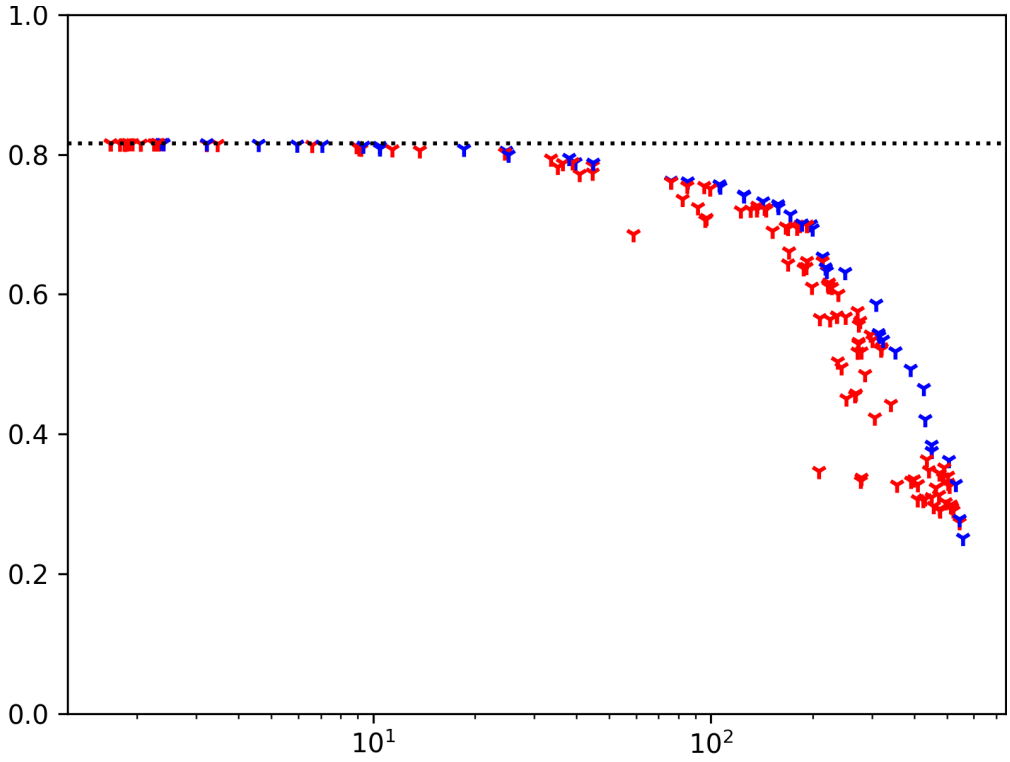


An experimental evaluation of image input degradation on machine learning performance

Master's thesis in Computer science and engineering

Oscar Andersson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

An experimental evaluation of image input degradation on machine learning performance

Oscar Andersson

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Software Engineering division
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

An experimental evaluation of image input degradation on machine learning performance
Oscar Andersson

Supervisor: Christian Berger, Department of Computer Science and Engineering
Examiner: Regina Hebig, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Software Engineering division
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A plot with results of a test optimisation run.

Typeset in LaTeX
Gothenburg, Sweden 2020

An experimental evaluation of image input degradation on machine learning performance

Oscar Andersson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Significant advancements in the field of machine learning have been made during the past decade due to artificial neural networks being feasible to compute. To train these neural networks, a large amount of data is needed, especially in the field of autonomous driving where image data needs to be stored at a large scale. Such data can be reduced in size significantly by using lossy video compression at the cost of losing visual fidelity. This trade-off could potentially be balanced such that the data size of a dataset is reduced while the reduction in data quality is not significantly affected. This thesis aims to establish the effects of video compression on machine learning algorithms performing computer vision tasks, specifically for autonomous driving. This involved evaluating the effect of certain encoders and coding parameters on a set of ML-algorithms. Multi-objective optimisation was performed to find sets of optimum coding parameters for each encoder evaluated, referred to as coding parameter sweet spots. Experiments were conducted to measure the impacts of video compression, using the optimum coding parameters', on machine learning algorithms. The experiment results indicated that some encoders' sweet spots were able to compress the data without significantly altering ML-performance. Collected experiment data was also used to compare encoders capabilities and behaviour when compressing data for ML usage. Finally, suggestions for how practitioners should evaluate and validate lossy video compression methods were provided.

Keywords: Machine learning, video compression, optimisation, codec, encoder, parameters, autonomous driving.

Acknowledgements

I would like to thank Revere labs for providing me with the computer hardware necessary for performing the data collection in time. I would also like to thank my thesis supervisors Christian Berger and Federico Giaimo for aiding me throughout this thesis.

Oscar Andersson, Gothenburg, June 2020

Contents

List of Figures **xi**

List of Tables **xv**

1 Introduction **1**
1.1 Background . . . 1
1.1.1 Video coding formats and video codecs . . . 2
1.1.2 Multi-objective optimisation . . . 2
1.1.3 Video compression for ML-data . . . 3
1.2 Research Purpose . . . 3
1.3 Research Questions and Goals . . . 4
1.4 Contributions . . . 5
1.5 Delimitations . . . 5
1.6 Structure of thesis report . . . 5

2 Related works **7**
2.1 Image quality and machine learning . . . 7
2.2 Multi-objective optimisation . . . 8

3 Methodology **11**
3.1 Selecting ML-algorithms . . . 11
3.1.1 Algorithm choice . . . 12
3.1.2 Cityscapes dataset . . . 13
3.1.3 Time constraints . . . 13
3.2 The multi-objective optimisation . . . 13
3.2.1 Optimisation setup . . . 15
3.2.2 The automated software . . . 16
3.2.3 The multi-objective genetic optimisation algorithm . . . 18
3.3 Experimental evaluation . . . 19
3.3.1 Research question I . . . 21
3.3.2 Research question II . . . 22
3.3.2.1 Data degradation . . . 23

4 Results **25**
4.1 Pareto optimal fronts . . . 25
4.1.1 HRNet . . . 25
4.1.2 GSCNN . . . 28

4.2 Research questions 30
4.3 Visual quality metrics and ML-performance 40

5 Analysis and Discussion **43**
5.1 Optimum coding parameters 43
5.2 Effects of lossy compression 43
5.2.1 Research question 1 44
5.2.2 Research question 2 45
5.3 Visual quality metrics and ML-performance 46
5.4 Coding parameters' sweet spots 47
5.4.1 NVIDIA accelerated encoders 48
5.4.2 Intel accelerated encoders 48
5.4.3 libx264 48
5.4.4 Final word on coding parameters 48
5.5 Encoding speed 49
5.6 Recommendations and best practices 49
5.7 Threats to validity 50

6 Conclusion **53**
6.1 Future Works 54

Bibliography **55**

A Appendix A **I**
A.1 Boxplots of collected situation data I
A.2 Shapiro-Wilk test results X
A.3 Encoding times XII

B Appendix B **XIII**
B.1 Coding parameters optimised XIII
B.2 Examples of degraded images XVII
B.3 Examples of video compression artefacts XX
B.4 Optimisation population runs XX
B.5 Sweetspot evaluation data XXVI
B.6 GSCNN's reduced validation set XXXVIII

List of Figures

1.1 Illustration of a Pareto frontier. 3

3.1 Graphic representation of the optimisation process. 18
3.2 An illustration of data collection for situations' baseline performance. 21
3.3 An illustration of data collection for sweet spot candidates. 21

4.1 Pareto frontiers for Hardware encoded H.264. 26
4.2 Pareto frontiers for Hardware encoded HEVC. 26
4.3 Pareto frontiers for Software encoders. 27
4.4 Pareto frontier for Hardware encoded VP9. 27
4.5 Pareto frontiers for Hardware encoded H.264. 28
4.6 Pareto frontiers for Hardware encoded HEVC. 28
4.7 Pareto frontiers for Software encoders. 29
4.8 Pareto frontier for Hardware encoded VP9. 29
4.9 Plots of T-scores for the HRNet sweet spot candidates. 31
4.10 Plots of T-scores for the GSCNN sweet spot candidates. 31
4.11 Sweet spot candidates' HRNet ML-performance mean and standard deviation of errors on the non degraded dataset, illustrated using plots. 32
4.12 Sweet spot candidates' GSCNN ML-performance mean and standard deviation of errors on the non degraded dataset, illustrated using plots. 32
4.13 Plots of T-scores for the HRNet sweet spot candidates compressing the rain-degraded dataset. 33
4.14 Plots of T-scores for the GSCNN sweet spot candidates compressing the rain-degraded dataset. 34
4.15 Plots of T-scores for the HRNet sweet spot candidates compressing the movement-degraded dataset. 34
4.16 Plots of T-scores for the GSCNN sweet spot candidates compressing the movement-degraded dataset. 35
4.17 Plots of T-scores for the HRNet sweet spot candidates compressing the noise-degraded dataset. 35
4.18 Plots of T-scores for the GSCNN sweet spot candidates compressing the noise-degraded dataset. 36
4.19 Sweet spot candidates' HRNet ML-performance mean and standard deviation of errors on the rainy dataset, illustrated using plots. 36
4.20 Sweet spot candidates' GSCNN ML-performance mean and standard deviation of errors on the rainy dataset, illustrated using plots. 37

4.21 Sweet spot candidates' HRNet ML-performance mean and standard deviation of errors on the added movement dataset, illustrated using plots. 37
4.22 Sweet spot candidates' GSCNN ML-performance mean and standard deviation of errors on the added movement dataset, illustrated using plots. 38
4.23 Sweet spot candidates' HRNet ML-performance mean and standard deviation of errors on the noisy dataset, illustrated using plots. 38
4.24 Sweet spot candidates' GSCNN ML-performance mean and standard deviation of errors on the noisy dataset, illustrated using plots. 39
4.25 Plots of mean ML-performance and PSNR error for the sweet spot candidates compressing the non degraded dataset. 40
4.26 Plots of mean ML-performance and PSNR error for the sweet spot candidates compressing the rainy dataset. 40
4.27 Plots of mean ML-performance and PSNR error for the sweet spot candidates compressing the dataset with simulated movement. 41
4.28 Plots of mean ML-performance and PSNR error for the sweet spot candidates compressing the noisy dataset. 41

A.1 Sweetspot candidates' HRNet ML-performance errors on the non degraded dataset, illustrated using boxplots. II
A.2 Sweetspot candidates' GSCNN ML-performance errors on the non degraded dataset, illustrated using boxplots. III
A.3 Sweetspot candidates' HRNet ML-performance errors on the rainy dataset, illustrated using boxplots. IV
A.4 Sweetspot candidates' GSCNN ML-performance errors on the rainy dataset, illustrated using boxplots. V
A.5 Sweetspot candidates' HRNet ML-performance errors on the movement dataset, illustrated using boxplots. VI
A.6 Sweetspot candidates' GSCNN ML-performance errors on the movement dataset, illustrated using boxplots. VII
A.7 Sweetspot candidates' HRNet ML-performance errors on the noisy dataset, illustrated using boxplots. VIII
A.8 Sweetspot candidates' GSCNN ML-performance errors on the noisy dataset, illustrated using boxplots. IX
A.9 Plots of mean ML-performance and the p-value from the Shapiro-Wilk test for the sweetspot candidates compressing the non degraded dataset. X
A.10 Plots of mean ML-performance and the p-value from the Shapiro-Wilk test for the sweetspot candidates compressing the rainy dataset. X
A.11 Plots of mean ML-performance and the p-value from the Shapiro-Wilk test for the sweetspot candidates compressing the dataset with simulated movement. XI
A.12 Plots of mean ML-performance and the p-value from the Shapiro-Wilk test for the sweetspot candidates compressing the noisy dataset. XI
A.13 A boxplot of the encoding times for each encoder. XII

B.1 Image examples of noisy dataset XVII
B.2 Image examples of rainy dataset XVIII
B.3 Image examples of movement dataset XIX
B.4 Cropped image examples of video compression artefacts using lib264. XX
B.5 Optimality of coding parameters of libx264 using on HRNet and a limited dataset XXI
B.6 Optimality of coding parameters of h264_nvenc using on HRNet and a limited dataset XXII
B.7 Optimality of coding parameters of hevc_nvenc using on HRNet and a limited dataset XXIII
B.8 Optimality of coding parameters of h264_vaapi using on HRNet and a limited dataset XXIV
B.9 Optimality of coding parameters of hevc_vaapi using on HRNet and a limited dataset XXV

List of Tables

3.1 A summary of the chosen ML-algorithms, repository and benchmark statistics (fetched 2020-06-24). 13
3.2 The encoders used in the optimisation. 14
3.3 The hardware specification of the Revere computer running the automated software. 15
3.4 The hardware specification of the Home computer running the automated software. 15
3.5 The software running on the Revere computer. 15
3.6 The software running on the Home computer. 16
3.7 The software running in the optimisation docker container. 16

5.1 A summary of the evaluated encoders. 50

B.1 The coding parameters used in the `h264_vaapi` optimisation. XIII
B.2 The coding parameters used in the `hevc_vaapi` optimisation. XIII
B.3 The coding parameters used in the `VP9_vaapi` optimisation. XIV
B.4 The coding parameters used in the `libx264` optimisation. XV
B.5 The coding parameters used in the `h264_nvenc` optimisation. XVI
B.6 The coding parameters used in the `hevc_nvenc` optimisation. XVI
B.7 The coding parameters used in the `libsvt_av1` optimisation. XVI
B.8 The coding parameters of each sweet spot candidate of hevc_nvenc for HRNet. XXVI
B.9 Aggregated evaluation data for each sweet spot candidate of hevc_nvenc for HRNet. XXVII
B.10 The coding parameters of each sweet spot candidate of hevc_nvenc for GSCNN. XXVII
B.11 Aggregated evaluation data for each sweet spot candidate of hevc_nvenc for GSCNN. XXVIII
B.12 The coding parameters of each sweet spot candidate of h264_nvenc for HRNet. XXVIII
B.13 Aggregated evaluation data for each sweet spot candidate of h264_nvenc for HRNet. XXVIII
B.14 The coding parameters of each sweet spot candidate of h264_nvenc for GSCNN. XXIX
B.15 Aggregated evaluation data for each sweet spot candidate of h264_nvenc for GSCNN. XXIX

B.16 The coding parameters of each sweet spot candidate of h264_vaapi for HRNet. . . . XXIX
B.17 Aggregated evaluation data for each sweet spot candidate of h264_vaapi for HRNet. . . . XXX
B.18 The coding parameters of each sweet spot candidate of h264_vaapi for GSCNN. . . . XXX
B.19 Aggregated evaluation data for each sweet spot candidate of h264_vaapi for GSCNN. . . . XXX
B.20 The coding parameters of each sweet spot candidate of hevc_vaapi for HRNet. . . . XXXI
B.21 Aggregated evaluation data for each sweet spot candidate of hevc_vaapi for HRNet. . . . XXXI
B.22 The coding parameters of each sweet spot candidate of hevc_vaapi for GSCNN. . . . XXXI
B.23 Aggregated evaluation data for each sweet spot candidate of hevc_vaapi for GSCNN. . . . XXXII
B.24 The coding parameters of each sweet spot candidate of vp9_vaapi for HRNet. . . . XXXII
B.25 Aggregated evaluation data for each sweet spot candidate of vp9_vaapi for HRNet. . . . XXXIII
B.26 The coding parameters of each sweet spot candidate of vp9_vaapi for GSCNN. . . . XXXIII
B.27 Aggregated evaluation data for each sweet spot candidate of vp9_vaapi for GSCNN. . . . XXXIV
B.28 The coding parameters of each sweet spot candidate of libx264 for HRNet. . . . XXXV
B.29 Aggregated evaluation data for each sweet spot candidate of libx264 for HRNet. . . . XXXVI
B.30 The coding parameters of each sweet spot candidate of libx264 for GSCNN. . . . XXXVII
B.31 Aggregated evaluation data for each sweet spot candidate of libx264 for GSCNN. . . . XXXVII

1
Introduction

The purpose, goal, contribution and delimitations of the conducted research are introduced in this section. In order to provide the reader with ample context for this thesis, this section also introduces key concepts and provides background information regarding why the research was done.

1.1 Background

Significant advancements have been made in the area of Machine Learning (ML) during the past decade. The idea of using Artificial Neural Networks (ANN) for ML have existed since the 40s, but the computational power to use them was not available until recently. Modern hardware can rapidly calculate the matrix operations required for ANNs, making complex ANN-based ML possible. Deep learning is a term referring to ML methods that use ANNs with several hidden layers. For the network to output the desired information, connections between the layers need to be tuned correctly. In the training process, data is put into the network and the network's output is compared to its desired output. A process called back-propagation will during training make corrections to the networks connections. With enough data to train on, the network's output will, in theory, start to learn the connection between the input and the desired output. The deeper and more complex the networks get, the more training data is required.
The automotive industry is currently investing significant amounts of resources in autonomous driving [1]. A great deal of data needs to be collected to properly train the deep neural networks that are developed for autonomous driving [2]. The data rate of the collected data needs to be reasonable if vast amounts of recording sessions are to be stored and used for training. The quality of that data will be reflected in the neural networks that train on them. To reach high levels of performance in autonomous driving tasks, high-quality training data is required. If thousands of hours of video recordings are to be collected and used for training ANNs for autonomous driving tasks, the data rate must be manageable for large scale use, while not compromising the data quality [3].
A common method of reducing the data rate of video content is lossy video compression. Lossy compression is an irreversible compression method that improves compression ratios at the cost of discarding information [4]. Video compression algorithms can reduce video file sizes significantly, but too much compression often results in bad video quality [5][6]. If video recordings are too heavily compressed and lose information, then there is a risk that it will result in degraded ML-performance.

1.1.1 Video coding formats and video codecs

Video content can be digitally represented by using video coding formats that dictate how video files and streams should be encoded and decoded. The process of encoding and decoding video content to and from the video coding format is performed by video encoders and decoders. These video encoders and decoders can implement the coding format by using either software algorithms processed using CPUs or specific hardware made for accelerating encoding and decoding to and from the coding format. Software and hardware that both encode and decode are called codecs (short for encoder-decoder). In this thesis, the term encoder will be used for both the encoders of codecs and the standalone encoders.
Video coding formats reduce the storage footprint of video content by using video compression, which can be either lossless or lossy [4]. With lossless video compression, the video content data can be fully reconstructed hence not suffering from any loss of image quality. However, video content encoded by lossless video compression can have a data footprint that is considerably larger than what is feasible for video streaming and storage. Lossy video compression can significantly reduce the data footprint at the cost of discarding data. Lossy video compression can be achieved by using intra-frame compression and inter-frame compression. Intra-frame compression compresses the information on a per-frame basis. For many coding formats, intra-frame compression involves partitioning each frame into blocks, transforming the pixels within these blocks into discrete cosine transforms, and using a technique called quantisation to reduce the resolution of frequency components that matter less for perceived image quality. Inter-frame compression can reduce the bitrate by finding similar information between frames and use motion compensated prediction to reconstruct parts of one frame from a reference frame [7]. Encoders that perform lossy compression can often be configured to target a specific bitrate, change the level of features to use, change encoding speed preset and specify certain optimisation methods. The choice of video coding format, video encoder and coding parameters affect:

- the quality of the video content,
- how computationally heavy it is to encode the video content,
- the data size of the encoded video content,
- the profile support required to decode the content

1.1.2 Multi-objective optimisation

Finding the optimum coding-parameters for a lossy encoder is not a trivial task. The compressed videos have several conflicting properties, which can be tuned by adjusting coding-parameters. Optimisation involves the search and selection of the best solutions with regards to an optimisation objective. When multiple objectives for the optimisation are involved, the solution that optimises one objective might be poor with regards to the other objective. Multi-objective optimisation with conflicting objectives do not have one optimal solution, but multiple solutions that balance the priority between the objectives differently. Among the solutions of a multi-objective optimisation run, there will be solutions that are not dominated by other solutions. These non-dominated solutions are Pareto optimal, which means that

there are no other solutions that further improve an objective without diminishing any of the other objectives. An illustration of dominated and non-dominated solutions are shown in Figure 1.1, where blue points are Pareto optimal (non-dominated) and red points are not Pareto optimal (dominated).

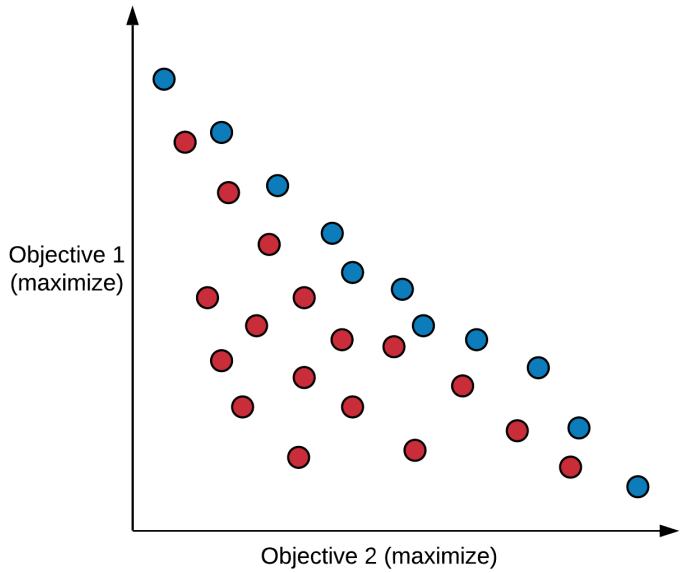


Figure 1.1: Illustration of a Pareto frontier.

The Pareto optimal solutions form a set called a Pareto frontier. The points along this frontier each represent the best possible solutions at given trade-offs between the objectives. An optimal solution of the frontier can be selected depending on what compromise one wants to make between the objectives.

1.1.3 Video compression for ML-data

There has been research about how different encoders and their coding parameters change video quality with regards to human perception of image quality [8][9]. However, as explained in Section 2, there seems to be a lack of studies about how various compression methods affect the actual performance of ML-algorithms. This issue prompted Revere labs (Resource for Vehicle Research at Chalmers) [10] to support this thesis' research, which aims to address the aforementioned issue.

1.2 Research Purpose

Inattentive use of lossy video compression for ML-data could result in losing information that cannot be recovered. ML-practitioners might not compress videos using lossy compression due to the aforementioned risk. By using a lossless or even a raw (uncompressed) format, the videos can reproduce the exact frames that the cameras

were capturing at the cost of producing larger files. For large scale implementation, such large file sizes could be very cumbersome for the automotive industry.
The purpose of this study is to establish how certain video compression encoders and coding parameters affect ML-performance. The research aims to provide the automotive industry with more in-depth information about how various compression methods affect the performance of ML-algorithm training and inference using compressed video input.

1.3 Research Questions and Goals

The goal of the thesis was to identify the optimum methods of video compression for ML-algorithms performing autonomous driving tasks and establish the effects that the compression methods inflict on these ML-algorithms. Two main research questions need to be answered to achieve this goal.

Research question I

Are there "sweet spots" for each encoder's coding parameters with respect to maximising the compression ratio while not impacting the ML-algorithms' performance?
With respect to human perception of quality, there is in most cases a point from which further increases to the bitrate only results in diminishing returns of quality improvement [5][6][11]. This leaves the question, does this phenomenon apply to ML-algorithms? With optimum coding parameters, will the rendered videos preserve a level of quality that has a negligible impact on ML-algorithms? If this is the case, which are the sets of coding parameters that allows for the lowest possible bitrate while preserving ML-performance?
Multi-objective optimisation was used to search for candidate sets of coding parameters over a wide range of trade-offs between ML-performance and compression ratio. The candidate sets found during the optimisation were then tested using experiments to verify their properties. The term "sweet spot" was used for sets of coding parameters that were found to be optimal for compressing.

Research question II

To which degree are machine learning algorithms affected by degrading the visual quality of video input data?
With lossy compression, one can achieve higher compression ratios for videos at the cost of introducing compression artefacts that degrade the visual quality[4]. Furthermore, image noise is known to influence ML-algorithms[12]. To which degree do these quality disturbances affect ML-algorithms?
The input degradation tests on ML-algorithms consisted of encoders configured with sweet spot sets of coding parameters (found during the optimisation) and various levels of noise. Through a series of experimental tests, the combined effects of video compression and noise on ML-algorithms were evaluated.

1.4 Contributions

Knowledge about how encoders and their settings generally affect ML-performance would help researchers and practitioners in the industry to make sustainable implementations of video compression. Researchers and practitioners can use encoder sweet spots and insights from this study as a base from which they adjust and validate the coding parameters to suit their needs and requirements. The conducted research closes the gap in the domain knowledge regarding video compression for machine learning.

1.5 Delimitations

Including too many ML-algorithms would take too much time. The ML-algorithms that were used to evaluate compression impact had to each be optimised for. Hence, the number of experimental units had to be reasonable to limit the time taken to perform the optimisation. Only a subset of the commonly used encoders was used during the experiment to limit the amount of time taken during the optimisation phase. The encoders used for the experiment were chosen based on the basis that they were commonly used, had encoding support on the hardware available for the experiment and had encoding times within reasonable bounds.
Testing one encoder configuration took a considerable amount of time. A full factorial experiment for finding the sweet spots was out of question due to the large number of factors and levels. A multi-objective optimisation was used instead to tune the coding parameters, which used considerably fewer tests than what a full factorial experiment would.
The automation of optimising coding parameters removed considerable amounts of manual labour and improved reliability by limiting the factor of human error or interference. The data collection for hypothesis testing was, to an extent, also automated due to the aforementioned reasons.

1.6 Structure of thesis report

The thesis report starts with an introduction that familiarises the reader to the research purpose, goal, contributions, scope and relevant theory. The related works section introduces the reader to relevant research in the field and in what ways this thesis adds to the body of research. The methodology section details the implementation of automated software that performed the multi-objective optimisation, which ML-algorithms to use for evaluation, the design of hypothesis tests and the data analysis for both optimisation and hypothesis test results. The results section presents data gathered from the optimisation and hypothesis tests that are relevant for answering the research questions. The results from the experiment are examined in the analysis and discussion section. Answers to the research questions were derived from these results and observations regarding the research goal are discussed. The final section of the report concludes the conducted research and proposes future work related to this thesis.

2
Related works

Research into related works was conducted to gain insight into what previous research has been done with regards to video compression, image quality effects on ML and multi-objective optimisation. The gathering of related works was conducted mainly by searching on using Google Scholar and IEEE Xplore using relevant keywords and also using research papers' references to find other papers of interest. Developer blogs for video services like Netflix [8] and Youtube [13] were consulted for insight into common video compression practices. The research paper of the study, which this thesis extends upon [14], was also consulted.

2.1 Image quality and machine learning

The quality of input data is an important factor that affects the performance of ML-algorithms. For ML-algorithms solving computer vision tasks, the input quality can be affected by image degradation like compression artefacts and noise.
In 2016 Dodge and Karam conducted a study [15] that explores how image quality affects deep neural networks. They tested how compression, blur, noise and contrast applied to images used as input affected the accuracy of four neural networks. They found that they could apply significant compression on the input while retaining good prediction accuracy. The results of their experiment show that for the selected neural networks, performance stayed quite consistent until the compression level becomes too intense (at JPEG level 8) and the prediction accuracy rapidly degraded.
Another study from 2018, *DeepN-JPEG* [16], acknowledged that image compression algorithms made for human perception, such as JPEG, are not an optimal solution for DNN systems with regards to compression ratios. With the use of DNNs, the authors developed an image compression framework that achieved approximately 3.5 times higher compression ratio than JPEG for the same level of accuracy for image recognition.
However, image compression only applies to intra (per frame) compression. Representing frame sequences as video content and using inter-frame compression can reduce redundant information among frames giving a more effective compression [7].
The influence of JPEG and H.264 compression on a pedestrian detection algorithm using night vision video input was investigated by Hintermaier et al. [17]. The researchers concluded that lossy compression standards like JPEG and H.264 could be suitable for computer vision applications and provide sufficient performance.
Andaló et al. proposed a framework for generating compressed videos that only contain relevant information for specific computer vision tasks [18]. The framework

composes video frames that contain only relevant information and compress them using video compression. They evaluated their framework's ability to compress UHD-videos used for facial feature detection using HEVC. They found that the framework was able to significantly compress the video while maintaining prediction accuracy.
An approach for adaptive video coding made for systems performing computer vision tasks was detailed by Galteri et al. in a 2018 study [19]. The authors measured the performance impact that traditional HEVC and their proposed method had on an object detection algorithm. They tested the compression methods' effect on object detection with varying bit-rates. Although their approach distinguished itself as the better performer, the standard HEVC was able to maintain sufficient performance to a certain degree.
While these and other similar studies [20][21] bring insight into how standard video compression methods perform with regards to computer vision, they do not provide an overview of how established encoders, with optimised coding parameters, compare to each other with regards to computer vision. Such knowledge is well established when it comes to human image quality perception [22][23][5][8][9].

2.2 Multi-objective optimisation

There are different types of approaches for the optimisation process that a multi-objective optimisation algorithm can take. Priori methods require preference information prior to the optimisation, while posteriori methods aim to generate the entire Pareto frontier. For this thesis, posteriori methods were explored due to the need to analyse encoders' performance over a wide range of trade-offs between ML-performance and compression ratio.
Holland presented a method of multi-objective optimisation based on the principle of natural selection and evolution [24]. The MOGA (Multi-Objective Genetic Algorithm) starts with an initial population of potential solutions that forms the first generation. Through an iterative process, the population will gradually evolve by mixing the best solutions from the previous generation into a new generation using genetic operations. The appeal of MOGAs for this thesis is their ability to converge to a Pareto optimal set as a whole, providing optimal solutions with varying tradeoffs[25].
Al-Abri, Li, Edirisinghe and Grecos proposed an optimisation framework for determining the optimum parameters for video encoders [26]. They used the NSGA-II MOGA and for optimisation objectives used encoding complexity, encoding memory usage, video bitrate and video distortion. However, each objective used by the MOGA was estimated by its objective function. The objective function used a linear regression model made from results of a profiling experiment. In a subsequent paper, Al-Barwani and Edirisinghe proposed a framework using an ML approach for determining significant coding parameters for HEVC video encoders [27], modelling encoder performance using a regression model and optimising coding parameters. They again used NSGA-II and objective functions based on regression models. Although these articles do not present any general knowledge on compression video data for ML-algorithm use, they show that MOGAs has been applied for optimising

video encoder parameters that claimed success.
Laurin and Eberlén [14] conducted a study in which they evaluated video compression algorithms' ability to deliver real-time video streams over UDP. Their study has both research question and methods that have some resemblance to the ones of this study. One of their research questions was: which compression algorithm provides the best video quality during certain conditions while conforming to given processing constraints?
To find the optimal choice of parameters to use for each encoder and situation, they chose to treat the search for optimal parameters as an optimisation problem. A Bayesian optimisation approach was used to find the parameters that maximise the video quality for each encoder and situation. They quantified the video quality by using the video quality assessment metric SSIM. The metric estimates the perceived image quality by comparing the structural similarity between original frames with compressed ones.
The goal of both studies was to find appropriate video compression algorithms to use for ML-algorithms solving autonomous driving tasks with regards to certain objectives. But significant differences between Laurin and Eberlén's study and this study were that:

- this study does not have real-time encoding constraints
- the optimisation problem of this study is a multi-objective optimisation problem due to compression ratio being a factor to maximise
- the way to quantify video quality: While Laurin and Eberlén use a metric that estimates human perception, this study uses evaluation results extracted from ML-algorithms solving autonomous driving tasks

Jourdan and Weck [28] applied MOGA to optimise wireless sensor network layouts. The optimisation objectives were total sensor coverage and network lifetime. With MOGA they obtained two different types of layouts, one having sensors packed together and the other organised sensors in a hub-and-spoke manner. The study shows how the optimal structure of the network changes when different objective trade-offs are made. While the research goals for Jourdan and Weck's study and this study are quite different, the expectations of this study were also to find characteristics of optimal solutions associated with certain objective trade-offs.

Summary of related works

A great deal of research has been published about video compression's effect with regards to human perception. While there is research regarding the effects of video compression on ML-algorithms performing computer vision, there is a lack of research that evaluates and compares various established video compression methods' ability to compress data for computer vision tasks. Optimal solutions for human and computer vision can differ when it comes to image and video compression. Due to this disconnect, one shall not assume that certain video compression methods that excel for human vision work well for computer vision.
When choosing video compression methods for machine learning applications, practitioners should base their choice on research about compression for computer vision, not quality metrics based on human perception. This study extends the current

knowledge of input compression by conducting an experiment observing the effects of modern methods of video compression on ML-algorithms. Statistical tests were performed to establish compression methods' ability to persistently preserve ML-performance, which is especially crucial in the field of autonomous driving. The research method used to a resembles that of previous studies regarding encoder optimisation with MOGA but optimises for actual ML-performance.

3
Methodology

The research methods used to answer the research questions had an experimental approach. To find sets of optimum coding parameters for each encoder, an extensive optimisation process was performed to find sweet spot candidates. A full factorial experiment would have vast amounts of treatments and require too much computational resources to perform. The process of finding the optimal configurations was instead seen as an optimisation problem, which was a more feasible approach given the time frame and resources available. The optimisation problem was finding the optimal configurations for encoding ML-algorithm input with regards to input quality and compression ratio.
To answer hypothesises regarding both research questions, experiments were designed to evaluate the optimal configurations' ability to compress datasets for ML, measure how ML-algorithms were affected by the compressions and establish which sets of encoding configurations did or did not significantly affect ML-performance.
In Section 3.1 the choices of machine learning algorithms with their respective datasets are presented and motivated. Details of the MOGA optimisation are presented in Section 3.2. In Section 3.2.1 the automated software, which conducts the optimisation, is outlined. The procedure for testing hypothesises and analysing data is presented in Section 3.3.

3.1 Selecting ML-algorithms

There are several types of tasks that ML-algorithms can perform and visual artefacts might affect ML-algorithms differently depending on which task they perform. Finding a good selection of ML-algorithms that are state-of-the-art and are relevant for AD ML-tasks was important for the generalisability of the study.
The search started on the software repository hosting site GitHub [29]. When exploring repositories and research about some autonomous driving task, many times there would be a benchmark that researchers measure their ML-algorithms against. These leaderboards were useful for finding the current state of the art ML-algorithms. The ML-algorithms' authors often wrote a research paper about their implementation and referred to another state of the art research of relevance. By going through leaderboards and checking which implementations were referred to often, a collection of ML-algorithm candidates were found. When selecting which repositories to use, a set of criteria were used. The ML-algorithms repository should:

- have algorithms that are **made to perform an autonomous driving task** like lane detection or object identification.

- **not be stagnant**, the ML-algorithm should have been developed on for the past year.
- be able to **work with on one or several datasets** that are available for academic usage. The dataset cannot be protected by a licence that does not allow free non-commercial use.
- be a **turnkey solution**, the ML-algorithms chosen should not take up considerable time to set up.
- not have an ML-algorithm or dataset that **require manual labour** for any pre-processing or labelling.
- **have scientific reports** about the implementation. Information in a report can be decisive when deciding if an algorithm is appropriate to use.
- have a **high amount of stars and forks** and/or being referred to often by other researchers.

After the choice of ML-algorithms was made, each algorithm was set up and verified to reproduce acclaimed results by algorithm creators.

3.1.1 Algorithm choice

The HRNet implementation [30] to solve the Cityscapes' Pixel-Level Semantic Labeling Tasks [31] was one of the highest performing ML-algorithm for performing that task. At the time of writing (2020-06-24) the algorithm held 10'th place of 219 entries on the Cityscapes leaderboard when using the standard Jaccard Index [32]. The implementation's repository was also popular with over 700 stars, 170 forks, was under an MIT-license and had a scientific paper associated with it. The implementation was one of the best-performing state of the art solutions for performing semantic segmentation, which was why it was selected as one of the ML-algorithms used. Another implementation that solves the same aforementioned task is GSCNN (Gated Shape CNNs for Semantic Segmentation) [33]. As with HRNet, the GSCNN algorithm scored well on the Cityscapes leaderboard, ranking 26'th of 219 (2020-06-24). GSCNN was also a popular repository with 608 stars and 128 forks and had a non-commercial use licence.

Both algorithms return the Intersection-Over-Union(IOU) measure when evaluating, which is a commonly used performance metric for semantic segmentation [34]. IOU provides a score of how well predicted bounding boxes match the ground truth bounding boxes. It is simply the area of overlap divided by the area of union. In practice, IOU penalizes misalignment of box-positions and also incorrect box sizes.

Both algorithms use pre-trained models provided by the ML-algorithm creators. The particular model selected for HRNet, `HRNetV2-W48 + OCR` [35], used just the training slice of Cityscapes during training and validation. The only model provided for GSCNN [36] was trained and validated with the full dataset. The data used for coding parameter optimisation uses the Cityscapes validation set, which means that HRNet performs inference on data it has not seen before. In contrast, GSCNN performs inference on data it has used for validation. Performing interference on familiar data is not something undesirable but is something that needs to be kept in mind when analysing the results.

3.1.2 Cityscapes dataset

The Cityscapes dataset [31] is an extensive dataset containing large amounts of labelled data that can be accessed by researchers. The dataset contained labelled frames that were taken seconds apart from each other. The thesis author was granted access to a 330GB dataset containing 19 previous and 10 trailing frames for each of these labelled frames. Having 30 frame sequences was necessary for evaluating the inter-frame compression capabilities of encoders.

3.1.3 Time constraints

Due to a longer inference time of GSCNN, its validation set was reduced for the coding parameter optimisation. The full validation set consists of 500 instances. The reduced validation set is a subset of the full validation set and consists of 177 instances. Usage of a reduced validation set for GSCNN significantly shortened the optimisation time and made the GSCNN data collection feasible. The 177 instances consist of 59 images randomly sampled from each of the three subsets in the validation set(Lindau, Frankfurt, Munster). The 177 instances are listed in Appendix B.6.

	Dataset	**Task**	**Model version**	**Val-set**	**Stars**	**Bench rank**
HRNet	Cityscapes	Pixel-Level Semantic Labeling	best_cityscapes_-checkpoint.pth (2019-09-23)	Full set	1185	10/219
GSCNN	Cityscapes	Pixel-Level Semantic Labeling	hrnet_ocr_cs_-8162_torch11.pth (2019-12-30)	Reduced set	608	26/219

Table 3.1: A summary of the chosen ML-algorithms, repository and benchmark statistics (fetched 2020-06-24).

3.2 The multi-objective optimisation

The problem was approached as a Pareto (multi-objective) optimisation problem with two objectives: maximising the compression ratio and maximising the performance of ML-algorithms. The two objectives were suspected to be conflicting objectives where there are no solutions that provide both an optimum compression ratio and an optimum ML-performance. The solutions of the optimisation problem at hand are called Pareto optimal (non-dominated) solutions. In this case, a solution is Pareto optimal when neither of the two objectives can be further maximised while preserving the optimality of the other objective.
A set of hardware encoders were chosen to represent hardware-based encoding. Both NVIDIA's NVENC and Intel's Quick Sync hardware are broadly available and can in many cases be utilised by ML practitioners. The hardware available for this thesis' data collection had support for the aforementioned encoders, hence they were used

to represent hardware-accelerated encoders. The coding formats that were evaluated are listed in Table 3.2. H.264 is a popular coding format that is commonly used for high definition video content and is supported by a wide range of devices [37]. H.264 was expected to perform worse than its more advanced successor HEVC (H.265) but was included as a baseline coding format used as a reference. VP9 is a well-used coding format, developed by Google, that performs well in comparison to H.264 and HEVC [38][22]. The software encoders used to implement H.264 was from x264, which is a popular software-based codec. NVIDIA and Intel's encoders for H.264, HEVC and VP9 were included to represent the hardware-accelerated encoders [39][40]. The encoding parameters, which were tuned during optimisation, are displayed in Tables under Section B.1. The preset and, in the case of Intel accelerated encoders, the compression level parameters adjust sets of encoding parameters and often regulates the trade-off between encoding speed and quality. Additional coding parameters, which overrides the default and preset parameters, enables further tuning of the encoder. The encoders x265 and libvpx, for HEVC and VP9 respectively, were considered to be evaluated but due to time constraints were not included in the experiments.

At the time when this thesis was conducted, AV1 was prominent new and open video coding format that may become a popular choice for coding video content. Encoders and decoders for AV1 were still under development and were not at as a mature stage as the ones for H.264, HEVC and VP9. The reference implementation for AV1, libaom [41], was evaluated to represent the AV1 coding format, but its encoding time was deemed too great to use in this study. Instead, the SVT-AV1 encoder [42] was chosen to represent AV1 due to its much shorter encoding time at a given achieved quality. The SVT-AV1 encoder still had a very lengthy encoding time, hence no sweet spot search was performed for it. The sweet spot candidates who represented AV1 was set up to use Constant Quantization Parameter (CQP) rate control and encoding preset 4, which was the preset with the best performance and had reasonable encoding times. Due to the encoding time, only five sweet spot candidates were created, which had CQP values spread out between 5 and 35.

Coding format	**Encoder**	**Acceleration type**
h.264	libx264	Software
h.264	h264_nvenc	NVIDIA NVENC
h.264	h264_vaapi	Intel® Quick Sync
HEVC	hevc_nvenc	NVIDIA NVENC
HEVC	hevc_vaapi	Intel® Quick Sync
VP9	vp9_vaapi	Intel® Quick Sync
AV1[1]	libsvt-av1	Software

Table 3.2: The encoders used in the optimisation.

[1] As explained in Section 3.2, AV1 was not optimised for but was instead represented by hand-picked coding parameter sets.

3.2.1 Optimisation setup

The two computer systems, which hardware specifications are detailed in Table 3.3 and 3.4, was used to perform the optimisation. The operating system, drivers and software applications installed on the computers, detailed in tables 3.5 and 3.6, stayed consistent during the optimisation. The two computers output equivalent compressed data for NVIDIA accelerated encoders due to the graphics cards having the `Turing NVENC engine`.

Component type	**Model**	**Specification**
CPU	Intel i9-9900	Stock
GPU	NVIDIA RTX 2080Ti	Stock
RAM	Corsair Vengeance 32GB	2666 MHz CL16
Motherboard	Gigabyte Z390 UD	
Storage	Seagate Firecuda	2TB

Table 3.3: The hardware specification of the Revere computer running the automated software.

Component type	**Model**	**Specification**
CPU	Ryzen 1700	Stock
GPU	NVIDIA GTX 1060Ti	Stock
RAM	Corsair Vengeance 16GB	2666 MHz CL16
Motherboard	Asus B350-PLUS	
Storage	Seagate Barracuda	1TB

Table 3.4: The hardware specification of the Home computer running the automated software.

Software	**Type of software**	**Version**
Arch Linux	Operating System	20.1 (5.5.1-arch1-1)
NVIDIA driver	GPU Driver	440.48.02
Intel iHD driver	GPU Driver	19.4.0
Intel i965 driver	GPU Driver	2.4.0
VA-API	API for video acceleration	1.6 (libva 2.6.0)
Docker	Virtualisation software	19.03.5-ce
Docker-compose	Docker tool: multi-container management	1.25.4
NVIDIA container runtime	GPU aware container runtime	3.1.4-1

Table 3.5: The software running on the Revere computer.

Software	Type of software	Version
Ubuntu	Operating System	18.04
NVIDIA driver	GPU Driver	440.48.02
Docker	Virtualisation software	19.03.5-ce
Docker-compose	Docker tool: multi-container management	1.25.4
NVIDIA container runtime	GPU aware container runtime	3.1.4-1

Table 3.6: The software running on the Home computer.

The different ML-algorithms used for evaluation each ran inside a docker container built for them to ensure replicability and consistency regardless of host system. The entire process of selecting, preparing and evaluating sets of coding parameters was automated by software written in `Python`, created by the thesis author, which also ran inside a dedicated docker container (see Table 3.7).

