Face-masked Face Blurring App

Ming Jun Lim

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Inmaculada Arnedillo-Sánchez

August 2022

Declaration

Permission to Lend and/or Copy

Face-masked Face Blurring App

Ming Jun Lim, Master of Science in Computer Science
University of Dublin, Trinity College, 2022

Supervisor: Inmaculada Arnedillo-Sánchez

Face-masked face blurring application is a tool to protect participants' privacy by anonymising faces in an image or video when creating a new dataset. Deep learning and neural network models require a vast training dataset for training. However, a dataset containing sensitive data must be anonymised before releasing it to the public to avoid legal issues. The dissertation focuses on developing a solution to blur face-masked faces in a massive set of video footage containing children wearing face masks performing various activities. The proposed design is a three-stage process involving face detection, facial landmark localisation and blurring. The experiments involve comparing feature extractors such as ResNet50, VGG16, and MobileNetV3 for the Faster R-CNN to detect face-masked faces, training loss function for ResNet18 CNN facial landmark localisation and different Gaussian kernel sizes of blurring. The models were trained using a public JD-landmark-mask dataset containing face-masked faces, bounding box and landmarks annotation. The approach uses transfer learning principles to train the models on a different dataset and is used on the new children's dataset. The ResNet50 Faster R-CNN model with custom Region Proposal Head (RPN) hyper-parameters trained with 30 epochs were selected with a mAP@Iou[0.5:0.9] score of 0.492. The ResNet18 CNN model was able to detect facial features with the presence of face masks with an L2 loss of 12.0285 after 100 epochs. The feature-based blurring obfuscated the faces while maintaining data utility of the face region, enabling future models to detect face and facial features. The results are promising, and the modular design enables switching each stage with different techniques to fit new applications.

Acknowledgments

Thank you, Dr Inmaculada Arnedillo-Sanchez and Dr Benoit Bossavit, for the weekly meetings with the feedback and tips to complete this dissertation work. It was a fantastic experience to work with a new dataset and create solutions for the problem with deep learning models relating to my course.

Thank you, Mum and Dad, for the unconditional support financially and emotionally to complete the course at Trinity College Dublin. They have provided me with huge freedom to explore many of my interests. Thank you to my friends for the support and cheers. They have made my journey in the Master's course very fulfilling!

Ming Jun Lim

University of Dublin, Trinity College
August 2022

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, the development in the computer vision field has been widely studied, improved and applied throughout our daily lives. Many computer vision algorithms involve deep learning and neural networks because they can construct feature extractors from learning features without needing much human intervention. The availability of large-scale labelled image datasets such as ImageNet and Microsoft COCO has made these algorithms even more popular and applicable (Szeliski, 2010). The accuracy and efficiency of these models have outperformed humans in image recognition, and the performance is subjected to the quantity and quality of annotated training data. Many visual data can be collected due to technological advancements such as cameras, smartphones and internet of things (IoT) devices. However, there is a rise concerning the invasion of users' privacy with the ethical aspects of data collection and strict regulations of processing such personal data. These strict regulations are invented to ensure no privacy issues are introduced (Kühl et al., 2020)(Goldsteen et al., 2020). The breakthroughs in deep learning have created business opportunities for companies to offer Machine Learning as a Service (MLaaS) commercial platforms (Raynal et al., 2020). The MLaaS enables a client to leverage computational power and memory resources to train a model for a machine learning application by uploading their labelled dataset. As such, the privacy of the dataset is extended to the services' premises, and sensitive data may legally restrict this kind of trust.

Face blurring application is a means of protecting an individual's privacy by face detection and obfuscation techniques like blurring. Models trained with an anonymised dataset showed promising results with only a slight decrease in performance compared to training with an unobfuscated dataset (Wu et al., 2020). Human activity recognition and movement behavioural science studies require a dataset of entire body parts. Movement analysis on growing children can be used for early detection of various movement disorders, saving much time as traditional methods require doctors and physiotherapists to perform tests such as general movement assessment (GMA) (Khan et al., 2018). Public datasets of these studies revolving around children have incorporated face blurring in their design to protect the identity of participants (Al-Jubouri et al., 2020). The method of obfuscation and blurring has a trade-off between target utility task performance and associated privacy budgets (Raynal et al., 2020). Hence a face blurring application should blur the face while maintaining core references to the facial landmarks. It can combine face detection and facial landmark detection, such RetinaFace (Deng et al., 2020) approach or be broken down into individual components before blurring the detected face.

Previously, most models were trained with little to no face occlusion datasets. The Coronavirus Disease 2019 (COVID-19) that emerged in 2019 has caused a significant impact in many countries worldwide. COVID-19 is a virus from the coronaviruses family infecting the lower respiratory tract and causing severe acute respiratory syndrome coronavirus 2 (Sars-CoV-2) disease (McKay, 2020). The transmission of COVID-19 spreads through respiratory droplets in close contact with an infected person when he or she speaks, coughs or sneezes, according to the World Health Organization (WHO). The World Health Organization recommends face masks to be worn by the general public in guidelines for mitigating the risk and impact of epidemic and pandemic influenza to limit the spread of respiratory diseases, including COVID-19 (Missoni et al., 2021)(Timelli and Girardi, 2021).

The dissertation explores face detection methods with occlusions such as the usage of face masks and obfuscation techniques preserving high similarity of the original image content, keeping markers to core reference points such as ears, eyes and mouth. This section contains the project area of setting the scope of the dissertation, the motivation for developing the face-masked face blurring application; the aim and objectives; and the structure of the entire dissertation.

1.1 Project Area

Face recognition is detecting faces from an image, extracting features from the background and comparing features against a database of known faces, also known as gallery database (Knežević et al., 2018). The dissertation's focus will be mainly on capturing the face regions of the participants using face localisation techniques such as face detection, face alignment, pixel-wise face parsing and 3D dense correspondence regression (Deng et al., 2020). These techniques are used in single-stage face detectors and multi-stage face detectors. The face detector aims at finding the coordinates of the bounding boxes capturing the face region. Face alignment, also known as facial landmark localisation, aims to find the coordinates of key points such as the core reference points of the facial features (Feng et al., 2018).

Privacy-preserving deep learning relates to the privacy issue mentioned in the previous chapter (1) to train an accurate model while ensuring privacy against human eye perception and AI-based reconstruction attacks (Popescu et al., 2022). The scope of the privacy-preserving approach selected by the dissertation is image obfuscation. The underlying information of the sensitive data in the image is modified by image obfuscation techniques (Popescu et al., 2022).

1.2 Motivation

The dissertation involves massive video footage of children wearing face masks performing various activities such as galloping, jumping forward, hopping left, push up, etc. The camera is placed in front of the children from a single static point with different lighting conditions, environments and low-resolution videos. The video footage is recorded in an uncontrolled setting, unlike another children dataset containing the entire body movement of children in (Al-Jubouri et al., 2020) with a rigorous recording environment and only a single class activity. The dataset contains 16 types of classes with participants doing activities moving towards the camera, on the same point and sideways. The face blurring application has challenges such as face moving towards the camera, head out of frame, hand obstructing facial features, orientation and angle of the face and incorrect usage of the mask. In some footage, there are multiple children and adults in frames overlapping each other at different distances.

The video footage containing children is considered sensitive data requiring a face blurring application to be applied before releasing and using the dataset to train models. Obfuscation techniques such as blurring should have different intensities in facial features to enable such information to be used for future machine learning applications. It is critical not to miss any faces to protect children's privacy. RetinaFace (Deng et al., 2020) is already doing face and facial landmark detection in one go using the same model. The pre-trained model trained using a subset of the WiderFace dataset with manual facial landmark annotations was found to work with the video footage. However, the proposed design of separating face and facial landmark detectors is supposed to provide more control in anonymising faces by enabling more options of models and datasets to choose from. Face and facial landmark detections have been studied separately in the past decade. Combining both detections is known as multi-task learning requiring the dataset to have both annotation types and lesser options.

1.3 Aims and Objectives

The dissertation aims to blur faces in video footage when creating a dataset while maintaining high similarity to the original images by keeping reference on facial features. The proposed design can be used for other video footage in the future involving face blurring. The objectives to achieve the aim of the dissertation are listed below.

- Explore face detection and facial landmark detection algorithms.
- Explore obfuscation techniques.
- Explore datasets containing face and facial landmark labels.
- Evaluate trained models and obfuscation techniques.
- Compare the proposed design against the pre-trained RetinaFace model.
- Verify the trained models on video footages.

1.4 Structure of the Dissertation

Chapter 1 introduces the background information of the face blurring application and defines the project scope of the dissertation. It includes the motivation of the challenges of video footage containing children, existing methods and the hypothesis of the proposed design. The aims and objectives are clearly defined in this chapter.
Chapter 2 reviews the literature surrounding face detection and facial landmark localisation in the domain of face masks along with obfuscation techniques. It explores, compares and provides a conclusion on the algorithms used in recent studies.
Chapter 3 explains the proposed design of the face-mask face blurring application used to anonymise raw video footage of a new dataset. It lists the requirements and covers the application's theoretical knowledge of the components. The dataset used to train the models, and evaluation techniques are identified in this chapter.
Chapter 4 describes the practical work done to develop the face-masked face blurring application. It implements the proposed design using Python programming language with a training pipeline of the application. Open-source libraries and repositories are utilised with flowcharts and code listing to create the solutions.
Chapter 5 analyses the results of the components of the face-masked face blurring application. It is structured by experiments leading to the final solution of the face-mask face blurring application.

Chapter 6 concludes the entire work of the dissertation and lists future work to extend this work. It reviews the achievements of the dissertation with the aim, objectives and requirements of the face-masked face blurring application.

Chapter 2

Literature Review

The face-masked face blurring application can be broken down into three components: face detection and face landmark localisation in face masks, and obfuscation technique to minimise loss of data utility when anonymising individuals. The section reviews and compare recent works in these domains, followed by a conclusion of the studies.

2.1 Background

The World Health Organisation (WHO) declared the COVID-19 outbreak a global pandemic in February 2020. The incubation period after infection is estimated to be between two to fourteen days before displaying symptoms. Individuals may not display symptoms such as asymptomatic and pre-symptomatic. These cases are responsible for half the transmission of COVID-19 (Rab et al., 2020). As such, non-pharmaceutical interventions such as face masks are used to cut the primary source of SARS-CoV2 droplets from spreading by an infected individual. The usage of face masks imposed by the government in the early phases of the outbreak to minimise transmission of COVID-19 has led to a low mortality rate achieved by several countries in Asia (Rab et al., 2020). The European Centre for Disease Prevention and Control (ECDC) and countries followed similar recommendations, with some governments making the use of face masks compulsory in public areas or where social distancing is impossible. The COVID-19 pandemic has made face masks a new normal in our day-to-day lives.

There has been an increase in face-masked face detection studies recently due to the usage of face masks since the COVID-19 outbreak. Most works detect face masks by classifying if a face mask is present and proposed designs with binary outputs: face mask detected or face mask missing (Kühl et al., 2020). However, detecting face regions with face masks is considered more challenging in face detection in object detection. Face masks cause occlusion to the face blocking many features used by detectors. Furthermore, there are many types and styles of face masks, such as medical masks, KN95 masks, N95 masks and cloth masks. Face masks can be worn incorrectly with different orientations, levels of occlusion and sizes by individuals. Earlier face mask face detection studies faced insufficient face masked face dataset (Singh et al., 2021), resulting in generating synthetic face masks on known face datasets such as WiderFace using tools such as MaskTheFace [1] and MaskedFace-Net dataset containing artificially generated images of face masks (Ding et al., 2021). Today there are massive datasets containing individuals with real face masks and data augmentation techniques to increase further the number of training data (Nagavi et al., 2021).

The rise of privacy concerns regarding data processing has also increased studies on obfuscation techniques. The recent development of facial verification and classifications have achieved over 99% accuracy using neural networks (Croft et al., 2021). Machine learning models are available for organisations to embed machine learning tasks into their applications (De Cristofaro, 2020). Machine learning models are susceptible to membership inference attacks to deduce if an individual is part of the dataset and attribute inference attacks to infer sensitive features of individuals in the dataset, enabling attackers to derive personal information from the training dataset (Hu et al., 2021). However, models trained using anonymised data have better protection against these attacks and are exempt from the obligations in these regulations (Goldsteen et al., 2020). Anonymised data are information that cannot reasonably, with certainty or degree of probability to associate with a particular consumer. Data collectors using anonymised data can perform processing tasks without facing ethical or legal issues. ”Are publicly available datasets free of privacy concerns?” from (Yang et al., 2021) such as ImageNet containing objects with people in the images.

2.2 Face Detection

Face detection refers to detecting and localising faces in digital images, and it is a key area in the field of Computer Vision and Pattern Recognition (Sethi et al., 2021a). The initial designs of face detectors date back to the early 2000s using hand-crafted features and traditional machine learning algorithms. The advancement of data processing using parallelisation and a dedicated Graphic Processing Unit (GPU) to increase bandwidth enabled practical usage of Convolutional Neural Networks (CNN) architectures capable of extracting features and classifying to be widely used (Talukdar et al., 2018). CNN used in deep learning learns features without prior knowledge to construct feature extractors learning in an end-to-end manner (Fan and Jiang, 2021; Liu et al., 2021). Generic object

[1] `https://github.com/aqeelanwar/MaskTheFace`

detection methods such as single-stage and multi-stage approaches are inherited by face detectors (Deng et al., 2019). Most of the methods proposed in related work originated from object detection applied to the face mask domain.

2.2.1 Haar Feature-based cascade classifier

Haar feature-based cascade classifier is an earlier object detection method using object matching methods of haar-like features. The haar-like features are formed through contrast variance by the differential of light and dark area represented as the sum of pixel intensity (Wilson and Fernandez, 2006; Arif et al., 2021). The haar cascaded classifier is an old object detector. However, it is still used in today's application due to its rapid detection using AdaBoost classifier by cascading the classifiers into stages and selecting only a subset of features. It uses a sliding window system where sub-windows are formed and failing to move onto the following stage results at the end of detection. The frontal face detection will not work on face masks because only some features such as the eyes and eyebrows are detected but not the nose, cheeks, lips and chins to be considered a face (Arif et al., 2021).

It is used in (Sikand et al., 2021) to detect faces, passing the face region as input to a CNN for a face mask. The model has an accuracy of 98.39% but is limited to only the frontal face of the person and relies on the detector's capability as it captures the region of interest to be input for the CNN. The paper uses only the accuracy metric to measure the performance of the classification of CNN and lacks the evaluation of face detection bounding boxes from the haar cascade classifier.

The haar-like features require facial features to be visible and detect a face if all features are present. The paper (Sikand et al., 2021) proposed a design using haar-like features to detect if a face mask is present by requiring all facial features to be visible. It assumes the current frame has a person and detects the presence of a face mask if the nose, mouth and chin facial features are not detected. Such systems face limitations such as incorrectly detecting face masks by covering these facial features with the hands resulting in face masks being detected.

2.2.2 Single Stage Detectors

Single Stage Detectors use a single pass to perform detections at the cost of some loss in accuracy but gaining higher speed (Chavda et al., 2021). It takes an input image and learns class probabilities and bounding box coordinates by treating region proposals as simple regression problems (Sethi et al., 2021b).

RetinaFace predicts bounding box coordinates, prediction score and key-points for all faces in a single pass through pixel-wise localisation (Chavda et al., 2021). The model can predict facial landmarks and the main components consist of a feature pyramid network (FPN), common head module and cascade multi-task loss (Deng et al., 2019). The paper (Chavda et al., 2021) found the RetinaFace model with the lowest inference time between Dlib and Multi-task Cascaded Convolutional Neural Network (MTCNN) models for face detection. It dealt with video footage enabling information from previous frames to be used on the current frame. The paper improved the performance in video detection by processing the frames sequentially through a modified version of centroid tracking from object tracking techniques. It reduced the computational resources and improved the robustness of the system (Chavda et al., 2021). There is a pre-trained model of RetinaFace publicly available in Github [2] trained on the WiderFace dataset consisting of unobstructed and obstructed faces. The study in (Deng et al., 2019) manually annotated the facial features points on a subset of the WiderFace dataset.

RetinaFaceMask (also known as RetinaMask) uses a Context Attention Module (CAM) to discriminate face and face mask features and feature pyramid network (Fan and Jiang, 2021). The backbone architecture uses ResNet, and a lighter version uses MobileNetV1. The light version has lower mean Average Precision (mAP) than the standard version, with a score of 92% and 94.8% on the AIZOO dataset due to the smaller-sized model and number of parameters.

