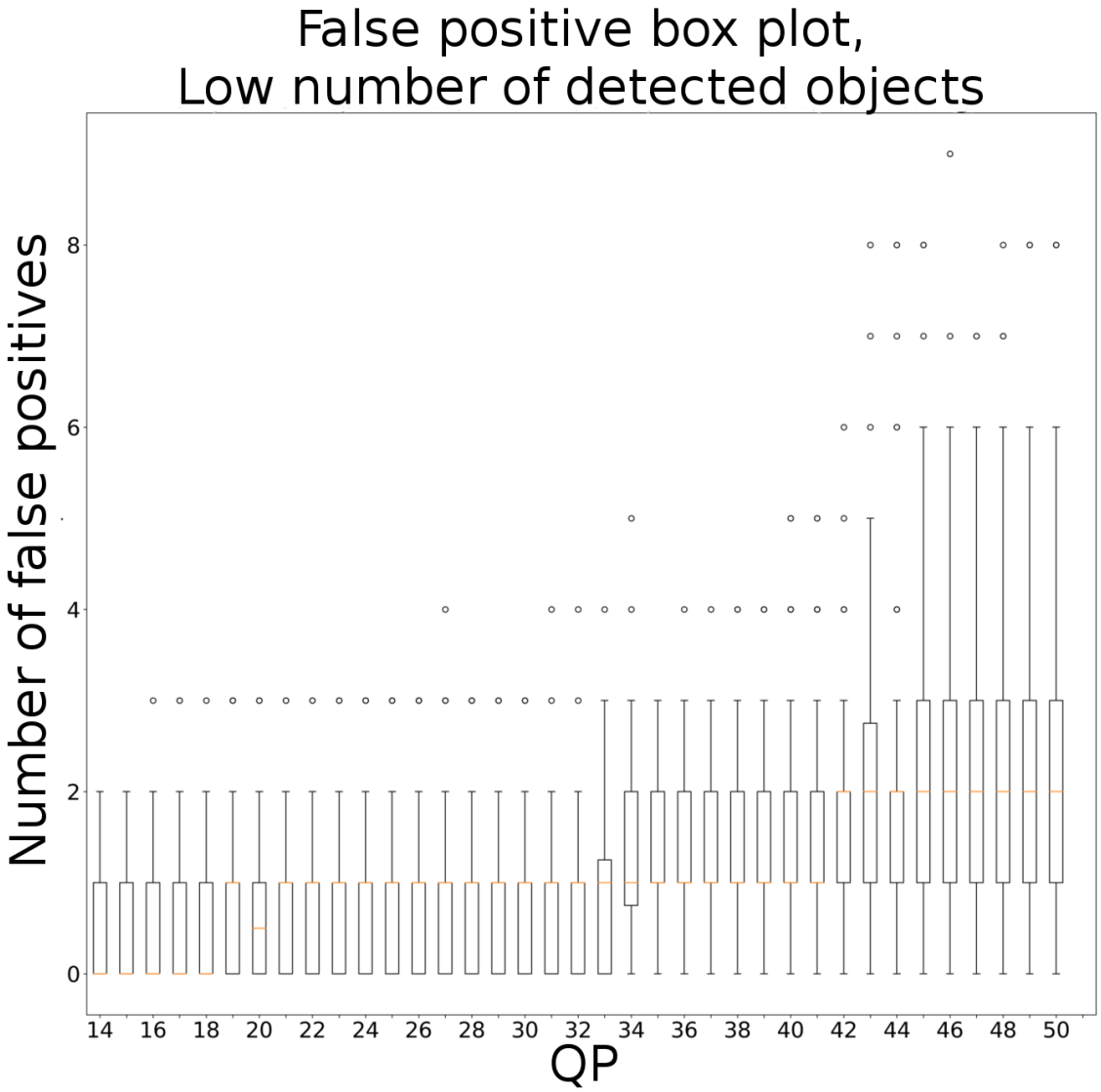


Figure 17: This box plot shows the distribution of false positives (see section 5.7.4) for frames compressed with different QP values in a sample data set. In this figure, only the 100 frames in the data set with the fewest detected objects in the uncompressed frame are included. For the concerned frames, the lowest number of detected objects in the uncompressed frame is 27, the highest number is 30.

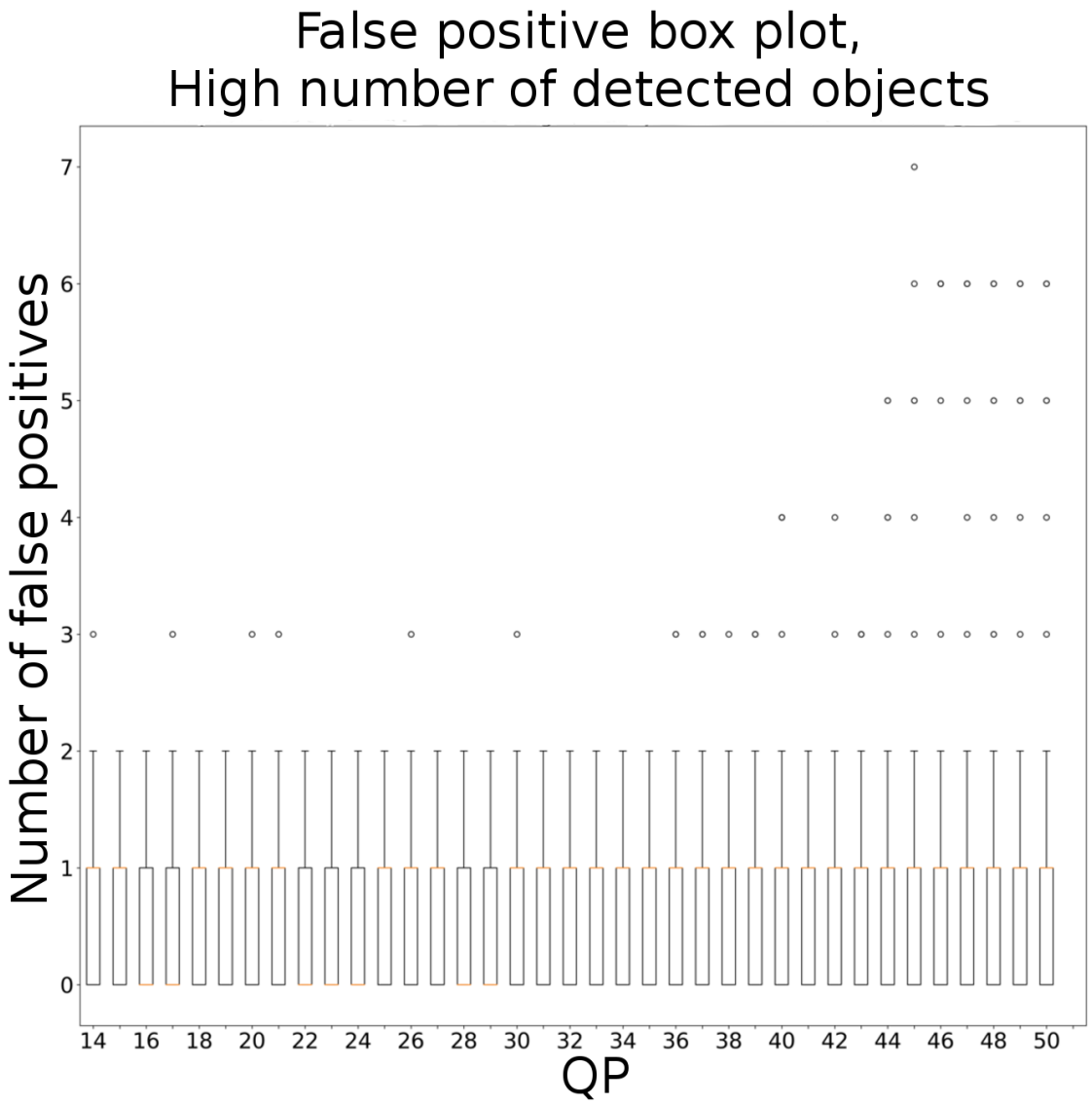


Figure 18: This box plot shows the distribution of false positives (see section 5.7.4) for frames compressed with different QP values in a sample data set. In this figure, only the 100 frames in the data set with the most detected objects in the uncompressed frame are included. For the included frames, the highest number of detected objects in the uncompressed frame is 35, the highest number is 38.

In figures 17 and 18 information is given regarding false positives. The figures display the distribution of false positives (section 5.7.4) for frames compressed with different QP values in a sample data set using box plots. The purpose of the figures is to determine if there is a relationship between the number of false positives (see section 7.5) and the number of detected objects in the uncompressed frame (see section 5.1.4).

