Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Magnus Stava

A Comparative study of model performance on multiple systems for real-time object detection.

Compare the real-time performance of state-of-the-art object detectors on multiple systems with an anonymization modification to enable usage in smart city domains.

Master's thesis in Simulation and Visualization
Supervisor: Prof. Ibrahim A. Hammed
Co-supervisor: Muhammad Umair Hassan PhD(c)
June 2023

NTNU
Norwegian University of
Science and Technology

Magnus Stava

A Comparative study of model performance on multiple systems for real-time object detection.

Compare the real-time performance of state-of-the-art object detectors on multiple systems with an anonymization modification to enable usage in smart city domains.

Master's thesis in Simulation and Visualization
Supervisor: Prof. Ibrahim A. Hammed
Co-supervisor: Muhammad Umair Hassan PhD(c)
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

ABSTRACT

With the development of smart cities, researchers have shown how data usage can make city management more efficient and improve the quality of life for the people in the city. In the domain of smart cities, the data collection must occur in real-time for effective decision-making. Additionally, with the use of data, particularly visual data, a concern is maintaining our right to privacy and ensuring that the collected data can not be misused to affect individuals' lives. This means that sensitive data must be anonymized. In this thesis, we look into two problems related to object detection. For the first part, we compare the performance of current state-of-the-art object detectors on both embedded and non-embedded devices. We have analyzed how they performed concerning the real-time aspect and how decreasing the input size can increase the real-time performance, but by then sacrificing some of the model's performance. For the second part, we have developed multiple extensions of the standard widerface dataset. We have analyzed the effects of these expansions on the data and the model's performance. With the presented result, we want to better understand how near the current development in hardware and object detection has brought us to running state-of-the-art object detectors in real-time, with extra attention to the performance of embedded devices. The presented results also give researchers access to new face datasets for future research. A final contribution is a novel modification of the YOLOv8 state-of-the-art object detector for anonymization purposes to enable implementation in smart city domains.

SAMMENDRAG

Med utviklingen av smarte byer, har forskere undersøkt hvordan bruken av data kan gjøre by forvaltningen mer effektiv og forbedre livskvaliteten for innbyggerne i byen. I smarte byer må datainnsamlingen foregå i sanntid for å sørge for en effektiv beslutningstaking. En bekymring knyttet til bruk av data, spesielt visuelle data, er å sikre at vårt personvern blir ivaretatt og sikre at innsamlet data ikke kan misbrukes til å påvirke enkeltpersoners liv. Dette betyr at sensitiv data må anonymiseres. I denne avhandlingen ser vi nærmere på to problemstillinger knyttet til objekt deteksjon. I den første delen sammenligner vi ytelsen til nåværende «state-of-the-art» objekt detektorer både på innebygde og ikke-innebygde enheter. Vi har analysert hvordan de presterer med hensyn til sanntid og hvordan en redusering i inndatastørrelsen ofrer noe av modellens ytelse, men øker modellens sanntidsytelse. I den andre delen presenterer vi flere utvidelser av den standardiserte "widerface" datasettet. Vi har analysert effekten av disse utvidelsene på modellens ytelse. Med de presenterte resultatene ønsker vi å øke forståelsen for hvor nær den nåværende utviklingen innen maskinvare og objekt deteksjon har brakt oss til å kjøre toppmoderne objekt detektorer i sanntid, med ekstra oppmerksomhet på ytelsen til innebygde enheter. De presenterte resultatene gir også forskere tilgang til nye ansiktsdatasett for fremtidig forskning. Et siste bidrag er en triviell anonymisering modifikasjon av «state-of-the-art» objekt detektoren YOLOv8 for å muliggjøre implementering i smart byer.

PREFACE

In a world, with increasing public surveillance and with better and better-performing algorithms for detection purposes, researchers need to keep the focus on our right to privacy. While writing this thesis, EU is getting closer to voting for their proposed AI legislation bill. This is an important first step, and our privacy has to be prioritized above profit. With sensitive information removed from the data, the remaining information contains value that can still improve many aspects of our life. The motivation behind the chosen topic is to contribute with shining a light upon this topic.

I want to offer my gratitude to my supervisor Ibrahim A. Hameed and Co-supervisor Muhammad Umair Hassan, for their support and guidance at important points during my thesis work. Additionally, I want to thank my family for their support and enthusiastic encouragement during the thesis's work and writing.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

List of all abbreviations in alphabetic order:

- **AI** Artificial intelligence
- **CPU** Central processing unit
- **CSP** Cross-stage partial
- **EU** European Union
- **FN** False negative
- **FPS** Frames per second
- **FP** False positive
- **GAN** Generative adversarial network
- **GB** Gigabyte
- **GPU** Graphical proscessing unit
- **HPC** High performing computing
- **NTNU** Norwegian University of Science and Technology
- **R-CNN** Region based neural networks
- **SGD** Stochastic gradient descent
- **SOTA** State of the art
- **SSD** Single stage detector
- **TN** True negative
- **TP** True positive
- **YOLO** You Only Look Once

CHAPTER

ONE

INTRODUCTION

1.1 Motivation

Smart city development is a keyword when planning the city of tomorrow. The main objective of a smart city is to utilize sensors and data collection for more innovative management of all aspects of the city. For privacy reasons, sensitive visual data must be anonymized before being used. For humans, this includes hiding our facial features. The security aspect of this data demands that it be anonymized in real-time to avoid any temporary storage of sensitive data susceptible to leakage or foreign attacks. For smart cities the data collection should be performed in real-time and on embedded devices for simpler implementation in the real-world.

This thesis aims to compare the performance of state-of-the-art (SOTA) detection models on multiple systems and research how the security-performance relationship affects real-time performance. In addition, we will contribute to the face detection community by extending the standard widerface dataset with additional data and modern augmentation techniques. These extensions will also be provided as standalone datasets. The performance effect these dataset versions have on the models will be evaluated and presented. The thesis also hopes to shine a light on real-time object detection and the privacy concern with large-scale implementation of object detection models in the real world. To achieve the objectives mentioned above, multiple trials were conducted on several devices to compare their performance when running the anonymization models with various settings, such as image resolution and training parameters. The findings will present the relationship between security and performance when optimizing for real-time. The findings will be presented for each of the systems. In addition, 5000 new images of faces with various complexity were collected to increase the data in the widerface dataset. These additional images were used to create multiple dataset versions using manual pre-processing and mosaic augmentation. The results present the effect of each of these variations.

1.2 Objective

To summarize the above points, the following research questions will be explored in greater detail throughout the thesis.

1. How does the real-time performance on edge devices, GPU, and CPU compare when running SOTA face anonymization models?
2. What is the impact of incorporating images from additional open-source datasets (Open Images V7) and the usage of modern data augmentation on the performance of face detection?

1.3 Literature review

Following is an overview of the advancements within the fields of object detection and anonymization.

1.3.1 Object detection

Object detection developed from two-stage models such as R-CNN [1] and Fast-R-CNN[2] to faster single-stage models such as YOLO[3] and SSD [4] where the object detection process is done in one neural network. Later advancement came with the usage of pre-defined anchor boxes for detection, in models such as YOLOv5 [5] and squeezedet [6]. Newer advancement have returned to anchor free models such as YOLOx [7] and YOLOv8. Anchor free models decreases computation, and increases their real-time as well as their generalization capabilities [7][8][9].

1.3.2 Face anonymization

Face anonymization has traditionally utilized methods such as blurring, black box, or pixelation [10]. Newer advancements have utilized GAN for swapping identity by creating an artificial one, preserving the contents within the image [11]. Other researchers have enabled encrypted privacy, allowing authorities to unlock individuals identity in particular scenarios [12]. Prior work presented a combination of traditional anonymization techniques utilizing blur and noise [13].

1.3.3 Evolution of face datasets

Face detection has been a common field for object detection, and likewise, datasets for face detection have evolved from early versions such as Faces In The Wild dataset [14]. Faces In The Wild was then used to develop more robust datasets such as FDDB dataset [15] and MALF [16] that contained more variation of images in terms of race and age groups. From there, a larger and more complexed dataset was presented with widerface, which has become the standard for unconstrained face detection, widerface expands both in size and complexity compared to earlier benchmark datasets [17].

1.4 Contributions

Due to the fast evolution within the technology industry, high-quality comparison studies of SOTA model are hard to come by. The presented thesis hopes to contribute to the research community by comparing the real-time performance of the YOLOv8 object detection model on various systems and generating additional face datasets ready for deployment for other researchers. In summary, the thesis aims to contribute with the following points.

- Compare the performance of face anonymization on multiple systems.
- Improve face detection by increasing the size of the standardized face dataset widerface.

1.5 Outline of the report

The report has the following structure, in chapter 2 the related theory is presented, chapter 3 presents the methodology behind the experiments. In chapter 4, the produced results are presented before a discussion regarding the results will be presented in section 5. The final chapter 6 will present a conclusion of the thesis work and future work.

1.6 Benefits of smart cities

As mentioned, a key factor in a smart city is gathering data from various sources, such as cameras. The purpose of this is to improve the general management in the city in terms of efficiency. Visual data collected in real-time from the city can help the city improve its management of traffic, crowd, and resources such as police, fire, and ambulance department.

1.7 Ethical considerations

As of writing this, the regulatory bodies of the EU are nearing their introduction of a legal framework for regulating the usage of AI and the requirement for implementing AI technology. The current bill named the *AI Act* will be voted for in June of 2023 [18]. The purpose of writing this is to showcase the ethical consideration that must be done and how anonymization techniques can help privacy concerning AI with implementation. In the proposed bill AI is placed into four risk groups, as seen below:

1. Unacceptable risk
    (a) This level of AI is prohibited. It can be systems such as real-time bio-metric identification or rankings of citizens using some social credit system.
2. High risk
    (a) It consists of the group of AI, which in some way affect people's lives. It can be within law enforcement, health service or education.
3. Limited risk
    (a) AI that does not directly affect people's lives and is permitted by only notifying users of what they are interacting with. The majority of AI systems will fall within this group.
4. Minimal risk
    (a) AI that has zero to minimal legislation requirements, only requiring minimal transparency. Systems such as spam filters and AI in video games would fall in this category.

Following the four risk categories, the usage of cameras for real-time data capture and decision-making in a smart-city context would depend on adequate anonymization of the people captured on video to avoid being classified as unacceptable. The theoretical basis and standard legal development of systems within each category still need to be established due to the fast development of the AI scene and due to the fact that the AI bill is the first of its kind in the world.

CHAPTER

TWO

THEORY

2.1 Deep learning

Deep learning became widely favorable around 2012 after showing significant improvements in speech recognition and later image classification. Following the two breakthroughs, deep learning has been used to improve performance in most areas imaginable. Some examples include vision tasks, medical diagnosis, speech recognition, speech generation, and more [19]. The power deep learning brought with it is partly due to convolution layers' effectiveness in mapping the relation between input and desired output. Continuing in the field of object detection, deep learning has proved useful due to neural networks ability to extract hierarchical relations and valuable information from high-dimensional data such as images [20]. In the field of object detection, deep learning models have evolved from R-CNN, where proposed regions are created and then fed into a support vector machine for classification [1], to more efficient models such as You Only Look Once (YOLO), where the whole image is treated at once and processed as a regression problem where the output is bounding boxes with its related probability for a class [21]. YOLO belongs to the group of single-stage object detectors, which need less computation compared to two-stage detectors like the R-CNN. Following the release of the initial YOLO, new versions have been developed, with each iteration improving performance and accuracy compared to its predecessor. The current latest release is YOLOv8, and its architecture will be elaborated. As of this writing, the research team at Ultralytics has released no official paper. However, by utilizing their code documentation and answers posted by the team on their forums, information about YOLOv8 architecture has been collected.

2.1.1 YOLOv8

Single-stage detectors such as YOLOv8 follow the traditional architecture setup as earlier version with a backbone layer, neck layer, and a final detection layer. The backbone layer is the largest part of the model and the most computationally de-

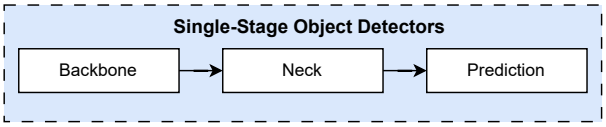


Figure 2.1.1: Architecture flow of single-stage detectors. Consisting of three sections, backbone, neck, and prediction layer.

manding. The backbone is responsible for extracting the features from the input image. The neck layer aggregates the outputs from the backbone and passes it forward to the detection layer, where the model first predicts if an object is present and then predicts the object's class. For a complete overview of the YOLOv8 architecture, please refer to appendix 6.1.

2.1.2 Backbone layer

YOLOv8 utilizes the CSPDarknet-53 network as the backbone to extract features from the input. Feature extraction can be understood as some calculations performed on the input image to obtain specific features that contain essential information of the object that we want to detect [19, p. 988]. CSPDarknet-53 performs feature extraction by utilizing dense blocks and cross-stage partial blocks, which are specific but simple calculations. Each block will be presented in simplicity to understand the concept of the different blocks.

Dense blocks consist of multiple connected convolution nodes with identical feature map sizes. Dense blocks assist in dealing with the vanishing gradient problem by interconnecting each node as shown in figure 2.1.2, meaning every new convolution node receives input from all prior nodes, ensuring that essential features are saved in the final feature map output. In terms of computation, the dense block grows linearly due to the input size growing accordingly to the number of nodes in the dense block as shown in equation 2.1, where x_0 is the input to the dense block, and x_1,x_2...,x_{n-1} are the corresponding inputs to the convolution layers within the dense block.

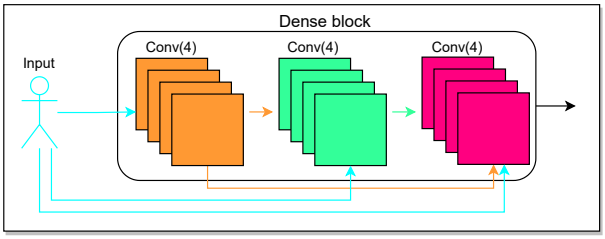


Figure 2.1.2: A simple overview of dense block utilized in the CSPDarknet-53 network as well as other updated backbone networks

$$\begin{aligned} x_1 &= w * [x_0] \\ x_2 &= w * [x_0, x_1] \\ x_n &= w * [x_0, x_1, ..., x_{n-1}] \end{aligned} \tag{2.1}$$

A issue arising from using dense blocks is the problem of redundant gradients within the network. Redundant gradient refers to the problem where the gradient in one layer in the dense block contains the same information as the other layers in the dense block. To tackle this problem, a solution was the introduction of cross-stage partial (CSP) blocks [22]. CSP blocks divide the input into two parts: Part one is passed directly through the CSP block. Part two is passed through a computation block, and in the end, parts one and two are concatenated to generate the final output from the CSP block, as seen in figure 2.1.3.

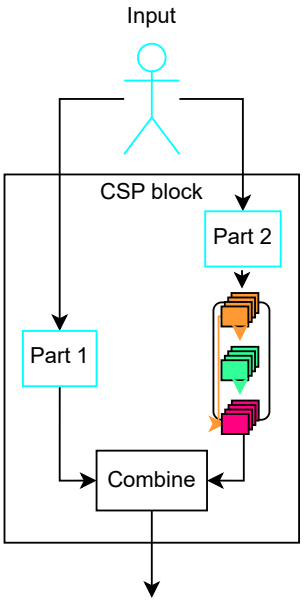


Figure 2.1.3: Simple overview of CSP-block utilized in the CSPDarknet-53 network.

In the CSPDarknet-53 network, the computation block within the CSP block is a dense block. This is not a limiting factor of the CSP block, and other backbones utilizing CSP blocks can use a different computation blocks. The usage of CSP and dense blocks have decreased the computational load of the network substantially increasing their runtime [22].

2.1.3 Neck layer

The neck layer in YOLOv8 utilizes $C2f$ blocks, which differs slightly from the $C3$ block found in the previous YOLOv5 version. The advantage with $C2f$ comes by concatenating all outputs from the bottleneck layers rather than just the output from the final bottleneck layer. This makes the neck layer more sensitive to details in the feature maps as multiple levels of resolution are included in the final aggregation [23].

2.1.4 Prediction layer

Whereas previous YOLO versions, such as YOLOv7 and YOLOv5, have used a coupled module for the prediction of the bounding box and the object, YOLOv8 uses a decoupled head layer, meaning it has one network for predicting the location of the bounding box and another network for class prediction. Using a decoupled head was proposed since researchers observed a misalignment between the classification's confidence and the localization accuracy [24]. The decoupled head utilizes a fully connected network for classification, and a convolution neural network for localization [24]. In addition to the decoupled head, YOLOv8 adopts an anchor-free approach, meaning no predefined boxes are used for predictions. This approach has become increasingly popular in the object detection community [8], and in the YOLO family, it was first incorporated in YOLOX [7]. Anchor-free approaches remove all the hyperparameters connected with anchor boxes, improving the out-of-the-box performance and making the anchor-free model more generalized [8][9].

