Uncovering the Invisible: Improving Face Detection Under Occlusions

Master Artificial Intelligence – Utrecht University

Eleni Veroni
e.veroni@students.uu.nl

8002614

Supervisor
Dr. Ronald Poppe

Second Supervisor
Dr. Albert Ali Salah

Utrecht University
Utrecht, the Netherlands
2023

Acknowledgements

First of all I am very thankful to my thesis supervisor Dr. Ronald Poppe for his continuous guidance and helpful feedback during the whole process. I'd also like to thank my second supervisor Dr. Albert Ali Salah and my daily supervisor Metehan Doyran, as well as the team from UMC Utrecht for their support.

Last but not least, I couldn't forget my family and friends in Greece and the Netherlands who were patient and supported me greatly these months. Without them I wouldn't be here, so thank you.

Contents

1 Introduction

Face detection is an important computer vision task entailing the correct identification of human faces in images and returning their bounding box coordinates. Over the years, face detection has seen extensive research improvements, ranging from traditional non-deep learning approaches using handcrafted features to modern Convolutional Neural Networks (CNNs), which have achieved state-of-the-art performance. Despite significant advancements, detecting faces accurately still has some challenges, particularly when faced with occlusions. Occlusions can be objects obstructing the face such as hats, scarves, hands and combined with variations in poses and image quality, pose significant obstacles for face detection. While face detection is often seen as a solved problem and some algorithms may achieve high performance on specific face datasets, this does not necessarily imply that they are applicable for all tasks and work well under all conditions. Modern implementations come with positive and negative aspects and are applicable to certain scenarios but cannot cover all the use cases. Figure 1 illustrates predictions of a face detector with confidence scores.

Figure 1: Example of face detection from [1]

Face detection is the first step for many applications such as face recognition, alignment, tracking and landmark prediction. Landmarks are anatomical points that correspond to a specific body part, for example a face can have landmarks around the eye regions, the mouth, the nose, and these help accomplish a specific task, like gaze detection or speaker detection.

Occlusions pose significant challenges, as they often lead to failures in detecting the face [3]. The type of occlusions can vary; they can be known (e.g., fixed occlusions around specific facial parts) and unknown, influencing the approach to solving the occlusion problem. Having previous information about their existence and location is beneficial, as it enables the use of visible face regions and potentially allows for occlusion-specific methods to improve detection.

One interesting case of face detection under occlusions is preterm infants. This is a specialized area of research where detection algorithms are developed and applied to preterm infants in neonatal intensive care units (NICUs). Preterm infants are defined as infants with a gestational age of less than 37 weeks. Some occlusion

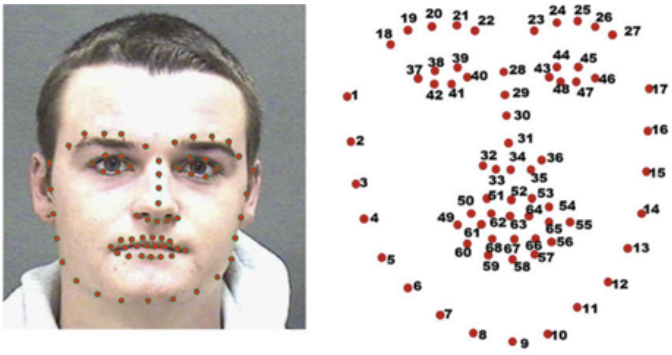

Figure 2: Example of landmark annotations from [2]

information in this case is already known such as the relative location of hats and feeding tubes. However, there is a number of inherent challenges on this task. First, gathering data in NICUs involves technical difficulties, such as maintaining a consistent camera setup, obtaining parental consent for data collection, and handling low-resolution videos with varying occlusions and poses. Infants in NICUs look different, their face does not resemble an adult face. They can wear hats, have medical devices attached to their hands, and experience occlusions due to their own movements or nurses' interventions. Additionally there can be variations in natural lighting throughout the recording, glares around the crib and other objects. Consistent publicly available datasets are almost impossible to obtain, so we will use part of the SLAPI dataset created at UMC Utrecht, a dataset containing videos of preterm infants of various gestational ages. The incubator has two low-cost RGB cameras set up, one on the side and one on the top so at least one of the views is of better quality.

In this thesis, we consider face detection for preterms as an occlusion aware task and make a generalized approach for this problem. We create a two-step pipeline: the first part of our approach is to train a face detector on public datasets with various occlusion levels and evaluate it. Then, we make use of explainability methods for CNNs, which generate heatmaps that highlight the most important regions for the prediction. The idea is that if certain facial parts are occluded, we can still make use of the location information of other parts to localize the face. Thus, we train models using the extracted heatmaps from the original datasets only for the whole face, and evaluate these models on the SLAPI dataset. We aim to determine if, despite the first trained face detector missing a face, we can still detect it using a trained heatmap model.

1.1 Research Goal & Research Questions

Our pipeline consists of two models, the first being the facial parts detector for RGB face images, and the second being the face detector for the heatmaps. Our approach can be broken down into several research aspects:

- Choosing public face datasets that contain some occlusions and variability, from which we can extract bounding boxes for the different facial parts. We decide to use the face and other facial parts, so even if the whole face is missed, there is still some information about its location by localizing

another detected facial part. The question formed here is *which facial parts can we split the face in and what is the models' performance?*
- If the first model misses some faces, we aim to see if they can still be localized by extracting heatmaps using an explainability method and training a model on these heatmaps. Thus we initially need to evaluate *which methods produce the most useful heatmaps and what is their performance on the same datasets as above?*
- Finally, the initial goal in mind was to perform face detection on preterm infants. Thus, the question here is *how does the first face detector perform on frames of infants with various occlusion levels?*. Then, for the frames where the face detection failed, we run the explainability based models and evaluate *whether the use of explainability based models has a positive effect on detecting more faces.*

1.2 Structure

The outline of the thesis is as follows. The 2nd chapter will give a summary of the relevant academic literature on face detection, occluded face detection and touch on explainability methods for deep learning. Chapter 3 will motivate and describe the proposed method to identify occluded faces taking into account specific challenges around preterm infants. The quantitative model results and comparisons of this work will be presented in Chapter 4 including plots and a description of their interpretation. Lastly in Chapter 5 we will discuss conclusions and gained insights as well as future improvements.

2 Related Work

This sections can be categorized into two main parts: In this first part we discuss some of the most prominent object detection algorithms. Then, we address face detection as it can be seen as a subtask of object detection and analyze the research there. Landmark prediction frameworks also employ face detection as a first step thus we give an overview of state of the art work. Later on, emphasis is given on the main topic of this thesis, which is occluded face detection. The focus is mainly on deep learning methods, and more precisely CNNs, as they are the current state of the art and have achieved significant improvement on the task [4]. It should be noted that since facial landmark prediction goes hand in hand with face detection, several works may address both tasks simultaneously. The second part of our pipeline focuses on gradient based explainability algorithms for CNNs, so some will also be described in this section.

2.1 Object Detection

Face detection is a specialized area within the broader field of object detection. Object detection refers to the family of algorithms that take an image and output the class and bounding box coordinates of the objects present. Before analyzing face detection algorithms it is best to see some concepts about general object detection and popular algorithms which can be organized into two main categories: One-Shot and Multi-stage algorithms. One-shot algorithms detect objects in a single step, making them ideal for real-time detection tasks. In contrast, multi-stage algorithms require more steps in the detection process, depending on the specific implementation and have slower performance. While they might take more time to produce results, Multi-stage algorithms often offer increased accuracy, making them suitable for different use cases. The choice between One-Shot and Multi-stage algorithms depends on the specific requirements of the task at hand.

2.1.1 *One-Shot Algorithms*

Here we present a review of the popular YOLO algorithms for object detection, as well as RetinaNet [5].

YOLO or **"You Only Look Once"** is a popular group of algorithms often used in real time applications. The first paper was released in 2015 [6] and since then there have been quite a few versions developed, like the established v5 and the latest ones being v7 and v8, still under development [7]. YOLO takes an image as input and predicts bounding boxes and class probabilities using a single end-to-end trained neural network. This method is less accurate compared two multi-shot approaches but quite faster [8], making it a good candidate for real time applications. YOLO models are composed of three basic blocks: the Backbone, the Neck, and the Heads, which perform predictions.

The Backbone is a pretrained network used to extract features from the images. The Neck extracts feature pyramids, meaning features on different sizes and scales, which makes the model very robust. Finally the detection Heads apply predefined anchor boxes to feature maps to get the final predictions which are the bounding box coordinates, the confidence scores and the class names.

Below is a small overview of the way YOLO handles input.

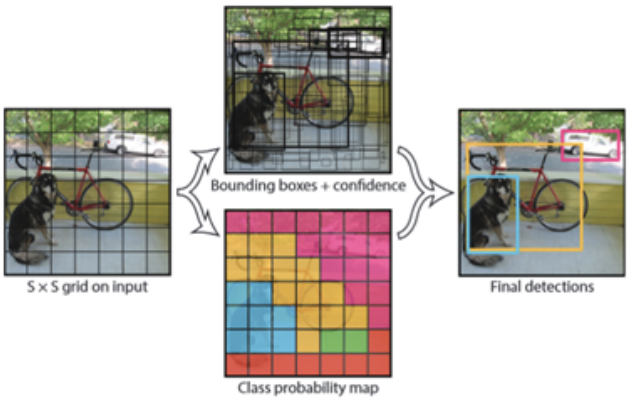

Figure 3: Simple explanation of YOLO detection process [6]

- The input image is divided into NxN grid cells depending on the image size and anchor boxes size, a typical number for YOLOv5 is 32x32. If the center of an object lies withing the grid cell, then that grid cell is responsible for detecting that object. Each cell predicts B boxes and confidence scores, namely in YOLOv5 each grid cell predicts threeanchor boxes. For each box, the model predicts the (x, y) center coordinates relative to the grid cell, the (width, height) of the box relative to the input image, a confidence score and class probabilities.