2.2.3 Multi Stage Detectors

Multi-stage detectors use multiple passes to perform detections. Most multi-stage detectors are done in two passes, such as Region-based Convolutional Neural Network (RCNN) and Faster-RCNN (Chavda et al., 2021). The selective search algorithm used in RCNN and Fast R-CNN impacts object detection's inference time, making real-time detection impossible. Instead, Faster R-CNN learns the region proposals using a Region Proposal Network (RPN) to propose likely anchors containing objects by ranking them (Ren et al., 2015; Singh et al., 2021). There is an overall performance to Faster R-CNN due to the detection stage and region proposal stage sharing layer improving feature representation (Singh et al., 2021). The paper (Eggert et al., 2017) evaluated Faster-RCNN on smaller objects with varying settings of the object detector. It is vital in the project as the video footage is recorded in lower resolution, and faces take up only a small section of the frame. The anchor scales and sizes of Faster-RCNN have to be tuned along with the feature extractor as deeper network architectures do not benefit from low-resolution classification problems (Eggert et al., 2017).

[2] `https://github.com/biubug6/Pytorch_Retinaface`

Multi-task Cascaded Convolutional Networks (MTCNN) is a three-stage process of detecting faces and facial landmarks. It consists of Proposal Network (P-Net) to obtain bounding boxes of candidate windows; Refine Network (R-Net) to reject false candidates; and detecting facial landmarks' positions (Zhang et al., 2016). MTCNN is used in (Vansh Gupta, 2021) to detect for faces as input to MobileNetV2 as backbone to detect mask. The paper did not include the proposed architecture results and fine-tuning of the MobileNetV2 architecture.

The paper (Sethi et al., 2021a) introduces an image complexity predictor to split images into hard and soft types. Single-stage detectors are applied to hard images, and two-stage detectors are applied to soft images. The authors note that these detectors have a trade-off between speed and accuracy; the proposed method benefits from both types of detectors as hard images have clear, distinct features while faces in soft images may be overlapped. The image complexity uses a smaller network VGG-f, an eight-layer deep CNN. It enabled the model to use both single-stage and two-stage detectors, increasing face detector scores but has no formal evaluation of the performance of image complexity.

2.2.4 Transfer Learning

Transfer learning is transferring knowledge from one domain to another by reusing pre-trained models with similar base networks without retraining the network (Asif et al., 2021). These models are trained on large datasets such as the ImageNet dataset containing 14 million images with more than 1,000 categories. As experimented in (Fan and Jiang, 2021) transfer learning for face mask detection can use face detection models due to the high correlation between the tasks. The weights of the pre-trained model can be used to initialise without random initialisation to save training time if the application is in the same domain.

MobileNetV2 is a mobile architecture designed for embedded hardware devices with limited computing resources (Sandler et al., 2018). It uses depth-wise separable convolutions with a depth-wise convolution layer as the filtering stage applying a single convolutional filter per input channel, followed by a point-wise convolution as a combination stage. The architecture has much lower complexity and parameters than standard convolutional kernels. It is based on the shortcut connections between thin bottleneck layers known as inverted residual structure (Sandler et al., 2018). MobileNetV2 is used in (Asif et al., 2021) as a feature extractor to classify if a person is wearing face masks and added six more layers consisting of average pooling layer, flattening layer, dense layer and drop-out layer followed by a softmax activation function to detect for a face mask. The paper uses Face Detection API from the Google ML tool kit as a black box to detect faces as a region of interest and pass it as input to the model.

InceptionV3 architecture developed by Google is a 48-layered convolutional neural network (Chowdary et al., 2020). It reduces number of parameters by factorizing convolutions with large filter size into smaller convolution layer and asymmetric convolution layers, e.g 5×5 convolution is factorised using two 3×3 convolution and 7×7 convolution is factorised using 1×7 and 7×1 convolution (Szegedy et al., 2016). InceptionV3 is used in (Chowdary et al., 2020) removed the last layer of the trained model and attached similar layers to (Asif et al., 2021) mentioned in the paragraph above.

ResNet-50 architecture is a 50 layers-deep convolutional neural network based on residual networks. It addresses vanishing/exploding gradients using residual blocks through skip connections in deep neural networks (He et al., 2016). ResNet-50 is used in (Loey et al., 2021) as feature detection network and YOLOv2 to detect for face-masked faces.

2.3 Face Alignment

Face alignment identifies the geometric structure of a human face given the location and size of a face to automatically determine face components (Li and Jain, 2015). The facial features can be extracted from face alignment computer vision technology. Partial occlusion covering some facial features is one of the challenges of face alignment, along with huge variance in facial appearance, lighting and noises (Ren et al., 2016). The application of face alignment is mainly used in face analysis tasks such as face recognition, facial animation, facial expression and understanding (Yang et al., 2015; Jourabloo et al., 2017; Feng et al., 2018). Face alignment is treated as a regression problem of identifying a point's x and y coordinates in the face structure. It is closely related to face detection, as a face detector is required to initialise face alignment by detecting the face region in an image. The face detection stage's quality affects face alignment as some face detectors have poor post-filter processes resulting in overlapped output rectangles (Ren et al., 2016).

2.3.1 Traditional Methods

Earlier face alignment techniques use hand-crafted features similar to face detection, such as Haar feature-based cascade classifiers, Scale-invariant feature transform (SIFT), etc. Similarly, several studies used the haar classifier to detect facial features. By limiting the image region to only the face area, the accuracy and efficiency increased as fewer areas were analysed to produce false positives (Wilson and Fernandez, 2006). The face structure was used in (Wilson and Fernandez, 2006) to separate the top part of the face for the eyes, the centre for the nose and the bottom for lips. The paper analyses only upright faces and not faces at different orientations. By fixing such facial features detection regions, the detector fails when faces are upside down. As Haar classifiers are affected by lighting conditions, pre-processing steps such as translation, scale and normalising to improve performance and also minimising over-fitting (Ding et al., 2021). The Haar cascaded classifiers are frequently mentioned in object detection due to their ability to form a strong classifier from a set of weak classifiers. Facial features can be detected through the cascaded-regression-based approaches but perform poorly in unconstrained faces due to their shallow structure (Feng et al., 2018).

2.3.2 Convolutional Neural Network

The recent methods use deep learning techniques involving Convolutional Neural Network (CNN) for facial landmark detection. Instead of hand-crafted feature extractors, CNN is used to extract features through training data using base networks from transfer learning. The deep neural network approach can scale hidden layers in the structure to increase the robustness of the intended goal.

The facial features were studied in a facial expression recognition system in (Liu et al., 2021) to classify the face expressions. The model's output is face expression labels, the facial features are encoded in the hidden layers, and coordinate values are not obtainable with the proposed design. The paper proposed a fusion network backbone for the feature extractor to balance speed and accuracy, including VGG-16 and ResNet (Liu et al., 2021) for more discriminative features. It is interesting as designing better convolutional neural network architectures is still based on intuitions and remains an open question.

The paper (Lin et al., 2021) detects facial features for fatigue detection and classifies if the face mask is worn correctly, incorrectly or missing. The proposed design consists of Multi-task Cascaded Convolutional Neural Network (MTCNN) to detect face and facial features, MobileNet, and Gradient Boosted Decision Trees (GBDT). The GBDT is a cascaded regression tree used for face alignment by randomly selecting coordinates of the detected facial features and rotating it to ensure the horizontal line relative to the eyes is at the centre of the picture. It is similar to a normalisation step as the same faces can vary in an unconstrained environment due to lighting, occlusion and posture. The proposed design uses a pre-trained MTCNN face detector and faces an issue when the input image is wearing a mask as the pre-trained model is not suited for occluded faces.

Regression-based approaches have been used for facial landmark localisation using different network types and loss functions. The paper (Feng et al., 2018) proposed a new loss function called ”Wing loss” and analysed the performance of existing neural networks in the domain of facial landmark localisation. The common loss functions used in regression-based approaches are L1 and L2 losses, but these losses face issues in correcting small errors. The wing loss increases the contribution of small and medium error samples in the network when training (Feng et al., 2018). Pose-based data balancing was applied to further increase the robustness of facial landmark localisation in unconstrained environments, as most datasets have faces in frontal view. The data balancing techniques applied are similar to other pre-processing techniques but include a clustering step to group similar faces together and expand the group with the least number of faces.

2.4 Obfuscation Technique

Obfuscation techniques are methods to hide sensitive information by either removing or altering features from images while retaining visual features for processing (Tekli et al., 2019). However, other visual cues such as clothing and height can infer the individual's identity from face-blurred images (Yang et al., 2021). These methods make a trade-off between privacy, quality and data utility of the dataset. In (Yang et al., 2021) showed a slight decrease in the performance of visual recognition systems trained on face obfuscated dataset, yet the models were still able to learn transferable features. An anonymised individual can still be identified through restoration-based attacks restoring original features and recognition-based attacks training on obfuscated information (Tekli et al., 2019).

In (Tekli et al., 2019) groups main obfuscation techniques into three categories: pixelating overlaying a pixel box such as down-sampling; blurring such as blur kernel or motion blur, and masking replacing black pixels onto original pixels. The paper evaluates these techniques against the attacks mentioned in the previous paragraph with structural and identity-based metrics and recommends the most robust obfuscation technique for CelebA. The paper has knowledge of the type of attack, and this information is not known in a real-world setting as public datasets are available to the public and attackers. As (Yang et al., 2021) argued that privacy is not guaranteed prior to not knowing the attacker's knowledge without losing datasets' utility; hence uses only the blurring obfuscation method with metrics focusing on transfer learnings. These methods on facial images failed to provide a formal privacy guarantee (Croft et al., 2021). Face-blurring technology is commonly used in publishing sensitive images or video footage. It is already integrated into Google Maps Street Map and YouTube before publishing video (Weiss, 2012).

2.4.1 Generative Adversarial Network

Generative Adversarial Network (GAN) is a model with two networks competing in a minimax game through an adversarial training scheme to generate realistic images (Wu et al., 2019; Croft et al., 2021; Goodfellow et al., 2020). GAN is unsupervised learning via generative modelling consisting of a generator learning the distribution of training samples and a discriminator to estimate whether the output from the generator is real or fake (Goodfellow et al., 2020).

Deep Convolutional Generative Adversarial Network (DCGAN) is an architecture for making training more stable in GAN (Radford et al., 2015). The paper provides guidelines in the convolutional layers for the generator and discriminator to design the network. Conditional GAN (cGAN) uses auxiliary information such as class labels to train the network enabling targeted output (Langr and Bok, 2019). In the project context, cGAN takes face image as input and outputs obfuscated face. In (Wu et al., 2019) uses cGAN with face verification to de-identify the input image and Structural Similarity Index (SSIM) in assessing perceptual image degradation as objective measurement. The proposed method can only be used for frontal faces and tight crop on the face region. These limitations are overcome in the proposed method by (Croft et al., 2021) using RingNet to capture different face poses and Mask R-CNN to remove background from the face. After PCA transformation, it is based on cGAN and obfuscates faces by adding noise onto the latent space. The shortcut connections in the encoder and decoder network are skipped to prevent leakage of sensitive information (Croft et al., 2021). Mask R-CNN enables obfuscated face to fit correctly with the existing background in the image. These obfuscation methods are applied to un-obfuscated faces without face masks.

Another privacy-preserving framework is adversarial face obfuscation by applying a perturbation to faces evading face recognition systems. GAN is used in (Deb et al., 2020) as a perturbation generator on an input image and a face recognition as the discriminator. It can use an input probe with a face to perform obfuscation attacks and impersonation attacks resulting in a decrease in confidence in face recognition systems. Another adversarial face obfuscation is proposed in (Chandrasekaran et al., 2020) without using GAN to apply perturbation only for obfuscation attack. However, adversarial face obfuscation can still be identified using normal human observation. Instead of restricting surveillance, blurring faces for anonymity is encouraged without facing legal issues (Marks, 2009).

2.5 Conclusion

COVID-19 has increased studies on face occlusions in the face and facial detection domain. Similar techniques and methods from object detection in other applications can be applied to face and facial detection. The quantity and quality of annotated dataset affect the performance of the trained model. Data augmentation techniques such as data expansion and oversampling can be used in a small dataset with insufficient samples to generate more training samples and increase diversity in the dataset. As the face and facial detection field has been widely studied, synthetic face masks can be generated and applied to clear faces in existing annotated datasets to create face masked datasets from open source tools. The main components of a neural network are the backbone, neck and head in object recognition problems (Sethi et al., 2021a). There are different neural network architectures with a varying number of hidden layers and complexity. The larger the model's size and the number of parameters, the greater its capability in extracting features resulting in higher accuracy but increasing computation resources. Transfer learning is another approach when dataset and time are limited, using weights from similar domains to the current application. It uses the backbone of other trained models to be frozen or updated in training. The trained models covered in (2.2.4) are publicly available, they can be used as the backbone and further fine-tuned to fit the project's solution in the neck and head component. Single-stage detectors and multi-stage detectors are based on speed and accuracy trade-offs. The video consists of several images in a second, and the model must process many images. To further improve the model's speed, hardware accelerators such as Graphics Processing Unit (GPU) can be used to increase the speed of training and inference. The blurring face obfuscation technique can evade human and facial recognition systems solely based on the region of interest on the head. It avoids strict regulations such as the EU General Data Protection Regulation (GDPR) and California Consumer Protection Act (CCPA), as the data is anonymised and de-identified. However, blurring completely removes data utility for processing, losing information on facial landmarks. Instead, GAN may be a better technique to obfuscate faces while retaining the facial landmarks by probing input images with noise to anonymise the individual. No related work was done using GAN as an obfuscation method on faces with face masks. However, GAN consumes much more resources than blurring image obfuscation, and the video footages have many frames and faces.

Chapter 3

Design

This chapter covers the requirements of the dissertation when designing the face-masked face blurring application. The unannotated video footages of the children's dataset are described in this chapter, along with the dataset selected to train the models. The overview and technical details of the proposed design are explained in this chapter. The kernel filters are one of the main focus of this chapter as it is involved in all stages throughout the proposed design, such as the Convolutional Neural Network (CNN) and Gaussian blur.

3.1 Requirements

The requirements of the face-masked face blurring application are high recall of face detection, processing frames in the video footage with reasonable time and protecting the privacy of the faces.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{3.1}$$

Recall is defined as the ratio of the number of detected true positive to the total actual positive as described in equation (3.1). True positive is the number of correct positive predictions made by the model. False positive is the number of incorrect positive predictions made by the model. In the face-masked face detection case, the recall is calculated by the number of correct faces detected by the model against the total number of actual faces in the frame. The recall increases when the model can correctly detect a face, vice versa for when the model misses a face and decreases when it detects for face incorrectly (false positive).

The face-masked face blurring application is used after collecting images or video footage before releasing the dataset or training a new model. It does not need to process frames in real time but processes frames in a reasonable time, at least 10 frames per second (FPS).

After processing images or video footage using the face-masked face blurring application, participants' faces should be anonymised and not identifiable. Other face detection or facial landmark localisation tools must still be able to detect faces and facial features of the processed images or video footage.

3.2 Overview of the Approach

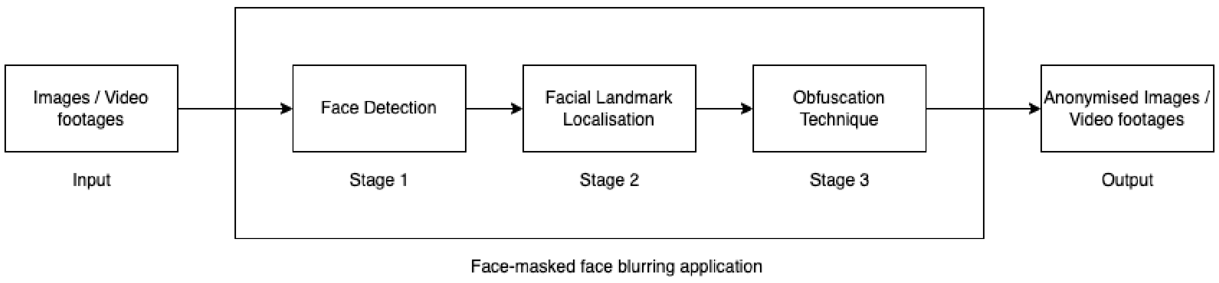
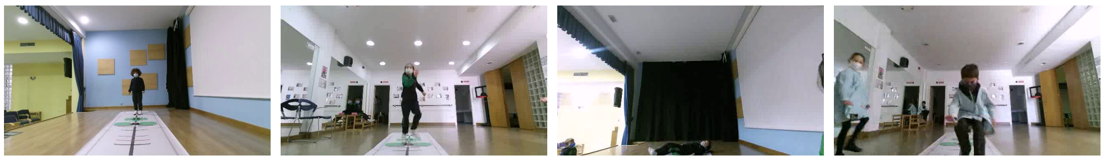

Images / Video footages
Input
Face Detection
Stage 1
Facial Landmark Localisation
Stage 2
Obfuscation Technique
Stage 3
Anonymised Images / Video footages
Output
Face-masked face blurring application

Figure 3.1: Overview of Face-masked Face Blurring Application Design

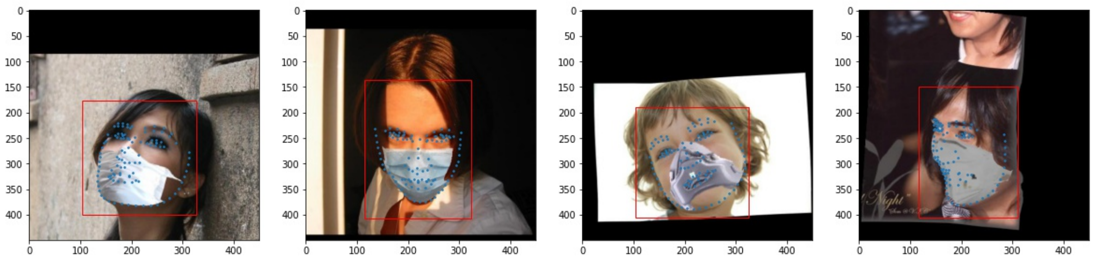
The proposed design for the face-masked face blurring application consists of three stages: face detection, facial landmark localisation and obfuscation technique. Figure 3.1 describes the application flow from input to output. This approach enables a huge selection of methods for each stage instead of a face and facial landmark localisation detector such as RetinaFace performing these tasks in a single go. Similarly, separating face detection and facial landmark localisation into two stages enables training to use different datasets. The face detector stage processes the entire input frame and outputs the region of one or more faces detected to the next stage. The facial landmark localisation stage processes only the face region, similar to related works covered in chapter 2.3. Different feature extractors in the backbone of the network architecture can be used and trained for the face detector and facial landmark localisation. The obfuscation stage has the core references of the facial landmarks and applies different levels of image obfuscation onto the face to maintain data utility.