2.1.5 Transfer learning

Deep learning takes advantage of a large amount of data to learn. Obtaining a large enough dataset is often impossible or too time-consuming to be realistic. By experimenting, researchers found that models trained on one domain can easily be altered to perform the same job on a similar domain. This method is called transfer learning, and it takes advantage of models trained on a similar domain and uses a smaller dataset to optimize the model for the specific domain [19, p. 832]. By optimizing, we utilize the pre-trained model and update the model weights for our purpose by continuing training on our domain. During training, the weights get updated by an optimization algorithm, such as gradient descent [19, p. 695].

2.1.6 Stochastic gradient descent, optimization algorithm

Updating the model weights is necessary to make the model learn a mapping between the input and desired output. In object detection, the optimization of the weights aims to help the model learn the high-level features of an object of interest. The optimization algorithm in YOLOv8 is stochastic gradient descent (SGD), which is based on gradient descent, with the difference being that stochastic gradient descent only utilizes a limited number of data examples for computation. From an example dataset of $N = 1000$, a subset of $n = 10$ can be used for optimizing the model for next iteration rather than the complete $N = 1000$ [19, p. 697]. This massively decreases the computational load and finds similar optimum values compared to gradient descent. In short, SGD tries to minimize the error between the desired output and the actual output, and this error is used to update the weights through backpropagation. The SGD algorithm is as follows [19, p. 695]:

Algorithm 1 Stochastic gradient descent

1: $\mathbf{w} \leftarrow$ any point in the parameter space
2: **while not** converged **do**
3: **for each** w_i **in w do**
4: $w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} \text{Loss(w)}$

Where α is denoted as the learning rate, which defines how quickly and with what rate the model should adapt to the problem.

$$\frac{\partial}{\partial w_i} \text{Loss(w)} = -\alpha(y - h_w(x)) \tag{2.2}$$

Inserting equation 2.2 into the algorithm 1 we get the following equation 2.3 representing the learning rule.

$$w_0 \leftarrow w_0 + \alpha(y - h_w(x)); w_1 \leftarrow w_1 + \alpha(y - h_w(x)) \times x \tag{2.3}$$

2.1.7 Backpropagation

Backpropagation refers to passing the error generated from the loss function backward in the network. This error is then used to update the weights according to the learning rule from equation 2.3 [19, p. 806].

2.2 Model training, hyperparameter optimization, overfitting and underfitting

An important aspect for efficiently training models and achieving a generalized model is appropriately selecting the models hyperparameters. Researchers have shown how proper selection of hyperparameters can boost model performance substantially [25][26]. Another benefit of correctly selecting the hyperparameters is to avoid the model under- or overfitting. For a simple understanding, an underfitted model does not achieve the ability to find any pattern in the training data. On the contrary, an overfitted model goes too far and memorizes the training data and loses the ability to perform on new unseen data [19, p. 673]. Generally, underfitting is easily solved by increasing the training, whereas overfitting has been the subject of much research to find suitable techniques to deal with the issue.

2.2.1 Weight decay

A popular technique to deal with overfitting is the introduction of weight decay which is an additional term in the loss function that ensures that the loss function does not grow out of control. The concept of weight decay existed before the introduction of deep learning models [27], but it is still very effective for deeper models. In modern terms, weight decay is often referred to as L2-regularization. By representing the loss function as L_0, the new loss function with L2 regularization becomes:

$$L(w) = L_0(w) + \lambda \sum_{i=1}^{N} w_i^2 \tag{2.4}$$

2.3 Anonymization filter

Anonymization has traditionally been done using methods such as blurring [28]. Although multiple blurring methods exist, such as gaussian blur and average blur, they share similar concepts, with the main difference being how much value they grant to each pixel inside the kernel. Presented in equation 2.5 and 2.6 is average blur. To achieve average blur, we define a kernel representing the size of the area for the blurring computation. The kernel is applied on each pixel in the image, and the center pixel value is replaced by the average value from the kernel as shown in equation 2.5.

$$K = \frac{1}{ab} * \lceil a \times b \rceil \tag{2.5}$$

With an example kernel size of 3×3 the equation 2.5 turn into equation 2.6.

$$K = \frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.6}$$

2.3.1 Improved generalization with data augmentation

As mentioned previously, a central problem with deep learning is the need for large amounts of data. This has resulted in a large amount of research on augmentation techniques where one artificially increases the amount of data by altering the existing data

to appear to be a new photo instead of collecting new raw data. The results from these augmentation techniques, which include, among others, rotating, cropping, and noise adding, have been shown to generate similar and even improved performance. An additional benefit of augmentation techniques is the creation of more robust models that are less sensitive to image variations [10]. For small object detection, augmentation techniques such as mosaic have shown promising results when the domain contains complex backgrounds, such as small object detection domains [29]. Mosaic combines four dataset examples into a new artificial image with increased complexity. This helps the model generalize for small objects by increasing the training complexity artificially.

2.4 Open Images V7 dataset

Following the difficulties of containing high-quality datasets for deep learning purposes, the research team behind the Open Images dataset hoped to help in this domain. Their dataset, currently in version seven, consists of nine million labeled images available under the CC-BY license. According to the research team, with easy access to a massive multi-class dataset, they want to encourage further research within the field of object detection. To ensure a high quality of the examples within the dataset, multiple pre-processing steps defined below were performed by the team [30]:

1. Near-duplicated images are removed.
2. Using safety filters on Flickr, Google SafeSearch, inappropriate images are removed.
3. Remove images that appear elsewhere on the internet. Ensures no invalid contributions according to CC-BY license.
4. Validate the labeling and boudnign box.

2.5 Evaluation metrics

The following section presents the metrics used for the evaluation methods. Proper evaluation metrics are essential for assessing the results presented in later chapters. Initial terminology for performance terminology is presented below:

- True positive (TP), detection is correctly classified as positive.
- False positive (FP), detection is falsely classified as positive.
- True negative (TN), detection is correctly classified as negative.
- False negative (FN), detection is falsely classified as negative.

Recall evaluates the models ability to detect true positive cases compared to the total number of positive cases in the data. Calculation shown in equation 2.7

$$recall = \frac{TP}{TP + FN} \tag{2.7}$$

Precision evaluates the model based on the ratio of how many of the positive predicted classes were positive. Precision is calculated as shown in equation 2.8

$$precision = \frac{TP}{TP + FP} \tag{2.8}$$

When dealing with a class-balanced dataset, a reliable metric to evaluate the models performance is the F1 score. Evaluating models with F1 score has the advantage of utilizing both precision and recall, generating a more general score. The F1 metric ranges from $0 - 1$, with a score of 1 being perfect. F1 score is calculates as shown in equation 2.9

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{2.9}$$

Confidence interval makes that the reported results more reliable by being the mean of the samples +/- some variance. The confidence interval is calculated as shown in equation 2.10. $\overline{x}$ represents the sample mean, z represents the z-score for the selected confidence interval, s is the sample standard deviation and n is the sample size.

$$confidenceinterval = \overline{x} \pm z * \frac{s}{\sqrt{n}} \tag{2.10}$$

CHAPTER

THREE

METHODS

3.1 Pre-proseccing

Presented are the additional pre-processing steps performed on the additional data downloaded from the open images v7 dataset, as well as the data augmentation methods utilized.

3.1.1 Open Images V7 extension

Due to the processing steps already performed on the dataset, as mentioned in the theory chapter and section 2.4, the additional work required before utilizing the dataset was decreased compared to collecting a similar amount of images from scratch. The additional pre-processing steps consist of removing images not within the domain of interest and removing images of lousy quality based on personal evaluation. From the Open Images dataset V7, a total of 5000 face images were downloaded. Using tools provided by Roboflow, a manual review of each image was performed. During the manual review, the requirements shown in table 3.1.1 were used to decide whether an image should be kept or deleted. Additional work was performed to ensure the bounding boxes were correctly placed and that all faces were labeled.

1. Images containing zero human faces.
2. Highly obscured faces.
3. Black and white images.
4. Highly blurred images.
5. Large single detection images.

Table 3.1.1: Extended dataset image criteria for manuall review

These five defined criteria ensure consistency throughout the manual review and that the images in the extended dataset have similar quality. From the initial downloaded dataset of 5000 images, 3043 images remain after the manual review. The majority of the deleted images fall under the category of large single-detection images. These were removed due to it not being relevant to our use case. Shown in figure (b)3.1.1 is a example of a

deleted image containing a large single detection, additional examples of deleted images can also be seen in (a)(c) 3.1.1. During the manual review it also became evident that compared to the original widerface dataset, our extension consists of less complex scenes. To increase the complexity of our extended dataset, we create a new version using mosaic augmentation. This should increase the complexity of the dataset, and possibly improve the models ability to detect smaller objects such as faces. An example of an image after mosaic has been applied can be seen in figure 3.1.2.

(a) **(b)** **(c)**

Figure 3.1.1: (a) black and white image (b) large single detection (c) highly obscured image

After generating the mosaic dataset version, an additional manual review was necessary to ensure that the bounding boxes were correctly placed. A common issue currently with the mosaic tool from roboflow is misaligned bounding boxes after mosaic augmentation has been applied.

Figure 3.1.2: Example image with applied mosaic augmentation, the new image consist of four images combined.

Evaluation will be performed on all dataset variations to validate the effect of the different methods applied on the dataset. This means that a total of four dataset variations will

be evaluated. All variations are listed in table 3.1.2. The performance of the datasets will be assessed based on the performance of the YOLOv8 model after being trained on each of the dataset variations.

1. Widerface.
2. Widerface + extended.
3. Widerface + extended filtered.
4. Widerface + extended filtered and augmented.

Table 3.1.2: The four variations of the dataset, including the original widerface.

3.2 YOLO variation selection

From YOLOv8 it exists five variations. For this thesis, three of these variations are selected to represent various complexity, with the smallest of the selected models being the YOLOv8n and the largest being the YOLOv8l. A comparison of all the YOLOv8 models with their network sizes is represented in table 3.2.1, with the selected models for this thesis being highlighted in bold text.

Model variance	Params(M)	FLOPs(B)
YOLOv8n	3.2	8.7
YOLOv8s	11.2	28.6
YOLOv8m	25.9	78.9
YOLOv8l	43.7	165.2
YOLOv8x	68.2	257.8

Table 3.2.1: Model size for all the YOLOv8 variations. The models used in the thesis is highlightes in bold text.

3.3 Implementation

Due to the size of the datasets used, it is necessary to have a system with enough computational capacity to train our model within a reasonable time. For the thesis, training of the models has been done utilizing the IDUN HPC cluster, which has enabled the training of multiple models simultaneously and with much faster training times compared to using personal computers. The usage of the cluster also helps avoid any storage limitations both in terms of storing the data and during training of the models.

3.3.1 IDUN HPC cluster

The IDUN high-performance computing (HPC) cluster is a powerful tool available to researchers at Norwegian University of Science and Technology. The goal of the IDUN cluster is to assist researchers with computational power and storage in cases where it becomes cumbersome to do it on ones personal computer. For this project, training of YOLOv8 models with the presented datasets was only possible with the usage of the IDUN HPC cluster. The general connection structure to the cluster is shown in figure 3.3.1.

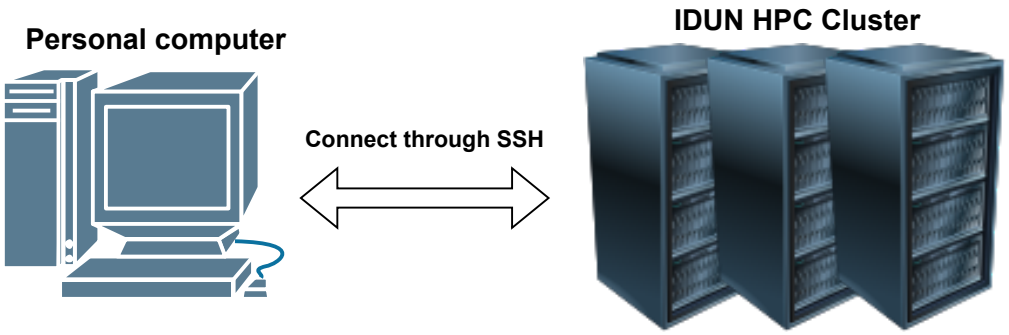


Figure 3.3.1: Connection to the HPC cluster is achieved through SSH, where each user controls their environment.

The IDUN cluster has multiple types of GPUs available. For our project and training of the YOLOv8 models, up to three Tesla V100 GPUs with 16GB memory was utilized, which made training very efficient.

3.3.2 System selection

For testing our models, and for performing the comparison study, a total of four systems have been used. The systems vary in computational power, from the least powerful being the jetson nano and the most powerfull being the Alienware stationary computer. Both the CPU and GPU from the two computers will be used, resulting in six total testing "systems".

1. **Jetson Nano** : Entry level embedded device for machine learning purposes from Nvidia.
2. **Jetson Orin** : High end embedded device for machine learning purposes from Nvidia.
3. **Acer Nitro Spin** : Low-end personal computer with older generation GPU (Nvidia 1030 GTX), and CPU (Intel I5)
4. **Alienware** : High-end personal stationary computer with new generation GPU (Nvidia 2060 RTX), and CPU (Intel I9).

Table 3.3.1: The four systems selected for the comparison study. The systems varies from low-end to high-end, and from embedded to non embedded.

3.3.3 YOLOv8, model training

The training of the models was done on the IDUN HPC cluster as described in section 3.3.1. Within the IDUN cluster, a python environment was set up with the necessary libraries to train the YOLOv8 models. To avoid extensive training, YOLOv8 models trained on the COCO dataset were used as starting point for our training, and optimized for face detection using the concepts of transfer learning as described in section 2.1.5. Hyperparameter tuning was done to find an optimal value for the learning rate. The other hyperparameters were left at default values. Among the training parameters offered by the YOLO training environment, the mosaic option was turned off due to it being shown to decrease performance when used on the widerface dataset [31]. In total, nine models were trained using the different variations of the dataset as shown in table 3.3.2.

Model	Widerface	Extended	Extended filtered	Mosaic
YOLOv8n	**YES**	**YES**	**YES**	**YES**
YOLOv8m	**YES**	**YES**	**YES**	**YES**
YOLOv8l	**YES**	**YES**	**YES**	**YES**

Table 3.3.2: A total of nine models were trained, three of each of the models as shown in the table above. YOLOv8n is the smallest model, YOLOv8m medium large, and YOLOv8l the largest of the three.

3.3.3.1 Input resolution sizes

Models are trained on two input resolutions. The first is the default size of YOLOv8 models, $640x480$ pixels. The other chosen resolution is $416x312$ pixels. This resolution is picked due to earlier models, such as YOLOv3, showing promising results with this resolution without substantially sacrificing the model performance [3].

3.4 Anonymization filter implementation

Anonymization is applied by altering the YOLOv8 code. By utilizing the output from YOLOv8 before it is presented to the end user, we are able to extract the bounding box coordinates. These coordinates are then used to copy the selected area on the image. This copied image is then passed through the anonymization algorithm consisting of blur and noise filter. The anonymized image is then copied back into the original image, which is then returned to the end-user as the final output. The general flow of the anonymization is showcased in algorithm 2.

Algorithm 2 Anonymizing output from YOLOv8

Precondition: Set of predicted bounding boxes *box*, noise variable N

```
1: function ANONYMIZE(img,boxes)
2:     if boxes ≠ 0 then
3:         for box in boxes do
4:             Extract width, height, x and y coordinates from box
5:             img0 ← img[y:y+height, x:x+width]
6:
7:             Apply anonymization filter(img0)
8:             img[y:y+height, x:x+width] ← img0
9:         end for
10:    end if
11:    return img
12: end function
```

3.5 Inference test framework

To ensure similar testing conditions, a simple testing framework is created. For evaluating the real-time performance, the main concern is the frames per second (fps) that the system can maintain during runtime. For this, ten videos are selected with varying complexity from zero to fifty detections. Output from each run is the average fps over the video. After each video has been run on the systems, we use the ten average fps values to calculate

the 95% confidence interval. The equation for calculating the confidence interval is given by equation 2.10. The flow can be visualized in figure 3.5.1

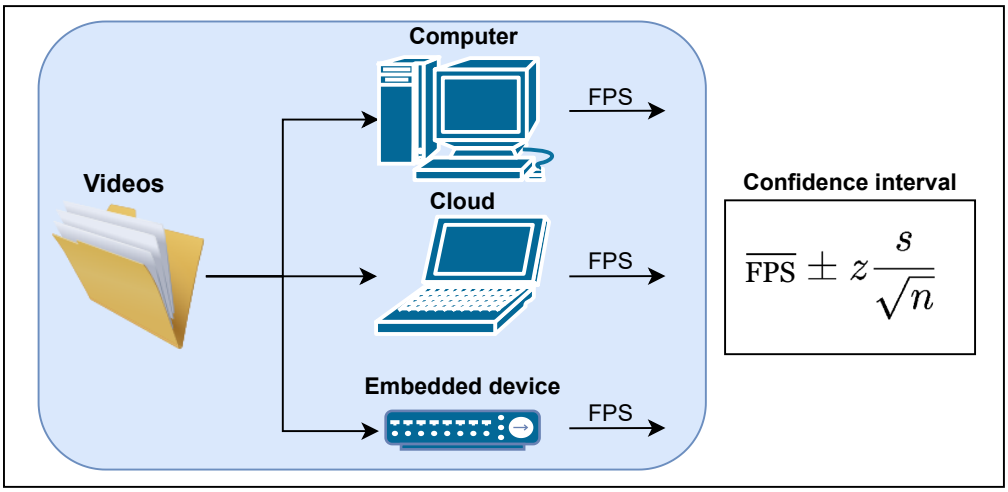


Figure 3.5.1: Testing framework for measuring the real-time capabilities of the systems.

