CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Effects of Video Compression formats on Neural Network Performance

Bachelor of Science Thesis in Software Engineering and Management

Marko Stanoevich
Jonathan Partain

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Supervisor: CHRISTIAN BERGER
Examiner: Richard Berntsson Svensson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Effects of Video Compression formats on Neural Network Performance

Marko Stanoevich
Department of Computer Science
and Engineering
University of Gothenburg
Gothenburg, Sweden
gusstanoma@student.gu.se

Jonathan Partain
Department of Computer Science
and Engineering
University of Gothenburg
Gothenburg, Sweden
gusparjo@student.gu.se

***Abstract*—The field of autonomous vehicles and driverless cars is a field which makes extensive use of machine learning and artificial intelligence, relying on it to make decisions. These decisions require a vast amount of data in order to be properly inferred. This data is often in the form of images from a video feed and an increase in the amount of data is directly correlated to more refined decision making. What if you could remove certain parts of the data by compressing the image input, and how would that influence the performance of the different machine learning algorithms? In this report we have two datasets that we compress in different ways. We then analyze the results of running a pre-trained neural network model on them, and compare it's performance to that of running the same neural network model on the non-compressed datasets. The results show that the removal of data via compression is not in a linear relationship with neural network performance, and that depending on the compression type, results may be favorable or unfavorable.**

I. Introduction

A. Background

Whilst the idea of driverless cars is not a new one, the field of autonomous vehicles is relatively young. The task of driving a vehicle and being aware of one's surroundings might seem trivial to a human. However, there is a vast underlying complexity to decision making during driving. Therefore, the task of designing a system which is able to not only get from point A to point B, but to do so through a complex and at times chaotic environment in a safe and efficient manner is anything but trivial. The developments in the field of Artificial Intelligence and Machine Learning (ML) have contributed immensely to the development of autonomous vehicles. Autonomous vehicles operate based on data about their surroundings which is gathered through cameras and other sensors such as radar and lidar. In order to process all of this data, ML is used as the basis for all decision making, ranging from lane changing and following the rules of traffic to collision avoidance and saving the lives of people on the road.

B. Problem Domain & Motivation

Autonomous driving requires a lot of video data to be analyzed quickly. Streaming and analyzing a large amount of data requires high and fast bandwidth. For these reasons using video compression might prove beneficial as it removes parts of the data from the video deemed as repetitive by the video codec in use, which decreases the size of said video and may lead to improved response times. However, as video compression infers the loss of data, some of the data being lost might be critical as the ML algorithm may be relying on said missing data to classify an object. We define critical data as data which if removed or altered has a causal relationship with decreased prediction performance of a ML algorithm. Thus, it is crucial to use a video compression format that ensures the best trade-off between unnecessary data reduction and retention of critical data.

C. Research Goal & Research Questions

In this paper we will focus on image data, used as input to a ML algorithm. How humans see and process video and images differs completely to how a computer views it. The perception of colors, shapes and size differences of objects is affected by the object's distance, lighting, familiarity as well as the image's resolution. The effects of compression on human perception of digital media can be quantified via the use of metrics such as the Structural Similarity (SSIM) [1] index and Peak Signal to Noise Ratio (PSNR). The goal for this research project is to determine what influence do lossy video codecs optimised for human vision have on the performance of a ML algorithm developed and trained for the purpose of tackling a problem in the domain of autonomous driving.

RQ1: How do video codec parameters optimised for SSIM/PSNR influence the performance of a selected machine learning algorithm?

RQ2: What video codec parameters for lossy encoding result in the best performance of a selected machine learning algorithm?

To answer these questions we will be making a comparison between the performance of a ML algorithm running on different variants of encoded datasets, using the observed algorithm's performance on the original dataset as a baseline

for comparison. The algorithm that we will be using is a pre-trained model that has been trained using non-compressed images to detect traffic cones and color of said traffic cones.

D. Contributions

Marko was responsible for comparing the data from each preset and tune to the original baseline data, and creating plots and graphs from said data.

Marko worked on the Introduction section, Related Work, Methodology, Results, Analysis and Discussion, as well as the Conclusion and Future Work section.

Jonathan was responsible for the automation of the data collection process, which extracted data from running YOLO on all images, grouping it by preset and tune.

Jonathan worked on the Introduction section, Related Work, Methodology, Results, Analysis and Discussion, as well as the Conclusion and Future Work section.

Marko took the lead on the Related work and Methodology section, while Jonathan did the same for the Results and Analysis and Discussion, with both members contributing to the others sections in various ways, ranging from minor fixes to several paragraphs.

E. Scope

As already mentioned, this study covers the effects that video compression optimized for human perception has on the performance of a neural network trained for the purpose of addressing an autonomous driving related domain problem. The neural network model is treated as a black box and is not re-trained during any point of this study. This study does not cover the effects of video compression on neural networks performance, in cases where said network model is trained to address a problem in a domain outside that of autonomous driving, nor does it cover the effects of video compression optimized for any purposes other than human perception.

F. Structure of the Article

This article is divided into six sections:

- In the first section we describe the background to the problem which we wish to address, state the motivation for our research, outline our research goal and questions and define the scope of our research.
- In the second section we elaborate on three studies that loosely relate to our research.
- In the third section we describe the setup of our experiment, namely the ML algorithm and compression codec which we are using, how we handle data compression, data collection and data analysis.
- In the fourth section we present the results from our experiment in a graphical format via the use of two dimensional plots and describe our corner cases.
- In the fifth section we analyze how our results relate to our research questions, discuss potential validity threats to our experiment and elaborate on the steps we took to mitigate them.
- In the sixth section summarize our entire study and discuss potential future work that could be carried out to add upon the results of our experiment.

II. Related Work

As of the time of writing and extent of our knowledge there are multiple studies on the affects of image quality on neural networks. However, none of them directly address the domain of autonomous driving. Nevertheless, they still examine the topic of neural network performance on compressed data, thus we will still examine five such studies as they relate to our topic.