- During training, the aim is to predict one box per object (class). Based on the predictions, the Intersection over Union (IoU) score is calculated between the ground-truth boxes and the predicted boxes and only the prediction with the highest IoU is kept. This process helps each bounding box become more adept at predicting specific objects of a certain size and aspect ratio, thus improving the recall score.
- Non-Maximum Suppression (NMS) is employed as a post-processing step to reject overlapping boxes if their IoU is greater than a given threshold, resulting in a single bounding box per object.

To give more insight on the design choices we will look some more into YOLOv5 components, since this is the algorithm used in this work.

Backbone: Yolov5 employs a CSP-Darknet53 [9] as its backbone. This is an altered version of YOLOv3's DarkNet53 [10] CNN that employs Cross Stage Partial (CSP) network strategy. The problem it solves is the use of duplicate gradients around some layers of the previous backbone, which reduces the computations and improves inference speed.

Neck: The Neck leverages an improved version of Spatial Pyramid Pooling Block, which aggregates the features from the input and returns a fixed length output. It's role is to separate the most important features of the context while maintaining speed.

Head: Three Detection heads are used, namely CNNs, which are the same as v3 and v4 of YOLO. Essentially these make the predictions, one is for small, one is for medium and one is for large objects. The outputs are the coordinates of the bounding boxes, the class names and confidence. NMS is performed afterwards.

Some other changes introduced in YOLOv5 include the use of Swish activation function for the convolutions in the hidden layers, and the Sigmoid for the output layers. This version additionally employs "dynamic anchor boxes" or "Autoanchor" to generate boxes. Anchor boxes are predefined boxes with various sizes and aspect ratios, useful for detecting objects of different sizes e.g. to detect a standing person, we would define a vertical box. Anchor boxes are useful in the Head of the model where the final predictions are generated. This new method uses clustering to group the ground truth boxes and then uses the centroids as anchors. This makes it possible for the anchor boxes to be more aligned with the detections' size and shape. Finally, a variant of the original loss function is used, named "CIoU" loss that improves the models performance on imbalanced datasets. The latest version of YOLO has a difference with the previous ones in terms of using anchor boxes. The v7 uses 9 anchor boxes, which allows it to detect a wider range of object shapes and sizes compared to previous versions, thus helping to reduce the number of false positives.

Another prominent one-shot object detection model is RetinaNet. It's a single network composed of a backbone and two sub-networks, one for bounding box regression and one for classification. It utilizes a focal loss function to tackle class imbalance, which essentially is the crossentropy loss with a modulating term applied to it, in order to focus learning on hard negative examples. We can analyze the different components.

- A feature Pyramid Network or FPN [11] is built on top of a ResNet [12] architecture and constitutes the backbone of this network. FPN is a feature extractor that takes a single-scale image and outputs proportionally sized feature maps at multiple levels, in a fully convolutional fashion. It uses a top-down architecture with lateral connections to fuse high-level semantic information from deeper layers with low-level features from earlier layers, enabling precise localization and strong recognition.
- Two sub-networks follow the backbone: the Classification and the Box Regression subnetworks. The first is for classifying anchor boxes and the second one performs regression from anchor to ground truth boxes containing the object. The regression subnet has a similar structure with the classification counterpart, but they both use separate parameters. Similarly to YOLO, RetinaNet also employs anchor boxes. These anchor boxes act as references for predicting the bounding boxes of objects in the image.
- Lastly, the focal loss conceptually assigns less weight to the positive examples and emphasizes the misclassified ones. In practice, the authors use an α -balanced loss:

$$FL(p_t) = -\alpha_t * (1 - pt)^{\gamma} * log(p_t)$$

where γ denotes the focusing parameter.

Overall RetinaNet can handle objects of various sizes, and reduces the impact of imbalanced classes. It is a powerful tool for object detection tasks as it is suitable for real-time applications without reducing computational efficiency.

2.1.2 *Multi-stage algorithms*

When it comes to **multi-stage** methods, **R-CNNs** are a very popular group of designed for object localization and recognition. The first paper introduced R-CNN [11] in 2014, and it is the first successful effort to use CNNs to detect, localize and segment objects.

The concept of Region Proposal Networks (RPN) is introduced, and is made out of three stages:

In the first stage, candidate bounding boxes of different scales are generated by using the selective search algorithm [13]. Selective search initially employs a segmentation algorithm to detect blobs in an image and ranks them based on a similarity metric such as color or texture. Similar blobs are merged together and the first step is repeated iteratively hence larger region proposals are generated in a bottom up manner. Then, a feature vector of size 4096 is extracted from each region proposal and lastly a pretrained SVM classifies the region as either background or one of the classes.

This approach came with some limitations such as slow and multi-level training because of the three separate stages, and each image had too many region proposals that had to be trained (around 2000 per image at test time). This made the training time quite high and computationally expensive, as well as dependent on the selective search. Additionally, making predictions on a high number of region proposals is very slow and not possible to use the approach for real time tasks. These were the limitations the 2015 method named Fast-RCNN [14] aimed to tackle.

In this case training is single stage using a multi-task loss, detections are more accurate and training updates all network layers. The model takes an image and multiple region proposals and feeds

them through a deep CNN. Each region is then passed through a new layer called Region of Interest(RoI) pooling layer that extracts features which are then passed through a fully-connected layer. After that, the network "splits", giving two outputs per region; class probabilities and bounding box regression offsets per class. Per region, this process is repeated more than once. This version of R-CNN is faster during training and inference, however it still requires many region proposals.

The next year, Faster R-CNN [15] was proposed by different authors and made a breakthrough in the object detection field. This time the RPN is created to both propose and refine region proposals as part of the training process. Then, in a single model architecture, these regions work in tandem with a Fast R-CNN model. Both RPN and Fast-RCNN utilize the same deep CNN output.

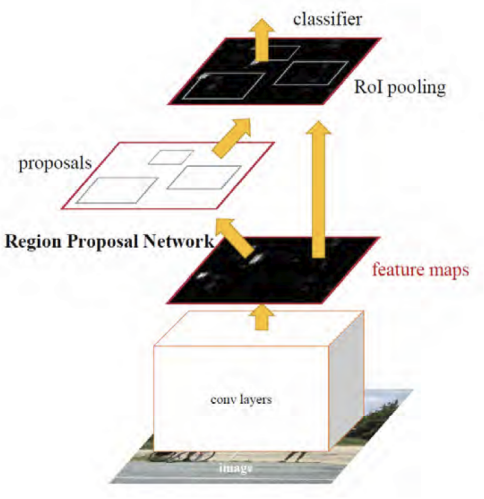

Figure 4: Faster R-CNN [15]

The RPN works by taking the output of a pre-trained deep CNN, such as VGG-16 [16], and passing a small network over the feature map and outputting multiple region proposals and a class per region. Region proposals are bounding boxes, based on anchor boxes designed to accelerate the process. The class prediction indicates the presence of an object (objectness) of the proposed region. The Faster R-CNN uses the RPN as an attention mechanism, which shares full-image convolutional features with the detection network thus making the proposal computation much cheaper and generating less regions. Since then quite a few approaches try to fine-tune Faster R-CNN on faces [17], [18].

Multi-stage detectors also come with limitations as they can suffer from bad lightning conditions or extreme poses [19]. Additionally, unlike one-shot approaches, the computation time depends on the number of faces, which in turn define the amount of region proposals so their performance can vary. One-shot detectors on the other hand are faster but compromise the accuracy [20]. Thus, the choice of an object detector depends on the task and the trade-off between speed and accuracy.

2.1.3 *Face Detectors*

Early approaches for face detection include feature based classifiers such as Viola-Jones and have remained in use for years [21]. However, the current state-of-the-art techniques for face detection are predominantly built on deep learning, particularly Convolutional Neural Networks (CNNs) [12]. CNNs take an image and perform a series of convolution and pooling operations to capture the most important features in it. They can process a big number of images with at a high speed making them quite effective. The development of large organized face datasets and the vast improvement of object detection methods has yielded many trained models that can be used for face detection. Face detectors can also be separated into the same categories according to how they perform the detection, namely one-shot such as RetinaFace[22] or multi-stage face detectors, like MTCNN [23], which also performs landmark prediction.

One-shot face detectors are used in real-time applications and their processing time remains stable no matter the amount of faces in an image [24]. They perform the feature extraction, generation of proposed regions that contain the object and face detection in a single step.

A popular one-shot face detector is the RetinaFace[22] framework. It's a multilevel face localization method, based on RetinaNet [25], that achieved outstanding performance on the WIDER Face dataset [26]. It is designed in such a way that it accomplishes three goals; face box prediction, 2D facial landmark localization and 3D vertices regression with the common target that all points should lie on the image plane. It has the ability to detect multiple faces and keypoints while being a lightweight approach.

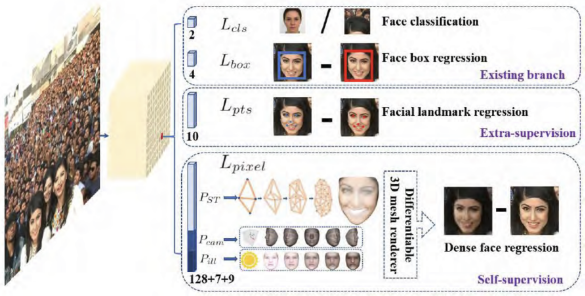

Figure 5: The RetinaFace method [22]

It incorporates the cascade-structure idea from MTCNN and the idea of 3D mask reconstruction from [27] and [28] to better localize the face. It uses a Feature Pyramid Network (FPN) and anchor boxes designed to detect faces at various scales and sizes. This ensures the accurate detection of both large and small faces within the same image.