Software	Type of software	Version
Python3	Python run-time environment	3.7.5
Pip3	Python package manager	19.3.1
ffmpeg	Library of video codecs	4.2
numpy	Python-library	1.18.1
pygmo	Python-library	2.14
matplotlib	Python-library	3.1.3
ffmpeg-python	Python-library	0.2.0
requests	Python-library	2.22.0
scikit-image	Python-library	0.16.2
opencv-python	Python-library	4.2.0.34

Table 3.7: The software running in the optimisation docker container.

3.2.2 The automated software

The goal of the automated software was to implement an optimisation algorithm and automate the process of systematically optimising the coding parameters for each encoder. Automating as much as possible minimised manual intervention, which could be error-prone and time inefficient. The optimisation algorithm performed fewer treatments than what a full factorial test would, which was important given the time it takes to prepare and apply each treatment. The automated software was able to work round-the-clock for days, running one optimisation for each combination of encoder and ML-algorithm.

The `PyGMO`-library [43] was used to construct the optimisation problem and the optimisation algorithm. The library provides a problem class and an algorithm class each requiring a User Defined Problem (UDP) and a User Defined Algorithm (UDA) respectively. The encoder optimisation problem was defined by a UDP class

written by the thesis author that details properties of the problem and the function for calculating decision vectors' fitness. The UDA used was `NSGA-II`, [44] which is a predefined UDA provided by `PyGMO`. `NSGA-II` is a multi-objective genetic algorithm that uses a genetic approach for finding the optimal set of coding parameters.

The optimisation algorithm was configurable with parameters for the number of generations to evolve and the level of cross over and mutation. These parameters were tuned to customise the algorithm for the best optimisation results. The set of values for each encoding parameter to evaluate was loaded from an encoder specific configuration file.

The optimisation algorithm needs an initial generation of decision vectors, which are complete sets of coding parameters, from which to start the optimisation. The initial decision vectors are initiated with datarate parameters (bitrate, CQP or CFR) uniformly spread between bounds set in the encoder specific configuration file. All other settings were randomly assigned values with bounds set in the parameter file. With the optimisation problem defined and the initial generation generated, the `NSGA-II` algorithm is able to perform the optimisation.

The optimisation algorithm calls the fitness function each time a set of coding parameters needs to be evaluated. During a call to the fitness function, the fitness function utilises utilities that prepares a dataset compressed using the coding parameters, requests an evaluation from the ML-algorithm and store fitness data for post optimisation analysis. These utilities were separated into modules depending on what type of service they provide.

The utility module `ffmpeg_utils` contains functions necessary for transcoding and preparing datasets. The module uses the `FFMPEG`-collection[45] of encoders and decoders along with the `FFMPEG-python` wrapper for applying the compression treatments. During a call from the fitness function, the utility's transcode function interprets decision vectors, transcodes the dataset of images into an encoded video and then transcodes the video to lossless-images readable by the ML-algorithms. The transcode function also returns the size of the encoded video, which is used to calculate the compression ratio for the decision vector.

The `rest_communication` module contains functions for RESTful-communication to the Docker containers hosting the ML-algorithms. The fitness function uses these functions to request ML-algorithms to evaluate transcoded datasets. The ML-algorithms, in turn, responds with the performance results achieved. The type of performance results can vary between ML-algorithms depending on what type of task is performed.

Relevant data were continuously stored throughout the optimisation process for post-optimisation analysis. The data of every decision vector evaluated along with its fitness and encoding time were stored in a CSV-file. When the optimisation process was complete, all vectors were tested for Pareto optimality. The non-dominated vectors contain the optimal coding parameters and were used for experimental evaluation described in Section 3.3.

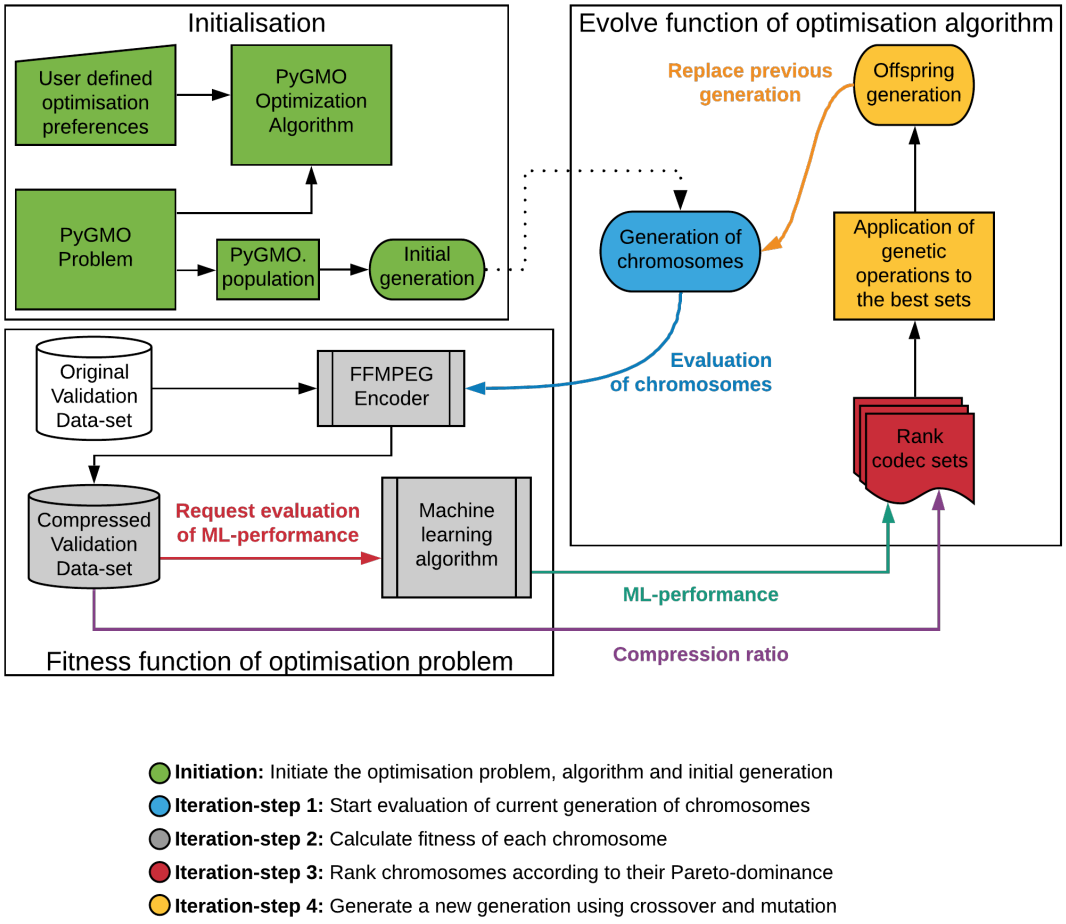


Figure 3.1: Graphic representation of the optimisation process.

3.2.3 The multi-objective genetic optimisation algorithm

The entire optimisation process for testing one encoder on one ML-algorithm is illustrated by Figure 3.1. The initial step of the process is to create the `PyGMO`-problem, algorithm and initial generation using the user-defined problem and algorithm classes. When the initiation is completed, the iterative genetic process begins:

Step 1 and 2: All decision vectors of the current generation are fed to an encoder using the compression utility, which creates compressed validation datasets. Every decision vector will have one corresponding compressed validation dataset. The sets of compressed validation data are fed to an ML-algorithm and their performance are measured.

Step 3: The `PyGMO`-algorithm will then rank the decision vectors based on Pareto optimality, where the objectives are the aforementioned ML-performance and compression ratio of each set.

Step 4: The algorithm creates a new generation of decision vectors by combining the best decision vectors using cross over. Random mutations to these sets are also applied. The new generation also includes clones of the best vectors from the previous generation. The end of the iteration has been reached and the new generation from the last paragraph replaces the old generation and the iterative process repeats.

The fitness of every evaluated set of encoding parameters was saved in a dictionary. When the optimisation is complete, the fitness dictionary is used to extract the Pareto optimal set of every evaluated set. The ML-algorithms are both deterministic, which enabled further use of the fitness dictionary for memoization, a technique of preventing redundant calculations. Before the data collection, a series of tests were made to determine the best parameters for `NSGA-II`. During these tests, a small portion of Cityscapes was used to optimise the coding parameters for each encoder. From all encoders except libx264, the population size was determined to 16 and the number of generations to 10 for the coding parameter optimisation. For libx264, the population size was determined to 20 and the number of generations to 8. These parameters were found to be the most optimal with regards to the limited amount of fitness calls. The results of these tests can be found in Appendix B.4. `NSGA-II` was also configured to use a crossover probability of 90% and a mutation probability of 5% after brief testing using different combinations of crossover and mutation probability.

3.3 Experimental evaluation

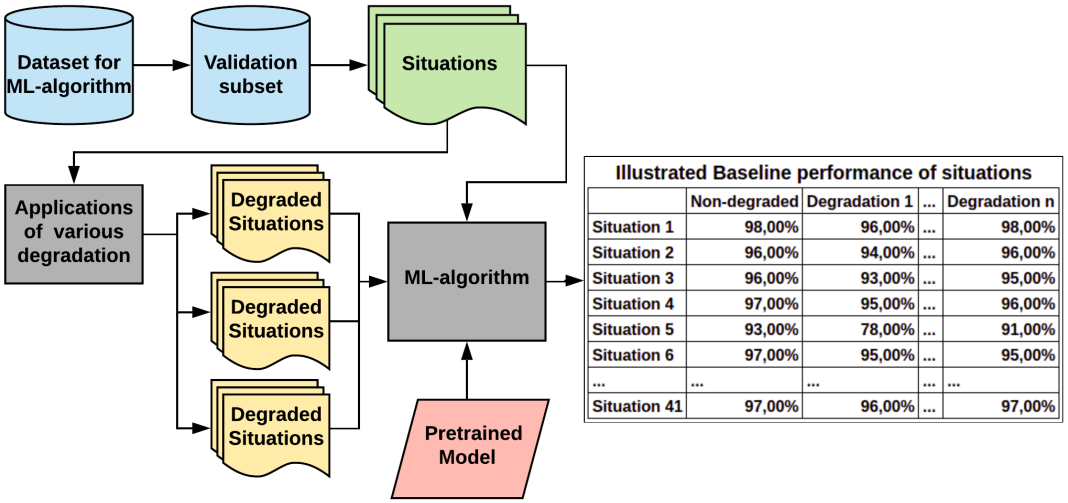
For each research question, a hypothesis was established. For each hypothesis, there are motivations for how the hypothesis is connected to the research questions, how it should be tested and the methodology to run the tests. The tests, along with other data analysis, were computed using Python-scripts written by the thesis author. The scripts uses the `statistics`, `scipy` and `math` modules to perform calculations necessary for each test. The degrees of freedom and critical values were manually extracted and used as constants in the scripts.

Optimisation results: The results from the optimisation process were to contain one or several sets of coding parameters that were Pareto optimal. The Pareto optimal sets contained a wide range of configurations that balanced the trade-offs between compression ratio and ML-performance differently. Although they all were optimum coding parameter sets, many traded away too much ML-performance to be considered as usable sweet spots. Hence, to avoid unnecessary data collection, only the configurations that could achieve an ML-performance of over 80% of the baseline were considered as being sweet spot candidates for hypothesis testing. It was assumed that at least one candidate sweet spot would be found during the optimisation. If there were to be no valid sweet spots, then it would have been a noteworthy discovery to report.

Hypothesis testing: The two-tailed hypothesises were tested using dependent T-tests along with Shapiro-Wilk tests. The T-tests are used to establish if the means between two related groups are significantly different. The Shapiro-Wilk tests are used to establish if the paired errors between two groups are normally distributed. Due to the normal distribution of errors assumption of the dependent T-test, the Shapiro-Wilk test was used first to see if that assumption hold. The chosen alpha level for the Shapiro-Wilk test and the two-tailed dependent T-test was 0.05. For the two-tailed dependent T-test, T-critical was 2.0, based on the 0.05 alpha level and the test's degrees of freedom. Two samples were compared during each test, an uncompressed sample and a compressed sample. Each uncompressed sample contained ML-performance results for one ML-algorithm using uncompressed input data. Each compressed sample contained ML-performance results for the same ML-algorithm but when using compressed input data. Both hypothesises involved the testing of samples from each ML-algorithm. Hence, several sets of samples were collected for each hypothesis, where each sample set was a collection of performance results for each ML-algorithm.

Situations: The dataset used by the ML-algorithms can be split up depending on situations. In this thesis, the tests compare paired groups of performance results, where each group consist of performance results of a set of situations. A situation is defined as one recording snippet and differs by location (city), driving conditions (traffic, speed of recording vehicle), weather and time of day. For every hypothesis, the encoders' ability to compress and preserve video quality were tested for a set of situations from the validation subset. 41 situations were randomly selected from the validation subset, where each situation was 10 consecutive label frames and along with their trailing and heading frames. The random selection created situations in which none of the label frames appears in other situations.

Data collection: The hypothesis testing for both research questions involved data collection of the ML-performance for each sweet spot candidate found during optimisation. The ML-performance was measured for each situation to get the baseline performance readings for the non-degraded and degraded versions of the set of situations (Illustration in Figure 3.2).
Data collection for the two hypothesises was done by applying the sweet spot candidates' compression to the situations, of non-degraded and degraded sets of situations, and measuring the ML-performance in each case (Illustration in Figure 3.3).
In addition to testing the hypothesises, all collected data were studied and discussed regarding certain trade-off characteristics of encoders, coding speed relative to achieved ML-performance and compression ratio, coding support and implementability of encoders.



Illustrated Baseline performance of situations

	Non-degraded	Degradation 1	...	Degradation n
Situation 1	98,00%	96,00%	...	98,00%
Situation 2	96,00%	94,00%	...	96,00%
Situation 3	96,00%	93,00%	...	95,00%
Situation 4	97,00%	95,00%	...	96,00%
Situation 5	93,00%	78,00%	...	91,00%
Situation 6	97,00%	95,00%	...	95,00%
...	...	...	...	...
Situation 41	97,00%	96,00%	...	97,00%

Figure 3.2: An illustration of data collection for situations' baseline performance.

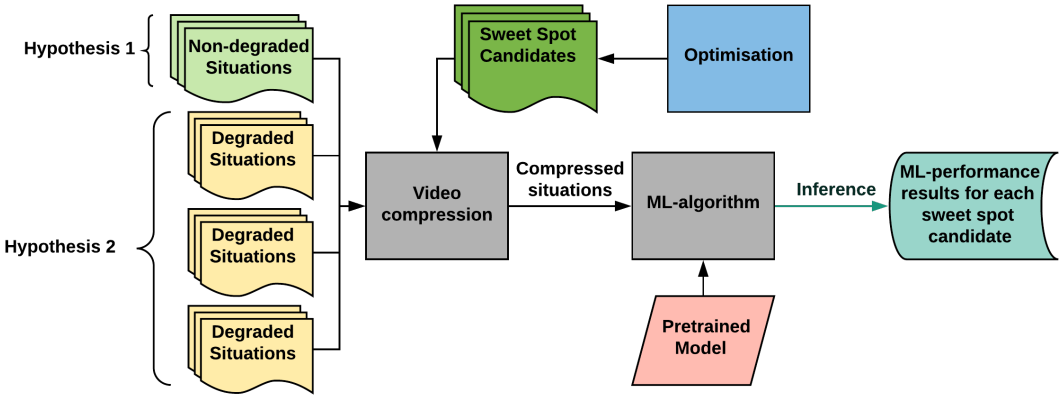


Figure 3.3: An illustration of data collection for sweet spot candidates.

3.3.1 Research question I

Answering: *Are there "sweet spots" for each encoder's coding parameters with respect to maximising the compression ratio while not impacting the ML-algorithms' performance?*

Hypothesis *1*: For each ML-algorithm there will be at least one configuration for each encoder that can create compressed datasets that achieves an ML-performance level that is not significantly different from that of the uncompressed dataset counterparts. If hypothesis *1* held, each tested lossy video encoders could potentially be configured to compress datasets without deteriorating them in a significant way.

Hypothesis *1* was tested using a dependent T-test and a Shapiro-Wilk test for each encoder's candidate sweet spots. For each test, one sample was the ML-performance baseline from the unmodified situations and the other sample was a set of ML-performance results for each situation where the situation was compressed using

the sweet spot configuration. The Shapiro-Wilk test was used to establish if the performance errors results between an uncompressed dataset and a sweet spot were normally distributed, which is an assumption for the dependent T-test. The dependent T-test was performed to determine if there is a significant difference between the mean ML-performance of the uncompressed dataset and the dataset compressed using a sweet spot. **Null hypothesis to test for each case:** There are no significant differences between the samples' mean. **Alternative hypothesis:** There is a significant difference between the mean of the samples.
In addition to hypothesis *1*, the mean and variance of the ML-performance were studied to establish differences between encoders. The compression ratio, ML-performance and encoding time were compared between encoders of both the same coding format and other coding formats. Differences found between encoders are presented in Section 4 and examined in Section 5.

3.3.2 Research question II

Answering: *To which degree are machine learning algorithms affected by degrading the visual quality of video input data?*
Hypothesis *2*: For each ML-algorithm, the compressed datasets of any encoder configuration and uncompressed datasets will at some point differ significantly as they both encounter degraded data. Proving hypothesis 2 could mean that the data rate needs to be further increased to ensure that an encoder can cope with certain types of degraded data.
Hypothesis *2* was tested using a dependent T-test and a Shapiro-Wilk test for each encoder's candidate sweet spots. For each test, one sample was the set of ML-performance for every situation where a fixed level of data degradation has been applied and the other sample was the same except for additionally applied compression using a sweet spot coding configuration. Tests were performed for each sweet spot, where each test evaluated the impact of a specific type of data degradation. The Shapiro-Wilk test was used to establish if the performance errors results between an uncompressed dataset and a sweet spot were normally distributed. The dependent T-test was used to determine if encoder configurations can handle the data degradation without resulting in significant differences between the mean ML-performance of the uncompressed dataset and the dataset compressed using a sweet spot. **Null hypothesis to test for each case:** There are no significant differences between the samples' mean. **Alternative hypothesis:** There is a significant difference between the samples' mean.
In addition to hypothesis *2*, the results from testing hypothesis *2* was studied to establish differences between encoders ability to handle degraded data. A comparison was made to see if there are differences to which data degradation levels encoders can keep their ML-performance similar to the uncompressed ML-performance at a given compression ratio. Furthermore, the results were studied to establish whether there are situations where a certain encoder should not be considered for real-world use due to inconsistent performance. Differences found between encoders is be presented in Section 4 and examined in Section 5.
To establish the extent to which lossy compression can reduce file sizes without

impacting performance, the test results from both hypothesis *1* and *2* were compared. With and without applied data degradation, encoders and coding parameters ability to reduce each datasets' size, while maintaining ML-performance on each ML-algorithm, was tested. These results were used to assess how useful lossy video compression could be for downsizing datasets.

3.3.2.1 Data degradation

Three degraded datasets were created for hypothesis *2*. The first dataset was degraded using Gaussian noise, representing fine-grained visual noise. The second dataset had a rain effect added to it, which represented visual artefacts with less granular disturbances. The third dataset had simulated movement added to each situation of the dataset, which may challenge the encoders' motion compensation and vector prediction capabilities. The Gaussian noise added to the first dataset was applied using `scikit-image`'s random noise function for each frame using a variance of 0.005. The rain effect degradation was achieved by using a version of the data augmentation library `Automold` [46] that was modified by the thesis author to being able to generate the degraded dataset. The dataset with added movement was generated using Python scripts that utilises functions of `scikit-image`'s transform module. The script applied a panning effect where the view of each 30 frame sequence was moving in a randomised direction and speed. The 20th frame, the labelled frame used for evaluation, is kept original with the 19 previous and 10 following frames gradually panning away from the original view. Empty space, which appears when the view is panned outside the bound of the frame, is filled in using the reflect mode of `transform.warp`. Examples of these images can be found in Appendix B.2.

4
Results

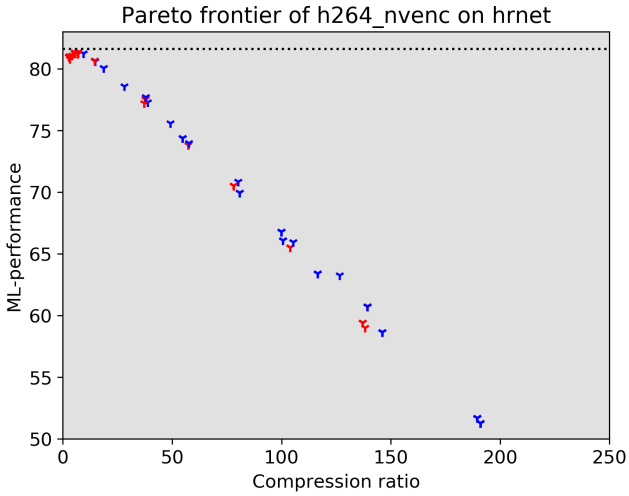
This section presents the data collected from the experiment and the test results. The Pareto optimal front achieved during optimisation is presented in Section 4.1 and the test results regarding the first and second research question are presented in Section 4.2.

4.1 Pareto optimal fronts

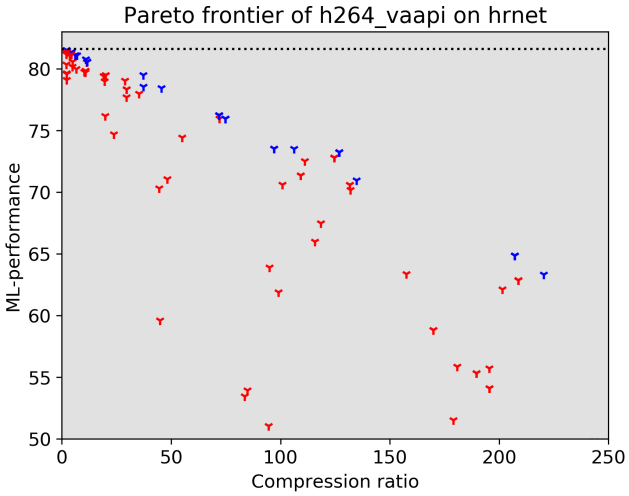
The multi-objective optimisation rendered several sets of coding parameters that were Pareto optimal for each evaluated encoder. The fitness of each evaluated set is plotted, where blue markings represent Pareto optimal sets and red points are dominated sets, in the graphs of this section. A horizontal dotted line, which represents the baseline ML-performance, is present in each plot. Desirable properties of a coding parameter set are ML-performance near the baseline and a high compression ratio. What can be observed, for each encoder, is a seemingly linear trade-off between ML-performance loss and compression ratio.
The optimisation rendered many non-dominated sets that could be considered potential sweet spots for encoding ML-data. However, as mentioned in Section 3.3, only non-dominated coding parameters that achieved an ML-performance over 80% of the baseline were selected as sweet spot candidates. The sweet spot candidates' coding parameters can be found in Appendix B.5.

4.1.1 HRNet

The Pareto frontiers achieved from optimisation of the encoders on HRNet are shown in Figures 4.1 to 4.4.

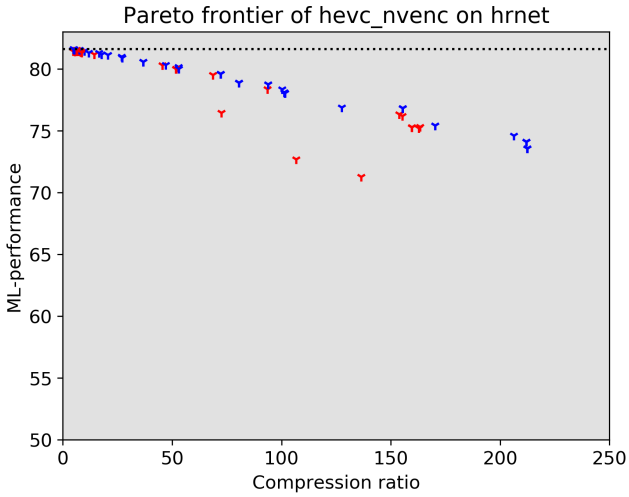


(a) NVIDIA accelerated H.264

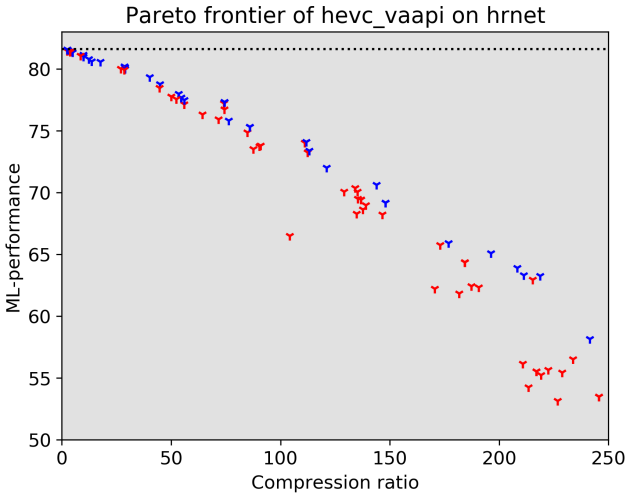


(b) Inte accelerated H.264

Figure 4.1: Pareto frontiers for Hardware encoded H.264.

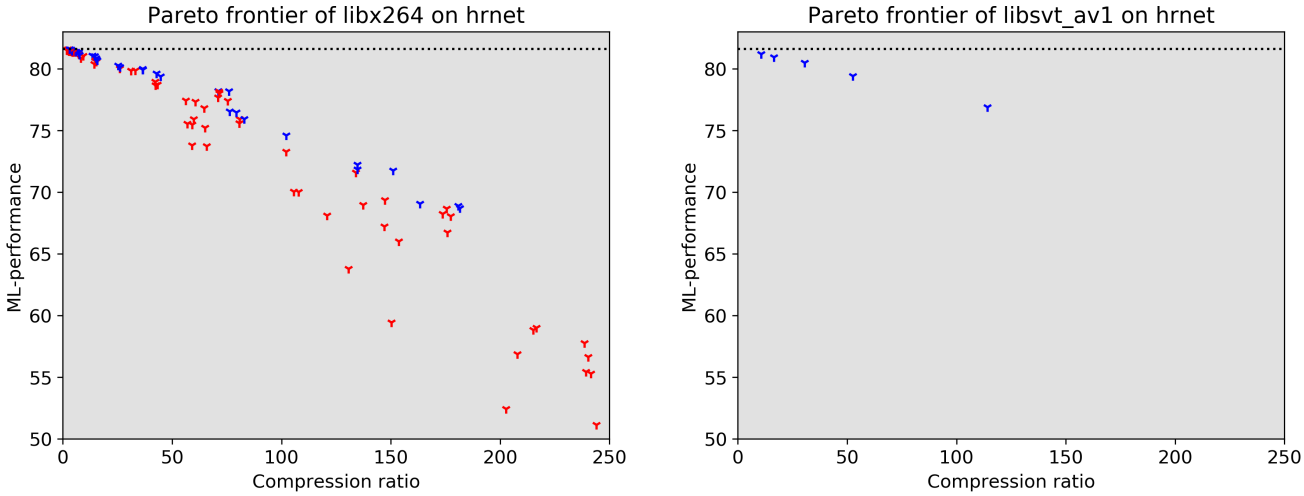


(a) NVIDIA accelerated HEVC



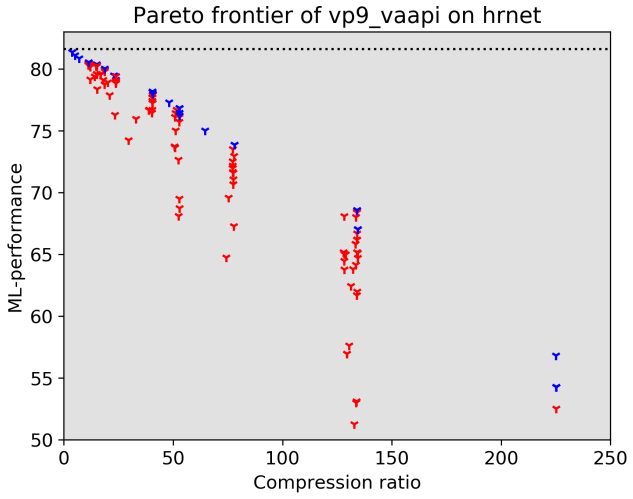
(b) Intel accelerated HEVC

Figure 4.2: Pareto frontiers for Hardware encoded HEVC.



(a) Software encoded H.264 **(b)** Software encoded AV1

Figure 4.3: Pareto frontiers for Software encoders.

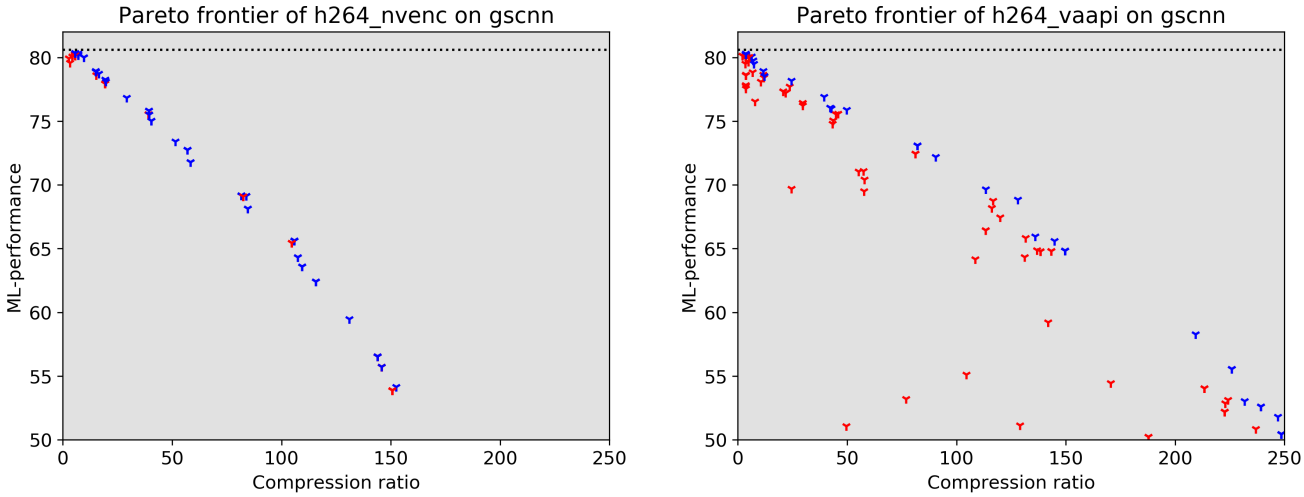


(a) Intel accelerated VP9

Figure 4.4: Pareto frontier for Hardware encoded VP9.

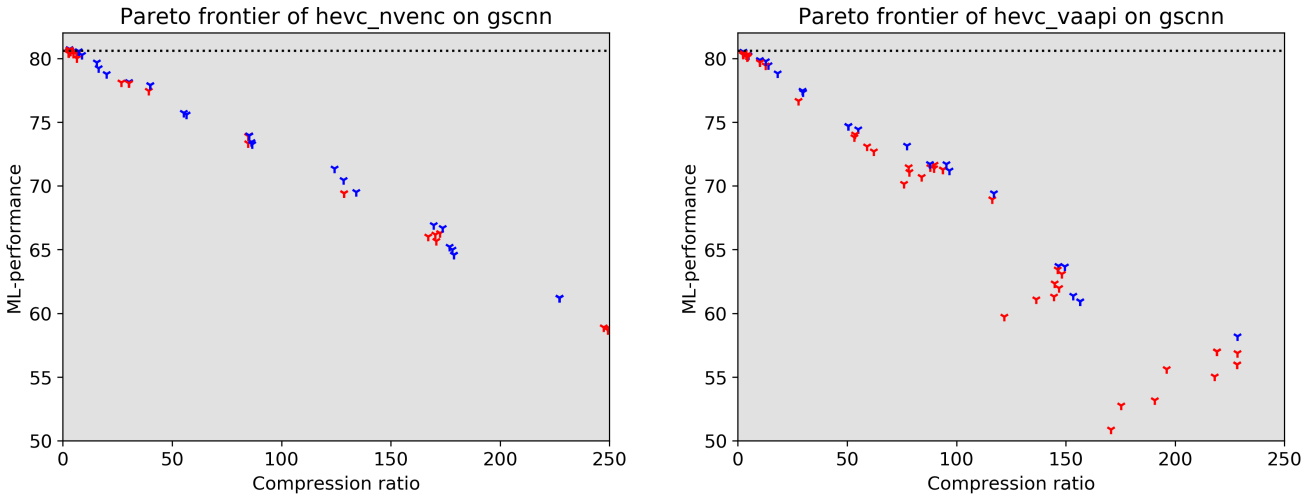
4.1.2 GSCNN

The Pareto frontiers achieved from optimisation of the encoders on GSCNN are shown in in Figures 4.5, 4.6, 4.7 and 4.8.



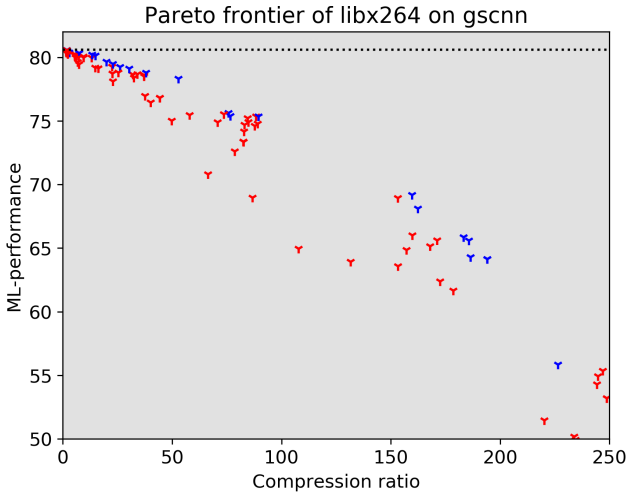
(a) NVIDIA accelerated H.264 **(b)** Intel accelerated H.264

Figure 4.5: Pareto frontiers for Hardware encoded H.264.

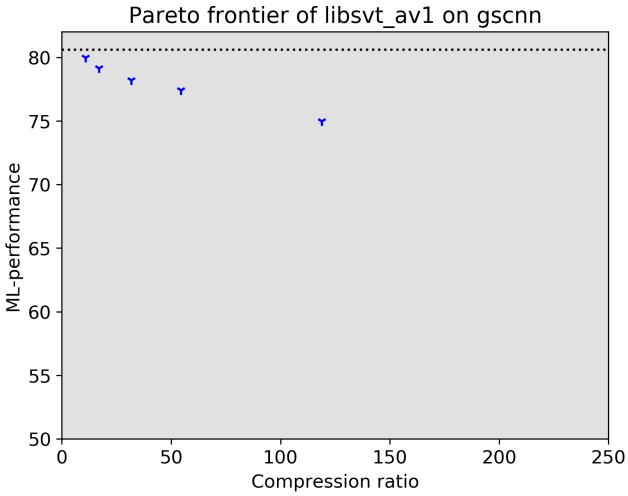


(a) NVIDIA accelerated HEVC **(b)** Intel accelerated HEVC

Figure 4.6: Pareto frontiers for Hardware encoded HEVC.

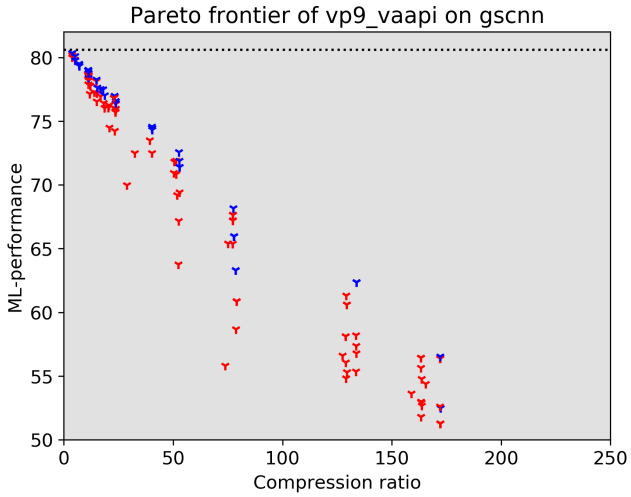


(a) Software encoded H.264



(b) Software encoded AV1

Figure 4.7: Pareto frontiers for Software encoders.



(a) Intel accelerated VP9

Figure 4.8: Pareto frontier for Hardware encoded VP9.

4.2 Research questions

A selection of sweet spot candidates were extracted from the optimisation results presented in Section 4.1. The sweet spot candidates, along with their encoder, represents the optimum video compression methods that can be used to downsize ML-datasets. The evaluation data collection, described in Section 3.3, was performed for each sweet spot candidate. The collected evaluation data was analysed and tested to answer the hypothesises and research questions.
Test results for both research questions are plotted in figures under Section 4.2. The test results are presented using scatter-plots, where the compression ratios and T-scores for each sweet spot tested placed on the x and y-axis respectively. The horizontal bar on the scatter-plots represents the critical T-value, which if exceeded, indicates that a sweet spot is significantly affecting the ML-algorithm used as the experimental unit. The sweet spot candidates chosen are shown in Appendix B.5 along with their test results and other statistics. Sweet spots, which errors were not normally distributed, do not fulfil the T-test assumption of normal distribution and hence are not included in these plots. Plots of the Shapiro-Wilk test results are shown in Appendix A.2.
In addition to the test results, the sweet spot candidates' ML-performance behaviour is illustrated in three different plots. Under Section 4.2, the mean ML-performance and standard deviation achieved by the encoder are plotted to further assess their differences. Box-plots, which illustrate the performance error distribution throughout the evaluated situations, are placed in Appendix A.1. For these plots, no sweet spots are excluded due to non-normal distribution of errors.

Research question I

Answering: *Are there "sweet spots" for each encoder's coding parameters with respect to maximising the compression ratio while not impacting the ML-algorithms' performance?*
The dotted line of the T-test plots represents the threshold that indicates if a sweet spot candidate's applied compression is significantly altering the ML-performance. Scores close to zero indicate an insignificant effect on ML-performance while high scores indicate that the ML-performance has been significantly altered.
The T-tests performed on collected data indicated that some encoders have sweet spots that reduce dataset footprint while not significantly affecting the ML-performance of HRNet and GSCNN. As expected, lower compression ratios allow the encoders to compress the data in a way that discards as little vital information as possible. The mean ML-performance plots showed results similar to the ones found in the plots for the optimisation. The standard deviation plots indicated that generally, the higher the compression ratio an encoder is given, the larger the differences between ML-performance errors among situations is. Further discussion regarding the results of research question 1 is held in Section 5.2.1.

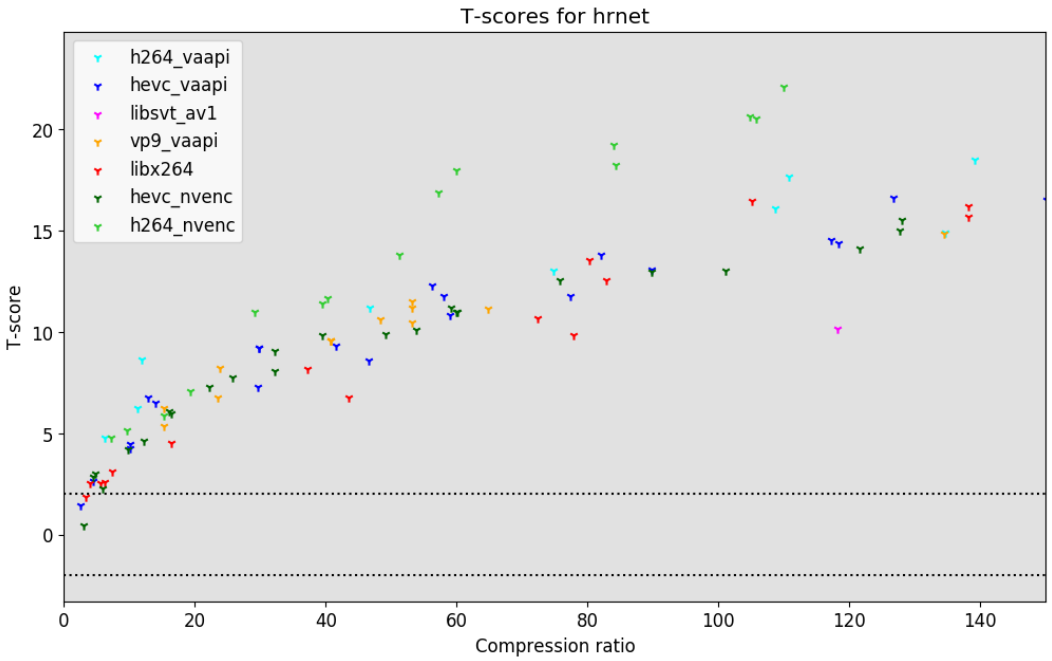


Figure 4.9: Plots of T-scores for the HRNet sweet spot candidates.

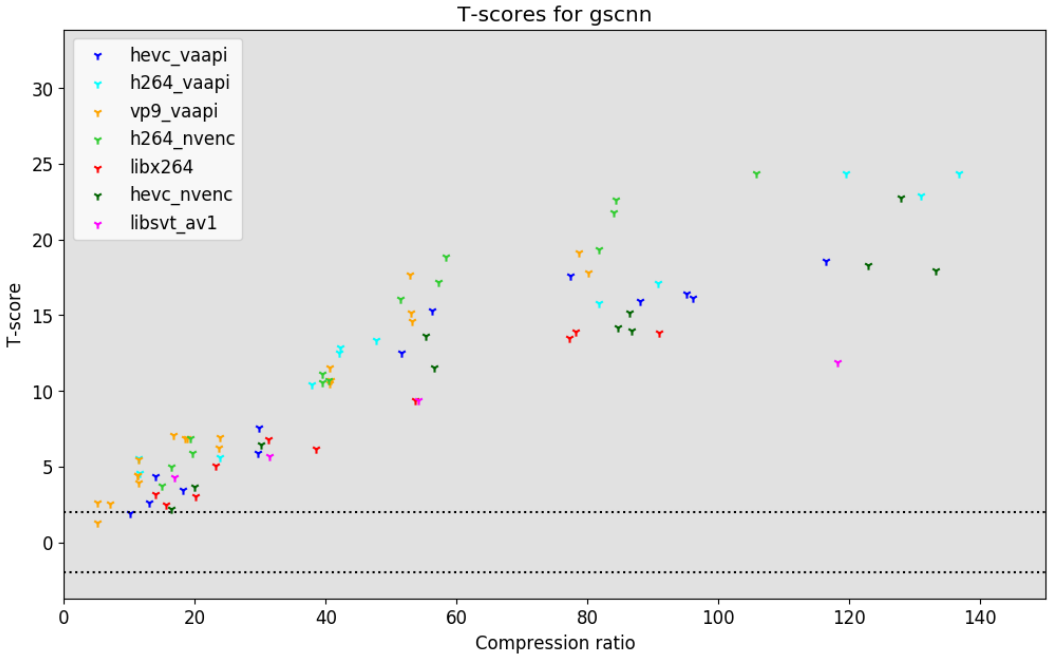


Figure 4.10: Plots of T-scores for the GSCNN sweet spot candidates.