A. Dodge & Karam

In their study study titled "Understanding How Image Quality Affects Deep Neural Networks" [2], Dodge and Karam evaluate the four different neural network models for image classification under five different quality distortions - blur, noise, contrast, JPEG and JPEG2000. Their result shows that existing neural networks are are in fact susceptible to image quality distortions, with blur and noise having the most profound affect.

B. Koziarski & Cyganek

In their study titled "Impact of Low Resolution on Image Recognition with Deep Neural Networks: An Experimental Study" [3], Koziarski and Cyganek evaluate the classification accuracy of notable neural network architectures as of its publish date. They also examine the potential application of super-resolution prior to classification and its effect on classification accuracy. Their experiment shows that there is merit to the use of super-resolution in cases where image resolution is not severely decreased as otherwise classification accuracy on very low resolution images remains low, concluding that contemporary neural networks remain significantly affected by low resolution.

C. Roy, Ghosh, Bhattacharya & Pal

In their study, "Effects on Degradation on Deep Neural Networks" [4] the authors evaluate an alternative architecture for image classification based on grouping neurons into so called capsules and a new dynamic routing protocol. Though this new capsule based architecture is showing promising results when used on existing datasets, their goal is to evaluate its performance on data containing inherent noise. In order to do so the authors compare the performance of six widely used Convolutional Neural networks (CNNs) on two different datasets under various image quality distortions. The authors show that high accuracy for classification when a dataset contains a very large number of classes is not currently attainable, however the new capsule architecture does in fact prove robust to certain image degradation. Performance is most affected in the presence of motion and Gaussian blur and salt and pepper noise.

The key take away from the mentioned literature on the subject seems to be that different blur and noise have the most detrimental affect on neural network prediction confidence.

D. Roy, Dziugaite & Ghahramani

This study, "A Study of the Effect of JPG Compression On Adversarial Images"[5], compares how adversarial images, which are natural images that have been modified in a way to fool an image classifier, that have been further compressed compare against normally compressed images, as almost all image classification data is composed of JPG images. The results of the study is that JPG compression does reverse the drop in classification accuracy, meaning that JPG compression definitively affects the way that a neural network views an image.

E. Quijas & Fuentes

In their study, "Removing JPEG Blocking Artefacts Using Machine Learning"[6], they write about JPEG compression, what results tbe compression can yield and what issues may appear in the resulting images. While the study is not directly related to our topic, it shows the effects that JPEG compression has on images, as well as how the compression can be tuned for better compression results. This means that with further work regarding how the images in our study are compressed, better results may be possible, as our study only used pre-existing presets for our compression parameters.

III. Methodology

In order to answer our research questions we are going to carry out an experiment. The basic idea is that we are going to use a pre-trained Machine Learning algorithm and evaluate its classification accuracy when said algorithm is used on lossy datasets. The different datasets will be inferred via the use of compression with different video codec parameters optimised for SSIM and PSNR.

In the context of our study the independent variables are the different datasets corresponding to different compression parameters. Thus we have multiple treatments in our experiment. The dependent variables are the number of detected cones and the subsequent prediction confidence percentages for each detected cone.

The computer system which we are using for our experiment has a Nvidia GeForce GTX 960M GPU, an Intel core i7 4720HQ CPU and 8GB of RAM. Our system is running Ubuntu 18.04.1 LTS.

A. Machine Learning Algorithm

In our experiment we are using YOLOv3 [7] as our neural network for object detection. YOLO standing for "You Only Look Once" is a neural network developed using the open source neural network framework - darknet. It functions by applying a single neural network to a full image, which divides the image into regions and predicts bounding boxes and probabilities for each region. Bounding boxes are weighed by the predicted probabilities. YOLOv3 stands for the latest version of YOLO, exhibiting the best speed and accuracy.

We are using a pre-trained neural network model developed at REVERE [8] with the purpose of detecting traffic cones and their respective colors from the point of view of a vehicle. We decided to use this model as it is within the confines of the automotive domain and as it provides two datasets to work with.

B. Compression

In order to compress our datasets we are employing ffmpeg [9]. We selected ffmpeg as the tool for compressing our datasets as it is a leading multimedia open source framework for encoding, decoding, transcoding, streaming and more, that exhibits great performance.

The video codec we will be using to compress our dataset is x264 [10]. We selected x264 as it is a popular open-source codec that is published under the GNU General Public License and because it comes with tunes optimized for SSIM and PSNR. It has 10 default presets: ultrafast, superfast, veryfast, faster, fast, medium, slow, slower, veryslow and placebo. We are examining the effects of compressing our data using every x264 preset with the exception of placebo, tuned for SSIM and PSNR respectively. We opted for not including the placebo preset as part of our experiment as its use is discouraged by the developers of ffmpeg.

The naming of the compression presets comes from how long it takes to compress an image using said preset. That means that the slower the compression is, the more the images that are processed are compressed, and more data is removed.

We have two datasets to compress, the first one is a smaller in volume annotated dataset used for training the neural network model, whilst the second one is a bigger in volume non-annotated dataset. As both datasets comprise of sequential jpg frames and as ffmpeg can only encode video files, we first compile each dataset into a mp4 file via the use of ffmpeg. After we have inferred the video files we then proceed with encoding them one by one by specifying the video codec, preset and tune. This results in 18 differently encoded variants for each dataset which we then convert back into sequential frames. However as the inferred sequential frames are in lossless png format and as our neural network model requires jpg files, we convert the png files back into jpg via the use of imagemagick [11]. This entire process is illustrated in Figure 3.1. We also wanted a baseline to compare everything against, which is a set of non-compressed images. The images went through the exact same process as the compressed datasets with the exception of encoding - they were compiled into a mp4 file, divided into frames and then converted back to jpg format. This way all of the files have been handled in the exact same way, with only compression and tune differentiating them.