As in figures 15 and 16, each figure has information for a specific set of frames. Figure 17 includes only the frames with the lowest amount of detected objects in the uncompressed frame, while figure 18 has information for the frames with the highest amount of detected objects. For figure 17 the lowest and highest amount of detected objects in the uncompressed frame is 27 and 30. For figure 18 those numbers are 35 and 38.

In figure 17, the number of false positives seems to be greater for higher QP values, as is indicated by the fact that both the median and the interquartile range (IQR) are shifted upwards as QP grows (see section 5 for explanation of box plot). However, this does not seem to be the case for figure 18, where the distributions look pretty similar for all compression rates. Therefore, it cannot be established with the same certainty as is done with false negatives in figures 15-16 that frames compressed at high QP values have more false positives.

Figure 17 does seem to have a larger amount of false positives when compared with figure 18. This

indicates that smaller numbers of detected objects in the uncompressed frame entails more false positives. This is the expected result.

8.3.4 Smoothing

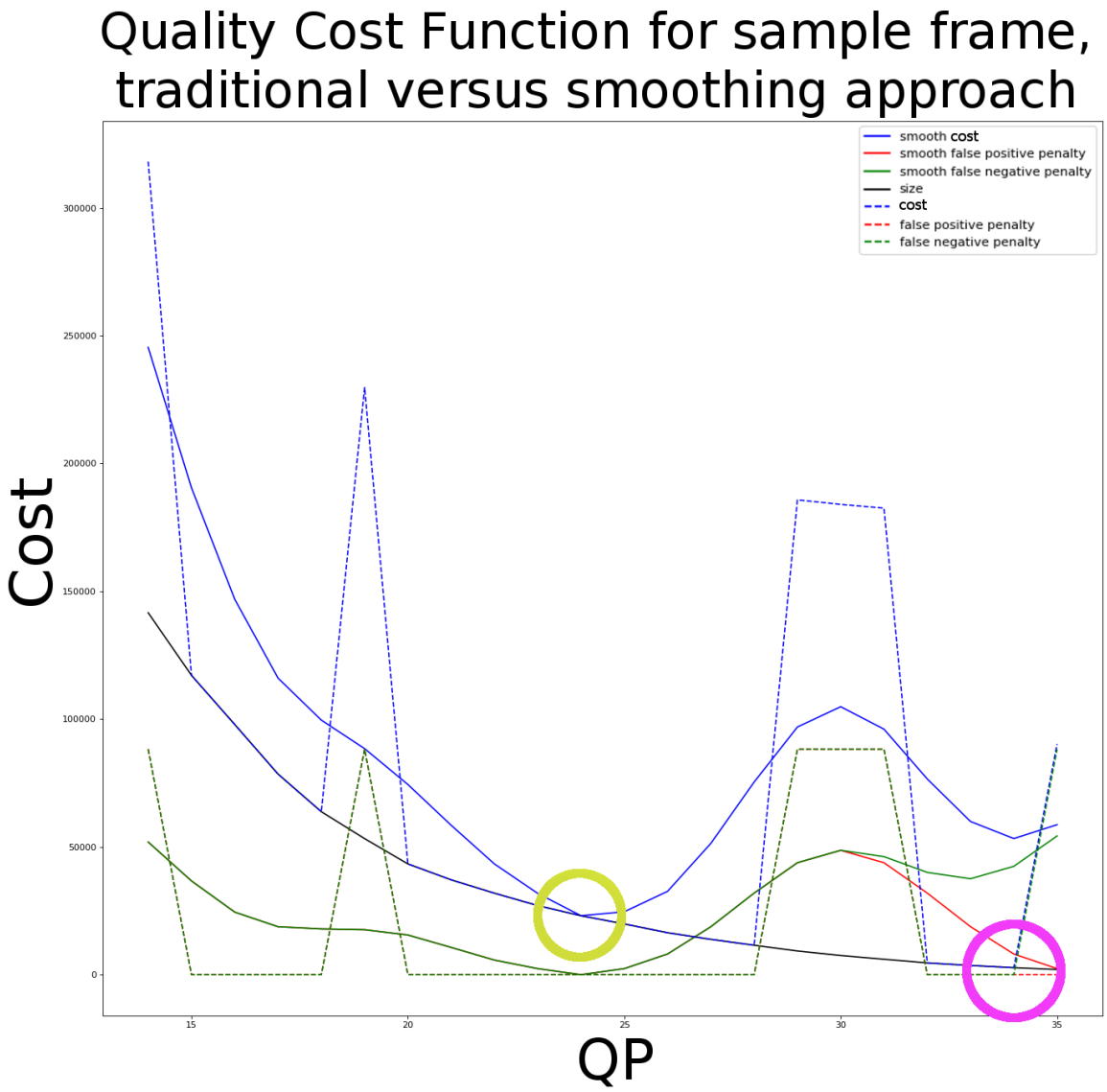


Figure 19: The quality cost function for detection error (equation 13) using smoothed detection error data (section 5.7.5). The original quality cost function and its associated penalties are also plotted with dashed lines. The purple circle indicates the minimum cost attained with the original function, while the yellow circle indicates the minimum cost for the smoothed data function.

Figure 19 shows the quality cost function for detection error (equation 13) using smoothed detection error data (section 5.7.5) alongside the original quality cost function. As in the case with the example quality cost function (figure 4), there is a frame size (black curve) that makes up the total cost with the associated penalties. The penalties applied due to false positives and negatives are the same size throughout the plot, except for the highest QP values where there are seemingly no false positives. As the figure indicates, the minimum costs are clearly quite different for the two functions. For this specific frame, the effect of applying smoothing is that a new, lower QP value has been chosen. As expected in section 7.5, a case where the optimal QP output is clearly altered has been identified. It would seem that smoothing has greatly decreased the influence from the detection error penalties at QP 19, 30 and 35.

8.4 Improving detection error training

8.4.1 Residual Interval Ratios (RIR)

QP interval	Training	Validation	Test
±1	0.7773	0.1842	0.2481
±3	0.9875	0.3149	0.3919
±5	0.9996	0.4348	0.5152
±7	1	0.5425	0.6197
±10	1	0.6791	0.7409

Table 5: Residual interval ratio performance (section 5.2.2) for the smoothing detection error network (section 7.1) after training.

Table 5 demonstrates the residual interval ratio performance for the smoothing detection error network (section 7.1) after training. The training is conducted with learning rate 0.00001 and L2 regularization parameter set to 0.0001 (section 5.3.1). Compare the results to the residual interval ratio performance for the same network, but where no smoothing is applied prior to generating optimal QP values (detection error network 1 in table 3). The performance is better without the smoothing for all set types. Why test set performance is higher than validation set performance is investigated further (section 7.8).

QP interval	Training	Validation	Test
±1	0.6227	0.1805	0.2159
±3	0.9568	0.3932	0.4494
±5	0.9946	0.5399	0.5840
±7	0.9994	0.6495	0.6772
±10	1	0.7612	0.7737

Table 6: Residual interval ratio performance (section 5.2.2) for the dropout detection error network (section 7.1) after training.

The residual interval ratio performance results (section 5.2.2) after training for the dropout detection error network (section 7.1) are seen in table 6. As compared to the corresponding network result where no dropout is applied (detection error network 1 in table 3) the performance is worse for the training set, but in some ways better for the validation and test sets. Acceptable performance for the validation set (according to section 5.2.2) is attained when using dropout. In the ±1 interval the original network is still the best, but for the other intervals the ratios are slightly higher with the dropout detection error network. The training is conducted with learning rate 0.00001 and L2 regularization parameter set to 0.0001 (section 5.3.1). Test set performance is higher than validation set performance, an issue which is further investigated (section 7.8).

QP interval	Training	Validation	Test
±1	0.7792	0.2016	0.2618
±3	0.9829	0.3777	0.4336
±5	0.9982	0.5041	0.5590
±7	0.9999	0.6026	0.6545
±10	1	0.7309	0.7659

Table 7: Residual interval ratio performance (section 5.2.2) for the data augmented detection error network (section 7.1) after training.

Table 5 demonstrates the residual interval ratio performance for the data augmented detection error network (section 7.1) after training. As compared to the corresponding network result where no data

augmentation is applied (detection error network 1 in table 3) the performance is worse for the training set, but more unclear in regard to validation and test sets. For the ±1 interval of the test and validation set the data augmented network performs worse. For the test set, ratios are slightly better in the other intervals but for the validation set performance the results pretty much break even. An acceptable residual interval ratio performance (as defined in 5.2.2) for the validation set is achieved when using data augmentation. Largely the ratios for test and validation set do not seem to have changed by much. Regarding the high test set performance in relation to validation set performance, please see section 7.8.