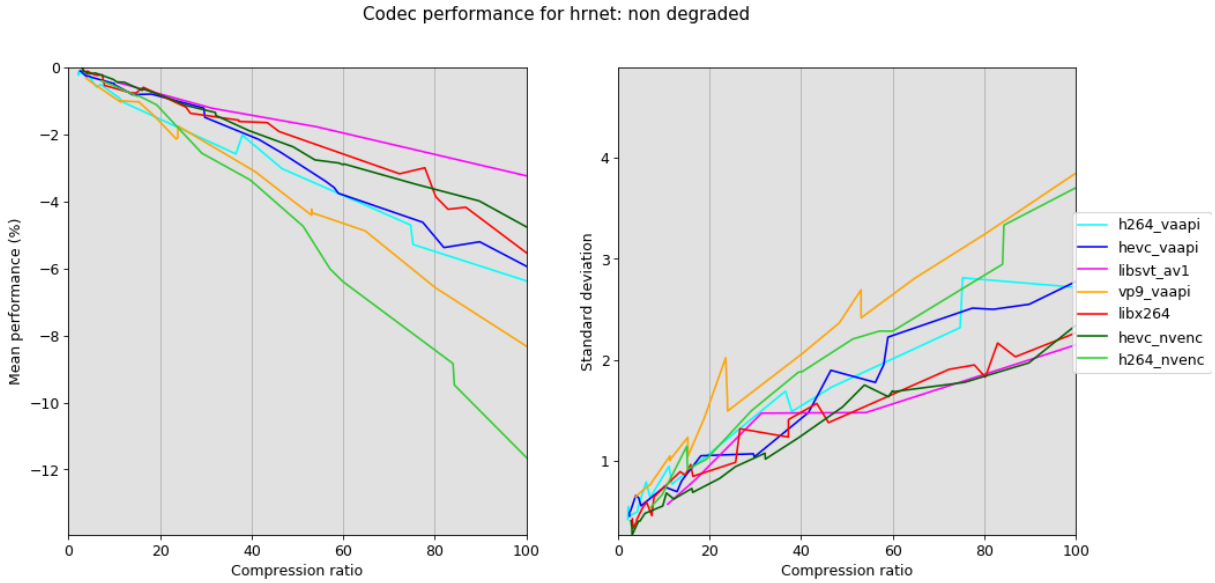


Figure 4.11: Sweet spot candidates' HRNet ML-performance mean and standard deviation of errors on the non degraded dataset, illustrated using plots.

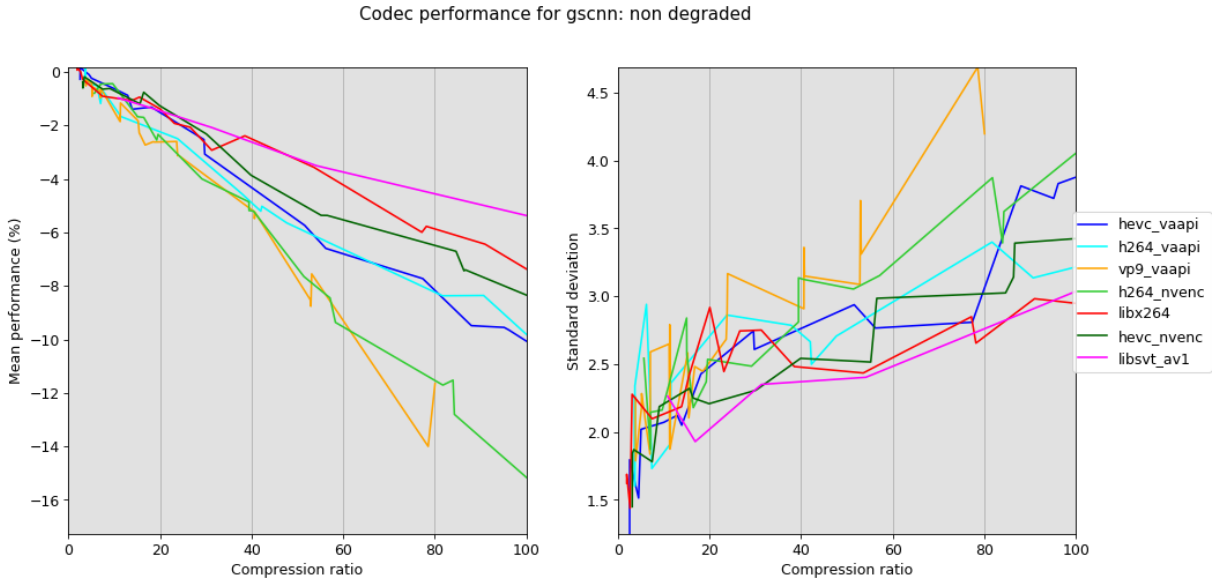


Figure 4.12: Sweet spot candidates' GSCNN ML-performance mean and standard deviation of errors on the non degraded dataset, illustrated using plots.

Research question 2

Answering: *To which degree are machine learning algorithms affected by degrading the visual quality of video input data?*
Similarly to the results of research question 1, the dotted line of T-test plots represents the threshold of significant change in ML-performance and scores close to zero indicate an insignificant effect on ML-performance. As explained in Sub-section 3.3.2.1, three versions of the dataset were created with various visual degradations added to them. The data collection and analysis results for the rain, Gaussian noise and movement degraded datasets are presented in plots unique for each degraded dataset.

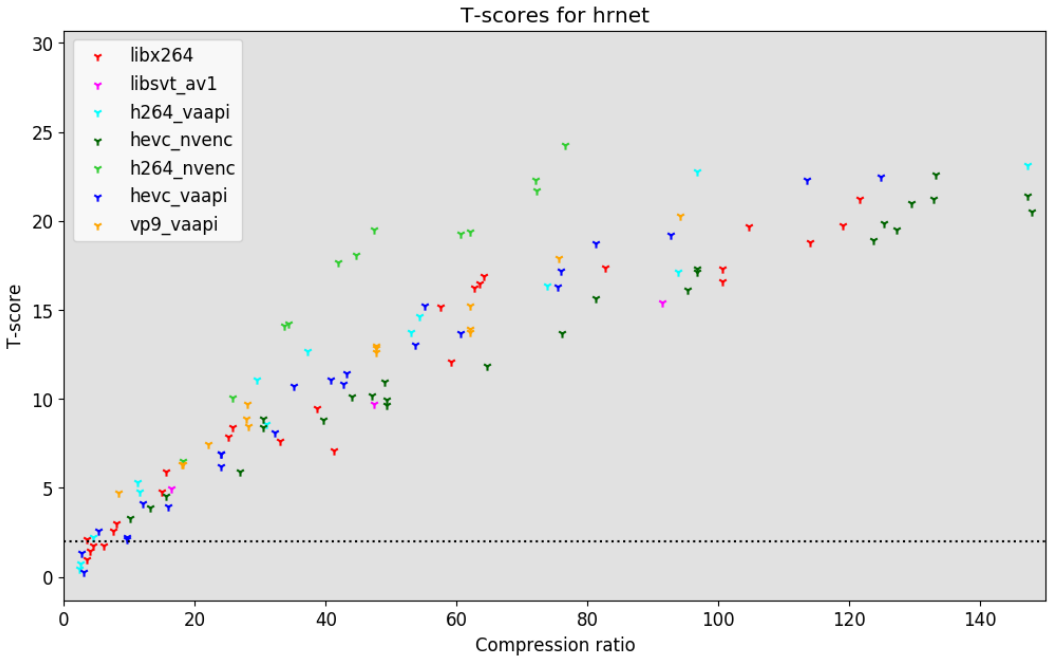


Figure 4.13: Plots of T-scores for the HRNet sweet spot candidates compressing the rain-degraded dataset.

In comparison to the non-degraded dataset, the datasets with added movement and rain effects saw a general increase in the scores the encoders achieved at a given compression ratio. To better distinguish significant differences, test results on the right-tail were presented as negative scores. The dataset with Gaussian noise got many sweet spot candidates that significantly altered the ML-performance, but in a way that increased the mean ML-performance.
For the datasets with added movement and rain, the mean ML-performance and standard deviation plots resembled the plots of the non-degraded dataset to a large extent. The mean ML-performance plots for the Gaussian noise dataset indicated, as with the T-score plots, that the ML-performance was generally improved. The standard deviation plots of the Gaussian noise dataset showed a more rapid increase in deviation at lower compression ratios that levelled off at higher compression ratios. Further discussion regarding the results of research question 2 is held in Section 5.2.2.

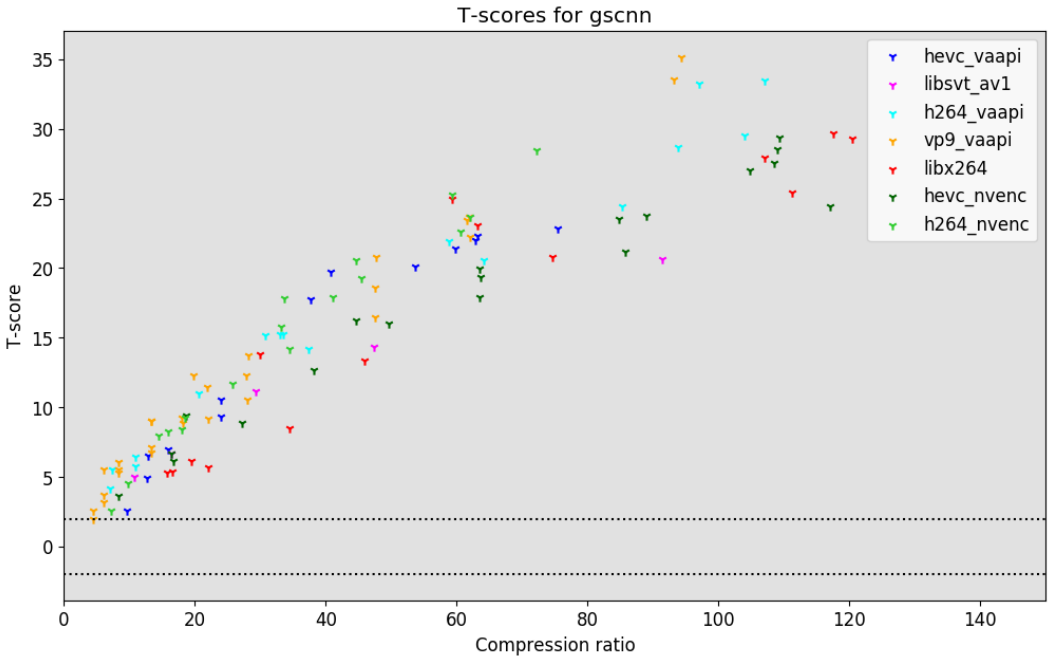


Figure 4.14: Plots of T-scores for the GSCNN sweet spot candidates compressing the rain-degraded dataset.

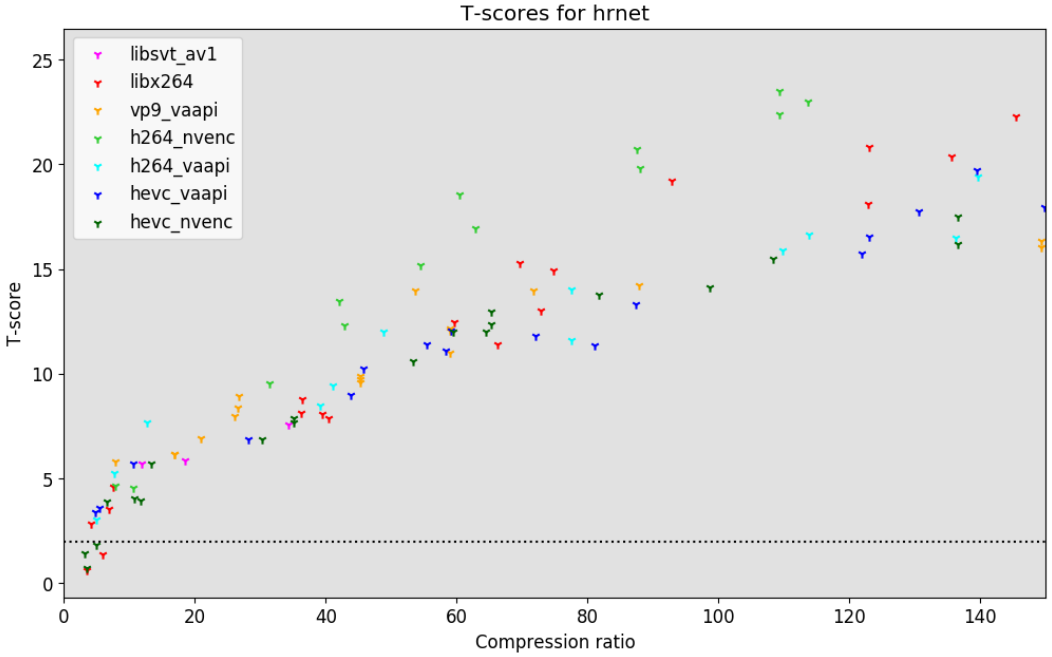


Figure 4.15: Plots of T-scores for the HRNet sweet spot candidates compressing the movement-degraded dataset.

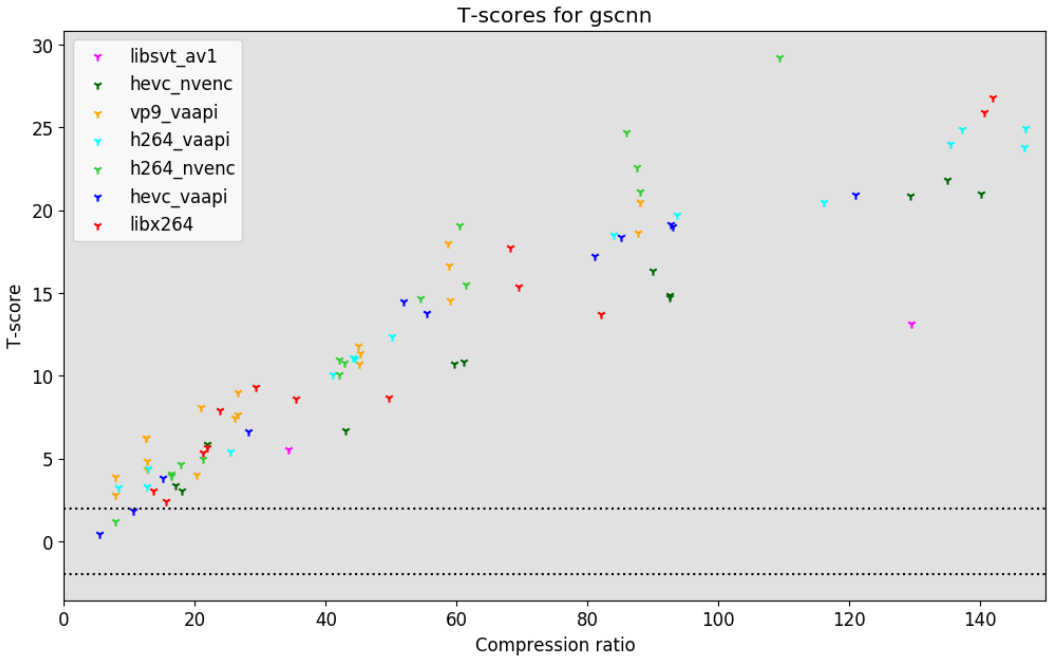


Figure 4.16: Plots of T-scores for the GSCNN sweet spot candidates compressing the movement-degraded dataset.

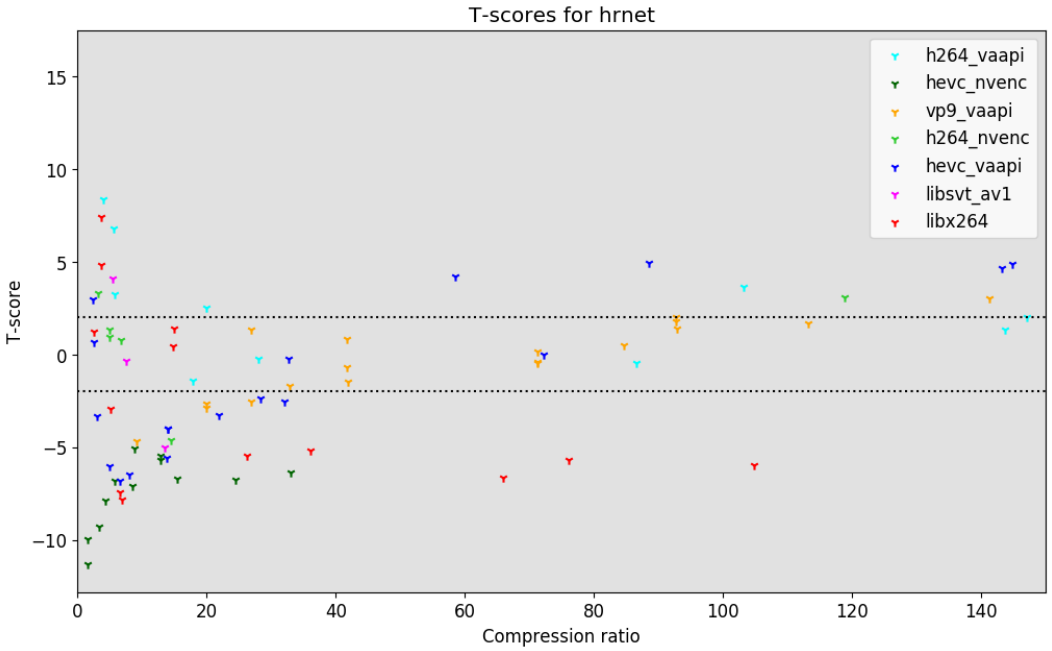


Figure 4.17: Plots of T-scores for the HRNet sweet spot candidates compressing the noise-degraded dataset.

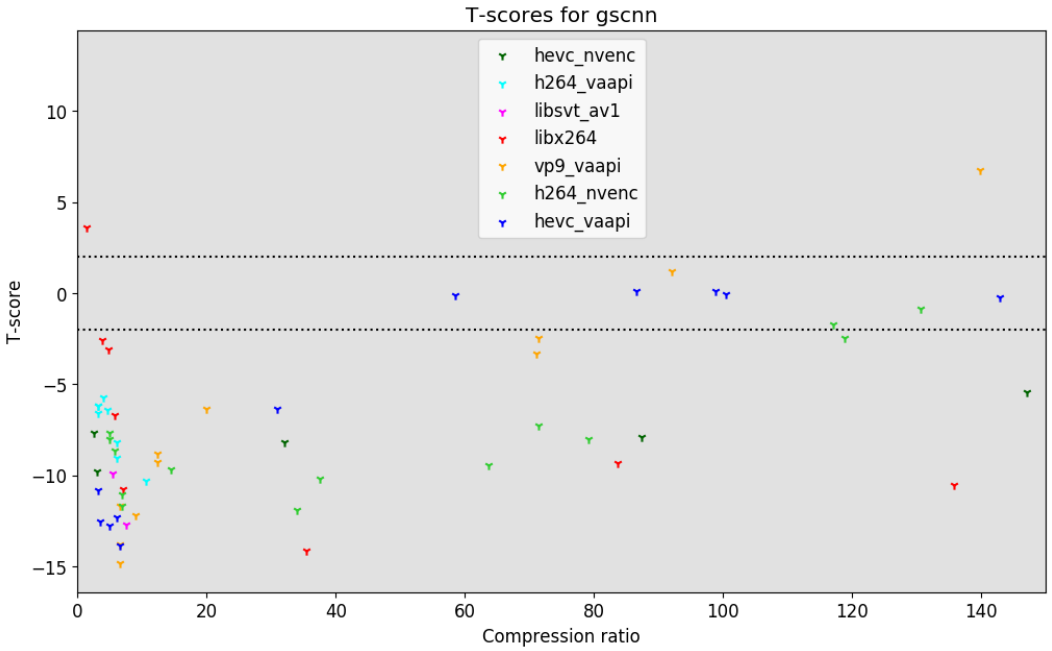


Figure 4.18: Plots of T-scores for the GSCNN sweet spot candidates compressing the noise-degraded dataset.

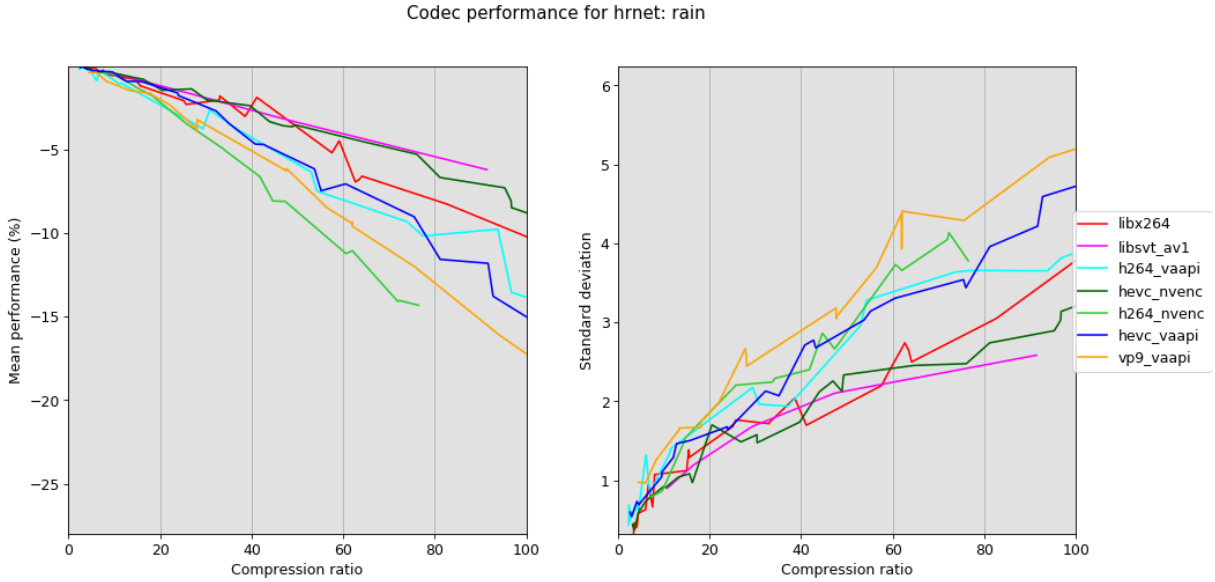


Figure 4.19: Sweet spot candidates' HRNet ML-performance mean and standard deviation of errors on the rainy dataset, illustrated using plots.

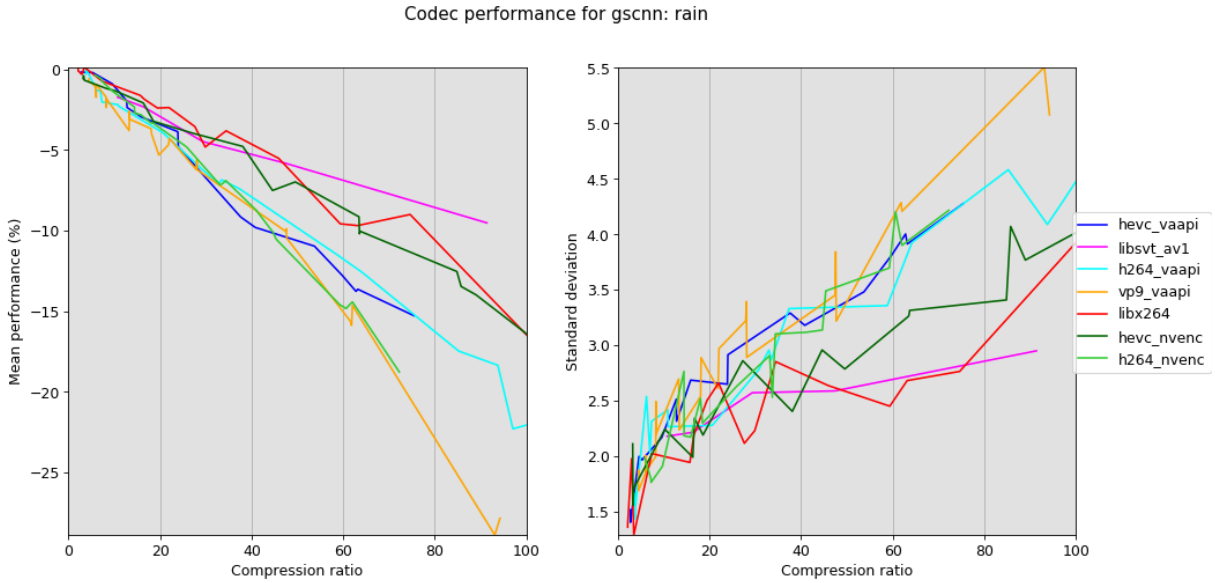


Figure 4.20: Sweet spot candidates' GSCNN ML-performance mean and standard deviation of errors on the rainy dataset, illustrated using plots.

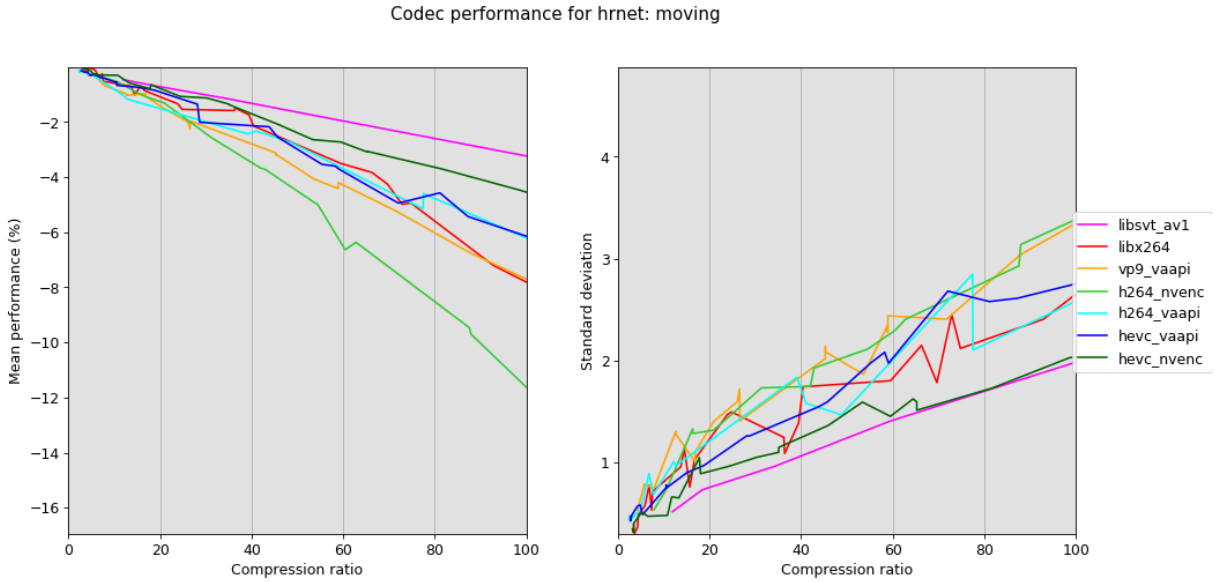


Figure 4.21: Sweet spot candidates' HRNet ML-performance mean and standard deviation of errors on the added movement dataset, illustrated using plots.

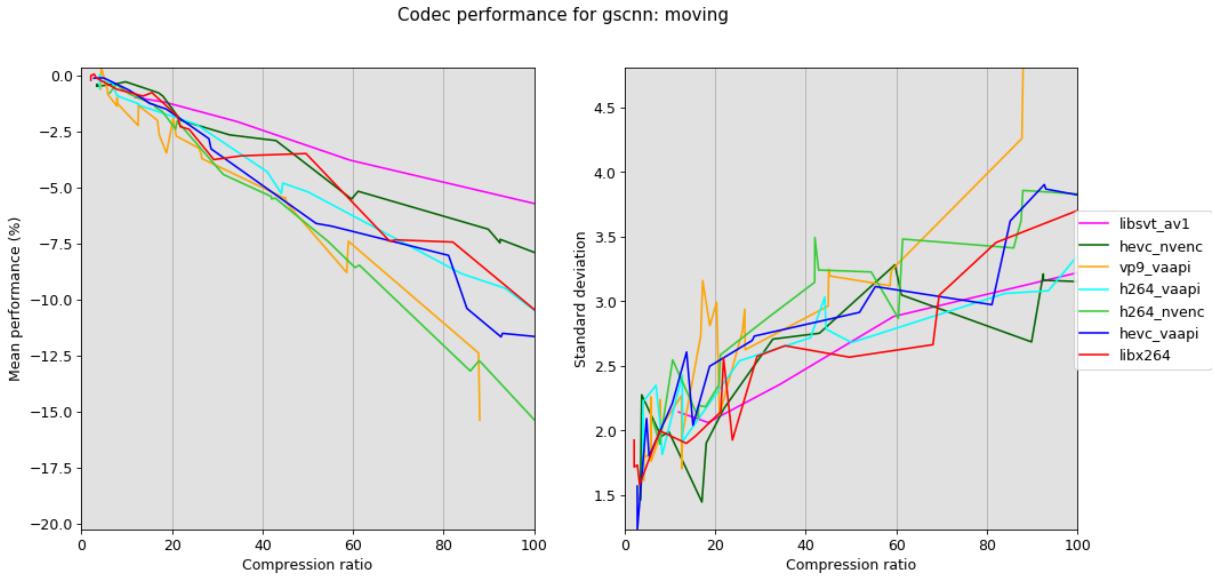


Figure 4.22: Sweet spot candidates' GSCNN ML-performance mean and standard deviation of errors on the added movement dataset, illustrated using plots.

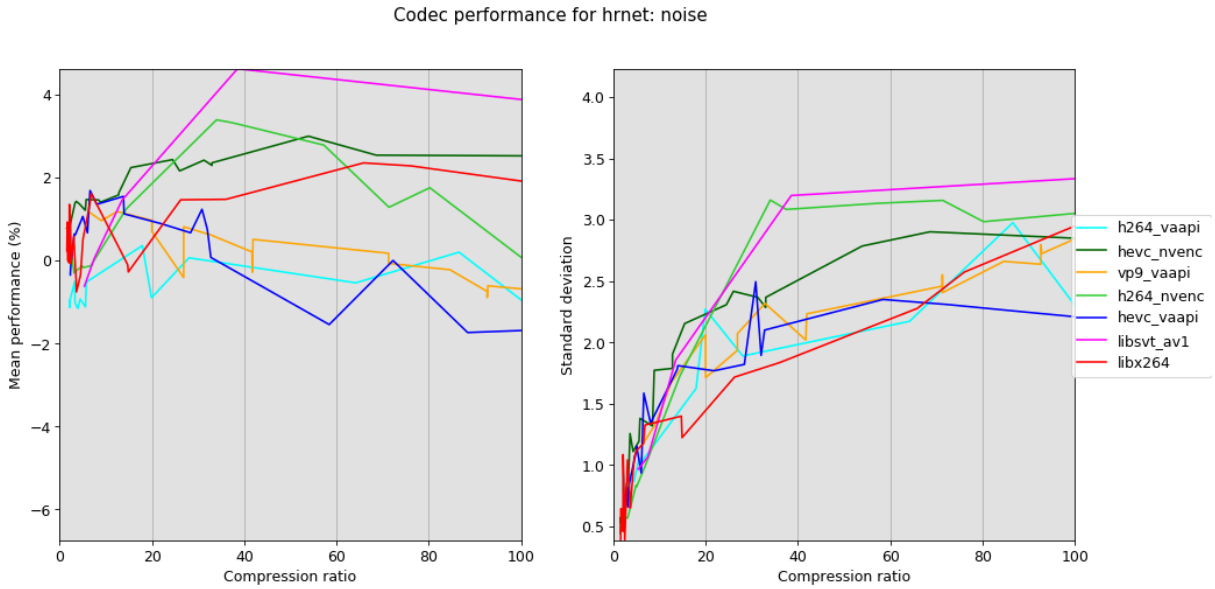


Figure 4.23: Sweet spot candidates' HRNet ML-performance mean and standard deviation of errors on the noisy dataset, illustrated using plots.

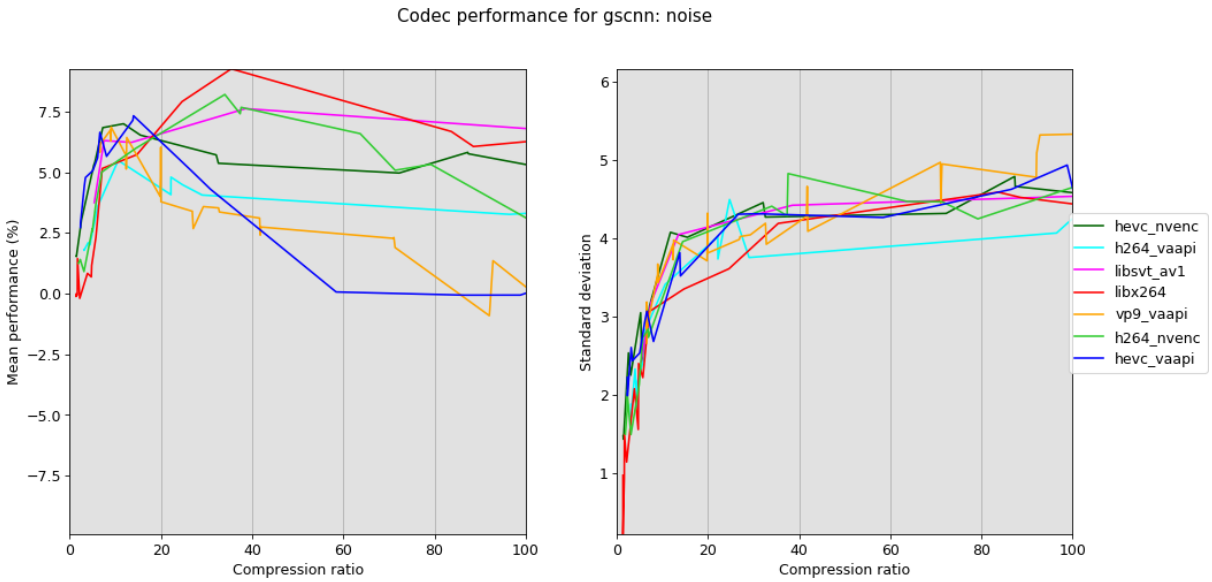
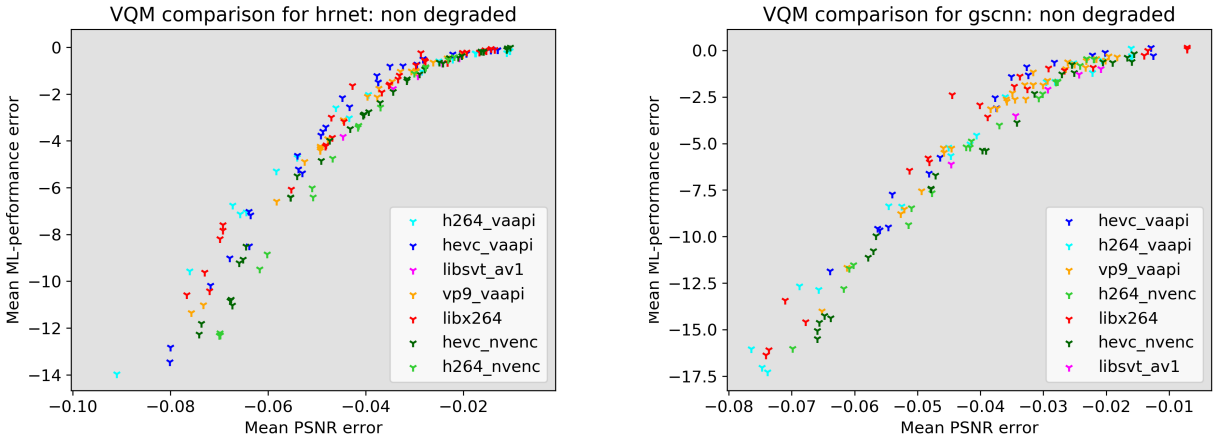


Figure 4.24: Sweet spot candidates' GSCNN ML-performance mean and standard deviation of errors on the noisy dataset, illustrated using plots.

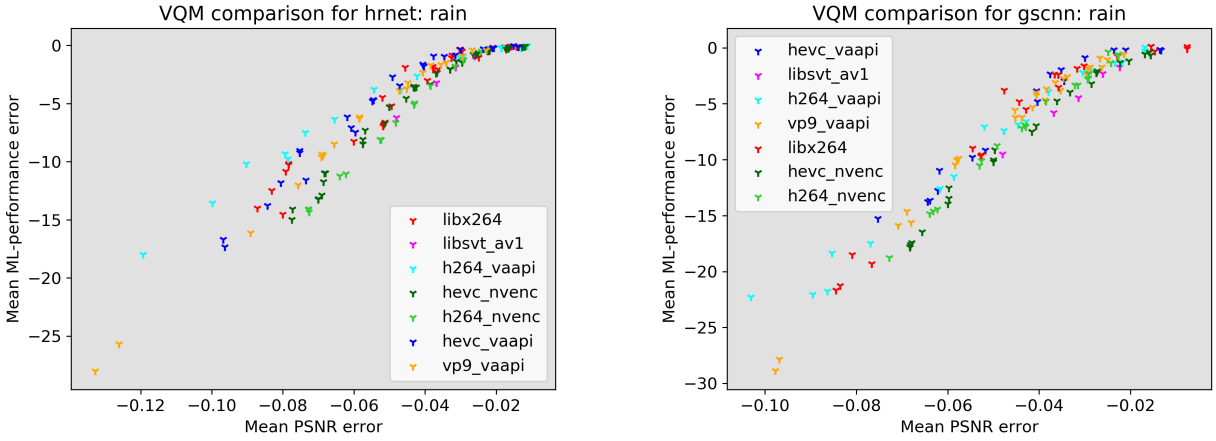
4.3 Visual quality metrics and ML-performance

While collecting performance data for the hypothesises, the PSNR for each situation was also calculated for each situation. Each sweet spot candidates' mean ML-performance error and mean PSNR is plotted in the following figures.



(a) HRNet PSNR results **(b)** GSCNN PSNR results

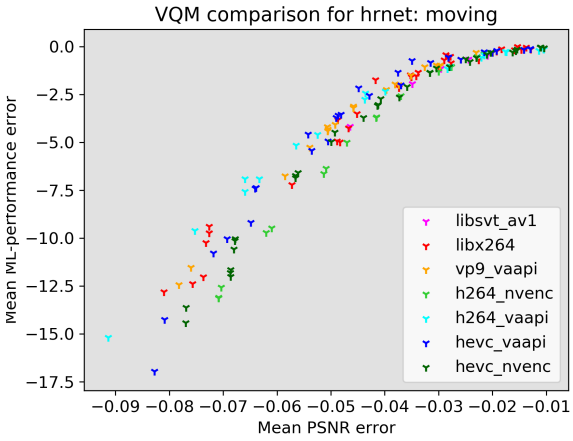
Figure 4.25: Plots of mean ML-performance and PSNR error for the sweet spot candidates compressing the non degraded dataset.



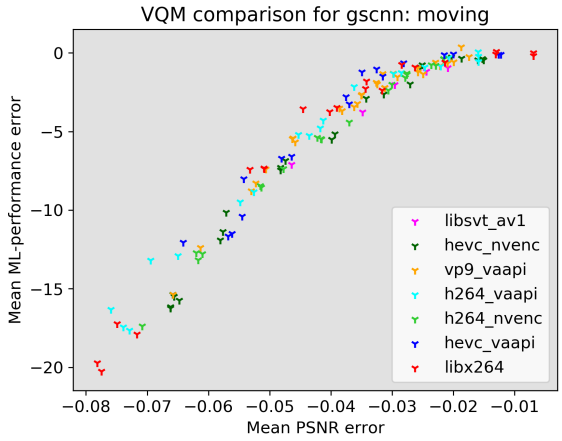
(a) HRNet PSNR results **(b)** GSCNN PSNR results

Figure 4.26: Plots of mean ML-performance and PSNR error for the sweet spot candidates compressing the rainy dataset.

A linear relationship between the mean PSNR and mean ML-performance can be observed for the non-degraded, rain-degraded and movement-degraded datasets. This may indicate that PSNR can be a useful metric for tuning coding parameters for ML-purposes. Further discussion regarding visual quality metrics can be found in Section 5.3.

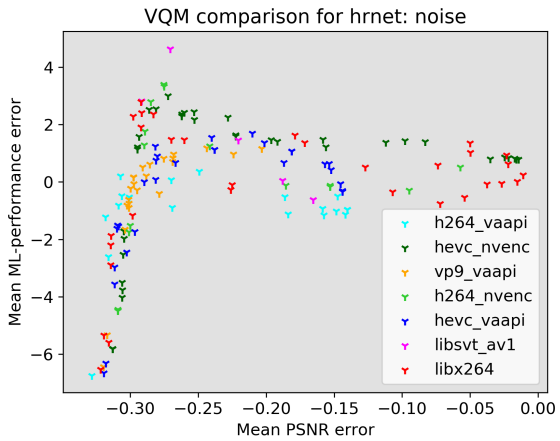


(a) HRNet PSNR results

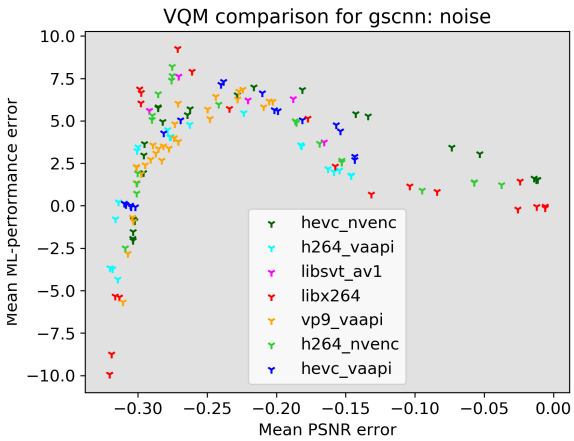


(b) GSCNN PSNR results

Figure 4.27: Plots of mean ML-performance and PSNR error for the sweet spot candidates compressing the dataset with simulated movement.



(a) HRNet PSNR results



(b) GSCNN PSNR results

Figure 4.28: Plots of mean ML-performance and PSNR error for the sweet spot candidates compressing the noisy dataset.

5
Analysis and Discussion

In this section, the results of the experiment are examined. Answers to the research questions were derived from these results and observations regarding the research goal are discussed.

5.1 Optimum coding parameters

The multi-objective optimisation rendered several sets of coding sweet spot candidates for each encoder. A linear relationship between the compression ratio and ML-performance could be observed for each encoder's non-dominated solutions. The difference of optimality between dominated and non-dominated solutions varied between encoders. Both NVIDIA accelerated encoders had dominated solutions that performed comparably to the non-dominated solutions. The software encoded H.264 and Intel accelerated encoder displayed a more substantial difference between dominated and non-dominated solutions.

The validation set performance of HEVC encoders was generally higher or equal to that of H.264 encoders at similar levels of compression ratio, with NVIDIA accelerated HEVC generally exceeded the Intel accelerated HEVC. Intel accelerated VP9, in the configuration used for this optimisation, did not fare as well as the other Intel accelerated encoders. Although the encoder used for representing AV1 was a pre-release of the "work-in-progress" SVT-AV1 encoder, the AV1 performance on the validation set was comparable to that of NVIDIA accelerated HEVC.

5.2 Effects of lossy compression

In this sub-section, the T-score and mean ML-performance results displayed in Section 4 are discussed along with additional plots from Appendix A.1.

For both research questions, several sweet spots of each encoder did not fulfil the normal distribution of errors assumption for the dependent T-test and could not be used for the T-test. A larger concentration of discarded sweet spots was in the 1-15 compression ratio region, as can be seen in figures in Appendix A.2. Although some sweet spot candidates could not be tested, analysis of all sweet spot candidates regardless of fulfilled T-test assumptions is done in the performance error analysis sub-sections. An example of how video compression artefacts can visually look like for the non-degraded and degraded data is provided in Appendix B.3.