To convert images in a directory into a video we use the command:

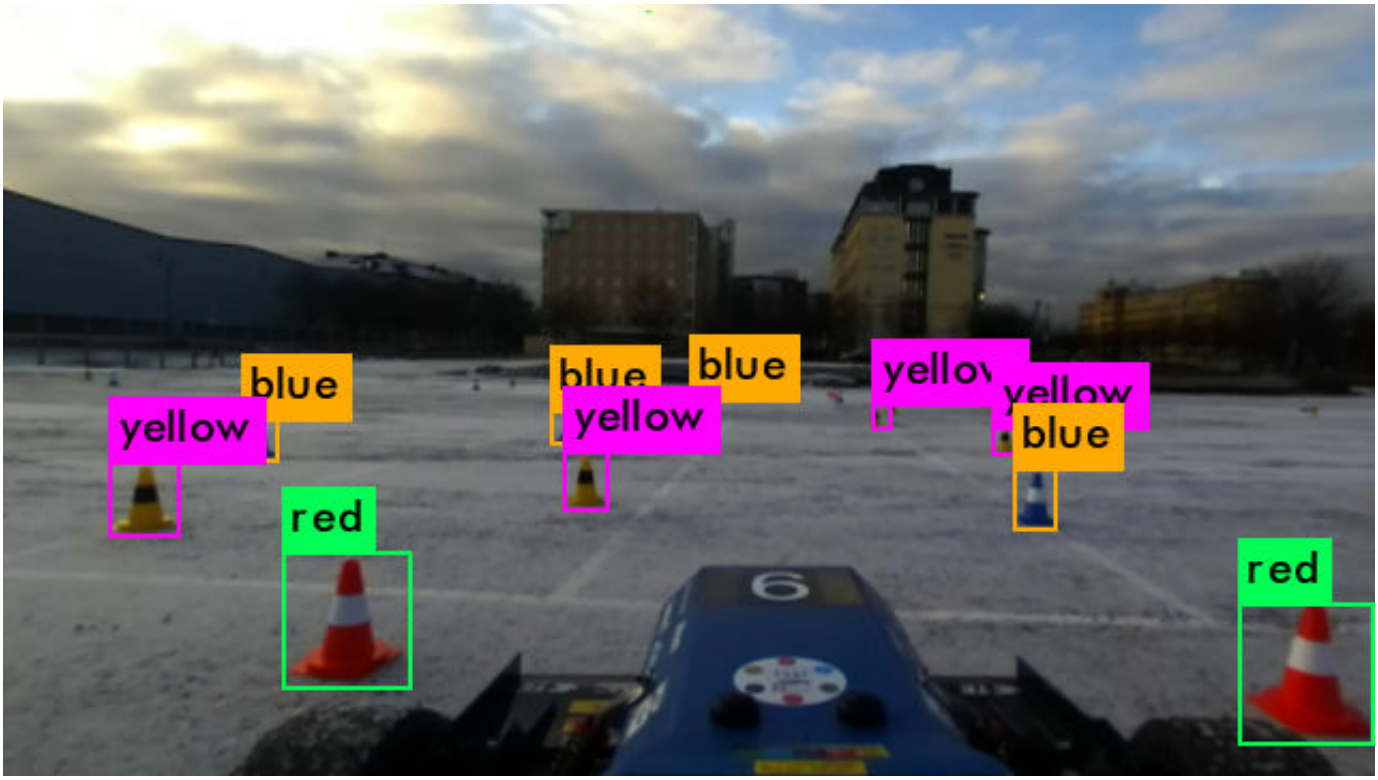

Figure 3.2
Example result after running YOLO on an image. Bounding boxes around each detected cone with a label stating the color of the cone. Taken from one of our completed executions of YOLO

```
$ ffmpeg -f image2 -i %d.jpg video.
    ↪ mp4
```

And to split a video into images we use the command:

```
$ ffmpeg -i video.mp4 %d.jpg
```

Between these commands, we encode/compress using the command:

```
$ ffmpeg -i video.mp4 -c:v libx264 -
    ↪ preset preset-name -tune tune-
    ↪ name encodedvideo.mp4
```

Then, to convert all of the png files in a directory to jpg files, we run:

```
$ mogrify -format jpg -quality 100 *.
    ↪ png
```

C. Data Collection

For the purposes of data collection we have created an automation script [12] which runs our neural network model on each one of the 18 different variants of our datasets, placing the results into different directories named after the respective preset and tune combination. The neural network uses jpg files as input and produces other jpg files with labeled bounding boxes drawn around every detected cone as output as can be seen in Figure 3.2, along with a text file

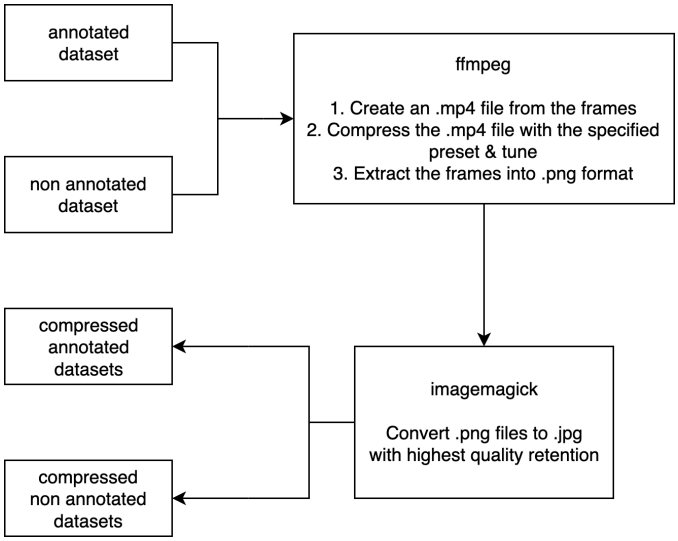

Figure 3.1
Data preparation sequence.

which includes all detected colors of cones and subsequent percentages corresponding the the prediction confidence for said colors. For the purpose of analysis we are basing our evaluation on the confidence percentage values, which can be found in the produced text files.