CHAPTER

FOUR

RESULTS

4.1 Object detection inference results

The smallest of the YOLOv8 models, the YOLOv8n has been used to generate the presented inference results. The selected systems used for the real-time tests were presented in section 3.3.2. In addition, all models presented in the following chapter have used identical training hyperparameters.

4.1.1 Inference results

Inference results are presented for all the systems with a 95% confidence interval. In figure 4.1.1, the inference results with an input size of 640×480 pixels are presented.

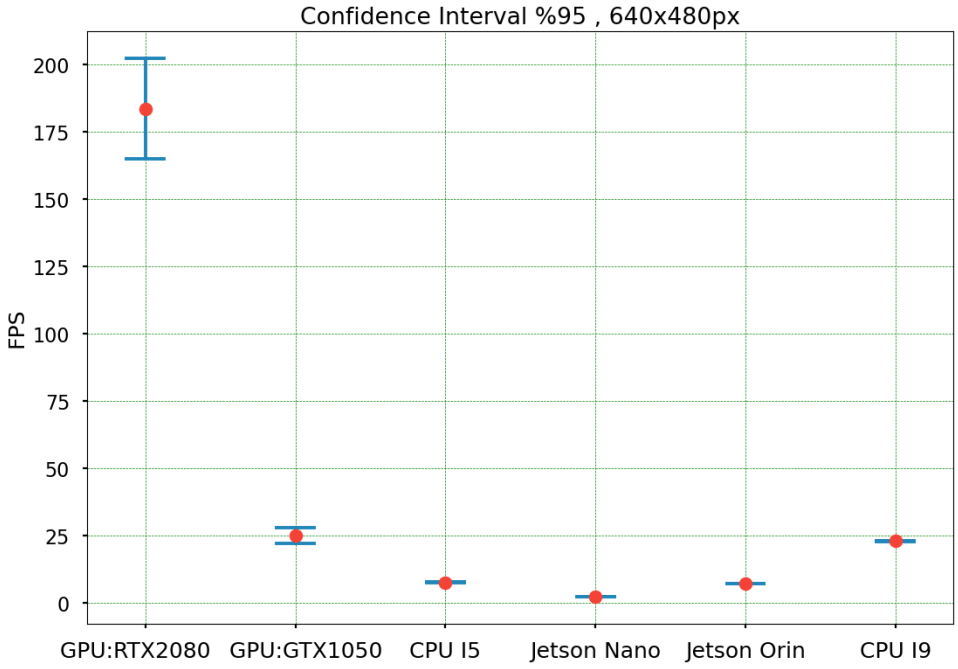


Figure 4.1.1: Model inference running on 640×480px image input

4.1.2 Inference on decreased input size

In figure 4.1.2 the inference results with an input size of 416×312 pixels is presented.

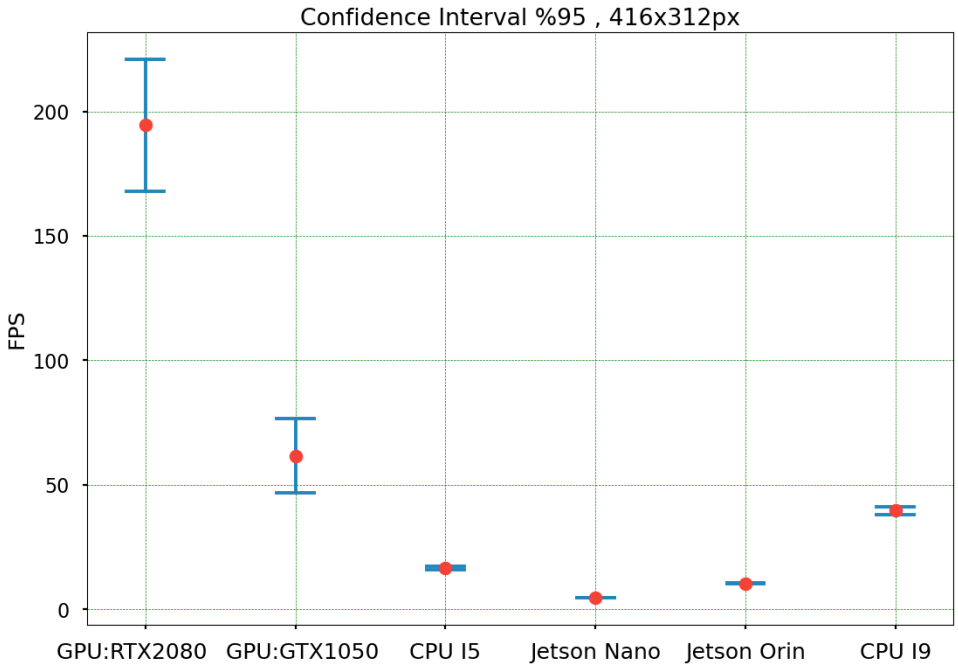
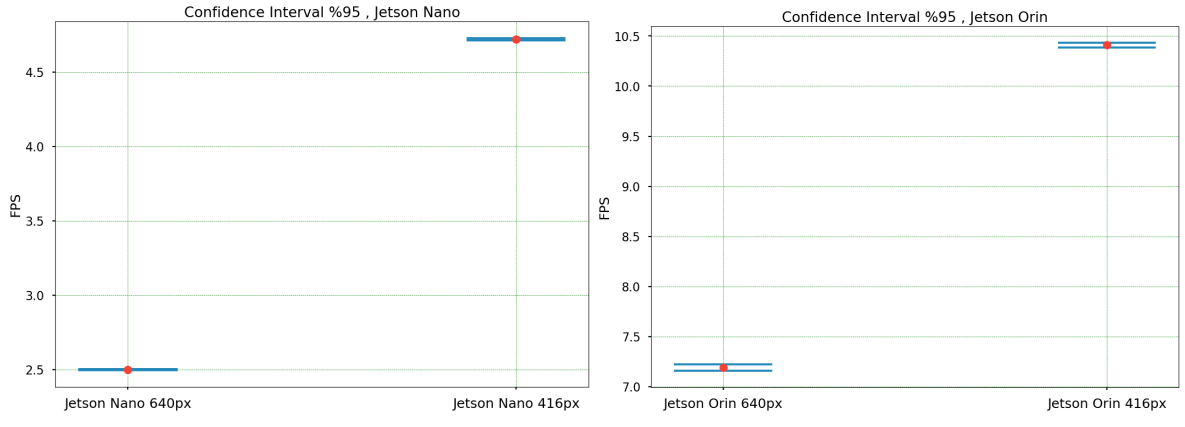


Figure 4.1.2: Model inference running on 416×312px image input

4.1.3 Inference comparison on the systems, 640×480px Vs 416×312px

Following is a comparison of each system with the two defined input sizes. At the end of the section, the table 4.1.1 will summarize the result will be presented.

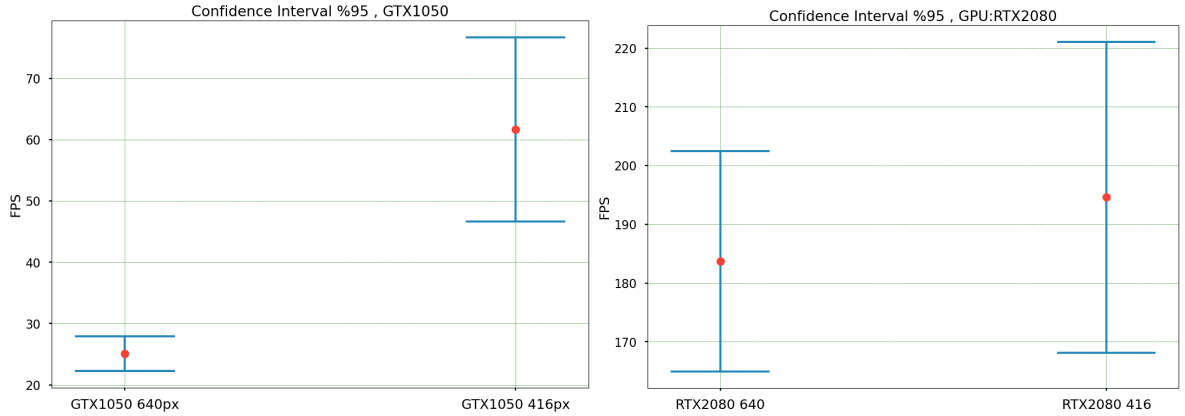
In figure 4.1.3, the inference comparison for the embedded devices, jetson nano and jetson orin is presented.



(a) **(b)**

Figure 4.1.3: (a) Jetson Nano, (b) Jetson Orin

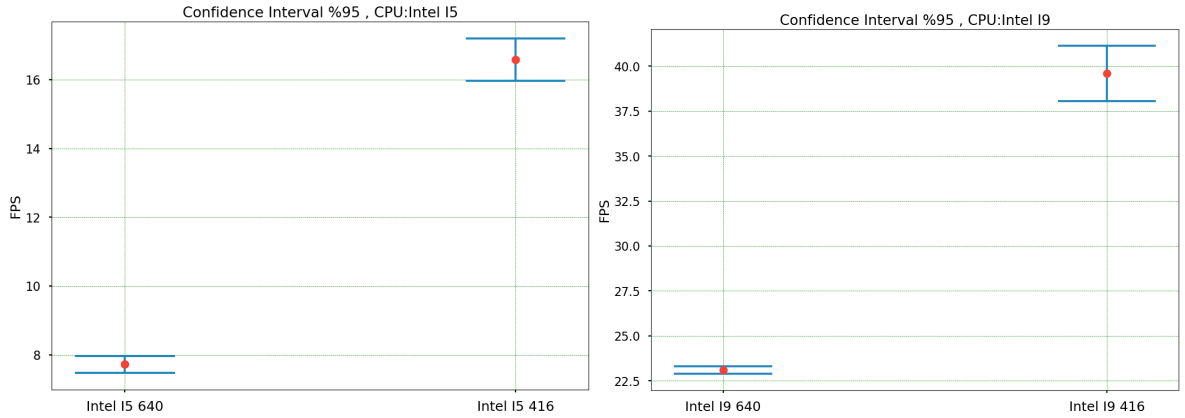
Presented in figure 4.1.4 is the inference results for the two GPU units used for testing, the GTX 1050 (a) and RTX 2080 (b).



(a) **(b)**

Figure 4.1.4: (a) GTX1050 (b) RTX2080

In figure 4.1.5, the inference result from the CPU units, Intel I5 (a) and Intel I9 (b) is presented.



(a) **(b)**

Figure 4.1.5: (a) Intel I5 (b) Intel I9

A combined overview of the performance of the system with regard to frame per second during runtime is summarized in table 4.1.1.

System	**Average fps with 640×480px**	**Average fps with 416×312px**
Jetson Nano	2.5	4.7
Jetson Orin	7.2	10.4
GTX 1050	25.1	61.7
RTX 2080	183.7	194.6
Intel I5	7.7	16.6
Intel I9	23.1	39.6

Table 4.1.1: Numeric summary of the performance comparison on the systems with the two different image resolution input.

4.1.4 Performance comparison with different input sizes

The following section presents how the decrease in input size affects the model performance. First is the presented precision score during model training shown in figure 4.1.6.

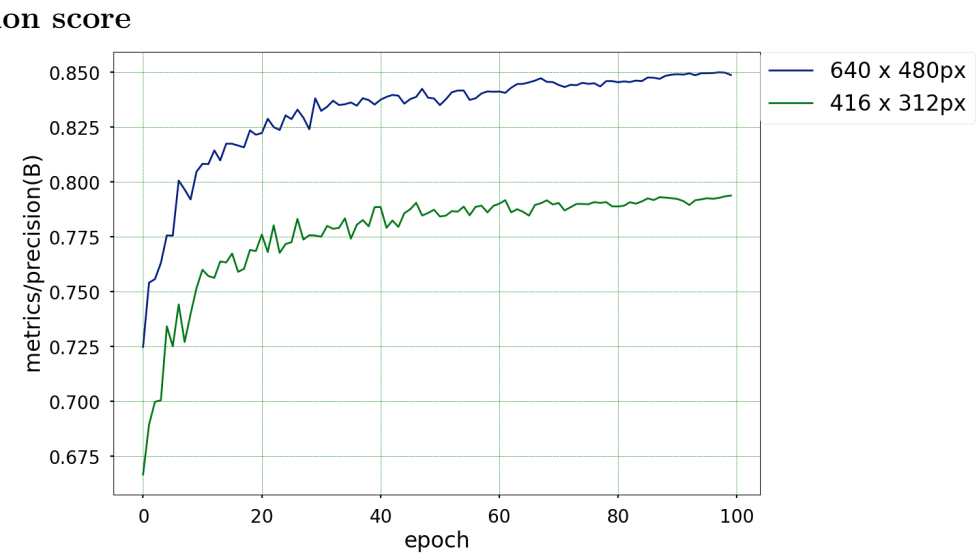


Figure 4.1.6: Precision score for YOLOv8n with input size 640×480px and 416×312px.

In figure 4.1.7 the recall score during training is presented.

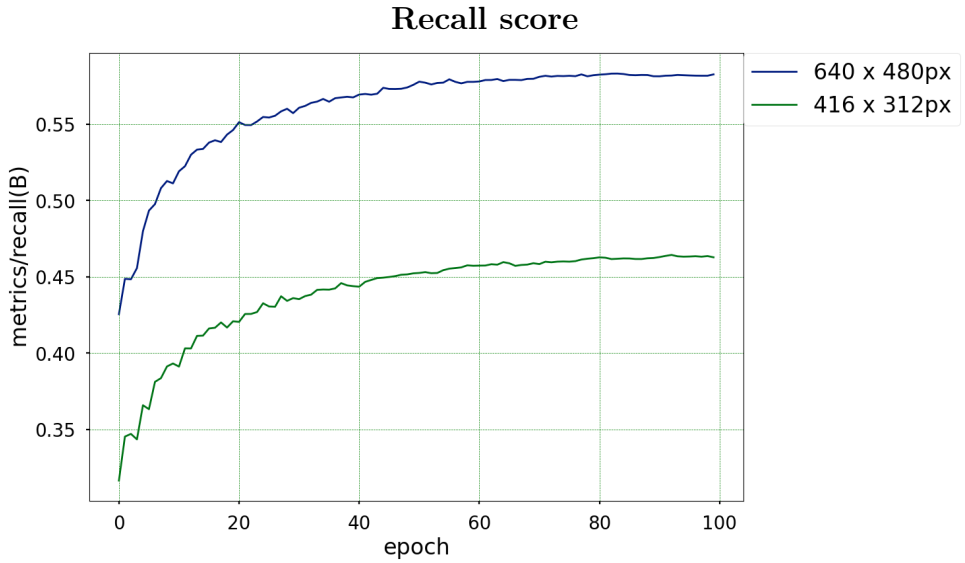


Figure 4.1.7: Recall score for YOLOv8n with input size 640×480px and 416×312px.

Final training score, the F1 is presented in figure 4.1.8.