5.2.1 Research question 1

Discussion regarding: *Are there "sweet spots" for each encoder's coding parameters with respect to maximising the compression ratio while not impacting the ML-algorithms' performance?*

Test-results

The T-tests performed on data collected from the various situations extracted from the Cityscapes dataset, shown in Figures 4.9 and 4.10, indicated that there are, for some encoders, parameter sweet spots that reduces the dataset size while not significantly affecting the ML-performance of HRNet and GSCNN. The Intel encoded HEVC had one such sweet spot for each ML-algorithm accompanied by one NVIDIA encoded HEVC and a libx264 sweet spot for HRNet and one VP9 sweet spot for GSCNN. Due to some sweet spot candidates failing T-test assumptions, a definite answer cannot be provided for research question 1. What can be concluded is that some encoders were able to maintain ML-performance, while other encoders could not be proven to being capable of insignificant ML-performance impacts.
Additionally, analysis of how the encoders compare when it comes to T-scores at given compression ratios is of great interest. The libx264 and AV1 encoders often achieved low T-scores relative to the other encoders while the hardware-accelerated H.264 encoders often achieved higher scores. The HEVC encoders were often positioned such that they were not better than libx264 and AV1, but not worse than the hardware encoded H.264 encoders. The VP9 encoder performed similarly to the HEVC encoders for HRNet, but for GSCNN performed more in line with the hardware encoded H.264 encoders. The video compression artefacts, in general, seems to affect HRNet to a larger extent than GSCNN.

Performance error analysis

The performance error analysis made in this subsection concerns Figures 4.11 to 4.12 and A.1 to A.2. The hardware-accelerated H.264 encoders generally perform worse than their HEVC counterparts with regards to mean performance error. libx264 showed good performance and affirmed that the software encoder is a well developed one and somewhat legitimise continued use of the encoder. The AV1 encoder was often on par or even outperformed the other encoders, with regards to mean performance error, while having a relatively concentrated spread of errors, which bodes well for the future of AV1 encoders. For both ML-algorithms, the Intel accelerated H.264 performed better and had a lesser spread of errors than its NVIDIA accelerated counterpart. For HEVC, the NVIDIA accelerated encoder performed better than its Intel accelerated counterpart. The NVIDIA accelerated H.264 and VP9 were among the encoders that performed the worst.
Each encoder had sweet spots that had several performance outliers, which were the result of cases where the sweet spot struggled or excelled at a particular situation. With regards to outliers, none of the encoders stood out as being particularly susceptible to outlier situations.
Generally, the higher the compression ratio an encoder is given, the larger the dif-

ferences between ML-performance errors among situations is. This means that, at high compression ratios, encoders might be able to maintain good performance in some situations while performing poorly in other situations. The spread of the errors differed between the encoders. The HEVC encoders generally saw slightly less spread than their H.264 counterparts. The libx264 and AV1 encoders generally had the most consistent performance while VP9 often had the largest spread among the encoders tested.

5.2.2 Research question 2

Discussion regarding: *To which degree are machine learning algorithms affected by degrading the visual quality of video input data?*

Test-results

The T-tests were performed on each degraded dataset, which results are shown in Figures 4.13 to 4.18. Contrary to hypothesis 2, there were coding sweet spots for each dataset that was able to compress the dataset in a way that did not affect the ML-performance significantly. Compared to the original dataset, the datasets with added movements and rain effects saw a general increase in the scores the encoders achieved at a given compression ratio. The coding parameter sweet spots that used set quantizers or quality in most cases saw a small increases to the T-score but also a larger decrease in the compression ratio.
The dataset with added movement saw both Intel accelerated encoders decreasing their scores and libx264 increasing its HRNet scores somewhat relative to the other encoders at a given level of compression ratio. The GSCNN results for the added movement dataset did not see any major changes in relative T-scores between the encoders. For the rainy dataset, the NVIDIA accelerated HEVC encoder relatively decreasing its scores while the AV1 encoder increased its scores for HRNet, but also to a lower degree for GSCNN.
For HRNet, the noisy dataset saw an impressive VP9 performance with regards not significantly altering performance. However, many sweet spots for the various encoders improved the performance such that the T-score got right-tailed results. NVIDIA accelerated HEVC and libx264 were encoders that stood out as significantly altering the error results through performance increases. For GSCNN, no encoders stood out as having notably higher right-tailed T-scores in general.

Performance error analysis

The performance error analysis made in this subsection concerns Figures 4.19 to 4.24 and A.3 to A.8. The movement dataset gave similar results, for both ML-algorithms, to the ones for the non-degraded dataset. The AV1 and NVIDIA accelerated HEVC encoders performed very well, while libx264 performed well but lost some of its performance advantage seen in the non-degraded dataset. The Intel accelerated HEVC encoder lost some performance relative to the other encoders while its H.264 equivalent saw some performance gains. Some VP9 sweet spots had, for both ML-algorithms, quite inconsistent performance compared to the other encoders. The

other encoders sweet spots had mostly similar spread of error, at the same level of compression ratio, as each other.
The rainy dataset, again, saw similar results to the non-degraded dataset. However, the NVIDIA accelerated HEVC improved its performance relative to the other encoders while its H.264 counterpart lost in some cases, performance at high compression ratios. For both ML-algorithms, some Intel accelerated HEVC sweet spots lost some performance at higher compression ratios.
When interpreting the results of the noisy dataset one must take into consideration that the quantization and filtering properties of the video encoders will in many instances cause the compressed ML-performance to be significantly higher than that of the uncompressed counterpart. In this case, low T-scores are not indicative of desirable compression performance. The box-plots reveals that coding sweet spots with low compression ratios generally has less error variance but may not improve the ML-performance as much as other sweet spots using heavier compression.
As with the previous evaluations, the hardware-accelerated H.264 encoders generally perform worse than their HEVC counterparts. The AV1 encoder was often on par or even outperformed the other encoders while having a more concentrated spread of errors, which bodes well for the future of AV1 encoders. At low compression ratios, several libx264 sweet spots achieved very low error variance and maintained ML-performance very similar to the uncompressed noisy dataset. At higher compression ratios, the libx264 sweet spots usually compared well to the other encoders but are some instances falling behind with regards to increasing ML-performance. At high compression ratios over 50, the Intel accelerated HEVC and H.264 encoders had more equal mean error, and in some cases, the H.264 encoder outperformed the HEVC one.
Although the tested degradations had coding sweet spots that were able to maintain ML-performance, this does not guarantee that these sweet spots are impervious to any data degradation. However, the data collected for the degraded datasets show encoders' behaviour for datasets with very granular disturbances(noise), less granular disturbances(rain effect) and with added movement that challenge the encoders' motion compensation and vector prediction capabilities.

5.3 Visual quality metrics and ML-performance

The additional PSNR data collected was used to provide insight into the extent that video compression artefacts affect PSNR in comparison to ML-performance. Scatter plots under Section 4.3 illustrate how the mean PSNR error and the mean ML-performance error correlate to each other. What can be observed for all datasets, with the exception of the noisy dataset, is that there is a pattern recurring for all encoders tested. The characteristics of this pattern are:

- Initially, at low levels of compression artefacts, mean PSNR is affected to a larger extent than the mean ML-performance. During this point, larger changes in PSNR only reflects smaller changes in ML-performance.
- At high levels of compression artefacts, a linear relationship can be observed between the mean PSNR error and mean ML-performance error.

- At a certain point, at approximately -0.03 mean PSNR for hrnet and -0.025 for GSCNN, the relationship between PSNR and ML-performance gradually shifts from the former characteristic, of stagnant ML-performance compared to PSNR, to the latter characteristics, where increased levels of video compression affect PSNR and ML-performance at rates that are linear to each other.

Although the encoders follow the same characteristics with regards to mean PSNR versus ML-performance, they are offset from each other such that some encoders achieve higher PSNR scores compared to other encoders at the same level ML-performance. The NVIDIA accelerated encoders and the AV1 encoder generally achieved higher mean PSNR scores at a certain level of ML-performance while libx264 and Intel accelerated encoders got lower mean PSNR scores. This means that there could be a slight difference between encoders in how their PSNR performance could be translated into ML-performance.

As expected, this is not the case for the dataset with Gaussian noise. For this dataset, the discrete cosine transform of the encoders can act as a noise filter due to high-frequency components being discarded. This translates to a pattern between mean PSNR and mean ML-performance in which PSNR measurements are not indicative of expected ML-performance. In this case, PSNR penalizes encoders heavily for denoising the data instead of reconstructing.

As previously mentioned, the models used for HRNet and GSCNN were trained on a dataset that was not compressed using the sweet spots. If trained using data augmentation simulating video compression, the characteristics of ML-performance with regards to PSNR may change such that the ML-performance will be able to withstand higher levels of compression where the PSNR would steadily decline.

5.4 Coding parameters' sweet spots

Optimisation on small slices were done for each encoder, where the rate control methods were compared. It was found that for libx264, CRF was was slightly better than the constant and variable bitrate and Constant Quantization Parameter(CQP) methods of rate control. CQP was chosen for all NVIDIA accelerated encoders but had quite similar performance to the constant and variable bitrate methods. For Intel accelerated H.264 and HEVC, CQP was also chosen for the same reasons as for the NVIDIA encoders. Unfortunately, the Intel accelerated VP9 was not functioning properly when using CQP had to be optimized using constant and variable bitrate control methods. The VP9 encoder might have been able to reach slightly higher levels of performance if it had been using CQP.

For the 30 frames per second dataset, increases to a group of pictures parameter seem insignificant after a while. Cases with values over 250 for this parameter seem to have a negligible effect on the results. For the encoders using the b-frames parameter, values were mostly around the 5 - 10 with the exception of VP9, which had many sweet spots with b-frames set to 0. Scenecut's values spread from values as low as 12 to values up to 56 in a uniform manner. For the b-frames, RC-lookahead and scenecut coding parameters, there should be no problem using the default values of the encoders.

More detailed information regarding the sweet spot candidates and their results can be found in Appendix B.5.

5.4.1 NVIDIA accelerated encoders

For the NVIDIA encoders, the "slow" and "fast" preset was common among the sweet spots, the surface parameter was commonly spread between 40 and 60, and the strict-GOP was most commonly disabled. The weighted pred enabled was mostly favoured among the NVIDIA encoders, which is interesting due to it disabling b-frames.

5.4.2 Intel accelerated encoders

The compression level parameter for the Intel encoders did not seem to have a significant impact. This might be due to the other parameters used for tuning overrides many preset parameter values. The number of reference frames for each P-frame had a widespread mostly in the region of 5-16, values over 5 do not seem to have a large impact on the ML-performance. A particular B-frame reference depth could not be observed.
For the VP9 sweet spots, the qmin parameter had values spanning mostly between 0 and 10 while qmax was mostly in the range of 45 to 69. This means that in most cases, quantizer limits were less restrictive. The VP9 "loop filter levels" varied between 24 and 55 and the "loop filter sharpness" was varying widely between 0 and 12. For VP9, the most often used rate control method was CBR. As previously mentioned, usage of CQP might have improved VP9's relative performance to the other encoders if CQP had been working as expected.

5.4.3 libx264

Trellis, intra-refresh, 8x8dct was most commonly enabled while mixed-refs and psy were most often disabled. The subpixel estimation complexity most commonly used were 1, 6 and 9. The max range for motion search was concentrated around 15 and 23. The hexagon and uneven multi hexagon were the most used motion estimation methods, while the exhaustive and transformed exhaustive method being less and diamond barely being used. No particular b-bias was preferred. There was several combinations of deblocking parameters that were used by the sweet spots with 2:1 being the most often recurring one. The spatial prediction was the most often occurring adaptive direct mode. The "normal" pyramidal b-frames was the most common. The RC-lookahead parameter usually saw values positioned above 30 with most values spread from 40 to 55. libx264's presets that span from "medium" to "placebo" has rc-lookaheads between 40 and 60, which indicates that the preset RC-lookahead values are sound for ML-purposes.

5.4.4 Final word on coding parameters

From what can be seen in from the sweet spots, the parameters used by the higher quality presets similar to what was found for the other coding parameters. For

most encoders, selecting high quality presets along with appropriate rate-control parameters will most likely results in good coding setups for video compression for ML. From such coding setups, practitioners can evaluate and tune the setups such that they provide a desirable level of compression.

5.5 Encoding speed

The encoding speed differed significantly between the encoders tested during this thesis. The encoding times were recorded during the optimisation for each evaluated set of coding parameters. In Appendix A.3, the encoding times for compressing the GSCNN validation slice, performed by the computer specified in Table 3.3, are presented using a boxplot. The boxplot shows the extensive time differences between encoders where the hardware-accelerated encoders have a significant time advantage. The NVIDIA accelerated encoders often took on average around 100 seconds to encode but could take over 200 seconds occasionally. The Intel accelerated encoders had on average times around 120 seconds for H.264 and VP9 while HEVC would take 160 seconds. libx264 had a much wider spread of encoding times and varied between 100 seconds and up to 575 seconds. The SVT AV1 encoder, which is at a pre-production state, was not included in the plot due to very high encoding times (usually over 1500 seconds).

5.6 Recommendations and best practices

Although the libx264 encoder performed well throughout the tests, the encoding time and CPU resources required make the encoder less suitable for real-world deployment. Due to the nature of the hardware-acceleration, NVENC and Quick Sync offloads encoding to dedicated hardware instead of CPU and GPU cores for ML and general-purpose computing. The hardware-accelerated HEVC encoders both have faster encoding-times and do not occupy resources that can be used elsewhere. The good performance combined with fast encoding times of these encoders makes them good candidates for compressing image sequences and video recordings for ML and AD. However, the NVIDIA accelerated HEVC is the recommended choice of encoder if one of the HEVC encoders is to be chosen over the other.
The rate control methods used for the Intel accelerated VP9 encoder during the optimisation and testing was constant and variable bitrate. The CQP method was used for the other Intel accelerated encoders, where the CQP method was deemed as slightly better than the bitrate control methods. VP9 might have been able to improve its relative performance to the other codecs if the CQP method had been properly working for it. Hence, it may be of interest for practitioners with access to hardware-accelerated VP9 to reevaluate the VP9 encoder's performance for their specific situation in relation to other encoding alternatives.
Even though the encoder was at a pre-production state, the SVT AV1 encoder was able to put up excellent ML-performance versus compression ratio. In the current state the SVT AV1 encoder, while utilising a stock i9-9900 CPU to nearly 100%, it takes a very long time to encode with the preset used for the sweet spots. Future

AV1 encoders, both software and hardware-accelerated, may offer high performance within reasonable encoding times. Hence, a recommendation for practitioners and researchers is to, for the moment, monitor the progress of these encoders and investigate them as they become viable for practical use.
The HRNet and GSCNN algorithms' networks used in this study were trained using the original uncompressed data. With models trained on data compressed using these coding sweet spots, the threshold at which the ML-algorithms are significantly affected, will most likely appear at higher compression ratios for all encoders. The effects of video compression methods can depend on the type of deep neural networks used, type of task performed, training data used and data augmentation used. Tests to ensure that an encoding setup is safe for deployment will have to be done on a case to case basis. Rigorous testing is strongly recommended to ensure maintained performance when choosing an suitable encoder and coding parameters for an ML-algorithm and dataset. A strong recommendation is to randomly select slices of the dataset, compress these sets of slices using a select set of encoder and coding parameter candidates and perform k-fold cross-validation on each set of slices. The encoders and coding parameters that are deemed to have the right trade-offs between ML-performance and compression can be considered for practical use. Before deployment of an encoder and coding parameters, rigorous cross-validation is recommended as not to degrade the quality of collected data permanently.

	libx264	**svt_av1**	**hevc_nv**	**hevc_va**	**h264_-nv**	**h264_-va**	**vp9_va**
Time	Slow	Very slow	Very quick	Quick	Very quick	Very quick	Very quick
ML-perf	Great	Great	Great	Good	Ok	Ok	Ok
Noise handling	Great	Great	Great	Ok	Good	Ok	Ok
Rain handling	Great	Great	Great	Good	Ok	Good	Ok
Movement handling	Good	Great	Great	Good	Ok	Good	Ok

Table 5.1: A summary of the evaluated encoders.

5.7 Threats to validity

Due to time constraints and unforeseen data-collection issues, only two ML-algorithms were chosen. Unfortunately, this impacts the generalisability of this study somewhat due to the small amount of ML-algorithms tested and only one type of AD ML-task(Pixel-Level Semantic Labeling) is evaluated. The effects of video compression artefacts might vary between algorithms performing the Pixel-Level Semantic Labeling task and the results of this study might not reflect the performance of each algorithm for this task. Furthermore, the effects of video compression artefacts might vary between algorithms of different AD ML-tasks. The effects of video compression on Pixel-Level Semantic Labeling might be different from that of other AD ML-tasks. Furthermore, the ML-algorithms were not retrained with the compressed dataset before evaluation due to time and computational resource constraints. The ML-algorithms may have been able to maintain their performance better if the tested ML-algorithms would have trained compressed data. This means that some sweet

spots that were deemed to degrade significantly may not have a significant effect on HRNet and GSCNN networks trained on compressed data. The limited amount of experimental units and not retraining ML-algorithms might pose as a threat to the external validity for the study. However, the two ML-algorithms tested provide a valuable insight into how AD ML-algorithms could be affected and the study's findings regarding the encoders' effects on the tested ML-algorithms can be used by practitioners as a basis for selecting encoders and coding parameters for their own testing and validation.

All data collection for this thesis was performed by the two computer systems detailed in Tables 3.3 and 3.4. Differences and inconsistencies in the encoding and inference performed by these systems would pose as a threat to validity. If the two systems would produce different encoding and ML-performance results, given the save encoding parameters and using the same ML-models, the internal validity would be jeopardised. The internal validity of the experiments would also be harmed if any system would inhibit inconsistent results throughout the data collection. However, both systems ran the same sets of Docker containers for encoding and performing ML-evaluation. Furthermore, there were no differences in performed hardware-acceleration capabilities between the systems with both systems having graphics cards with the Turing NVENC engine and Quick Sync only being performed by the Intel system. Both systems produced the same deterministic results with regards to both encoding and ML-evaluation. Hence, the usage of multiple computer systems was not deemed to invalidate the data collected during this thesis.

The Intel accelerated Intel encoder for VP9 did not function properly for the CQP parameter rate control. With CQP, the VP9 encoder might have been able to perform might impact the internal validity due to a sub-optimal choice of rate control. Hence, this fact is taken into consideration in conclusions and recommendations made regarding the encoder.

The encoders tested in this thesis were chosen due to their popularity in the field, performance and hardware support on the computer systems available for data collection. The capability of both software and hardware encoders for the same video coding format can, as seen in this report, vary significantly. There are many software and hardware encoders for the video coding formats that practitioners can use to compress data. The tested encoders represent how well a video coding format can perform, but there is a limitation to the generalisability of the thesis' findings. Practitioners, that consider encoders that were not evaluated in this thesis, will have to perform their own evaluation to verify the encoders' capabilities.

The optimality of coding parameter sweet spots is important for the validity of the study. If the sweet spots for one encoder are not optimal, then they will not be representative of the best performance that the encoder can achieve. Hence, the validity of comparisons between the encoders will be threatened if one or several encoders' performance is not properly represented. Due to time constraints, it was necessary to use an optimisation approach instead of a full or fractional factorial design. However, the validity of the coding parameter optimisation might be threatened by not performing optimisation with larger populations and more generations. Results and conclusions drawn from the experiment might be altered if the optimisation algorithm converges at a local optimum and not at a set of encoder settings that

are globally optimal. Tests showed that the optimisation consistently reached the same level of optimality during test optimisation runs, but there is still a possibility that coding parameter sweet spots found were not globally optimal, which could undermine conclusions and recommendations made regarding the encoders.
Pre-trained networks were used to achieve a state of the art level of performance for the ML-algorithms. The network used for HRNet was trained and validated on the training slice of the Cityscapes dataset. However, for GSCNN the network was trained on the training slice and validated on the validation slice of the dataset. This means that the GSCNN network was already familiar to the data that was used to optimise and evaluate the dataset. Hence, the validity of comparisons between the two ML-algorithms' susceptibility to compression artefacts could be affected. However, this is a fact that was taken into consideration when analysing the results. Data analysis focused on how the encoders compare against each other for each ML-algorithm and not directly comparing each ML-algorithms susceptibility to data degradation.
The validation set used for GSCNN was reduced, as explained in Section 3.1.3. The use of a reduced set of instances could have influenced the coding parameter optimisation for GSCNN. There is a risk that the optimisation became specific for the reduced set of images and was not generally optimised for all possible instances. The instances of the reduced set were randomly selected from the full validation set. However, to balance the selection between cities, the random selection was intentionally set up to chose 59 instances from each city. But there is still a chance that the instances selected were unintentionally biased towards certain situations, e.g. having more city traffic and less highway traffic. Hence, there is a possibility that the sweet spot candidates for GSCNN are skewed for certain situations. However, due to each encoder being optimised on the reduced dataset, comparison between encoder results should be valid due to the encoders being optimised under the same conditions.
The field of AD is surrounded by ethical dilemmas regarding vehicle manoeuvring during accidents. The video compression might have unforeseen effects on the inference in the form of bias and unexpected performance degradation. Application of video compression, on training data or a live video feed used for inference, might alter ML-algorithms' decision-making during critical situations. Skin-colour, age (child vs adult) and clothing are examples of an individuals' attributes that might be affected to different extents by video compression. This thesis did not involve analysis on how representative the results are for all people and if any groups were potentially susceptible to compression artefacts that cause a decline in ML-performance.

6
Conclusion

During this thesis, the effects of degraded visual quality on autonomous driving tasks were studied to establish the capabilities of standard lossy compression methods. The conducted research expand the body of knowledge in the field of computer vision to include the capabilities for lossy video compression and recommendations for practitioners. In order to configure each encoder to perform optimally, sets of optimum coding parameters were extracted through multi-objective optimisation for each encoder. It was found that the tested dataset could be, in some cases, compressed such that the dataset size was reduced while the ML-performance was maintained.

An assumption for the dependent t-test for this study is a normal distribution of the ML-performance error between uncompressed and compressed datasets, which some sweet spots did not have. This caused some sweet spots to be excluded from being tested. The dependent t-tests, performed on the sweet spots that fulfilled the normality assumption, showed that there was at least one sweet spot that could maintain a performance level that did not significantly differ from that of the uncompressed dataset. Contrary to hypothesis 1, not every encoder was able to compress the dataset without significantly impacting the ML-performance. However, the evaluation was performed on ML-algorithms that have trained on uncompressed data. Training on a compressed dataset or using data augmentation that applies compression may result in ML-algorithms becoming less prone to be affected by video compression artefacts.

A clear difference between the encoder's achieved ML-performance and compression ratio could be observed, with AV1 and HEVC encoders emerging as the most prominent candidates. The HEVC encoders tested have the benefit of being hardware-accelerated, which gives them an advantage when it comes to encoding time. The tested AV1 encoder demonstrated the potential of AV1 as a future prominent coding format. AV1 is currently not a very mature coding format but may in time become an established and robust coding format with hardware-acceleration support. The performance of libx264 during the experiments shows why it is such a well-regarded encoder. libx264 can at times keep up with the HEVC encoders, but the encoding resources required in relation to the hardware-accelerated HEVC encoders makes libx264 a suboptimal choice for real-world deployment. No compelling arguments can be made to recommend the hardware-accelerated H.264 encoders over their HEVC counterparts. The evaluation of sweet spot candidates revealed that their spread of ML-performance error between uncompressed and compressed datasets could differ to a large extent, both within and between encoders. With regards to spread of such errors, the VP9 sweet spots stood out as generally having a broader

spread of errors than the other encoders and libx264, NVIDIA accelerated H.264 and AV1 generally having lower spreads. The performance of VP9, which was not impressive during testing, may have been able to fare better, given that alternate rate control methods were functioning and used during the experiments.
The degraded datasets affected the ML-performance differently. The ML-performance was generally improved, for the dataset representing granular noise degradation, with the use of video compression that suppressed the noise. However, the datasets with added rain and movement effects saw video compression degrade the ML-performance achieved at a given compression ratio.
The use of encoders, with a good set of coding parameters and sufficient data rate budget, could significantly reduce the size of datasets while maintaining the inference performance of ML-algorithms. An important note to practitioners is to, before deployment of an encoder, validate its impacts on training and inference data.

6.1 Future Works

This study did not investigate how ML-algorithms that have trained on data with compression artefacts cope with compressed input. Future research could investigate how prominent encoders and sweet spots affect ML-algorithms using cross-validation. However, due to the computationally heavy nature of such work, a prerequisite for such studies is to have access to enough compute resources. Such studies might also require hand-picked sets of coding parameters to evaluate instead of using a lengthy optimisation process. Future studies that require encoder parameter optimisation are strongly recommended to limit the number of encoders and coding parameters used and limit the size of the dataset used during the optimisation.

Bibliography

[1] CB Insight, *40+ corporations working on autonomous vehicles*, https://www.cbinsights.com/research/autonomous-driverless-vehicles-corporations-list/, (visited on 17-10-2019). [Online].

[2] C. Gutierrez, *What it takes to build and train neural networks for autonomous vehicles*, Jun. 22, 2018. [Online]. Available: https://www.altoros.com/blog/what-it-takes-to-build-and-train-neural-networks-for-autonomous-vehicles/ (visited on 01/15/2020).

[3] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving", *Journal of Field Robotics*, Nov. 2019, ISSN: 1556-4967. DOI: 10.1002/rob.21918. [Online]. Available: http://dx.doi.org/10.1002/rob.21918.

[4] K. Sayood, "Chapter 1 - introduction", in *Introduction to Data Compression (Fifth Edition)*, ser. The Morgan Kaufmann Series in Multimedia Information and Systems, K. Sayood, Ed., Fifth Edition, Morgan Kaufmann, 2018, pp. 4–5, ISBN: 978-0-12-809474-7. DOI: https://doi.org/10.1016/B978-0-12-809474-7.00001-X.

[5] M. Uhrina, J. Frnda, L. Sevcik, and M. Vaculík, "Impact of h.264/avc and h.265/hevc compression standards on the video quality for 4k resolution", *Advances in Electrical and Electronic Engineering*, vol. 12, pp. 372–373, Dec. 2014. DOI: 10.15598/aeee.v12i4.1216.

[6] A. Aaron, Z. Li, M. Manohara, J. D. Cock, and D. Ronca. (Apr. 19, 2017). Per-title encode optimization. (visited on 26-10-2020), [Online]. Available: https://medium.com/netflix-techblog/per-title-encode-optimization-7e99442b62a2.

[7] K. Sayood, "Chapter 19 - video compression", in *Introduction to Data Compression (Fifth Edition)*, ser. The Morgan Kaufmann Series in Multimedia Information and Systems, K. Sayood, Ed., Fifth Edition, Morgan Kaufmann, 2018, p. 689, ISBN: 978-0-12-809474-7. DOI: https://doi.org/10.1016/B978-0-12-809474-7.00019-7.

[8] Netflix, *Netflix research: Video encoding & quality*, https://research.netflix.com/articles, (visited on 14-10-2019). [Online].

[9] B. Adsumilli, S. Benting, C. Chen, A. Kokaram, and Y.-C. Lin, *Making high quality video efficient*, https://youtube-eng.googleblog.com/2018/04/making-high-quality-video-efficient.html, (visited on 26-10-2019). [Online].

[10] Chalmers University Of Technology, *Resource for vehicle research at chalmers*, https://www.chalmers.se/en/researchinfrastructure/revere/Pages/default.aspx, (visited on 20-01-2020). [Online], Nov. 5, 2019.

[11] C. Chen, Y. Lin, A. C. Kokaram, and S. Benting, "Encoding bitrate optimization using playback statistics for http-based adaptive video streaming", *CoRR*, vol. abs/1709.08763, 2017. arXiv: 1709.08763. [Online]. Available: http://arxiv.org/abs/1709.08763.

[12] T. Nazaré, G. De Barros Paranhos da Costa, W. Contato, and M. Ponti, "Deep convolutional neural networks and noisy images", in. Jan. 2018, pp. 416–424, ISBN: 978-3-319-75192-4. DOI: 10.1007/978-3-319-75193-1_50.

[13] YouTube, *Engineering and developers blog*, https://youtube-eng.googleblog.com, (visited on 13-01-2020). [Online].

[14] E. Laurin and J. Eberlén, "Performance analysis of large-scale realtime video stream configurations", Nov. 2019. [Online]. Available: http://hdl.handle.net/2077/62540.

[15] S. F. Dodge and L. J. Karam, "Understanding how image quality affects deep neural networks", *CoRR*, vol. abs/1604.04004, 2016. arXiv: 1604.04004. [Online]. Available: http://arxiv.org/abs/1604.04004.

[16] Z. Liu, T. Liu, W. Wen, L. Jiang, J. Xu, Y. Wang, and G. Quan, "Deepn-jpeg: A deep neural network favorable jpeg-based image compression framework", *CoRR*, vol. abs/1803.05788, 2018. arXiv: 1803.05788. [Online]. Available: http://arxiv.org/abs/1803.05788.

[17] T. Hase, W. Hintermaier, A. Frey, T. Strobel, U. Baumgarten, and E. Steinbach, "Influence of image/video compression on night vision based pedestrian detection in an automotive application", in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, May 2011, pp. 1–5. DOI: 10.1109/VETECS.2011.5956649.

[18] F. A. Andaló, O. A. B. Penatti, and V. Testoni, "Transmitting what matters: Task-oriented video composition and compression", in *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, Oct. 2016, pp. 72–79. DOI: 10.1109/SIBGRAPI.2016.019.

[19] L. Galteri, M. Bertini, L. Seidenari, and A. Del Bimbo, "Video compression for object detection algorithms", in *2018 24th International Conference on Pattern Recognition (ICPR)*, Aug. 2018, pp. 3007–3012. DOI: 10.1109/ICPR.2018.8546064.

[20] L. Pearlstein, I. Patel, and A. J. Aved, "Video coding layer optimization for target detection", in *2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, Oct. 2016, pp. 1–7. DOI: 10.1109/AIPR.2016.8010597.

[21] E. Kafetzakis, C. Xilouris, M. Kourtis, M. Nieto, I. Jargalsaikhan, and S. Little, "The impact of video transcoding parameters on event detection for surveillance systems", Dec. 2013, pp. 333–338. DOI: 10.1109/ISM.2013.64.

[22] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar, "Performance comparison of h.265/mpeg-hevc, vp9, and h.264/mpeg-avc encoders", in *2013 Picture Coding Symposium (PCS)*, Dec. 2013, pp. 394–397. DOI: 10.1109/PCS.2013.6737766.

[23] S. Winkler and C. Faller, "Audiovisual quality evaluation of low-bitrate video", *Proceedings of SPIE - The International Society for Optical Engineering*, Mar. 2005. DOI: 10.1117/12.596852.

[24] J. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[25] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering", *Structural and Multidisciplinary Optimization*, vol. 26, pp. 369–395, Apr. 2004. DOI: https://doi.org/10.1007/s00158-003-0368-6.

[26] F. Al-Abri, X. Li, E. A. Edirisinghe, and C. Grecos, "Multi-objective performance optimization of h.264 avc encoder", in *2009 IEEE/SP 15th Workshop on Statistical Signal Processing*, Aug. 2009, pp. 725–728. DOI: 10.1109/SSP.2009.5278457.

[27] M. Al-Barwani and E. A. Edirisinghe, "A machine learning based framework for parameter based multi-objective optimisation of a h.265 video codec", in *2016 Future Technologies Conference (FTC)*, Dec. 2016, pp. 553–559. DOI: 10.1109/FTC.2016.7821661.

[28] D. B. Jourdan and O. L. de Weck, "Layout optimization for a wireless sensor network using a multi-objective genetic algorithm", in *2004 IEEE 59th Vehicular Technology Conference. VTC 2004-Spring (IEEE Cat. No.04CH37514)*, vol. 5, May 2004, 2466–2470 Vol.5. DOI: 10.1109/VETECS.2004.1391366.

[29] GitHub, Inc, *Github*, https://github.com/, (visited on 22-10-2019). [Online].

[30] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang, *High-resolution representations for labeling pixels and regions*, 2019. arXiv: 1904.04514 [cs.CV].

[31] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, *The cityscapes dataset for semantic urban scene understanding*, 2016. arXiv: 1604.01685 [cs.CV].

[32] Cityscapes Dataset, *Benchmark suite*, https://www.cityscapes-dataset.com/benchmarks/, (visited on 26-November-2019). [Online].

[33] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, "Gated-scnn: Gated shape cnns for semantic segmentation", *ICCV*, 2019.

[34] E. Tiu. (Sep. 10, 2019). Metrics to evaluate your semantic segmentation model, [Online]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2 (visited on 01/19/2020).

[35] Y. Yuan, X. Chen, and J. Wang, *Hrnet-segmentation model: "hrnetv2-w48+ocr"*, https://github.com/HRNet/HRNet-Semantic-Segmentation/tree/HRNet-OCR#segmentation-models, (website visited and model downloaded on 28-1-2020). [Online].

[36] ——, *Gscnn-segmentation "pretrained model"*, https://github.com/nv-tlabs/GSCNN, (website visited and model downloaded on 16-2-2020). [Online].

[37] Bitmovin, *Video developer report 2018*, https://go.bitmovin.com/hubfs/Bitmovin-Video-Developer-Report-2018.pdf, (visited on 18-01-2020). [Online].

[38] ——, *Best video codec: An evaluation of av1, avc, hevc and vp9*, https://bitmovin.com/av1-multi-codec-dash-dataset/, (visited on 18-01-2020). [Online].

[39] NVIDIA Corporation, *Nvidia® video codec sdk*, https://developer.nvidia.com/nvidia-video-codec-sdk, (visited on 18-01-2020). [Online].

[40] Intel Corporation, *Intel® quick sync video*, https://www.intel.com/content/www/us/en/architecture-and-technology/quick-sync-video/quick-sync-video-general.html, (visited on 18-01-2020). [Online].

[41] FFMPEG, *Libaom av1*, https://trac.ffmpeg.org/wiki/Encode/AV1, (visited on 20-1-2020). [Online].

[42] Open Visual Cloud, *Scalable video technology for av1*, https://github.com/OpenVisualCloud/SVT-AV1, (visited on 10-2-2020). [Online].

[43] Pagmo development team, *Pygmo documentation*, https://esa.github.io/pagmo2/, (visited on 4-01-2020). [Online].

[44] ——, *Nsga-ii algorithm - pygmo documentation*, https://esa.github.io/pagmo2/docs/python/algorithms/py_algorithms.html, (visited on 4-01-2020). [Online].

[45] K. Kroening, *Ffmpeg-python*, https://github.com/kkroening/ffmpeg-python, (visited on 4-01-2020). [Online].

[46] U. Saxena, *Automold - specialized augmentation library for autonomous vehicles*, https://towardsdatascience.com/automold-specialized-augmentation-library-for-autonomous-vehicles-1d085ed1f578, (visited on 28-3-2020). [Online], May 3, 2018.

A

Appendix A

A.1 Boxplots of collected situation data

These boxplots include sweetspots in the range of 0-100 compression ratio to increase readability of plots.

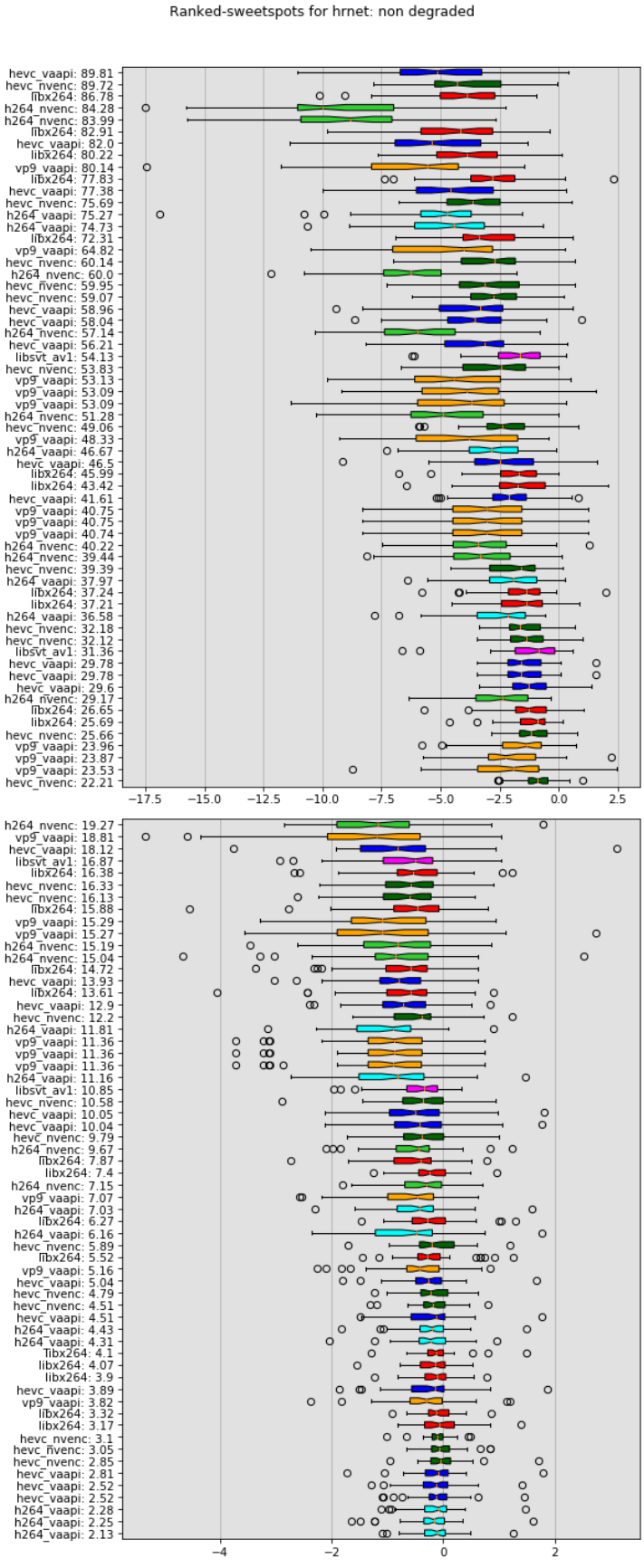


Figure A.1: Sweetspot candidates' HRNet ML-performance errors on the non degraded dataset, illustrated using boxplots.

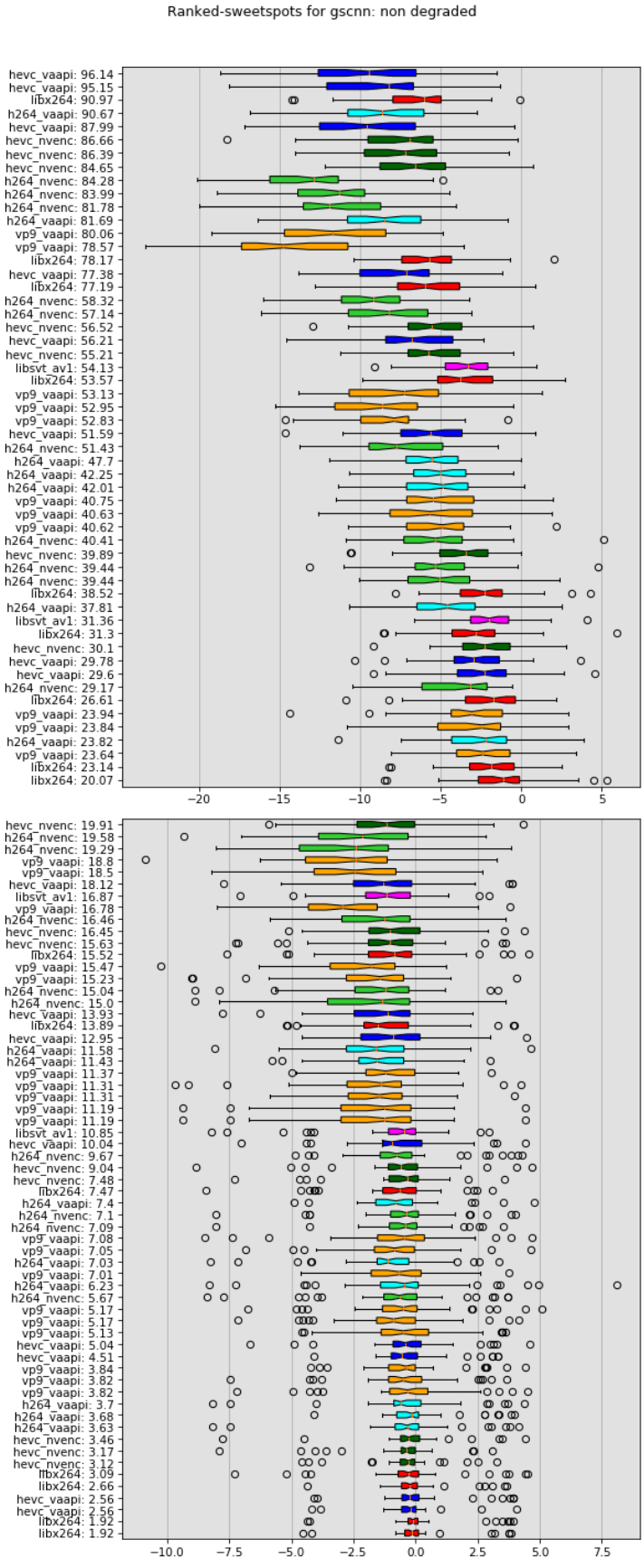


Figure A.2: Sweetspot candidates' GSCNN ML-performance errors on the non degraded dataset, illustrated using boxplots.

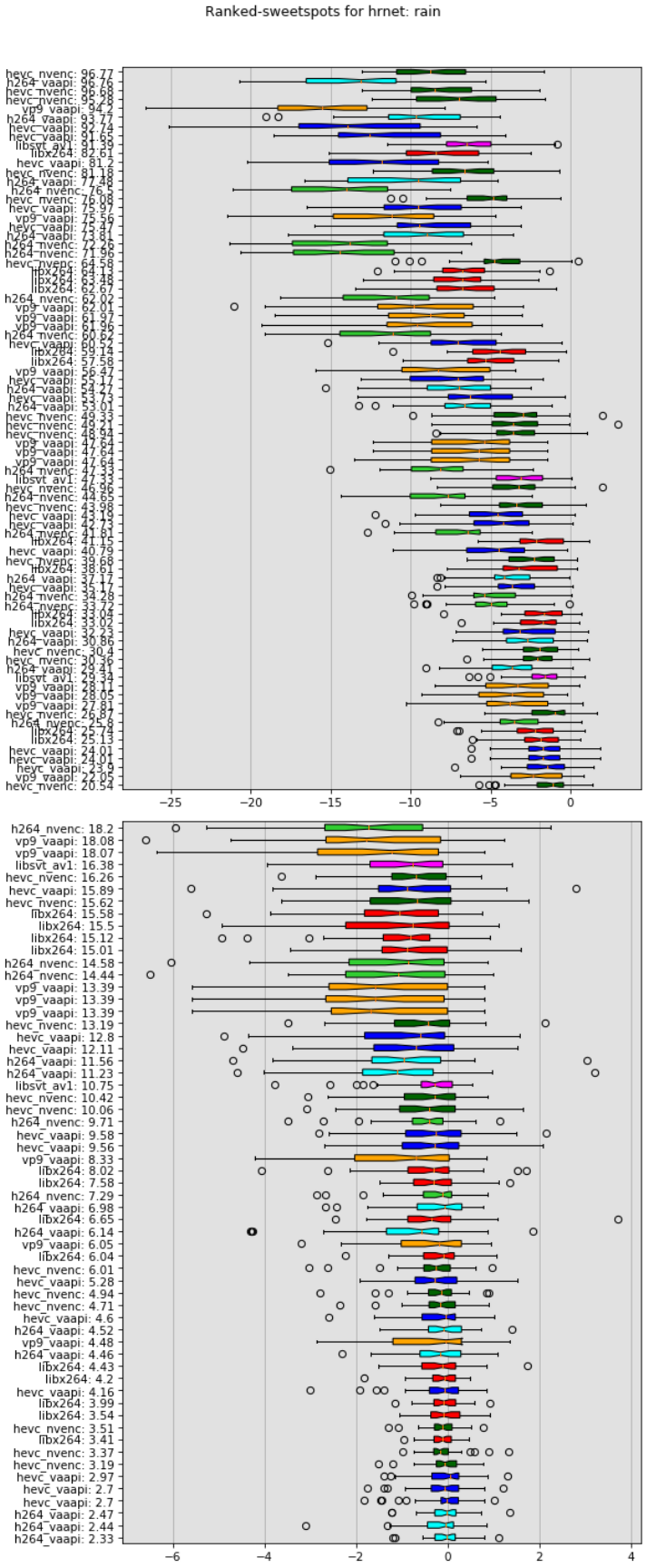


Figure A.3: Sweetspot candidates' HRNet ML-performance errors on the rainy dataset, illustrated using boxplots.