8.5 Video encoding test

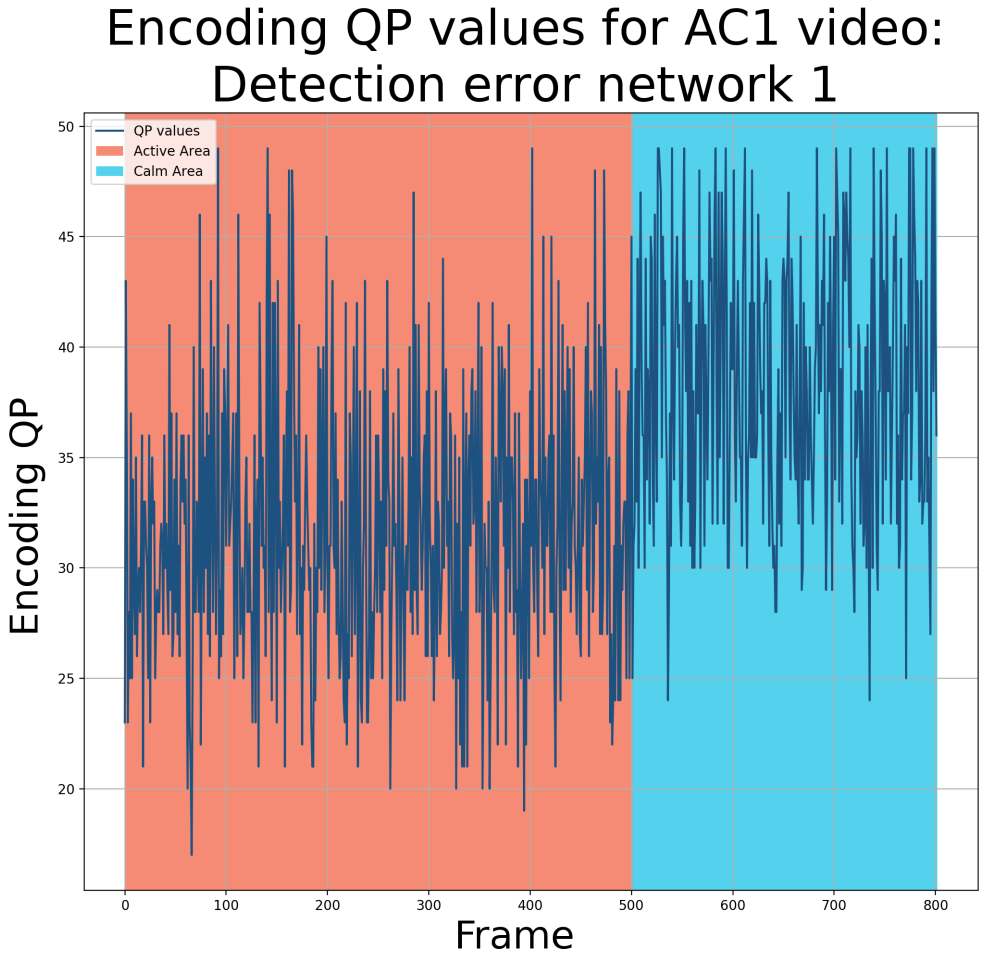


Figure 20: The chosen encoding QP values for the AC1 video (see section 7.7.1) using detection error network 1 (see section 7.1). The red and blue areas represent the active sequence and calm sequence of the encoded video respectively.

Figure 20 demonstrates the encoding QP values chosen by detection error network 1 when encoding video AC1 (section 7.7.1). The mean of encoding QP values is $QP_a = 31.63$ for the active area and $QP_c = 38.38$ for the calm area. It is evident from the plot that QP values are consistently higher for the calm sequence, the mean QP also being significantly higher. This indicates that the network does adapt its output after the cut to the calm sequence so that more compression is applied. Predicted QP values have a lot of local variety, the output oscillates heavily which is sure to have an effect on the overall quality of the video.

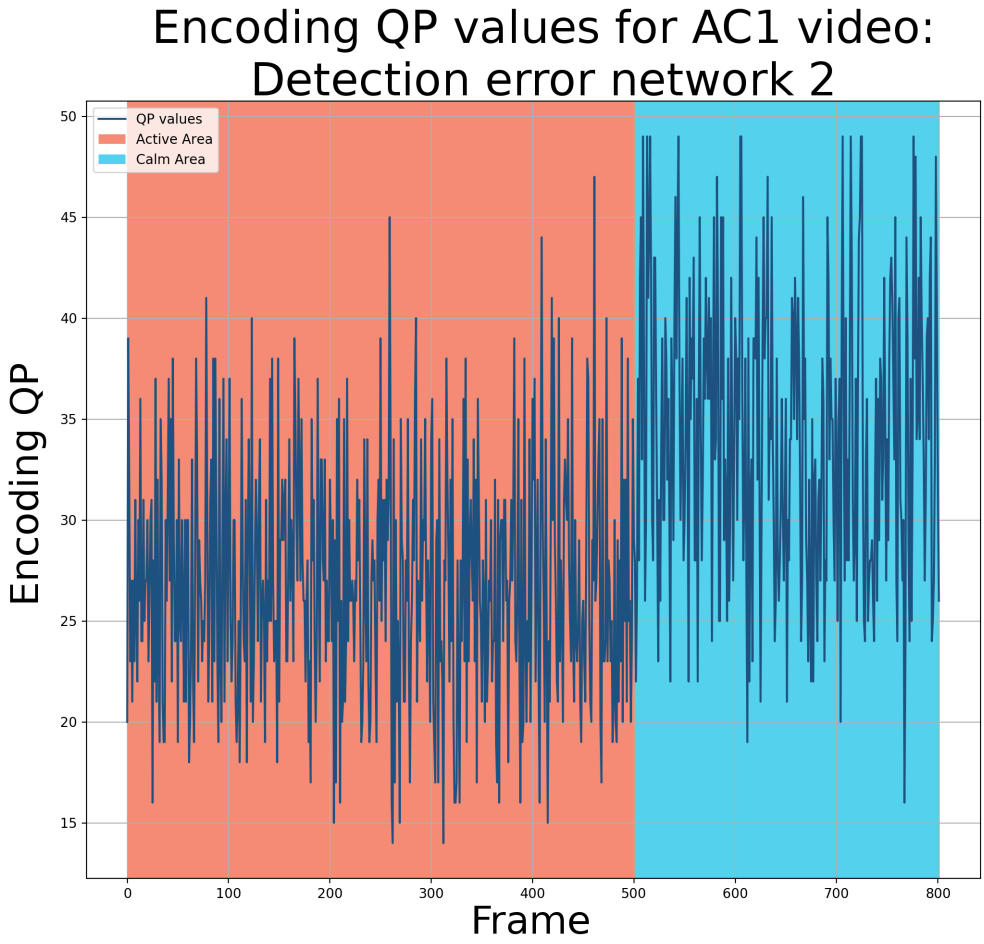


Figure 21: The chosen encoding QP values for the AC1 video (see section 7.7.1) using detection error network 2. The red and blue areas represent the active sequence and calm sequence of the encoded video respectively.

The encoding QP values for AC1 given when using detection error network 2 are seen in figure 21. As stated in section 7.1, the difference between detection error network 1 and 2 is that they have different penalty modifiers ($m = 0.2$ and $m = 0.7$ respectively, refer to section 13). As this network has a higher penalty modifier, it is expected that output QP values be generally smaller (based on the distribution of QP value labels from the quality cost function, see results in section 8.3.2). The mean of encoding QP values is $QP_a = 26.9$ for the active area and $QP_c = 34.17$ for the calm area. The trend of the QP output is quite similar to the results in figure 20, the network adapting its output to increase compression when going into the calm sequence. However, looking at the mean QP values and comparing them to figure 20, it stands clear that the selected QP values become quite a lot smaller when predicting QP values using detection error network 2.