RetinaFace consists of a backbone network for feature extraction, namely a ResNet-50 pretrained on ImageNet [29] and fine-tuned on WIDER Face. Afterwards, one more pyramid layer is added on top of the FPN, whose role is to generate pyramid features at different scales by taking the feature maps from the backbone. Then, context

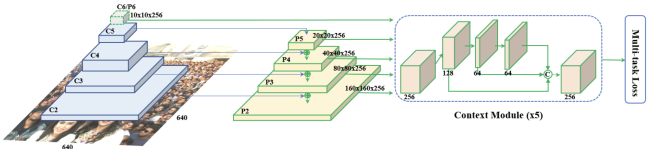

Figure 6: Overview RetinaFace [22]

modules are applied to the pyramid levels to capture more information about the surroundings. These are consequently passed through the last detection stage, which is composed of two subnetworks: the classification and the regression sub-network. The first is in charge of predicting the object class while the later generates the bounding box coordinates. Additionally, convolutional layers within the context module and FPNs lateral connections are replaced with the Deformable Convolutional Network (DCN). DCNs introduce a flexible convolution operation that enhances the ability of the network to handle complex and variable shapes in the input data. Finally, a multitask loss is used to achieve better localization, consisting of a face classification loss, a face box regression loss, a facial landmark regression loss and a dense regression loss. The latter is a function of the width and height of the anchor, the camera, texture and illumination parameters. It's used to compare the pixel-wise difference of the rendered and the original 2D face. As seen on Figure 22 showing the worlds largest selfie consisting of 1151 people, RetinaFace can find around 900 faces with a threshold of 0.5.

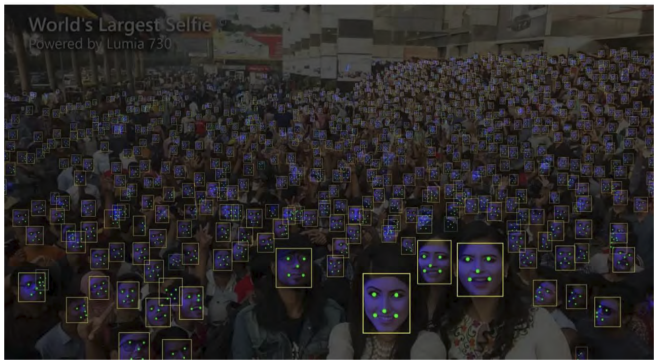

Figure 7: RetinaFace Detections on Worlds' Largest Selfie [22]

2.2 Facial Landmark Detection

Facial landmark detection is the task of finding coordinates of points in the face that relate to certain areas such as eyes, nose. One use of landmarks is in audiovisual speaker detection, where we are interested to detect the speaker and it's possible to achieve this by looking how the specific landmarks around the mouth move by measuring their distance from an absolute reference such as the nose [30]. Another use of landmarks is for facial expression recognition [31]. In such approaches the first step includes face detection, application of a landmark detector on the face, extracting features form these landmarks and feeding them to a neural network to classify facial expressions. Other popular uses include face filters on Instagram, face animation creation, monitoring of driver's tiredness to avoid an accident by sounding an alarm.

To give a formal definition of the facial landmark detection problem, let I be an input image of size $W \; x \; H \; x \; C$, where W denotes width, H denotes height and C denotes the number of color channels. The aim of facial landmark detection is to find a function that predicts a landmark matrix from the input image I, with the X and Y coordinates of the landmarks on the image. The number of predicted landmarks depends on how many landmarks have been annotated on the dataset used to train the landmark detection algorithm. To predict facial landmarks, the initial step is to perform face detection and then perform regression to find the x and y coordinates of the landmarks. According to the categorization found in survey [32], modern facial landmark prediction methods that rely on neural networks fall under these categories:

- **Direct Regression**
  Direct regression is a simple method to predict the coordinates of facial landmarks given an input face image by using a trained CNN. Modern approaches utilize pretrained networks as backbone like MobileNetV2 [33] or Resnet [12]. Direct regression has the benefit of being simpler and more direct than other, more sophisticated techniques like heatmap regression. Due to the fact that it does not take into account the spatial correlations between various landmarks like heatmap regression does, it might be less resistant to occlusions
- **Heatmap Regression**
  This method's key idea is to train a model that creates a heatmap for each landmark, where each pixel denotes the probability that the associated location in the image is that landmark. A heatmap is generated for each landmark by plotting the Gaussian distribution. A common approach to generating heatmaps is to use a neural network, such as a CNN, to predict the heatmaps from the input image. The network is trained using a dataset of images labeled with the coordinates of the landmarks. During training, the network learns to generate heatmaps that have high values at the locations of the landmarks and low values elsewhere. During inference, the model accepts a face image, outputs heatmaps and to converts them into direct coordinates, the argmax of all pixels is calculated and translated back to the original pixel in the input image.
  Heatmaps are effective because they consider the structural relation between facial parts. Thus a model can be trained on heatmaps instead of direct coordinates. During inference, the model accepts a face image and outputs heatmaps. To convert them into direct coordinates, the argmax of all pixels in the heatmap is calculated and translated back to the original pixel in the input image.

The benefit of heatmap based models is that they are more robust to different poses and occlusions. For example if part of the mouth is covered, the heatmap values for the mouth will still be high for the visible region. They can also be easily integrated into a wider

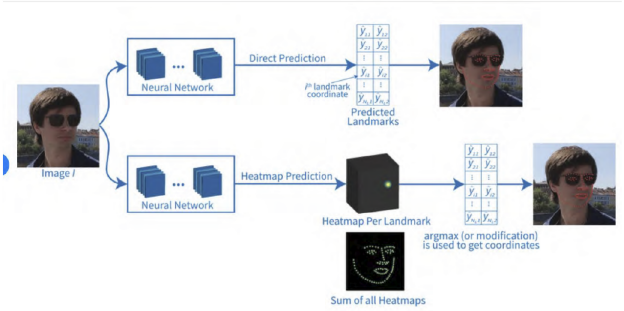

Figure 8: Direct and Heatmap based Regression [32]

range of applications such as facial expression recognition and face alignment. Some disadvantages can be that the model is sensitive in choice of hyperparameters and its efficiency can suffer is they are not choose carefully. When the spatial relationships between the landmarks are not clearly specified, heatmap regression may occasionally be less accurate. For instance, it could be challenging for the model to successfully predict one facial landmark without also accurately predicting the others if the locations of the landmarks are highly correlated.

2.2.1 *Frameworks*

This subsection presents an established method for predicting facial landmarks.

The **Multi-Task Cascaded Convolutional Neural Network (MTCNN)** [23] is a multistage face detector that detects five facial landmarks if there is a face. It's often encountered in literature under multi-stage face detectors and not as a landmarking algorithm but since it also detects landmarks, it will be discussed in this section.

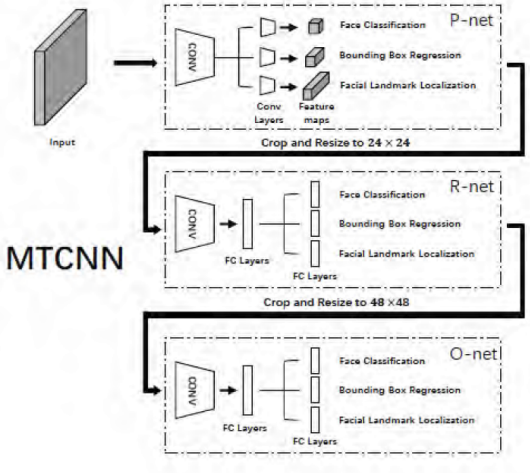

Figure 9: MTCNN Structure [24]

It relies on the inherent relationship between face location and the landmark locations. It is built with three stages of CNNs connected in a cascade fashion, and also produces facial landmarks. As a preprocessing step, the input image is rescaled into different sizes to form an image pyramid, which is fed to the initial stage. During the first stage, the Proposal Network (P-Net) is a fully convolutional network that obtains candidate windows and their bounding box regression vectors after refinement. The candidates are fed into the second stage which is the Refinement Network (R-Net). The R-Net is a CNN that further reduces the number of candidates by performing Non-Maximum suppression (NMS) to merge overlapping ones. The output is whether there is a face and coordinates for facial landmarks. The third and final stage is the Output Network (O-Net), also a CNN which describes the face in more detail and outputs five landmark positions for eyes, nose and mouth.

2.3 Face Detection Under Occlusions

Occlusions are an issue that cause face detection models to fail because of the lack of information about the occluded part. This is because if the face is occluded then it can't be detected. Occlusions are hard to model because of their variance in size, shape and along with other factors such as luminance can make the task quite challenging. Moreover, collecting a dataset with realistic occlusions and the corresponding non-occluded image, and training a deep learning algorithm is rather difficult to do. Consequently the performance of facial detection algorithms greatly relies on the type and location of the obstacles. Examples of occlusions found on a face are glasses, masks, hands, scarfs. Thus, for solving a specific task, knowing if and where there are occlusions can help adjust the model by providing it with useful information about the location and occlusion type. The question now becomes what to do with the occlusions? One approach in face detection is to first detect the occluded part and then perform recognition based on the unoccluded part. Another approach is to address this problem as a face recovery problem. These methods try to recover the whole face from the occluded region either by reconstruction or by inpainting. The latter inpainting methods don't really perform face recognition but rather try to repair the image. Understanding the type and location of occlusions in a particular task can lead to effective model adjustments. This understanding helps to provide critical information about the occlusion, thus allowing for more precise modeling.

2.3.1 *Frameworks*

In this part we analyze some popular face detectors that deal with occlusions.

The **Face Attention Network** [34] is able to deal with detecting multiple, multi-scale faces under occlusions without slowing down speed and simultaneously reducing the number of false positive detections. It follows a similar single-shot architecture as RetinaNet discussed previously, but is optimized for face detection. It consists of two stages, the first is the RPN to generate region proposals for faces and the second is the actual Face Attention Network (FAN). FAN consists of a backbone network, in this case a ResNet50 to extract features and a novel anchor-level attention module to handle occlusions. The latter gives emphasis on the unoccluded parts of the faces such as eyes, which are more useful for the detection. The attention module receives the feature maps from the backbone

network and creates attention maps with the most important parts. The difference from the traditional usage of attention is that it is first given to an exponential operation and then dot with the feature maps from the backbone. FAN comes with five detection stages, each associated with an anchor of a specific scale similarly to YOLO, meaning that one stage can detect larger faces, another one smaller faces and so on. The attention highlights for each stage the important parts of the face which are associated with the ground truth bounding boxes.