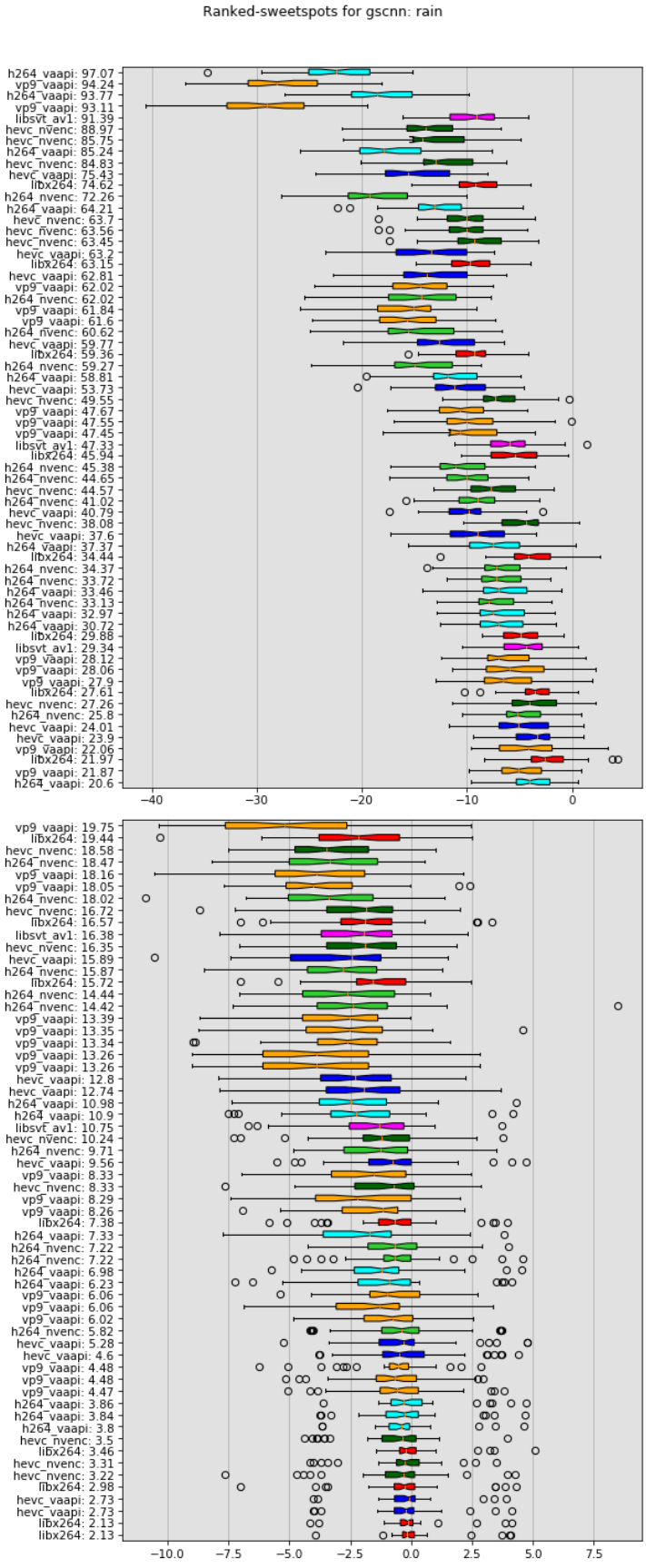


Figure A.4: Sweetspot candidates' GSCNN ML-performance errors on the rainy dataset, illustrated using boxplots.

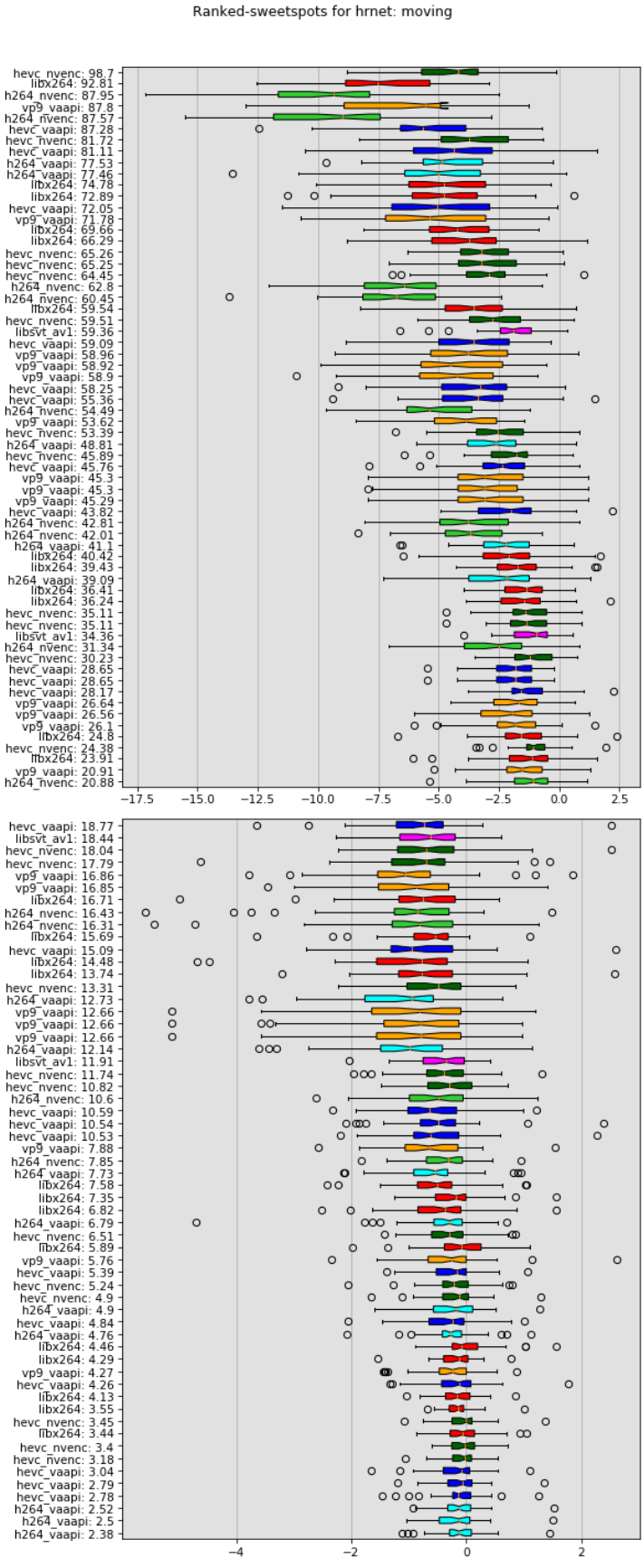


Figure A.5: Sweetspot candidates' HRNet ML-performance errors on the movement dataset, illustrated using boxplots.

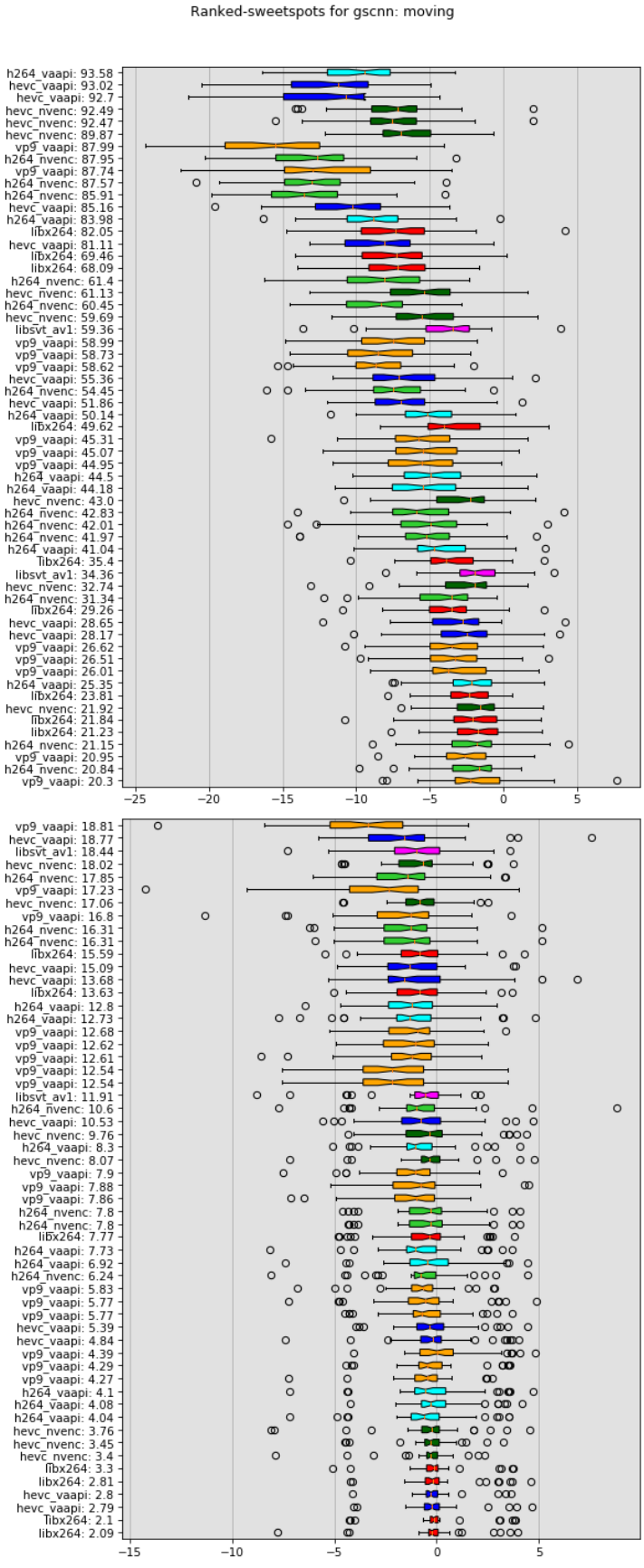


Figure A.6: Sweetspot candidates' GSCNN ML-performance errors on the movement dataset, illustrated using boxplots.

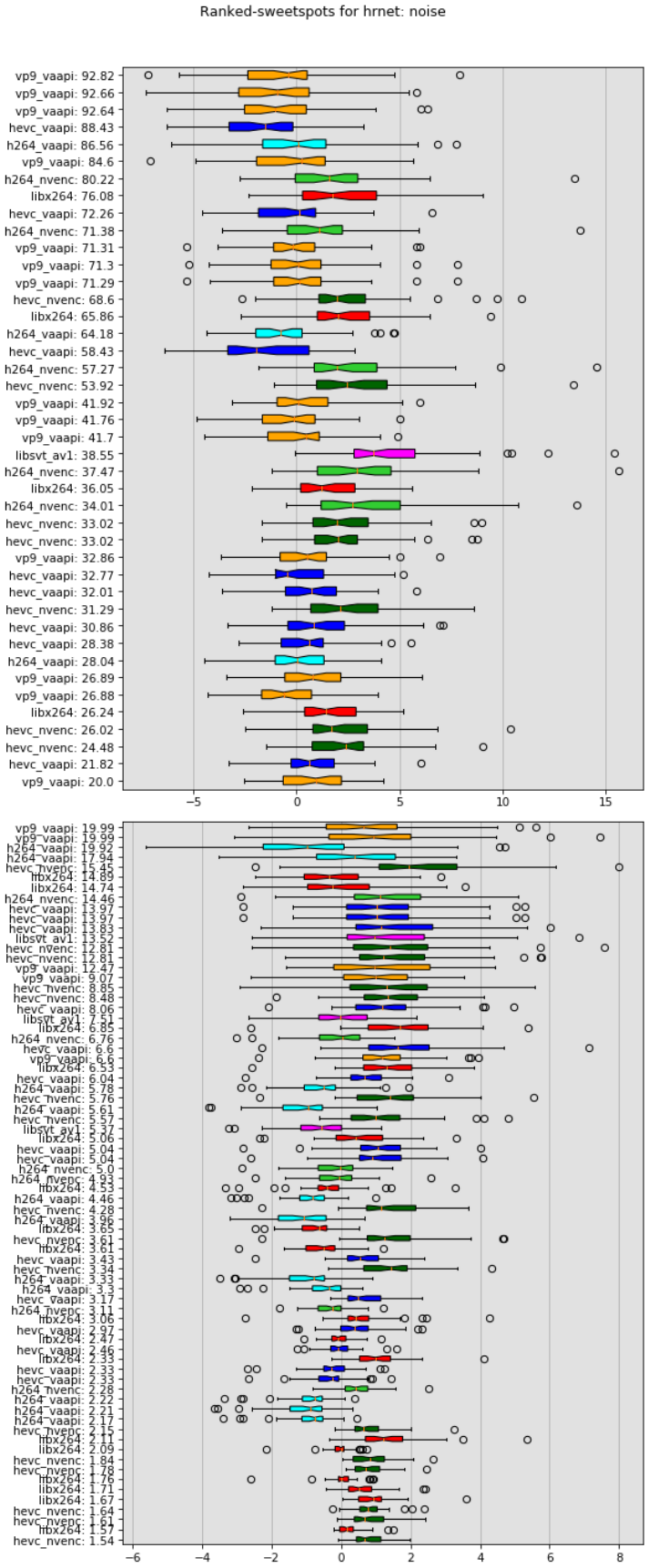


Figure A.7: Sweetspot candidates' HRNet ML-performance errors on the noisy dataset, illustrated using boxplots.

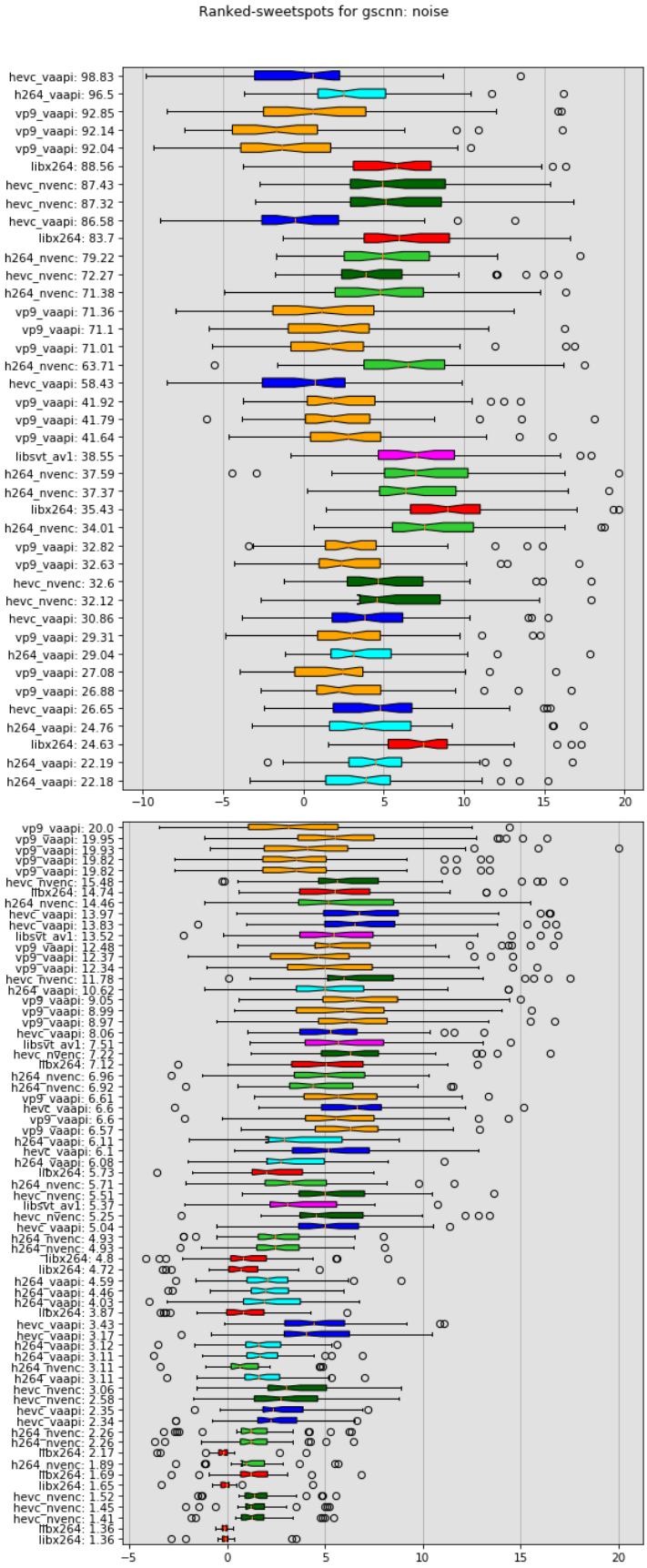
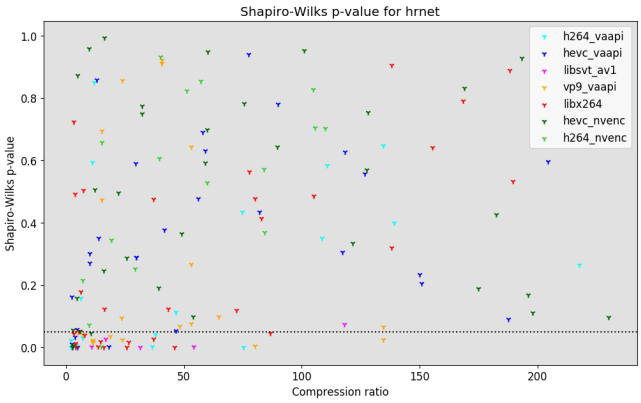
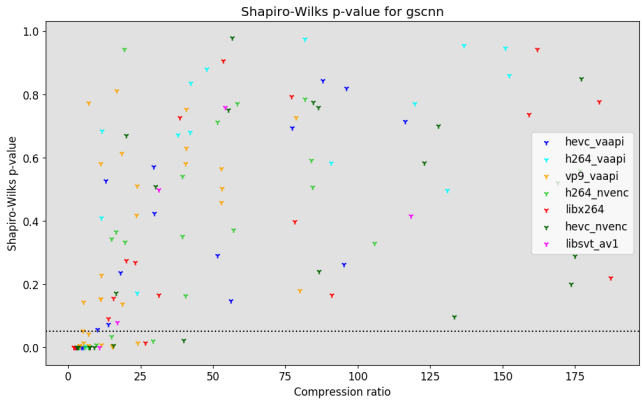


Figure A.8: Sweetspot candidates' GSCNN ML-performance errors on the noisy dataset, illustrated using boxplots.

A.2 Shapiro-Wilk test results

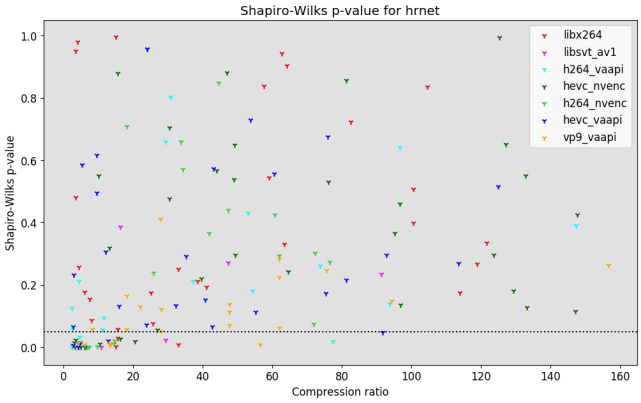


(a) HRNet Shapiro-Wilk results

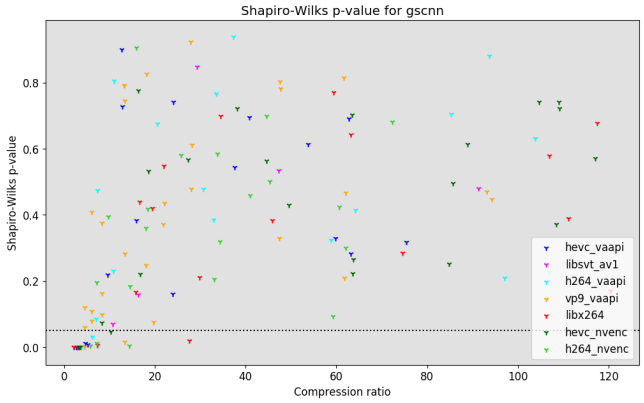


(b) GSCNN Shapiro-Wilk results

Figure A.9: Plots of mean ML-performance and the p-value from the Shapiro-Wilk test for the sweetspot candidates compressing the non degraded dataset.

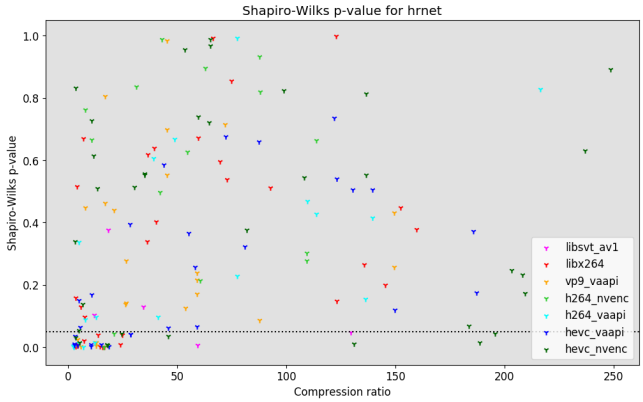


(a) HRNet Shapiro-Wilk results

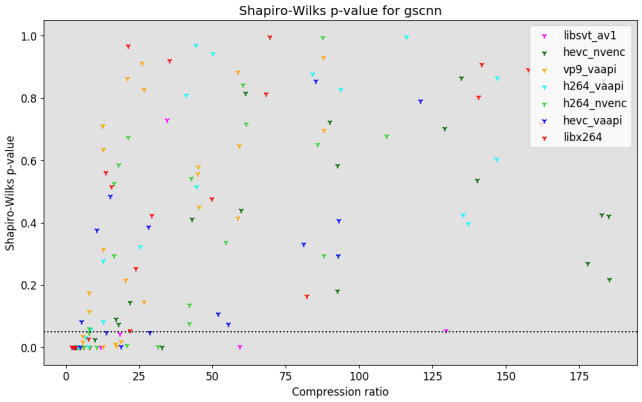


(b) GSCNN Shapiro-Wilk results

Figure A.10: Plots of mean ML-performance and the p-value from the Shapiro-Wilk test for the sweetspot candidates compressing the rainy dataset.

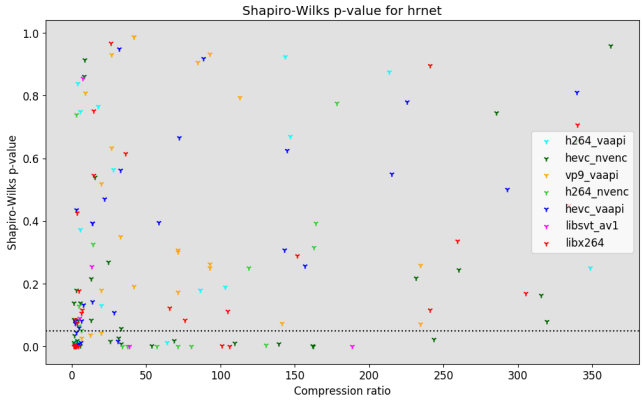


(a) HRNet Shapiro-Wilk results

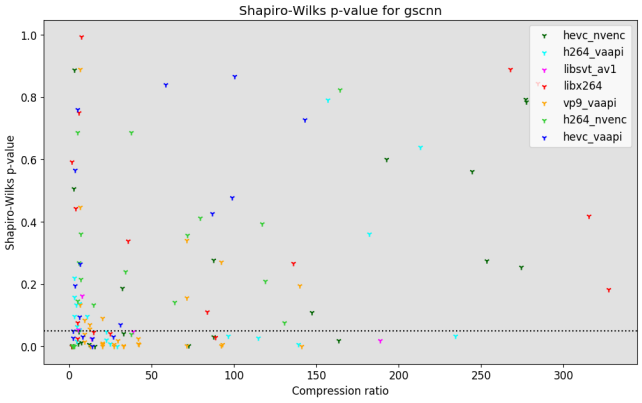


(b) GSCNN Shapiro-Wilk results

Figure A.11: Plots of mean ML-performance and the p-value from the Shapiro-Wilk test for the sweetspot candidates compressing the dataset with simulated movement.



(a) HRNet Shapiro-Wilk results



(b) GSCNN Shapiro-Wilk results

Figure A.12: Plots of mean ML-performance and the p-value from the Shapiro-Wilk test for the sweetspot candidates compressing the noisy dataset.

A.3 Encoding times

The Revere computer, stated in Table 3.3, performed all optimisation performed on GSCNN. The encoding times for each evaluated set of encoding parameters was recorded and is show in boxplot A.13.

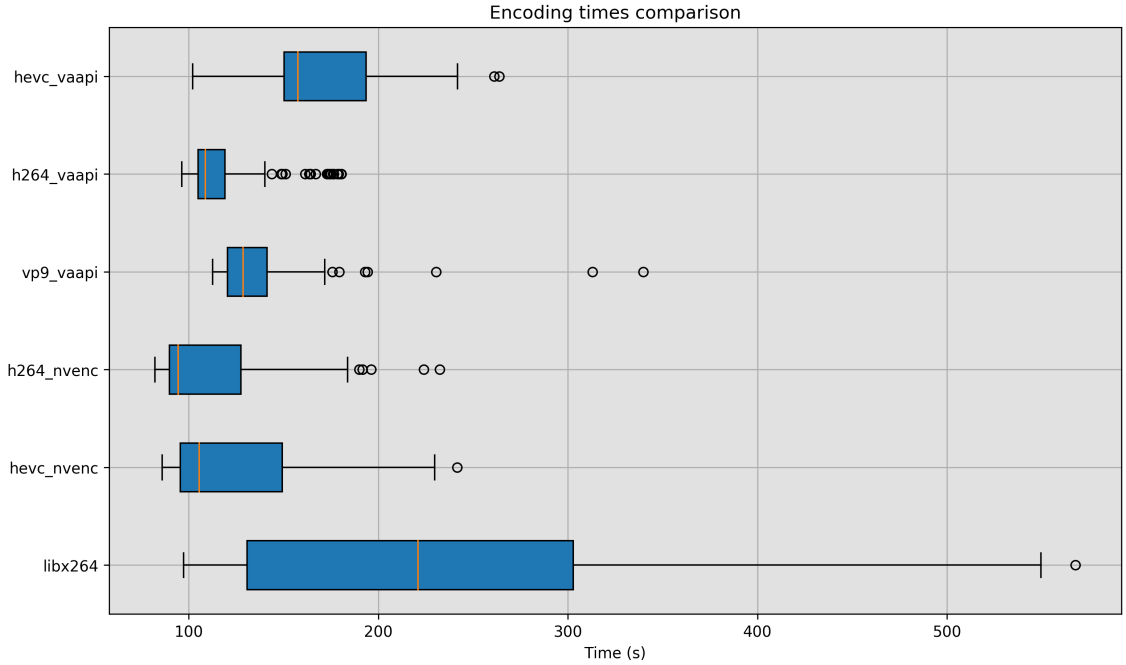


Figure A.13: A boxplot of the encoding times for each encoder.

B

Appendix B

B.1 Coding parameters optimised

Parameter	Bounds	Comments
qp	[1 - 40]	Constant quantization parameter rate control method
rc_mode	"CQP"	Rate control mode
coder	"ac" "vlc"	Entropy encoder
compression-level	"1" "4" "7"	Speed versus quality tradeoff (higher values are faster / worse quality)
g	[50 - 1000]	Maximum distance between I-frames
sc_threshold	[0 - 60]	Adjusts the sensitivity of x264's scenecut detection
bf	[1 - 16]	Limit to the max number of B-frames
refs	[1 - 16]	How many references can be used
b_depth	[0 - 5]	Maximum B-frame reference depth

Table B.1: The coding parameters used in the `h264_vaapi` optimisation.

Parameter	Bounds	Comments
qp	[1 - 40]	Constant quantization parameter rate control method
rc_mode	"CQP"	Rate control mode
compression-level	"1" "4" "7"	Speed versus quality tradeoff (higher values are faster / worse quality)
g	[50 - 1000]	Maximum distance between I-frames
sc_threshold	[0 - 60]	Adjusts the sensitivity of x264's scenecut detection
bf	[1 - 16]	Limit to the max number of B-frames
refs	[1 - 16]	How many references can be used
b_depth	[0 - 5]	Maximum B-frame reference depth

Table B.2: The coding parameters used in the `hevc_vaapi` optimisation.

Parameter	Bounds	Comments
bitrate	[1 - 150]	Set target bitrate
bufratio	[0.25 - 4]	Buffersize in relation to bitrate
rc_mode	"CBR" "VBR"	Rate control mode
compression-level	"1" "4" "7"	Speed versus quality tradeoff (higher values are faster / worse quality)
g	[50 - 1000]	Maximum distance between I-frames
sc_threshold	[0 - 60]	Adjusts the sensitivity of x264's scenecut detection
bf	[1 - 16]	Limit to the max number of B-frames
refs	[1 - 16]	How many references can be used
qmin	[0 - 14]	Minimum quantizer value
qmax	[15 - 69]	Maximum quantizer value
loop_filter_-level	[0 - 63]	loop filter parameter
loop_filter_-sharpness	[0 - 15]	loop filter parameters

Table B.3: The coding parameters used in the `VP9_vaapi` optimisation.

Parameter	Bounds	Comments
crf	[1 - 40]	Quality for constant quality mode
aq-strenght	[0.1 - 1.9]	Sets strenght of blocking and blurring reduction in flat and textured areas
preset	"medium" "slow" "slower" "veryslow" "placebo"	Encoding presets
tune	"none" "stillimage" "film" "grain"	Presets for encoding param tuning
me_method	"dia" "hex" "umh" "esa" "tesa"	Motion estimation method
coder	"ac" "vlc"	Entropy encoder
cmp	"chroma" "sad"	Full pixel motion estimation comparison algorithm
aq-mode	"none" "variance" "auto-variance"	AQ method
weightp	"none" "simple" "smart"	Weighted prediction method for P-frames
b-pyramid	"none" "strict" "normal"	Method for keeping of some B-frames as references
direct-pred	"none" "spatial" "tempo-ral" "auto"	Direct MV prediction mode
deblock-alpha	[-2 - 2]	Loop filter parameter
deblock-beta	[-2 - 2]	Loop filter parameters
psy	[0 - 1]	Usage of psychovisual optimizations
weightb	[0 - 1]	Enable weighted prediction for B-frames
b_strategy	[0 - 2]	Adaptive B-frame placement decision algorithm
g	[50 - 1000]	Maximum distance between I-frames
keyint_min	[10 - 40]	Minimum distance between I-frames
intra-refresh	[0 - 1]	Periodic Intra Refresh instead of IDR frames
trellis	[0 - 2]	When CABAC is enabled, set usage of Trellis, which regulates optimal rounding choices for the maximum rate-distortion score.
sc_threshold	[0 - 60]	Adjusts the sensitivity of x264's scenecut detection
bf	[1 - 16]	Limit to the max number of B-frames
me_range	[12 - 24]	Max range of the motion search
subq	[1 - 9]	Determines algorithm to use for subpixel motion searching and partition decision
rc-lookahead	[20 - 60]	Number of frames to look ahead for frametype and ratecontrol
b-bias	[-100 - 100]	influences how often B-frames are used
mixed-refs	[0 - 1]	Enables the use of one reference per partition
8x8dct	[0 - 1]	allow choise between 8x8 and 4x4 frequency trans-form size

Table B.4: The coding parameters used in the `libx264` optimisation.

Parameter	Bounds	Comments
qp	[1 - 40]	Constant quantization parameter rate control method
preset	"fast" "medium" "slow" "hp" "hq" "bd"	Encoding preset
b_ref_mode	"Disabled" "Middle"	Use B frames as references
rc	"constqp"	Ratecontrol method
surfaces	[0 - 64]	Number of concurrent surfaces
nonref_p	[0 - 1]	Enable automatic insertion of non-reference P-frames
weighted_pred	[0 - 1]	Enable weighted prediction (but disables b-frames)
strict_gop	[0 - 1]	Minimizes GOP-to-GOP rate fluctuations

Table B.5: The coding parameters used in the `h264_nvenc` optimisation.

Parameter	Bounds	Comments
qp	[1 - 40]	Constant quantization parameter rate control method
preset	"fast" "medium" "slow" "hp" "hq" "bd"	Encoding preset
b_ref_mode	"Disabled" "Middle"	Use B frames as references
rc	"constqp"	Ratecontrol method
surfaces	[0 - 64]	Number of concurrent surfaces
nonref_p	[0 - 1]	Enable automatic insertion of non-reference P-frames
weighted_pred	[0 - 1]	Enable weighted prediction
strict_gop	[0 - 1]	Minimizes GOP-to-GOP rate fluctuations

Table B.6: The coding parameters used in the `hevc_nvenc` optimisation.

Parameter	Bounds	Comments
qp	"5" "8" "14" "23" "35"	Constant quantization parameter rate control method
rc	"CQP"	Rate control mode
preset	"4"	Encoding preset

Table B.7: The coding parameters used in the `libsvt_av1` optimisation.

B.2 Examples of degraded images

Figure B.1: Image examples of noisy dataset

Figure B.2: Image examples of rainy dataset

(a) 4th before label frame

(b) Label frame

(c) 4th after label frame

Figure B.3: Image examples of movement dataset

B.3 Examples of video compression artefacts

The images in the following figure show an example of how heavy video compression artefacts, of libx264 targeting a low bitrate, look like for non-degraded and degraded data.

(a) Non-degraded

(b) Movement-degraded

(c) Rain-degraded

(d) Noise-degraded

Figure B.4: Cropped image examples of video compression artefacts using lib264.

B.4 Optimisation population runs

3 runs with varying population sizes were run to test each the consistency of optimised coding parameters. The performance of these runs is HRNet's mIOU using a limited validation slice of the cityscapes dataset.

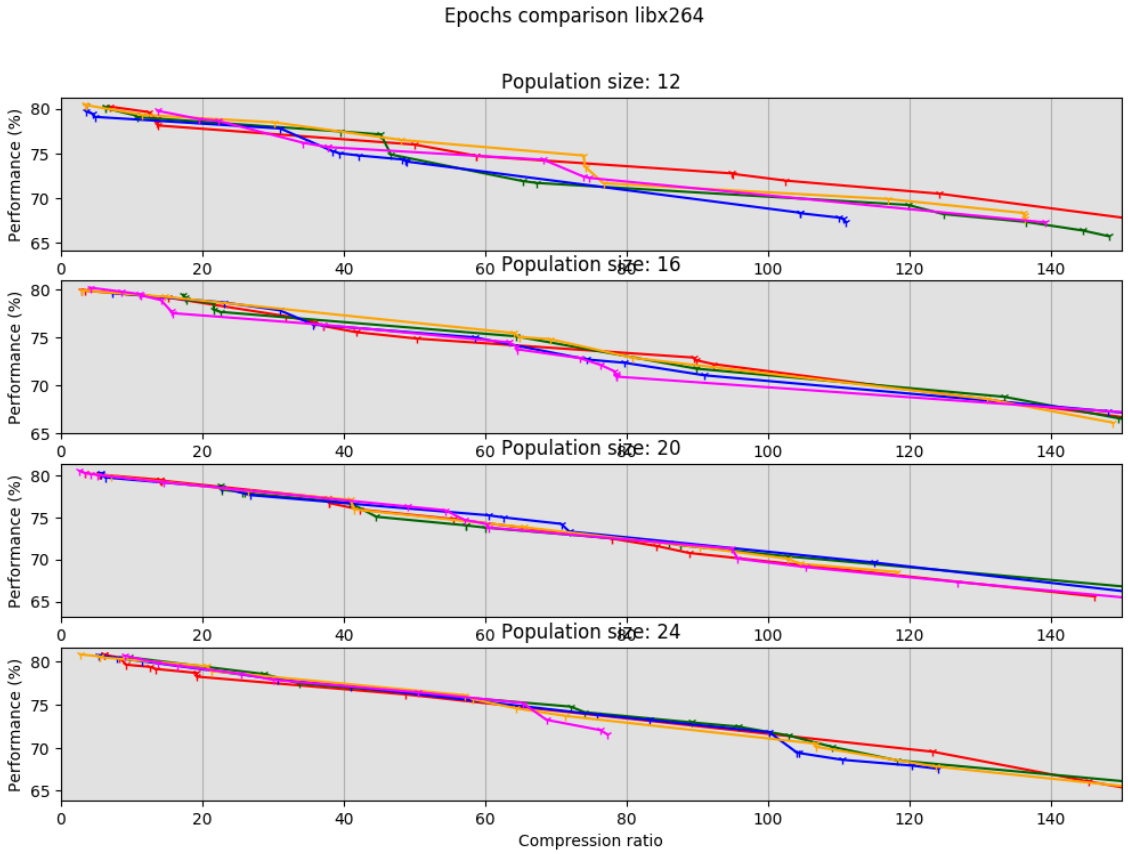


Figure B.5: Optimality of coding parameters of libx264 using on HRNet and a limited dataset

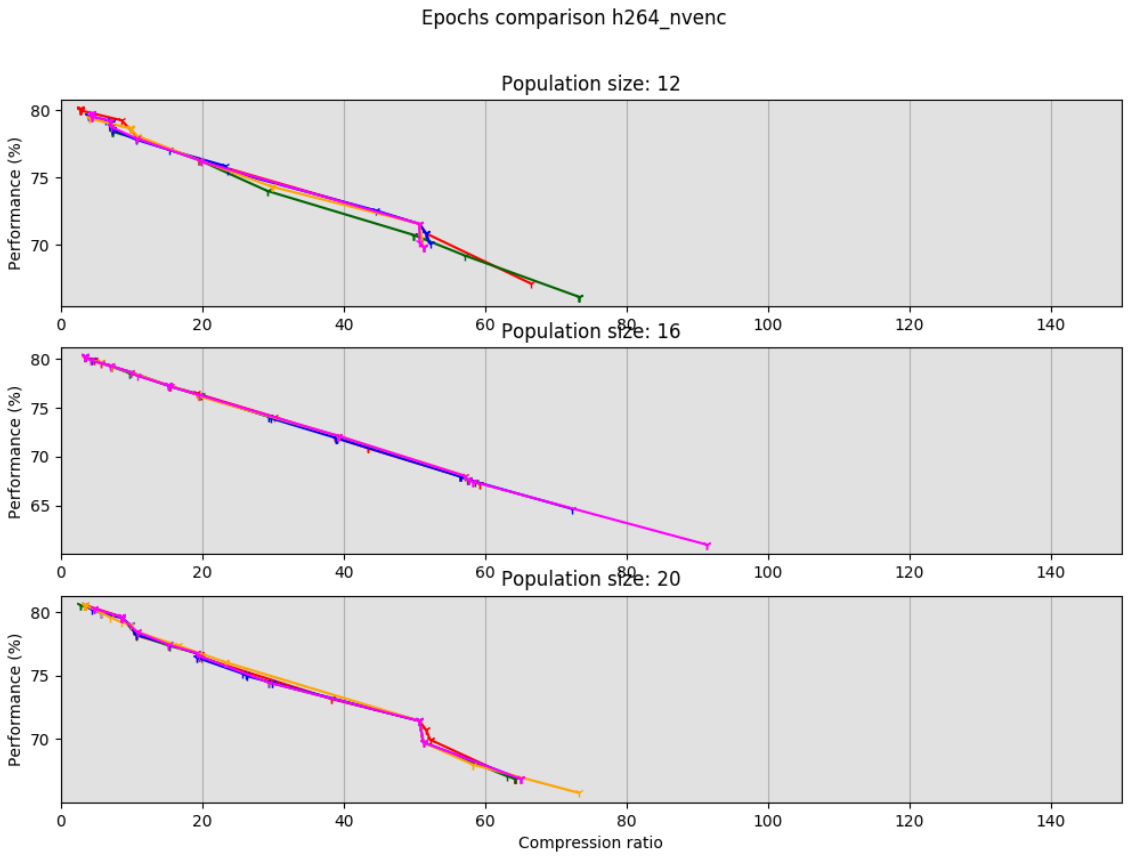


Figure B.6: Optimality of coding parameters of h264_nvenc using on HRNet and a limited dataset

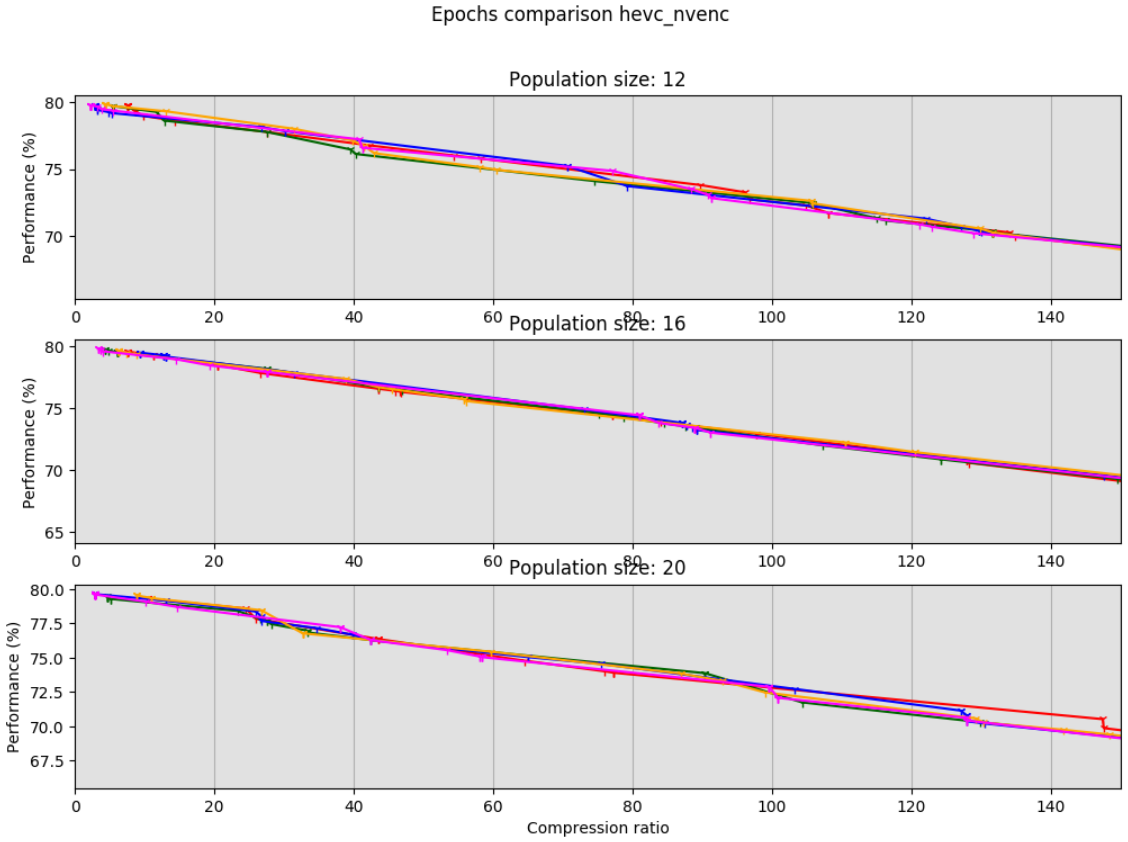


Figure B.7: Optimality of coding parameters of hevc_nvenc using on HRNet and a limited dataset