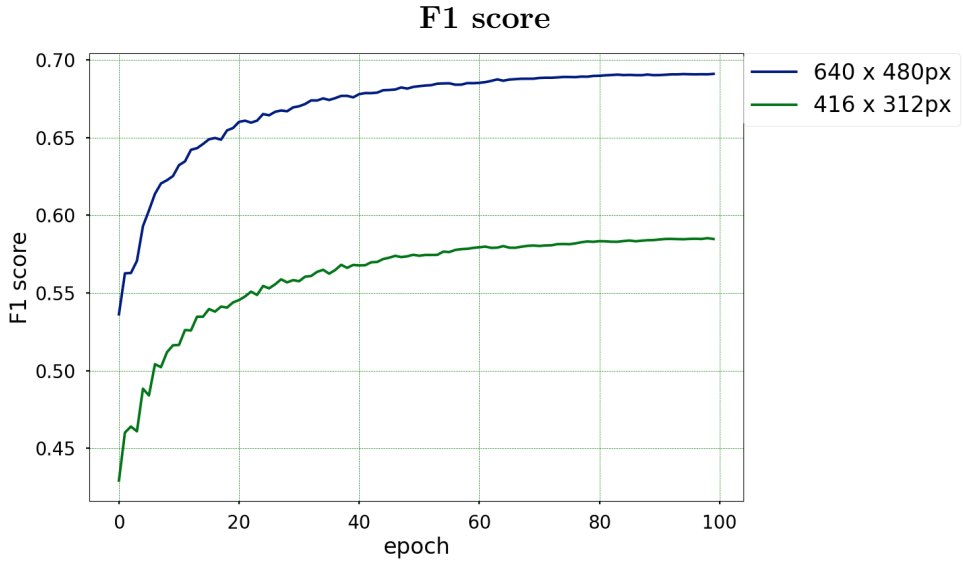
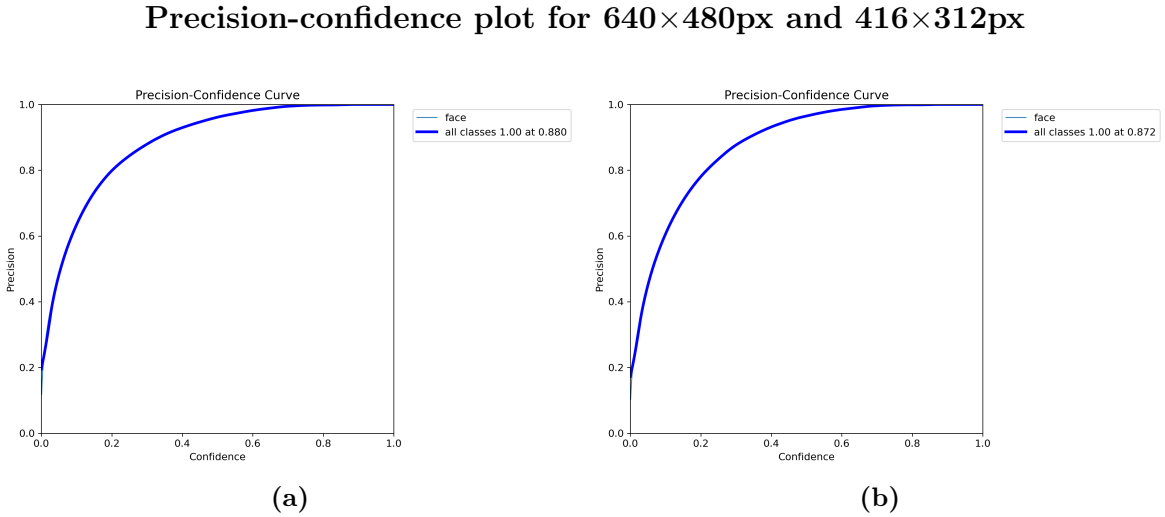


Figure 4.1.8: F1 score during training of yolov8n models with two different image size as input.

Figure 4.1.9 presents the precision-confidence curve representing how the precision evolves relative to the confidence score.

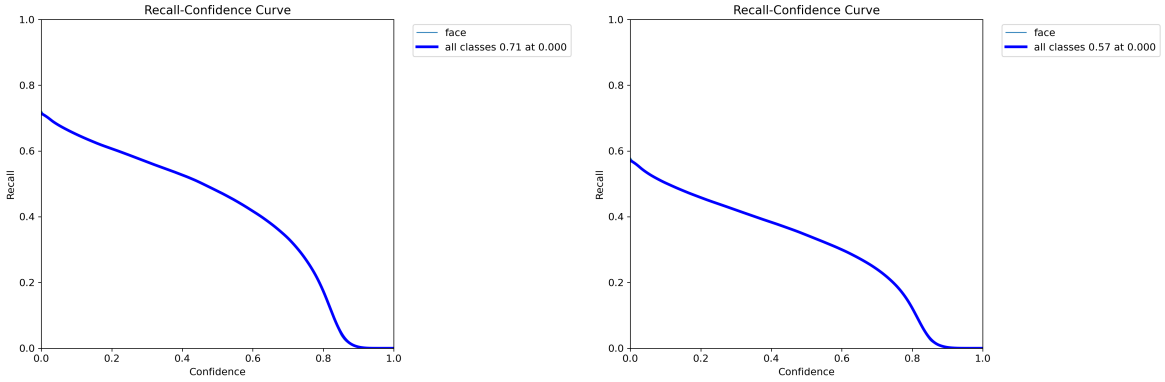


(a) **(b)**

Figure 4.1.9: Precision confidence curve for two resolution sizes: (a)640×480px (b)416×312px

Figure 4.1.10 presents the recall-confidence curve representing how the precision evolves relative to the confidence score.

Recall-confidence plot for 640×480px and 416×312px

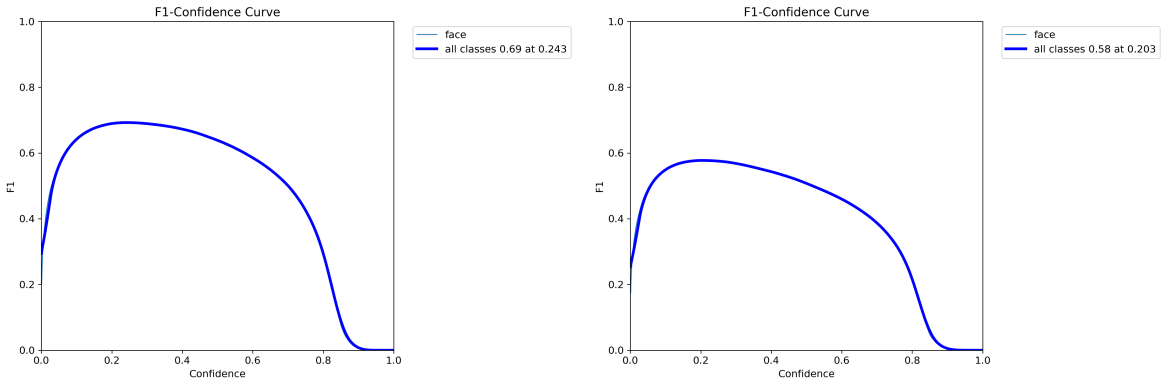


(a) **(b)**

Figure 4.1.10: (a) 640×480px (b) 416×312px

Figure 4.1.11 showcases the F1 score relation to the confidence score.

F1-confidence plot for 640×480px and 416×312px



(a) **(b)**

Figure 4.1.11: (a) 640×480px (b) 416×312px

Performance evaluation for final training epoch is shown in table 4.1.2.

Evaluation metric	**640×480px**	**416×312px**
Precision	84.9%	79.8%
Recall	58%	46,4%
F1 score	68.9%	58.7%

Table 4.1.2: Numeric comparison of performance on 640×480px input size and 416×312px based on the final training epoch.

Performance evaluation on the validation data shown in table 4.1.3.

Evaluation metric	640×480px	416×312px
Precision	83.1%	77.9%
Recall	57.1%	43.6%
F1 score	67%	55.9%

Table 4.1.3: Numeric comparison of performance on 640×480px input size and 416×312px on validation data.

4.2 Dataset benchmark model

The following section presents initial models trained on the original widerface to generate a benchmark for comparison.

4.2.1 YOLOv8 benchmark

Presented in figure 4.2.1 is the precision and recall evolution during training for each of the benchmark models, YOLOv8n, YOLOv8m, and YOLOv8l.

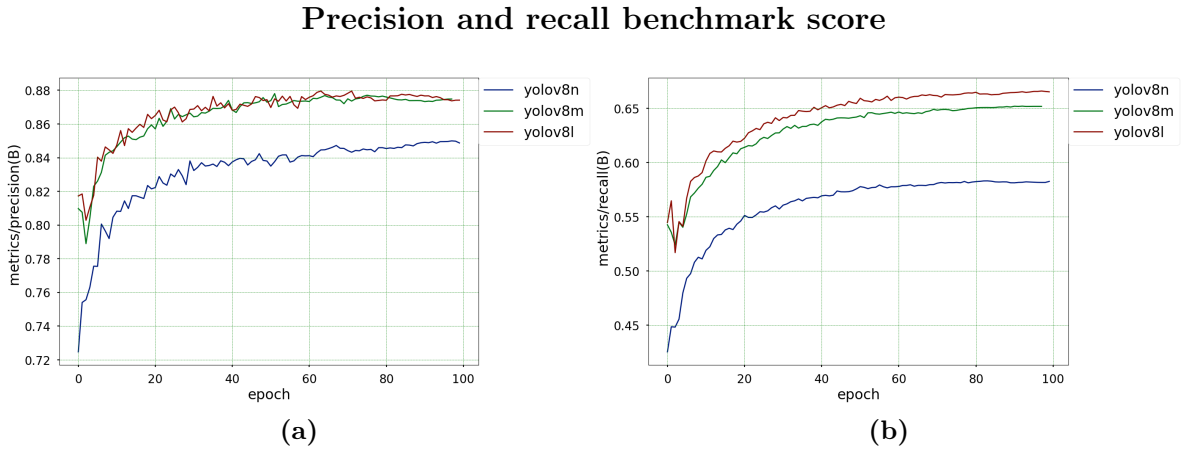


Figure 4.2.1: (a) Precision and (b) Recall score of the YOLOv8 (nano, medium and large) benchmark models.

F1 score of the benchmark models is shown in figure 4.2.2.

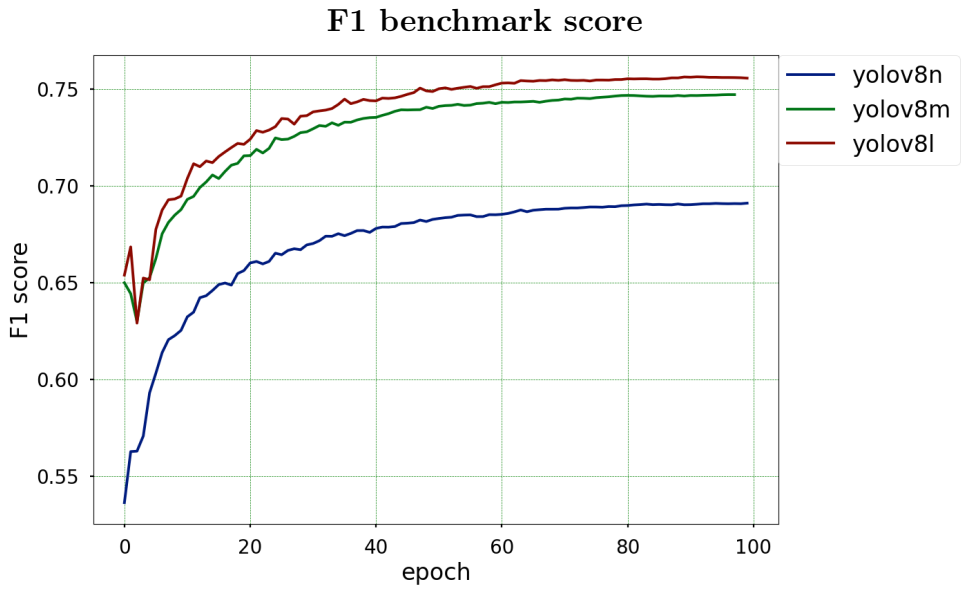


Figure 4.2.2: F1 training score of the YOLOv8 (nano, medium and large) benchmark models.

Numeric performance overview for the benchmark models.

Model	Precision	Recall	F1
YOLOv8n	84.9%	58.2%	69.1%
YOLOv8m	87.4%	65.2%	74.7%
YOLOv8l	87.4%	66.6%	75.6%

Table 4.2.1: Numeric comparison of performance for the benchmark models.

4.3 Dataset comparison compared to benchmark

The following section will compare the benchmark models performance to that of the models trained on the newly created datasets explained in previous chapters. See table 4.3.4 for a quick overview of the datasets.

4.3.1 Benchmark models comparison

Initial benchmark analysis for all the models trained on the different datasets is presented in figure 4.3.1.

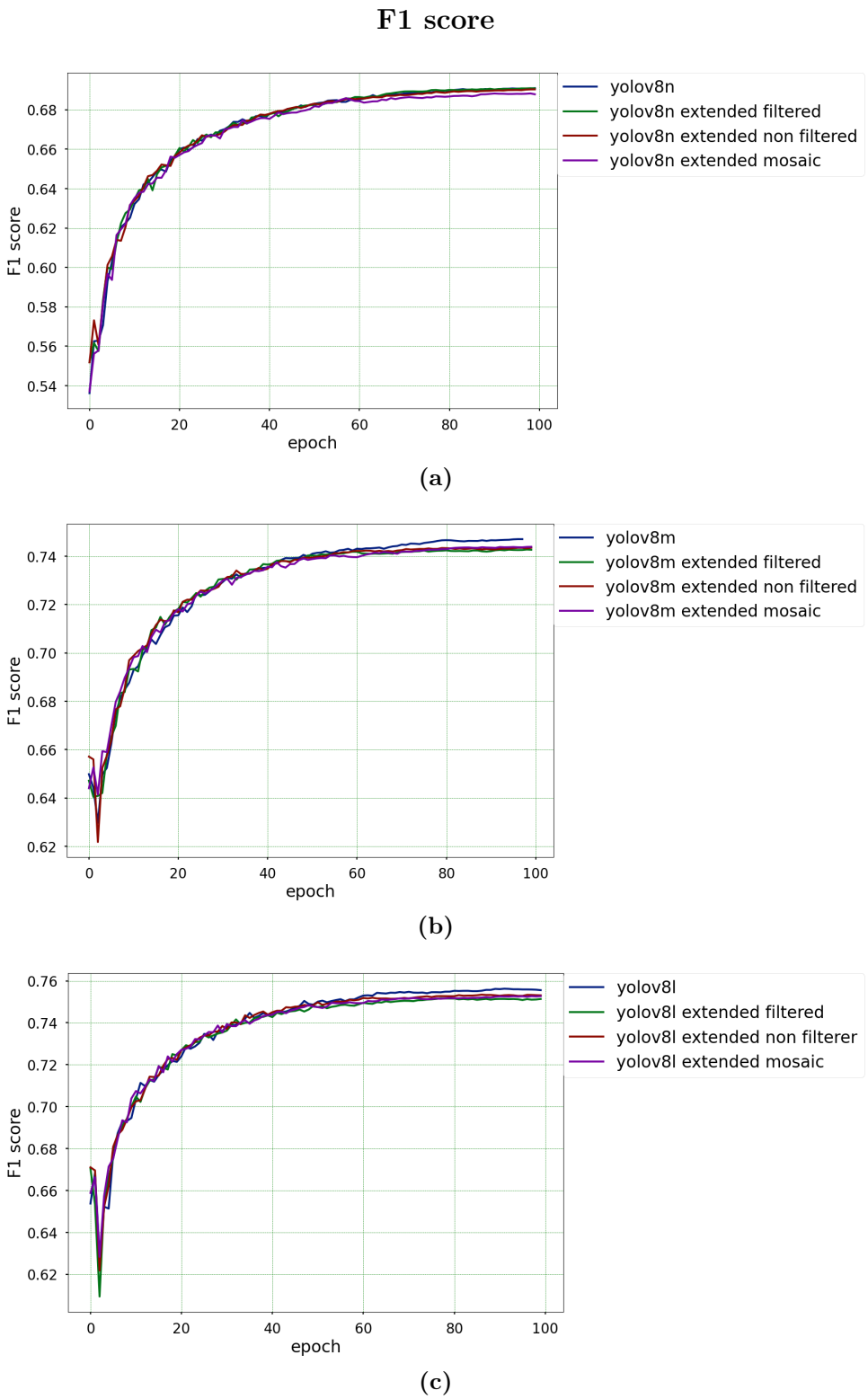


Figure 4.3.1: F1 score: (a) YOLOv8n (b) YOLOv8m (c) YOLOv8l

In figure 4.3.2 we present the evolution of the precision score during training plotted

against the benchmark.