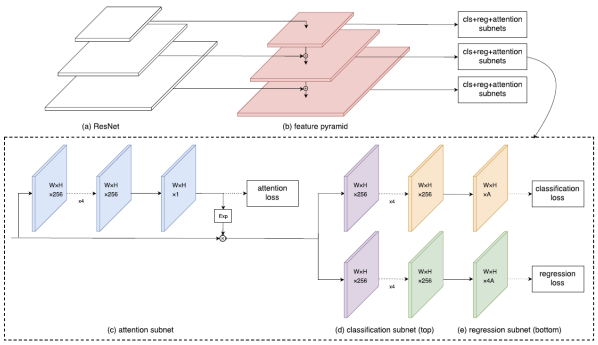

Figure 10: Overview FAN architecture [34]

Then, the attention and feature maps are concatenated and passed through multiple CNN layers to produce the final output. From Figure 10 and 11 we see that different layers represent different face sizes. The network is trained end-to-end to minimize the classification and localization loss by using anchor based multi-task loss. From experiments on the MAFA [35] and WiderFace [26] datasets the network has a higher Average Precision score compared to other approaches.

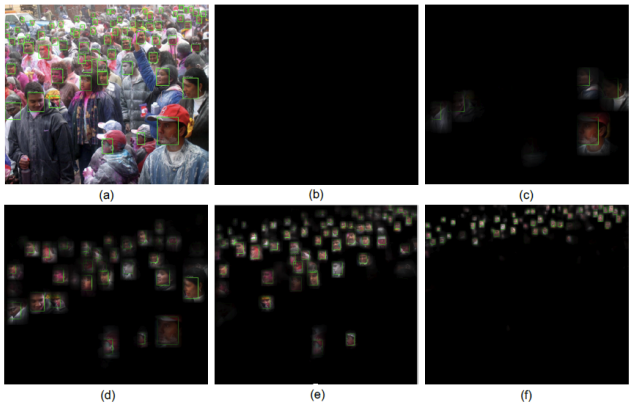

Figure 11: Top left is the original image, the rest are attention maps proposed by the network at different scales[34]

Another advanced CNN approach to handle challenges such as illumination and occlusions is **Contextual Multi-Scale Region-based CNN (CMS-RCNN)** [36]. The network, similarly to the human intuition, looks at features from multiple scales and at potential body regions to decide if there's a face present. It's an advancement of Faster R-CNN that was previously discussed with the addition of body context information and the ability to deal with tiny faces.

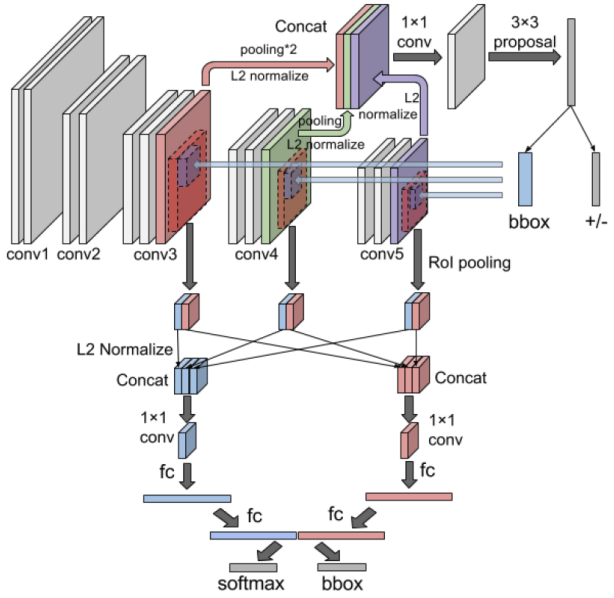

Figure 12: CMS-RCNN architecture [36]

Like FAN, it consists of a *Multi-Scale Region Proposal Network (MS-RPN)* to generate candidate regions and of a *Contextual Multi-Scale CNN (CMS-CNN)* to perform inference for the face candidate regions. Looking at Figure 12, the upper part corresponds to the MS-RPN, the lower part is the CMS-CNN. Within it, the blue blocks are the face features, red are the body features and they are processed simultaneously. The assumption is that if there's a face, then there must also be a body with a fixed spatial relationship to the face. However this may not be true in all cases, but can be interesting for specific datasets such as preterm babies that are shown lying down. To capture the body features, extra RoI-pooling operations are done for each region proposal. The first five convolutional layers are from VGG-16 and their parameters have been initialized by the pretrained version. For every candidate, a bounding box and a confidence score are calculated, and finally a threshold is applied to keep the specific face regions.

When it comes to the task of facial landmark detection, the paper [37] presents a robust method using **Occlusion-adaptive deep networks**. The proposed method is able to handle different types of occlusions (e.g. hair, hats, glasses) more effectively than previous approaches by making the CNN less sensitive to occlusions. It utilizes a multi-task learning approach that includes estimating the occlusions in addition to the traditional facial landmark detection

task. The occlusion estimation task allows the model to better understand the context of the occlusions, which leads to more accurate facial landmark detection.

In more detail the framework consists of three modules:

- *A distillation module*: obtains clean representation by filtering occlusion features adaptively via self-attention. It infers the occlusion probability map based on the high level features it captures.
- *A low rank learning module*: recovers missing features by labelling the inter correlation features of faces and utilizes geometric features as well.
- *A geometry-aware module*: captures geometrical information about the face such as proximity, position to assist the low rank learning module.

The geometry module mitigates the problem that CNN's don't capture long term dependencies efficiently. This means that they capture information inside their receptive field, however they don't take into account the geometrical similarities that are found in human faces e.g. the eye-eye distance. Facial components do share some structural relations and such relations can be useful in detecting landmarks, thus implementing a method to model them would be smart. The module uses the outer product of two feature maps to catch such facial dependencies. The role of the distillation module is to filter out occluded regions, even background, in order to make the CNN more robust. Then, the geometric features from the first module and the clean face from the second module are merged into one high-dimensional feature map and fed to the low-rank learning module. This high dimensional feature map, although more inclusive, may not be a completely accurate representation of the face because some features have been removed. These features might have been useful to have, as they can be correlated to other existing ones, but currently this relationship is lost. The low-rank module aims to recover such features by learning a shared structural matrix that encodes such correlations, simultaneously eliminating redundant ones. The method is evaluated on several benchmarks and demonstrates improved performance compared to previous methods.

2.4 Explainability Methods for CNNs

Explainable AI methods help visualize and understand why black box deep learning models make their predictions. This is achieved by highlighting the most important parts on an input image that play a role in the prediction, making the model interpretable. The methods discussed here fall under the category of Class Activation Mapping (CAM). They use the activations of the convolutional layers to generate heatmaps that highlight the important regions for detecting an object, and can do this per class or by averaging over all classes and producing a single heatmap. Their common element is performing a weighted operation between the feature maps and the learned weights. The produced heatmap is then overlayed on the original image to give insight and meaning to the result. Most methods are gradient based meaning they utilize the flow of gradients, but when this information is not available, gradient free alternatives are also available.

The most prominent gradient based method is **GradCAM (Gradient-weighted Class Activation Mapping)** [38], and many other works have been inspired from it. It requires a CNN model with a Global Average Pooling layer and softmax activation. The last convolutional layers of a model contains high-level semantic and detailed spatial information. The method obtains this layers' feature maps and gradients by forward passing the image. The gradients are averaged across height and width dimensions in the global average pooling layer, and the output is one weight per feature map that shows its importance in the final prediction. Then, these weights are elementwise multiplied with their feature maps and summed across the channel dimension to output a single heatmap. Finally, a ReLU activation is applied to eliminate negative values and enhance the visibility.

$$\text{Grad-CAM Heatmap} = \sum_i \alpha_i^c \cdot \text{ReLU}\left(\sum_j \frac{\partial y^c}{\partial A_{ij}} \cdot A_{ij}\right)$$

Where:

Grad-CAM : The final heatmap highlighting important regions

c : Target class index

α_i^c : Weight associated with the activation map

$\frac{\partial y^c}{\partial A_{ij}}$: Gradient of the target class score wrt the activation map

A_{ij} : Activation value at position (i, j)

The method works very well and most of the other methods stem from it.

One extension, **GradCAM++** [39] aims to improve the localization accuracy by using second order gradients. It's very useful especially in the presence of multiple objects withing an image. The intuition behind it is than the existence of similar objects with small variations in an image can excite different feature maps in GradCAM, resulting in a misrepresentation of some of them. The heatmap is calculated by taking the RELU of the first order gradients and multiplying it by the second order gradients. This results in a more detailed heatmap that highlights better the finer details.

Another method, **HiResCAM** [40] is proposed to solve the problem of the averaging step of GradCAM. The authors find that the averaging of the feature importance weights limits the final visualization and can result in inaccuracies due to the low dimensionality of the averaging step. They propose element wise multiplication of the gradients and the feature map before the summing step. Then they prove that this method is more accurate than GradCAM and show how the latter can highlight irrelevant regions that do not increase the class score for a class.

Axiom-based Grad-CAM (XGradCAM) [41] is yet again another extension of GradCAM, meant to scientifically set two axioms to make the method reliable; sensitivity and conservation. Sensitivity means that each response in the explanation should be equivalent to the change in output resulting from the removal of the corresponding feature from the input. Conservation indicates that the sum of the explanation responses should match in magnitude of the model output. The authors come up with a new formula that implements spatial attention coming from intermediate layers.

The weight of the feature map is calculated by taking the weighted average of the gradient by solving an optimization problem. Compared to GradCAM this implementation better highlights individual instances per class.

Gradient free methods offer an interesting alternative when gradient information is not available. We will look at two models, **ScoreCAM** [42] and **EigenCAM** [43].

ScoreCAM discards the use of gradients, and calculates the weights from the forward pass of the network. Then they are linearly combined with the activation maps to produce the output. In more detail, there are two phases in this approach. The first one is the forward pass where the activation maps per class are extracted. Then these are masked on the original image and this temporary input is passed through the CNN and through fully connected layers to obtain the weights. Finally, the activation maps from phase one and the weights of phase two are linearly combined and passed through ReLU activation, similarly to the previous approaches.