We measured the time it takes for our script to evaluate

all variants of a dataset. Our datasets vary in volume, the smaller dataset comprises of 318 frames, whilst the bigger one comprises of 746 frames. Due to this the it takes 18 minutes and 46 seconds to evaluate the original small dataset and a total of 5 hours and 55 minutes to evaluate all encoded variants of the small dataset. To evaluate the original big dataset it takes 42 minutes and 52 seconds whilst to evaluate all encoded variants of the big dataset it takes a total of 13 hours and 42 minutes. These measurements are inherently related to our specific system setup and can be affected by additionally running processes. For this reason evaluation and time measurement were carried out after a fresh reboot without starting any additional processes.

D. Data Validity

For the purposes of ensuring data validity, i.e. ensuring that the algorithm does not detect cones where there are none, we have carried out a manual sanity check on the results from each preset and tune combination.

E. Data Analysis

For the purposes of data analysis we have created a number of python comparison scripts [13].

Each line of the generated text file contains a color, and a percentage value which represents how confident the algorithm is that a cone with said color can be found in the image.

As our goal is to compare the difference in performance of our neural network, and that performance can quantified via the sum of prediction confidence percentages for each detected cone, these percentages are all added up to a total for each dataset. This is the done for all existing frames for each combination of compression preset and tune. Thus we are able to compare the total prediction confidence of every encoded dataset to that of the original dataset as a baseline.

Even though one of our datasets has annotations for cone positions, we opted for not using said annotations as when further inspected, not all cones seem to be marked in many of the frames.

As one of the initial goals with compressing our data is to reduce file size, we are also taking into account what effect does each combination of preset and tune have on the total size of a compressed dataset. We use the total size of the original datasets as the baseline for comparison.

When comparing the results from the annotated dataset to those of the non-annotated dataset, thus inferring an average result it is important to mention that the non-annotated dataset is larger than the annotated one. This ends up skewing the average results to closer resemble the results of the non-annotated data than those of the annotated data. This in itself is not inherently negative, as the algorithm treats all data the same, though it is still important to mention as to ensure that the results are as non-biased as possible.

IV. Results

In the case of the annotated datasets as illustrated on Figure 4.1, superfast PSNR exhibits the best performance of 99.92% however it also exhibits the second highest file size of 96.22%. Ultrafast PSNR exhibits the lowest filesize of 87.37%, however it also exhibits the second worst performance of 97.02%. Ultrafast SSIM exhibits the worst performance of 96.56% and an average file size of 92.46%. Superfast SSIM exhibits the highest file size of 97.82%, however it also exhibits the second best performance of 98.95%.

In the case of the non-annotated datasets as illustrated on Figure 4.2, ultrafastSSIM exhibits the best performance of 102.48% whilst also exhibiting the fourth highest file size of 101.85%. Slower PSNR exhibits the lowest file size of 92.24% and an average performance of 84.11%. Veryfast PSNR exhibits the worst performance of 78.16% as well as exhibiting the fifth highest file size of 100.6%. Superfast SSIM exhibits the highest file size of 112.22%, however it also exhibits the third best performance of 88.73%.

As illustrated on Figure 4.3, on average ultrafast SSIM exhibits the best performance of 99.52%, whilst also exhibiting the fifth highest file size of 97.16%. Ultrafast PSNR exhibits the lowest file size of 91.69% as well as exhibiting the second best performance of 98.32%. Veryfast PSNR exhibits the worst performance of 88.07% and exhibits the sixth highest file size of 97.06%. Superfast SSIM exhibits the highest file size of 105.02%, however it also exhibits the third best performance of 93.84%.

V. Analysis and Discussion

In almost all cases when looking at the average results from both datasets, almost every encoding preset shows similar results for both SSIM and PSNR. One noticeable pattern is that the same encoding preset with SSIM has a slightly higher average performance percentage, as well as increased file size when compared to its PSNR counterpart. However, when looking only at the results from the annotated dataset, we can observe a pattern of better performance for the same encoding presets combined with PSNR rather that SSIM, and a general decrease in file size. Regarding the first research question about how the performance of the machine learning algorithm is influenced by the different encoding presets tuned for SSIM and PSNR, we can see that algorithm performance is reduced in almost all cases but ultrafast SSIM and a general decrease in data file size is observed in all but superfast and veryfast, for both SSIM and PSNR tuning.

When evaluating what combinations of presets and tunes are applicable for use, all of the preset/tune combination that exhibit an increase rather than a decrease in file size in comparison to the baseline, should be discarded. In all cases but ultrafast SSIM for the non-annotated dataset such combinations exhibit decreased performance, on top of the increase in file size, meaning there is no objective value in their application. Examples of such combinations observed for the non-annotated datasets are superfast SSIM, superfast PSNR, veryfast SSIM and veryfast PSNR, whilst in the case of the combined average results from both datasets examples for such combinations are superfast SSIM and superfast PSNR.