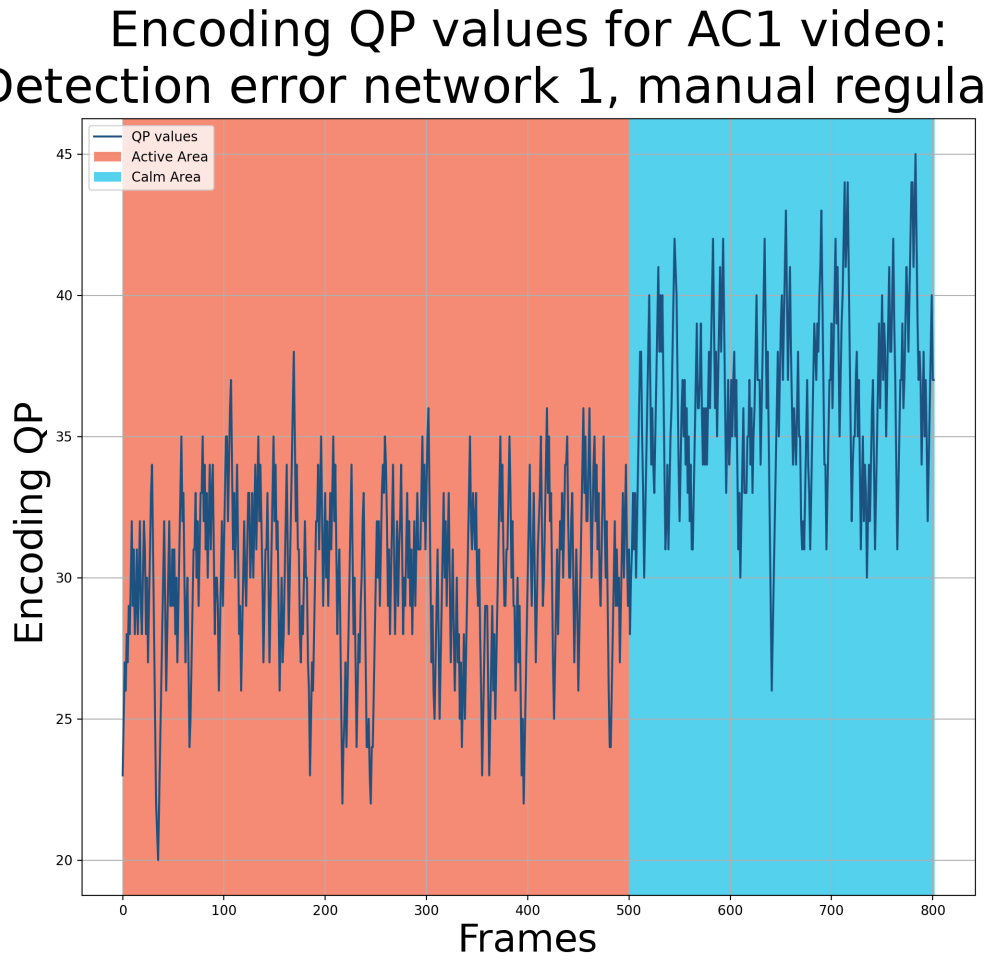


Figure 22: The chosen encoding QP values for the AC1 video (see section 7.7.1) using detection error network 1 (see section 7.1) along with manual regulation in the implemented rate controller. The manual regulation prevented the QP value from increasing by more than 2 and decreasing by more than 3 from frame-to-frame. The red and blue areas represent the active sequence and calm sequence of the encoded video respectively.

When adding manual regulation, as in figure 22, encoding QP values are restricted by how much they are allowed to change from frame-to-frame. The mean of encoding QP values is $QP_a = 30.04$ for the active area and $QP_c = 36.26$ for the calm area. As compared to the corresponding unregulated encoding QP values in figure 20, there is less local variance in the QP value between frames, although output still has a lot of oscillation. The overall trend of increasing compression in the calm area remains the same with manual regulation and is actually more visible with less oscillation.

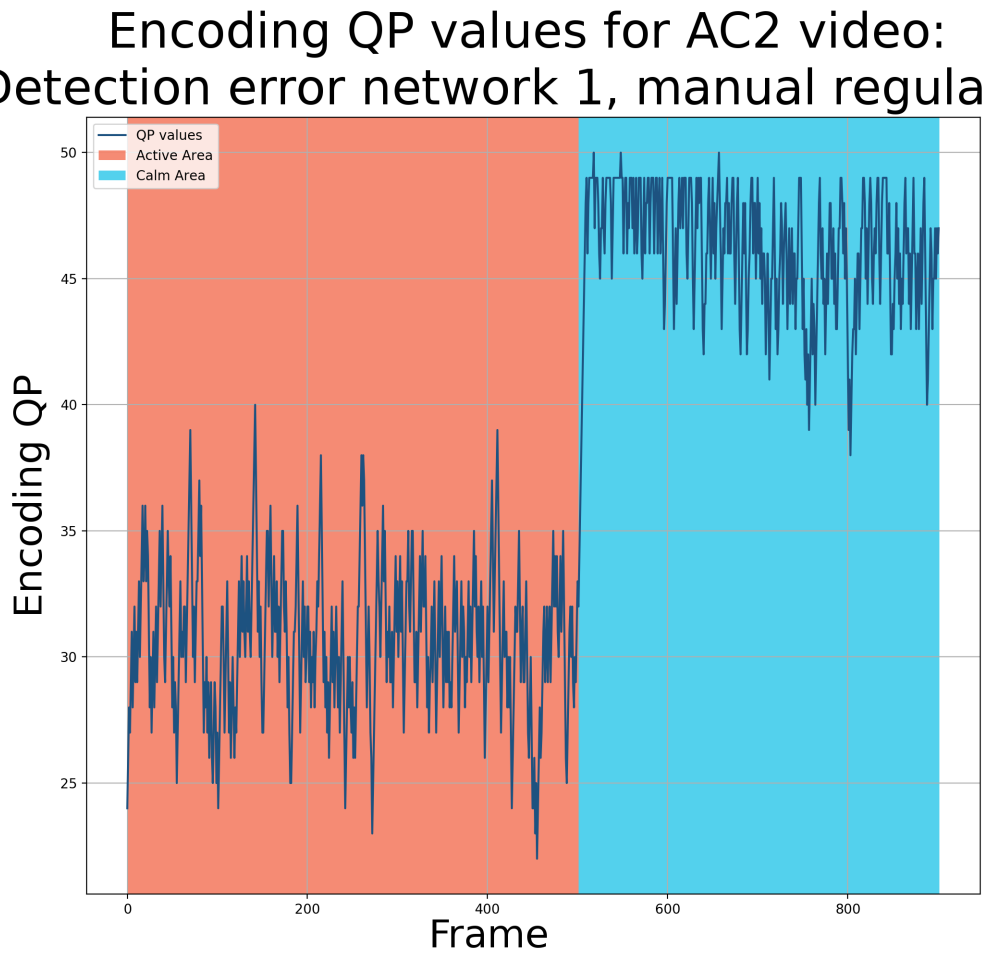


Figure 23: The chosen encoding QP values for the AC2 video (see section 7.7.1) using detection error network 1 (see section 7.1) along with manual regulation in the implemented rate controller. The manual regulation prevented the QP value from increasing by more than 2 and decreasing by more than 3 from frame-to-frame. The red and blue areas represent the active sequence and calm sequence of the encoded video respectively.

Figure 23 demonstrates the encoding QP values chosen by detection error network 1 when encoding video AC2 (section 7.7.1). Manual regulation is applied here as well. The mean of encoding QP values is $QP_a = 30.69$ for the active area and $QP_c = 46.18$ for the calm area. The generalization performance for detection error network 1 is poor as indicated by the residual interval ratio results in table 3. However, the trend regarding QP output observed for video AC1 seen in figures 20-22 seems to have carried over to AC2 regardless, as the video has much higher compression in the calm area. The difference in QP mean value between the active and calm areas is much greater here than what was the case for the AC1 encodings (figures 20-22). Whether this has to do with differences in video content or the performance of the network on test set data remains inconclusive.