The methods of each stage for the face-masked face blurring application are Faster R-CNN, convolutional neural network and blur. The face detection stage is important as undetected faces are not processed and obfuscated. A two-stage detector is selected as a high recall is required, and real-time processing is not needed. The convolutional neural network for facial landmark localisation can be a less complex model compared to Faster R-CNN as the facial landmark localisation focuses on a smaller region. The blurring stage



(a) (b) (c) (d)

Figure 3.2: Frames from video footages



(a) (b) (c) (d)

Figure 3.3: Images and annotations from the JD-landmark-mask dataset

uses computer vision techniques such as Gaussian blur with varying kernel sizes to control the blur level on the face regions.

3.3 Dataset

3.3.1 Video Footages

The video footage containing children is recorded in an unconstrained environment; they have no face bounding boxes and facial landmark annotations, as shown in figure (3.2). It requires the face-masked face blurring application to anonymise the faces of children to protect the identity of the participants and avoid legal regulations. The face detection and facial landmark localisation models require annotated dataset for training. After training the models, the face-masked face blurring application has to be applied to the video footage. It is similar to the transfer learning principles where the weights are trained with another dataset and used for another application in the same domain.

3.3.2 JD-landmark-mask Dataset

The 3rd Grand Challenge of 106-Point Facial Landmark Localization, also known as JD-landmark-mask (Xiang et al., 2021) dataset, is selected for training the face detection and facial landmark localisation models. The dataset was released during the COVID-19 pandemic in 2021 with 27,000 face images, including real and virtual masks separated by the training, validation, and test sets. Only the training and validation sets are publicly available, while the authors kept the test set. It consists of face bounding box and facial landmark annotations. Figure (3.3) are images with both annotations from the dataset. The dataset has varying image resolution, head pose and most adults with some children samples. The entire image frame will be used to train the face detection Faster R-CNN model, while the facial landmark localisation CNN will use only the cropped face for training.

3.4 Faster R-CNN

Faster R-CNN is a two-stage detector which extends Fast R-CNN by using a Region Proposal Network (RPN) to improve the running time of the detection network. It started from region-based CNN (R-CNN) using CNN to classify and adjust possible regions of interest from a selective search algorithm (Girshick et al., 2014). Fast R-CNN came next to improve the performance of the model by sharing computations using Region of Interest (ROI) pooling (Girshick, 2015). Faster R-CNN shares the feature map from the CNN layers for the RPN and Fast R-CNN.

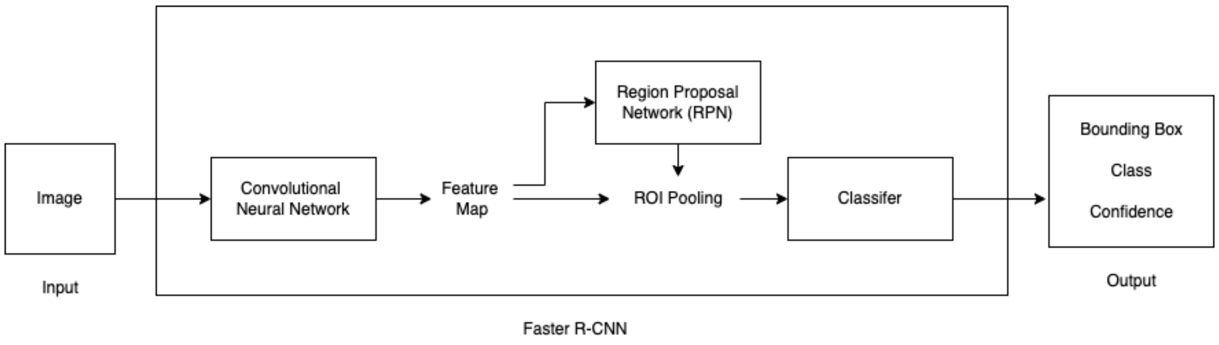


Figure 3.4: Faster R-CNN Architecture

Figure (3.4) is the architecture diagram of Faster R-CNN (Ren et al., 2015) which describes the flow from the input image, components of Faster R-CNN and output of the detector. The convolutional neural network can use different backbones from transfer learning with pre-trained weights as feature extractors creating a feature map. RPN uses the feature map to output possible ROIs. ROI pooling converts the feature map of the ROI proposal section into a fixed-size map for the classifier. The output is the coordinates of the bounding boxes, class label and confidence of the detector. Training of faster R-CNN uses back-propagation to update the weights of the network. It has four losses: RPN classification loss, RPN regression loss, classifier loss and classifier regression loss.

3.4.1 Region Proposal Network (RPN)

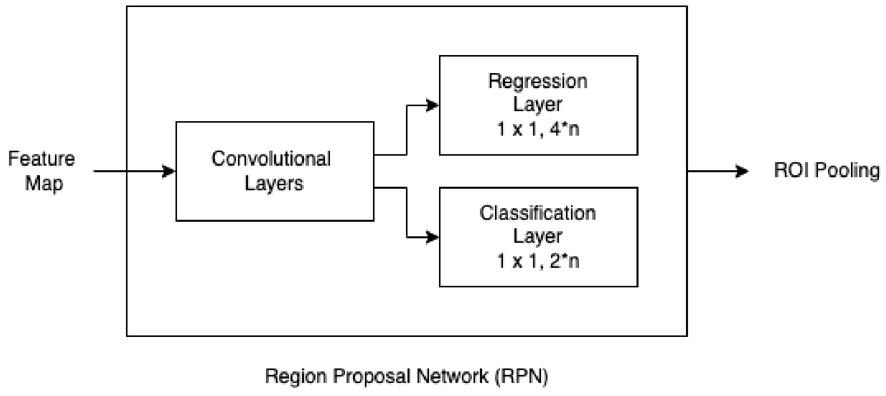

Feature Map
Convolutional Layers
Regression Layer 1 x 1, 4*n
Classification Layer 1 x 1, 2*n
ROI Pooling
Region Proposal Network (RPN)

Figure 3.5: Region Proposal Network (RPN) Architecture

Figure (3.5) is the architecture of the Region Proposal Network (RPN) using feature map input and outputs proposed regions to the ROI pooling layer. RPN aims to generate proposals using a shared feature map with the classifier of Faster R-CNN. It uses a referenced centre point known as anchors to generate different combinations of aspect ratios and sizes through a sliding window of the feature map. The convolutional layer in RPN acts as an intermediate layer to extract feature maps for proposal generation. The regression layer predicts the box parameters d_x, d_y, d_w, d_h. The classification layer predicts a binary output of whether an object is present. The hyperparameters of RPN are the anchor sizes and aspect ratios, and it is represented as n in figure (3.5).

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{3.2}$$

$$\text{smooth}L_1(x) = \begin{cases} 0.5x^2 & \text{if } |x| \leq 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

$$\text{IoU}(A, B) = \frac{A \cap B}{A \cup B} \tag{3.3}$$

Equation (3.2) is a multi-task loss function of training the RPN consisting of a classification loss and regression loss (Ren et al., 2015). The classification loss $\sum_i L_{cls}(p_i, p_i^*)$ is the sum of loss over all anchors and two classes. The regression loss $\sum_i p_i^* L_{reg}(t_i, t_i^*)$ depends on the positive anchors p_i^* and evaluates the difference between t_i and t_i^* using smooth L1 function defined in equation (3.4.1). The λ parameter is used to balance the weight of regression and classification terms. The anchors are labelled as positive and negative class labels using Intersection over Union in equation (3.3). The positive anchor labels are labelled either by the highest IoU with the ground truth box or at IoU of at least 0.7 with any ground truth box. The negative anchor labels are labelled when the IoU is less than 0.3. The RPN training minimises the loss function using optimisation techniques.

3.4.2 Classifier Network

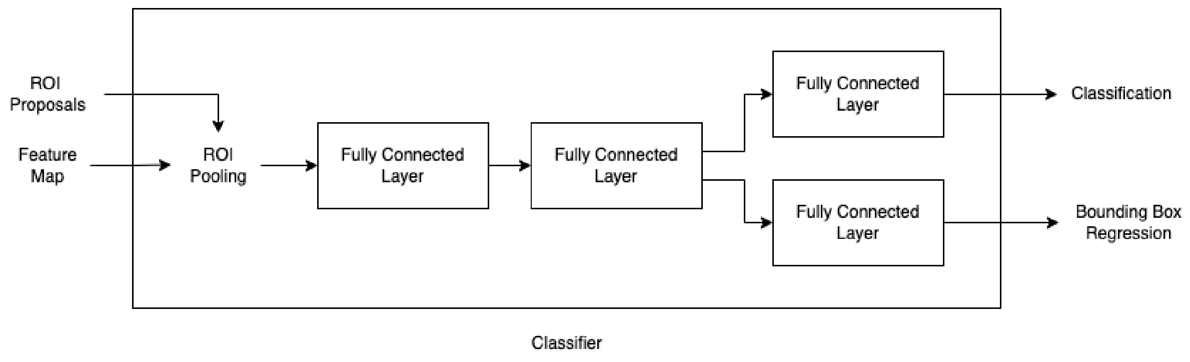


Figure 3.6: Classifier Network Architecture

Figure (3.6) is the architecture of the classifier network in Faster R-CNN using the feature map and the proposal region from RPN as inputs. It is the second and last stage of the detector outputting the x, y, width and height of the bounding box with class labels. Unlike the RPN in section (3.4.1), the classification outputs the actual class instead of binary output.

$$\text{Cross Entropy Loss} = -\sum_i y_i \times log y_i^* \tag{3.4}$$

The classifier network uses a multi-task loss function similar to the RPN loss function. The classification loss uses Cross Entropy loss equation (3.4) where i is the class labels. Cross Entropy loss measures the probability of the model's prediction of the label against the truth label.

3.5 Convolutional Neural Network

Convolutional Neural Network (CNN) is a sequence of input, hidden, and output layers using mainly convolutional, pooling and fully connected types of layers. It is common to have many layers in a CNN as the earlier layers capture local image features and deeper layers capture larger features by combining these local image features. The receptive field of CNN is larger for deeper layers.

3.5.1 Convolutional Layer

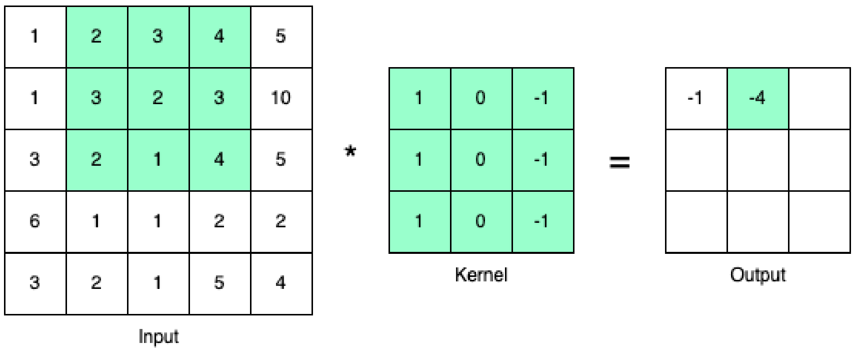


Figure 3.7: Convolutional Layer Simple Example

Figure (3.7) is a simple example of a convolutional layer convolving an input with a kernel. The kernel slides across the input, performing a weighted sum to get the output. A CNN learns the weight (values) of the convolutional kernel from a cost function and gradient descent.

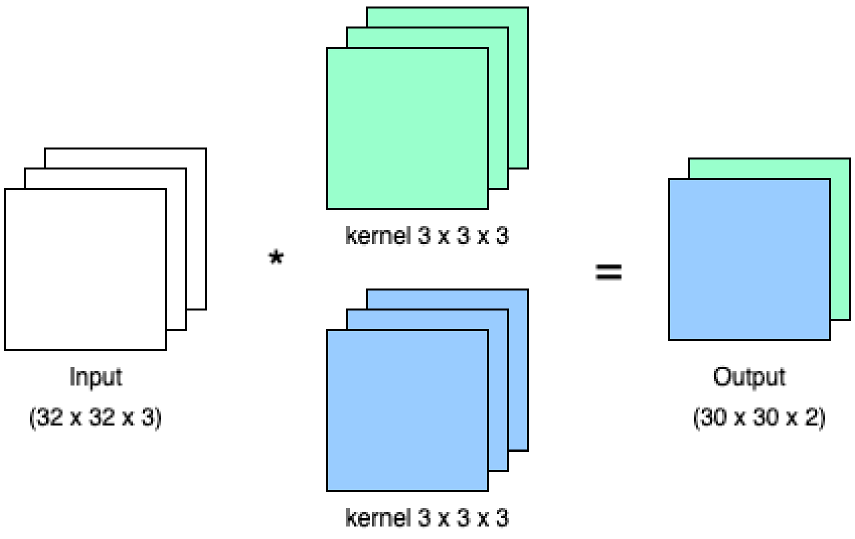


Figure 3.8: Convolutional Layer Channels and Filters Example

Figure (3.8) is another convolutional layer with multiple channels and filters. The input has three channels similar to an RGB image. A greyscale image has one channel and uses convolution of 1 channel. The convolutional layer has two filters, each with three channels to convolve against the corresponding channels of the input. The output is the result of the convolutional kernels. Padding can be used to add extra rows and columns to maintain the output size. The default convolutional layer has a stride of 1, which moves the kernel by one column/row at each step. Strided convolution uses a larger stride value to further reduce the size of the output.

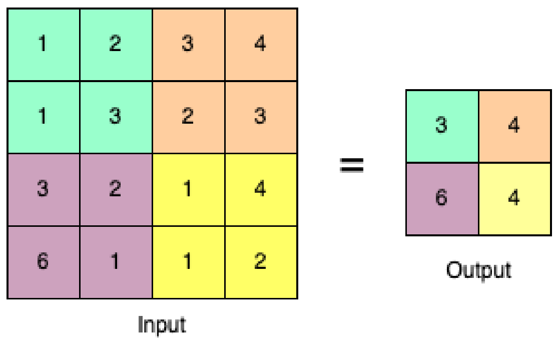


Figure 3.9: Max-Pool Layer Example

The pooling layer is another commonly used layer in a CNN to reduce the size of the output. Figure (3.10) is an example of a max pooling layer using a 2 x 2 kernel size with a stride of 2. It selects the maximum value in the kernel instead of a weighted sum and reduces the output size. Average pooling is another type of pooling layer which outputs the average of the block.

3.5.2 Fully Connected Layer

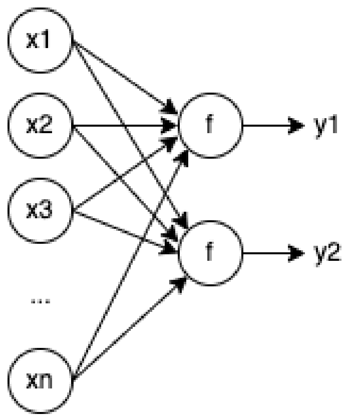


Figure 3.10: Fully Connected Layer Example

$$y = f(w^T x) \tag{3.5}$$

A fully Connected Layer (FC) layer is usually used at the end of a CNN to generate the final output, such as labels or values. The inputs of the layer are connected to every output layer with a non-linear activation function, as shown in figure (3.10). FC layer using the convolutional layer as input requires a flattening stage to convert the matrix input to a list. Equation (3.5) is the output function for each output node in the FC layer, and it is the function of a weighted sum of the input values. x denotes the input, w

denotes the trainable weights, $f(\cdot)$ denotes the non-linear function such as ReLU, softmax etc.