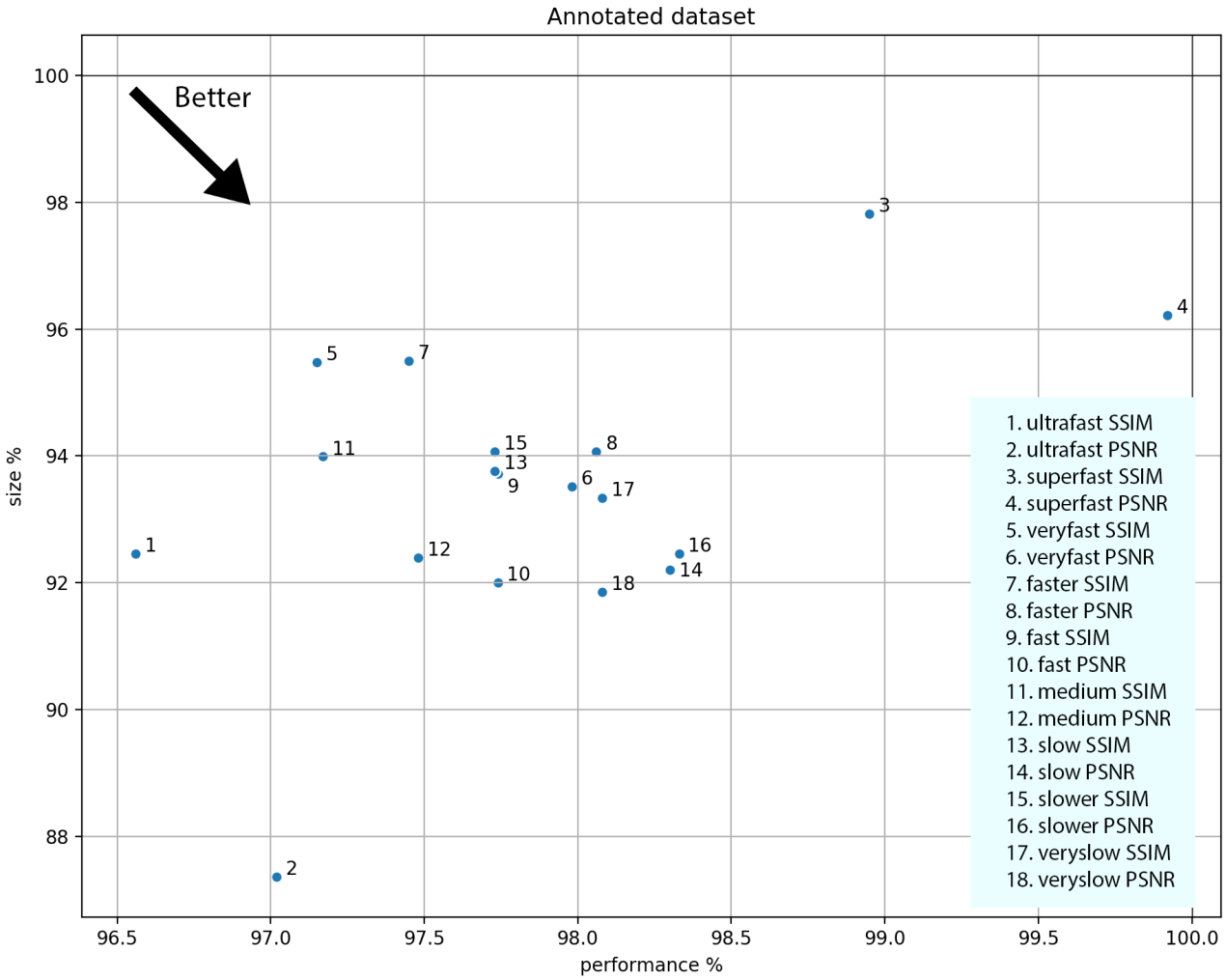

Figure 4.1
Neural network performance and file size in comparison to the baseline in the context of the annotated variants.

We observed a decrease in neural network performance in almost all cases for each compression preset and tune combination. The only time that a higher than the baseline performance was observed was in the case of the ultrafast preset with SSIM tuning from the non-annotated dataset. This dataset did however have a larger total file size than the baseline.

The dataset with almost the the same performance of 99.63%, and lower file size of 96% is the ultrafast preset with PSNR taken from the non-annotated dataset, and superfast with PSNR with superfast SSIM close behind from the annotated dataset. When we calulate the average of both datasets, nothing was observed for exhibiting higher than the baseline performance. The closest example is ultrafast SSIM followed closely by ultrafast PSNR, exhibiting 1% decrease in perfomance, and with ultrafast PSNR being about 5% smaller compared to the baseline.

These results show us that the best performing preset overall is ultrafast, with the PSNR version being the better when it comes to size vs performance. While the SSIM variant has a higher average performance, it is almost the same size as the baseline, and therefore takes about the same bandwidth for a similar performance. The PSNR variant has a lot smaller size, with very similar performance, making that combination the best overall result when taking size into account.

Most of the observed results are not very surprising. When compressing and removing data from an image, neural network performance will decrease as this introduces the possibility that the lost data may critical. What is surprising is how the non-annotated dataset performed with ultrafast SSIM, returning a higher performance percentage than the baseline.

Looking at the results from both the annotated and non-annotated dataset, we can see that a number of preset and tune combinations stand out more than the rest. These are the two fastest presets with both tunes - ultrafastSSIM, ultrafastPSNR, superfastSSIM and superfastPSNR, which stand