8.6 Data Examination

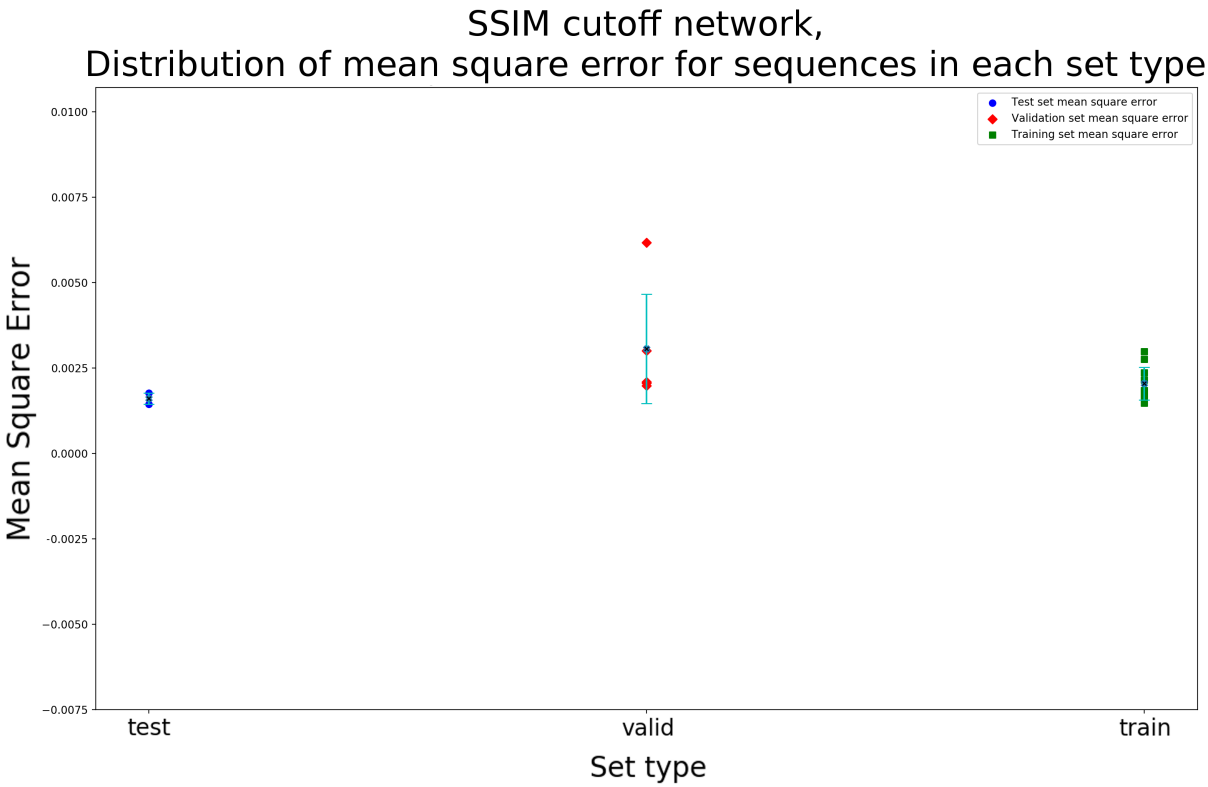


Figure 24: Each marker shows mean square error performance (equation 4) by the SSIM cutoff network (section 7.1) for video sequences assigned to each set type. The mean and standard deviation of the distribution for the mean square errors for each set type is also displayed.

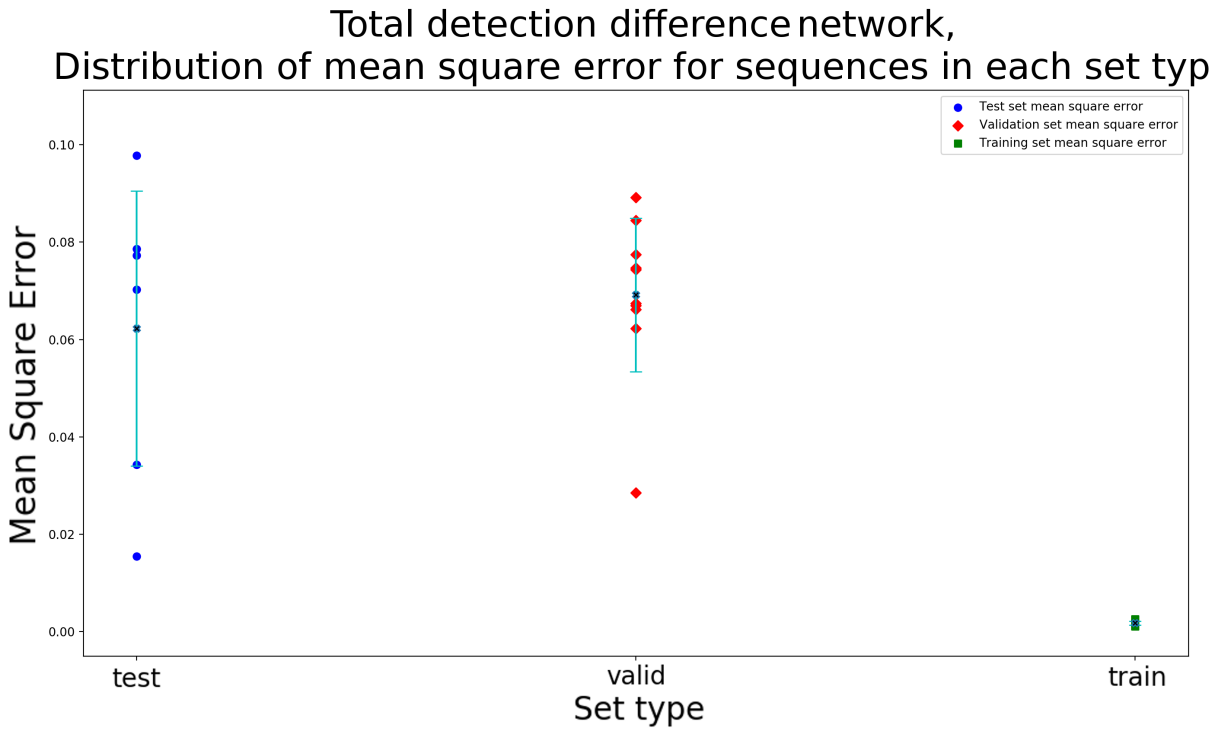


Figure 25: Each marker shows mean square error performance (equation 4) by the total detection difference network (section 7.1) for video sequences assigned to each set type. The mean and standard deviation of the distribution for the mean square errors results of each set type is also displayed.

Figure 26: Each marker shows mean square error performance (equation 4) by detection error network 1 (section 7.1) for video sequences assigned to each set type. The mean and standard deviation of the distribution for the mean square errors results of each set type is also displayed.

Network	Training set	Validation set	Test set
SSIM cutoff network	13925	5471	2470
Other	38637	11306	6921

Table 8: This table shows how many input examples with corresponding labels (see supervised learning, section 5.2.1) are available for each set type in the case of the SSIM cutoff network (section 7.1) and the other networks.

Network	Training set	Validation set	Test set
SSIM cutoff network	10	5	2
Other	30	10	6

Table 9: The number of video sequences used to generate data for the SSIM cutoff network (section 7.1).

For the SSIM cutoff network (section 7.1), the expected result was not met (see section 7.8):

- The mean of the performance results for the validation set is not within the established boundaries of neither the training nor test set.
- The mean of the performance results for the training set is not within the established boundaries of the test set.

Even when extending the boundaries to two standard deviations rather than one, the above results remain the same, so it appears there is some discord between the distributions for each set type. Figure 24 illustrates the mean square error distribution for each set type, rather large differences in variance (indicated by the standard deviance) can be seen for the distributions.

In the case of total detection difference network as well as all detection error networks, it seems there is not the same case for concern. All networks produce the same result:

- The mean of performance results for the test and validation sets are in each other's established boundaries.
- The mean of performance results for the test and validation sets are not in the established boundaries of the training set.
- The mean of performance results for the training set is not in the established boundaries of the validation and test set.

It appears then that test and validation sets have distributions for the performance result that are in accordance with each other. On the other hand, the corresponding distribution of the training set seems to be in discord with the distributions of both of the other set types. Figures 24 illustrate the mean square error distribution for each set type in the case of the total detection difference network and detection error network 1. The same broad observations can be made for both figures:

- Training set performance results are very good as compared to the validation and test sets, with very small variance as well. The figure suggests that overfitting is an issue for both networks, which is corroborated by the corresponding residual interval ratio performance results (tables 2 and 3). All training performance results are very closely clustered together, the validation and test set performance results are both clearly much worse.
- Comparing the validation and test set, the test set distribution has a lower mean (better mean performance) and also a larger variance as indicated by the standard deviation in both figures.

Table 8 illustrates the amount of available data for the SSIM cutoff network as well as the other networks (total detection difference network and the detection error networks). The difference is very large, the total amount of labeled input examples available to the other networks is more than twice as big ($56264/21866 \approx 2.5$). The number of videos (see table 9) they are taken from is almost three times as large ($46/16 \approx 2.7$). As the amount of different video sequences determines the variance in video data it is reasonable to assume this has a rather large impact on the distributions for performance results in each set type.

9 Discussion

9.1 SSIM cutoff training

9.1.1 Training Results

According to the residual interval ratio (section 5.2.2) performance as given by table 1 the SSIM cutoff network (see section 7.1) has acceptable performance with regard to the training, validation and test set (as acceptable performance is defined in section 5.2.2). The fact that almost all predicted values have their residuals in a ± 5 residual interval for training, validation and test sets suggest both very high performance and very good generalization performance as well.

The training and validation loss curves seen in figure 6 clearly indicate that both losses decrease over time.

The residual interval ratio performance, as well as the training and validation loss curves clearly demonstrate that training with video meta data is possible. Test set performance is actually higher than performance for training and validation sets which is strange. This is investigated further in sections 7.8, 8.6 and 9.8.

As SSIM and other general image quality measurements are not interesting for surveillance video (see introduction in section 1), there is no real point in investigating SSIM cutoff as a quality measurement further. However, the results do confirm that training with the video meta data as input data and QP values based on a custom quality measurement as labels can result in a high-performing network.