3.5.3 Feature Extractor

Based on the concept of transfer learning, the backbone of a pre-trained CNN model can be used on different object detection applications in the same domain. The FL output layers of a pre-trained CNN model are removed and replaced with a new output layer for the new application. The backbone extracts features encoded as a feature map as shown in figure (3.4). The face detection and facial landmark localisation compare backbones such as VGG16, MobileNetV3, ResNet50 and ResNet18.

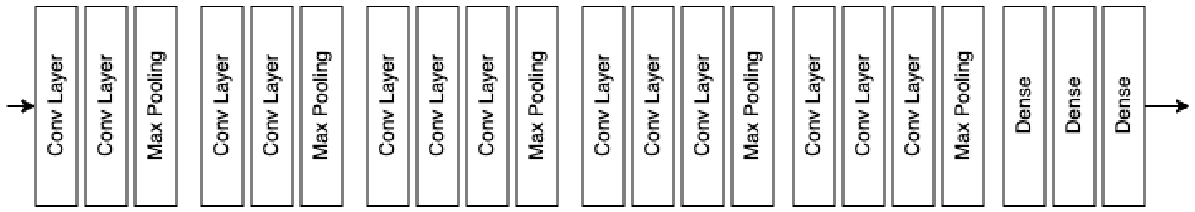


Figure 3.11: VGG16 Network Architecture

VGG16 uses 16 weight layers and small 3×3 kernels with many layers to have large receptive field. Figure (3.11) is the network architecture of VGG16, and it has a pattern of convolutional layers followed by a max pooling layer.

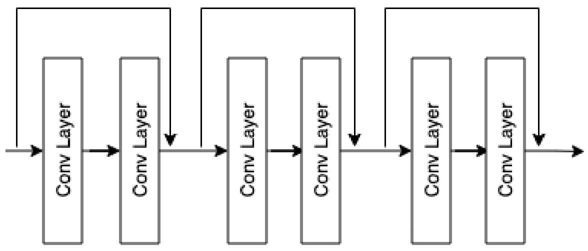


Figure 3.12: ResNet Skip Connection

ResNet uses a similar 3×3 kernel to VGG but uses a strided convolutional layer for downsampling. Figure (3.12) is the skip connection used in ResNet to deal with vanishing gradient problem (degradation) as the depth of network increases (He et al., 2016). The skip connection adds input from earlier layers to the output. Face detection uses ResNet50 with 50 deep layers. The facial landmark localisation uses ResNet18 with 18 deep layers.

MobileNetV3 is a lightweight CNN architecture used for mobile devices. It is similar to ResNet using skip connection but introduces a bottleneck block consisting of a piece-wise and depth-wise convolutional layer to reduce the number of parameters. Depth-wise convolutional layer applies a convolutional filter for each channel of the input. Piece-wise convolutional layer uses a 1×1 kernel. Besides reducing parameter size, the combination of a depth-wise and piece-wise convolutional layer is used to expand and shrink the outputs in the CNN.

3.5.4 Facial Landmark Localisation

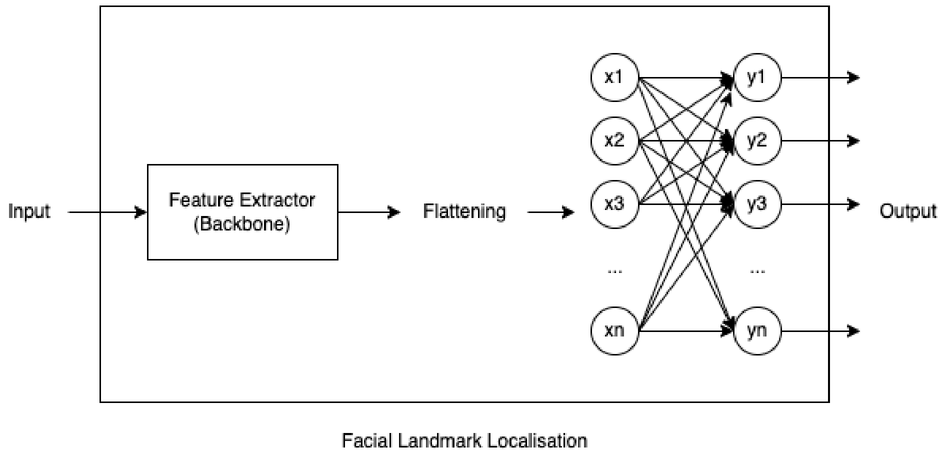


Figure 3.13: Facial Landmark Localisation Network Architecture

Figure (3.13) is the network architecture of the facial landmark localisation model consisting of a feature extractor, flattening and fully connected layers. The output of the model is the coordinates of the facial features. By training the model with the selected JD-landmark-mask dataset in section (3.3.2) with 106 facial landmark points, the model's output will be 212 predicting the x and y values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i^*)^2 \tag{3.6}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - Y_i^*| \tag{3.7}$$

The loss function used to train the network is the Mean Squared Error (MSE) as shown in equation (3.6) and equation (3.7) is the Mean Absolute Error, n denotes the number of samples, Y_i denotes the prediction by the model and Y_i^* denotes the ground truth of the sample. MSE is the same as $\ell 2$ loss, and MAE is the same as $\ell 1$ loss.

3.6 Blur

Gaussian blur is a type of linear filtering that performs a neighbourhood operator similar to kernels convolutional layers by a small neighbourhood weighted sum of input pixel values (Szeliski, 2010).

$$g = f \bigotimes h \tag{3.8}$$

$$g(i,j) \sum_{k,l} f(i+k, j+l) h(k,l) \tag{3.9}$$

Equation (3.8) is the compacted form of equation (3.9) using the correlation operator. The $h(k, l)$ term denotes the kernel and i, j denotes the pixel values.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}} \tag{3.10}$$

Gaussian blur uses a normal distribution kernel equation (3.10) with varying kernel sizes. Larger Gaussian kernel results in more blur, and vice versa for a smaller Gaussian kernel. The blurring is applied to the last stage of the face-masked face blurring application. Different intensities of blur can be applied to the face and facial features from the results of face detection and facial landmark localisation.

3.7 Evaluation

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{3.11}$$

The mean Average Precision (mAP) is a common metric used to evaluate object detection challenges such as the PASCAL Visual Object Classes (VOC) (Everingham et al., 2010), MS Common Objects in Context (Lin et al., 2014), etc. The precision and recall curve is summarised by the AP metric, and a high score requires precision at all levels of recall. The mAP metric requires an IoU threshold as object detectors output confidence. It uses detection above the selected IoU threshold to calculate the true positive (TP), false positive (FP) and false negative (FN). The recall and precision are calculated as described by equation (3.1) and (3.11) respectively. The precision and recall are interpolated to 11 points from 0 to 1 in a precision-recall curve. The mAP is the area under the precision-recall curve.

Chapter 4

Implementation

The chapter covers the implementation solution by starting with the overview, followed by each component, such as Faster R-CNN, CNN and Gaussian Blur.

4.1 Overview of the Solution

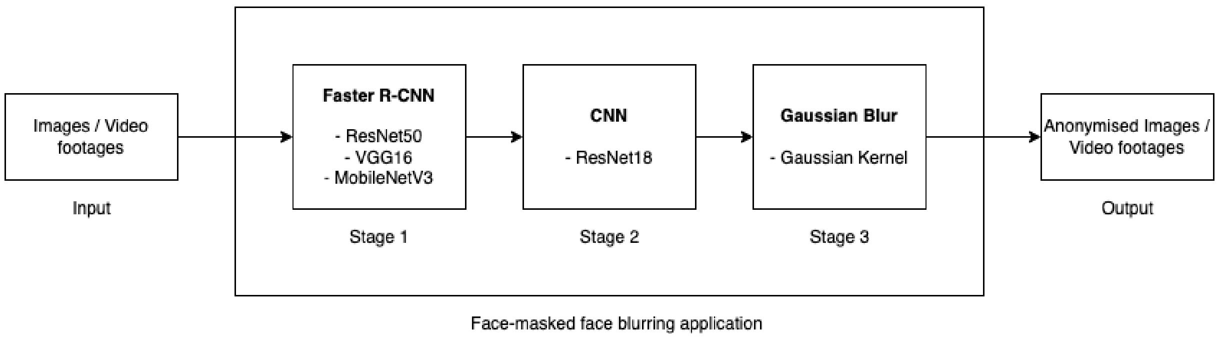
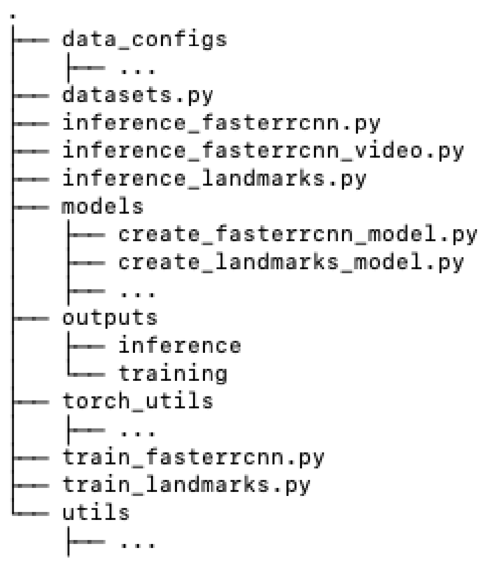


Figure 4.1: Face-masked Face Blurring Application Design Solution

The selected components for the proposed design of the face-masked face blurring application are implemented in Python programming language. Figure (4.1) provides an overview of the implemented solutions in the dissertation with different types of feature extractors for face detection and facial landmark localisation and kernel sizes for face blurring. The faster R-CNN and CNN in stages 1 and 2 are deep learning models that require training using an annotated dataset. As the face-masked face blurring application is used to blur faces in a new dataset, the models are trained using other annotated datasets such as the JD-landmark-mask dataset described in section (3.3.2).

Python is a high-level programming language and open source, making it free to use; it has many libraries and packages for computer vision techniques. The application will mainly use the PyTorch library (Paszke et al., 2019) containing many tensor compu-



```
.
├── data_configs
│   ├── ...
├── datasets.py
├── inference_fasterrcnn.py
├── inference_fasterrcnn_video.py
├── inference_landmarks.py
├── models
│   ├── create_fasterrcnn_model.py
│   ├── create_landmarks_model.py
│   ├── ...
├── outputs
│   ├── inference
│   └── training
├── torch_utils
│   ├── ...
├── train_fasterrcnn.py
├── train_landmarks.py
└── utils
    ├── ...
```

Figure 4.2: The project structure of the face-masked face blurring application

tations and deep neural networks, enabling more time to train the models instead of reinventing the wheel. The source code of the face-masked face blurring application is stored in a Github repository [1]. The code and structure of the folder directory follows similarly to an open-sourced "fastercnn-pytorch-training-pipeline" repository [2] as shown in figure (4.2). The `data_configs` directory contains configuration settings such as the path to the dataset, class names and the number of landmarks as the face-masked face blurring application can be used elsewhere and trained with different datasets. Following PyTorch's required API, the `dataset.py` is reimplemented to fit the JD-landmark-mask dataset annotations. The `models` directory contains model instantiating functions for face detection and facial landmark localisation. The `outputs` directory is used by the training and inference code to generate trained models and predictions of models. The `torch_utils` and `utils` contains boilerplate code for utility functions such as plotting, logging, metrics and etc.

The PyTorch library can only utilise CUDA GPUs from NVIDIA for training and inference by loading models and data onto the GPU. The training and evaluation are done on Google Colab due to its support for hardware accelerators such as NVIDIAs GPU and TPU to decrease the processing time. Google Colab hosts the web-based interactive python development environment Jupyter Notebook on the cloud connected to Google's resources. The code repository is pulled, updated and ran on the cells of Google Colab, and the outputs are saved in Google Drive, reducing the risks of data loss.

[1] `https://github.com/limm5/CS7CS5-Dissertation`

[2] `https://github.com/sovit-123/fastercnn-pytorch-training-pipeline`

4.2 Face Detection

The PyTorch library has already implemented a Faster R-CNN class based on the original paper (Ren et al., 2015). It allows instantiating the class with various hyper-parameters covered in the previous chapter (3.4).

```
from torchvision.models.detection import FasterRCNN
...
    model = FasterRCNN(
        backbone = backbone, # feature extractor
        num_classes = num_classes, # 2
        rpn_anchor_generator = anchor_generator,
        box_roi_pool = roi_pooler
    )
```

Listing 4.1: PyTorch API for instantiating a Faster R-CNN model

The code snippet in listing (4.1) shows an example of creating a Faster R-CNN model using PyTorch's API. The pre-trained Faster R-CNN model provided by PyTorch is trained on the COCO dataset containing 80 classes. However, the face detection stage of the face-masked face blurring application only requires a single class "face" and "background". When reusing the pre-trained model for the face detection stage, the final fully connected layers must be updated to output only 2 classes. The following subsections include more details on each of the parameters of the `FasterRCNN(...)` class function.

4.2.1 Training

```
python train_fasterrcnn.py
    --model MODEL
    --config DATA_CONFIG
    --epoch NUM_EPOCH
    --batch-size NUM_BATCH_SIZE
    --project-name PROJECT_NAME
    --weights MODEL_WEIGHT
```

Listing 4.2: Command for Faster R-CNN training with arguments

The command to start or resume training a face detection model is described by listing (4.2) using the `train_fasterrcnn.py` script. The `MODEL` argument selects a model from the `create_fasterrcnn_model.py` file from the `models` directory. The `NUM_EPOCH` is the number of times model trains with the entire dataset, and `BATCH_SIZE` is the number of samples model trains per batch. The `PROJECT_NAME` is just a name given to the trained model. The `MODEL_WEIGHT` is a model checkpoint used to resume an existing training.

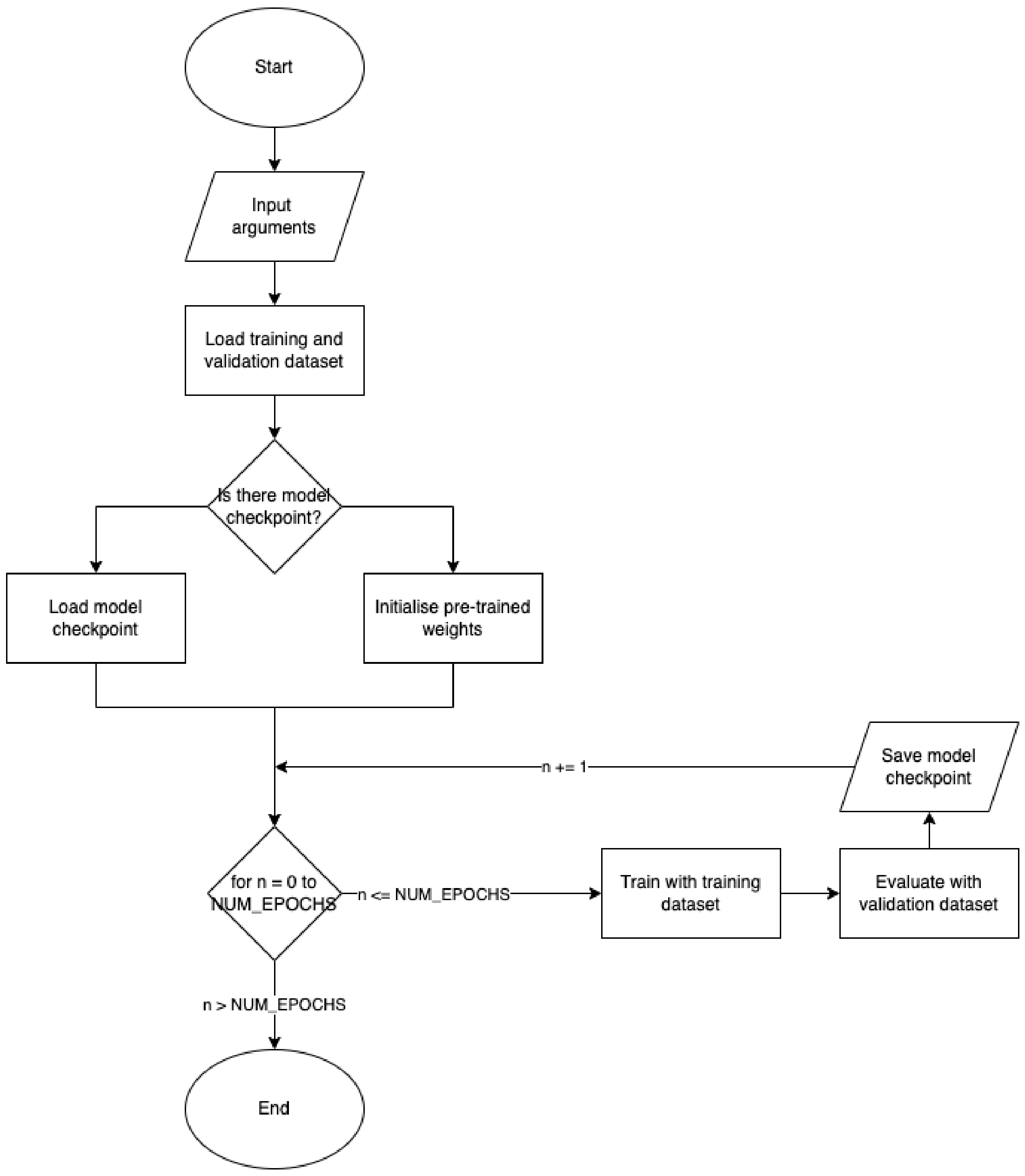
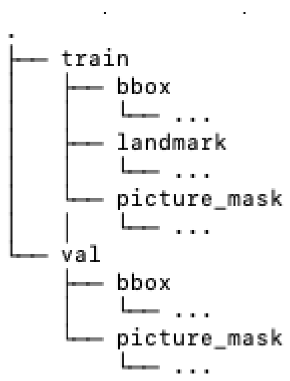