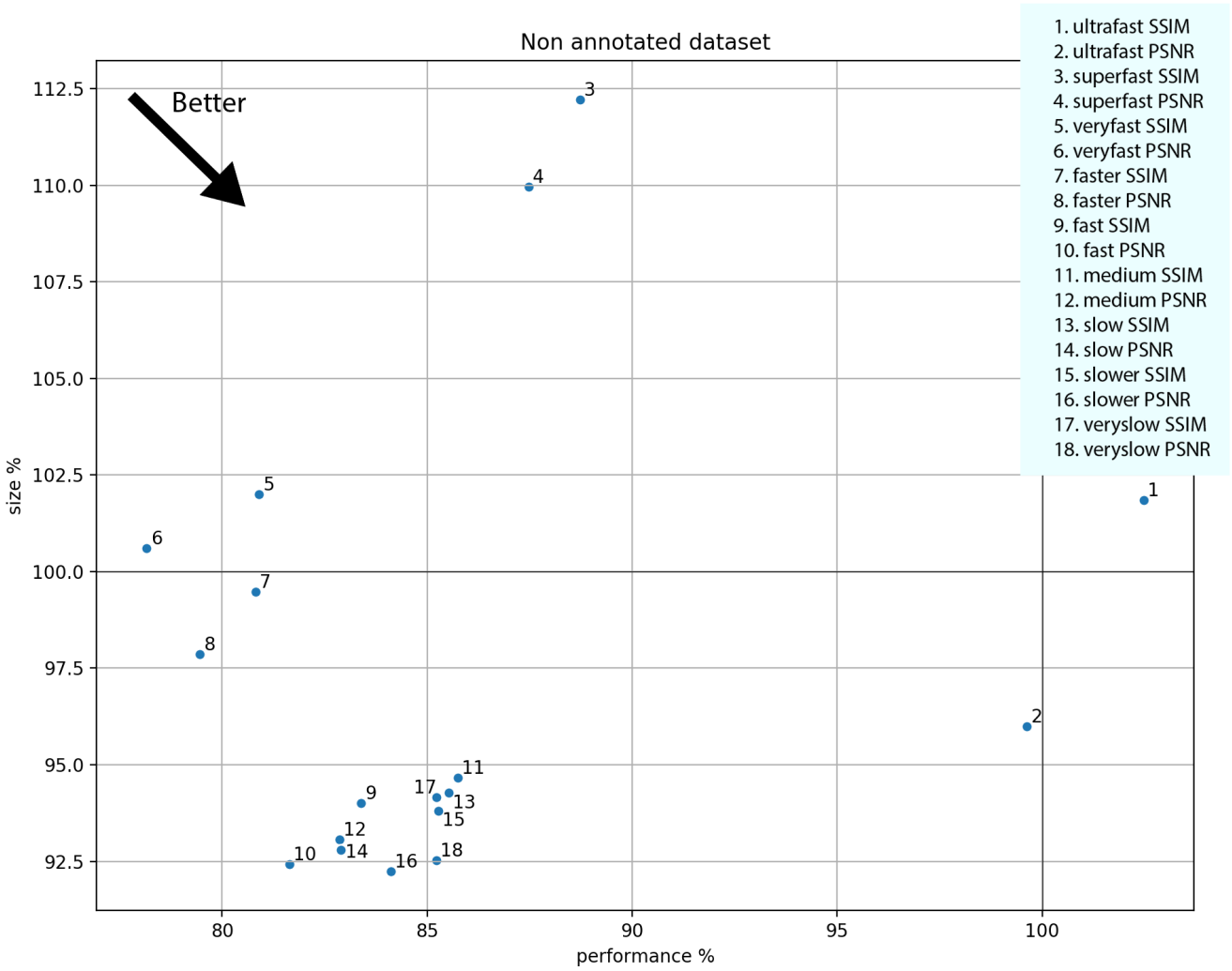

Figure 4.2
Neural network performance and file size in comparison to the baseline in the context of the non-annotated variants.

out from the rest in both performance and size. In order to determine whether these examples are outliers we employed the Interquartile range (IQR) method as our statistical test for the combined average results, which set's up a "fence" outside the first and third quirtiles, where any results that fall beyond the bounds of this "fence" are considered outliers. We illustrate this via the use of two box plots as can be seen in figures *5.1* and *5.2*.

We observe that based on the IQR method, ultrafastSSIM and ultrafastPSNR can be cosnidered as outliers in terms of neural network performance, whilst superfastSSIM can be considered an outlier in terms of file size after compression.

One of the reasons that these specific presets are the ones that stand out, is because they are the closest to the original data compared to every other preset, as they are compressed using the fastest presets, meaning they are the least compressed as well. That being said, two of the presets with the least performance in the annotated dataset was the ultrafast preset, as well as being the preset with the smallest size. This result shows that the compression is not always perfectly reliable, and results can vary despite using the same presets.

It is also important to understand the difference between the two datasets. The images in the annotated dataset were taken with snow on the ground and different lighting conditions compared to the non-annotated images which were taken with no snow on the ground, and different lighting. This makes it so that a direct comparison cannot be made between the two datasets, which is why we compare the presets against an execution of non-compressed images, and why the data between the two presets can vary in many different ways.

Despite us having the annotations for the cone positions for one dataset, we cannot use them as the base truth for the annotated dataset. The reason for this is because that would require a modification of the algorithms output so that it calculates and notes down the positions of each detected cone, which is beyond the scope of our capabilities. It is