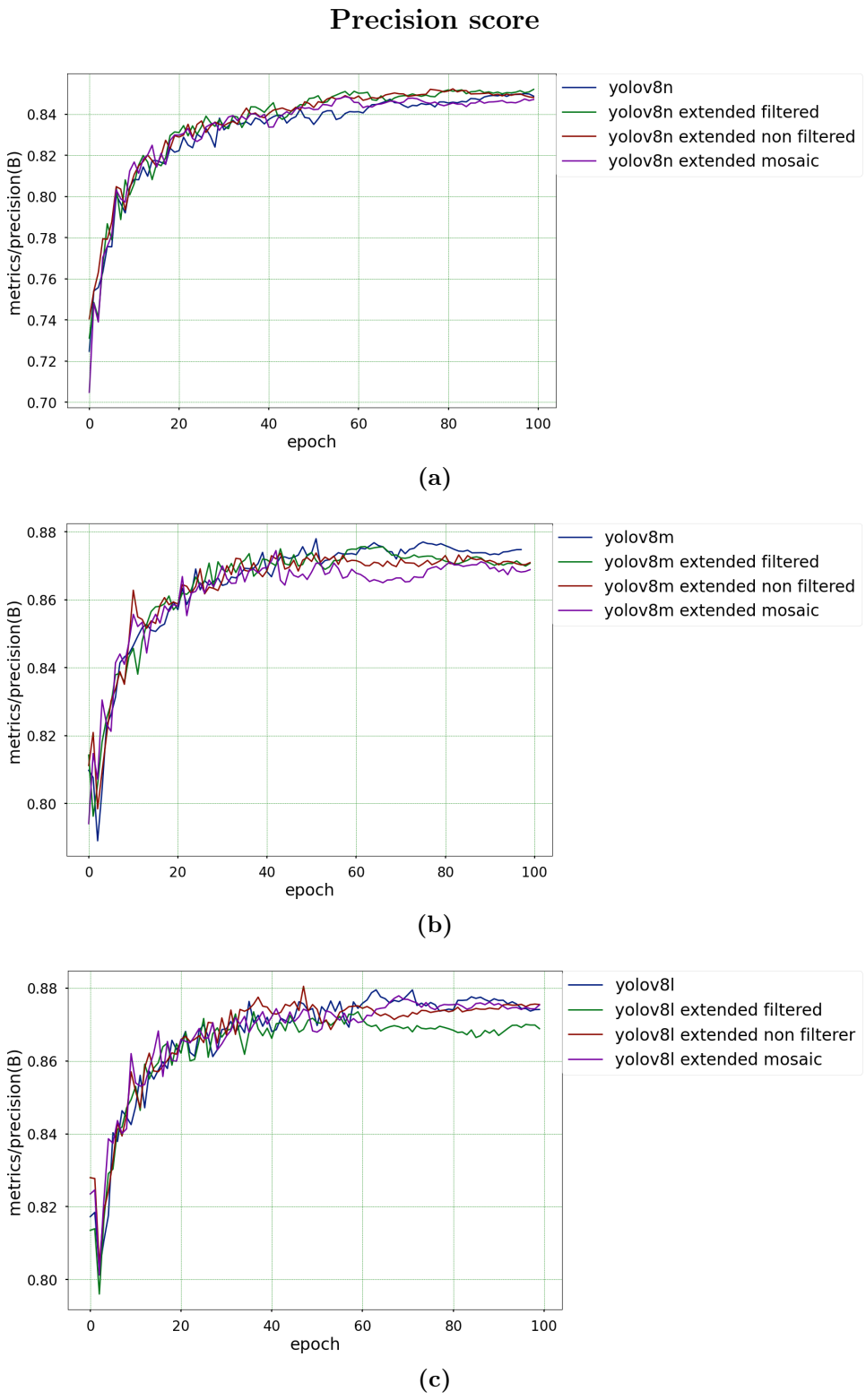


Figure 4.3.2: Precision score: (a) YOLOv8n (b) YOLOv8m (c) YOLOv8l

Lastly for initial comparison the recall score from the models training is shown in figure 4.3.3 plotted against the benchmark.

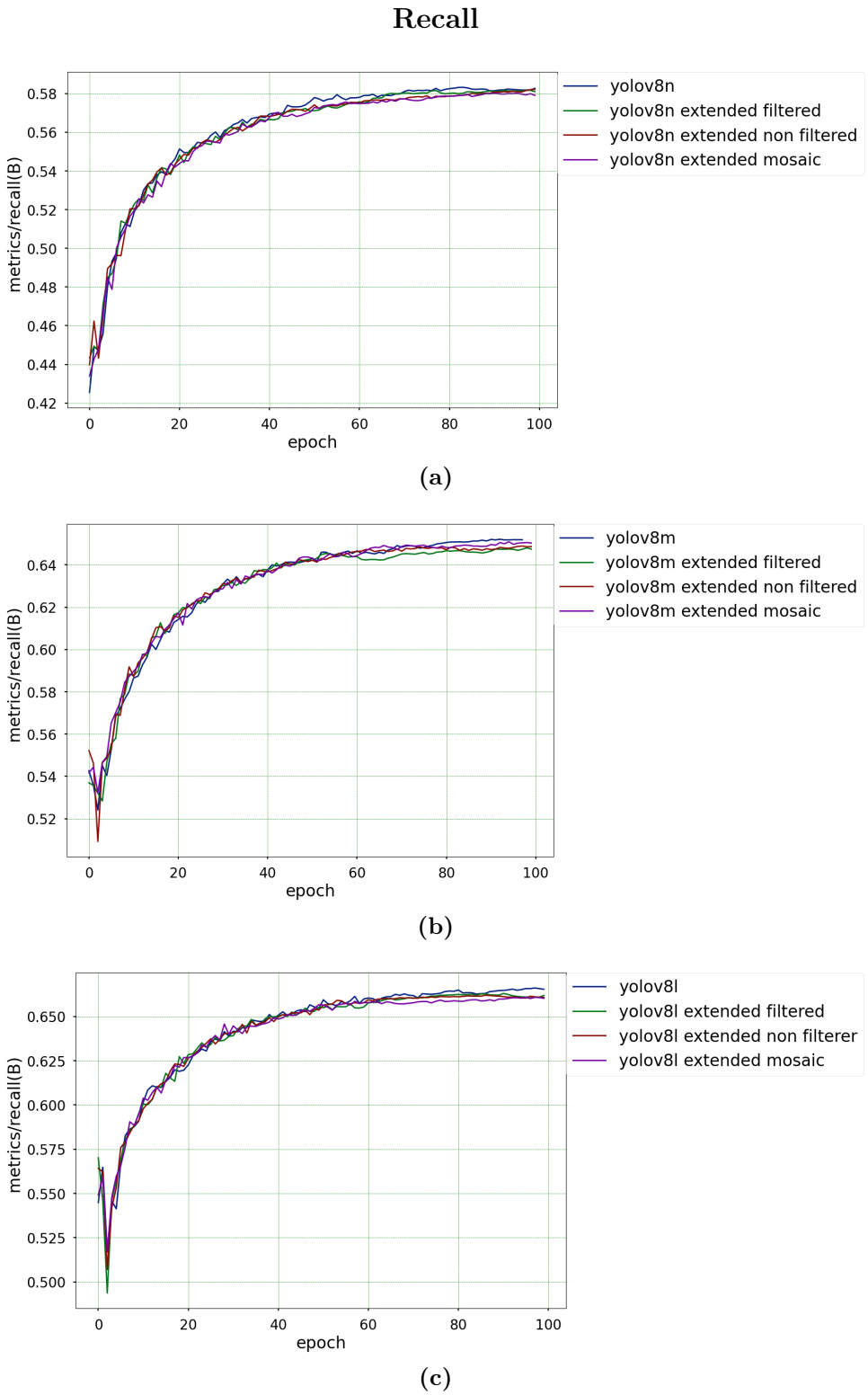


Figure 4.3.3: Recall score: (a) YOLOv8n (b) YOLOv8m (c) YOLOv8l

Following is a numeric summary of the above figures. In table 4.3.1 is the comparison for the YOLOv8n models, in table 4.3.2 is the comparison for YOLOv8m models and for comparison of the YOLOv8l models see table 4.3.3.

Model	Precision	Recall	F1
YOLOv8n	84.9%	58.2%	69.1%
YOLOv8n non filter	84.8%	58.2%	69.0%
YOLOv8n filter	85.2%	58.1%	69.1%
YOLOv8n mosaic	84.7%	57.9%	68.9%

Table 4.3.1: Numeric comparison of performance for all the variations for YOLOv8n.

Model	Precision	Recall	F1
YOLOv8m	87.4%	65.2%	74.7%
YOLOv8m non filter	87.1%	64.9%	74.4%
YOLOv8m filter	87.1%	64.7%	74.3%
YOLOv8m mosaic	86.9%	65%	74.4%

Table 4.3.2: Numeric comparison of performance for all the variations for YOLOv8m.

Model	Precision	Recall	F1
YOLOv8l	87.4%	66.6%	75.6%
YOLOv8l non filter	87.5%	66%	75.3%
YOLOv8l filter	86.9%	66.2%	75.1%
YOLOv8l mosaic	87.5%	66%	75.3%

Table 4.3.3: Numeric comparison of performance for all the variations for YOLOv8l.

4.3.2 Datasets label size

Presented are general analytics of the dataset and the contents within. Shown in figure 4.3.4 are plots showing instances of faces in the datasets, size of bounding boxes, location of the instance, and size of the instance relative to the image.

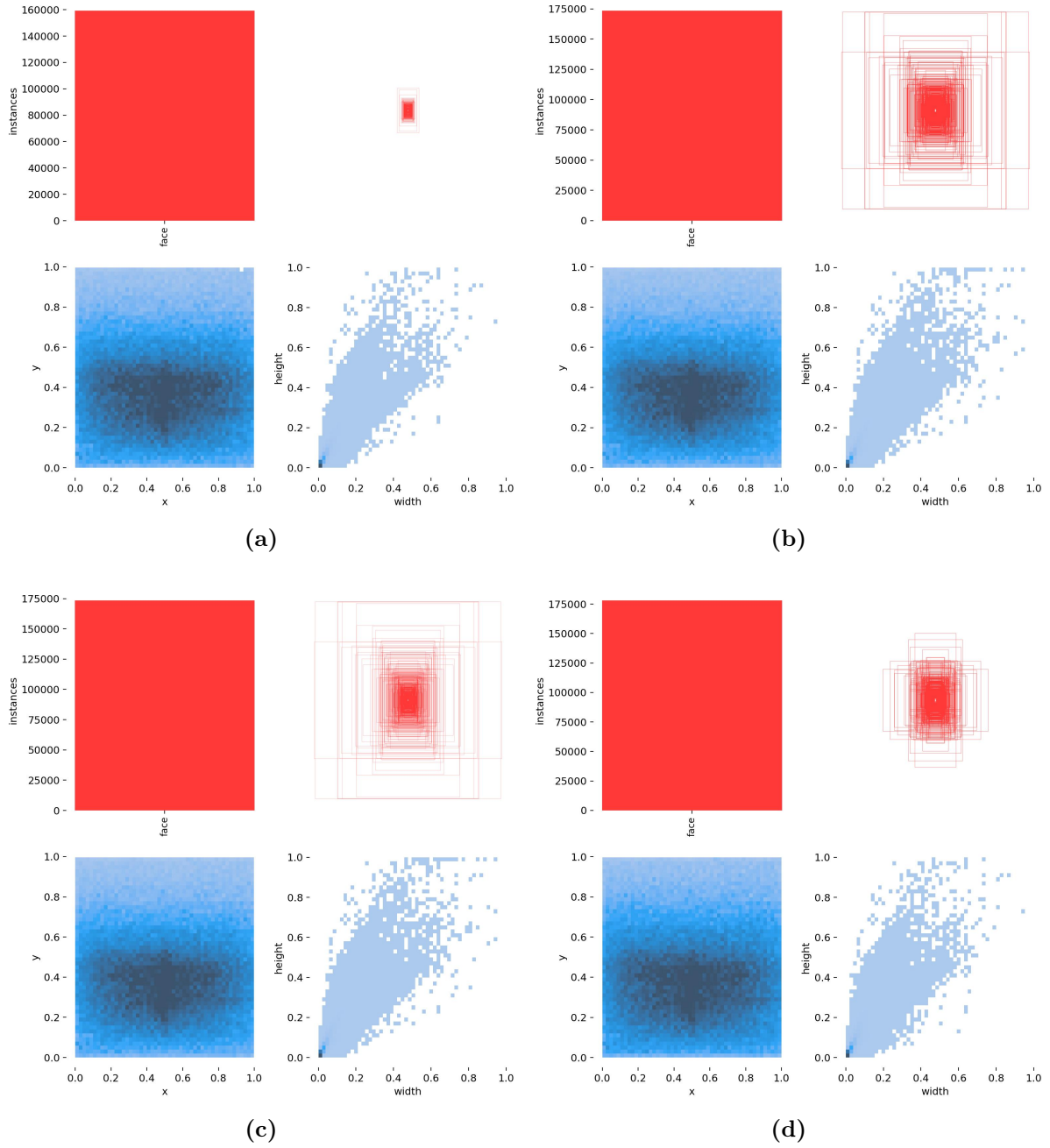


Figure 4.3.4: (a) Original widerface dataset (b) Extended dataset without any additional pre-processing steps (c) Extended dataset with additional pre-processing steps (d) Extended dataset with mosaic augmentation

Additional label analytics for the datasets in their standalone versions are presented in figure 4.3.5.

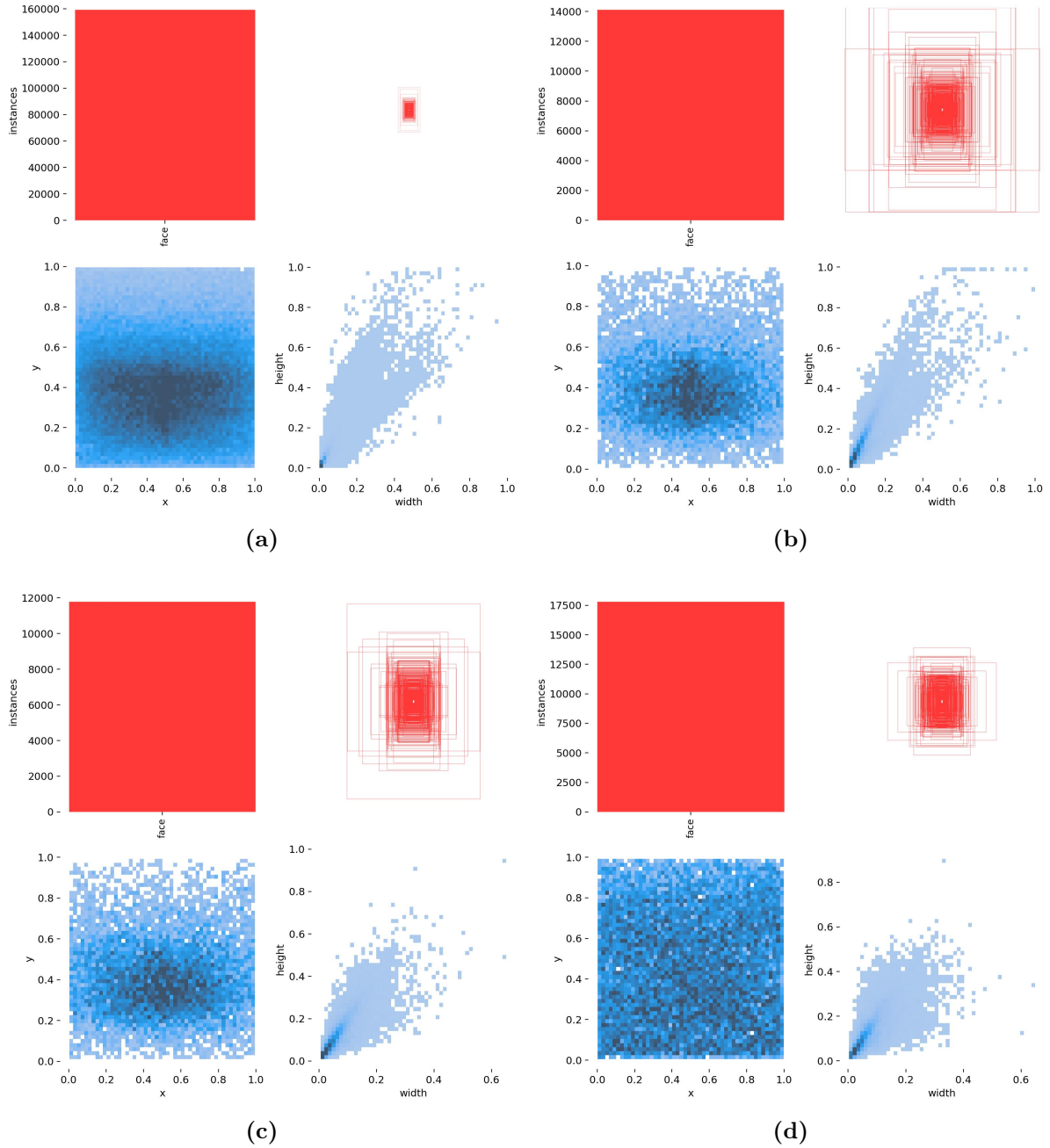


Figure 4.3.5: (a) Original widerface dataset (b) Extended standalone dataset without any additional pre-processing steps (c) Extended standalone dataset with additional pre-processing steps (d) Extended standalone dataset with mosaic augmentation

Table 4.3.4 shows a numeric summary of the new datasets combined with the widerface dataset. In addition, in table 4.3.5, the same numeric summary is shown for the standalone version of the new datasets. Lastly, a new metric, called instances per image is shown in both tables.