EigenCAM is another gradient free method. Similar to GradCAM, the feature maps from the forward pass are passed through Global Average pooling and multiplied with the weights to generate the heatmaps. Then, the covariance matrix per class is calculated. It represents the relationship between different activations in the activation maps. Furthermore, eigen decomposition is performed and returns the eigenvalues and eigenvectors. The former signify the importance of each eigenvector and only the highest scoring are kept. The next step is to reconstruct the activation maps from the remaining eigenvectors so that they represent the most discriminative features withing the activation maps. The eigen activation maps are reweighted with the eigenvalues so their importance is emphasized and them summed to form the final heatmap. EigenCAMs' visualizations provide more crisp and targeted regions and only highlights relevant parts on the heatmap.

3 Methods

This chapter contains more information about the problem, the proposed solution and details about the form of the datasets required for this task. We start by motivating why it's necessary to implement a custom face detector and then go into more detail about the separate aspects of the pipeline. Figure 13 presents the overview method.

3.1 Problem Analysis

There are several well-known facial landmark detector implementations based on RetinaNet[22], and MTCNN [23]. We want to see how they can perform on images of preterm infants. Specifically, we tested:

- FaceNet [44], a pretrained MTCNN for face detection which outputs the face bounding box and 5 landmarks (2 for eyes, 2 for mouth corners, 1 for nose).
- RetinaFace [45], the face detection module of the InsightFace project [46]. This model uses pretrained MobileNet weights and outputs the face bounding box and the same 5 landmarks as above.

To choose one for this thesis, these models were evaluated on their ability to accurately detect facial landmarks around areas such as the eyes, nose, and mouth, in a variety of settings. Given that the end goal is to perform face detection on preterm infants, we gathered 12 random images of preterm infants from the Internet, most of which contain occlusions such as tubes and medical devices around the face. Each images contains only one infant so we would expect to have 12 faces detected. Through experimentation it is concluded that both face detectors perform badly, especially FaceNet fails to detect faces while on the other hand RetinaNet identifies irrelevant parts of the frame as "face", and the landmarks are very out of place. Some example images with detections can be found in Figures 23 and 22 in the Appendix.

3.2 Proposed Solution

This small experiment emphasizes the need to create a customized face detector for the task as existing implementations provide inaccurate detections. Note that the photos collected online generally had good lighting conditions and image quality, something that doesn't always apply for the UMC preterm infant dataset. In the UMC dataset the videos have variable lighting conditions, the view may change, the infants move, the nurses interact with them and their head and face may be covered with a hat or medical equipment. We choose to train a YOLOv5 model for our specific task, since it is rather straightforward to train it, it's highly customizable and the current implementation provides a handful of useful insights on the training procedure, the parameters and makes it easier to debug it. We begin by building a custom dataset with annotated bounding boxes for different facial parts and some degree of occlusions. We then extract and save the coordinates form the provided landmark annotations.

In the first part of the implementation, the dataset is fed to YOLOv5 for training. Then in the second part, we extract and save heatmaps by running explainability algorithms from the model based on the weights of the first trained model. These are then refed to another YOLOv5 model which we train from scratch, with the aim of detecting only the whole face for the cases where the previous model failed. This method intrinsically tackles the occlusion issue as it "guides" the model into better localizing the whole face from the previous information about the location of the facial parts. Figure 13 is a schematic of the solution, where the top part corresponds to the first part of the implementation and the bottom part to the second part.

The final step and one of the aims of this thesis is to test the first part of this implementation on videos with various occlusion degrees and evaluate the whole face detection. Then for the cases where the face detection failed, we extract explainability heatmaps and test different methods to see whether this allows more faces to be detected.

3.2.1 ***Choices for the first part of the implementation (RGB Model)***

The first part of the implementation heavily relies on the dataset creation, the choice of classes, the bounding box extraction, and certain design choices for YOLO. These choices are based on experimentation so they will be presented in the Results section. The choices that were made from the beginning is to use pretrained weights and specifically YOLOV5-medium weights as they provide a good trade-off between speed and accuracy and are good for medium sized datasets. Data augmentation techniques are also used

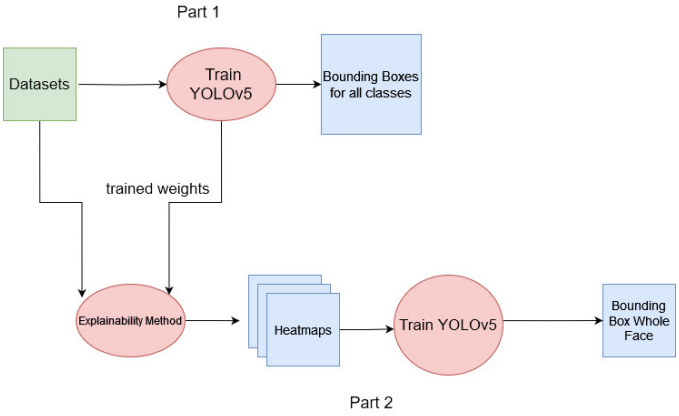

Figure 13: Proposed Method

during training as they provide more variability and increase robustness, but their precise values are set by running an evolutionary algorithm.

3.2.2 *Choices for the second part of the implementation (CAM Models)*

In order to generate Class Activation Maps, the best weights of the previous 10 class trained RGB network are used. These produce maps showing highlighted regions that have an effect on the prediction. There is a variety of methods (both gradient-based and gradient-free) modified for the YOLOv5 architecture that utilize the feature maps of any specified layer before the non-differentiable NMS module. We choose to work with GradCAM++ and HiresCAM, extracting features from layer -2 of YOLO, as they produce more accurate and crisp activation maps. The -2 is the default layer and corresponds to the last C3 Dense block, which is a modified Bottleneck CSP consisting of Convolutional layers and two Bottleneck layers. The last convolutional layers capture high-level abstract information so the generated heatmaps are representative. An illustration of the GradCAM++ and HiresCAM heatmaps for an image can be found in Figure 26 in the Appendix.

For extracting heatmaps there are two options:

(1) Get one heatmap per input image which is the average over all the classes or (2) Per input image get as many heatmaps as the number of classes, and each heatmap corresponds to each classes' activation. These can the be stacked together into a 10-channel TIF file. In our approach we experiment with both ways. We train all the models from scratch without pretrained weights and with the YOLOv5-medium model configuration, which is the medium sized YOLO version. The reason behind this choice is that larger models have higher mAP but more parameters, thus are slower to train and run. Smaller models on the other hand compromise the mAP thus the safe option is the medium model, which is suitable for mobile applications. We change the channel number to 1 for the averaged grayscale heatmaps and 10 for the 10 stacked heatmaps per class. No data augmentation is used and the model is trained only on the "whole face" class which we are interested in. One of YOLO's constraints is the use of letterboxing, a technique to make the input images square while maintaining the aspect ratio and preventing distortion. The function adds a border around the image and works only for 3-dimensional images so we adjusted it to work for 10-channel and grayscale images.

3.2.3 *Choices for testing the pipeline*

We put our implementation to test by following the process depicted in Figure 14. We start by validating the RGB model on our labelled data. This will output metrics and plots which are used to make comparisons between the models. We find the optimal confidence value which gives the best combination of precision and recall from the F1-confidence curve. We then run inference on our data using this optimal value and separate the images where the face was not detected. We proceed to the second part and extract heatmaps for the images without a face detection using the four different methods described above. Occasionally, the model might not detect classes in an image, resulting in blank heatmaps that must be removed. Lastly, we repeat the validation process for the four different CAM-based models, determine their optimal confidence values, and perform inference. Through this procedure, we can observe the total number of missed detections for each model, and thereby evaluate and draw conclusions about their efficiency.

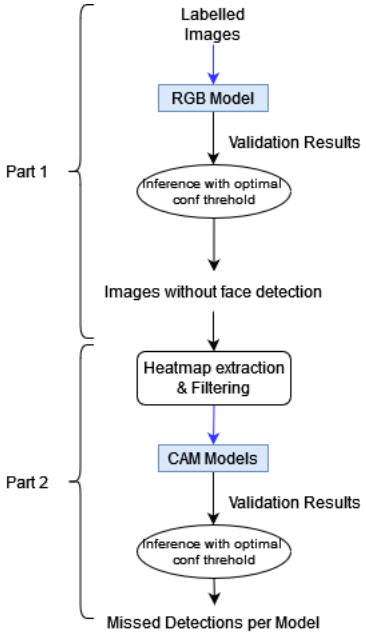

Figure 14: Testing Method

3.3 Data

To train a YOLO model, we need a dataset with images and bounding box coordinates. Each image corresponds to one label file which contains all the annotated box coordinates per class. YOLO's authors suggest having more than 1500 images per class, more than 10.000 instances per class, variety and a fully labelled dataset. Thus we aim to follow these rules to establish a proper baseline performance.

There are datasets that contain the bounding box around the whole face, but since we train on different facial parts as well, finding such data is not possible so we need to build it. The provided UMC dataset at the time of writing does not contain landmark annotations, thus can't be used for training. There are multiple datasets in literature that contain faces, with or without occlusions, either natural or made up ones. The images can have different lightning, contrast, rotation, zoom etc. A lot of datasets have unrealistic synthetic occlusions, too big or misplaced, and can hinder learning. In many cases they contain too few types of occlusions such as hats or sunglasses that are limited compared to what is found in the wild. In other cases there are wrong ground truth annotations in the data that can confuse the algorithm. A lot of these datasets also tend to be small. Thus it is challenging to pick one to evaluate the performance of an algorithm.

We aim for datasets with facial landmark annotations for at least one face, so that we can extract bounding boxes for the facial parts we need easily. The images need to have some variance, different scales, resemblance to realistic faces and some degree of occlusion to fit the problem statement. Therefore we don't look into synthetic faces but go for datasets that contain various images found online, which naturally contain occlusions. Most of them follow the iBUG 68 landmark annotation protocol (seen on the right side of Figure 2) and can contain some extra annotated landmarks. Landmarks can convey occlusion information, for example by having negative values for external occlusions and -1 for self occlusions (meaning the facial part is not visible). We account for these landmarks by skipping the self occluded parts and converting the externally occluded values into positive ones. This automatically adds occlusion information in our dataset to make the model more robust. To extract the bounding boxes we consider which facial parts are useful to be detected and create rectangles around them while also mapping the parts into classes. We get the rectangle coordinates, normalize and center them so that they are in the correct format for YOLO and generate labels files. There is the case that in the original dataset, one image contains multiple copies and landmarks, one for every annotated face. In this case we keep one copy of the image and merge the bounding box coordinates into one label file. Additionally, it can be that one image has multiple faces but only one annotated. We try to keep these images in the training set but not in the validation set as this will result in an increase in False Positives. The model may correctly detect more than one face but if it's not annotated, it is considered a mistake.