9.2 Total detection difference training

9.2.1 Training Results

The residual interval ratio performance results seen in table 2 confirms acceptable performance (as defined in section 5.2.2) for the training and test sets with a majority of values being in the ± 5 interval for both cases. The performance for the validation set does not quite reach the acceptable level of performance ($\approx 46\%$ inside the ± 5 interval). The very high performance of the training set as compared to validation and test sets indicate poor generalization performance and overfitting.

The training and validation loss both clearly decrease over time, as seen in figure 7. Validation loss is large compared to training loss throughout, a further indicator of bad generalization performance.

The expected result was acceptable performance for residual interval ratio performance in all set types and a decrease in training and validation loss over time (see section 7.3.1). The latter is well-established for both the training and validation loss based on the loss plot. Acceptable performance is reached for the test set, but the large difference in performance when comparing the test and validation sets' performance to the training set still means that there is probably a large issue with overfitting. The training results are thus mixed for the total detection difference quality measurement.

Further, it seems strange that test set performance is better than validation set performance, an issue which is further investigated in sections 7.8, 8.6 and 9.8.

9.2.2 Analysis

The results in section 8.2.3 (see figures 8-11) seem to indicate that using total detection difference as quality measurement carries with it multiple issues.

The plot in figure 11 shows the number of detected objects for different compressed versions of the sample frame by YOLO label, as well as the total number of detected objects. Many of the compressed frames at lower QP values have a larger total detection difference than when compression is considerably higher (the size of the total detection difference for the highest compression is = 1). According to the total detection quality measurement, the compressed frames with higher compression rate are actually deemed to have a better quality in this case, so high QP values will suffer fewer imposed penalties in the quality cost function (equation 11). As stated in section 7.3, the expected result is a larger total detection difference for the higher QP values. This is not seen in the results.

The sample frame for which YOLO results were studied (figures 8-10) had a total number of seven detection errors (according to the project-specific definition in section 5.7.4 for the highest compression version of the frame. This obviously indicates a rather large difference in YOLO detection performance, which should correspond to a rather large penalty in the quality cost function. The total detection difference quality cost function would only have applied one penalty here (as the size of the total detection difference is one), while using detection error as quality measurement would have resulted in seven. Of

course the size of the individual penalty also matters, but the latter case seems like a more appropriate response to the change in YOLO performance if individual penalties are of the same size.

Consider the box plot in figure 12. The box plots show distributions for total detection difference at different QP values. As mentioned in section 7.3.3, this plot is studied to determine if the size of the total detection difference is greater for high QP values, which is also the expected result. The fact that the median moves away from zero and that the interquartile range and box plot whiskers have larger intervals for higher QP values (see explanation of box plots in section 5), indicates that this is true. It is also evident that at high QP values, there are generally fewer detected objects (as the majority of total detection differences are always > 0).

A similar plot, but where bounding boxes of confidence $< 90\%$ (see section 5.7) have been removed is seen in 13. Here, the median also migrates away from zero-difference and the intervals of the interquartile ranges grow for higher QP values in the box plots. The placement of medians, size of interquartile ranges and box plot whisker intervals as compared to figure 12 clearly indicate that total detection difference is greater for higher QP values after confidence filtering, which is the expected result (section 7.3.3). Compared to figure 12, the trend of higher QP values having fewer detected objects is also enhanced. This result indicates that YOLO confidence filtering can be an efficient strategy to establish clearer trends in the data, which might improve training results.

9.3 Detection error training

9.3.1 Training Results

The residual interval ratio (section 5.2.2) observed for detection error network 1 (section 7.1), in other words when using penalty modifier $m = 0.2$ (section 5.7.7) is seen in table 3. According to section 5.2.2, the network has acceptable performance with regard to residual interval ratio performance for the training and test sets while the performance for the validation set is just below the required limit (≈ 49 % of predicted QP residuals inside the ± 5 interval).

Table 4, for detection error network 2 where $m = 0.7$ is used has a very similar result. The residual interval ratio performance is acceptable for both training and test sets, but does not reach the acceptable performance level for the validation set.

For both detection error networks (section 7.1) the end result of training is mixed. Training set residual interval ratio performance is very high. However, the validation and test sets' performance is clearly worse by comparison, so overfitting seems like an issue here. As the residual interval ratio performances are very similar for the networks, it seems the choice of penalty modifiers m does not change performance by much.

Both network have better performance for the test set than validation set, a strange result which is further investigated in sections 7.8, 8.6 and 9.8.

9.4 Analysis

The distributions of optimal QP (section 5.5.2) values obtained with different values of the penalty modifier (see section 5.7.7) can be seen in figure 14. For the greater penalty modifier m, the peak at QP 50 is heavily reduced and a greater number of QP values in the range $QP < 35$ are chosen. This indicates that when increasing the penalty modifier m, it can be expected that the distribution of optimal QP values as given by equation 13 will shift towards lower QP values, which is expected as well.

Figures 15-18 relates the number of false positives / negatives to the number of detected objects in an uncompressed version of a target frame. The plots show:

- There are more false negatives when the uncompressed frame has more objects detected.
- There are more false positives when the uncompressed frame has less objects detected.
- All detection errors seem to increase with QP value.

These are the expected results (section 7.5) as well. It would be interesting to see how filtering by YOLO confidence (section 5.7) affects these data trends. As can be seen in 9.2.2, applying confidence filtering can result in existing trends becoming more clearly defined, which could have an impact on training results here as well.

The effect of applying smoothing to the false positive / negative data prior to use in the quality cost function (section 5.7.5) can be seen in figure 19. In short, for the selected frame the effect of large spikes

from false positive / negative penalties are greatly reduced. A significantly different QP value is chosen as well (a difference of 10 QP values as compared to the original), which is in line with the expected result. This seems to indicate that smoothing can be an efficient strategy to reduce noise in the detection error data and significantly alter optimal QP value output.

9.5 Improving detection error training

9.5.1 Training Results

Table 5 shows the residual interval ratio performance for smoothing detection error network 1 (section 7.1). Generalization performance is apparently not improved when smoothing is applied, which is not the expected result (section 7.6.1). Further, it actually appears that training set performance is slightly worse (when comparing ratios in the ± 1 interval). Generalization performance in regard to the results in table 3 is not improved, which goes against what is expected (section 7.6.1). There could be many reasons as to why performance has worsened after smoothing is applied, one possibility is that the smoothing itself distorts important features in the data.

Tables 6 and 7 show the dropout detection error network and data augmented detection error network (see section 7.1 residual interval ratio results. The change, as compared to the original detection error network 1 results (table 3) is not large in either case.

In the case of the dropout detection error network, the following is seen:

- Consistently worse training set performance, an expected consequence of adding more regularization.
- Noticeably lower ratios in the ± 1 residual interval range for both validation and test set performance.
- Slightly higher ratios in the other residual intervals for the validation and test sets.

While the change in training set performance is an expected result of added regularization, the test and validation set performance changes are harder to interpret. It seems that overall, the prediction results are slightly better, although it appears to have come at the cost of losing very good prediction results for some subset of the input examples in the validation and test sets. The expected result was an improvement in generalization performance, but it is hard to determine if the observed change in performance is an unambiguous improvement. Further experimentation with other configurations of dropout rates might be necessary to fully evaluate the ability of dropout layers to improve generalization performance.

The results for the data augmented detection error network have changed in similar ways as compared to the original detection error network 1 results:

- Consistently worse training set performance, but a less severe decrease in performance than what was the case for the dropout detection error network.
- Noticeably lower ratios in the ± 1 residual interval range for both validation and test set performance, but a smaller decrease than what was the case for the dropout detection error network.
- Generally slightly higher ratios in the other residual intervals for the validation and test sets. However, as opposed to the case of the dropout detection error network, this is not the case for all the ratios (in some cases ratios have decreased) and the increases are smaller as well.

As in the case of the dropout detection error network, it is difficult to assert whether generalization performance has actually improved. Generally, the changes to generalization performance are less extreme than when dropout layers are added. As suggested in the case of dropout layers, more experimentation may need to be done to fully evaluate the data augmentation approach, specifically with regard to the type of gaussian noise that is added to features. For example it might be best to not add gaussian noise to all feature vectors, but rather some subset. Also, experimentation with different gaussian distributions (equation 7) seems like an appropriate route for future experimentation.

9.6 Video Encoding Test

9.6.1 Video Encoding Results

For the AC1 video (see section 7.7.1), almost all uses of detection error networks in the implemented rate controller show a clear adaptation in the choice of QP values (figures 20, 21 and 22) when moving from the active to the calm sequence of the AC1 video. These results are as expected, indicating that the detection error networks have some real-life applicability.