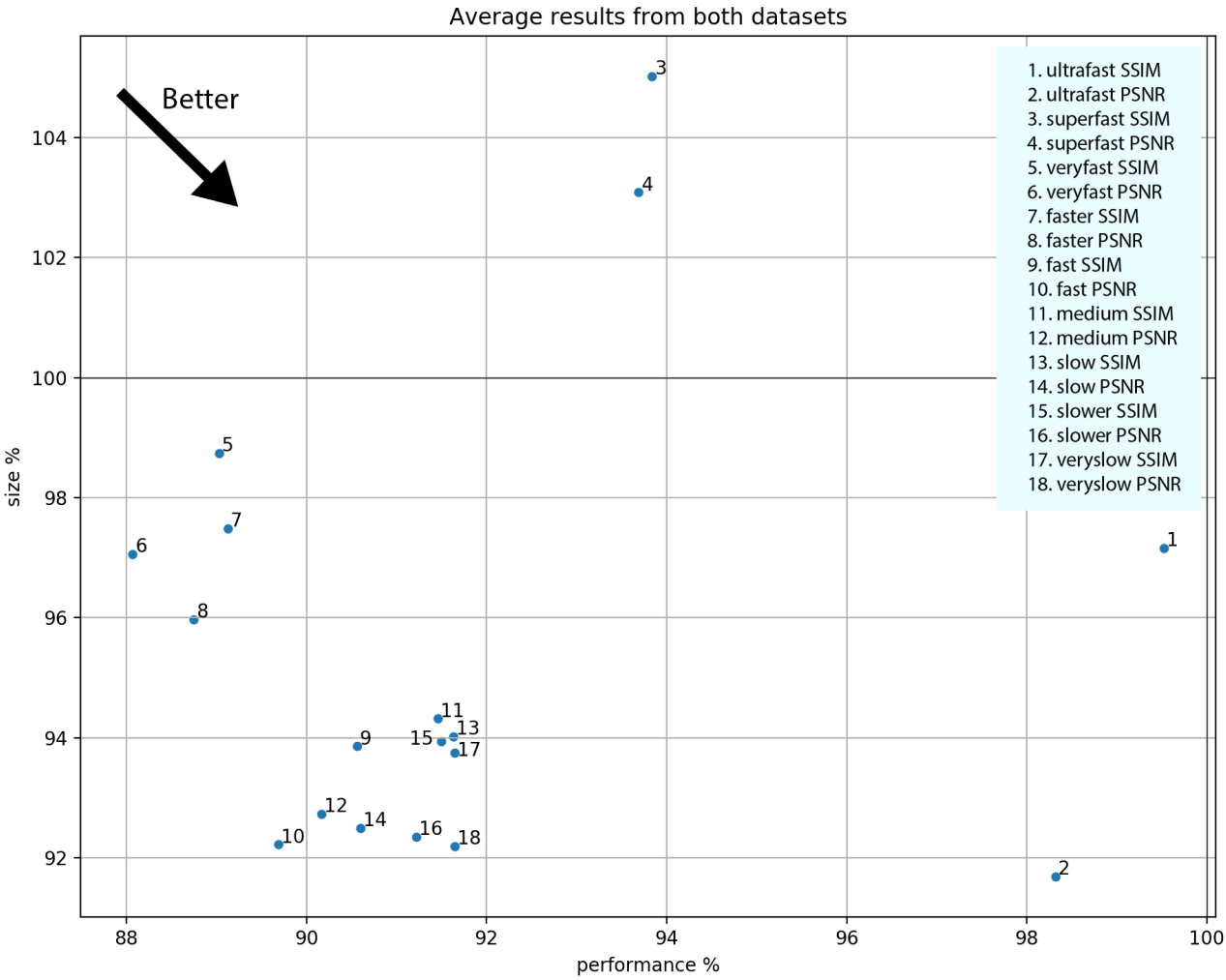

Figure 4.3
Average neural network performance and file size in comparison to the baseline.

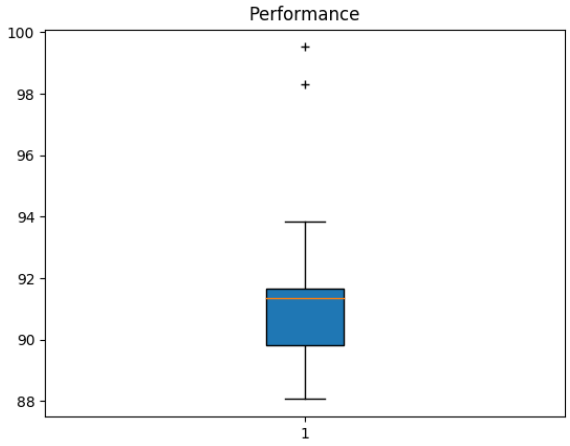

Figure 5.1
Boxplot for neural network performance.

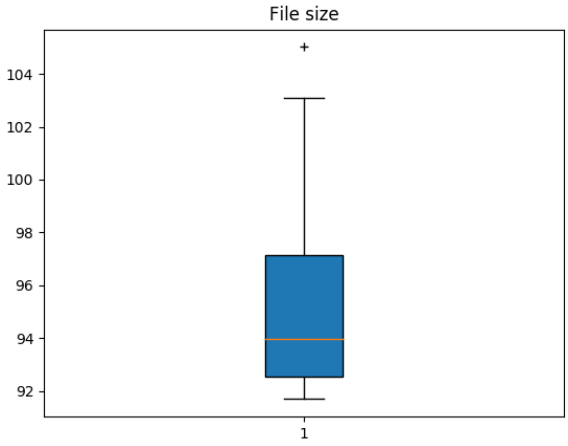

Figure 5.2
Boxplot for file size after compression.

also important to note that not all cones in each image are annotated, so making a direct comparison to them would yield incorrect results.

Dodge and Karam[2] delved into the topic of image quality based on distortion, showing that blur and noise had the largest effect on image classification neural networks. This can be identified in our results as well, where the images tuned for PSNR, which has less noise than SSIM, had overall better detection relative to their filesize. The amount of data that we have is however not enough to draw any conclusions from, but can instead serve as data to validate some of our results.

Koziarski and Cyganek[3] wrote about low resolution images and the effect they have on a neural network. While we write about compression, similarities can still be seen between low resolution images and highly compressed images, at least from a human point of view [14]. This helps show that higher compression looks similar to low resolution images in certain situations, but they are not directly comparable. Compared to our data, you can see that the higher compression we use, the worse the results tend to be, which is in line with the conclusion that Koziarski and Cyganek drew, which proves that contemporary neural networks are significantly affected by low resolution, while our data shows that higher compression yields worse results.

Roy, Ghosh, Bhattacharya and Pal[4] wrote about degraded images on six widely used Convolutional Neural Networks. The key takeaway here is similar to what Dodge and Karam had, which is that blur and noise had the largest effect on the different neural networks.

In the study made by Roy, Dziugaite and Ghahramani[5], they tested how compressing adversarial images affected an image classifier, which is a type of machine learning algorithm that learns to detect certain types of objects or animals. Their results were that by compressing the images, the image classifier had an easier time identifying whatever object was on the image correctly, even though a human could not tell the difference of the adversarial effect. This however, was not always the case, and there were exceptions.

This shows that there is a lot more data that is being read by the machine learning algorithm that affects, and that compressing images definitively affects how easy or hard it can be for a machine learning algorithm to identify items. We observed a similar effect when identifying cones in images with the different presets, and the results we were given were not always as expected. Some of the presets gave a lower detection percentage compared to their collective size, which could mean a number of things. What could be done to improve our knowledge of what affects the compression the most would be to manually control all of the variables when compressing images, changing one at a time and seeing the effects on machine learning algorithm.

Quijas and Fuentes study, "Removing JPEG Blocking Artefacts Using Machine Learning"[6], shows in more detail what effects JPG compression has on images, and how they used machine learning to tune the compression for the best results possible.

A. Threats to Validity

1) Construct Validity:

A threat to construct validity would be if our neural network is not legitimate. By this we mean that it is not one that is recognized as having sufficiently good performance as to be used as the solution to a real world problem. In order to mitigate this threat we selected YOLO, which is a proven real-time object detection system. YOLOv3, the third release of the model is extremely fast and accurate.

2) Internal Validity:

We have a number of events that may threaten the internal validity of some results, so we have taken steps to prevent and reduce said risks ensuring that the data is as comparable as possible.