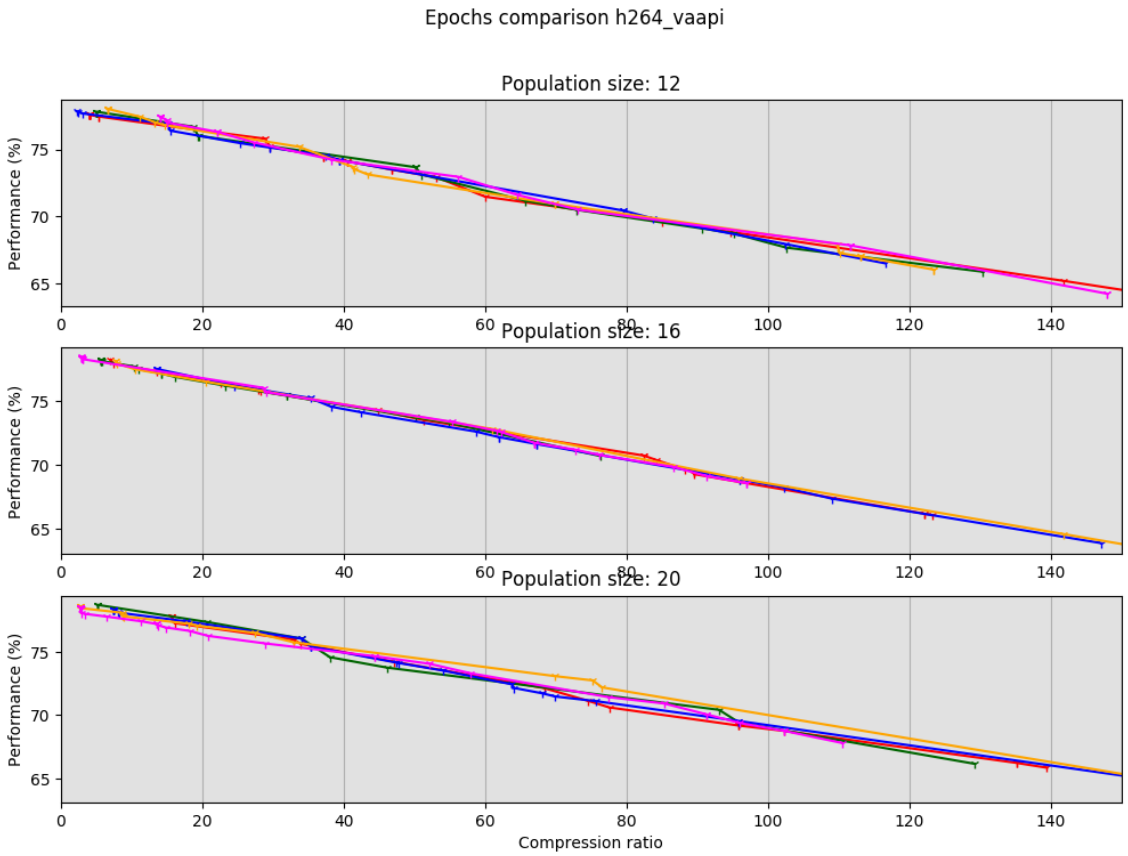


Figure B.8: Optimality of coding parameters of h264_vaapi using on HRNet and a limited dataset

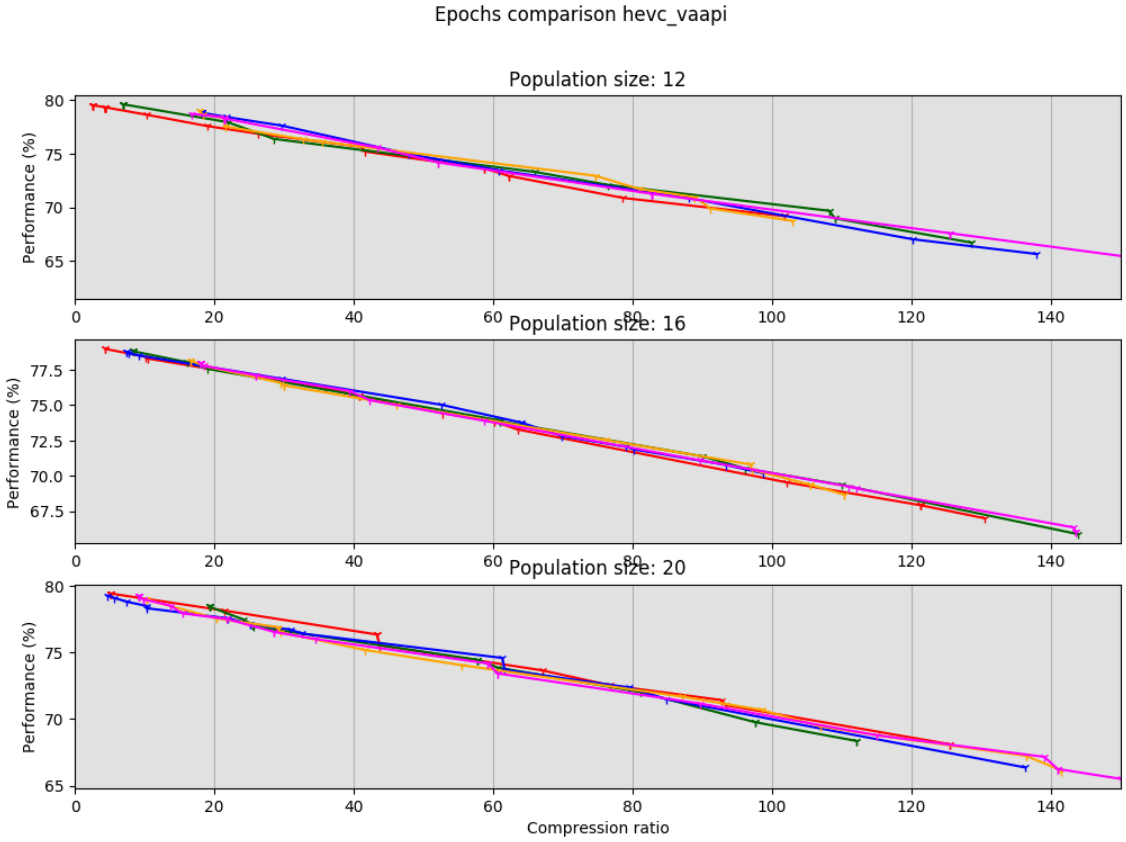


Figure B.9: Optimality of coding parameters of hevc_vaapi using on HRNet and a limited dataset

B.5 Sweetspot evaluation data

Pairs of tables for each encoder and ML-alg combination. One table contains the sweet spots found during the optimisation and the other table contains aggregated evaluation data collected for each sweet spot candidate. The compression ratio, error and PSNR values of the aggregated evaluation data table are mean values while the T-value is the single T-value calculated for a sweet spot.

Index	hevc_nvenc - Coding parameters (HRNet)
1	qp: 5.0, preset: fast, b_ref_mode: 2, surfaces: 40.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
2	qp: 5.0, preset: fast, b_ref_mode: 2, surfaces: 40.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
3	qp: 6.0, preset: fast, b_ref_mode: 0, surfaces: 44.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
4	qp: 6.0, preset: slow, b_ref_mode: 0, surfaces: 52.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
5	qp: 6.0, preset: slow, b_ref_mode: 2, surfaces: 8.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
6	qp: 8.0, preset: hq, b_ref_mode: 0, surfaces: 36.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 1.0
7	qp: 13.0, preset: hp, b_ref_mode: 2, surfaces: 28.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
8	qp: 13.0, preset: hp, b_ref_mode: 2, surfaces: 24.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 1.0
9	qp: 16.0, preset: fast, b_ref_mode: 0, surfaces: 40.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
10	qp: 16.0, preset: slow, b_ref_mode: 2, surfaces: 8.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
11	qp: 16.0, preset: fast, b_ref_mode: 2, surfaces: 20.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
12	qp: 18.0, preset: hp, b_ref_mode: 2, surfaces: 32.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
13	qp: 20.0, preset: slow, b_ref_mode: 2, surfaces: 20.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
14	qp: 20.0, preset: medium, b_ref_mode: 2, surfaces: 32.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 1.0
15	qp: 20.0, preset: slow, b_ref_mode: 2, surfaces: 20.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
16	qp: 23.0, preset: slow, b_ref_mode: 0, surfaces: 20.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
17	qp: 23.0, preset: hq, b_ref_mode: 0, surfaces: 54.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 0.0
18	qp: 25.0, preset: hp, b_ref_mode: 0, surfaces: 44.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
19	qp: 25.0, preset: fast, b_ref_mode: 0, surfaces: 44.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 0.0
20	qp: 25.0, preset: hp, b_ref_mode: 0, surfaces: 44.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 0.0
21	qp: 25.0, preset: hp, b_ref_mode: 0, surfaces: 40.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
22	qp: 26.0, preset: medium, b_ref_mode: 2, surfaces: 28.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
23	qp: 28.0, preset: slow, b_ref_mode: 2, surfaces: 20.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 1.0
24	qp: 28.0, preset: medium, b_ref_mode: 2, surfaces: 28.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
25	qp: 30.0, preset: slow, b_ref_mode: 0, surfaces: 50.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
26	qp: 30.0, preset: slow, b_ref_mode: 0, surfaces: 52.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 0.0
27	qp: 30.0, preset: slow, b_ref_mode: 0, surfaces: 20.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
28	qp: 33.0, preset: medium, b_ref_mode: 2, surfaces: 40.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
29	qp: 33.0, preset: medium, b_ref_mode: 2, surfaces: 20.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
30	qp: 33.0, preset: medium, b_ref_mode: 2, surfaces: 40.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
31	qp: 33.0, preset: medium, b_ref_mode: 2, surfaces: 40.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
32	qp: 33.0, preset: medium, b_ref_mode: 2, surfaces: 28.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
33	qp: 33.0, preset: medium, b_ref_mode: 2, surfaces: 28.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
34	qp: 35.0, preset: slow, b_ref_mode: 2, surfaces: 40.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
35	qp: 35.0, preset: bd, b_ref_mode: 0, surfaces: 46.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 1.0

Table B.8: The coding parameters of each sweet spot candidate of hevc_nvenc for HRNet.

hrnet Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	2.85	Fail	0.0	0.99	3.19	Fail	-0.06	0.99	3.18	1.44	-0.08	0.99	1.54	-11.34	0.78	0.99
2	3.05	0.45	-0.02	0.99	3.37	Fail	-0.11	0.99	3.4	0.72	-0.03	0.99	1.61	Fail	0.79	0.98
3	3.1	Fail	-0.1	0.99	3.51	Fail	-0.12	0.99	3.45	Fail	-0.03	0.99	1.64	-9.98	0.82	0.98
4	4.51	2.87	-0.18	0.98	4.71	Fail	-0.17	0.98	4.9	1.86	-0.14	0.98	1.78	Fail	0.84	0.98
5	4.79	3.01	-0.19	0.98	4.94	Fail	-0.27	0.98	5.24	Fail	-0.23	0.98	1.84	Fail	0.87	0.98
6	5.89	2.26	-0.17	0.98	6.01	Fail	-0.34	0.98	6.51	3.92	-0.29	0.98	2.15	Fail	0.8	0.96
7	9.79	4.19	-0.36	0.98	10.06	3.31	-0.48	0.98	10.82	4.03	-0.3	0.98	3.34	-9.26	1.39	0.92
8	10.58	Fail	-0.44	0.98	10.42	Fail	-0.47	0.98	11.74	3.96	-0.41	0.98	3.61	Fail	1.43	0.9
9	12.2	4.61	-0.44	0.98	13.19	3.91	-0.64	0.97	13.31	5.71	-0.58	0.98	4.28	-7.91	1.37	0.89
10	16.13	6.08	-0.69	0.98	16.26	Fail	-0.8	0.97	17.79	Fail	-0.79	0.98	5.57	Fail	1.21	0.84
11	16.33	5.98	-0.64	0.98	15.62	4.55	-0.77	0.97	18.04	Fail	-0.64	0.98	5.76	-6.84	1.47	0.84
12	22.21	7.3	-0.94	0.97	20.54	Fail	-1.47	0.97	24.38	Fail	-1.07	0.97	8.48	-7.11	1.46	0.8
13	25.66	7.75	-1.14	0.97	26.87	5.94	-1.38	0.97	30.23	6.87	-1.12	0.97	8.85	-5.09	1.41	0.81
14	32.12	8.05	-1.35	0.97	30.36	8.39	-2.07	0.97	35.11	7.66	-1.37	0.97	12.81	-5.49	1.63	0.78
15	32.18	9.06	-1.43	0.97	30.4	8.88	-2.05	0.97	35.11	7.87	-1.35	0.97	12.81	-5.68	1.59	0.78
16	39.39	9.87	-1.89	0.97	39.68	8.81	-2.39	0.96	45.89	Fail	-2.09	0.96	15.45	-6.67	2.24	0.77
17	49.06	9.91	-2.37	0.96	43.98	10.11	-3.35	0.96	53.39	10.62	-2.64	0.96	24.48	-6.76	2.44	0.75
18	53.83	10.11	-2.76	0.96	46.96	10.18	-3.59	0.96	59.51	12.02	-2.72	0.96	26.02	Fail	2.17	0.75
19	59.07	11.21	-2.86	0.96	48.94	10.99	-3.65	0.96	64.45	11.98	-3.04	0.96	31.29	Fail	2.42	0.74
20	59.95	11.0	-2.9	0.96	49.21	9.95	-3.58	0.96	65.26	12.97	-3.06	0.96	33.02	-6.38	2.36	0.74
21	60.14	10.97	-2.88	0.96	49.33	9.67	-3.52	0.96	65.25	12.38	-3.08	0.96	33.02	Fail	2.3	0.74
22	75.69	12.56	-3.48	0.96	64.58	11.88	-4.55	0.95	81.72	13.75	-3.71	0.96	53.92	Fail	3.0	0.73
23	89.72	12.98	-3.99	0.95	76.08	13.68	-5.29	0.95	98.7	14.14	-4.48	0.95	68.6	Fail	2.54	0.72
24	101.11	13.02	-4.84	0.95	81.18	15.63	-6.69	0.95	108.31	15.46	-4.97	0.95	109.72	Fail	2.52	0.71
25	121.55	14.1	-5.49	0.95	95.28	16.15	-7.3	0.94	130.89	Fail	-6.58	0.94	139.24	Fail	1.58	0.71
26	127.66	14.98	-6.4	0.94	96.68	17.11	-8.1	0.94	136.56	16.18	-6.85	0.94	162.07	Fail	1.21	0.71
27	127.99	15.54	-6.42	0.94	96.77	17.33	-8.49	0.94	136.5	17.48	-6.73	0.94	162.37	Fail	1.13	0.71
28	168.94	16.62	-8.51	0.94	123.57	18.9	-10.98	0.93	183.72	18.58	-10.04	0.93	231.53	4.66	-1.97	0.7
29	174.9	17.15	-9.07	0.93	125.27	19.85	-10.97	0.93	188.81	Fail	-10.13	0.93	243.2	Fail	-1.71	0.7
30	182.45	17.89	-9.22	0.93	127.19	19.5	-11.72	0.93	195.56	Fail	-10.58	0.93	260.04	5.87	-2.52	0.69
31	193.37	19.48	-11.02	0.93	129.4	21.02	-12.88	0.93	203.42	19.66	-12.0	0.93	285.27	8.81	-3.76	0.69
32	196.0	18.92	-10.78	0.93	132.85	21.25	-13.23	0.93	208.32	18.65	-11.65	0.93	315.49	8.49	-3.5	0.69
33	197.83	17.84	-10.83	0.93	133.18	22.62	-13.18	0.93	209.58	20.8	-11.81	0.93	319.29	10.17	-4.02	0.69
34	218.82	19.63	-11.81	0.93	147.76	20.54	-14.07	0.92	236.88	20.72	-13.63	0.92	338.86	12.4	-5.8	0.69
35	230.22	21.63	-12.26	0.93	147.14	21.39	-14.98	0.92	248.76	20.48	-14.43	0.92	362.28	12.26	-5.82	0.69

Table B.9: Aggregated evaluation data for each sweet spot candidate of hevc_nvenc for HRNet.

Index	hevc_nvenc - Coding parameters (GSCNN)
1	qp: 3.0, preset: bd, b_ref_mode: 2, surfaces: 31.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
2	qp: 3.0, preset: slow, b_ref_mode: 0, surfaces: 38.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
3	qp: 3.0, preset: hq, b_ref_mode: 2, surfaces: 64.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 1.0
4	qp: 12.0, preset: hq, b_ref_mode: 2, surfaces: 63.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
5	qp: 14.0, preset: slow, b_ref_mode: 2, surfaces: 0.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 0.0
6	qp: 17.0, preset: fast, b_ref_mode: 0, surfaces: 51.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
7	qp: 17.0, preset: fast, b_ref_mode: 0, surfaces: 43.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
8	qp: 17.0, preset: fast, b_ref_mode: 0, surfaces: 3.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 1.0
9	qp: 20.0, preset: fast, b_ref_mode: 2, surfaces: 4.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
10	qp: 22.0, preset: slow, b_ref_mode: 2, surfaces: 38.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 1.0
11	qp: 24.0, preset: hp, b_ref_mode: 0, surfaces: 34.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
12	qp: 24.0, preset: slow, b_ref_mode: 0, surfaces: 64.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 1.0
13	qp: 27.0, preset: hp, b_ref_mode: 0, surfaces: 4.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
14	qp: 27.0, preset: hp, b_ref_mode: 0, surfaces: 30.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 0.0
15	qp: 27.0, preset: hp, b_ref_mode: 0, surfaces: 51.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 1.0
16	qp: 30.0, preset: fast, b_ref_mode: 2, surfaces: 43.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
17	qp: 30.0, preset: fast, b_ref_mode: 0, surfaces: 3.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 1.0
18	qp: 30.0, preset: fast, b_ref_mode: 2, surfaces: 51.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
19	qp: 32.0, preset: hp, b_ref_mode: 0, surfaces: 60.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 1.0
20	qp: 32.0, preset: bd, b_ref_mode: 2, surfaces: 31.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 1.0
21	qp: 32.0, preset: hp, b_ref_mode: 0, surfaces: 51.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 1.0
22	qp: 32.0, preset: hp, b_ref_mode: 0, surfaces: 51.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 1.0
23	qp: 32.0, preset: hp, b_ref_mode: 0, surfaces: 51.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 1.0

Table B.10: The coding parameters of each sweet spot candidate of hevc_nvenc for GSCNN.

gscnn Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	3.12	Fail	-0.38	0.98	3.22	Fail	-0.56	0.98	3.4	Fail	-0.46	0.98	1.41	Fail	1.54	0.99
2	3.17	Fail	-0.61	0.98	3.31	Fail	-0.38	0.98	3.45	Fail	-0.37	0.98	1.45	Fail	1.52	0.99
3	3.46	Fail	-0.18	0.98	3.5	Fail	-0.66	0.98	3.76	Fail	-0.46	0.98	1.52	Fail	1.63	0.99
4	7.48	Fail	-0.65	0.98	8.33	3.61	-1.18	0.98	8.07	Fail	-0.34	0.98	2.58	-7.7	3.05	0.95
5	9.04	Fail	-0.62	0.98	10.24	Fail	-1.32	0.98	9.76	Fail	-0.26	0.98	3.06	-9.8	3.45	0.93
6	15.63	Fail	-1.19	0.97	16.35	6.67	-2.07	0.97	17.06	3.35	-0.76	0.97	5.25	Fail	5.28	0.87
7	16.45	2.17	-0.76	0.97	16.72	6.15	-2.25	0.97	18.02	3.04	-0.9	0.97	5.51	Fail	5.42	0.86
8	19.91	3.66	-1.26	0.97	18.58	9.38	-3.21	0.97	21.92	5.84	-1.98	0.97	7.22	Fail	6.83	0.82
9	30.1	6.44	-2.32	0.97	27.26	8.86	-3.96	0.97	32.74	Fail	-2.63	0.97	11.78	Fail	6.99	0.78
10	39.89	Fail	-3.86	0.97	38.08	12.7	-4.77	0.96	43.0	6.72	-2.89	0.97	15.48	Fail	6.52	0.77
11	55.21	13.66	-5.36	0.96	44.57	16.23	-7.5	0.96	59.69	10.73	-5.5	0.96	32.6	Fail	5.36	0.74
12	56.52	11.52	-5.37	0.96	49.55	16.01	-6.97	0.96	61.13	10.82	-5.15	0.96	32.12	-8.2	5.71	0.74
13	84.65	14.21	-6.71	0.95	63.45	17.93	-9.14	0.95	89.87	16.32	-6.85	0.95	72.27	Fail	4.96	0.72
14	86.39	15.16	-7.44	0.95	63.56	19.92	-10.19	0.95	92.49	14.77	-7.29	0.95	87.32	Fail	5.82	0.71
15	86.66	13.97	-7.4	0.95	63.7	19.37	-10.03	0.95	92.47	14.87	-7.46	0.95	87.43	-7.9	5.76	0.71
16	122.9	18.32	-9.96	0.94	84.83	23.53	-12.53	0.94	129.24	20.89	-10.15	0.94	147.13	-5.47	3.66	0.7
17	127.77	22.78	-10.77	0.94	85.75	21.16	-13.46	0.94	134.89	21.83	-11.37	0.94	163.41	Fail	2.99	0.7
18	133.2	17.97	-11.12	0.94	88.97	23.73	-13.97	0.94	140.17	21.02	-11.91	0.94	192.45	-2.62	1.95	0.7
19	168.93	22.35	-14.24	0.94	104.7	27.05	-17.41	0.93	177.84	23.61	-15.48	0.93	244.44	1.1	-0.8	0.7
20	173.55	25.98	-14.37	0.94	117.11	24.44	-16.45	0.93	182.71	23.19	-15.73	0.94	253.6	1.07	-0.88	0.7
21	174.93	21.08	-14.61	0.93	108.52	27.54	-17.58	0.93	183.3	23.58	-15.37	0.93	274.21	2.07	-1.52	0.7
22	176.63	21.68	-15.05	0.93	109.05	28.52	-17.81	0.93	185.27	22.15	-16.24	0.93	276.91	3.03	-1.93	0.7
23	177.13	23.25	-15.49	0.93	109.24	29.36	-17.84	0.93	185.16	22.26	-16.16	0.93	277.61	3.12	-2.04	0.7

Table B.11: Aggregated evaluation data for each sweet spot candidate of hevc_nvenc for GSCNN.

Index	h264_nvenc - Coding parameters (HRNet)
1	qp: 11.0, preset: bd, b_ref_mode: 0, surfaces: 51.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 1.0
2	qp: 14.0, preset: slow, b_ref_mode: 2, surfaces: 0.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 0.0
3	qp: 17.0, preset: fast, b_ref_mode: 0, surfaces: 43.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
4	qp: 17.0, preset: fast, b_ref_mode: 0, surfaces: 51.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
5	qp: 19.0, preset: hq, b_ref_mode: 2, surfaces: 8.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 1.0
6	qp: 22.0, preset: hp, b_ref_mode: 0, surfaces: 21.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 1.0
7	qp: 24.0, preset: slow, b_ref_mode: 0, surfaces: 32.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
8	qp: 24.0, preset: slow, b_ref_mode: 2, surfaces: 51.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
9	qp: 26.0, preset: bd, b_ref_mode: 0, surfaces: 51.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 1.0
10	qp: 27.0, preset: hp, b_ref_mode: 0, surfaces: 4.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 0.0
11	qp: 27.0, preset: hq, b_ref_mode: 0, surfaces: 22.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
12	qp: 30.0, preset: slow, b_ref_mode: 0, surfaces: 34.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
13	qp: 30.0, preset: fast, b_ref_mode: 2, surfaces: 38.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 1.0
14	qp: 32.0, preset: hp, b_ref_mode: 0, surfaces: 60.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 1.0
15	qp: 32.0, preset: hp, b_ref_mode: 0, surfaces: 4.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
16	qp: 32.0, preset: slow, b_ref_mode: 2, surfaces: 40.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 1.0

Table B.12: The coding parameters of each sweet spot candidate of h264_nvenc for HRNet.

hrnet Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	7.15	4.81	-0.4	0.98	7.29	Fail	-0.36	0.98	7.85	4.64	-0.39	0.98	2.28	Fail	0.51	0.94
2	9.67	5.15	-0.53	0.98	9.71	Fail	-0.5	0.98	10.6	4.57	-0.52	0.98	3.11	3.32	-0.29	0.91
3	15.04	Fail	-0.86	0.97	14.44	Fail	-1.21	0.97	16.31	Fail	-1.04	0.97	4.93	0.94	-0.12	0.85
4	15.19	5.87	-0.84	0.97	14.58	Fail	-1.26	0.97	16.43	Fail	-1.05	0.97	5.0	1.32	-0.17	0.85
5	19.27	7.09	-1.12	0.97	18.2	6.5	-1.78	0.97	20.88	Fail	-1.3	0.97	6.76	0.8	-0.12	0.81
6	29.17	10.99	-2.56	0.96	25.8	10.06	-3.47	0.96	31.34	9.53	-2.57	0.96	14.46	-4.63	1.25	0.76
7	39.44	11.4	-3.34	0.96	33.72	14.12	-4.95	0.96	42.01	13.49	-3.67	0.96	34.01	Fail	3.4	0.72
8	40.22	11.65	-3.42	0.96	34.28	14.21	-5.08	0.96	42.81	12.31	-3.7	0.96	37.47	Fail	3.33	0.72
9	51.28	13.78	-4.75	0.95	41.81	17.67	-6.63	0.95	54.49	15.19	-5.01	0.95	57.27	Fail	2.79	0.72
10	57.14	16.87	-6.02	0.95	44.65	18.09	-8.09	0.95	60.45	18.54	-6.63	0.95	71.38	Fail	1.29	0.71
11	60.0	17.95	-6.4	0.95	47.33	19.5	-8.11	0.95	62.8	16.96	-6.36	0.95	80.22	Fail	1.76	0.71
12	83.99	19.21	-8.84	0.94	62.02	19.37	-11.07	0.94	87.57	20.71	-9.47	0.94	118.86	3.11	-1.52	0.7
13	84.28	18.21	-9.48	0.94	60.62	19.28	-11.24	0.94	87.95	19.81	-9.7	0.94	130.55	Fail	-1.75	0.7
14	104.85	20.64	-12.31	0.93	71.96	22.31	-14.11	0.93	109.25	23.48	-13.1	0.93	162.85	8.8	-4.45	0.69
15	105.73	20.54	-12.21	0.93	72.26	21.74	-14.04	0.93	109.33	22.39	-13.13	0.93	164.07	8.38	-4.44	0.69
16	109.95	22.11	-12.32	0.93	76.5	24.28	-14.33	0.93	113.66	22.97	-12.57	0.93	178.06	9.31	-4.49	0.69

Table B.13: Aggregated evaluation data for each sweet spot candidate of h264_nvenc for HRNet.

Index	h264_nvenc - Coding parameters (GSCNN)
1	qp: 9.0, preset: slow, b_ref_mode: 0, surfaces: 51.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
2	qp: 11.0, preset: bd, b_ref_mode: 0, surfaces: 0.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 1.0
3	qp: 11.0, preset: medium, b_ref_mode: 0, surfaces: 26.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
4	qp: 14.0, preset: slow, b_ref_mode: 2, surfaces: 0.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 0.0
5	qp: 17.0, preset: fast, b_ref_mode: 0, surfaces: 3.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 1.0
6	qp: 17.0, preset: fast, b_ref_mode: 0, surfaces: 43.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
7	qp: 18.0, preset: slow, b_ref_mode: 0, surfaces: 38.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
8	qp: 19.0, preset: fast, b_ref_mode: 0, surfaces: 51.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 1.0
9	qp: 19.0, preset: slow, b_ref_mode: 0, surfaces: 51.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 1.0
10	qp: 22.0, preset: hp, b_ref_mode: 0, surfaces: 21.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 1.0
11	qp: 24.0, preset: slow, b_ref_mode: 0, surfaces: 34.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
12	qp: 24.0, preset: fast, b_ref_mode: 2, surfaces: 51.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
13	qp: 24.0, preset: bd, b_ref_mode: 0, surfaces: 46.0, nonref_p: 0.0, weighted_pred: 1.0, strict_gop: 0.0
14	qp: 26.0, preset: fast, b_ref_mode: 2, surfaces: 51.0, nonref_p: 0.0, weighted_pred: 0.0, strict_gop: 0.0
15	qp: 27.0, preset: hp, b_ref_mode: 0, surfaces: 4.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 0.0
16	qp: 27.0, preset: hp, b_ref_mode: 0, surfaces: 51.0, nonref_p: 1.0, weighted_pred: 0.0, strict_gop: 0.0
17	qp: 30.0, preset: fast, b_ref_mode: 2, surfaces: 38.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 0.0
18	qp: 30.0, preset: slow, b_ref_mode: 0, surfaces: 38.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
19	qp: 30.0, preset: fast, b_ref_mode: 2, surfaces: 51.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0
20	qp: 32.0, preset: hp, b_ref_mode: 0, surfaces: 64.0, nonref_p: 1.0, weighted_pred: 1.0, strict_gop: 1.0

Table B.14: The coding parameters of each sweet spot candidate of h264_nvenc for GSCNN.

gscnn Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	5.67	Fail	-0.8	0.98	5.82	Fail	-0.35	0.98	6.24	Fail	-0.77	0.98	1.89	Fail	1.24	0.96
2	7.09	Fail	-0.45	0.98	7.22	Fail	-0.58	0.98	7.8	1.22	-0.36	0.98	2.26	Fail	1.38	0.94
3	7.1	Fail	-0.45	0.98	7.22	2.54	-0.7	0.98	7.8	Fail	-0.4	0.98	2.26	Fail	1.4	0.94
4	9.67	Fail	-0.44	0.98	9.71	4.57	-1.36	0.98	10.6	Fail	-0.81	0.98	3.11	Fail	0.9	0.91
5	15.0	3.72	-1.65	0.97	14.42	Fail	-2.32	0.97	16.31	3.98	-1.3	0.97	4.93	-8.01	2.67	0.85
6	15.04	Fail	-1.68	0.97	14.44	7.93	-2.71	0.97	16.31	4.06	-1.39	0.97	4.93	-7.68	2.56	0.85
7	16.46	5.01	-1.7	0.97	15.87	8.27	-2.81	0.97	17.85	4.64	-1.58	0.97	5.71	-8.63	3.68	0.83
8	19.29	6.89	-2.55	0.97	18.02	8.43	-3.32	0.97	20.84	Fail	-2.38	0.97	6.96	-11.69	5.0	0.81
9	19.58	5.9	-2.34	0.97	18.47	9.24	-3.32	0.97	21.15	4.99	-2.02	0.97	6.92	-11.03	4.9	0.81
10	29.17	Fail	-4.0	0.96	25.8	11.7	-4.8	0.96	31.34	Fail	-4.4	0.96	14.46	-9.67	5.98	0.76
11	39.44	11.08	-4.86	0.96	33.72	17.86	-7.06	0.96	42.01	10.07	-5.49	0.96	34.01	-11.9	8.2	0.72
12	39.44	10.58	-5.18	0.96	33.13	15.77	-7.16	0.96	41.97	10.96	-5.39	0.96	37.37	Fail	7.4	0.72
13	40.41	10.67	-5.2	0.96	34.37	14.21	-6.89	0.96	42.83	10.78	-5.46	0.96	37.59	-10.17	7.67	0.72
14	51.43	16.05	-7.65	0.95	41.02	17.93	-8.73	0.95	54.45	14.65	-7.38	0.95	63.71	-9.43	6.59	0.71
15	57.14	17.17	-8.45	0.95	44.65	20.58	-10.09	0.95	60.45	19.1	-8.56	0.95	71.38	-7.27	5.08	0.71
16	58.32	18.82	-9.36	0.95	45.38	19.31	-10.53	0.95	61.4	15.54	-8.45	0.95	79.22	-8.01	5.32	0.71
17	81.78	19.36	-11.71	0.94	59.27	25.26	-14.59	0.94	85.91	24.72	-13.18	0.94	117.05	-1.7	1.32	0.7
18	83.99	21.76	-11.52	0.94	62.02	23.66	-14.42	0.94	87.57	22.57	-12.78	0.94	118.86	-2.46	1.93	0.7
19	84.28	22.63	-12.8	0.94	60.62	22.58	-14.83	0.94	87.95	21.12	-12.72	0.94	130.55	-0.84	0.73	0.7
20	105.73	24.36	-16.02	0.93	72.26	28.49	-18.77	0.93	109.33	29.22	-17.38	0.93	164.07	3.35	-2.48	0.69

Table B.15: Aggregated evaluation data for each sweet spot candidate of h264_nvenc for GSCNN.

Index	h264_vaapi - Coding parameters (HRNet)
1	qp: 1.0, rc_mode: CQP, coder: vlc, compression_level: 4, g: 105.0, sc_threshold: 52.0, bf: 5.0, refs: 13.0, b_depth: 1.0
2	qp: 1.0, rc_mode: CQP, coder: ac, compression_level: 4, g: 176.0, sc_threshold: 32.0, bf: 9.0, refs: 8.0, b_depth: 3.0
3	qp: 1.0, rc_mode: CQP, coder: ac, compression_level: 4, g: 176.0, sc_threshold: 32.0, bf: 9.0, refs: 4.0, b_depth: 1.0
4	qp: 1.0, rc_mode: ICQ, coder: ac, compression_level: 1, g: 176.0, sc_threshold: 32.0, bf: 9.0, refs: 4.0, b_depth: 1.0
5	qp: 1.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 105.0, sc_threshold: 52.0, bf: 3.0, refs: 5.0, b_depth: 1.0
6	qp: 4.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 105.0, sc_threshold: 52.0, bf: 3.0, refs: 13.0, b_depth: 5.0
7	qp: 11.0, rc_mode: CQP, coder: ac, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 2.0, refs: 16.0, b_depth: 5.0
8	qp: 14.0, rc_mode: CQP, coder: vlc, compression_level: 1, g: 1000.0, sc_threshold: 36.0, bf: 7.0, refs: 14.0, b_depth: 1.0
9	qp: 14.0, rc_mode: CQP, coder: vlc, compression_level: 7, g: 240.0, sc_threshold: 36.0, bf: 7.0, refs: 13.0, b_depth: 1.0
10	qp: 18.0, rc_mode: ICQ, coder: ac, compression_level: 1, g: 557.0, sc_threshold: 52.0, bf: 4.0, refs: 8.0, b_depth: 2.0
11	qp: 18.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 105.0, sc_threshold: 52.0, bf: 3.0, refs: 13.0, b_depth: 1.0
12	qp: 19.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 105.0, sc_threshold: 36.0, bf: 7.0, refs: 13.0, b_depth: 1.0
13	qp: 24.0, rc_mode: CQP, coder: ac, compression_level: 4, g: 105.0, sc_threshold: 52.0, bf: 3.0, refs: 13.0, b_depth: 5.0
14	qp: 24.0, rc_mode: CQP, coder: ac, compression_level: 1, g: 557.0, sc_threshold: 52.0, bf: 4.0, refs: 13.0, b_depth: 5.0
15	qp: 25.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 176.0, sc_threshold: 52.0, bf: 3.0, refs: 13.0, b_depth: 5.0
16	qp: 27.0, rc_mode: CQP, coder: ac, compression_level: 4, g: 105.0, sc_threshold: 52.0, bf: 3.0, refs: 13.0, b_depth: 5.0
17	qp: 27.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 105.0, sc_threshold: 52.0, bf: 3.0, refs: 13.0, b_depth: 1.0
18	qp: 27.0, rc_mode: ICQ, coder: ac, compression_level: 1, g: 899.0, sc_threshold: 52.0, bf: 4.0, refs: 4.0, b_depth: 2.0
19	qp: 32.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 105.0, sc_threshold: 52.0, bf: 3.0, refs: 8.0, b_depth: 2.0

Table B.16: The coding parameters of each sweet spot candidate of h264_vaapi for HRNet.

hrnet Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	2.13	Fail	-0.15	0.99	2.33	0.48	-0.03	0.99	2.38	Fail	-0.13	0.99	2.17	Fail	-0.96	0.86
2	2.25	Fail	-0.23	0.99	2.44	Fail	-0.23	0.99	2.5	Fail	-0.2	0.99	2.21	Fail	-1.13	0.86
3	2.28	Fail	-0.15	0.99	2.47	0.75	-0.06	0.99	2.52	Fail	-0.14	0.99	2.22	Fail	-0.96	0.86
4	4.31	Fail	-0.25	0.98	4.46	2.25	-0.25	0.98	4.76	Fail	-0.28	0.98	3.33	Fail	-1.01	0.85
5	4.43	Fail	-0.25	0.98	4.52	Fail	-0.15	0.98	4.9	3.03	-0.25	0.98	3.3	Fail	-0.49	0.85
6	6.16	4.8	-0.59	0.97	6.14	Fail	-0.89	0.97	6.79	Fail	-0.45	0.98	3.96	8.39	-1.15	0.84
7	7.03	Fail	-0.51	0.98	6.98	Fail	-0.29	0.98	7.73	5.27	-0.56	0.98	4.46	Fail	-0.93	0.84
8	11.16	6.25	-0.92	0.97	11.23	5.35	-1.11	0.97	12.14	Fail	-1.04	0.97	5.61	6.77	-1.11	0.82
9	11.81	8.64	-1.03	0.97	11.56	4.78	-1.05	0.97	12.73	7.67	-1.16	0.97	5.78	3.28	-0.52	0.81
10	36.58	Fail	-2.58	0.95	29.41	11.09	-3.77	0.95	39.09	8.48	-2.43	0.96	19.92	2.52	-0.89	0.73
11	37.97	Fail	-2.02	0.96	30.86	8.6	-2.64	0.96	41.1	9.45	-2.33	0.96	17.94	-1.43	0.36	0.75
12	46.67	11.21	-3.03	0.96	37.17	12.69	-3.84	0.95	48.81	12.03	-2.75	0.96	28.04	-0.23	0.07	0.73
13	74.73	13.0	-4.7	0.95	53.01	13.73	-6.35	0.93	77.53	14.0	-4.6	0.95	64.18	Fail	-0.53	0.7
14	75.27	Fail	-5.29	0.94	54.27	14.65	-7.5	0.93	77.46	11.59	-5.15	0.94	86.56	-0.44	0.2	0.69
15	108.62	16.11	-6.75	0.93	77.48	Fail	-10.18	0.91	109.76	15.87	-6.91	0.93	103.22	3.65	-1.22	0.68
16	110.7	17.67	-7.04	0.94	73.81	16.38	-9.31	0.92	113.85	16.66	-6.91	0.94	143.62	1.35	-0.48	0.69
17	134.63	14.9	-7.12	0.93	93.77	17.14	-9.78	0.92	136.27	16.47	-7.57	0.93	146.97	2.0	-0.82	0.69
18	139.17	18.5	-9.56	0.92	96.76	22.78	-13.56	0.9	139.55	19.39	-9.61	0.92	213.29	5.93	-2.6	0.68
19	217.73	23.53	-13.96	0.91	147.22	23.12	-17.98	0.88	216.41	25.17	-15.18	0.91	348.68	14.9	-6.75	0.67

Table B.17: Aggregated evaluation data for each sweet spot candidate of h264_vaapi for HRNet.

Index	h264_vaapi - Coding parameters (GSCNN)
1	qp: 6.0, rc_mode: CQP, coder: ac, compression_level: 1, g: 747.0, sc_threshold: 11.0, bf: 5.0, refs: 6.0, b_depth: 4.0
2	qp: 6.0, rc_mode: CQP, coder: ac, compression_level: 1, g: 683.0, sc_threshold: 28.0, bf: 2.0, refs: 16.0, b_depth: 4.0
3	qp: 6.0, rc_mode: CQP, coder: ac, compression_level: 1, g: 747.0, sc_threshold: 16.0, bf: 5.0, refs: 5.0, b_depth: 1.0
4	qp: 4.0, rc_mode: ICQ, coder: ac, compression_level: 1, g: 747.0, sc_threshold: 16.0, bf: 7.0, refs: 5.0, b_depth: 1.0
5	qp: 11.0, rc_mode: CQP, coder: ac, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 2.0, refs: 16.0, b_depth: 5.0
6	qp: 6.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 747.0, sc_threshold: 16.0, bf: 8.0, refs: 6.0, b_depth: 4.0
7	qp: 9.0, rc_mode: ICQ, coder: ac, compression_level: 1, g: 240.0, sc_threshold: 37.0, bf: 5.0, refs: 6.0, b_depth: 4.0
8	qp: 9.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 747.0, sc_threshold: 16.0, bf: 5.0, refs: 6.0, b_depth: 4.0
9	qp: 14.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 683.0, sc_threshold: 36.0, bf: 5.0, refs: 5.0, b_depth: 1.0
10	qp: 18.0, rc_mode: ICQ, coder: ac, compression_level: 1, g: 769.0, sc_threshold: 36.0, bf: 7.0, refs: 5.0, b_depth: 1.0
11	qp: 21.0, rc_mode: CQP, coder: ac, compression_level: 4, g: 176.0, sc_threshold: 36.0, bf: 2.0, refs: 16.0, b_depth: 4.0
12	qp: 21.0, rc_mode: CQP, coder: ac, compression_level: 1, g: 683.0, sc_threshold: 28.0, bf: 2.0, refs: 16.0, b_depth: 5.0
13	qp: 19.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 683.0, sc_threshold: 16.0, bf: 7.0, refs: 5.0, b_depth: 1.0
14	qp: 24.0, rc_mode: CQP, coder: ac, compression_level: 1, g: 747.0, sc_threshold: 36.0, bf: 5.0, refs: 13.0, b_depth: 1.0
15	qp: 25.0, rc_mode: CQP, coder: ac, compression_level: 1, g: 240.0, sc_threshold: 28.0, bf: 4.0, refs: 13.0, b_depth: 1.0
16	qp: 25.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 769.0, sc_threshold: 36.0, bf: 7.0, refs: 5.0, b_depth: 1.0
17	qp: 27.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 747.0, sc_threshold: 28.0, bf: 2.0, refs: 16.0, b_depth: 1.0
18	qp: 27.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 747.0, sc_threshold: 16.0, bf: 4.0, refs: 13.0, b_depth: 5.0
19	qp: 27.0, rc_mode: ICQ, coder: ac, compression_level: 1, g: 683.0, sc_threshold: 16.0, bf: 7.0, refs: 5.0, b_depth: 1.0
20	qp: 27.0, rc_mode: ICQ, coder: ac, compression_level: 4, g: 747.0, sc_threshold: 16.0, bf: 7.0, refs: 5.0, b_depth: 1.0

Table B.18: The coding parameters of each sweet spot candidate of h264_vaapi for GSCNN.

gscnn Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	3.63	Fail	-0.28	0.98	3.8	Fail	-0.28	0.98	4.04	Fail	-0.55	0.98	3.11	-6.21	1.8	0.85
2	3.68	Fail	0.12	0.98	3.84	Fail	-0.27	0.98	4.08	Fail	0.06	0.98	3.11	-6.17	1.81	0.85
3	3.7	Fail	-0.41	0.98	3.86	Fail	0.04	0.98	4.1	Fail	-0.22	0.98	3.12	-6.6	1.78	0.85
4	6.23	Fail	-0.51	0.98	6.23	Fail	-1.3	0.98	6.92	Fail	-0.32	0.98	4.03	-5.75	2.09	0.85
5	7.03	Fail	-1.18	0.98	6.98	4.17	-1.32	0.98	7.73	Fail	-0.84	0.98	4.46	Fail	2.01	0.84
6	7.4	Fail	-0.67	0.97	7.33	5.56	-2.01	0.97	8.3	3.24	-0.92	0.98	4.59	-6.42	2.18	0.84
7	11.43	5.56	-1.66	0.97	10.98	6.42	-2.27	0.97	12.73	3.31	-1.25	0.97	6.08	-9.02	3.51	0.82
8	11.58	4.55	-1.68	0.97	10.9	5.79	-2.19	0.97	12.8	4.39	-1.31	0.97	6.11	-8.17	3.59	0.82
9	23.82	5.59	-2.5	0.96	20.6	10.99	-3.91	0.96	25.35	5.44	-2.16	0.96	10.62	-10.29	5.48	0.78
10	37.81	10.45	-4.55	0.96	30.72	15.17	-6.58	0.96	41.04	10.08	-4.28	0.96	22.19	Fail	4.79	0.74
11	42.01	12.52	-5.21	0.95	32.97	15.28	-7.06	0.95	44.18	11.09	-5.25	0.96	22.18	Fail	4.07	0.72
12	42.25	12.88	-5.02	0.96	33.46	15.26	-6.84	0.96	44.5	11.01	-4.78	0.96	24.76	Fail	4.48	0.72
13	47.7	13.36	-5.65	0.96	37.37	14.22	-7.4	0.95	50.14	12.39	-5.19	0.95	29.04	Fail	4.06	0.72
14	81.69	15.77	-8.37	0.95	58.81	21.95	-11.51	0.94	83.98	18.48	-8.83	0.95	96.5	Fail	3.26	0.7
15	90.67	17.07	-8.36	0.95	64.21	20.54	-12.6	0.94	93.58	19.7	-9.48	0.95	114.7	Fail	3.47	0.7
16	119.53	24.32	-12.85	0.93	85.24	24.4	-17.47	0.92	116.04	20.51	-12.89	0.94	139.15	Fail	-0.77	0.68
17	130.9	22.91	-12.66	0.93	93.77	28.72	-18.35	0.91	135.49	24.02	-13.19	0.93	156.73	-0.29	0.21	0.69
18	136.69	24.35	-16.03	0.92	97.07	33.26	-22.29	0.9	137.13	24.89	-16.31	0.92	182.04	4.64	-3.65	0.68
19	150.94	29.38	-17.28	0.93	103.89	29.49	-21.75	0.91	146.72	23.78	-17.64	0.93	234.54	Fail	-4.33	0.69
20	152.31	30.26	-17.04	0.93	106.99	33.46	-22.07	0.91	146.91	24.95	-17.44	0.93	213.02	5.3	-3.71	0.68

Table B.19: Aggregated evaluation data for each sweet spot candidate of h264_vaapi for GSCNN.