4 Results

This section contains the results and visualization of the proposed method.

4.1 Dataset Creation

This part presents exact statistics behind the custom dataset created, named "WFLW-MERL" as it's a mix of WFLW and MERL-RAV.

Table 1 presents a description of the original datasets used and below there are more details about them.

Original Datasets	Type of Images	Total Amount
WFLW	One or multiple faces	10000
MERL RAV	Mainly one face	> 19000
Hand Over Face	One face	4384

Table 1: Descriptions of the original datasets used for training and evaluation

The **WFLW** [47] (Wider Facial Landmarks in the Wild) dataset contains images with one or multiple people and has 98 annotations in the form of (x,y) coordinates per face, as well as coordinates of the upper left corner and lower right corner of detection rectangle. It also includes rich feature annotations such as make-up, occlusion, pose, illumination, blur, expression for analysis purposes. For the WFLW dataset, 6000 images are used as training data, 125 for validation and 794 for testing. One notable consideration is that the dataset contains only one annotated face per image, although multiple faces may be present. To ensure dataset consistency, a review was conducted, resulting in the removal of certain images with many faces from the validation and testing sets, ensuring the presence of only annotated faces.

The **MERL-RAV** [48] dataset contains over 19000 faces with a full range of poses. It is split based on the head poses, namely frontal, left, right, left half, right half. Each of these parts is split into training and testing set. The data follows the 68 landmarks annotation protocol from iBUG [49] and additionally each landmark belongs to 1 of the 3 occlusion classes: unoccluded, externally occluded, self-occluded (not estimated). Externally occluded landmarks correspond to visible face areas in the image that have an obstruction (e.g. hats, glasses) and have negative values (-x, -y) . Self-occluded landmarks are caused by extreme head poses and are not visible in the image, and are in the form (-1, -1). From the MERL-RAV dataset we keep only the images with frontal, left half and right half head poses to have a degree of variation. This gives 8601 images for training, 1656 for validation and 804 for testing. Both test sets consist of data that has not been used during the model training phase, and they are annotated with ground truth landmarks.

The **Hand Over Face**[50] dataset is used for testing the approach. It consists of 4384 frames randomly selected form videos of a sitting person moving their hands around the face inside a lab. The videos were recorded twice with a Kinect camera and have two backgrounds: lab and wall background. Kinect can record depth stream and color stream video. In this case we kept the RGB images coming from that video. The video resolution is 1920x1080 and mainly the upper body part is visible.

We create the **WFLW-MERL** dataset by merging WFLW and MERL-RAV to create a larger dataset and to ensure diversity in the training data. This is possible because both individual datasets have landmark annotations and we can apply the same process to extract the same bounding boxes from the landmarks. While they do not have the same amount of landmarks annotated, it's possible to find the index of the landmarks of interest based on the used annotation protocol. To do so, we create lists with the coordinates of the landmarks we are interested in that correspond to a facial part. We can then create a bounding rectangle around the landmarks (namely with OpenCV boundingRect() function), and get the top-left (x,y) coordinates, the width and height of the

bounding box. We proceed to calculate the center coordinates from the top-left coordinates and normalize all the values, so they are in a suitable format to train YOLO. In Figure 15, the left image shows the ground truth landmark annotations and the right image the extracted bounding boxes per class.

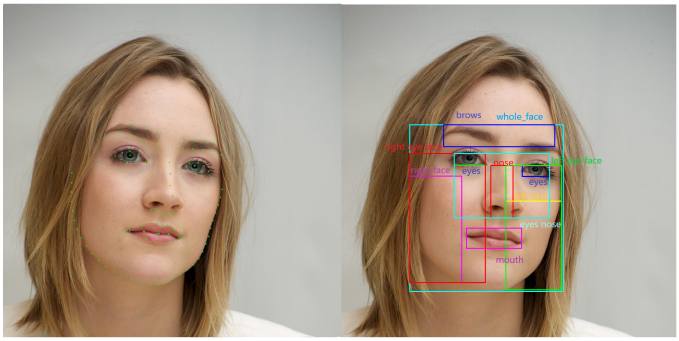

Figure 15: Example image from WFLW. Left: landmark annotations Right: Generated bounding boxes

Table 2 presents the training and validation split. Again, note that the validation and test sets contain only annotated faces so we can report on the performance.

Original Datasets	Train	Val	Test
WFLW	6000	125	794
MERL RAV	8601	1656	804

Table 2: Custom dataset train and test splits

We choose to make bounding boxes for the following facial parts:

- *eyes*: the left and right eye are concatenated in one class as they are very similar and it would be very difficult to find their differences
- *nose*
- *whole face*: box ranging from above the brows to below the chin, and from leftmost to rightmost visible cheek
- *right face*: box ranging from mid-ear until below chin, and from rightmost cheek landmark to right tip of mouth
- *left face*: box ranging from mid-ear until below chin, and from leftmost cheek landmark to left tip of mouth
- *brows*: box around both the brow area
- *left eye face*: concatenation of the left eye and the left face landmark coordinates
- *right eye face*: concatenation of the right eye and the right face landmark coordinates
- *eyes nose*: concatenation of landmark coordinates of nose and both eyes
- *mouth*: box around the mouth area

The eyes, nose, whole face, brows, mouth are the 5 most prominent facial parts, and the rest of the classes are a combination. The whole face class box fully contains the boxes of the other classes. The reasoning behind them came by looking at the provided preterm infant videos and considering which facial parts could possibly be well detected. For example, if the infant is on its side, instead of detecting the whole face we could get right face, left face, and left face with the eye, right face with the eye. If the infant has occlusion around the mouth, the model could detect the upper facial part so that would be the eyes and nose class. We expect that one missed class should not affect greatly the performance of the other classes, especially for the classes that do not have a natural overlap such as the nose and mouth.

4.2 Losses & Metrics

YOLOv5 uses a sum of three different loss functions: classification loss, bounding box regression loss, and objectness (confidence) loss. The first is a crossentropy loss, the second is a mean squared error based loss and the last is confidence of an object being present in an anchor box.

To evaluate and understand an object detection models' performance, there are some common metrics used in literature.

The Intersection over Union (IoU) is the amount of overlap between the predicted bounding box and the ground truth bounding box. During NMS an IoU threshold is set to remove overlapping predictions that exceed it. First the detections are sorted based on the highest confidence scores, the IoU between the highest scoring detection and the rest is calculated and if it exceeds the threshold, the lowest scoring detections are made redundant. The process repeats until all sorted detections have been processed, and results in non-overlapping high confidence predictions.

The precision is the ratio of true positives divided by the total number of positive predictions:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

It gives information about how well the model can classify the actual face or facial part without classifying irrelevant objects. Recall is the ratio of true positives divided by the total number of actual objects:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

It shows the models' ability to classify and localize positive samples, without missing them, which is relevant in our case as we don't want to miss any existing faces. The F1-score is the weighted average of the precision and recall,

$$\text{F1} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

The Average Precision (AP) determines the algorithms' precision at different levels of recall by measuring the area under the Precision-Recall curve, over all classes. Thus, AP@0.5 means the AP is calculated at an IoU threshold of 50%, so the predicted and ground truth bounding box have overlap (IoU) of at least 50%. The mean AP is the AP calculated over different IoU thresholds. For example, mAP 0.5_0.95 is the average precision over multiple thresholds with a step of 0.05.

For YOLOv5, the mAP is calculated based on the PR curve for a confidence of 0.001. During the calculation of recall an epsilon value of 1e-16 is added to the denominator. This is the same for the calculation of the F1-confidence curve, which is used later to

determine the optimal confidence threshold for detection. In our approach we use these results from the validation to get an overview of the models performance and make comparisons. We also perform inference (detection) with the calculated optimal confidence thresholds to determine which images have no face detections and continue with the second part of the pipeline.

4.3 Training - First RGB model

YOLOv5 can perform hyperparameter evolution for a number of generations to find the optimal values for around 30 hyperparameters for our data. It employs a genetic algorithm for a default of 300 generations. There are two key probabilities in genetic algorithms: mutation and crossover. The crossover signifies the likelihood of generating a new solution from two existing solutions. This means picking the best features to create a better performing offspring, however it can lead to staying within a range of suboptimal solutions. On the other hand the mutation probability signifies the chance of adding diversity in the population, which increases the search space. In this work mutation is used with an 80% probability and a 0.04 variance to create new offspring based on a combination of the best parents from all previous generations. This prevents the genetic algorithm from getting stuck in a local minimum or converging to a suboptimal solution, however it vastly increases computation time for our case. The value to be maximized is fitness, a weighted combination of mAP, mAP@0.5 contributes 10% of the weight and mAP@0.5:0.95 contributes the remaining 90%. Because the process takes too long to converge and due to limited resources and technical issues that arise with the GPUs when training for a long time, we trained for 42 generations, with the best results coming from generation 12. Table 3 presents the values for several parameters.

Hyperparameter	Value
Starting lr	0.0072
Optimizer	SGD
IoU threshold	0.2
Hue	0.0107
Value	0.42373
Saturation	0.52782
Translation	0.07382
Scale	0.48072

Table 3: Hyperparameter evolution values (gen. 42) used in the first YOLO model for training

We train the first model on the facial parts for 300 epochs with Early Stopping enabled. The metric used in Early Stopping is fitness, the previously mentioned weighted combination of mAP. We used the hyperparameters and data augmentations from the evolutionary algorithm as parameters to the model and the pretrained "yolov5m" weights for starting weights.

Furthermore, YOLOv5 automatically checks the label values for negative or non-normalized values (between 0-1) discards them and their corresponding images. In our case this only happens for the training set, from the initial 14601 images we have 14580 remaining.