The local variation from frame-to-frame is large for the encoding QP values. The large fluctuations in QP value choices is clearly seen in figures 20 and 21. A closer, manual inspection of the values shows that the change from frame-to-frame is generally much larger than the desired $\pm 4 - 5$ expected in section 7.7.1. Even with manual regulation (see section 7.7.2) there is a lot of local variance (see figure 22). This indicates that it might be prudent for quality cost functions (see section 5.5.3) to have a regulatory component that prevent this behaviour.

The AC2 video is used to study the robustness (generalization performance) of the rate controller (section 7.7.1). The generalization performance is poor for the detection error networks (tables 3-4) with regard to residual interval ratio performance (section 5.2.2). But looking at the encoding of AC2 with detection error network 1 (the manually regulated version is seen in figure 23), the network clearly has the ability to adapt to the calmer scene in the test set video as well. This indicates that the expected result has been attained, as a similar adaptation to the one in AC1 is seen. At the same time, one can discern some major differences:

- The average encoding QP difference between the active and calm sequence is approximately twice as big as it is for the AC1 video.
- The slope at the beginning of the calm sequence is much more steep than it is for AC1, even with manual regulation.

This indicates that for AC2, the implemented rate controller is inclined to make more extreme, less nuanced choices of encoding QP values. One possibility is that the AC2 video is just sufficiently different from the AC1 video for these compression choices to be motivated, although the videos appear to be very similar in their content. Another possibility is that the severity of the overfitting in the detection error networks prevents more nuanced decisions regarding the choice of encoding QP values for unseen data.

Another point of interest mentioned in section section 7.7.1 is the ability of the network to adapt to local changes. The results here are inconclusive, the unstable output of the network makes it difficult to discern adaptation to smaller changes. Evaluating this ability to make smaller changes is left to future work.

9.7 On-Camera Implementation

One goal in the thesis project was to evaluate the possibility of transferring the implemented rate controller to an on-camera implementation (section 4). However, due to time constraints the possibility of performing benchmarks to evaluate the required performance from a camera to efficiently run the implemented rate controller is limited. Conversations with Axis staff that work with implementing deep learning solutions for cameras yielded some key factors that were important for an implementation:

- **Memory requirements**: The camera must be able to hold the input features in memory while running.
- **Network complexity**: This means that the number of layers and nodes in the network becomes very relevant, as they determine the number of operations for each input.
- **Frequency**: The frequency of operations.
- **Parallel operations**: While running the network by itself on the camera might work, it also needs to be ascertained that other crucial operations can run at the same time.
- **Network implementation choices**: Some choices can be made in future network implementations that might keep down performance requirements. Small measures such as using signed activation functions rather than computationally expensive sigmoidal functions is one such example.

9.8 Data Examination

The results for the total detection difference network, as well as all detection error networks (7.1) were not quite what was expected in section 7.8.2:

- The training set mean for video sequence performance results (mean square error, equation 4) was not inside the boundaries established by the mean and standard deviations of the other set type performance distributions.
- Corresponding means for validation and test sets were not inside the boundaries of the training set.

It has already been established in the residual interval ratio performance results (section 5.2.2), seen in tables 1-6, that training set performance is a lot better than validation and test set performance for these networks. It seems reasonable then that overfitting (section 5.2.3) is responsible for the difference in performance result distributions when comparing the training set to the validation and test set, rather than issues with the set type selection process.

The fact that test set performance is better than validation set for all the residual interval ratio results is still strange. Figures 25 and 26 both show that the test set has a larger variance and lower mean for the distribution of performance results. But simultaneously the results indicate that the performance result distributions are quite alike. Based on these basic results, it seems more likely that the disparity in performance results between validation and test set is more related to statistical variance than a systematic error in choosing training, validation and test sets.

The SSIM cutoff network results are a bit more concerning. In this case, performance as measured by the residual interval ratio result (table 1) was in a similar range for all set types. Despite this, the results in section 8.6 with regard to this network have several instances were the performance result distributions are not in accordance with each other. So overfitting can not be blamed for differences between the training set and the other set types.

Table 9 shows how many videos sequences are used to produce the labeled input examples for each set type:

- The test set only takes data from two videos. Judging by figure 24 is appears that the two videos also have small mean square errors. The low variance of the performance results and the high performance for the included videos is in line with the fact that the test set has the highest performance for the residual interval ratio results.
- The validation set takes data from five videos, The figure indicates a possible outlier value, some video with unusually high mean square error. Going by the size of the associated standard deviation in the figure, this potential outlier has a high impact on the variance of the distribution for performance results. The outlier and high variance of the performance result distribution are expected based on the fact that validation set performance was the worst in table 1.
- The training set has 10 videos to take data from, performance results seem to be in the same range as the validation set when the outlier is not considered.

As mentioned in results section 8.6 the amount of labeled input examples is ≈ 2.5 larger for the other networks and the number of videos they are taken from is ≈ 2.7 times larger. This could be one reason as to why the other networks do not show any discord between performance result distributions for the test and validation set. To be fair, such a comparison is difficult to make, as the labels the network predicts are based on very different concepts. Based on the results for SSIM cutoff network it seems possible that the issue with test performance for the network is related to a mixture of the current selection process and low amounts of available data for this network. In other words, when there are few video sequences to choose from, it seems that the current selection process can cause a set type to have very homogeneous data.

The low number of videos used for each set type may have resulted in disproportionately high variance for the validation set and low variance for the test set. A suggested improvement to the selection process based on these results is to stop assigning entire video sequences to set types. Instead, it is probably best to first produce the labeled input examples to be used from the selected frames in all videos, and to then randomly appoint each input example to a set set type.

10 Conclusion

The purpose of this thesis project has been to be act as an exploratory work regarding the possibility of using a machine learning solution to implement a rate controller. As mentioned in the problem formulation 4), one of the main goals is to identify quality measurements (section 5.5.1) that can be used to determine the best compression rate for a video frame based on some metric. During the course of the project, a number of possible quality measurements have been evaluated.

SSIM cutoff (section 5.6.1), total detection difference (section 5.7.1) and detection error (section 5.7.6) are all proposed quality measurements that have been used to output theoretically optimal QP values (section 5.5.2) for encoding a video. For each such set of QP value labels, an attempt has been made to train using a multilayer perceptron neural network architecture, acting on encoding history and imaging platform data as input data (section 5.2.3). Performance of the trained networks was evaluated using a custom performance measurement referred to as residual interval ratio (section 5.2.2). Performance for optimal QP value produces by SSIM cutoff was very high and had good generalization performance, but as SSIM is deemed to have little relevance to quality in a surveillance video context, this result is not very interesting for future work. Issues with generalization performance were present both when total detection difference and detection error were used as quality measurements (tables 2, 3-4). It was determined that total detection difference did not seem to provide as detailed a quality measurement as detection error (section 9.2.2), the latter measurement appears the most promising for future work. Efforts were made to improve generalization performance for the detection error networks using various strategies (section 7.6.1), results were ambiguous as to whether or not generalization performance improved with the strategies.

Having trained a set of networks, an attempt was made at evaluating their real-life applicability (section 7.7.1). An implementation of a rate controller was created that used trained networks to select encoding QP values for two different videos. One video was constructed from the training set and one from the test set, each was divided into an active and calm sequence (section 7.7.1). It was found that the detection error networks managed to adapt their QP based on the activity in a sequence (figures 20-23), indicating that there is real-life applicability for the implemented rate controller. However, concerns were also raised regarding the ability of the networks to adapt to small changes as well as the large local variance when choosing QP values. Future projects should look into these issues more conscientiously.

Part of the project was dedicated to investigate the potential of the implemented rate controller to be transferred to an on-camera implementation (section 9.7). However, time constraints prevented a thorough investigation of this issue, and so it is left to future work.

An investigation was conducted regarding the generally high performance of all networks on test set data as compared to other set types. It was concluded that a few changes to the current selection process for training, validation and test sets could be beneficial to increase variance within each set type.

11 Future Work

11.1 Training

The majority of efforts in this project have been directed towards evaluating quality measurements (section 5.5.1) and their potential. Less work has been put into developing effective, optimized network architectures and training strategies for each one. As a a result, there are probably major improvements that can be made in this regard, some suggestions follow below.

11.1.1 Network Architecture

The multilayer perceptron network architecture used throughout the project (section 7) was very simple. The time-dependent nature of the problem means there are potential gains in using for example a recurrent network with long-term short memory (LSTM) units.

If a choice is made to use multilayer perceptron architecture going forward, it is still worth investigating whether a self-organizing map (SOM) network could be used as a possible pre-step to the network. A possible gain with using SOM is that input data with very high dimensions could be mapped more efficiently, meaning that more available data could be used as input.

11.1.2 Using a built-in loss function

The use of a quality cost function (section 5.5.3) as a pre-step to training is probably something that should be filtered out with future implementations. Simply optimizing for a predetermined QP with mean square error removes the nuance in the quality cost functions. For this reason, the effect of implementing such functions as loss functions in Tensorflow should be investigated.