Dataset version	Images	Instances	Instances per image
Widerface	12879	159421	12.4
Widerface + raw extension	17879	173522	9.7
Widerface + filtered extension	15922	171187	10.8
Widerface + mosaic extension	15876	179919	11.3

Table 4.3.4: Number of images within each dataset variation as well as number of instances.

Dataset standalone version	Images	Instances	Instances per image
Widerface	12879	159421	12.4
Raw extension	5000	14101	2.8
Filtered extension	3043	11766	3.9
Mosaic extension	2875	20498	7.1

Table 4.3.5: Number of images within each standalone dataset variation as well as number of instances.

4.3.3 Anonymization filter

Detection results with anonymization filter implemented. The following result is generated by running the YOLOv8n variation of YOLOv8.

(a) **(b)**

(c)

Figure 4.3.6: (a) Original image (b) Anonymization with blur filter (c) Anonymization with blur and noise filter

CHAPTER

FIVE

DISCUSSION

5.1 Inference results, usable for real-time implementation?

The inference results shown in section 4.1 present the real-time capabilities of each of the systems. The initial inference results for the input resolution 640×480 pixels show great performance for systems such as the RTX 2080 that has access to state of the art processing unit as well as more memory compared to the other systems shown in the figure 4.1.1. The older GPU, GTX 1050 achieves the second-best performance but substantially below the performance of the RTX 2080.

For the embedded devices, the older, traditional jetson nano performs way below a level that would prove useful in these scenarios. Another interesting observation is the performance of the CPU, which traditionally has not been favored for running object detection models due to their way of dealing with computations in sequence. Nevertheless, the results show modern CPU possess enough power to run models for real-time purposes but are not favored compared to their more efficient GPU within their own computers. Therefore, the practical use-case for running CPU models is irrelevant when considering a real-world implementation.

5.1.1 Embedded devices inference performance

For the jetson nano, the performance with input resolution 640×480 pixels is, as previously mentioned, not usable for real-time implementation. By decreasing the input resolution to 416×312 pixels, the amount of pixels passed as input to the models is reduced 42%. By running the same inference test we achieve a new set of results. In figure 4.1.2, it is seen that the performance of the jetson nano increases substantially with an fps increase of 88%. In terms of a numeric value, this equals to a fps during runtime of 4.7, which still does not enable any general real-time implementations.

When considering the newer jetson orin, the performance on 640×480 pixels is within a limit that would be used for real-time implementation. However, the performance of 7.2 fps can also be argued to be outside the real-time requirements depending on the implementation domain. The results from running with decreased input size increases

the fps performance to 10.4 fps which turns out to be a 44% fps increase. The side-by-side comparison is shown in figure 4.1.3. In terms of implementation capabilities, a performance of 10.4 fps is substantial enough to be implemented in real-world scenarios, even for faster-moving domains. More promising is the future advancement in hardware, which can push the embedded performance higher, making embedded devices perform well above the limit for all real-time scenarios.

5.1.2 GPU and CPU inference performance

The results from the GPUs are the two highest fps results when comparing all the systems. In addition, the performance on the GPUs is interesting in showing how a system performs without much limitations. The RTX 2080 with 12GB memory performs substantially above and beyond anything that would be required for real-time performance. A noticeable difference is the improvement when decreasing the input size is less compared to the other systems, as seen in table 4.1.1. One hypothesis for this is due to the bottleneck in the computer not being the GPU but instead the CPU, which is responsible for feeding the input data to the GPU during runtime. This would support the less improvement observed since the limiting factor is the CPU rather than the GPU.

For the older GPU, the GTX 1050 has a much less impressive performance compared to the RTX 2080. In figure (a) 4.1.4, the performance for both input resolutions 640×480 and 416×312 is shown usable for real-time usage. Compared to the RTX2080 the inference increase is more substantial when decreasing the input size, with more than a doubling of the fps during runtime. Similarly, the performance on the two CPU units, the intel I5 and intel I9, both perform well enough for real-time applications. From figure 4.1.1 and figure 4.1.2, it can be seen that the performance of the intel I5 is comparable to the jetson orin while intel I9 is the third best-performing system overall.

5.1.2.1 Limiting memory in embedded devices

One possible limiting aspect of the real-time performance on the two embedded devices is the lack of available memory. Both computer systems provide the GPU and CPU with greater RAM memory capabilities, especially the GPU, which has additional specific memory for fast computation. This lack of memory in embedded devices results from the size limitations when developing embedded hardware.

5.2 Performance effect with decreased input size

The most important aspect of implementing a system for real-time anonymization purposes is the model's reliability. As mentioned in the previous section, the inference result is generated by running tests on the YOLOv8n model variation due to it being the smallest YOLOv8 model. However, YOLOv8n is also the worst-performing version of the YOLOv8 models because it is optimized for fast runtime. Therefore, properly analyzing the effect of decreasing the input size from the initial 640x480 down to 412x316 is crucial. Initial evaluation can be seen by examining the models precision, recall, and F1 score as presented in the figures 4.1.9, 4.1.10 and 4.1.11. The precision score goes from 84.9% with resolution 640×480 to 79.8% with resolution size 416×312. Similarly, the recall score goes from 58% down to 46.4%. In terms of percentage reduction, the precision score decreases 6%, and the recall score decreases 20%. In terms of F1 score, which considers both precision and recall, a reduction of around 14% is observed as an effect of decreasing the input resolution. One aspect that the result does not show is the complexity of the faces that are not detected and anonymized. By using the widerface, a

large amount of the images contained within the dataset do not exceed many pixels and would be counted as a very small object, making it difficult to detect and recognize even without applying any anonymization filter. But even in a scenario where faces are small enough that models struggle to identify the details of the face, a model unable to detect and anonymize those faces will struggle to be approved based on the possible regulation proposed by EU.

5.2.1 Privacy concern, maximizing recall performance

A common issue with deep learning models is the tradeoff between recall and precision, where to achieve a high recall, you need to sacrifice precision and vice versa. For face anonymization, the goal is to maximize the recall since we do not wish to let any faces go through our model not anonymized. The figure 4.1.10 showcases how the recall score changes depending on the confidence. Similarly, figure 4.1.9 showcases how the precision changes according to the confidence. In this situation, confidence means how sure the model needs to be before labeling instances in the data as a face. This means that a maximum recall score is achieved when the confidence is close to zero. Running models with confidence set to zero would result in many wrongly labeled instances but also with most of the total amount of faces detected. Since we are dealing with face anonymization, the consequences of not detecting a visible face can be massive, as it would make the model illegal based on the proposed AI legislation from EU. Therefore, a natural solution would be to run the model with a low confidence score, for example ($confidence = 0.15$), and accept some extra non-face instances rather than some extra non-anonymized faces, or in other words, accept a decrease in precision to increase the recall score.

5.3 Dataset initial benchmark

From the results presented in section 4.3.1 an interesting observation is the similar performance in terms of precision between the YOLOv8m and YOLOv8l models, with recall being the evaluation metric that separates them. One notable decision to be aware of is that the hyperparameters were tuned to maximize the performance on the YOLOv8n model. The same hyperparameters were used to train the larger models without additional tuning. This was done mainly due to the YOLOv8n being the model of choice for the earlier real-time tests. Other than that, the benchmark results are as expected, with the larger models outperforming the smaller models.

5.3.1 Benchmark compared to the other model variations

Section 4.3 presents the performance of models trained on the different variations of the widerface dataset. For a recap of the different dataset variations see table 4.3.4. These variations are then plotted against our benchmark models trained on the original widerface dataset. The new models are shown to perform similarly to our benchmark model when looking at the F1 score in figure 4.3.1. Looking at the presented precision score in figure 4.3.2 and recall score in figure 4.3.3, there is no clear view if any of our additional data give our models any sort of advantage compared to the benchmark models. So although the newly created datasets offer up to 12% additional instances, this does not give at first sight any additional benefit in performance.

5.3.2 Amount of instances and label sizes in the proposed datasets

In order to understand the effect of our additional data, additional evaluation is presented in figure 4.3.4. Presented are four metrics for our labels. To recap these were

- Number of instances
- Location of instance in the image
- Size of the instance
- Bounding box for instances

For a quick summary of the contents within the datasets, the largest, the mosaic version, contains 20498 additional instances compared to the original widerface, and the smallest, the filtered extension version, contains 14101 additional instances. Using the visual representation given in figure 4.3.4 we can quickly notice that the main difference between the original dataset and our three newly created datasets is the size of the label bounding boxes. In figure 4.3.4 a more clear and intense coloring represents the most commonly used label sizes. From figure (d) 4.3.4, it is also notable that our mosaic dataset does not contain the same number of large detections compared to the two other generated datasets, as expected when using mosaic augmentation. Although when compared to the label sizes within the original widerface (a) it is still noticeably larger. This can also be observed by manually inspecting the widerface dataset, which, as mentioned before, contains many small and complex instances. A possible hypothesis for the lack of improvement shown in section 4.3 can be ascribed to our evaluation method, which consists of utilizing the widerface validation data, which naturally would be most similar in label size and complexity to our benchmark model, which is not affected by the additional data introduced in the other dataset variations.

For a quick and extra evaluation, the same figure as described above is shown for the standalone versions of the datasets in figure 4.3.5. This figure makes it easier to see each newly created dataset's variation. One notable observation is that (b) and (c) from figure 4.3.5 lack instances located on the edges, with a majority located closer to the center. Additionally, the standalone mosaic dataset shown in (d) from figure 4.3.5 distributes the location of the instances all over the image.

5.3.3 Increased amount of instances but decreased complexity

To extend on the information shown in table 4.3.4, a metric of particular interest is the instances per image shown in the table's last column. This metric tries to show how the extended dataset versions consist of noticeably less complex scenes compared to the original widerface dataset. The without any additional pre-processing contains only 9.7 instances per image on average, down from 12.4 in the original widerface. This also presents the benefit of utilizing mosaic as it is the most similar to the original in label size and in terms of instances per image with a value of 11.3. Additional tests on additional dataset would help evaluate the general robustness of the model.

5.4 Effectiveness of the anonymization filter

From figure (a)(b)(c) 4.3.6 it is noticeable that the standard (b) blur effect does a good job of making it hard to identify the facial features of individuals. The benefit of utilizing the additional noise filter shown in figure (c) 4.3.6 is to avoid the static feature of the blur filter. The random noise makes the pixel values over the anonymized face change randomly for each frame. The goal of this is to make it harder for de-anonymization models to reverse engineer the facial features from the anonymized face. From figure 4.3.6 we also notice the effectiveness of the detection model when the image has good

contrast and good lighting condition. From figure 4.3.6 by visual inspection, it appears to have a perfect score in face detection and anonymization. The image used to display the anonymization methods would also represent a typical input for a surveillance camera deployed around cities. It also showcases that by ensuring that the location of cameras is chosen to provide the best detection conditions, the general performance presented in earlier sections can be improved. Another possible solution to avoid the worse performance in bad conditions would be to only run the models while the condition is above a certain quality limit. In other words, have an additional model evaluate the condition for running detection models.

CHAPTER

SIX

CONCLUSIONS

The work presented in the thesis hopes to showcase the current progress of SOTA object detection models and to show how these models can be modified for anonymization purposes and how various systems perform in terms of their real-time capabilities. Finally, multiple additional face datasets are presented as an extension of the widerface dataset.

The results present how these models run on a wide selection of systems, including two embedded devices showing the possibilities for real-world implementation. The applications of systems like these would be for data collection for researchers and data collection and usage for the management of smart cities. The presented results show how embedded systems achieve a performance of 10 fps when using an input size of 416x312 pixels, and are at a point where the possibility for deployment in the real-world for real-time purposes is possible. In addition, the presented results showcase that modern CPU can run real-time detection models, which used to be only possible for more computationally efficient devices such as GPU.

Multiple extensions of the popular and widely used widerface were created for the second part of the thesis. The challenge of improving a high-quality dataset is well presented in the results section. Although no apparent improvement in model performance was observed, a possible reason for this can be attributed to the dataset used for evaluating performance, which was the validation data of the original widerface. Nevertheless, the conclusion is that the presented results showcase no definite improvement with the different variations dataset but also no definite decrease in performance. Among the newly created datasets, the version utilizing data augmentation contains the most complex scenes when evaluated based on label size and instances per image. Still, there is a possibility that the additional data, although not improving the model performance, increases the model's robustness by utilizing larger label annotation, which is a lacking area within the original widerface. The amount of complex scenes is a unique characteristic of the widerface that makes it increasingly complex to improve on.

For a final summary, the presented thesis shows promising results for the smallest version of SOTA object detection model with a simple modification for anonymizing detected instances. It also presents the current inference during runtime on multiple systems. In addition to the points mentioned above, three additional datasets have been presented

with an evaluation of their performance effect. The datasets can be used as unique datasets for training or for extending other existing datasets.

6.1 Future work

For future work, one area for additional research is the real-time aspect. Additional tests should be performed running the models for a longer duration and how to integrate privacy concerning real-time systems effectively and securely into smart city domains, the last part is particularly interesting as more AI legislation bills will arrive in the future

Another area for future work is the additional evaluation of our new datasets. It is necessary to evaluate our generated dataset on other available datasets, to better validate any possible improvement through improved performance or robustness in our models. For summary, the following points are suggested for future work:

- Real-time test over longer period of times.
- Real-world prototype, and how to integrate privacy concerning systems into smart cities.
- Additional evaluation of the quality of the datasets, in terms of possible increased model robustness.

REFERENCES

[1] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

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

[3] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[4] Wei Liu et al. "Ssd: Single shot multibox detector". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer. 2016, pp. 21–37.

[5] Glenn Jocher et al. *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements*. Version v3.1. Oct. 2020. DOI: 10.5281/zenodo.4154370. URL: https://doi.org/10.5281/zenodo.4154370.

[6] Bichen Wu et al. *SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving*. 2019. arXiv: 1612.01051 [cs.CV].

[7] Zheng Ge et al. *YOLOX: Exceeding YOLO Series in 2021*. 2021. arXiv: 2107.08430 [cs.CV].

[8] Zhi Tian et al. "Fcos: A simple and strong anchor-free object detector". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.4 (2020), pp. 1922–1933.

[9] Shifeng Zhang et al. "Bridging the Gap Between Anchor-Based and Anchor-Free Detection via Adaptive Training Sample Selection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[10] Luis Perez and Jason Wang. "The Effectiveness of Data Augmentation in Image Classification using Deep Learning". In: *CoRR* abs/1712.04621 (2017). arXiv: 1712.04621. URL: http://arxiv.org/abs/1712.04621.

[11] Zhenfei Chen et al. "Privacy preservation for image data: a gan-based method". In: *International Journal of Intelligent Systems* 36.4 (2021), pp. 1668–1685.

[12] Junwu Zhang, Mang Ye, and Yao Yang. "Learnable Privacy-Preserving Anonymization for Pedestrian Images". In: *Proceedings of the 30th ACM International Conference on Multimedia*. 2022, pp. 7300–7308.

[13] Muhammad Umair. Hassan, Stava Magnus, and A. Hameed. Ibrahim. "Deep privacy based face anonymization for smart city applications". In.

[14] Gary B. Huang et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst, Oct. 2007.

[15] Vidit Jain and Erik Learned-Miller. *FDDB: A Benchmark for Face Detection in Unconstrained Settings*. Tech. rep. UM-CS-2010-009. University of Massachusetts, Amherst, 2010.

[16] Bin Yang et al. "Fine-grained Evaluation on Face Detection in the Wild". In: *Automatic Face and Gesture Recognition (FG), 11th IEEE International Conference on*. IEEE. 2015.

[17] Shuo Yang et al. "WIDER FACE: A Face Detection Benchmark". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[18] EUROPEAN COMMISSION. *LAYING DOWN HARMONISED RULES ON ARTIFICIAL INTELLIGENCE (ARTIFICIAL INTELLIGENCE ACT) AND AMENDING CERTAIN UNION LEGISLATIVE ACTS*. `https://artificialintelligenceact.eu/the-act/`. 2022.