Figure 16 shows the amount and distribution of the total labels. The top left bar plot presents the label frequency, the top right is a schematic of the bounding boxes, the bottom left shows the frequency of values for the (x,y) top left box coordinates, the bottom right does the same for the width and height of the boxes. It can be observed that the "eyes" class is double in amount as we extract the two eyes from the landmark annotations and merge them into one class. Furthermore the majority of x-coordinates range from 0.2-0.4 and the y-coordinates from 0.2-0.6, so the bounding boxes are located around the lower and center part of the image. The width and height values tend to be small, located on the origin of the plot, as most of the face parts such as eyes (which are double), nose, mouth cover a relatively small area.

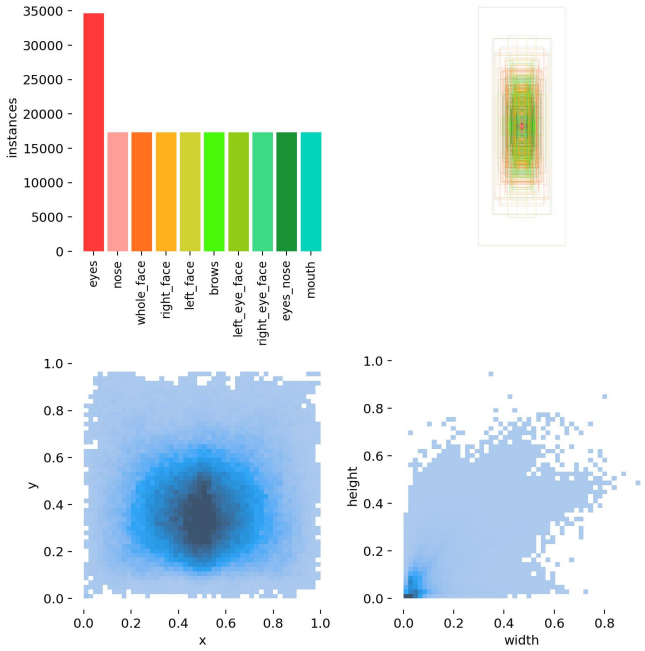

Figure 16: Label information first model.

4.4 Part 1: RGB Model Results

Figure 18 shows the confusion matrix for the classes and Figure 24 in the Appendix shows the loss plots and validation metrics. The default threshold values are kept for the validation, so the confidence threshold is 0.001 and the NMS IoU threshold is 0.5. It is important to keep the confidence value at 0.001 as changing it affects the computation of the PR curve leading to incorrect mAP values. The reported precision and recall scores are presented at the maximum F1 confidence threshold, which should produce their best balance. For this case it's 0.373 for all the classes as illustrated in Figure 17.

The validation loss stops improving so the model finishes training at epoch 85. From the plots, the box loss and classification loss drop quite quickly around epoch 10 and continue dropping, the first one starting around 0.1 and reaching 0.05 and the second

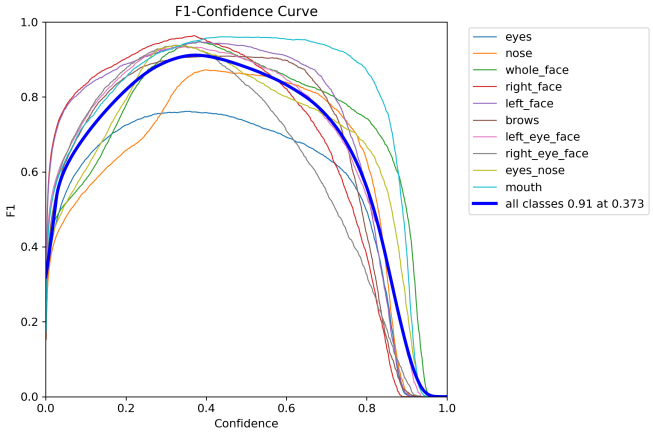

Figure 17: F1-confidence curve on the validation set for the 10class RGB model

one starting at 0.03 and ending at 0.017 for the final epoch. The objectness loss starts from 0.034 and drops more slowly to 0.022. In the meantime the respective losses for the validation set seem to converge around epoch 40 while the precision, recall and mAP scores all reach high values. More specifically, precision and recall reach over 0.9, mAP@0.5 reaches 0.93 and naturally mAP@0.5-0.95 is lower at 0.7.

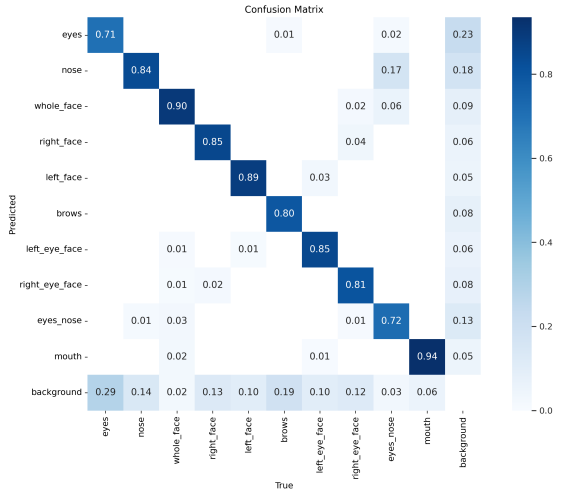

Figure 18: Confusion matrix on the validation set for first model

From the confusion matrix on the validation set the highest accuracy (94%) is for the mouth, while the lowest for the eyes (71%). The whole face class has an accuracy of 90%. The rest of the classes have a bit lower of a lower score perhaps because they might have occlusions around these areas for the specific images. We notice a number of instances classified as either eyes, nose or eyes nose are actually part of the background. One reason could be that the parts cover a small area of the face so the model has difficulty depicting their individual features. It could also be that the model detects an object that resembles a face in the background. Overall, the performance is very good, the validation images contain faces of various scales so that means that the model is quite capable of detecting most of them. The overall accuracy is 83.1%.

Furthermore, there are two test sets with unseen images and labels, from the MERL-RAV and the WFLW dataset. The first one is considered easier as it contains clear faces without much background noise while the second one is more diverse and challenging, containing random images depicting one face in various settings.

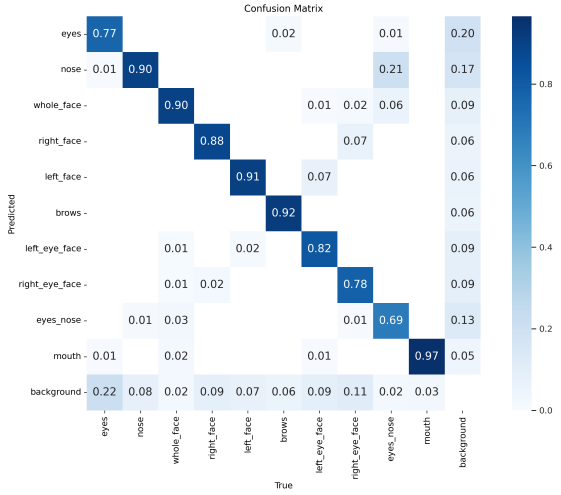

Figure 19: Confusion matrix on the MERL-RAV test set for first model

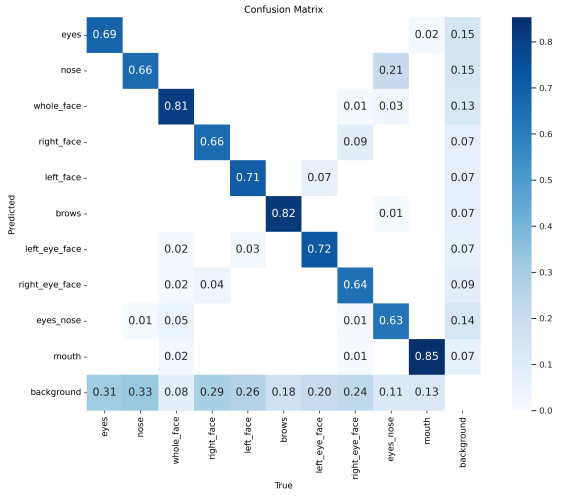

Figure 20: Confusion matrix on the WFLW test set for first model

Observing Figure 19, the classes mouth, brows and left face have higher accuracy while eyes-nose, eyes and right-eye-face have the

lower scores. Overall, the accuracy is high for all classes at 85.4%. For the eyes class we have a similar amount of false positives (22%) and false negatives (23%), so the eyes are either completely missed or being detected where there is background. Additionally for 21% of the predictions of nose, the ground truth is eyes-nose due to the overlap of these classes. From the WFLW dataset, the mouth, brows and the whole face have higher accuracy while the right-eye-face and eye-nose classes have the lowest. Furthermore by observing the bottom row, there are quite a few instances where face parts are predicted as background while in reality they belong to a class, meaning the model misses some faces or face parts. However this is expected given the quality and noise found in these images. The overall accuracy is 74.7%, naturally lower than the previous case. These results are in line with the expectations given the structure and difficulty of the datasets.

Table 4 illustrates the models' performance on different metrics for the 10 class RGB model. For the validation set the values are high with a noticeable drop in the mAP 0.5-0.95 which is expected as the IoU threshold increases. The performance is similar in the MERL-RAV dataset where all the values are consistently higher than for the WFLW. For both datasets, the precision is higher than the recall meaning there are less false positives than false negatives. The model misclassifies objects that are not faces in fewer cases but it may miss more faces that are present.

Set	P	R	F1	mAP50	mAP50_95
Val	0.919	0.906	0.912	0.939	0.633
MERL RAV	0.910	0.902	0.906	0.935	0.618
WFLW	0.808	0.762	0.784	0.802	0.506

Table 4: Precision, recall, mAP (over all classes) on the validation and two test sets for the RGB model trained on 10 classes

We also evaluate the 10 class model only on the whole face class, making it a binary classification task and the results are shown on Table 5.