Index	hevc_vaapi - Coding parameters (HRNet)
1	qp: 1.0, rc_mode: CQP, compression_level: 4, g: 176.0, sc_threshold: 40.0, bf: 2.0, refs: 13.0, b_depth: 5.0
2	qp: 1.0, rc_mode: CQP, compression_level: 4, g: 810.0, sc_threshold: 40.0, bf: 2.0, refs: 6.0, b_depth: 4.0
3	qp: 2.0, rc_mode: CQP, compression_level: 4, g: 176.0, sc_threshold: 12.0, bf: 11.0, refs: 7.0, b_depth: 5.0
4	qp: 1.0, rc_mode: ICQ, compression_level: 1, g: 874.0, sc_threshold: 4.0, bf: 2.0, refs: 6.0, b_depth: 4.0
5	qp: 6.0, rc_mode: CQP, compression_level: 4, g: 557.0, sc_threshold: 44.0, bf: 5.0, refs: 5.0, b_depth: 2.0
6	qp: 3.0, rc_mode: ICQ, compression_level: 7, g: 303.0, sc_threshold: 56.0, bf: 4.0, refs: 14.0, b_depth: 1.0
7	qp: 11.0, rc_mode: CQP, compression_level: 1, g: 240.0, sc_threshold: 16.0, bf: 7.0, refs: 2.0, b_depth: 5.0
8	qp: 11.0, rc_mode: CQP, compression_level: 1, g: 937.0, sc_threshold: 16.0, bf: 7.0, refs: 2.0, b_depth: 5.0
9	qp: 9.0, rc_mode: ICQ, compression_level: 7, g: 683.0, sc_threshold: 24.0, bf: 12.0, refs: 8.0, b_depth: 3.0
10	qp: 14.0, rc_mode: CQP, compression_level: 7, g: 1000.0, sc_threshold: 20.0, bf: 1.0, refs: 10.0, b_depth: 3.0
11	qp: 14.0, rc_mode: CQP, compression_level: 7, g: 1000.0, sc_threshold: 20.0, bf: 6.0, refs: 10.0, b_depth: 5.0
12	qp: 17.0, rc_mode: CQP, compression_level: 4, g: 747.0, sc_threshold: 0.0, bf: 11.0, refs: 12.0, b_depth: 5.0
13	qp: 17.0, rc_mode: CQP, compression_level: 4, g: 747.0, sc_threshold: 0.0, bf: 13.0, refs: 12.0, b_depth: 5.0
14	qp: 17.0, rc_mode: CQP, compression_level: 4, g: 810.0, sc_threshold: 0.0, bf: 13.0, refs: 12.0, b_depth: 5.0
15	qp: 20.0, rc_mode: CQP, compression_level: 1, g: 874.0, sc_threshold: 4.0, bf: 2.0, refs: 6.0, b_depth: 4.0
16	qp: 20.0, rc_mode: CQP, compression_level: 1, g: 874.0, sc_threshold: 4.0, bf: 6.0, refs: 9.0, b_depth: 2.0
17	qp: 19.0, rc_mode: ICQ, compression_level: 7, g: 810.0, sc_threshold: 4.0, bf: 3.0, refs: 11.0, b_depth: 4.0
18	qp: 20.0, rc_mode: ICQ, compression_level: 4, g: 176.0, sc_threshold: 40.0, bf: 2.0, refs: 9.0, b_depth: 4.0
19	qp: 20.0, rc_mode: ICQ, compression_level: 4, g: 747.0, sc_threshold: 4.0, bf: 2.0, refs: 6.0, b_depth: 4.0
20	qp: 24.0, rc_mode: CQP, compression_level: 1, g: 874.0, sc_threshold: 40.0, bf: 2.0, refs: 13.0, b_depth: 3.0
21	qp: 22.0, rc_mode: ICQ, compression_level: 1, g: 430.0, sc_threshold: 52.0, bf: 6.0, refs: 15.0, b_depth: 3.0
22	qp: 24.0, rc_mode: CQP, compression_level: 1, g: 493.0, sc_threshold: 9.0, bf: 6.0, refs: 7.0, b_depth: 2.0
23	qp: 27.0, rc_mode: CQP, compression_level: 1, g: 176.0, sc_threshold: 4.0, bf: 2.0, refs: 6.0, b_depth: 4.0
24	qp: 27.0, rc_mode: CQP, compression_level: 1, g: 874.0, sc_threshold: 4.0, bf: 2.0, refs: 13.0, b_depth: 3.0
25	qp: 27.0, rc_mode: ICQ, compression_level: 4, g: 50.0, sc_threshold: 60.0, bf: 2.0, refs: 6.0, b_depth: 4.0
26	qp: 27.0, rc_mode: CQP, compression_level: 1, g: 874.0, sc_threshold: 4.0, bf: 15.0, refs: 6.0, b_depth: 4.0
27	qp: 27.0, rc_mode: ICQ, compression_level: 1, g: 874.0, sc_threshold: 40.0, bf: 2.0, refs: 16.0, b_depth: 4.0
28	qp: 29.0, rc_mode: ICQ, compression_level: 1, g: 176.0, sc_threshold: 40.0, bf: 2.0, refs: 16.0, b_depth: 4.0
29	qp: 29.0, rc_mode: CQP, compression_level: 1, g: 874.0, sc_threshold: 4.0, bf: 13.0, refs: 12.0, b_depth: 4.0

Table B.20: The coding parameters of each sweet spot candidate of hevc_vaapi for HRNet.

hrnet Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	2.52	Fail	-0.11	0.99	2.7	Fail	-0.1	0.99	2.78	Fail	-0.12	0.99	2.33	Fail	-0.3	0.86
2	2.52	1.44	-0.11	0.99	2.7	1.33	-0.12	0.99	2.79	Fail	-0.11	0.99	2.33	2.99	-0.34	0.86
3	2.81	Fail	-0.12	0.99	2.97	0.3	-0.03	0.98	3.04	Fail	-0.16	0.99	2.46	0.69	-0.06	0.85
4	3.89	Fail	-0.27	0.98	4.16	Fail	-0.22	0.98	4.26	Fail	-0.2	0.98	2.97	-3.34	0.43	0.85
5	4.51	2.64	-0.26	0.98	4.6	Fail	-0.21	0.98	4.84	3.41	-0.31	0.98	3.17	Fail	0.64	0.85
6	5.04	Fail	-0.3	0.98	5.28	2.59	-0.3	0.98	5.39	3.58	-0.27	0.98	3.43	Fail	0.63	0.84
7	10.04	4.25	-0.49	0.97	9.56	2.24	-0.37	0.97	10.53	Fail	-0.56	0.97	5.04	-6.05	1.07	0.82
8	10.05	4.48	-0.52	0.97	9.58	2.13	-0.37	0.97	10.54	Fail	-0.54	0.97	5.04	Fail	1.07	0.82
9	12.9	6.75	-0.73	0.97	12.11	4.14	-0.84	0.97	10.59	5.71	-0.67	0.97	6.04	Fail	0.67	0.81
10	13.93	6.51	-0.81	0.96	12.8	Fail	-0.93	0.96	15.09	Fail	-0.73	0.97	6.6	-6.83	1.69	0.79
11	18.12	Fail	-0.8	0.97	15.89	3.99	-0.94	0.97	18.77	Fail	-0.84	0.97	8.06	-6.49	1.36	0.8
12	29.6	7.31	-1.22	0.96	23.9	6.22	-1.63	0.96	28.17	6.86	-1.35	0.96	13.83	-5.58	1.55	0.76
13	29.78	9.23	-1.49	0.96	24.01	6.96	-1.78	0.96	28.65	Fail	-2.01	0.96	13.97	-3.99	1.13	0.76
14	29.78	9.23	-1.49	0.96	24.01	6.96	-1.78	0.96	28.65	Fail	-2.01	0.96	13.97	-3.99	1.13	0.76
15	41.61	9.34	-2.15	0.96	32.23	8.11	-2.7	0.95	43.82	8.99	-2.17	0.96	21.82	-3.25	0.9	0.72
16	46.5	8.61	-2.55	0.96	35.17	10.74	-3.48	0.95	45.76	10.26	-2.55	0.96	28.38	-2.37	0.67	0.73
17	56.21	12.29	-3.4	0.95	40.79	11.06	-4.68	0.95	55.36	11.42	-3.53	0.95	30.86	Fail	1.24	0.72
18	58.04	11.79	-3.59	0.95	42.73	10.87	-4.71	0.95	58.25	11.08	-3.6	0.95	32.01	-2.55	0.75	0.72
19	58.96	10.83	-3.76	0.95	43.19	11.46	-4.79	0.95	59.09	12.04	-3.71	0.95	32.77	-0.24	0.08	0.72
20	77.38	11.79	-4.62	0.95	53.73	13.04	-6.17	0.94	81.11	11.35	-4.57	0.95	58.43	4.2	-1.54	0.69
21	82.0	13.78	-5.38	0.95	55.17	15.24	-7.48	0.94	72.05	11.8	-4.94	0.95	72.26	-0.01	0.0	0.71
22	89.81	13.08	-5.21	0.95	60.52	13.69	-7.07	0.94	87.28	13.32	-5.43	0.95	88.43	4.93	-1.73	0.7
23	117.15	14.54	-7.16	0.94	75.47	16.33	-9.03	0.92	121.84	15.74	-7.36	0.94	143.21	4.68	-1.5	0.69
24	118.32	14.4	-7.02	0.94	75.97	17.2	-9.23	0.92	123.07	16.56	-7.41	0.94	144.81	4.86	-1.62	0.69
25	126.66	16.63	-8.49	0.94	81.2	18.74	-11.58	0.93	130.52	17.75	-9.19	0.94	156.58	6.1	-2.44	0.7
26	150.14	16.49	-9.01	0.93	91.65	Fail	-11.82	0.92	139.52	19.69	-10.04	0.93	225.37	7.29	-2.97	0.69
27	150.98	17.01	-10.17	0.93	92.74	19.22	-13.79	0.92	149.72	17.95	-10.78	0.93	215.16	9.67	-3.55	0.69
28	187.78	18.13	-12.82	0.92	113.55	22.32	-17.32	0.9	187.35	21.78	-14.25	0.92	292.77	13.84	-6.31	0.68
29	204.43	20.21	-13.45	0.92	124.85	22.51	-16.68	0.9	185.61	22.48	-16.96	0.92	339.42	13.67	-6.65	0.68

Table B.21: Aggregated evaluation data for each sweet spot candidate of hevc_vaapi for HRNet.

Index	hevc_vaapi - Coding parameters (GSCNN)
1	qp: 1.0, rc_mode: CQP, compression_level: 4, g: 937.0, sc_threshold: 32.0, bf: 12.0, refs: 7.0, b_depth: 4.0
2	qp: 1.0, rc_mode: CQP, compression_level: 7, g: 810.0, sc_threshold: 12.0, bf: 11.0, refs: 7.0, b_depth: 5.0
3	qp: 6.0, rc_mode: CQP, compression_level: 4, g: 557.0, sc_threshold: 44.0, bf: 5.0, refs: 5.0, b_depth: 2.0
4	qp: 3.0, rc_mode: ICQ, compression_level: 7, g: 303.0, sc_threshold: 56.0, bf: 4.0, refs: 14.0, b_depth: 1.0
5	qp: 11.0, rc_mode: CQP, compression_level: 1, g: 240.0, sc_threshold: 16.0, bf: 7.0, refs: 2.0, b_depth: 5.0
6	qp: 10.0, rc_mode: ICQ, compression_level: 1, g: 367.0, sc_threshold: 40.0, bf: 2.0, refs: 6.0, b_depth: 4.0
7	qp: 14.0, rc_mode: CQP, compression_level: 7, g: 1000.0, sc_threshold: 20.0, bf: 1.0, refs: 10.0, b_depth: 3.0
8	qp: 14.0, rc_mode: CQP, compression_level: 7, g: 1000.0, sc_threshold: 20.0, bf: 6.0, refs: 10.0, b_depth: 5.0
9	qp: 17.0, rc_mode: CQP, compression_level: 4, g: 747.0, sc_threshold: 0.0, bf: 11.0, refs: 12.0, b_depth: 5.0
10	qp: 17.0, rc_mode: CQP, compression_level: 4, g: 747.0, sc_threshold: 0.0, bf: 13.0, refs: 12.0, b_depth: 5.0
11	qp: 19.0, rc_mode: ICQ, compression_level: 7, g: 50.0, sc_threshold: 28.0, bf: 3.0, refs: 7.0, b_depth: 5.0
12	qp: 19.0, rc_mode: ICQ, compression_level: 7, g: 874.0, sc_threshold: 28.0, bf: 3.0, refs: 11.0, b_depth: 4.0
13	qp: 24.0, rc_mode: CQP, compression_level: 1, g: 367.0, sc_threshold: 40.0, bf: 2.0, refs: 6.0, b_depth: 4.0
14	qp: 24.0, rc_mode: CQP, compression_level: 4, g: 176.0, sc_threshold: 12.0, bf: 6.0, refs: 9.0, b_depth: 2.0
15	qp: 24.0, rc_mode: CQP, compression_level: 4, g: 176.0, sc_threshold: 12.0, bf: 8.0, refs: 4.0, b_depth: 4.0
16	qp: 24.0, rc_mode: CQP, compression_level: 4, g: 810.0, sc_threshold: 33.0, bf: 8.0, refs: 4.0, b_depth: 4.0
17	qp: 27.0, rc_mode: CQP, compression_level: 1, g: 874.0, sc_threshold: 32.0, bf: 2.0, refs: 6.0, b_depth: 1.0

Table B.22: The coding parameters of each sweet spot candidate of hevc_vaapi for GSCNN.

gscnn Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	2.56	Fail	-0.28	0.99	2.73	Fail	-0.22	0.99	2.79	Fail	-0.1	0.99	2.34	Fail	2.72	0.86
2	2.56	Fail	0.16	0.99	2.73	Fail	-0.15	0.99	2.8	Fail	-0.1	0.99	2.35	Fail	2.9	0.86
3	4.51	Fail	-0.1	0.98	4.6	Fail	-0.19	0.98	4.84	Fail	-0.09	0.98	3.17	-10.83	4.41	0.85
4	5.04	Fail	-0.24	0.98	5.28	Fail	-0.18	0.98	5.39	0.47	-0.13	0.98	3.43	-12.55	4.78	0.84
5	10.04	1.95	-0.63	0.97	9.56	2.61	-0.88	0.97	10.53	1.84	-0.63	0.97	5.04	-12.77	5.06	0.82
6	12.95	2.65	-0.88	0.97	12.74	4.96	-1.95	0.96	13.68	Fail	-1.03	0.97	6.1	-12.28	5.59	0.8
7	13.93	4.37	-1.4	0.96	12.8	6.5	-2.35	0.96	15.09	3.85	-1.23	0.97	6.6	-13.85	6.64	0.79
8	18.12	3.48	-1.32	0.97	15.89	7.0	-2.94	0.97	18.77	Fail	-1.48	0.97	8.06	Fail	5.65	0.8
9	29.6	5.9	-2.53	0.96	23.9	9.32	-3.86	0.96	28.17	6.65	-2.8	0.96	13.83	Fail	7.14	0.76
10	29.78	7.55	-3.08	0.96	24.01	10.54	-4.8	0.96	28.65	Fail	-3.26	0.96	13.97	Fail	7.32	0.76
11	51.59	12.51	-5.74	0.95	37.6	17.78	-9.14	0.95	51.86	14.48	-6.59	0.95	26.65	Fail	5.06	0.73
12	56.21	15.28	-6.6	0.95	40.79	19.71	-9.79	0.95	55.36	13.8	-6.71	0.95	30.86	-6.38	4.3	0.72
13	77.38	17.62	-7.73	0.95	53.73	20.14	-10.96	0.94	81.11	17.26	-8.01	0.95	58.43	-0.09	0.06	0.69
14	87.99	15.93	-9.49	0.95	59.77	21.43	-12.76	0.94	85.16	18.36	-10.38	0.95	86.58	0.09	-0.07	0.7
15	95.15	16.44	-9.55	0.94	62.81	22.01	-13.76	0.94	92.7	19.12	-11.65	0.94	98.83	0.09	-0.07	0.69
16	96.14	16.13	-9.64	0.94	63.2	22.3	-13.63	0.94	93.02	19.03	-11.5	0.94	100.36	-0.04	0.03	0.69
17	116.31	18.6	-11.84	0.94	75.43	22.85	-15.27	0.92	120.86	20.93	-12.05	0.94	142.88	-0.21	0.15	0.69

Table B.23: Aggregated evaluation data for each sweet spot candidate of hevc_vaapi for GSCNN.

Index	vp9_vaapi - Coding parameters (HRNet)
1	b:v: 150.0, bufratio: 3.883, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 0.0, qmax: 59.0, loop_filter_level: 55.0, loop_filter_sharpness: 9.0
2	b:v: 110.0, bufratio: 3.883, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 0.0, qmax: 59.0, loop_filter_level: 55.0, loop_filter_sharpness: 9.0
3	b:v: 80.0, bufratio: 3.883, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 0.0, qmax: 59.0, loop_filter_level: 55.0, loop_filter_sharpness: 9.0
4	b:v: 50.0, bufratio: 3.883, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 6.0, qmin: 7.0, qmax: 47.0, loop_filter_level: 51.0, loop_filter_sharpness: 12.0
5	b:v: 50.0, bufratio: 3.883, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 7.0, qmax: 29.0, loop_filter_level: 55.0, loop_filter_sharpness: 11.0
6	b:v: 50.0, bufratio: 3.883, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 7.0, qmax: 59.0, loop_filter_level: 55.0, loop_filter_sharpness: 9.0
7	b:v: 36.933, bufratio: 0.75, rc_mode: 1.0, compression_level: 1, g: 230.0, sc_threshold: 52.0, bf: 0.0, refs: 5.0, qmin: 7.0, qmax: 62.0, loop_filter_level: 38.0, loop_filter_sharpness: 7.0
8	b:v: 36.933, bufratio: 0.75, rc_mode: 1.0, compression_level: 1, g: 230.0, sc_threshold: 52.0, bf: 0.0, refs: 5.0, qmin: 7.0, qmax: 15.0, loop_filter_level: 34.0, loop_filter_sharpness: 4.0
9	b:v: 30.4, bufratio: 2.0, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 4.0, qmin: 7.0, qmax: 62.0, loop_filter_level: 34.0, loop_filter_sharpness: 4.0
10	b:v: 23.867, bufratio: 1.0, rc_mode: 1.0, compression_level: 4, g: 747.0, sc_threshold: 16.0, bf: 2.0, refs: 3.0, qmin: 10.0, qmax: 47.0, loop_filter_level: 51.0, loop_filter_sharpness: 12.0
11	b:v: 23.867, bufratio: 1.0, rc_mode: 0.0, compression_level: 4, g: 747.0, sc_threshold: 16.0, bf: 2.0, refs: 3.0, qmin: 10.0, qmax: 47.0, loop_filter_level: 51.0, loop_filter_sharpness: 12.0
12	b:v: 23.867, bufratio: 1.0, rc_mode: 0.0, compression_level: 4, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 3.0, qmin: 10.0, qmax: 47.0, loop_filter_level: 51.0, loop_filter_sharpness: 12.0
13	b:v: 14.067, bufratio: 3.883, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 7.0, qmax: 69.0, loop_filter_level: 46.0, loop_filter_sharpness: 10.0
14	b:v: 14.067, bufratio: 1.25, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 7.0, qmax: 69.0, loop_filter_level: 46.0, loop_filter_sharpness: 10.0
15	b:v: 14.067, bufratio: 1.321, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 15.0, qmin: 7.0, qmax: 69.0, loop_filter_level: 46.0, loop_filter_sharpness: 10.0
16	b:v: 11.848, bufratio: 1.25, rc_mode: 0.0, compression_level: 4, g: 747.0, sc_threshold: 16.0, bf: 0.0, refs: 5.0, qmin: 7.0, qmax: 69.0, loop_filter_level: 46.0, loop_filter_sharpness: 10.0
17	b:v: 10.8, bufratio: 2.0, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 7.0, qmin: 5.0, qmax: 26.0, loop_filter_level: 63.0, loop_filter_sharpness: 1.0
18	b:v: 10.8, bufratio: 2.0, rc_mode: 0.0, compression_level: 7, g: 230.0, sc_threshold: 52.0, bf: 0.0, refs: 7.0, qmin: 5.0, qmax: 26.0, loop_filter_level: 47.0, loop_filter_sharpness: 1.0
19	b:v: 10.8, bufratio: 2.0, rc_mode: 0.0, compression_level: 4, g: 176.0, sc_threshold: 40.0, bf: 0.0, refs: 7.0, qmin: 5.0, qmax: 26.0, loop_filter_level: 63.0, loop_filter_sharpness: 1.0
20	b:v: 8.849, bufratio: 2.0, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 11.0, qmax: 63.0, loop_filter_level: 17.0, loop_filter_sharpness: 0.0
21	b:v: 50.0, bufratio: 3.75, rc_mode: 1.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 16.0, qmin: 11.0, qmax: 63.0, loop_filter_level: 17.0, loop_filter_sharpness: 0.0
22	b:v: 4.267, bufratio: 3.962, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 7.0, qmax: 59.0, loop_filter_level: 55.0, loop_filter_sharpness: 9.0
23	b:v: 4.267, bufratio: 2.0, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 11.0, qmax: 63.0, loop_filter_level: 17.0, loop_filter_sharpness: 0.0

Table B.24: The coding parameters of each sweet spot candidate of vp9_vaapi for HRNet.

hrnet Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	3.82	Fail	-0.32	0.98	4.48	Fail	-0.4	0.98	4.27	Fail	-0.31	0.98	6.6	Fail	1.16	0.8
2	5.16	Fail	-0.46	0.98	6.05	Fail	-0.4	0.97	5.76	Fail	-0.31	0.98	9.07	-4.65	0.96	0.78
3	7.07	Fail	-0.63	0.97	8.33	4.72	-0.93	0.97	7.88	5.8	-0.67	0.97	12.47	Fail	1.19	0.76
4	11.36	Fail	-1.03	0.97	13.39	Fail	-1.49	0.97	12.66	Fail	-1.03	0.97	20.0	-2.63	0.7	0.73
5	11.36	Fail	-0.99	0.97	13.39	Fail	-1.48	0.97	12.66	Fail	-0.99	0.97	19.99	-2.87	0.83	0.73
6	11.36	Fail	-1.0	0.97	13.39	Fail	-1.49	0.97	12.66	Fail	-0.98	0.97	19.99	Fail	0.97	0.73
7	15.27	5.36	-1.03	0.97	18.08	6.32	-1.71	0.96	16.86	6.15	-1.06	0.97	26.88	1.34	-0.41	0.72
8	15.29	6.26	-1.02	0.97	18.07	6.35	-1.66	0.96	16.85	6.16	-0.97	0.97	26.89	-2.53	0.82	0.72
9	18.81	Fail	-1.46	0.97	22.05	7.48	-2.32	0.96	20.91	6.9	-1.51	0.96	32.86	-1.7	0.62	0.71
10	23.53	6.77	-2.13	0.96	27.81	8.91	-3.71	0.95	26.1	8.0	-1.99	0.96	41.7	-0.67	0.21	0.71
11	23.87	8.21	-2.1	0.96	28.05	9.69	-3.88	0.95	26.56	8.4	-2.26	0.96	41.76	0.86	-0.28	0.7
12	23.96	Fail	-1.75	0.96	28.11	8.44	-3.23	0.96	26.64	8.94	-1.96	0.96	41.92	-1.47	0.51	0.71
13	40.74	9.59	-3.11	0.96	47.64	12.85	-6.15	0.94	45.29	9.9	-3.12	0.95	71.3	-0.4	0.16	0.7
14	40.75	9.57	-3.11	0.96	47.64	12.65	-6.29	0.94	45.3	9.6	-3.2	0.95	71.31	0.18	-0.07	0.7
15	40.75	9.57	-3.11	0.96	47.64	13.0	-6.18	0.94	45.3	9.74	-3.17	0.95	71.29	-0.47	0.18	0.7
16	48.33	10.6	-3.91	0.95	56.47	Fail	-8.49	0.93	53.62	13.98	-4.06	0.95	84.6	0.53	-0.22	0.7
17	53.09	10.49	-4.41	0.95	62.01	13.95	-9.61	0.93	58.96	11.02	-4.19	0.95	92.82	1.43	-0.61	0.7
18	53.09	11.21	-4.23	0.95	61.97	15.23	-9.35	0.93	58.9	12.05	-4.41	0.95	92.64	1.8	-0.74	0.7
19	53.13	11.49	-4.33	0.95	61.96	13.78	-9.43	0.93	58.92	12.16	-4.32	0.95	92.66	2.04	-0.89	0.7
20	64.82	11.13	-4.88	0.95	75.56	17.93	-12.01	0.92	71.78	13.98	-5.25	0.95	113.18	1.7	-0.81	0.7
21	80.14	Fail	-6.58	0.94	94.2	20.29	-16.13	0.91	87.8	14.24	-6.75	0.94	141.27	3.03	-1.65	0.7
22	134.52	14.83	-11.35	0.92	156.67	28.73	-28.01	0.87	149.36	16.32	-12.44	0.92	234.46	10.37	-6.44	0.68
23	134.53	Fail	-11.01	0.93	156.68	29.16	-25.68	0.87	149.36	16.04	-11.54	0.92	234.43	8.06	-5.33	0.68

Table B.25: Aggregated evaluation data for each sweet spot candidate of vp9_vaapi for HRNet.

Index	vp9_vaapi - Coding parameters (GSCNN)
1	b:v: 150.0, bufratio: 2.25, rc_mode: 0.0, compression_level: 4, g: 557.0, sc_threshold: 4.0, bf: 5.0, refs: 13.0, qmin: 0.0, qmax: 51.0, loop_filter_level: 25.0, loop_filter_sharpness: 2.0
2	b:v: 150.0, bufratio: 2.0, rc_mode: 0.0, compression_level: 4, g: 176.0, sc_threshold: 40.0, bf: 0.0, refs: 4.0, qmin: 0.0, qmax: 15.0, loop_filter_level: 34.0, loop_filter_sharpness: 4.0
3	b:v: 150.0, bufratio: 3.75, rc_mode: 0.0, compression_level: 4, g: 430.0, sc_threshold: 44.0, bf: 7.0, refs: 11.0, qmin: 0.0, qmax: 29.0, loop_filter_level: 55.0, loop_filter_sharpness: 11.0
4	b:v: 110.0, bufratio: 2.25, rc_mode: 0.0, compression_level: 4, g: 557.0, sc_threshold: 4.0, bf: 5.0, refs: 13.0, qmin: 0.0, qmax: 51.0, loop_filter_level: 25.0, loop_filter_sharpness: 2.0
5	b:v: 110.0, bufratio: 2.0, rc_mode: 0.0, compression_level: 4, g: 176.0, sc_threshold: 40.0, bf: 0.0, refs: 4.0, qmin: 0.0, qmax: 15.0, loop_filter_level: 34.0, loop_filter_sharpness: 4.0
6	b:v: 110.0, bufratio: 3.75, rc_mode: 0.0, compression_level: 4, g: 430.0, sc_threshold: 44.0, bf: 7.0, refs: 11.0, qmin: 0.0, qmax: 29.0, loop_filter_level: 55.0, loop_filter_sharpness: 11.0
7	b:v: 80.0, bufratio: 2.25, rc_mode: 0.0, compression_level: 4, g: 557.0, sc_threshold: 4.0, bf: 5.0, refs: 13.0, qmin: 0.0, qmax: 51.0, loop_filter_level: 25.0, loop_filter_sharpness: 2.0
8	b:v: 80.0, bufratio: 3.75, rc_mode: 0.0, compression_level: 4, g: 430.0, sc_threshold: 44.0, bf: 7.0, refs: 11.0, qmin: 0.0, qmax: 29.0, loop_filter_level: 55.0, loop_filter_sharpness: 11.0
9	b:v: 80.0, bufratio: 2.0, rc_mode: 0.0, compression_level: 4, g: 176.0, sc_threshold: 40.0, bf: 0.0, refs: 4.0, qmin: 0.0, qmax: 15.0, loop_filter_level: 34.0, loop_filter_sharpness: 4.0
10	b:v: 50.0, bufratio: 3.75, rc_mode: 0.0, compression_level: 4, g: 430.0, sc_threshold: 44.0, bf: 7.0, refs: 11.0, qmin: 2.0, qmax: 29.0, loop_filter_level: 55.0, loop_filter_sharpness: 11.0
11	b:v: 50.0, bufratio: 3.25, rc_mode: 0.0, compression_level: 4, g: 430.0, sc_threshold: 44.0, bf: 7.0, refs: 11.0, qmin: 0.0, qmax: 29.0, loop_filter_level: 55.0, loop_filter_sharpness: 11.0
12	b:v: 50.0, bufratio: 3.25, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 1.0, refs: 16.0, qmin: 3.0, qmax: 44.0, loop_filter_level: 17.0, loop_filter_sharpness: 0.0
13	b:v: 50.0, bufratio: 3.25, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 1.0, refs: 16.0, qmin: 3.0, qmax: 41.0, loop_filter_level: 63.0, loop_filter_sharpness: 4.0
14	b:v: 50.0, bufratio: 3.25, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 53.0, bf: 0.0, refs: 13.0, qmin: 1.0, qmax: 62.0, loop_filter_level: 38.0, loop_filter_sharpness: 14.0
15	b:v: 36.933, bufratio: 0.75, rc_mode: 1.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 7.0, qmax: 62.0, loop_filter_level: 38.0, loop_filter_sharpness: 14.0
16	b:v: 36.933, bufratio: 0.75, rc_mode: 0.0, compression_level: 4, g: 176.0, sc_threshold: 40.0, bf: 0.0, refs: 4.0, qmin: 7.0, qmax: 62.0, loop_filter_level: 38.0, loop_filter_sharpness: 14.0
17	b:v: 33.667, bufratio: 2.25, rc_mode: 0.0, compression_level: 4, g: 557.0, sc_threshold: 4.0, bf: 5.0, refs: 13.0, qmin: 12.0, qmax: 51.0, loop_filter_level: 25.0, loop_filter_sharpness: 2.0
18	b:v: 30.4, bufratio: 2.0, rc_mode: 1.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 5.0, qmin: 7.0, qmax: 15.0, loop_filter_level: 12.0, loop_filter_sharpness: 4.0
19	b:v: 30.4, bufratio: 2.0, rc_mode: 0.0, compression_level: 4, g: 176.0, sc_threshold: 40.0, bf: 0.0, refs: 4.0, qmin: 7.0, qmax: 15.0, loop_filter_level: 34.0, loop_filter_sharpness: 4.0
20	b:v: 23.867, bufratio: 0.843, rc_mode: 1.0, compression_level: 4, g: 937.0, sc_threshold: 44.0, bf: 0.0, refs: 11.0, qmin: 10.0, qmax: 60.0, loop_filter_level: 51.0, loop_filter_sharpness: 12.0
21	b:v: 23.867, bufratio: 1.25, rc_mode: 0.0, compression_level: 7, g: 600.0, sc_threshold: 24.0, bf: 1.0, refs: 16.0, qmin: 10.0, qmax: 47.0, loop_filter_level: 51.0, loop_filter_sharpness: 12.0
22	b:v: 23.867, bufratio: 1.0, rc_mode: 0.0, compression_level: 7, g: 963.0, sc_threshold: 24.0, bf: 0.0, refs: 16.0, qmin: 0.0, qmax: 47.0, loop_filter_level: 51.0, loop_filter_sharpness: 12.0
23	b:v: 14.067, bufratio: 1.25, rc_mode: 0.0, compression_level: 4, g: 810.0, sc_threshold: 40.0, bf: 1.0, refs: 16.0, qmin: 14.0, qmax: 69.0, loop_filter_level: 46.0, loop_filter_sharpness: 10.0
24	b:v: 14.067, bufratio: 1.25, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 1.0, refs: 16.0, qmin: 14.0, qmax: 69.0, loop_filter_level: 46.0, loop_filter_sharpness: 10.0
25	b:v: 14.067, bufratio: 1.0, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 15.0, bf: 0.0, refs: 6.0, qmin: 14.0, qmax: 69.0, loop_filter_level: 46.0, loop_filter_sharpness: 10.0
26	b:v: 10.8, bufratio: 3.5, rc_mode: 0.0, compression_level: 7, g: 113.0, sc_threshold: 52.0, bf: 2.0, refs: 3.0, qmin: 3.0, qmax: 63.0, loop_filter_level: 17.0, loop_filter_sharpness: 0.0
27	b:v: 10.8, bufratio: 3.443, rc_mode: 0.0, compression_level: 7, g: 489.0, sc_threshold: 52.0, bf: 2.0, refs: 3.0, qmin: 3.0, qmax: 44.0, loop_filter_level: 17.0, loop_filter_sharpness: 0.0
28	b:v: 10.8, bufratio: 1.0, rc_mode: 0.0, compression_level: 7, g: 303.0, sc_threshold: 24.0, bf: 0.0, refs: 6.0, qmin: 10.0, qmax: 26.0, loop_filter_level: 63.0, loop_filter_sharpness: 1.0
29	b:v: 50.0, bufratio: 3.75, rc_mode: 1.0, compression_level: 7, g: 113.0, sc_threshold: 28.0, bf: 3.0, refs: 11.0, qmin: 2.0, qmax: 29.0, loop_filter_level: 55.0, loop_filter_sharpness: 11.0
30	b:v: 50.0, bufratio: 3.75, rc_mode: 1.0, compression_level: 4, g: 430.0, sc_threshold: 24.0, bf: 0.0, refs: 1.0, qmin: 6.0, qmax: 41.0, loop_filter_level: 63.0, loop_filter_sharpness: 4.0

Table B.26: The coding parameters of each sweet spot candidate of vp9_vaapi for GSCNN.

gscnn Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	3.82	Fail	-0.36	0.98	4.47	1.99	-0.57	0.98	4.29	Fail	-0.24	0.98	6.57	-14.84	6.17	0.79
2	3.82	Fail	-0.52	0.98	4.48	2.6	-0.76	0.98	4.27	Fail	-0.59	0.98	6.61	-13.79	6.14	0.8
3	3.84	Fail	-0.3	0.98	4.48	Fail	-0.73	0.98	4.39	Fail	0.38	0.98	6.6	-11.68	5.81	0.79
4	5.13	1.32	-0.45	0.98	6.02	3.68	-1.06	0.97	5.77	Fail	-0.66	0.98	8.97	-12.19	6.75	0.77
5	5.17	2.59	-0.92	0.98	6.06	3.15	-0.92	0.97	5.77	Fail	-0.6	0.98	9.05	Fail	6.86	0.78
6	5.17	Fail	-0.55	0.98	6.06	5.5	-1.72	0.97	5.83	Fail	-0.82	0.98	8.99	Fail	6.29	0.77
7	7.01	2.57	-0.74	0.97	8.26	5.51	-1.72	0.97	7.86	Fail	-1.35	0.98	12.34	-9.28	5.69	0.75
8	7.05	Fail	-0.97	0.97	8.29	6.06	-2.36	0.97	7.9	3.86	-1.22	0.97	12.37	-8.81	5.13	0.75
9	7.08	Fail	-0.62	0.97	8.33	5.31	-1.82	0.97	7.88	2.77	-0.97	0.97	12.48	Fail	6.43	0.76
10	11.19	4.44	-1.83	0.97	13.26	8.99	-3.79	0.96	12.54	6.25	-2.21	0.97	19.82	Fail	4.0	0.73
11	11.19	4.44	-1.83	0.97	13.26	8.99	-3.79	0.96	12.54	6.25	-2.21	0.97	19.82	Fail	4.0	0.73
12	11.31	5.5	-1.61	0.97	13.35	6.75	-2.58	0.97	12.61	Fail	-1.53	0.97	19.95	Fail	6.02	0.73
13	11.31	Fail	-1.86	0.97	13.34	7.11	-2.74	0.97	12.62	4.88	-1.3	0.97	19.93	Fail	4.82	0.73
14	11.37	3.97	-1.16	0.97	13.39	Fail	-3.08	0.96	12.68	4.37	-1.33	0.97	20.0	-6.35	3.78	0.73
15	15.23	Fail	-1.84	0.97	18.05	9.26	-3.67	0.96	16.8	Fail	-1.98	0.97	26.88	Fail	3.39	0.72
16	15.47	Fail	-2.28	0.97	18.16	8.84	-3.99	0.96	17.23	Fail	-2.64	0.96	27.08	Fail	2.67	0.72
17	16.78	7.06	-2.73	0.96	19.75	12.29	-5.31	0.96	18.81	Fail	-3.44	0.96	29.31	Fail	3.58	0.71
18	18.5	6.84	-2.61	0.97	21.87	11.43	-4.66	0.96	20.3	4.03	-1.88	0.97	32.63	Fail	3.52	0.72
19	18.8	6.84	-2.62	0.97	22.06	9.2	-4.27	0.96	20.95	8.1	-2.68	0.96	32.82	Fail	3.36	0.71
20	23.64	6.22	-2.6	0.96	27.9	12.32	-6.2	0.96	26.01	7.48	-3.23	0.96	41.64	Fail	3.11	0.71
21	23.84	6.94	-3.13	0.96	28.06	10.52	-5.58	0.95	26.51	7.67	-3.52	0.96	41.79	Fail	2.4	0.71
22	23.94	Fail	-3.11	0.96	28.12	13.76	-6.21	0.95	26.62	9.0	-3.69	0.96	41.92	Fail	2.74	0.71
23	40.62	11.52	-5.23	0.95	47.45	18.62	-10.04	0.94	44.95	11.81	-5.47	0.95	71.01	Fail	2.27	0.7
24	40.63	10.46	-5.49	0.95	47.55	16.44	-9.87	0.94	45.07	10.7	-5.42	0.95	71.1	-3.32	2.31	0.7
25	40.75	10.69	-5.26	0.96	47.67	20.81	-10.46	0.94	45.31	11.36	-5.67	0.95	71.36	-2.45	1.89	0.7
26	52.83	17.68	-8.52	0.95	61.6	23.43	-15.62	0.93	58.73	16.65	-8.29	0.95	92.04	1.22	-0.91	0.7
27	52.95	15.14	-8.76	0.95	61.84	23.71	-15.88	0.93	58.62	18.02	-8.78	0.95	92.14	Fail	-0.69	0.7
28	53.13	14.61	-7.55	0.95	62.02	22.27	-14.63	0.93	58.99	14.58	-7.38	0.95	92.85	Fail	1.35	0.7
29	78.57	19.12	-14.0	0.93	93.11	33.57	-28.87	0.9	87.99	20.46	-15.37	0.93	139.84	6.74	-5.67	0.69
30	80.06	17.78	-11.65	0.94	94.24	35.11	-27.84	0.9	87.74	18.6	-12.38	0.94	141.08	Fail	-2.81	0.69

Table B.27: Aggregated evaluation data for each sweet spot candidate of vp9_vaapi for GSCNN.