Start
Input arguments
Load training and validation dataset
Is there model checkpoint?
Load model checkpoint
Initialise pre-trained weights
n += 1
Save model checkpoint
for n = 0 to NUM_EPOCHS
n <= NUM_EPOCHS
Train with training dataset
Evaluate with validation dataset
n > NUM_EPOCHS
End

Figure 4.3: The Faster R-CNN Training Flowchart

Figure (4.3) is the flowchart of training Faster R-CNN models using the `train_fasterrcnn.py` script file in the directory figure (4.2). The training and validation dataset is loaded using the file paths from the `config` file. The model is created, and weights are loaded from a model checkpoint or initialised from a pre-trained weight depending if the model checkpoint is passed into the arguments. The training begins by looping over the entire dataset depending on the `NUM_EPOCHS`. In each epoch, the model is trained using mini-



```
.
├── train
│   ├── bbox
│   │   └── ...
│   ├── landmark
│   │   └── ...
│   └── picture_mask
│       └── ...
└── val
    ├── bbox
    │   └── ...
    └── picture_mask
        └── ...
```

Figure 4.4: JD-landmark-mask Dataset Folder Structure

batch gradient descent, evaluated with the validation dataset and the model checkpoint is saved. Mini-batch gradient descent trains the model by splitting the dataset into batches based on `batch_size`. The `batch_size` can be set to the number of training samples, also known as batch gradient descent but suffers from hardware memory limitation due to the enormous data size and optimisation stuck at a local minima. The goal of gradient descent is to reach the global minima, which is the point where the loss is the minimum. Using a batch size of 1, also known as stochastic gradient descent (SGD), training will take a lot of time because SGD estimates the gradient producing noise, but these noises enable descent to escape local minimas. Mini-batch gradient descent enables descent to escape local minima and produce lesser noise. These issues are due to the non-convex nature of deep learning models. The model performs a forward propagation and calculates error and backpropagation to update the model weights. The model checkpoint is saved to ensure the resumption of training if the model has not trained to the end of `NUM_EPOCH`.

4.2.2 Preprocessing

```
import albumentations as A
from albumentations.pytorch import ToTensorV2

def get_train_aug(img_size=512):
    return A.Compose([
        A.MotionBlur(blur_limit=3, p=0.5),
        A.Blur(blur_limit=5, p=0.5),
        A.RandomBrightnessContrast(
            brightness_limit=0.2, p=0.5
        ),
        A.ColorJitter(p=0.5),
        # A.Rotate(limit=10, p=0.2),
        A.RandomGamma(p=0.2),
        A.RandomFog(p=0.2),
        A.Downscale(
            scale_min=0.05, scale_max=0.25,
            always_apply=True
        ),
        A.LongestMaxSize(
            max_size=[64,128,256,img_size],
            interpolation=1, always_apply=True
        ),
        A.PadIfNeeded(
            min_height=img_size,
            min_width=img_size, border_mode=0,
            value=(0,0,0)
        ),
        ToTensorV2(p=1.0),
    ], bbox_params={
        'format': 'pascal_voc',
        'label_fields': ['labels']
    })
```

Listing 4.3: Face Detection Preprocessing using Albumentations

The Albumentations python library is used for pre-processing the dataset when loading it into the PyTorch's `DataLoader` class. Listing (4.3) describes the usage of the Albumentations API in pre-processing stage to transform the image and labels. The pre-processing transform includes blurring, colour, pixelation, resizes, etc.

4.2.3 Faster R-CNN with ResNet50

```
import torchvision
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor

def create_model(num_classes, pretrained=True):
    # Load Faster RCNN pre-trained model
    model = torchvision.models.detection.fasterrcnn_resnet50_fpn(
        pretrained=pretrained
    )

    # Get the number of input features
    in_features = model.roi_heads.box_predictor.cls_score.in_features
    # new head for the detector with required number of classes
    model.roi_heads.box_predictor = FastRCNNPredictor(
        in_features, num_classes
    )

    return model
```

Listing 4.4: ResNet50 Faster R-CNN with original hyper-parameters

```
!python train_fasterrcnn.py
    --model fasterrcnn_resnet50_fpn
    --config data_configs/colab.yaml
    --project-name resnet50_defaults
    --epochs 30
```

Listing 4.5: Command to train default ResNet50 Faster R-CNN model

The code snippet listing (4.4) from `fasterrcnn_resnet50_fpn.py` in `models` directory creates a Faster R-CNN model with ResNet50 backbone and updates the head to fit the correct number of classes. The ResNet50 backbone includes a Feature Pyramid Network (FPN) neck before the Faster R-CNN head. The FPN takes the single scale of the feature map from the last layer of the backbone and outputs multiple proportionally sized levels feature maps. The command to train the default ResNet50 Faster R-CNN with the `train_fasterrcnn.py` is shown in listing (4.5) with 30 epochs and the `data_config` used in Google Colab.

```
...
anchor_sizes = ((16,), (32,), (64,), (128,), (256,))
aspect_ratios = ((0.5, 1.0, 1.5),) * len(anchor_sizes)

anchor_generator = AnchorGenerator(
    sizes=anchor_sizes,
    aspect_ratios=aspect_ratios
)

roi_pooler = torchvision.ops.MultiScaleRoIAlign(
    featmap_names=['0', "1", "2", "3"],
    output_size=7,
    sampling_ratio=2
)

# put the pieces together inside a FasterRCNN model
model = FasterRCNN(
    backbone,
    num_classes=num_classes,
    rpn_anchor_generator=anchor_generator,
    box_roi_pool=roi_pooler
)

return model
```

Listing 4.6: ResNet50 Faster R-CNN with custom hyper-parameters

```
!python train_fasterrcnn.py
    --model fasterrcnn_resnet50_custom
    --config data_configs/colab.yaml
    --project-name fasterrcnn_resnet50_custom
    --epochs 30
```

Listing 4.7: Command to train custom ResNet50 Faster R-CNN model

Similarly, the code snippet in listing (4.7) from `fasterrcnn_resnet50_custom.py` creates a Faster R-CNN model with ResNet50 backbone but with custom hyper-parameters such as the Anchor Generator in the Region Proposal Network and the ROI pooler.

4.2.4 Faster R-CNN with VGG16

```
...
# load the pretrained VGG16 backbone
vgg16_model = models.vgg16(pretrained=True)
backbone = vgg16_model.features
backbone.out_channels = 512

anchor_generator = AnchorGenerator(
    sizes=((16, 32, 64, 128, 256),),
    aspect_ratios=((0.5, 1.0, 2.0),)
)

# Feature maps to perform RoI cropping.
roi_pooler = torchvision.ops.MultiScaleRoIAlign(
    featmap_names=['0', "1", "2", "3"],
    output_size=7,
    sampling_ratio=2
)

# Final Faster RCNN model.
model = FasterRCNN(
    backbone=backbone,
    num_classes=num_classes,
    rpn_anchor_generator=anchor_generator,
    box_roi_pool=roi_pooler
)

return model
```

Listing 4.8: VGG16 Faster R-CNN with custom hyper-parameters

```
python train_fasterrcnn.py
    --model fasterrcnn_vgg16
    --config data_configs/colab.yaml
    --project-name fasterrcnn_vgg16
    --epochs 30
```

Listing 4.9: Command to train custom ResNet50 Faster R-CNN model

The code snippet listing (4.8) from `fasterrcnn_vgg16.py` creates a Faster R-CNN model with a VGG16 backbone. The `backbone.out_channels` is set to 512 due to the VGG16 final layer size. Similarly, it uses a smaller anchor size for the `AnchorGenerator` compared to the default anchor sizes.

4.2.5 Faster R-CNN with MobileNetV3

```
...
# load the pretrained mobilenet backbone
```

```
mobilenetv3large = models.mobilenet_v3_large(pretrained=True)
backbone = mobilenetv3large.features
backbone.out_channels = 960

anchor_generator = AnchorGenerator(
    sizes=((16, 32, 64, 128, 256),),
    aspect_ratios=((0.5, 1.0, 2.0),)
)

roi_pooler = torchvision.ops.MultiScaleRoIAlign(
    featmap_names=['0'],
    output_size=7,
    sampling_ratio=2
)

# Final Faster RCNN model.
model = FasterRCNN(
    backbone=backbone,
    num_classes=num_classes,
    rpn_anchor_generator=anchor_generator,
    box_roi_pool=roi_pooler
)
return model
```

Listing 4.10: MobileNetV3 Faster R-CNN with custom hyper-parameters

```
python train_fasterrcnn.py
    --model fasterrcnn_mobilenetv3_large
    --config data_configs/colab.yaml
    --project-name final_fasterrcnn_mobilnetv3_large
    --epochs 30
```

Listing 4.11: Command to train custom MobileNetV3 Faster R-CNN model

The code snippet in listing (4.10) from `fasterrcnn_mobilenetv3_large.py` creates a Faster R-CNN model with MobileNetV3 backbone. The `AnchorGenerator` anchor sizes are the same as the ResNet50 custom and VGG16. Listing (4.11) is the terminal command to train a MobileNetV3 Faster R-CNN model with 30 epochs.

4.3 Facial Landmark Localisation

The facial landmark localisation uses the PyTorch library similarly to the face detection in section (4.2) with ResNet18 as a backbone and a custom head.

4.3.1 Training

The `train_landmarks.py` script file in the directory in figure (4.2) follows a similar training flowchart to the `train_fasterrcnn.py` in figure (4.3). To train the model, the training script has an additional `loss` argument to select between $\ell 1$ and $\ell 2$ loss. However, the JD-landmark-mask dataset has no facial landmark annotations for the validation set, the PyTorch `random_split` function is used to split the 90:10 ratio for the training dataset into train and validation sets.

4.3.2 Preprocessing

The pre-processing transformation functions are implemented separately using `imutils` for image processing, ratios, and trigonometry calculation to process landmark annotation transforms.

```
landmarks[:,0] = (landmarks[:,0]/image_width) * img_size[0]
landmarks[:,1] = (landmarks[:,1]/image_height) * img_size[1]
```

Listing 4.12: Facial Landmark Localisation Preprocessing Resize Annotations

```
transformation_matrix = torch.tensor([
    [+cos(radians(angle)), -sin(radians(angle))],
    [+sin(radians(angle)), +cos(radians(angle))]
])
```

Listing 4.13: Facial Landmark Localisation Preprocessing Rotate Annotations

```
def crop_face(self, image, landmarks, crops):
    left = int(crops['left'])
    top = int(crops['top'])
    width = int(crops['width'])
    height = int(crops['height'])

    image = TF.crop(image, top, left, height, width)
    .
    landmarks = torch.tensor(landmarks) - torch.tensor([[left, top]])
    return image, landmarks
```

Listing 4.14: Facial Landmark Localisation Preprocessing Crop Face

Listing (4.12) and (4.12) are a few code snippets from the `transforms.py` file for training in facial landmark localisation. An additional crop face transformation is applied as the facial landmark localisation is only trained on the face region instead of the whole image.

4.3.3 CNN with ResNet18

```
...
self.model=models.resnet18(pretrained=pretarined)
self.model.conv1=nn.Conv2d(
    1, 64, kernel_size=7, stride=2, padding=3, bias=False
)
self.model.fc = nn.Linear(
    self.model.fc.in_features, num_classes
)
...
```

Listing 4.15: Facial Landmark Localisation ResNet18 model

The code snippet in listing (4.15) is the ResNet18 CNN model used for facial landmark localisation. Similarly, the head FC layer is updated to fit the number of landmark annotations. Instead of 106 output values, 212 is used as the model has to output 106 x and y values.

4.4 Blurring

```
import cv2
import numpy as np


...
# convert frame to rgb
frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
vis_image = frame.copy()

for annotation in annotations:
    x_min, y_min, x_max, y_max = map(int,annotation['bbox'])
    x_min = np.clip(x_min, 0, x_max)
    y_min = np.clip(y_min, 0, y_max)

    # apply blur onto face
    vis_image[y_min:y_max, x_min:x_max] = cv2.GaussianBlur(
        vis_image[y_min:y_max, x_min:x_max],
        (25,25),
        -1
    )
...
```

Listing 4.16: Uniform Blur on Face

```
...
# convert frame to rgb
frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
vis_image = frame.copy()

for annotation in annotations:
    x_min, y_min, x_max, y_max = map(int,annotation['bbox'])
    x_min = np.clip(x_min, 0, x_max)
    y_min = np.clip(y_min, 0, y_max)

    # apply blur onto face
    vis_image[y_min:y_max, x_min:x_max] = cv2.GaussianBlur(
        vis_image[y_min:y_max, x_min:x_max],
        (25,25),
        -1
    )

    # apply feature based blur
    for landmark_id, (x, y) in enumerate(annotation['landmarks'][:2]):
        eye_x_min, eye_x_max = np.clip(
            int(x)-eye_size,eye_size,None
        ), int(x)+eye_size
        eye_y_min, eye_y_max = np.clip(
            int(y)-eye_size,eye_size,None
        ), int(y)+eye_size
        eye_frame = ori_image[
            eye_y_min:eye_y_max,eye_x_min:eye_x_max
        ].copy()
        eye_frame = cv2.GaussianBlur(eye_frame, (11,11),-1)
        # replace
        vis_image[eye_y_min:eye_y_max,eye_x_min:eye_x_max] = eye_frame
...
```

Listing 4.17: Feature Based Blur on Face

The blurring stage uses mainly the `GaussianBlur` function from OpenCV library (Bradski, 2000). There is no training involved at this stage but hyper-parameter tuning on the kernel sizes to balance the trade-off between privacy and data utility of the face by rerunning the face detection and facial landmark localisation. The code snippet in listing (4.16) applies a uniform blur onto the entire face area. It loops through the detected area from stage 1 results `annotations` variable, clips the annotation to be within the face area and applies Gaussian blur. The code snippet in listing (4.17) extends the uniform blur to feature-based blur by applying a smaller Gaussian kernel to the facial features. There is an additional loop onto the `landmarks` list applying Gaussian blur and replacing the pixels onto the final face area.

4.5 Summary

The implementation of face-masked face blurring reuses many existing open source tools and libraries to develop the proposed design in section (3.2). The training of each model can be done separately with different datasets by updating the `datasets.py`, running the `train_fasterrcnn.py` and `train_landmarks.py` script files. There are several standard annotation formats for datasets, such as COCO, Pascal VOC or custom format requiring custom implementation of loading data for training. The models in the face-masked face blurring application can be swapped accordingly to improve their performance. The structure of the project directory and implementation is important for scaling in the future to add new models. The existence of cloud services such as Google Colab has enabled training complex models with huge datasets possible without owning expensive hardware. The face-masked face blurring application implementation can be trained and used to blur faces in a new dataset recorded in an unconstrained setting to protect participants' privacy.

Chapter 5

Evaluation

5.1 Face Detection

Face detection is used to detect face-masked faces in the frame to be processed in the following stages. The following subsections describe experiments leading to the final solution of the face detection model used in the Face-masked face blurring application. The models are trained to an epoch of 30 due to time constraints as an epoch takes up approximately an hour.

5.1.1 Faster R-CNN ResNet50 Default

The default hyper-parameters from the original paper (Ren et al., 2015) were used with a ResNet50 backbone as the initial experiment. The `AnchorGenerator` used in the Region Proposal Network (RPN) has anchor sizes of $32, 64, 128, 256, 512$ and the aspect ratio of $0.5, 1.0, 2.0$. The RPN proposes such region of interest to the Faster R-CNN classifier.