In order for the data extracted from running the machine learning algorithm to be as consistent as possible, all evaluation of the datasets with the neural network is carried out on a single hardware setup, with no additionally running processes. This ensures as consistent results as possible.

In order to mitigate potential human error when running the neural network on each variant of the datasets, we have created the aforementioned automation script which, runs the neural network on all variants of a dataset one after the other, and stores the results in different directories.

To avoid errors when comparing data, we created the aforementioned comparison scripts to automate the comparison process and return the results. This ensured that there would be no human error as in incorrect results when calculating differences, or in inconsistent rounding errors.

An issue that we had was running the machine learning algorithm using lossless png files, as png images could not be used for our neural network. Instead, we had to convert them to jpg files. we did so via the use of the command 'mogrify' which is part of the imagemagick tool available for linux, with the '–quality 100' flag. This flag is used to keep the jpg compression as unobtrusive as possible, keeping it in line with the quality of a png file as much as possible. According to the documentation of imagemagick, this means that the chroma channels are not downsampled, and is a request for a non-lossy compression.

3) External Validity:

An external validity threat to our study would be if the neural network model is not trained for the purpose of performing a task specific to the domain of autonomous driving, in which case our result would not be generalizable enough as to apply to said domain. In order to mitigate this validity threat we selected a model that is trained for the task of detecting traffic cones and the colors of said traffic cones. This task is within the domain of autonomous driving, as an autonomous vehicle must have the ability to detect traffic cones.

Another external validity threat to our study would be if the data which we use is not varied enough as to be generalizable to real world conditions. In order to mitigate this validity threat we are evaluating the neural network performance based on two datasets. The two datasets are collected in

different weather conditions, one comprises of scenes taken during daytime semi-snowy winter conditions, whilst the other comprises of scenes taken during a clear evening.

VI. Conclusion and Future Work

In this study we experimentally evaluated what effects does x264 encoding optimized for human perception have on neural network performance in the domain of autonomous vehicles. We did so with the purpose of potentially proving the applicability of x264 video encoding as a means of minimizing the amount of data needed for the adequate performance of machine learning algorithms used for autonomous driving whilst retaining human perception levels of said data as much as possible.

We encoded two datasets with one of nine common x264 presets - ultrafast, superfast, veryfast, faster, fast, medium, slow, slower or veryslow, and one of two tunes optimized for human perception - SSIM or PSNR. Thus we inferred eighteen different variants for each dataset corresponding to every combination of a preset and tune. We selected a pre-trained neural network model that was trained for the purpose of detecting traffic cones and colors of said traffic cones. We then ran our pre-trained neural network on both datasets and on each of their corresponding eighteen variants. After this we made two comparisons. First compared the results of the neural network performance on the encoded variants of the datasets to that of the original datasets, after which we compared the total sizes of the encoded variants of the datasets to that of the original datasets. The main findings of our experiments are the following:

- Some preset and tune combinations show objectively unfavorable results, with a neural network performance reduction of more than 10% whilst with a data size increase of more than 10% such as in the case of the superfast preset tuned for both SSIM and PSNR.
- It is possible to retain up to 97% of neural network performance whilst at the same time reducing data file size by more than 22% with use of the ultrafast preset tuned for PSNR. This makes a good case for the potential applicability of this specific preset and tune combination.
- For more than half of the preset and tune combinations neural network performance is retained between 90% up to 92% whilst data file size is reduced between 95% down to 92%.

Additional future work would be required in order to solidify the results of this experiment. Such work would benefit from expanding on the x264 options, namely instead of evaluating the presets, to further tweak parameters in more detail and evaluate the resulting encoded data.

Future work on the topic would stand to benefit from including even more varied datasets, ones that comprise of additional weather conditions e.g. rain or a lower light environment, as well as ones in which cones may be snowed on or have debris or mud splashed on them, all of which would further represent different real world conditions.

Future work on the topic would also stand to benefit from evaluating the performance of a different neural network model developed and trained for the purposes of addressing a different task within the domain of autonomous driving.

REFERENCES

[1] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, p. 600612, 2004.

[2] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, 2016.

[3] M. Koziarski and B. Cyganek, "Impact of low resolution on image recognition with deep neural networks: An experimental study," *International Journal of Applied Mathematics and Computer Science*, vol. 28, no. 4, p. 735744, 2018.

[4] P. Roy, S. Ghosh, S. Bhattacharya, and U. Pal, "Effects of degradations on deep neural network architectures," *CoRR*, vol. abs/1807.10108, 2018. [Online]. Available: http://arxiv.org/abs/1807.10108

[5] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of JPG compression on adversarial images," *CoRR*, vol. abs/1608.00853, 2016. [Online]. Available: http://arxiv.org/abs/1608.00853

[6] J. Quijas and O. Fuentes, "Removing jpeg blocking artifacts using machine learning," in *2014 Southwest Symposium on Image Analysis and Interpretation*, April 2014, pp. 77–80.

[7] J. Redmon, https://pjreddie.com/darknet/yolo/. [Online]. Available: https://pjreddie.com/darknet/yolo/

[8] Chalmersfsd, "chalmersfsd/yolo-perception," Apr 2019. [Online]. Available: https://github.com/chalmersfsd/yolo-perception

[9] "Ffmpeg," https://ffmpeg.org/. [Online]. Available: https://ffmpeg.org/

[10] VideoLAN, "x264," https://www.videolan.org/developers/x264.html. [Online]. Available: https://www.videolan.org/developers/x264.html

[11] I. S. LLC, "Convert, edit, or compose bitmap images," https://imagemagick.org/index.php.

[12] J. Partain, "Automation script," https://github.com/MStanoevich/ThesisProject/tree/master/bash-scripts, May 2019.

[13] M. Stanoevich, "Comparison scripts," https://github.com/MStanoevich/ThesisProject/tree/master/comparison, Jun 2019.

[14] "Higher resolution or lower compression jpg's?" http://users.wfu.edu/matthews/misc/graphics/ResVsComp/JpgResVsComp.html, accessed: August 2019.