[19] T. M. Mitchell. *Machine Learning*. McGraw Hill, Mar. 1997, p. 2.

[20] Norvig Stuart Russel Peter. *Artificial Intelligence. A Modern Approach*. Pearson, 2022.

[21] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: `1506.02640 [cs.CV]`.

[22] Chien-Yao Wang et al. "CSPNet: A new backbone that can enhance learning capability of CNN". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 390–391.

[23] BEN LE datature. *Get Started with Training a YOLOv8 Object Detection Model*. 2023. URL: `https://www.datature.io/blog/get-started-with-training-a-yolov8-object-detection-model` (visited on 05/20/2023).

[24] Yue Wu et al. "Rethinking Classification and Localization for Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[25] Matthias Feurer and Frank Hutter. "Hyperparameter optimization". In: *Automated machine learning: Methods, systems, challenges* (2019), pp. 3–33.

[26] Li Yang and Abdallah Shami. "On hyperparameter optimization of machine learning algorithms: Theory and practice". In: *Neurocomputing* 415 (2020), pp. 295–316.

[27] S. Bos and E. Chug. "Using weight decay to optimize the generalization ability of a perceptron". In: *Proceedings of International Conference on Neural Networks (ICNN'96)*. Vol. 1. 1996, 241–246 vol.1. DOI: `10.1109/ICNN.1996.548898`.

[28] Nishant Vishwamitra et al. "Blur vs. block: Investigating the effectiveness of privacy-enhancing obfuscation for images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 39–47.

[29] Wang Hao and Song Zhili. "Improved mosaic: algorithms for more complex images". In: *Journal of Physics: Conference Series*. Vol. 1684. 1. IOP Publishing. 2020, p. 012094.

[30] Open Images V7Open Images V7. *Open Images Dataset V7 and Extensions*. data retrieved from Open Images V7, `https://storage.googleapis.com/openimages/web/index.html`. 2022.

[31] Delong Qi et al. "YOLO5Face: Why reinventing a face detector". In: *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*. Springer. 2023, pp. 228–244.

APPENDICES

A - GITHUB REPOSITORY

All code and link to the datasets used in this document are included in the Github repository linked below. Further explanations are given in the readme-file.

Github repository link

- `https://github.com/Magnsta/SOTA-comparison-study-`

B - PRE-PROJECT REVISIONED REPORT

Deep privacy based face anonymization for smart city applications

Muhammad Umair Hassan*, Magnus Stava, Ibrahim A. Hameed

Department of ICT and Natural Sciences, Norwegian University of Science and Technology, Ålesund, Norway

Abstract

Amounts of images containing identifiable features have multiplied dramatically in an increasingly digital world where data is gathered on a large scale through smart city surveillance systems and smartphones. In order to protect privacy, it is essential to look into methods that can anonymize individuals in real-time before the data is stored in databases. This work proposes to anonymize the real-time data collected through different data collection devices and anonymize it before being stored in databases leveraging YOLO-based face detection. We evaluate the performance of our proposed methodology on the *WIDER-FACE* benchmark with the included easy, medium, and complex subsets.

Keywords: Face anonymization, Face detection, Deep learning, Privacy protection

1. Introduction

The development of information and communication technologies (ICT) and the expansion of personal private data has brought a great utility value in individuals' lives and society. However, serious privacy concerns are also becoming prominent, and there is a severe demand to preserve people's privacy in collected data through video cameras. Face detection is the ability of a computer program to detect faces in digital images. It is the first step before other operations, such as face identification [1] and face de-identification [2], can occur. Face identification is the process of identifying a specific individual from an image, while face anonymization removes identifiable features from an image. Specifically, face identification has created a concern for our right to privacy due to a large number of digital images available in the modern world [3]. Traditional object detectors struggle to operate in real-time due to using a two-stage architecture. However, more contemporary state-of-art models designed with a single-stage model architecture have increased computational efficiency making them more usable for real-world implementation.

Smart cities rely on unprecedented amounts of data. The construction of smart cities have brought a higher quality of life as well as the threats to the people's privacy have also been raised [4]. The protection of privacy got an increased interest after the European Union implemented the General Data Protection Regulation (GDPR). Hiding the identity of individuals has been common practice to reduce exposure to individuals put in an exposed position. This could be individuals part of a criminal investigation, interviewed by the news, or data collection for research purposes, to mention some. The standard technique is to manually apply a blur filter which reduces the information within the filter by averaging the pixels by using a kernel window moving across the selected area. Applying blur will make the facial features in the image become less recognizable by decreasing the detail. Other filters are also often used, but they follow the same technique as blurring, with the difference being the values in the kernel.

Following the advances in deep learning (DL), researchers have been interested in methods for anonymizing individuals without losing any original data information. In contrast to blurring, which reduces information in the selected area. A group of researchers has utilized generative adversarial networks (GANs) to anonymize identifiable features while maintaining the complete surrounding information. They achieved this by training the GAN to perform image inpainting on images with identifiable features anonymized, only passing in knowledge about the location of facial features such as eyes, nose, and mouth [2]. Other researchers, such as Nousi et al. [5], explored the usage of autoencoders to decompress the facial information to a

*Corresponding author

Email address: muhammad.u.hassan@ntnu.no (Muhammad Umair Hassan)

low-dimensional representation and then reconstruct a face that is noticeably different compared to the original [5]. Both methods have showcased the possibility of anonymizing individuals while preserving a face's visual attributes and the image's information. Compared to traditional techniques such as blurring or pixelation, deep learning-based methods are more computationally heavy and less usable for real-time implementation. Another approach for anonymizing faces is to combine traditional methods, such as blurring, with state-of-the-art performance face detectors. This method is much less computationally expensive, enabling real-time operations.

In this work, we leverage a single-stage face detection network to detect faces in complex scenes and look into what separates the single-stage networks and techniques for effectively anonymizing faces. We made the following contributions to this paper. (i) We propose a novel technique to preserve the identity of persons coming in the images/videos without destroying the primary objective of data collection. (ii) We compare implementations of a selection of face detection algorithms on a standard benchmark. (iii) We explore and compare different techniques for face anonymization. (iv) We validate the results through quantitative and qualitative evaluations.

The organization of this paper is as follows. Section 2 presents the works related to this study, while in Section 3, we discuss our proposed methodology. The performance evaluation of this work is discussed in Section 4. Finally, we conclude this work in Section 5.

2. Related Works

Yang et al. [6] proposed a face mask anonymization method in which the Putting off Mask Face module is designed to remove the mask from a person's face, allowing the protected face to be recovered. Wen et al. [7] proposed a framework called IdentityDP that utilizes deep privacy-based perturbation directly on the identity representation to ensure privacy protection while preserving the visual similarity of the attribute representation without any changes. Zhai et al. [8] proposed the A3GAN anonymization method, which uses facial attribute manipulations instead of ultimately generating a new face. To achieve this, they introduced a new generative network architecture that includes a suppressive convolutional unit (SCU) and an attribute-aware injective network (AINet). Qiu et al. [9] developed a novel framework named QM-VAE. This approach first de-identifies the facial image before reconstructing its utility. To enhance the quality of the generated face images, they integrated vector quantization into the structure of the generative model. Cheng et al. [10] proposed methods for de-identifying 2D and 3D faces while still preserving facial attributes. These methods use Auto Encoder and GANs. Using these techniques, images can be de-identified while maintaining the integrity of facial features. Kim et al. [11] presented a technique to safeguard the identification of a person captured by the CCTV IoT system by considering the risk level. By adjusting the information disclosure level based on the individual's risk level, the method can prevent the indiscriminate disclosure of private information and minimize privacy breaches. Ding et al. [12] proposed a technique for anonymizing face recognition data, where only the authorized model and human eyes can recognize the face images. The study defines constraints for boundary walking strategy. The paper also presents an attack on the FACE++ API to confirm the effectiveness of the proposed method in real-world scenarios.

3. Methodological Description

3.1. Face detection

Face detection as an object detection task is challenging due to the variety of sizes faces can appear in images, making it necessary for the models to effectively detect tiny, small, and large faces. Detecting tiny objects is more complicated due to fewer pixels being responsible for the identifiable features that make up a face. The fact that small objects are represented by fewer pixels is critical to understand to develop good performing models for tiny objects since primary convolution or pooling layers can make the crucial features lost [13][14]. We present the framework of our proposed work in Fig. 1.

3.2. You only look once

You Only Look Once (YOLO) algorithm, first published in 2015, is a single-stage algorithm that manages to achieve real-time object detection [15]. In 2021 the **YOLOv5-Face** was published, a specific-purpose object detector designed for face detection [16]. The research team behind YOLOv5-Face decided to modify the existing state-of-art detector YOLOv5 for their specific purpose instead then designing their model, with the argument that a face is just another object with many of the same features as other objects such as pose, scale, occlusion, etc. The main contribution of the YOLOv5-Face can be summarized as follows. (i) Inclusion of a STEM block. (ii) Reducing the kernel size in strategic locations. (iii) Additional output block with a large

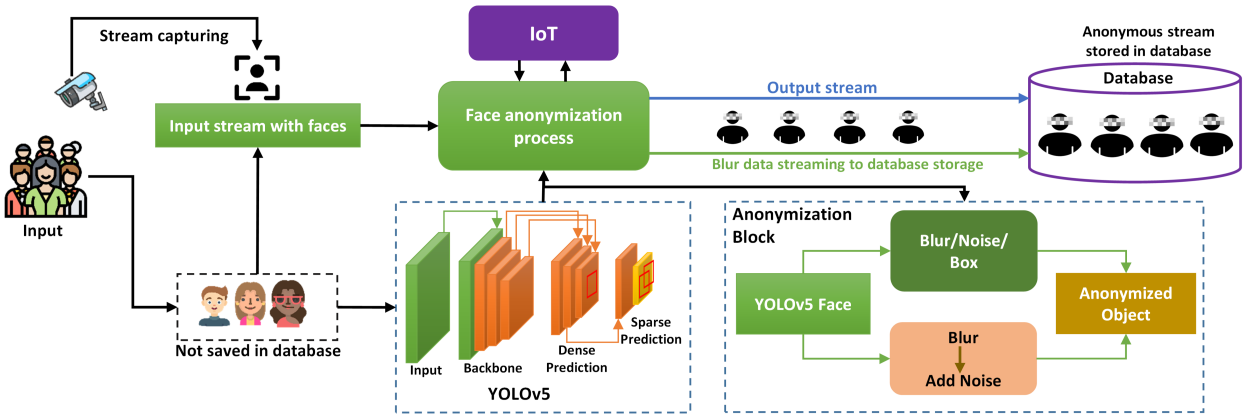


Figure 1. Our proposed framework for face anonymization. The input data is captured through different data collection devices, and YOLOv5-based facial detection is applied to detect all recognizable faces from the streams. Blurring filters are applied before the data is sent for storage. For YOLOv5, we apply skip connections-based methodology; the previous layer's output is the input to Conv layer 1 while also skipping through and being sum together with the output from Conv layer 2. The flow of model happens in anonymization block and the anonymization happens after the object detector.

stride value of 64. (iv) Discovery of worse performance on small faces when specific augmentation techniques such as flipping and mosaic are used [16].

Some key modifications in YOLOv5 for face detection are landmark regression head with replacement of focus layer, block for SSP with intervention of a kernel [16]. We also used P6 block with stride of 64 with enhanced methods for face detection augmentation methods with two new designs of ShuffleNetv2. For baseline of face detection, we used YOLOv5 detector. This includes neck, backbone and head. An SSP and PAN are involved in this with integration of classification and regression in the head. Landmarks on human face are used for alignment and recognition and mostly these points are 68 are simplified to five main points. But the general object detector includes them as regression head. These outputs will be used in alignments before sending to network of face recognition. For defining the small errors detection wing-loss was developed because general loss function was not able to it.

$$\text{wing}(x) = \begin{cases} w \cdot \ln(1 + |x|/e), & \text{if } x < w \\ |x| - C, & \text{otherwise} \end{cases} \quad (1)$$

where X is the signal of error where nonlinear part range $(-w, w)$ is set by non-negative w. e belongs to the curvature whereas $C = wwln(1 + w/e)$ remains constant which connects the linear/nonlinear segments.

In YOLOv5, the loss function for detecting objects includes information about the bounding box, class, and probability of the object. Additionally, there is a separate loss function for landmark regression, represented as $loss_L$, which measures the discrepancy between the predicted and true landmark points. This loss is calculated for each of the 10 landmark points using a specific formula as following.

$$\text{loss}_L(s) = \sum_i \text{wing}\,(s_i - s_i') \quad (2)$$

where $s = s_i$, and $i = 1, 2, ..., 10$. While discussing the small kernel SSP the output feature maps of the backbone are directed to an SPP (Spatial Pyramid Pooling) block before being passed to the feature aggregation block in the neck. The purpose of the SPP block is to increase the receptive field and isolate the most significant features. This is achieved without the use of fully connected layers that are specific to certain input image dimensions. SPP generates a fixed-size output, regardless of the input size, making it a more flexible and versatile approach compared to traditional CNN models. Different size of kernel like 7×7, 5×5 and 3×3 are used to detect small faces.

The spatial resolution for YOLO layers decrease due to multiple down layers for detect objection. FPN was developed in YOLOv3 for small grained characteristics. But this take a long path for crossing multiple layers so PAN was introduced as bottom up approach. In FPN, object predictions are made separately on different scales, which may lead to redundant predictions. In contrast, the PAN model fuses the output feature maps of the bottom-up augmentation. Multiple output blocks P3, P4, P5 were introduced in YOLOv5. But in this YOLO5Face a P6 block was also added with feature map $10 \times 10 \times 16$ with 64 stride.

3.3. Face anonymization

The task of anonymizing faces can be done in multiple ways. The simplest and most basic method is to apply a solid box over the predicted bounding box. Given a predicted bounding box ($\mathbf{m} \times \mathbf{n} \times \mathbf{p}$) where $\mathbf{p}$ is the pixel value, the anonymized area becomes ($\mathbf{m} \times \mathbf{n} \times$ **(50, 50, 50)**). This ensures a complete anonymization of the individual in the frame, assuming the model has a high performance. A disadvantage of using this technique is that it completely removes all the information contained within the bounding box, making it not very practical if the image should be utilized later.

To avoid the complete loss of information within the bounding box, another technique is to apply Gaussian noise $N(0, 0.5)$. The Gaussian noise slide the pixel values towards the normal distribution decreasing detail. Blurring also ensures that more of the original information in the frame is retained. An issue with blurring techniques is the possibility of reversing the effect if the strength of the blur is not strong enough.

A third technique is applying a layer of noise to the area within the bounding box. This technique would cover the facial features on a level that falls between the box and blur technique. The main advantage with applying noise to the bounding box region is to make it hard for bad players to use the image for any bad purposes.

The implementation of the anonymization step would be implemented as an additional step after the fully connected layer. In terms of altering of code this would be implemented in the same step as the drawing of the bounding box, but before the image is presented to the user to ensure full privacy. In Fig. 1 the full YOLOv5-Face architecture is represented as one block, with the output going into the anonymization block where one of the three techniques is applied. For more detail of the YOLOv5-Face architecture we refer to [16].

Figure 1 shows the YOLOv5 architecture, which consists of a backbone (CSP DarkNet53), a neck (Dense Prediction), and a head (Sparse Prediction). The backbone extracts features from the input image, the neck combines those features at different scales, and the head predicts bounding boxes, objectness scores, and class probabilities for each anchor at each scale. The architecture is designed to be efficient and accurate, and has achieved state-of-the-art performance on several object detection benchmarks.

4. Performance Evaluation

In order to have an effective way of comparing the performance of the different models, a standardized benchmark is advantageous. One of the main benchmarks used for face detection is the *WIDER-FACE* dataset [17]. The *WIDER-FACE* benchmark provides three subsets for evaluating the models, easy, medium, and hard. Images in the hard category typically contain faces that are smaller, partly obstructed, or are presented in conditions that make it challenging for models and even humans to point out, an example of a hard photo can be seen in Fig. 2. To ensure equal test conditions one system has been used throughout the proposed work. The system boosts an high-performing GPU and CPU enabling for faster runtimes compared to a standard personal computer have been used. The exact specification of the computer can be seen in Table 1.

Figure 2. Example of a complex image showcasing a challenging scenario with many small faces for the model to detect.

Table 1
Specifications for the alienware computer and the embedded device jetson nano.

Machine	**GPU**
Alienware x15 R2	NVIDIA GeForce RTX 3060 16 GB
Jetson Nano	128-core Maxwell 4 GB

In Table 2 the performance of two of the models presented by Qi et al. [16] is shown, the *YOLOv5s6* represent their lightest and fastest model. For comparison their best performing model the *YOLOv5x6* can also be viewed. From this it is noticeable the the difference becomes more apparent when the input image is belonging to the category of hard images.