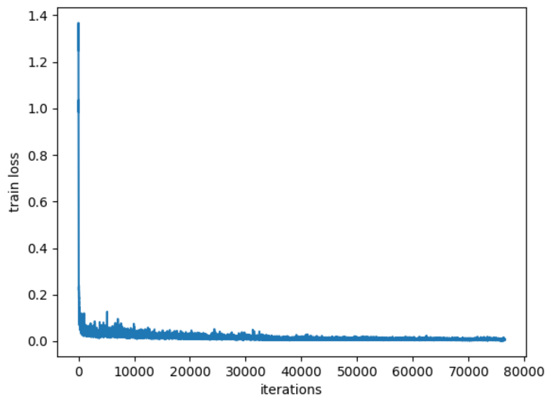


Figure 5.1: Faster R-CNN ResNet50 Default Training Loss

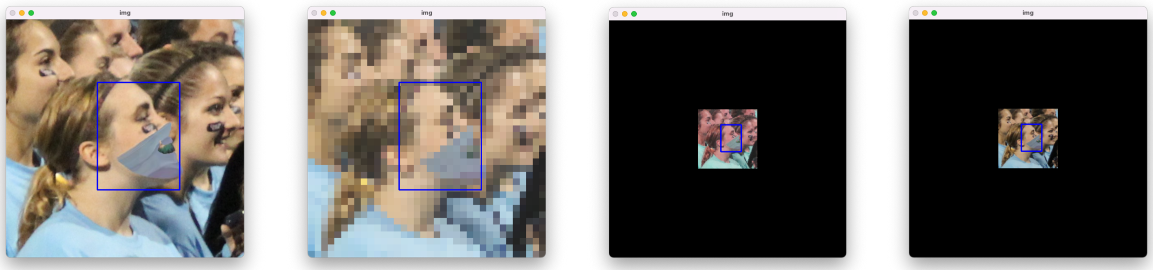

(a) Original (b) Downscale (c) Resize and Pad (d) Colour Jitter

Figure 5.2: JD-landmark-mask dataset Image with preprocessing step

Figure (5.1) is the training loss plot of the model, it has converged to a loss value 0.006. The model achieved a mAP@IOU[0.5] score 0.544, mAP@IOU[0.75] score 0.541 and mAP@IOU[0.5:0.95] score 0.514 on the validation dataset. These mAP scores are good, considering that the training was only 30 epochs. Similar method using Faster R-CNN in the COCO dataset achieves only 0.362 for mAP@IOU[0.5:0.95], 0.591 for mAP@IOU[0.5] and 0.39 for mAP@IOU[0.75] (Redmon and Farhadi, 2018). The average FPS for inference on GPU is 19.831. However, the model fails to make any face detection on the children's dataset video footage. The model is over-fitted to the JD-landmark-mask dataset, and it fails to generalise on other datasets like the children dataset.

5.1.2 Faster R-CNN ResNet50 Default with Pre-Processing

The same default hyper-parameters for the ResNet50 Faster R-CNN similar to section (5.1.1) were used with an additional pre-processing step on the dataset before training. Following the principles of transfer learning where a model trained on a different dataset can be applied to another application as long as it is in the same domain. The training dataset has high-quality images and faces taking up the frame, while the children's dataset is low-quality videos with faces far away from the camera.

Figure (5.2) is an example image with the pre-processing techniques applied, and it is much more similar to the video footage in figure (3.2).

Figure (5.3) is the training loss plot of the model with pre-processing. The loss value after 30 epochs is 0.030. The evaluation on the validation dataset has a mAP@IOU[0.5] score 0.539, mAP@IOU[0.75] score 0.537 and mAP@IOU[0.5:0.95] score 0.489. The training loss is higher with pre-processing, and the training loss plot is noisier. There is a slight drop in the mAP score with pre-processing as the dataset becomes harder due to an increase in various faces with different sizes, blur, etc. Figure (5.4) are the model's prediction on the children dataset. The model can make some detection on the video footage but have poor bounding box fitting. The default hyper-parameters for the RPN

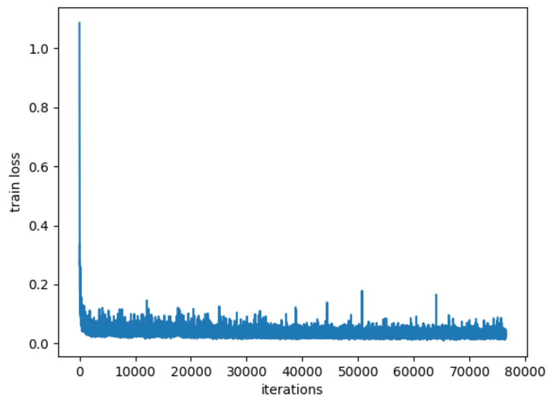


Figure 5.3: Faster R-CNN ResNet50 Default with Preprocessing Training Loss

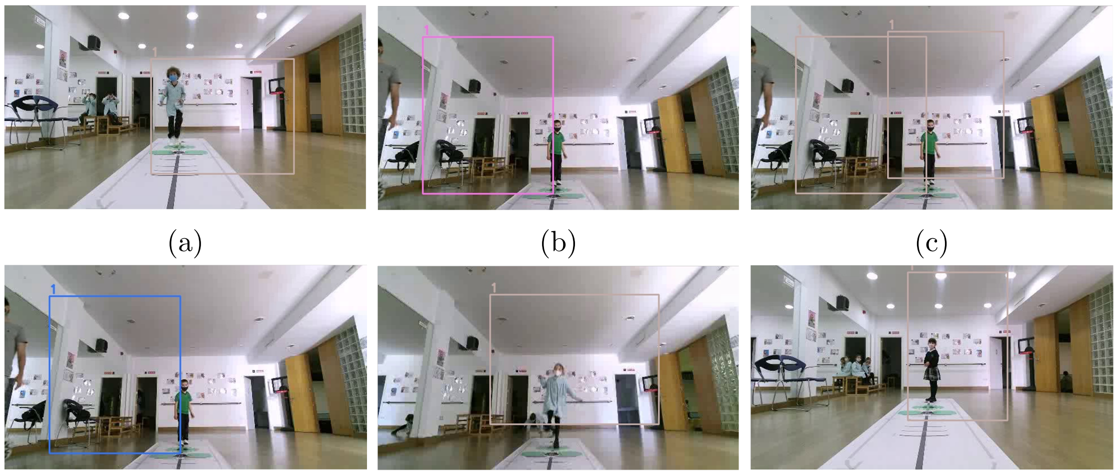


Figure 5.4: Faster R-CNN with Default Parameters and Preprocessing Predictions on Video Footages

propose large region of interest. The original paper (Ren et al., 2015) dealt with objects taking up half the frame sizes but is not in the children dataset case.

5.1.3 Faster R-CNN ResNet50 Custom Hyper-parameters

The hyper-parameters of the RPN in ResNet50 Faster R-CNN are updated to propose smaller regions of interest with anchor sizes of $16, 32, 64, 128, 256$ and the aspect ratio of $0.5, 1.0, 1.5$ to increase the bounding box fit. The change in aspect ratio is because faces are rectangular, and face angles may be vertical or horizontal.

Figure (5.6c) is the training loss plot of the ResNet50 Faster R-CNN with custom hyper-parameters for the RPN. The loss value after 30 epochs is 0.0195. The evaluation

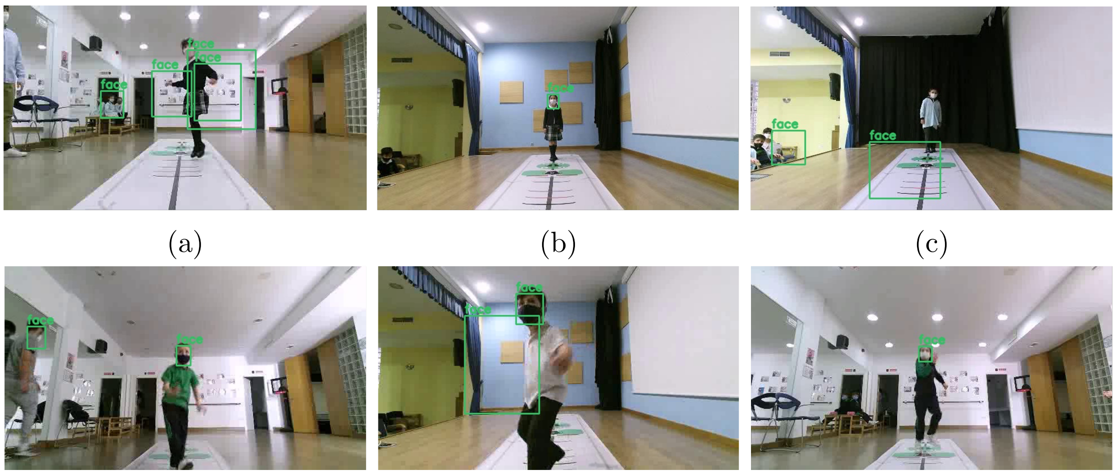

(a) (b) (c) (d) (e) (f) Resize and Pad

Figure 5.5: Faster R-CNN with Custom Hyper-parameters Predictions on Video Footages

Average Precision	**Faster R-CNN Feature Extractors**		
	MobileNetV3	**VGG16**	**ResNet50**
IOU = 0.5:0.95	0.420	0.472	0.492
IOU = 0.5	0.468	0.520	0.541
IOU = 0.75	0.465	0.517	0.538

Table 5.1: Mean Average Precision Results After 30 Epochs

on the validation dataset has a mAP@IOU[0.5] score 0.541, mAP@IOU[0.75] score 0.538 and mAP@IOU[0.5:0.95] score 0.492. The training loss is lower with the smaller proposal region of interest from the RPN, and the validation mAP score is higher than the default hyper-parameters. Figure (5.5) are the results of the model's prediction on the frames of children's dataset video footages. The prediction uses a threshold of 0.3, resulting in some false positives where it detects for faces in no faces region. The bounding box predicted by the model has a better fitting than the default hyper-parameters in section (5.1.2). Small objects are a challenge for object detection techniques, and the paper (Eggert et al., 2017) focuses on Faster R-CNN tuning, especially the RPN.

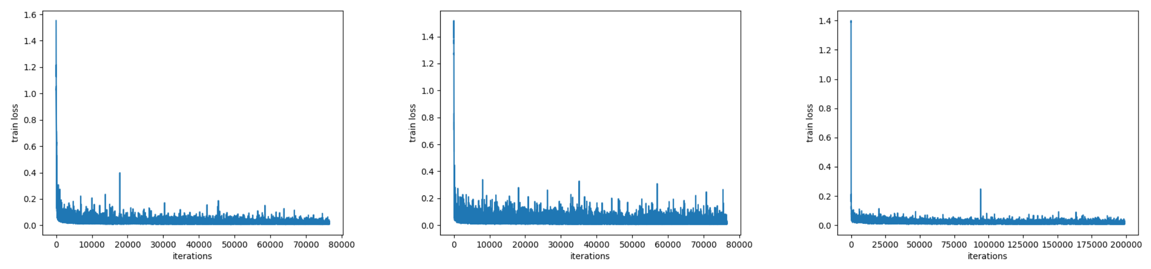
5.1.4 Faster R-CNN with Varying Feature Extractors

The feature extractor backbones mentioned in figure (4.1): ResNet50, VGG16 and MobileNetV3 are compared with the Faster R-CNN network. These models use preprocessing and custom hyper-parameters in the previous experiments (5.1.2) and (5.1.3).

Table (5.1) and (5.2) are the results of the Faster R-CNN mean Average Precision (mAP) and summary of the models. The MobileNetV3 backbone uses the large version as shown in listing (4.10). The small and large versions of MobileNetV3 were publicly

Metrics	**Faster R-CNN Feature Extractors**		
	MobileNetV3	**VGG16**	**ResNet50**
Training Loss	0.0164	0.0185	0.0191
No of Parameters	60,569,221	43,863,957	41,299,161
Size (MB)	727.4	527.1	495.2
Inference CPU (FPS)	0.755	0.105	0.132
Inference GPU (FPS)	31.541	22.857	19.831

Table 5.2: Faster R-CNN Varying Feature Extractor Summary

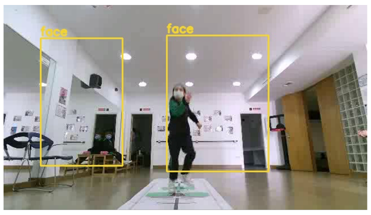


(a) MobileNetV3 (b) VGG16 (c) ResNet50

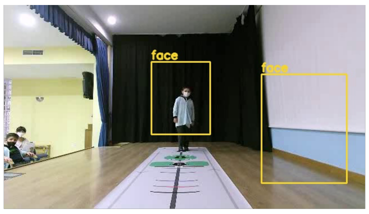
Figure 5.6: Faster R-CNN Varying Feature Extractor Training Loss

released in (Howard et al., 2019) for low and high resources usage. The MobileNetV3 Faster R-CNN model has the lowest training loss, but the mAP scores perform the worst when evaluated with the validation set after training. It is the largest model with the most number of parameters and memory size and is over-fitted to the training data resulting in poor performance in unseen data. With a larger number of parameters, the complexity of the model increases, enabling it to memorise the training data. The use of depth-wise convolutional layer and piece-wise convolutional layer in MobileNetV3 targeting low-powered resources such as mobile devices enables the model to perform the fastest in both CPU and GPU inference. The ResNet50 Faster R-CNN model has the best mAP scores after 30 epochs of training. It has the least number of parameters and memory size but performs the lowest for inference on GPU and CPU due to the operations in the backbone. The VGG16 Faster R-CNN model achieves an mAP score between the MobileNetV3 and ResNet50. Similarly, the model size is between the other two models, but the inference speed on GPU and CPU is slightly faster than ResNet50 but much slower than MobileNetV3. Training the models with more epochs may result in different results as 30 epochs are considered low. Other models such as the pre-trained RetinaFace model was trained with 250 epochs. The trained models have to be generalised to work with the children dataset, but that dataset has no annotations. Instead, a few frames were selected and manually evaluated.

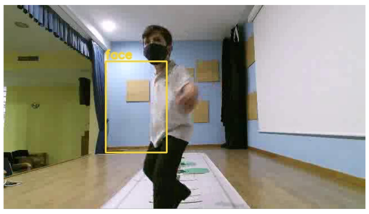
Figure (5.7) are the results from the MobileNetV3 model on the frames of video



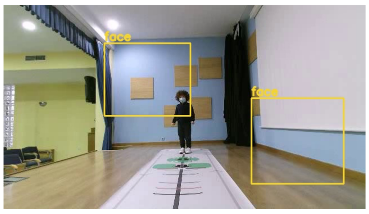
(a)



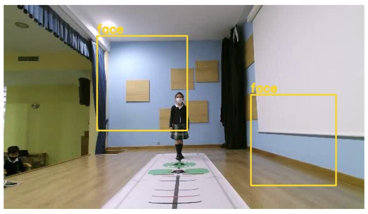
(b)



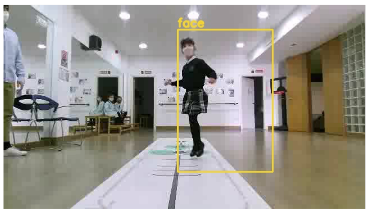
(c)



(d)

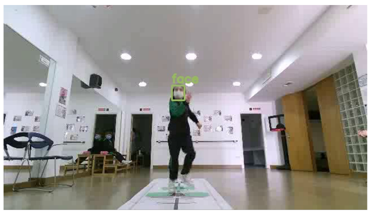


(e)

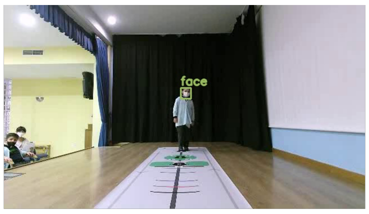


(f)

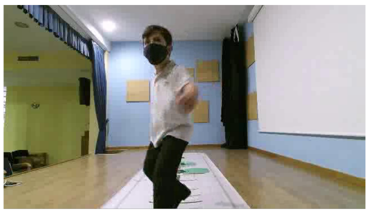
Figure 5.7: Faster R-CNN MobileNetV3 Predictions on Video Footages



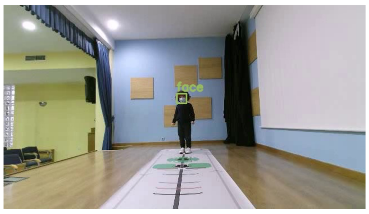
(a)



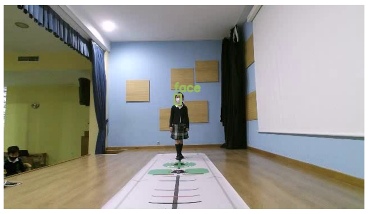
(b)