Index	libx264 - Coding parameters (HRNet)
1	aq-str: 1.092, crf: 3.0, preset: faster, tune: film, me_method: esa, coder: vlc, cmp: sad, aq-mode: variance, weightp: smart, b-pyr: strict, dir-pred: none, db-alpha: 2.0, db-beta: -2.0, weightb: 0.0, b_strat: 0.0, g: 600.0, keyint_min: 39.0, psy: 1.0, i-refresh: 1.0, trellis: 0.0, sc_th: 60.0, bf: 11.0, me_range: 23.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 0.0, 8x8dct: 0.0
2	aq-str: 1.142, crf: 3.0, preset: slower, tune: stillimage, me_method: umh, coder: ac, cmp: sad, aq-mode: none, weightp: simple, b-pyr: normal, dir-pred: none, db-alpha: 2.0, db-beta: -2.0, weightb: 0.0, b_strat: 1.0, g: 300.0, keyint_min: 31.0, psy: 0.0, i-refresh: 0.0, trellis: 1.0, sc_th: 12.0, bf: 6.0, me_range: 20.0, subq: 6.0, rc-la: 41.0, b-bias: 58.0, mix-refs: 0.0, 8x8dct: 0.0
3	aq-str: 1.125, crf: 3.0, preset: faster, tune: film, me_method: umh, coder: ac, cmp: sad, aq-mode: none, weightp: simple, b-pyr: normal, dir-pred: spatial, db-alpha: 1.0, db-beta: -2.0, weightb: 0.0, b_strat: 1.0, g: 300.0, keyint_min: 31.0, psy: 1.0, i-refresh: 0.0, trellis: 1.0, sc_th: 12.0, bf: 6.0, me_range: 20.0, subq: 6.0, rc-la: 41.0, b-bias: 58.0, mix-refs: 0.0, 8x8dct: 0.0
4	aq-str: 1.142, crf: 3.0, preset: faster, tune: film, me_method: umh, coder: vlc, cmp: sad, aq-mode: none, weightp: simple, b-pyr: normal, dir-pred: auto, db-alpha: -2.0, db-beta: -1.0, weightb: 0.0, b_strat: 1.0, g: 300.0, keyint_min: 31.0, psy: 1.0, i-refresh: 0.0, trellis: 0.0, sc_th: 12.0, bf: 6.0, me_range: 20.0, subq: 6.0, rc-la: 41.0, b-bias: 58.0, mix-refs: 0.0, 8x8dct: 0.0
5	aq-str: 0.668, crf: 7.0, preset: slow, tune: grain, me_method: tesa, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: normal, dir-pred: spatial, db-alpha: 2.0, db-beta: 1.0, weightb: 0.0, b_strat: 0.0, g: 600.0, keyint_min: 29.0, psy: 1.0, i-refresh: 1.0, trellis: 0.0, sc_th: 6.0, bf: 3.0, me_range: 23.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0
6	aq-str: 0.763, crf: 7.0, preset: faster, tune: stillimage, me_method: hex, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: strict, dir-pred: auto, db-alpha: 0.0, db-beta: 2.0, weightb: 1.0, b_strat: 1.0, g: 700.0, keyint_min: 26.0, psy: 1.0, i-refresh: 1.0, trellis: 0.0, sc_th: 9.0, bf: 9.0, me_range: 21.0, subq: 10.0, rc-la: 28.0, b-bias: -48.0, mix-refs: 0.0, 8x8dct: 1.0
7	aq-str: 0.195, crf: 9.0, preset: slow, tune: none, me_method: esa, coder: vlc, cmp: sad, aq-mode: autovariance, weightp: none, b-pyr: normal, dir-pred: auto, db-alpha: -2.0, db-beta: -1.0, weightb: 1.0, b_strat: 0.0, g: 500.0, keyint_min: 18.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 28.0, bf: 1.0, me_range: 17.0, subq: 1.0, rc-la: 43.0, b-bias: -5.0, mix-refs: 1.0, 8x8dct: 1.0
8	aq-str: 0.763, crf: 7.0, preset: faster, tune: stillimage, me_method: hex, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: strict, dir-pred: auto, db-alpha: 0.0, db-beta: 2.0, weightb: 1.0, b_strat: 1.0, g: 700.0, keyint_min: 26.0, psy: 1.0, i-refresh: 1.0, trellis: 0.0, sc_th: 6.0, bf: 3.0, me_range: 12.0, subq: 5.0, rc-la: 22.0, b-bias: 16.0, mix-refs: 0.0, 8x8dct: 1.0
9	aq-str: 0.289, crf: 10.0, preset: veryfast, tune: none, me_method: umh, coder: vlc, cmp: sad, aq-mode: none, weightp: none, b-pyr: normal, dir-pred: temporal, db-alpha: -1.0, db-beta: -2.0, weightb: 0.0, b_strat: 2.0, g: 500.0, keyint_min: 18.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 28.0, bf: 11.0, me_range: 17.0, subq: 1.0, rc-la: 43.0, b-bias: 7.0, mix-refs: 0.0, 8x8dct: 1.0
10	aq-str: 1.711, crf: 15.0, preset: medium, tune: stillimage, me_method: hex, coder: ac, cmp: chroma, aq-mode: variance, weightp: none, b-pyr: normal, dir-pred: temporal, db-alpha: 1.0, db-beta: 0.0, weightb: 0.0, b_strat: 2.0, g: 50.0, keyint_min: 34.0, psy: 0.0, i-refresh: 0.0, trellis: 1.0, sc_th: 0.0, bf: 10.0, me_range: 15.0, subq: 4.0, rc-la: 30.0, b-bias: -90.0, mix-refs: 1.0, 8x8dct: 0.0
11	aq-str: 1.711, crf: 15.0, preset: medium, tune: stillimage, me_method: hex, coder: ac, cmp: sad, aq-mode: variance, weightp: none, b-pyr: normal, dir-pred: temporal, db-alpha: 1.0, db-beta: 2.0, weightb: 0.0, b_strat: 0.0, g: 950.0, keyint_min: 34.0, psy: 0.0, i-refresh: 0.0, trellis: 1.0, sc_th: 0.0, bf: 10.0, me_range: 15.0, subq: 4.0, rc-la: 59.0, b-bias: 36.0, mix-refs: 1.0, 8x8dct: 0.0
12	aq-str: 0.384, crf: 13.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: none, weightp: simple, b-pyr: normal, dir-pred: auto, db-alpha: -2.0, db-beta: -1.0, weightb: 1.0, b_strat: 0.0, g: 1000.0, keyint_min: 40.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 48.0, bf: 12.0, me_range: 18.0, subq: 10.0, rc-la: 60.0, b-bias: 69.0, mix-refs: 0.0, 8x8dct: 0.0
13	aq-str: 0.384, crf: 13.0, preset: slower, tune: stillimage, me_method: umh, coder: ac, cmp: sad, aq-mode: none, weightp: simple, b-pyr: normal, dir-pred: auto, db-alpha: -2.0, db-beta: -1.0, weightb: 0.0, b_strat: 2.0, g: 500.0, keyint_min: 18.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 28.0, bf: 11.0, me_range: 18.0, subq: 10.0, rc-la: 60.0, b-bias: 69.0, mix-refs: 0.0, 8x8dct: 0.0
14	aq-str: 1.142, crf: 17.0, preset: fast, tune: stillimage, me_method: esa, coder: vlc, cmp: chroma, aq-mode: autovariance, weightp: none, b-pyr: normal, dir-pred: none, db-alpha: 0.0, db-beta: 1.0, weightb: 1.0, b_strat: 2.0, g: 750.0, keyint_min: 23.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 12.0, bf: 6.0, me_range: 20.0, subq: 6.0, rc-la: 41.0, b-bias: 58.0, mix-refs: 0.0, 8x8dct: 0.0
15	aq-str: 0.953, crf: 17.0, preset: fast, tune: stillimage, me_method: esa, coder: vlc, cmp: chroma, aq-mode: autovariance, weightp: none, b-pyr: normal, dir-pred: none, db-alpha: 0.0, db-beta: 1.0, weightb: 1.0, b_strat: 2.0, g: 750.0, keyint_min: 23.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 12.0, bf: 6.0, me_range: 20.0, subq: 6.0, rc-la: 41.0, b-bias: 58.0, mix-refs: 0.0, 8x8dct: 0.0
16	aq-str: 0.858, crf: 22.0, preset: slow, tune: grain, me_method: tesa, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: strict, dir-pred: none, db-alpha: -2.0, db-beta: 0.0, weightb: 1.0, b_strat: 2.0, g: 450.0, keyint_min: 27.0, psy: 0.0, i-refresh: 1.0, trellis: 0.0, sc_th: 51.0, bf: 16.0, me_range: 12.0, subq: 9.0, rc-la: 45.0, b-bias: -100.0, mix-refs: 0.0, 8x8dct: 1.0
17	aq-str: 0.858, crf: 22.0, preset: slow, tune: grain, me_method: tesa, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: strict, dir-pred: none, db-alpha: -2.0, db-beta: 0.0, weightb: 1.0, b_strat: 2.0, g: 450.0, keyint_min: 13.0, psy: 0.0, i-refresh: 1.0, trellis: 0.0, sc_th: 51.0, bf: 16.0, me_range: 12.0, subq: 8.0, rc-la: 45.0, b-bias: -100.0, mix-refs: 0.0, 8x8dct: 1.0
18	aq-str: 0.858, crf: 22.0, preset: slow, tune: grain, me_method: tesa, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: strict, dir-pred: none, db-alpha: -2.0, db-beta: 0.0, weightb: 1.0, b_strat: 1.0, g: 300.0, keyint_min: 31.0, psy: 1.0, i-refresh: 0.0, trellis: 1.0, sc_th: 12.0, bf: 6.0, me_range: 20.0, subq: 6.0, rc-la: 45.0, b-bias: -100.0, mix-refs: 0.0, 8x8dct: 1.0
19	aq-str: 0.858, crf: 22.0, preset: slow, tune: grain, me_method: tesa, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: normal, dir-pred: spatial, db-alpha: 1.0, db-beta: 0.0, weightb: 0.0, b_strat: 2.0, g: 50.0, keyint_min: 36.0, psy: 0.0, i-refresh: 1.0, trellis: 0.0, sc_th: 6.0, bf: 9.0, me_range: 12.0, subq: 9.0, rc-la: 29.0, b-bias: 37.0, mix-refs: 0.0, 8x8dct: 0.0
20	aq-str: 1.521, crf: 26.0, preset: slower, tune: grain, me_method: dia, coder: ac, cmp: chroma, aq-mode: autovariance, weightp: simple, b-pyr: none, dir-pred: spatial, db-alpha: 1.0, db-beta: 0.0, weightb: 0.0, b_strat: 2.0, g: 50.0, keyint_min: 36.0, psy: 0.0, i-refresh: 0.0, trellis: 0.0, sc_th: 41.0, bf: 4.0, me_range: 16.0, subq: 9.0, rc-la: 52.0, b-bias: -5.0, mix-refs: 1.0, 8x8dct: 0.0
21	aq-str: 1.521, crf: 26.0, preset: slower, tune: grain, me_method: dia, coder: ac, cmp: chroma, aq-mode: autovariance, weightp: simple, b-pyr: none, dir-pred: spatial, db-alpha: -1.0, db-beta: 0.0, weightb: 0.0, b_strat: 0.0, g: 600.0, keyint_min: 29.0, psy: 0.0, i-refresh: 1.0, trellis: 0.0, sc_th: 6.0, bf: 3.0, me_range: 12.0, subq: 8.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0
22	aq-str: 0.289, crf: 26.0, preset: veryfast, tune: grain, me_method: tesa, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: normal, dir-pred: spatial, db-alpha: 1.0, db-beta: -2.0, weightb: 0.0, b_strat: 1.0, g: 950.0, keyint_min: 32.0, psy: 0.0, i-refresh: 0.0, trellis: 0.0, sc_th: 41.0, bf: 7.0, me_range: 12.0, subq: 9.0, rc-la: 52.0, b-bias: -16.0, mix-refs: 1.0, 8x8dct: 1.0
23	aq-str: 0.289, crf: 26.0, preset: veryfast, tune: grain, me_method: tesa, coder: ac, cmp: chroma, aq-mode: variance, weightp: smart, b-pyr: strict, dir-pred: none, db-alpha: 1.0, db-beta: -2.0, weightb: 0.0, b_strat: 1.0, g: 950.0, keyint_min: 32.0, psy: 0.0, i-refresh: 0.0, trellis: 0.0, sc_th: 41.0, bf: 7.0, me_range: 12.0, subq: 9.0, rc-la: 52.0, b-bias: 37.0, mix-refs: 1.0, 8x8dct: 1.0
24	aq-str: 0.289, crf: 26.0, preset: veryfast, tune: grain, me_method: tesa, coder: ac, cmp: chroma, aq-mode: autovariance, weightp: simple, b-pyr: none, dir-pred: spatial, db-alpha: 1.0, db-beta: 2.0, weightb: 0.0, b_strat: 2.0, g: 950.0, keyint_min: 32.0, psy: 0.0, i-refresh: 0.0, trellis: 0.0, sc_th: 41.0, bf: 7.0, me_range: 12.0, subq: 9.0, rc-la: 52.0, b-bias: 37.0, mix-refs: 1.0, 8x8dct: 1.0
25	aq-str: 0.668, crf: 28.0, preset: slow, tune: grain, me_method: tesa, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: strict, dir-pred: none, db-alpha: -2.0, db-beta: 0.0, weightb: 0.0, b_strat: 2.0, g: 100.0, keyint_min: 39.0, psy: 1.0, i-refresh: 1.0, trellis: 0.0, sc_th: 60.0, bf: 11.0, me_range: 23.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0
26	aq-str: 0.668, crf: 28.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: variance, weightp: smart, b-pyr: strict, dir-pred: none, db-alpha: 2.0, db-beta: -2.0, weightb: 0.0, b_strat: 1.0, g: 300.0, keyint_min: 31.0, psy: 1.0, i-refresh: 0.0, trellis: 1.0, sc_th: 12.0, bf: 6.0, me_range: 20.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0
27	aq-str: 0.668, crf: 28.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: variance, weightp: smart, b-pyr: strict, dir-pred: none, db-alpha: 2.0, db-beta: -2.0, weightb: 1.0, b_strat: 1.0, g: 300.0, keyint_min: 37.0, psy: 1.0, i-refresh: 0.0, trellis: 1.0, sc_th: 12.0, bf: 6.0, me_range: 20.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0
28	aq-str: 0.668, crf: 28.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: autovariance, weightp: simple, b-pyr: none, dir-pred: spatial, db-alpha: 1.0, db-beta: 0.0, weightb: 0.0, b_strat: 2.0, g: 50.0, keyint_min: 36.0, psy: 1.0, i-refresh: 0.0, trellis: 1.0, sc_th: 12.0, bf: 6.0, me_range: 20.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0
29	aq-str: 0.668, crf: 28.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: variance, weightp: smart, b-pyr: strict, dir-pred: none, db-alpha: 2.0, db-beta: -2.0, weightb: 0.0, b_strat: 1.0, g: 300.0, keyint_min: 37.0, psy: 1.0, i-refresh: 0.0, trellis: 0.0, sc_th: 12.0, bf: 6.0, me_range: 20.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0
30	aq-str: 0.668, crf: 28.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: none, dir-pred: spatial, db-alpha: 2.0, db-beta: 1.0, weightb: 0.0, b_strat: 0.0, g: 600.0, keyint_min: 29.0, psy: 1.0, i-refresh: 1.0, trellis: 0.0, sc_th: 6.0, bf: 3.0, me_range: 12.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0
31	aq-str: 0.668, crf: 28.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: none, dir-pred: temporal, db-alpha: -1.0, db-beta: 1.0, weightb: 0.0, b_strat: 2.0, g: 50.0, keyint_min: 36.0, psy: 0.0, i-refresh: 0.0, trellis: 0.0, sc_th: 41.0, bf: 4.0, me_range: 16.0, subq: 9.0, rc-la: 52.0, b-bias: -5.0, mix-refs: 1.0, 8x8dct: 0.0

Table B.28: The coding parameters of each sweet spot candidate of libx264 for HRNet.

hrnet Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	3.17	Fail	-0.05	0.99	3.54	0.98	-0.07	0.98	3.44	0.63	-0.04	0.99	1.71	Fail	0.63	0.98
2	3.32	1.84	-0.09	0.99	3.41	2.1	-0.11	0.99	3.55	Fail	-0.13	0.99	1.57	Fail	0.24	0.99
3	3.9	2.54	-0.15	0.98	3.99	1.45	-0.09	0.98	4.13	2.83	-0.15	0.99	1.76	Fail	0.01	0.98
4	4.07	Fail	-0.2	0.98	4.2	Fail	-0.15	0.98	4.29	Fail	-0.17	0.98	1.67	Fail	0.93	0.98
5	4.1	Fail	-0.11	0.98	4.43	1.78	-0.16	0.98	4.46	Fail	-0.0	0.98	2.09	Fail	-0.05	0.97
6	5.52	2.52	-0.21	0.98	6.04	1.78	-0.18	0.98	5.89	1.36	-0.12	0.98	2.47	1.23	-0.07	0.96
7	6.27	2.6	-0.24	0.97	6.65	Fail	-0.33	0.97	6.82	3.53	-0.43	0.97	2.11	Fail	1.36	0.95
8	7.4	3.14	-0.22	0.98	7.58	2.57	-0.27	0.98	7.35	Fail	-0.24	0.98	3.06	Fail	0.59	0.93
9	7.87	Fail	-0.54	0.97	8.02	3.01	-0.5	0.97	7.58	4.6	-0.51	0.97	2.33	Fail	1.03	0.95
10	13.61	Fail	-0.76	0.97	15.01	4.79	-0.84	0.97	13.74	Fail	-0.7	0.97	4.53	Fail	-0.35	0.89
11	14.72	Fail	-0.77	0.97	15.5	Fail	-1.07	0.97	14.48	Fail	-1.02	0.97	5.06	-2.94	0.51	0.87
12	15.88	Fail	-0.65	0.98	15.12	Fail	-1.03	0.98	15.69	Fail	-0.7	0.98	3.61	4.84	-0.54	0.95
13	16.38	4.54	-0.6	0.97	15.58	5.89	-1.19	0.97	16.71	Fail	-0.86	0.97	3.65	7.44	-0.75	0.93
14	25.69	Fail	-1.19	0.97	25.13	7.9	-2.08	0.96	23.91	Fail	-1.34	0.97	6.53	-7.42	1.36	0.83
15	26.65	Fail	-1.37	0.97	25.74	8.39	-2.31	0.96	24.8	Fail	-1.54	0.97	6.85	-7.83	1.62	0.82
16	37.21	8.2	-1.58	0.96	33.02	7.64	-2.05	0.96	36.24	8.14	-1.58	0.96	14.89	1.42	-0.27	0.77
17	37.24	Fail	-1.62	0.96	33.04	Fail	-1.8	0.96	36.41	8.76	-1.49	0.96	14.74	0.43	-0.09	0.77
18	43.42	6.75	-1.65	0.96	41.15	7.13	-1.89	0.95	39.43	8.06	-1.74	0.96	36.05	-5.16	1.48	0.73
19	45.99	Fail	-1.91	0.96	38.61	9.49	-3.03	0.96	40.42	7.87	-2.14	0.96	26.24	-5.48	1.47	0.74
20	72.31	10.69	-3.18	0.96	57.58	15.16	-5.21	0.95	59.54	12.46	-3.5	0.95	65.86	-6.63	2.36	0.72
21	77.83	9.86	-3.0	0.95	59.14	12.1	-4.49	0.95	66.29	11.41	-3.83	0.95	76.08	-5.69	2.29	0.7
22	80.22	13.54	-3.86	0.95	64.13	16.92	-6.61	0.95	69.66	15.26	-4.25	0.95	106.01	Fail	2.41	0.71
23	82.91	12.53	-4.24	0.95	62.67	16.24	-6.95	0.95	72.89	13.03	-4.99	0.95	100.79	Fail	1.91	0.71
24	86.78	Fail	-4.17	0.95	63.48	16.46	-6.81	0.95	74.78	14.93	-4.94	0.95	104.73	-5.98	2.79	0.71
25	105.15	16.46	-6.06	0.94	82.61	17.37	-8.27	0.94	92.81	19.21	-7.21	0.94	151.72	2.84	-1.17	0.7
26	138.18	16.21	-7.82	0.93	100.57	16.59	-10.17	0.92	123.04	20.8	-9.39	0.93	240.7	6.84	-2.19	0.69
27	138.21	15.69	-7.59	0.93	100.57	17.3	-10.28	0.92	122.92	18.09	-9.72	0.93	240.72	6.15	-1.86	0.69
28	155.56	18.79	-8.19	0.93	104.63	19.7	-10.83	0.92	135.64	20.37	-10.23	0.93	259.26	7.37	-2.88	0.69
29	168.38	17.88	-9.62	0.93	113.94	18.8	-12.5	0.92	145.44	22.27	-12.38	0.92	305.27	12.2	-5.34	0.68
30	188.24	19.35	-10.58	0.92	118.92	19.75	-14.01	0.91	159.82	19.92	-12.8	0.92	339.8	16.04	-6.53	0.68
31	189.48	19.73	-10.42	0.93	121.62	21.25	-14.52	0.92	152.44	21.39	-12.03	0.93	334.26	10.15	-5.59	0.68

Table B.29: Aggregated evaluation data for each sweet spot candidate of libx264 for HRNet.

Index	libx264 - Coding parameters (GSCNN)
1	aq-str: 1.237, crf: 1.0, preset: slow, tune: stillimage, me_method: hex, coder: ac, cmp: chroma, aq-mode: autovariance, weightp: smart, b-pyr: none, dir-pred: none, db-alpha: 1.0, db-beta: 2.0, weightb: 1.0, b_strat: 2.0, g: 350.0, keyint_min: 10.0, psy: 0.0, i-refresh: 0.0, trellis: 1.0, sc_th: 41.0, bf: 4.0, me_range: 16.0, subq: 9.0, rc-la: 52.0, b-bias: 26.0, mix-refs: 0.0, 8x8dct: 1.0
2	aq-str: 1.237, crf: 1.0, preset: slow, tune: stillimage, me_method: hex, coder: ac, cmp: chroma, aq-mode: autovariance, weightp: smart, b-pyr: none, dir-pred: none, db-alpha: 1.0, db-beta: 2.0, weightb: 1.0, b_strat: 2.0, g: 350.0, keyint_min: 10.0, psy: 0.0, i-refresh: 0.0, trellis: 1.0, sc_th: 41.0, bf: 4.0, me_range: 16.0, subq: 9.0, rc-la: 52.0, b-bias: 26.0, mix-refs: 1.0, 8x8dct: 1.0
3	aq-str: 1.501, crf: 8.0, preset: slower, tune: grain, me_method: dia, coder: ac, cmp: chroma, aq-mode: autovariance, weightp: simple, b-pyr: none, dir-pred: spatial, db-alpha: 1.0, db-beta: 2.0, weightb: 1.0, b_strat: 2.0, g: 500.0, keyint_min: 18.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 28.0, bf: 1.0, me_range: 17.0, subq: 1.0, rc-la: 43.0, b-bias: -5.0, mix-refs: 1.0, 8x8dct: 0.0
4	aq-str: 1.142, crf: 3.0, preset: faster, tune: film, me_method: esa, coder: vlc, cmp: chroma, aq-mode: variance, weightp: smart, b-pyr: strict, dir-pred: none, db-alpha: 2.0, db-beta: -2.0, weightb: 0.0, b_strat: 1.0, g: 300.0, keyint_min: 31.0, psy: 1.0, i-refresh: 0.0, trellis: 1.0, sc_th: 12.0, bf: 6.0, me_range: 20.0, subq: 6.0, rc-la: 41.0, b-bias: 58.0, mix-refs: 0.0, 8x8dct: 0.0
5	aq-str: 0.479, crf: 11.0, preset: medium, tune: grain, me_method: dia, coder: ac, cmp: sad, aq-mode: none, weightp: simple, b-pyr: normal, dir-pred: spatial, db-alpha: -2.0, db-beta: 1.0, weightb: 0.0, b_strat: 0.0, g: 900.0, keyint_min: 21.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 32.0, bf: 5.0, me_range: 22.0, subq: 3.0, rc-la: 32.0, b-bias: 5.0, mix-refs: 1.0, 8x8dct: 1.0
6	aq-str: 1.711, crf: 15.0, preset: medium, tune: stillimage, me_method: hex, coder: ac, cmp: chroma, aq-mode: variance, weightp: none, b-pyr: normal, dir-pred: temporal, db-alpha: 1.0, db-beta: 0.0, weightb: 0.0, b_strat: 2.0, g: 50.0, keyint_min: 34.0, psy: 0.0, i-refresh: 0.0, trellis: 1.0, sc_th: 16.0, bf: 8.0, me_range: 19.0, subq: 9.0, rc-la: 48.0, b-bias: -16.0, mix-refs: 0.0, 8x8dct: 1.0
7	aq-str: 1.711, crf: 15.0, preset: medium, tune: stillimage, me_method: hex, coder: ac, cmp: chroma, aq-mode: autovariance, weightp: smart, b-pyr: normal, dir-pred: none, db-alpha: 1.0, db-beta: 2.0, weightb: 1.0, b_strat: 2.0, g: 350.0, keyint_min: 34.0, psy: 0.0, i-refresh: 0.0, trellis: 1.0, sc_th: 0.0, bf: 10.0, me_range: 15.0, subq: 4.0, rc-la: 30.0, b-bias: -90.0, mix-refs: 1.0, 8x8dct: 0.0
8	aq-str: 0.289, crf: 16.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: autovariance, weightp: simple, b-pyr: none, dir-pred: spatial, db-alpha: 1.0, db-beta: 2.0, weightb: 1.0, b_strat: 2.0, g: 249.0, keyint_min: 18.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 28.0, bf: 11.0, me_range: 17.0, subq: 1.0, rc-la: 52.0, b-bias: -34.0, mix-refs: 0.0, 8x8dct: 1.0
9	aq-str: 0.289, crf: 16.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: autovariance, weightp: simple, b-pyr: none, dir-pred: spatial, db-alpha: 1.0, db-beta: 2.0, weightb: 1.0, b_strat: 2.0, g: 249.0, keyint_min: 18.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 28.0, bf: 11.0, me_range: 17.0, subq: 1.0, rc-la: 43.0, b-bias: 7.0, mix-refs: 0.0, 8x8dct: 1.0
10	aq-str: 0.953, crf: 17.0, preset: fast, tune: stillimage, me_method: esa, coder: vlc, cmp: chroma, aq-mode: autovariance, weightp: none, b-pyr: normal, dir-pred: none, db-alpha: 0.0, db-beta: 1.0, weightb: 1.0, b_strat: 1.0, g: 750.0, keyint_min: 23.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 16.0, bf: 8.0, me_range: 19.0, subq: 9.0, rc-la: 48.0, b-bias: -16.0, mix-refs: 0.0, 8x8dct: 1.0
11	aq-str: 1.805, crf: 19.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: none, dir-pred: spatial, db-alpha: -2.0, db-beta: 1.0, weightb: 0.0, b_strat: 2.0, g: 100.0, keyint_min: 32.0, psy: 0.0, i-refresh: 0.0, trellis: 0.0, sc_th: 3.0, bf: 15.0, me_range: 22.0, subq: 4.0, rc-la: 35.0, b-bias: 79.0, mix-refs: 0.0, 8x8dct: 0.0
12	aq-str: 0.808, crf: 22.0, preset: slow, tune: grain, me_method: tesa, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: strict, dir-pred: none, db-alpha: -2.0, db-beta: 0.0, weightb: 1.0, b_strat: 2.0, g: 450.0, keyint_min: 27.0, psy: 0.0, i-refresh: 1.0, trellis: 1.0, sc_th: 28.0, bf: 5.0, me_range: 17.0, subq: 1.0, rc-la: 45.0, b-bias: -100.0, mix-refs: 0.0, 8x8dct: 1.0
13	aq-str: 1.237, crf: 20.0, preset: slow, tune: stillimage, me_method: hex, coder: ac, cmp: chroma, aq-mode: autovariance, weightp: smart, b-pyr: none, dir-pred: none, db-alpha: 1.0, db-beta: 2.0, weightb: 1.0, b_strat: 2.0, g: 500.0, keyint_min: 18.0, psy: 0.0, i-refresh: 0.0, trellis: 0.0, sc_th: 60.0, bf: 11.0, me_range: 16.0, subq: 9.0, rc-la: 52.0, b-bias: -31.0, mix-refs: 0.0, 8x8dct: 1.0
14	aq-str: 1.521, crf: 26.0, preset: slower, tune: grain, me_method: dia, coder: ac, cmp: chroma, aq-mode: autovariance, weightp: simple, b-pyr: normal, dir-pred: spatial, db-alpha: 1.0, db-beta: 2.0, weightb: 0.0, b_strat: 0.0, g: 950.0, keyint_min: 36.0, psy: 0.0, i-refresh: 0.0, trellis: 0.0, sc_th: 41.0, bf: 4.0, me_range: 16.0, subq: 9.0, rc-la: 52.0, b-bias: -5.0, mix-refs: 1.0, 8x8dct: 0.0
15	aq-str: 1.521, crf: 26.0, preset: slower, tune: grain, me_method: esa, coder: ac, cmp: sad, aq-mode: autovariance, weightp: simple, b-pyr: normal, dir-pred: spatial, db-alpha: 1.0, db-beta: 2.0, weightb: 0.0, b_strat: 0.0, g: 950.0, keyint_min: 36.0, psy: 0.0, i-refresh: 1.0, trellis: 0.0, sc_th: 0.0, bf: 4.0, me_range: 16.0, subq: 9.0, rc-la: 52.0, b-bias: -5.0, mix-refs: 1.0, 8x8dct: 0.0
16	aq-str: 0.858, crf: 26.0, preset: slower, tune: grain, me_method: tesa, coder: ac, cmp: chroma, aq-mode: autovariance, weightp: simple, b-pyr: normal, dir-pred: spatial, db-alpha: 1.0, db-beta: 2.0, weightb: 0.0, b_strat: 0.0, g: 100.0, keyint_min: 39.0, psy: 0.0, i-refresh: 0.0, trellis: 1.0, sc_th: 18.0, bf: 4.0, me_range: 16.0, subq: 9.0, rc-la: 52.0, b-bias: 37.0, mix-refs: 1.0, 8x8dct: 1.0
17	aq-str: 0.668, crf: 28.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: none, dir-pred: auto, db-alpha: -2.0, db-beta: -1.0, weightb: 1.0, b_strat: 2.0, g: 500.0, keyint_min: 25.0, psy: 1.0, i-refresh: 1.0, trellis: 1.0, sc_th: 60.0, bf: 11.0, me_range: 23.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0
18	aq-str: 0.668, crf: 28.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: none, dir-pred: auto, db-alpha: -2.0, db-beta: -1.0, weightb: 1.0, b_strat: 2.0, g: 350.0, keyint_min: 10.0, psy: 0.0, i-refresh: 0.0, trellis: 1.0, sc_th: 41.0, bf: 4.0, me_range: 20.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0
19	aq-str: 0.668, crf: 28.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: variance, weightp: simple, b-pyr: none, dir-pred: spatial, db-alpha: -2.0, db-beta: 1.0, weightb: 0.0, b_strat: 2.0, g: 100.0, keyint_min: 39.0, psy: 1.0, i-refresh: 1.0, trellis: 0.0, sc_th: 60.0, bf: 11.0, me_range: 23.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0
20	aq-str: 0.668, crf: 28.0, preset: veryfast, tune: none, me_method: umh, coder: ac, cmp: sad, aq-mode: variance, weightp: smart, b-pyr: normal, dir-pred: spatial, db-alpha: 1.0, db-beta: 2.0, weightb: 0.0, b_strat: 2.0, g: 100.0, keyint_min: 39.0, psy: 1.0, i-refresh: 1.0, trellis: 0.0, sc_th: 21.0, bf: 11.0, me_range: 23.0, subq: 7.0, rc-la: 54.0, b-bias: 48.0, mix-refs: 1.0, 8x8dct: 1.0

Table B.30: The coding parameters of each sweet spot candidate of libx264 for GSCNN.

gscnn Sweet spot	Orig: CR	Orig: T	Orig: err	Orig: PSNR	Rdeg: CR	Rdeg: T	Rdeg: err	Rdeg: PSNR	Mdeg: CR	Mdeg: T	Mdeg: err	Mdeg: PSNR	Ndeg: CR	Ndeg: T	Ndeg: err	Ndeg: PSNR
1	1.92	Fail	0.07	0.99	2.13	Fail	0.11	0.99	2.09	Fail	-0.19	0.99	1.36	Fail	-0.03	0.99
2	1.92	Fail	0.18	0.99	2.13	Fail	-0.08	0.99	2.1	Fail	0.01	0.99	1.36	3.59	-0.12	0.99
3	2.66	Fail	0.0	0.99	2.98	Fail	-0.3	0.99	2.81	Fail	0.08	0.99	1.65	Fail	-0.06	0.99
4	3.09	Fail	-0.27	0.99	3.46	Fail	0.15	0.98	3.3	Fail	-0.09	0.99	1.69	Fail	1.44	0.98
5	7.47	Fail	-0.91	0.98	7.38	Fail	-0.72	0.98	7.77	Fail	-0.59	0.98	2.17	Fail	-0.2	0.97
6	13.89	3.14	-1.07	0.97	15.72	5.29	-1.61	0.97	13.63	3.04	-0.9	0.97	4.8	-3.07	1.15	0.9
7	15.52	2.51	-0.94	0.97	16.57	5.38	-1.84	0.97	15.59	2.44	-0.75	0.97	3.87	-2.56	0.83	0.92
8	20.07	3.05	-1.39	0.97	19.44	6.1	-2.39	0.96	21.23	5.35	-1.79	0.97	4.72	Fail	0.68	0.87
9	23.14	5.04	-1.93	0.97	21.97	5.68	-2.36	0.96	21.84	5.67	-2.27	0.97	5.73	-6.69	2.32	0.84
10	26.61	Fail	-2.07	0.97	27.61	Fail	-3.54	0.96	23.81	7.93	-2.38	0.97	7.12	-10.76	5.15	0.82
11	31.3	6.82	-2.93	0.96	29.88	13.8	-4.81	0.96	29.26	9.3	-3.74	0.96	14.74	Fail	5.72	0.77
12	38.52	6.17	-2.39	0.96	34.44	8.53	-3.8	0.95	35.4	8.6	-3.57	0.96	24.63	Fail	7.91	0.74
13	53.57	9.38	-3.56	0.96	45.94	13.39	-5.51	0.96	49.62	8.65	-3.47	0.96	35.43	-14.14	9.26	0.73
14	77.19	13.48	-6.0	0.95	59.36	24.99	-9.57	0.95	68.09	17.74	-7.38	0.95	83.7	-9.31	6.67	0.7
15	78.17	13.93	-5.77	0.95	63.15	23.11	-9.68	0.95	69.46	15.39	-7.32	0.95	88.56	Fail	6.06	0.7
16	90.97	13.83	-6.44	0.95	74.62	20.82	-8.99	0.95	82.05	13.75	-7.42	0.95	135.76	-10.53	6.88	0.7
17	158.98	31.55	-13.42	0.93	106.96	27.91	-18.5	0.92	140.61	25.89	-17.23	0.93	267.91	7.41	-5.32	0.68
18	161.99	24.71	-14.58	0.93	111.2	25.44	-19.31	0.92	141.88	26.8	-17.88	0.93	284.52	7.44	-5.37	0.69
19	183.24	32.06	-16.37	0.93	117.51	29.66	-21.68	0.92	157.61	27.41	-20.26	0.92	315.61	12.98	-9.92	0.68
20	187.39	29.64	-16.08	0.93	120.48	29.27	-21.25	0.92	162.03	28.3	-19.72	0.92	327.65	12.01	-8.73	0.68

Table B.31: Aggregated evaluation data for each sweet spot candidate of libx264 for GSCNN.

B.6 GSCNN's reduced validation set

The names of the label files used for the reduced GSCNN validation set. The GSCNN validation set is a subset of the full validation set.

- frankfurt_000000_000576_leftImg8bit.png
- frankfurt_000000_002196_leftImg8bit.png
- frankfurt_000000_003357_leftImg8bit.png
- frankfurt_000000_003920_leftImg8bit.png
- frankfurt_000000_005543_leftImg8bit.png
- frankfurt_000000_005898_leftImg8bit.png
- frankfurt_000000_008206_leftImg8bit.png
- frankfurt_000000_009688_leftImg8bit.png
- frankfurt_000000_011007_leftImg8bit.png
- frankfurt_000000_013240_leftImg8bit.png
- frankfurt_000000_013382_leftImg8bit.png
- frankfurt_000000_019607_leftImg8bit.png
- frankfurt_000000_020880_leftImg8bit.png
- frankfurt_000000_022254_leftImg8bit.png
- frankfurt_000001_002512_leftImg8bit.png
- frankfurt_000001_002646_leftImg8bit.png
- frankfurt_000001_004736_leftImg8bit.png
- frankfurt_000001_004859_leftImg8bit.png
- frankfurt_000001_005410_leftImg8bit.png
- frankfurt_000001_007973_leftImg8bit.png
- frankfurt_000001_008200_leftImg8bit.png
- frankfurt_000001_008688_leftImg8bit.png
- frankfurt_000001_009058_leftImg8bit.png
- frankfurt_000001_010156_leftImg8bit.png
- frankfurt_000001_010600_leftImg8bit.png
- frankfurt_000001_010830_leftImg8bit.png
- frankfurt_000001_011835_leftImg8bit.png
- frankfurt_000001_012519_leftImg8bit.png
- frankfurt_000001_012699_leftImg8bit.png
- frankfurt_000001_013016_leftImg8bit.png
- frankfurt_000001_013496_leftImg8bit.png
- frankfurt_000001_014565_leftImg8bit.png
- frankfurt_000001_019698_leftImg8bit.png
- frankfurt_000001_020693_leftImg8bit.png
- frankfurt_000001_021406_leftImg8bit.png
- frankfurt_000001_028232_leftImg8bit.png
- frankfurt_000001_029236_leftImg8bit.png
- frankfurt_000001_034816_leftImg8bit.png
- frankfurt_000001_041074_leftImg8bit.png
- frankfurt_000001_044227_leftImg8bit.png
- frankfurt_000001_050149_leftImg8bit.png
- frankfurt_000001_051807_leftImg8bit.png
- frankfurt_000001_054884_leftImg8bit.png
- frankfurt_000001_057954_leftImg8bit.png
- frankfurt_000001_059119_leftImg8bit.png
- frankfurt_000001_060906_leftImg8bit.png
- frankfurt_000001_062016_leftImg8bit.png
- frankfurt_000001_062396_leftImg8bit.png
- frankfurt_000001_065160_leftImg8bit.png
- frankfurt_000001_066092_leftImg8bit.png
- frankfurt_000001_067735_leftImg8bit.png
- frankfurt_000001_068063_leftImg8bit.png
- frankfurt_000001_070099_leftImg8bit.png
- frankfurt_000001_072155_leftImg8bit.png
- frankfurt_000001_075296_leftImg8bit.png
- frankfurt_000001_075984_leftImg8bit.png
- frankfurt_000001_079206_leftImg8bit.png
- frankfurt_000001_083029_leftImg8bit.png
- frankfurt_000001_083852_leftImg8bit.png
- lindau_000000_000019_leftImg8bit.png
- lindau_000001_000019_leftImg8bit.png
- lindau_000002_000019_leftImg8bit.png
- lindau_000003_000019_leftImg8bit.png
- lindau_000004_000019_leftImg8bit.png
- lindau_000005_000019_leftImg8bit.png
- lindau_000006_000019_leftImg8bit.png
- lindau_000007_000019_leftImg8bit.png
- lindau_000008_000019_leftImg8bit.png
- lindau_000009_000019_leftImg8bit.png
- lindau_000010_000019_leftImg8bit.png
- lindau_000011_000019_leftImg8bit.png
- lindau_000012_000019_leftImg8bit.png
- lindau_000013_000019_leftImg8bit.png
- lindau_000014_000019_leftImg8bit.png
- lindau_000015_000019_leftImg8bit.png
- lindau_000016_000019_leftImg8bit.png
- lindau_000017_000019_leftImg8bit.png
- lindau_000018_000019_leftImg8bit.png
- lindau_000019_000019_leftImg8bit.png
- lindau_000020_000019_leftImg8bit.png
- lindau_000021_000019_leftImg8bit.png
- lindau_000022_000019_leftImg8bit.png
- lindau_000023_000019_leftImg8bit.png
- lindau_000024_000019_leftImg8bit.png
- lindau_000025_000019_leftImg8bit.png
- lindau_000026_000019_leftImg8bit.png
- lindau_000027_000019_leftImg8bit.png
- lindau_000028_000019_leftImg8bit.png

- lindau_000029_000019_leftImg8bit.png
- lindau_000030_000019_leftImg8bit.png
- lindau_000031_000019_leftImg8bit.png
- lindau_000032_000019_leftImg8bit.png
- lindau_000033_000019_leftImg8bit.png
- lindau_000034_000019_leftImg8bit.png
- lindau_000035_000019_leftImg8bit.png
- lindau_000036_000019_leftImg8bit.png
- lindau_000037_000019_leftImg8bit.png
- lindau_000038_000019_leftImg8bit.png
- lindau_000039_000019_leftImg8bit.png
- lindau_000040_000019_leftImg8bit.png
- lindau_000041_000019_leftImg8bit.png
- lindau_000042_000019_leftImg8bit.png
- lindau_000043_000019_leftImg8bit.png
- lindau_000044_000019_leftImg8bit.png
- lindau_000045_000019_leftImg8bit.png
- lindau_000046_000019_leftImg8bit.png
- lindau_000047_000019_leftImg8bit.png
- lindau_000048_000019_leftImg8bit.png
- lindau_000049_000019_leftImg8bit.png
- lindau_000050_000019_leftImg8bit.png
- lindau_000051_000019_leftImg8bit.png
- lindau_000052_000019_leftImg8bit.png
- lindau_000053_000019_leftImg8bit.png
- lindau_000054_000019_leftImg8bit.png
- lindau_000055_000019_leftImg8bit.png
- lindau_000056_000019_leftImg8bit.png
- lindau_000057_000019_leftImg8bit.png
- lindau_000058_000019_leftImg8bit.png
- munster_000001_000019_leftImg8bit.png
- munster_000003_000019_leftImg8bit.png
- munster_000006_000019_leftImg8bit.png
- munster_000007_000019_leftImg8bit.png
- munster_000012_000019_leftImg8bit.png
- munster_000015_000019_leftImg8bit.png
- munster_000016_000019_leftImg8bit.png
- munster_000019_000019_leftImg8bit.png
- munster_000023_000019_leftImg8bit.png
- munster_000024_000019_leftImg8bit.png
- munster_000033_000019_leftImg8bit.png
- munster_000035_000019_leftImg8bit.png
- munster_000036_000019_leftImg8bit.png
- munster_000044_000019_leftImg8bit.png
- munster_000046_000019_leftImg8bit.png
- munster_000049_000019_leftImg8bit.png
- munster_000050_000019_leftImg8bit.png
- munster_000052_000019_leftImg8bit.png
- munster_000053_000019_leftImg8bit.png
- munster_000056_000019_leftImg8bit.png
- munster_000059_000019_leftImg8bit.png
- munster_000061_000019_leftImg8bit.png
- munster_000064_000019_leftImg8bit.png
- munster_000069_000019_leftImg8bit.png
- munster_000071_000019_leftImg8bit.png
- munster_000072_000019_leftImg8bit.png
- munster_000080_000019_leftImg8bit.png
- munster_000085_000019_leftImg8bit.png
- munster_000086_000019_leftImg8bit.png
- munster_000087_000019_leftImg8bit.png
- munster_000090_000019_leftImg8bit.png
- munster_000091_000019_leftImg8bit.png
- munster_000099_000019_leftImg8bit.png
- munster_000100_000019_leftImg8bit.png
- munster_000101_000019_leftImg8bit.png
- munster_000103_000019_leftImg8bit.png
- munster_000104_000019_leftImg8bit.png
- munster_000107_000019_leftImg8bit.png
- munster_000109_000019_leftImg8bit.png
- munster_000110_000019_leftImg8bit.png
- munster_000118_000019_leftImg8bit.png
- munster_000119_000019_leftImg8bit.png
- munster_000121_000019_leftImg8bit.png
- munster_000123_000019_leftImg8bit.png
- munster_000124_000019_leftImg8bit.png
- munster_000125_000019_leftImg8bit.png
- munster_000131_000019_leftImg8bit.png
- munster_000134_000019_leftImg8bit.png
- munster_000136_000019_leftImg8bit.png
- munster_000142_000019_leftImg8bit.png
- munster_000149_000019_leftImg8bit.png
- munster_000154_000019_leftImg8bit.png
- munster_000158_000019_leftImg8bit.png
- munster_000159_000019_leftImg8bit.png
- munster_000163_000019_leftImg8bit.png
- munster_000164_000019_leftImg8bit.png
- munster_000167_000019_leftImg8bit.png
- munster_000170_000019_leftImg8bit.png
- munster_000172_000019_leftImg8bit.png