Table 2
Yolov5-Face WIDERFACE benchmark result from large and small model.

Difficulty	**Easy**	**Medium**	**Hard**	**Params(M)**	**Flops(G)**
YOLOv5x6	96.67	95.08	86.55	141.158	88.665
YOLOv5s6	95.48	93.66	82.8	7.075	5.751

All presented results have been obtained by running them on YOLOv5-Face algorithm on the corresponding

Figure 3. Three different techniques for anonymizing faces that ensures fast performance. To the left the face is anonymized by a solid box. In the middle a blur filter is applied. To the right noise has been applied.

system as mentioned Table 1. From Fig. 3 we can observe the three standard techniques. Anonymization by using a solid box is as expected very effective and makes it impossible for any sort of reconstruction that would appear similar to the original image. On the downside it is not visual pleasing to look at and does affect the image to a very large extent. In the middle of Fig. 3 a blur effect has been applied and the amount of information retained is much greater then when compared to the solid box. But it is also clear looking at the blur effect in Fig. 3 that a reconstruction to obtain the original image is more likely to be successful compared to the solid box method. Finally, to the right in Fig. 3 random noise has been added to the bounding box, and although it appears very similar to the solid box, it is possible to notice slightly where some of the facial features would appear such as eyes and mouth, and the outline of the face. It can be assumed to be harder for bad players to exploit if substantial amount of noise is added to the image.

Figure 4 shows an input of a complex scene with multiple small faces that are detected, and noise filters are applied to them, and finally, a blurring effect is applied to hide the identities of detected faces in the complex scene. For the fourth technique, the same amount of blurring and noise is applied as previously, and the only difference is the combination of the methods. From Fig. 5 it is noticeable that the facial features have become much less visible with the combination of the two methods. The information that is retained in the image is subsequently less than just blurring. The output image is more pleasant to observe compared to the solid box and random noise technique, as seen in Fig. 3. We show the F1 curve and precision-recall curves in Fig. 6a and 6b, respectively. The frames per second was measured for the YOLOv5-Face model as well as the newly released YOLOv7. The performance on Alienware x15 R2 is shown in Table 3. The source for the test is the attached computer webcam, and the FPS is calculated while running by counting number of frames the model has processed. To ensure that our privacy methods have not affected the speed of out model we also measure the FPS of the models without any anonymization techniques. From out experiments there were no visible difference on running the models with or without anonymization. Considering how simple calculations these anonymization are this is not surprising but validates the possibility to run face detectors in real-time with the additional anonymization option.

5. Conclusion

In conclusion, this work proposes a YOLO-based face detection methodology for anonymizing real-time data collected through different devices before storing it in databases. We evaluate the proposed technique on the *WIDER-FACE* benchmark and compare various face detection algorithms, exploring and comparing different techniques for face anonymization. Our work proposes a novel technique to preserve the identity of persons in the images/videos without compromising the primary objective of data collection. We also investigate single-stage face detection networks and techniques for effectively anonymizing faces. Our findings provide valuable insights into the current state-of-the-art detector functions and how these innovations can be transferred to other models. We have also discussed the importance of models working with privacy to function with high performance over a longer period. Overall, this work contributes to enhancing privacy protection in an increasingly digital world where data is gathered on a large scale through smart city surveillance systems and smartphones.

Figure 5. The fourth technique, combining blur and noise as compared to only blurring

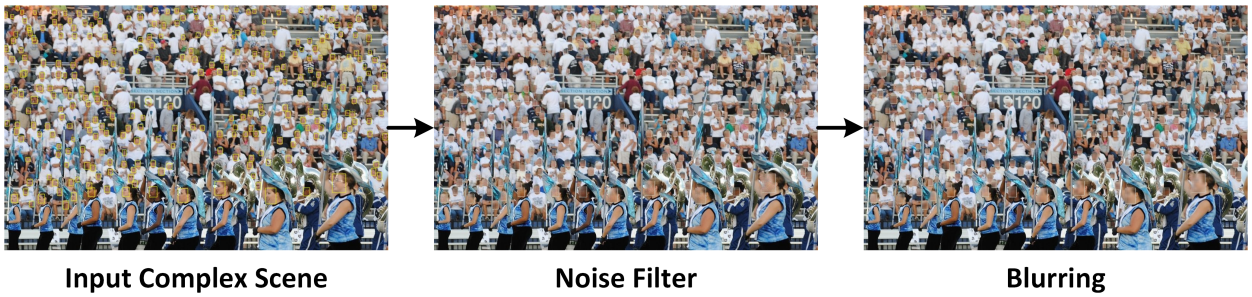
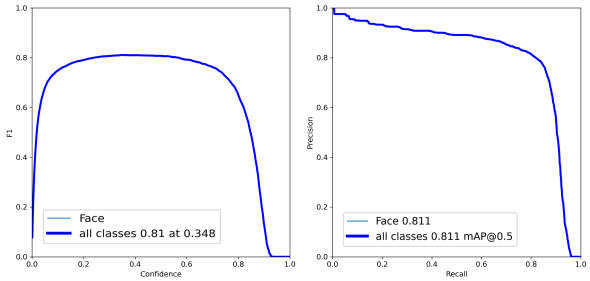


Figure 4. An example of complex scene face detection and applying noise filters and blurring effects to hide the identity of people coming in data collection devices.

Table 3
The frames per second for the YOLOv5-Face and YOLOv7-tiny with anonymization.

Models	FPS
Yolov5-Face	20.1
Yolov7-tiny (custom trained)	24.2



(a) F1 Curve (b) PR Curve

Figure 6. F1 and PR curves resulted by our proposed technique.

References

[1] R. Ranjan, A. Bansal, J. Zheng, H. Xu, J. Gleason, B. Lu, A. Nanduri, J.-C. Chen, C. D. Castillo, R. Chellappa, A fast and accurate system for face detection, identification, and verification, IEEE Transactions on Biometrics, Behavior, and Identity Science 1 (2) (2019) 82–96. doi:10.1109/TBIOM.2019.2908436.

[2] H. Hukkelås, R. Mester, F. Lindseth, Deepprivacy: A generative adversarial network for face anonymization, in: International symposium on visual computing, Springer, 2019, pp. 565–578.

[3] S. Ribaric, A. Ariyaeeinia, N. Pavesic, De-identification for privacy protection in multimedia content: A survey, Signal Processing: Image Communication 47 (2016) 131–151.

[4] J. Laufs, H. Borrion, B. Bradford, Security and the smart city: A systematic review, Sustainable cities and society 55 (2020) 102023.

[5] P. Nousi, S. Papadopoulos, A. Tefas, I. Pitas, Deep autoencoders for attribute preserving face de-identification, Signal Processing: Image Communication 81 (2020) 115699.

[6] Y. Yang, Y. Huang, M. Shi, K. Chen, W. Zhang, Invertible mask network for face privacy preservation, Information Sciences 629 (2023) 566–579.

[7] Y. Wen, B. Liu, M. Ding, R. Xie, L. Song, Identitydp: Differential private identification protection for face images, Neurocomputing 501 (2022) 197–211.

[8] L. Zhai, Q. Guo, X. Xie, L. Ma, Y. E. Wang, Y. Liu, A3gan: Attribute-aware anonymization networks for face de-identification, in: Proceedings of the 30th ACM International Conference on Multimedia, 2022, pp. 5303–5313.

[9] Y. Qiu, Z. Niu, B. Song, T. Ma, A. Al-Dhelaan, M. Al-Dhelaan, A novel generative model for face privacy protection in video surveillance with utility maintenance, Applied Sciences 12 (14) (2022) 6962.

[10] K. H. Cheng, Z. Yu, H. Chen, G. Zhao, Benchmarking 3d face de-identification with preserving facial attributes, in: 2022 IEEE International Conference on Image Processing (ICIP), IEEE, 2022, pp. 656–660.

[11] J. Kim, N. Park, De-identification mechanism of user data in video systems according to risk level for preventing leakage of personal healthcare information, Sensors 22 (7) (2022) 2589.

[12] K. Ding, T. Hu, X. Liu, W. Niu, Y. Wang, X. Zhang, Targeted anonymization: A face image anonymization method for unauthorized models, in: 2022 IEEE International Conference on Multimedia and Expo (ICME), IEEE, 2022, pp. 1–6.

[13] C. Xianbao, Q. Guihua, J. Yu, Z. Zhaomin, An improved small object detection method based on yolo v3, Pattern Analysis and Applications 24 (3) (2021) 1347–1355.

[14] Y. Liu, P. Sun, N. Wergeles, Y. Shang, A survey and performance evaluation of deep learning methods for small object detection, Expert Systems with Applications 172 (2021) 114602.

[15] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

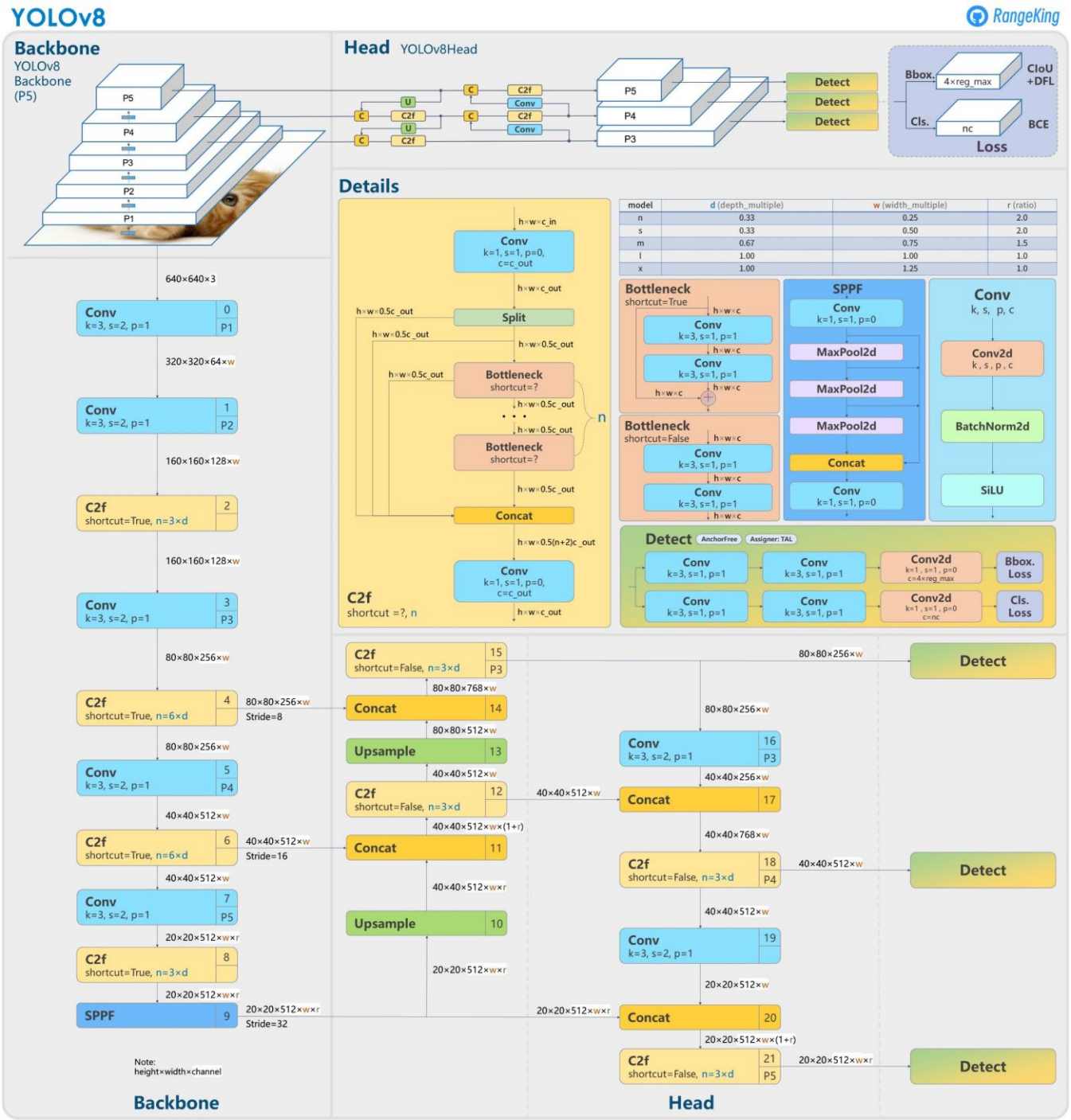
[16] D. Qi, W. Tan, Q. Yao, J. Liu, Yolo5face: why reinventing a face detector, arXiv preprint arXiv:2105.12931.

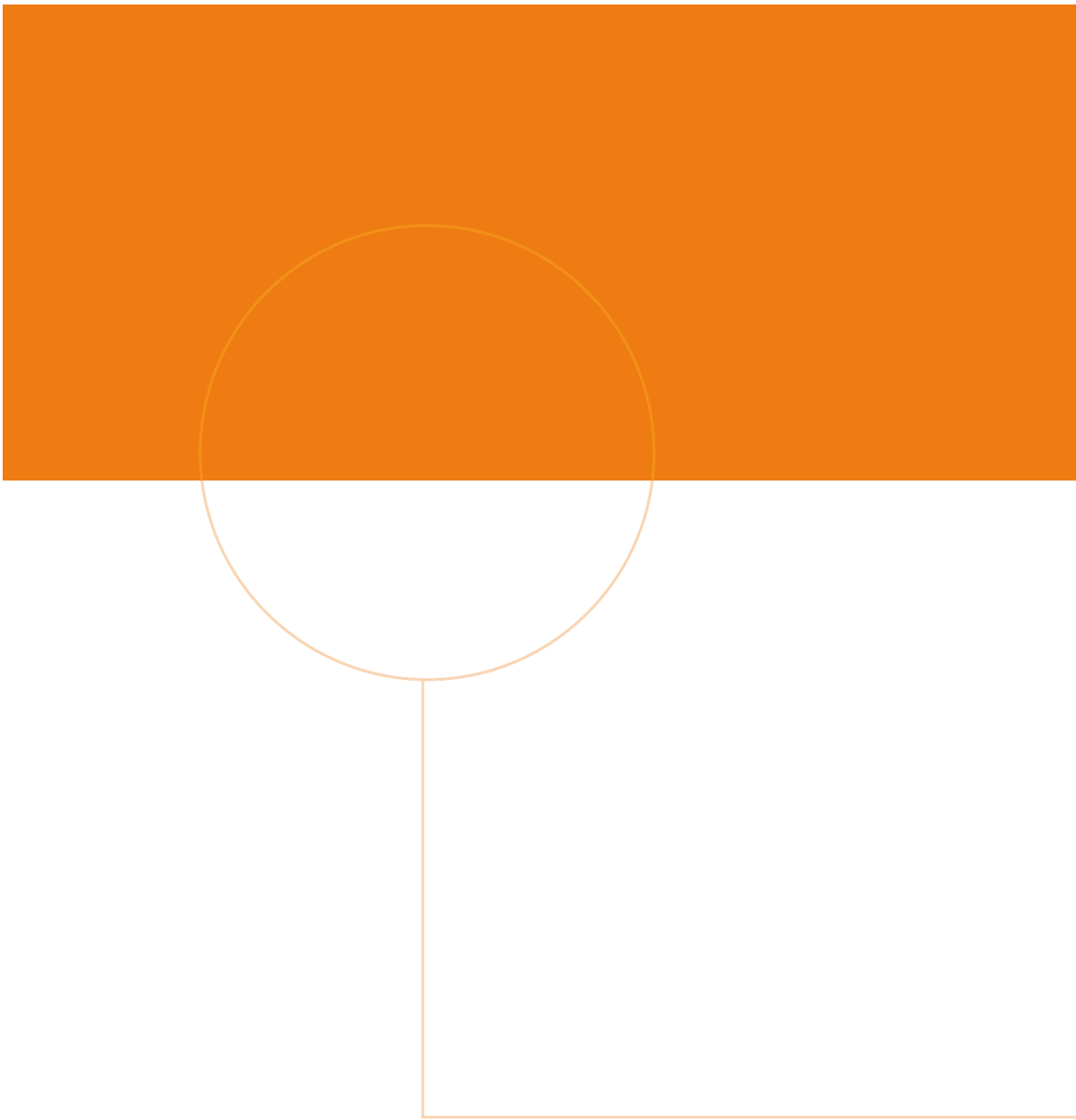
[17] S. Yang, P. Luo, C. C. Loy, X. Tang, Wider face: A face detection benchmark, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

C - YOLOV8 MODEL ARCHITECTURE

A complete overview of the YOLOv8 architecture.

YoloV8 Architecture





NTNU

Norwegian University of
Science and Technology