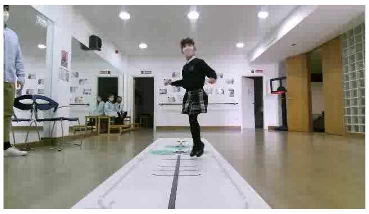
(c)



(d)



(e)



(f)

Figure 5.8: Faster R-CNN VGG16 Predictions on Video Footages

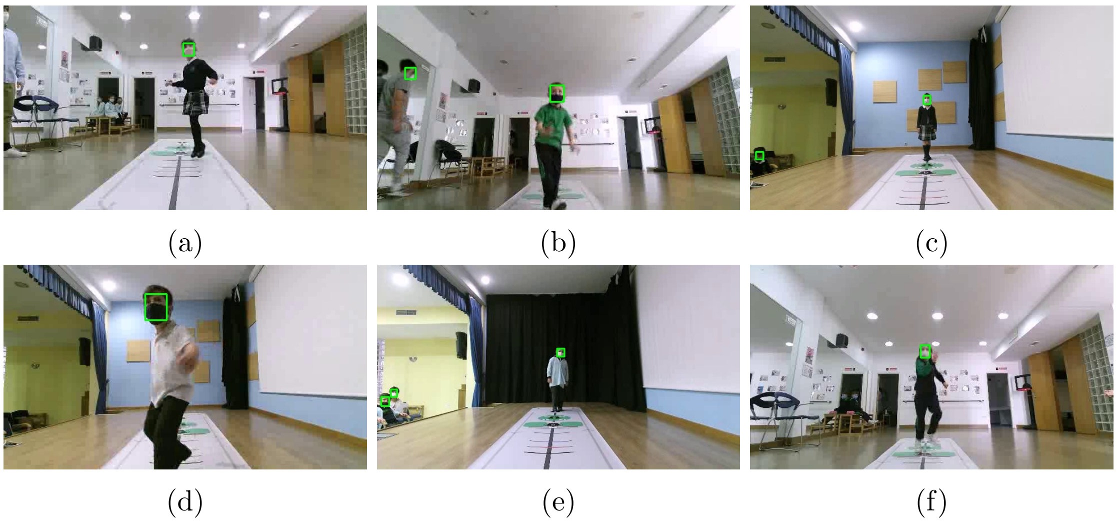
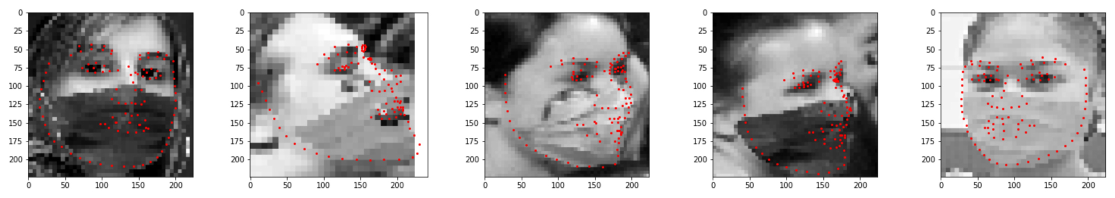


Figure 5.9: RetinaFace Predictions on Video Footages

footage. The bounding boxes predicted by the model have poor fitting even with the smaller anchor sizes and aspect ratio. It fails to understand the features in the frame of the video footage. Figure (5.8) are the results from the VGG16 model on the frames of video footage. Even with a threshold of 0.3, the model is not making any false positives but has false negatives where it fails to detect a face. The results from the ResNet50 model on the frames of video footage were described previously in section (5.1.3). Based on the mAP scores on the validation dataset after training 30 epochs and the evaluation of the frames of video footages from the children dataset, the selected feature extractor for the Faster R-CNN model is ResNet50.

5.1.5 Pre-trained RetinaFace Model

Figure (5.9) is the results of a pre-trained RetinaFace model found in section (2.2.2) applied onto the children dataset video footages. It can detect most of the face-masked faces in the video footage but fails on certain frames due to motion blur when children are performing quick movements, poor lighting conditions, pixelation of the frame, etc. Interestingly when the model was evaluated on the validation dataset of JD-landmark-mask dataset with preprocessing, it has a mAP@IOU[0.5] score 0.080, mAP@IOU[0.75] score 0.069 and mAP@IOU[0.5:0.95] score 0.059. The RetinaFace model performs poorly on the JD-landmark-mask dataset with pre-processing but can detect faces on the video footage. The training data affects the performance on test data as shown in section (5.1.1) and (5.1.2) with the presence of preprocessing. As the goal of this face-masked face blurring application is to blur faces in the children dataset, the WIDER FACE dataset (Yang et al., 2016) used to train the RetinaFace can also be used to train the Faster R-



(a) (b) (c) (d) (e)

Figure 5.10: Facial Landmark Training Images

CNN model. It is a much larger dataset, but RetinaFace added annotations manually for the facial landmarks to train the model. The Faster R-CNN model can use the entire WIDER FACE dataset for training as it does not require the facial landmarks annotation.

5.2 Facial Landmark Localisation

The facial landmark localisation processes the face detection output in stage 1 in figure (4.1). Figure (5.10) are the training images for the facial landmark localisation models. The training images use only the cropped faces from the bounding box annotation in the dataset, and it does not use the entire frame such as the figure (3.3). The training is run for 100 epochs as an epoch is a lot faster, taking approximately 2 minutes due to the smaller feature extractor, resulting in faster forward and backward propagation.

5.2.1 ResNet18 CNN With L2 Loss

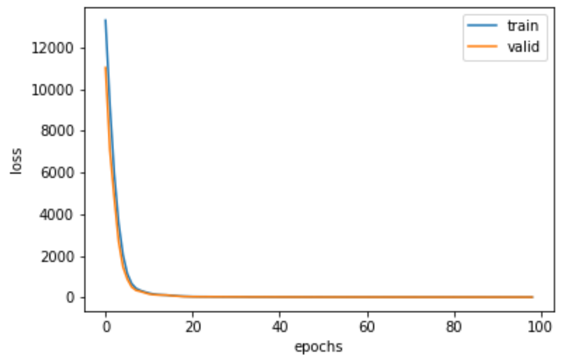


Figure 5.11: ResNet18 CNN Training and Validation Loss with L2 Loss

Figure (5.11) is the training and validation loss plot of the ResNet18 CNN trained with L2 loss function. At the start of the training, both the losses were high even though the ResNet18 feature extractor was using pre-trained weight. The pre-trained weight

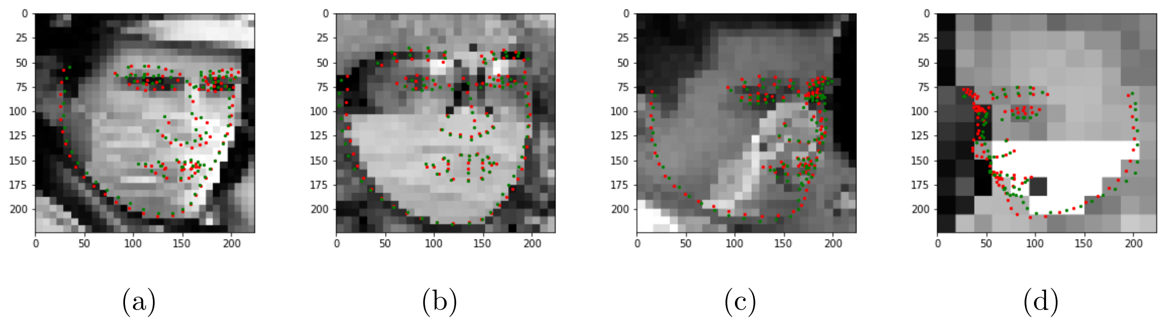


Figure 5.12: ResNet18 CNN L2 Model Predictions on Validation dataset

of the backbone [1] was trained on ImageNet containing annotations of photographs with many classes and not face region, and it was not able to extract features from the face region. After 100 epochs, it converges to a training L2 loss of 11.5677 and validation L2 loss of 17.6659. The model was generalising well between 15 to 50 epochs as the validation loss was lower than the training loss. After 50 epochs, the model was over-fitting as the training loss was decreasing while validation loss hovers between 16 to 18 loss. Figure (5.12) are the prediction results on the validation dataset from the ResNet18 CNN trained with L2 loss where the green points are the ground truth, and red points are the predictions. Training the facial landmark localisation model with a face-masked image enables it to predict with face masks. The nose and mouth facial features are occluded, but the model can make predictions of the features.

5.2.2 ResNet18 CNN With L1 Loss

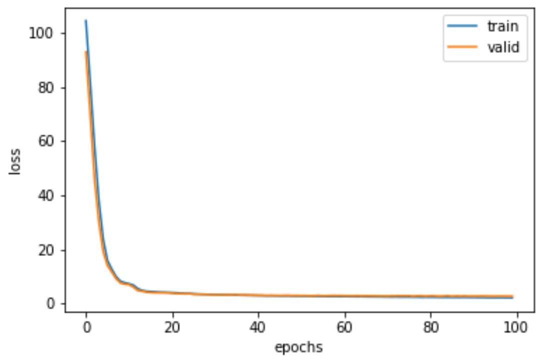
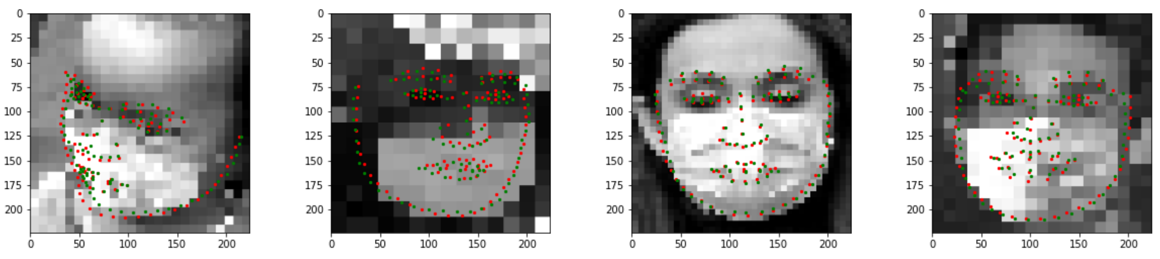


Figure 5.13: ResNet18 CNN Training and Validation Loss with L1 Loss

Figure (5.13) is the training and validation loss plot of the ResNet18 CNN trained with L1 loss function. Similar to the model trained with L2 loss in section (5.2.1), the model

[1]`https://pytorch.org/vision/stable/models.html`



(a) (b) (c) (d)

Figure 5.14: ResNet18 CNN L1 Model Predictions on Validation dataset

	L1 Loss	**L2 Loss**
ResNet18 CNN with L2 Loss	2.5398	12.0285
ResNet18 CNN with L1 Loss	2.5633	12.0480

Table 5.3: ResNet18 CNN Loss Function comparison

uses the same pre-trained ResNet18 backbone. After 100 epochs, the model converges to a training L1 loss of 2.2078 and validation L1 loss of 2.7523. The L1 loss values are much smaller compared to L2 loss.

Table (5.3) is a comparison between the ResNet18 CNN trained with L1 and L2 loss functions, and the models are evaluated on the validation dataset. The results are slightly different than section (5.2.1) and (5.2.2) because of the randomisation in the preprocessing step. Both the L1 and L2 losses are calculated for comparison. The model trained with L2 loss has a slightly lower score. The L2 loss is sensitive to outliers (Feng et al., 2018) as the squared error as shown in equation (3.6) will be huge. Instead, the sensitivity of L2 loss enables the model to better learn the different face orientations compared to the L1 loss. The ResNet18 CNN trained with L2 loss is selected for the facial landmark localisation stage.

5.2.3 Faster R-CNN ResNet50 + ResNet18 CNN

Figure (5.15) is the result of the trained Faster R-CNN with ResNet50 feature extractor face detector and ResNet18 CNN facial landmark localisation on the video footage. The face detector uses a threshold of 0.3. The facial landmark localisation is able to detect the facial features when the face detector correctly detects a face. When the face detector detects incorrect regions, such as false positives or poor bounding box fitting, the facial landmark localisation model processes the region and predicts incorrect points of the facial feature. As the trained facial landmark localisation model has never seen any data outside of the face region.

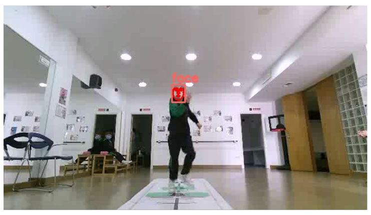
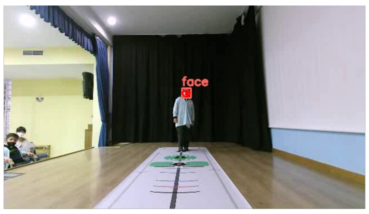
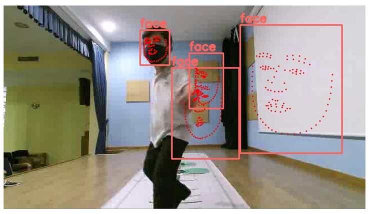
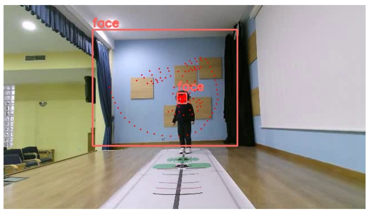
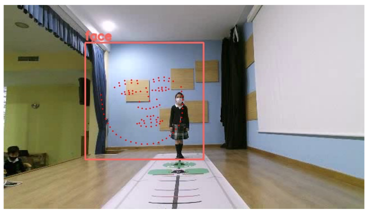
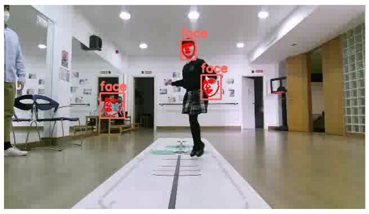

(a)

(b)

(c)

(d)

(e)

(f)

Figure 5.15: Faster R-CNN ResNet50 and ResNet18 CNN results

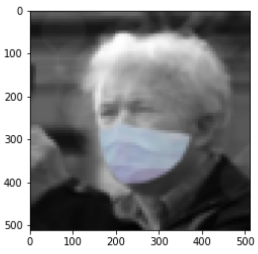
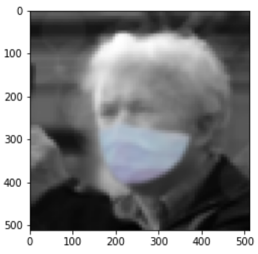
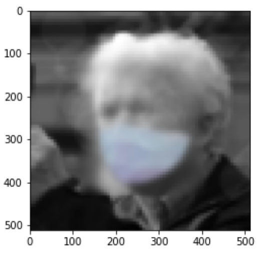
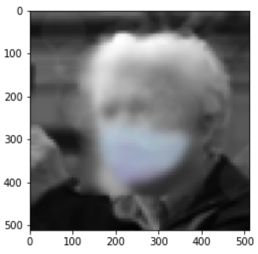

(a) (5,5)

(b) (25,25)

(c) (45,45)

(d) (65,65)

Figure 5.16: Uniform Blur Applied Onto Validation Dataset

5.3 Blurring

The experiments related to blurring use the selected ResNet50 Faster R-CNN and ResNet18 CNN trained models. The first experiment is a uniform blur, with the same blur intensity applied to the entire face. It uses the bounding box annotation from the dataset to apply different Gaussian kernel sizes and evaluate the blurred dataset with the trained models. The second experiment is a feature-based blur, where a different level of intensity of blur is applied to the face. Similarly, it uses the annotations from the dataset and trained models for evaluation.

5.3.1 Uniform Blur

Figure (5.16) are examples of different Gaussian Kernel sizes applied to the validation dataset using the bounding box annotation. As the size of the Gaussian Kernel increases, the intensity of the blur increases.