11.1.3 Further regularization

The improvements to generalization performance (section 5.2.3) attained with dropout layers (section 5.4.3) show some possibility of improving generalization performance. Trying different configurations of dropout layers and dropout rate could possibly yield an improvement in generalization performance.

11.1.4 Input data

Regarding input data, the following should be attempted:

- Use more time steps for the training inputs, currently only the history of the past 5 frames are used for each input during training (section 7).
- Use more of the imagining platform meta data. Due to time constraints and to some degree insufficient training data, many parameters from the imaging platform were left out. If more data can be obtained, or if more efficient methods of mapping the data (section 11.1.1) are used, using more of the imaging platform data would be more viable.

11.2 Data

11.2.1 Video

In this thesis project the same scene was filmed for all the data. Filming at different times of day and changing angles for the scene can produce some degree of variety, but filming new scenes entirely would have greater impact on the data set. Filming more scenes that are significantly different from the current one should improve the robustness of the network, allowing it to perform well for a larger range of scenes. Access to new scenes would also help demonstrate how well the result for the current trained network generalize to other scenes.

11.3 Quality measurements

11.3.1 Improving detection error quality measurement

The detection error quality measurement (section 5.7.6) seems promising. The networks trained with optimal QP values (section 5.5.2) from the associated quality cost function show an ability to adapt compression rate output to video content. However, to curtail the issue with unstable output (section 9.6.1), further regulation should be added. It seems that the quality cost function could need some type of added penalty that discourages changing the QP value by too much from frame-to-frame.

Using the YOLO network (section 5.7) by itself might also be an issue. Looking at the object detection result, it is obvious that certain objects are not detected by YOLO. By using R-CNN or SSD networks along with YOLO and pooling the object detection results, it is possible that more objects would be detected. Having more detected objects in the original frame would probably result in more nuanced penalties to costs in the quality cost function (equation 13) when detection errors are present.

To make the trained network more sensitive to small changes in video rather than just large ones, it might be a good idea to use motion detection. Axis has already used pixel-oriented motion detection for some camera software, the general consensus being that it provides some usefulness but is very sensitive to noise. If motion detection was instead implemented by tracking bounding boxes (in the uncompressed frame), the result would probably be less sensitive to noise. At the same time the motion detection measurement might also be more binary. It would only indicate whether or not an object of interest (such as a car or person) is on the move, rather than how much movement is currently going on. Even so, such a measurement could be of use. If a set of objects are determined to be moving in the uncompressed frame, not detecting such an object in the compressed frame should probably be associated with a penalty. This could make the trained network more sensitive to instances of for example cars driving into the scene.

11.3.2 Object re-identification

A further improvement on the existing object detection quality measurements could use the concept of object re-identification. The matching algorithm (section 5.7.3) already tries to match objects between the uncompressed frame and the compressed version of the frame by comparing bounding boxes. Finding a matching object in the compressed version of the frame has the requirement that YOLO detects an object there despite the frame quality decreasing. Further, if the detected objects is overlapping with some other object, the current matching algorithm might also erroneously chose the overlapping object as a match.

An improvement on the current matching would be to use template matching. One could cut out the detected objects (or just the interesting detected objects such as people and cars) from the original frame. Using template matching to try and find the cutouts in the compressed frame could then be be used as a way to evaluate quality, by looking at the number of matched objects. Further, if the template matching algorithm is not designed to perform well despite image distortions, the result would indicate both that objects were re-identified in the compressed frame and that they had similar image quality to the original frame for the selected objects.

11.3.3 Human annotation

Using human annotation to some extent might improve the selected QP values. Human annotators would not have to look at every single frame and determine an optimal QP. One possibility would be to choose a subset of optimal QP values (section 5.5.2), and have annotators review the frames where for example:

- The chosen QP value is very low or high.
- The chosen QP value is very different from the previous frame QP.

For "extreme" cases, human annotators could be presented with sample images and give some sort of input about how the QP should be changed.

11.4 Embedded solution

As mentioned previously (section 9.7), there was not enough time to properly investigate the possibility of an on-camera implementation of the rate controller used in this project. It is however of great interest to dig deeper into this issue.

References

Chen, T., Liu, H., Shen, Q., Yue, T., Cao, X. & Ma, Z. (2017), Deepcoder: A deep neural network based video compression, *in* '2017 IEEE Visual Communications and Image Processing (VCIP)', pp. 1–4.

Cybenko, G. (1989), 'Approximation by superpositions of a sigmoidal function', *Mathematics of Control, Signals and Systems* **2**(4), 303–314.
URL: *https://doi.org/10.1007/BF02551274[Accessed 5 Oct. 2018]*

Domingos, P. (2012), 'A few useful things to know about machine learning', *Communications of the ACM* **55**(10), pp. 78–87.
URL: *http://doi.acm.org/10.1145/2347736.2347755 [Accessed 5 Oct. 2018]*

Girshick, R. B., Donahue, J., Darrell, T. & Malik, J. (2013), 'Rich feature hierarchies for accurate object detection and semantic segmentation', *CoRR* **abs/1311.2524**.
URL: *http://arxiv.org/abs/1311.2524[Accessed 5 Oct. 2018]*

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press.
URL: *http://www.deeplearningbook.org [Accessed 5 Oct. 2018]*

Guyon, I. (1996), A scaling law for the validation-set training-set size ratio, Technical report, AT&T Bell Laboratory.

ITU Radiocommunication Sector (2011), Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios, Technical report, International Telecommunications Union.
URL: *https://www.itu.int/rec/R-REC-BT.601 [Accessed 5 Oct. 2018]*

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C. & Berg, A. C. (2015), 'SSD: single shot multibox detector', *CoRR* **abs/1512.02325**.
URL: *http://arxiv.org/abs/1512.02325[Accessed 5 Oct. 2018*

Mandyam, G., Ahmed, N. & Magotra, N. (1997), 'Lossless image compression using the discrete cosine transform', *J. Vis. Comun. Image Represent.* **8**(1), pp. 21–26.
URL: *http://dx.doi.org/10.1006/jvci.1997.0323[Accessed 5 Oct. 2018]*

Mitchell, T. (1997), *Machine Learning*, WCB/McGraw-Hill.

Perez, L. & Wang, J. (2017), 'The effectiveness of data augmentation in image classification using deep learning', *CoRR* **abs/1712.04621**.
URL: *http://arxiv.org/abs/1712.04621[Accessed 5 Oct. 2018]*

Poynton, C. (2003), *Digital video and HDTV: Algorithms and Interfaces*, San Francisco, CA: Morgan Kauffman Publishers.

Redmon, J., Divvala, S. K., Girshick, R. B. & Farhadi, A. (2015), 'You only look once: Unified, real-time object detection', *CoRR* **abs/1506.02640**.
URL: *http://arxiv.org/abs/1506.02640[Accessed 5 Oct. 2018]*

Redmon, J. & Farhadi, A. (2018), 'Yolov3: An incremental improvement', *CoRR* **abs/1804.02767**.
URL: *http://arxiv.org/abs/1804.02767[Accessed 5 Oct. 2018]*

Singh, M. P., Arya, K. V. & Sharma, K. (2009), Video compression using self organizing map and pattern storage using hopfield neural network, *in* '2009 International Conference on Industrial and Information Systems (ICIIS)', pp. 272–278.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: A simple way to prevent neural networks from overfitting', *Journal of Machine Learning Research* **15**, pp. 1929–1958.
URL: *http://jmlr.org/papers/v15/srivastava14a.html[Accessed 5 Oct. 2018]*

Tkalcic, M. & Tasic, J. F. (2003), Colour spaces: perceptual, historical and applicational background, *in* 'The IEEE Region 8 EUROCON 2003. Computer as a Tool.', Vol. 1, pp. 304–308.

Toderici, G., O'Malley, S. M., Hwang, S. J., Vincent, D., Minnen, D., Baluja, S., Covell, M. & Sukthankar, R. (2015), 'Variable rate image compression with recurrent neural networks', *CoRR* **abs/1511.06085**.
URL: *http://arxiv.org/abs/1511.06085[Accessed 5 Oct. 2018]*

Toderici, G., Vincent, D., Johnston, N., Hwang, S. J., Minnen, D., Shor, J. & Covell, M. (2016), 'Full resolution image compression with recurrent neural networks', *CoRR* **abs/1608.05148**.
URL: *http://arxiv.org/abs/1608.05148[Accessed 5 Oct. 2018]*

Wang, Z. & Bovik, A. C. (2009), 'Mean squared error: Love it or leave it? a new look at signal fidelity measures', *IEEE Signal Processing Magazine* **26**(1), pp. 98–117.