		Average Precision		
		IoU = 0.5	**IoU = 0.75**	IoU = 0.5:0.95
Gaussian Kernel Sizes	**(5,5)**	0.5331	0.5296	0.4850
	(15,15)	0.5296	0.5270	0.4790
	(25,25)	0.5109	0.5086	0.4614
	(35,35)	0.5072	0.5030	0.4586
	(45,45)	0.5042	0.5004	0.4545
	(55,55)	0.4899	0.4829	0.4375
	(65,65)	0.4841	0.4822	0.4329

Table 5.4: Results of ResNet50 Faster R-CNN mAP with Uniform Blur

		L2 Loss
Gaussian Kernel Sizes	**(5,5)**	13.4391
	(15,15)	13.9086
	(25,25)	17.0735
	(35,35)	25.0273
	(45,45)	38.9237
	(55,55)	60.0098
	(65,65)	91.9704

Table 5.5: Results of ResNet18 CNN l2 loss with Uniform Blur

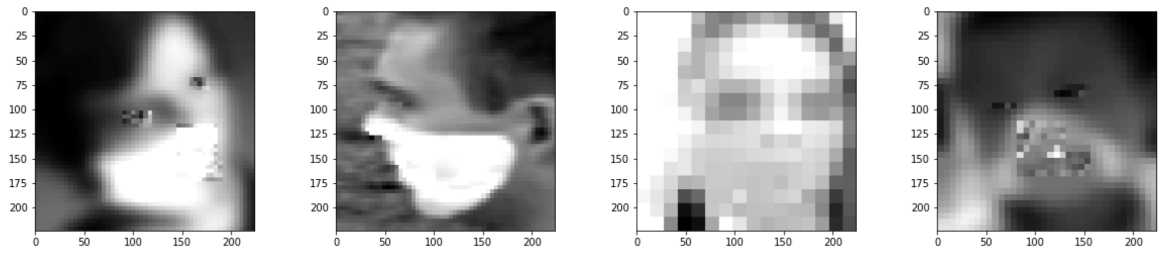
Table (5.4) is the results of varying Gaussian kernel sizes with the mAP scores of trained ResNet50 Faster R-CNN model. The face detection model is still able to detect faces with uniform blur. A larger Gaussian kernel size results in more blur, decreasing the performance of the face detection model.

Table (5.5) is the l2 loss result of trained ResNet18 CNN facial localisation model with varying Gaussian kernel sizes. Similarly, the performance of the facial localisation decreases as the l2 loss error increases when a larger Gaussian kernel is used. The feature extractor of the model is not able to extract the obfuscated facial features in the face.

5.3.2 Feature Based Blur

Figure (5.17) are examples of the feature-based blur of the validation dataset with a larger Gaussian kernel applied to the face region and a smaller Gaussian kernel applied to the facial features using the landmarks annotation.

Table (5.6) is the results of trained ResNet50 Faster R-CNN model with feature-based blur. The performance on the feature-based blur is similar to uniform blur, with some Gaussian kernel sizes performing slightly better and slightly poorer due to the randomisation in the split and pre-processing. The JD-landmark-mask dataset has no landmark annotation on the validation dataset; hence the training dataset is split into



(a) (b) (c) (d)

Figure 5.17: Feature Based Blur Applied Onto Validation Dataset

		Average Precision		
		IoU = 0.5	**IoU = 0.75**	**IoU = 0.5:0.95**
Gaussian Kernel Sizes	**(5,5) , (3x3)**	0.5254	0.5254	0.4888
	(15,15) , (3x3)	0.5135	0.5131	0.4772
	(25,25) , (3x3)	0.5002	0.5000	0.4651
	(35,35) , (3x3)	0.5007	0.5003	0.4626
	(45,45) , (3x3)	0.4908	0.4895	0.4518
	(55,55) , (3x3)	0.4835	0.4823	0.4422
	(65,65) , (3x3)	0.4689	0.4680	0.4288

Table 5.6: Results of ResNet50 Faster R-CNN mAP with Feature Based Blur

training and validation sets. When performing the evaluation, the validation set may be different for each iteration.

Table (5.7) is the results of the trained ResNet18 CNN facial localisation model with varying Gaussian kernel sizes for the face region and a fixed (3,3) Gaussian kernel on the facial features. Figure (5.18) is the plot of the ResNet18 CNN L2 loss of uniform blur and feature-based blur. The Gaussian kernel sizes of (5,5) and (15,15) are very similar for both types of blur, with feature-based blur performing slightly poorer. With larger Gaussian kernel sizes of at least (25,25), the model's performance on feature-based blur outperforms the facial landmark localisation on uniform blur. The loss of uniform blur

		L2 Loss
Gaussian Kernel Sizes	**(5,5) , (3x3)**	13.5110
	(15,15) , (3x3)	14.1677
	(25,25) , (3x3)	15.9813
	(35,35) , (3x3)	19.1821
	(45,45) , (3x3)	23.5520
	(55,55) , (3x3)	28.0473
	(65,65) , (3x3)	33.1247

Table 5.7: Results of ResNet18 CNN l2 loss with Feature Based Blur

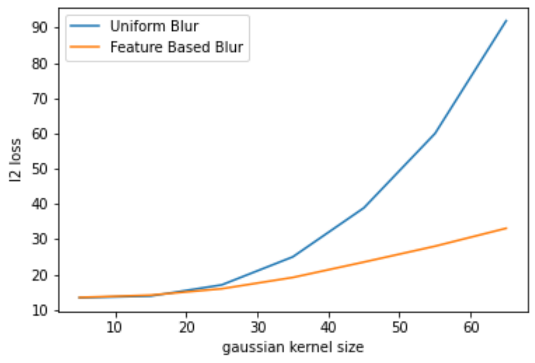


Figure 5.18: ResNet18 CNN L2 Loss Uniform vs Feature Based Blur

increases much greater than a feature-based blur. The feature-based blur can retain core references to the facial features while blurring the face of the participants.

Chapter 6

Conclusions & Future Work

6.1 Conclusion

The trained models and selected computer vision techniques for the proposed design of the face-masked face blurring application showed promising results in blurring face-masked faces in an image or video footage. It can be used to anonymise face-masked faces, protecting participants' privacy while maintaining the data utility of the facial features. The solution detects face and facial landmarks and then applies a feature-based blurring in an automated fashion. It requires annotated datasets to train the deep learning models in face detection and facial landmark localisation. Furthermore, the proposed design is modular, enabling more choices of models and datasets for training and inference.

The dissertation has achieved its aim of developing a face-masked face blurring application by completing the objectives listed in section (1.3). The implemented solution met most of the design requirements in section (3.1) by processing the frames with an average of 18 frames per second with the ResNet50 Faster R-CNN face detection model, ResNet18 CNN facial landmark localisation and Gaussian blur. The output from the face-masked face blur application retains data utility of the facial features as the face detectors and facial landmark localisation was still able to detect the features with a slight drop in performance. The high recall of the face detection was not achieved in the design requirements as the trained ResNet50 Faster R-CNN was only achieving between 0.40 to 0.50 mAP due to limited resources to train a huge number of epochs.

6.2 Future Work

The current dissertation work only explored training Faster R-CNN models with 30 epochs. Instead, a larger number of epochs can be explored and evaluated. A huge number of parameters in the model require much more training, such as the MobileNetV3 large feature extractor. The WIDER Face dataset used to train RetinaFace and worked with the video footage can be used to train the face detector. It is a much larger dataset with clear and occluded faces than the JD-landmark-mask dataset. Similarly, the facial landmark localisation mode can be trained with different dataset types to increase its performance. The CNN model can vary the number of landmark outputs to only the number of facial features. The JD-landmark-mask has 106 facial landmark points in the dataset, but only a subset of it was used containing eyes, nose and mouth. By focusing on these specific facial features may result in better facial landmark localisation performance as the training can focus on these areas. Generative Adversarial Networks (GAN) can be used to protect the privacy of the participants by maintaining all data utility of the face region. It is another obfuscation technique that replaces original faces with realistic fake faces. There are little to no GAN approaches to obfuscating face-masked faces.

Bibliography

Al-Jubouri, A. A., Ali, I. H., and Rajihy, Y. (2020). Generating 3d dataset of gait and full body movement of children with autism spectrum disorders collected by kinect v2 camera. *Compusoft*, 9(8):3791–3797.

Arif, Y. R., Putrada, A. G., and Pahlevi, R. R. (2021). An evaluation of a modified haar-like features based classifier method for face mask detection in the covid-19 spread prevention. In *2021 International Symposium on Electronics and Smart Devices (ISESD)*, pages 1–5. IEEE.

Asif, S., Wenhui, Y., Tao, Y., Jinhai, S., and Amjad, K. (2021). Real time face mask detection system using transfer learning with machine learning method in the era of covid-19 pandemic. In *2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 70–75. IEEE.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools.*

Chandrasekaran, V., Gao, C., Tang, B., Fawaz, K., Jha, S., and Banerjee, S. (2020). Face-off: Adversarial face obfuscation. *arXiv preprint arXiv:2003.08861.*

Chavda, A., Dsouza, J., Badgujar, S., and Damani, A. (2021). Multi-stage cnn architecture for face mask detection. In *2021 6th International Conference for Convergence in Technology (I2CT)*, pages 1–8. IEEE.

Chowdary, G. J., Punn, N. S., Sonbhadra, S. K., and Agarwal, S. (2020). Face mask detection using transfer learning of inceptionv3. In *International Conference on Big Data Analytics*, pages 81–90. Springer.

Croft, W. L., Sack, J.-R., and Shi, W. (2021). Differentially private facial obfuscation via generative adversarial networks. *Future Generation Computer Systems.*

De Cristofaro, E. (2020). An overview of privacy in machine learning. *arXiv preprint arXiv:2005.08679.*

Deb, D., Zhang, J., and Jain, A. K. (2020). Advfaces: Adversarial face synthesis. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–10. IEEE.

Deng, J., Guo, J., Ververas, E., Kotsia, I., and Zafeiriou, S. (2020). Retinaface: Single-shot multi-level face localisation in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5203–5212.

Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., and Zafeiriou, S. (2019). Retinaface: Single-stage dense face localisation in the wild. *arXiv preprint arXiv:1905.00641*.

Ding, Y., Li, Z., and Yastremsky, D. (2021). Real-time face mask detection in video data. *arXiv preprint arXiv:2105.01816*.

Eggert, C., Brehm, S., Winschel, A., Zecha, D., and Lienhart, R. (2017). A closer look: Small object detection in faster r-cnn. In *2017 IEEE international conference on multimedia and expo (ICME)*, pages 421–426. IEEE.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338.

Fan, X. and Jiang, M. (2021). Retinafacemask: A single stage face mask detector for assisting control of the covid-19 pandemic. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 832–837. IEEE.

Feng, Z.-H., Kittler, J., Awais, M., Huber, P., and Wu, X.-J. (2018). Wing loss for robust facial landmark localisation with convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2235–2245.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Goldsteen, A., Ezov, G., Shmelkin, R., Moffie, M., and Farkash, A. (2020). Anonymizing machine learning models. *arXiv preprint arXiv:2007.13086*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324.

Hu, H., Salcic, Z., Sun, L., Dobbie, G., Yu, P. S., and Zhang, X. (2021). Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*.

Jourabloo, A., Ye, M., Liu, X., and Ren, L. (2017). Pose-invariant face alignment with a single cnn. In *Proceedings of the IEEE International Conference on computer vision*, pages 3200–3209.

Khan, M. H., Schneider, M., Farid, M. S., and Grzegorzek, M. (2018). Detection of infantile movement disorders in video data using deformable part-based model. *Sensors*, 18(10):3202.

Knežević, K., Mandić, E., Petrović, R., and Stojanović, B. (2018). Blur and motion blur influence on face recognition performance. In *2018 14th Symposium on Neural Networks and Applications (NEUREL)*, pages 1–5. IEEE.

Kühl, N., Martin, D., Wolff, C., and Volkamer, M. (2020). "healthy surveillance": Designing a concept for privacy-preserving mask recognition ai in the age of pandemics. *arXiv preprint arXiv:2010.12026*.

Langr, J. and Bok, V. (2019). *GANs in action: deep learning with generative adversarial networks*. Manning Publications.

Li, S. Z. and Jain, A. (2015). *Encyclopedia of biometrics*. Springer Publishing Company, Incorporated.

Lin, B., Mu, Y., Fu, Z., Li, C., and Duan, X. (2021). A network for detecting facial features during the covid-19 epidemic. In *Proceedings of the 5th International Conference on Control Engineering and Artificial Intelligence*, pages 141–146.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Liu, J., Feng, Y., and Wang, H. (2021). Facial expression recognition using pose-guided face alignment and discriminative features based on deep learning. *IEEE Access*, 9:69267–69277.

Loey, M., Manogaran, G., Taha, M. H. N., and Khalifa, N. E. M. (2021). Fighting against covid-19: A novel deep learning model based on yolo-v2 with resnet-50 for medical face mask detection. *Sustainable cities and society*, 65:102600.

Marks, P. (2009). Face-blurring technology to protect your cctv privacy. *NEW SCIENTIST*, 201(2693):20.

McKay, B. (2020). What we know about the coronavirus, from symptoms to who is at risk; health officials are worried about a growing number of cases outside china. Name - World Health Organization; Copyright - Copyright 2020 Dow Jones & Company, Inc. All Rights Reserved; Last updated - 2021-09-10.

Missoni, E., Armocida, B., and Formenti, B. (2021). Face masks for all and all for face masks in the covid-19 pandemic: Community level production to face the global shortage and shorten the epidemic. *Disaster medicine and public health preparedness*, 15(1):e29–e33. Date completed - 2021-04-29; Date created - 2020-06-25; Date revised - 2021-05-13; SuppNotes - Erratum In: Disaster Med Public Health Prep. 2021 Feb;15(1):e50 33531100; Last updated - 2021-05-19.

Nagavi, T. C., Sharanya, R., Minchu, S., Kumar, R., et al. (2021). Covid-19 face mask detection using deep learning. In *2021 6th International Conference on Signal Processing, Computing and Control (ISPCC)*, pages 141–145. IEEE.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Popescu, A. B., Taca, I. A., Vizitiu, A., Nita, C. I., Suciu, C., Itu, L. M., and Scafa-Udriste, A. (2022). Obfuscation algorithm for privacy-preserving deep learning-based medical image analysis. *Applied Sciences*, 12(8):3997.

Rab, S., Javaid, M., Haleem, A., and Vaishya, R. (2020). Face masks are new normal after covid-19 pandemic. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, 14(6):1617–1619.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Raynal, M., Achanta, R., and Humbert, M. (2020). Image obfuscation for privacy-preserving machine learning. *arXiv preprint arXiv:2010.10139*.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ren, S., Cao, X., Wei, Y., and Sun, J. (2016). Face alignment via regressing local binary features. *IEEE Transactions on Image Processing*, 25(3):1233–1245.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Sethi, S., Kathuria, M., and Kaushik, T. (2021a). Face mask detection using deep learning: An approach to reduce risk of coronavirus spread. *Journal of Biomedical Informatics*, 120:103848.

Sethi, S., Kathuria, M., and Kaushik, T. (2021b). A real-time integrated face mask detector to curtail spread of coronavirus. *CMES-Computer Modeling in Engineering and Sciences*, pages 389–409.

Sikand, D., Raguru, J. K., and Sharma, V. K. (2021). Alert system for face mask detection using cnn. In *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pages 764–769. IEEE.

Singh, S., Ahuja, U., Kumar, M., Kumar, K., and Sachdeva, M. (2021). Face mask detection using yolov3 and faster r-cnn models: Covid-19 environment. *Multimedia Tools and Applications*, 80(13):19753–19768.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Szeliski, R. (2010). *Computer vision: algorithms and applications.* Springer Science & Business Media.

Talukdar, J., Gupta, S., Rajpura, P., and Hegde, R. S. (2018). Transfer learning for object detection using state-of-the-art deep neural networks. In *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 78–83. IEEE.

Tekli, J., Al Bouna, B., Couturier, R., Tekli, G., Al Zein, Z., and Kamradt, M. (2019). A framework for evaluating image obfuscation under deep learning-assisted privacy attacks. In *2019 17th International Conference on Privacy, Security and Trust (PST)*, pages 1–10. IEEE.

Timelli, L. and Girardi, E. (2021). Effect of timing of implementation of containment measures on covid-19 epidemic. the case of the first wave in italy. *Plos one*, 16(1):e0245656.

Vansh Gupta, R. R. (2021). Face mask detection using mtcnn and mobilenetv2. *International Research Journal of Engineering and Technology (IRJET)*, 08(3):309–312.

Weiss, T. R. (2012). Youtube face-blurring tool looks to protect some privacy. *eWeek*, page 9.

Wilson, P. I. and Fernandez, J. (2006). Facial feature detection using haar classifiers. *Journal of computing sciences in colleges*, 21(4):127–133.

Wu, Y., Yang, F., Xu, Y., and Ling, H. (2019). Privacy-protective-gan for privacy preserving face de-identification. *Journal of Computer Science and Technology*, 34(1):47–60.

Wu, Z., Wang, H., Wang, Z., Jin, H., and Wang, Z. (2020). Privacy-preserving deep action recognition: An adversarial learning framework and a new dataset. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Xiang, M., Liu, Y., Liao, T., Zhu, X., Yang, C., Liu, W., and Shi, H. (2021). The 3rd grand challenge of lightweight 106-point facial landmark localization on masked faces. In *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE.

Yang, H., He, X., Jia, X., and Patras, I. (2015). Robust face alignment under occlusion via regional predictive power estimation. *IEEE Transactions on Image Processing*, 24(8):2393–2403.

Yang, K., Yau, J., Fei-Fei, L., Deng, J., and Russakovsky, O. (2021). A study of face obfuscation in imagenet. *arXiv preprint arXiv:2103.06191.*

Yang, S., Luo, P., Loy, C.-C., and Tang, X. (2016). Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533.

Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503.