Set	P	R	F1	mAP50	mAP50_95
Val	0.958	0.939	0.948	0.981	0.872
MERL RAV	0.940	0.947	0.943	0.981	0.867
WFLW	0.842	0.881	0.861	0.892	0.765

Table 5: Precision, recall, mAP on the validation and two test sets for the RGB model trained on 10 classes and evaluated with the whole face class

Observing Table 5 we see a consistency on the models' performance on the datasets, meaning it still performs better on the validation set and the MERL-RAV test set. Comparing the two tables above we notice that the performance is improved for the whole face class across all metrics. This means the model is better at detecting only the face compared to the other classes. This can be explained as the model can pick up various features to identify a face, which can be easier to distinguish. Moreover the face naturally covers a larger part of the image and contains all the other classes, without having a big overlap with one of them.

4.4.1 *One Class Model*

We train the same model on a single class, the whole face, to compare the results between the 10 class model and the single class model, and to later evaluate the CAM based models. Out of the 1620 validation samples, 1617 faces are detected correctly, 35 are false positives (predicted faces but are actually background) and 3 are false negatives (predicted background but is face). Note again that an image on the validation set can contain more than one face. Comparing to the 10 class validation set, we have 1618 correct detections, 32 false positives and 2 false negatives. Both the models detect objects in the background that are not faces and this can be because of the features the model captures and the similarity of a background object to a face. The complexity of the 10 class model slightly increases the number of missed detections as the model picks up different features to separate the classes but can miss an actual face.

Table 6 presents the results for the validation and the test sets.

Set	P	R	F1	mAP50	mAP50_95
Val	0.993	0.994	0.994	0.995	0.946
MERL RAV	0.991	0.990	0.990	0.995	0.939
WFLW	0.925	0.860	0.891	0.956	0.870

Table 6: Precision, recall, mAP on the validation and two test sets for the RGB model trained on one class

Using the whole face as the only class during training yields higher performance than training on 10 classes, when comparing Table 5 and 6. Comparing between the datasets, we see that for the model trained on one class, the precision and recall for MERL-RAV are nearly equal (99.1%), and are comparable to the previous model. Since the images on this dataset have better quality, then it is logical that the model can detect both the whole face and distinguish among the rest of the face parts as well. For the WFLW on one class, the recall value (86%) is lower than the precision (92.5%), following a similar pattern to the previous case. Comparing to the multiclass model we notice a similar trend, since WFLW is more complicated and contains variable poses, the multiclass model struggles to perform accurate classification. Some face parts naturally overlap so it is no surprise that all the metrics' values are higher for the single class model. The only value that is higher on the 10 class model is the recall of the whole face class in the WFLW dataset (88.1%) than (86%) for the single class one. This means there are a few more false negatives in the latter so the model can miss some faces. These faces can be more challenging so since there are no more classes to detect then this increases the false negatives. An additional explanation for the higher values of the first model is that the 10 class model can capture common features within classes and give less importance to class specific features. Thus, it's easier to miss some faces. The F1-score in Table 6 for the validation set is at 99.4%, for the MERL-RAV it drops slightly to 99.0% and lastly for the WFLW as expected it drops to 89.1%. The training and validations loss and metrics plots are found in Figure 25 in the Appendix.

Additionally, the models are tested on the unannotated Hand Over Face dataset for whole face detection. For testing, we set the

confidence threshold to 0.4 and the IoU threshold to 0.6 to filter out low confidence detections which could be irrelevant. We choose a confidence of 0.4 because from Figure 17, the green curve that corresponds to the whole face class peaks at that point. Out of the 4384 faces in the dataset, the 10 class model detects 3985, so 90.9% of the faces. The model trained on the whole face class detects 3701 faces, so 84.4%. This can be an indication that adding more classes related to the face can help the model localize it that is why the 10 class model shows better performance than the model trained on the whole face class only.

By visually inspecting the remaining frames without a detected whole face, the participants had almost fully covered their face so it makes sense why the model couldn't locate it.

4.5 Part 2: CAM based model Training Results

In the second part of the implementation, GradCAM++ and HiresCAM heatmaps are extracted using the weights of the trained model on the 10 facial parts. The heatmaps are extracted (1) by getting the average heatmap over all class activations and (2) by getting one heatmap per class and stacking them together. This means there can also be blank heatmaps if there are no detections for a class. Thus, we have four models. Table 7 presents the results for the validation and the test sets for both CAM methods. The loss plots are found in the Appendix (Figures 27, 28, 29, 30).

CAM Model	Set	P	R	F1	mAP50	mAP 50_95
Grad++ (avg)	Val	0.983	0.974	0.978	0.993	0.733
	MERL	0.977	0.979	**0.978**	0.993	0.732
	WFLW	0.855	0.786	0.819	0.828	0.530
Hires (avg)	Val	0.943	0.936	0.939	0.960	0.573
	MERL	0.939	0.963	**0.949**	0.971	0.562
	WFLW	0.765	0.778	0.771	0.781	0.379
Grad++ (10 class)	Val	0.975	0.979	0.977	0.992	0.740
	MERL	0.969	0.98	**0.974**	0.993	0.730
	WFLW	0.868	0.789	0.827	0.832	0.514
Hires (10 class)	Val	0.948	0.963	0.955	0.969	0.568
	MERL	0.938	0.961	**0.949**	0.971	0.562
	WFLW	0.808	0.848	0.828	0.827	0.406

Table 7: Precision, recall, mAP on the validation and two test sets for the GradCAM++ model and the HiresCAM based models. Heatmaps are produced by averaging over all predicted classes or only on the whole face class

The results indicate that the performance of the four models is comparable. For all cases, the MERL-RAV dataset displays a higher F1 score with values over 94.9%. The highest F1 score (97.8%) is in the GradCAM++ model trained on the average heatmap and the two lowest F1 scores (both 94.9%) come from the HiresCAM models. It's expected that this test set performs better as it contains frontal and slightly rotated faces with good lighting conditions and is not as random as the WFLW. The latter has F1 scores ranging from 77.1% to 82.8%, the lowest score (77.1%) coming from the HiresCAM model trained on the average heatmap and the best one (82.8%) from the HiresCAM model trained on the 10 class heatmaps. Analysing the mean average precision at an IoU threshold from 0.5 to 0.95, we can observe that both the GradCAM++ based methods have higher scores than the HiresCAM methods. For the MERL-RAV set these are at the scale of 70%, and at 50% for the WFLW. Meanwhile for the HiresCAM models the respective values are at 50% and at 38-40%. This indicates that GradCAM++ based models show increased performance while the complexity of the data increases while the HiresCAM based equivalents suffer a larger drop in average precision. This trend is also found when looking at the mean average precision score at 0.5 threshold. The GradCAM++ based models' score is decreased by around 20-30% as the threshold increases, whereas for the HiresCAM this is around 30-40%.

These findings can be explained by the nature of the heatmaps produced by the methods. The GradCAM++ heatmaps highlight various regions that play a role in the prediction, and they can also indicate contextual regions that are not so relevant. On the other hand, the HiresCAM heatmaps are sparse and only highlight the important part without the surroundings. As the IoU threshold increases, the HiresCAM predictions, being less detailed, can have a negative impact on the mAP. The next part of the thesis tests the models on various degrees of occlusion and then it can be easier to make conclusions as to which one works better for the task.

4.5.1 *Testing on missed detections from MERL & WFLW*

The four models are tested on the images of MERL-RAV and WFLW test sets where the face was not detected. First, the images are selected which gives 25/804 missed face detections for MERL-RAV and 39/792 for WLFW. Then the heatmaps are extracted and the empty ones are removed. Empty heatmaps mean that the model failed to detect any class for the particular image. This process returns 25 heatmaps for the first test set meaning none were removed, and 34 for the second. The optimal confidence threshold is determined from running validation on the sets, and the NMS IoU is set to 0.6. Table 8 presents the detection results. Precision measures the percentage of correct predictions and coverage the proportion of total heatmaps that the model is able to make predictions for.

CAM Model	Set	Conf	F1	Total Preds	Precision (Test Set)	Coverage (Test Set))
Grad++ (avg)	MERL	0.713	1	25	100%	100%
	WFLW	0.165	0.41	17	88.24%	50%
HiresCAM (avg)	MERL	0.697	0.96	23	100%	92%
	WFLW	0.391	0.62	23	86.96%	67.6%
Grad++ (10class)	MERL	0.663	1	0	0	0%
	WFLW	0.645	0.57	0	0	0%
HiresCAM (10class)	MERL	0.377	0.94	10	40%	40%
	WFLW	0.344	0.65	10	60%	29.4%

Table 8: Optimal confidence, max F1 score, total detections, correct detections and coverage when testing the models for the optimal confidence value coming from the respective F1-confidence curve obtained during validation for the 4 different CAM models on MERL-RAV & WFLW

- **GradCAM++ (avg):**
  MERL-RAV: 100% correct detections out of 25 heatmaps.
  WFLW: 15 correct detections out of 34, with 17 total detections, and 2 incorrect ones.

- **HiresCAM (avg):**
  MERL-RAV: 23 correct detections out of 25, with no incorrect ones.
  WFLW: 20 correct detections out of 34, with 23 total detections, and 3 incorrect ones.
- **GradCAM++ (10class):**
  MERL-RAV: 0 correct detections out of 25, with 25 incorrect ones.
  WFLW: 0 correct detections out of 34, with 25 incorrect ones.
- **HiresCAM (10class):**
  MERL-RAV: 4 correct detections out of 25, with 10 total detections, and 6 incorrect ones.
  WFLW: 6 correct detections out of 34, with 10 total detections, and 4 incorrect ones.

From these insights we can see that the avg models perform better especially for the MERL-RAV dataset. The GradCAM++(avg) does not miss any detections on MERL-RAV and performs decently on the more complex WFLW. Its 10class counterpart however does not detect any faces, which might indicate a flaw in the model. The HiresCAM (avg) model also shows strong performance, having detected more faces correctly for the WFLW dataset, and it's total correct detections equal to 86.96% which is close to the GradCAM++(avg) model at 88.24%. The HiresCAM(avg) model though has more coverage meaning it detected more heatmaps to begin with. The HiresCAM (10class) models shows room for improvement but performs better on the WFLW dataset (60%) than on the MERL-RAV (40%), and has some more coverage for the WFLW dataset.

4.5.2 *Testing on Hand Over Face*

We tested the four CAM models on images of the Hand Over Face that had no detections for the first 10-class model, so that gave a starting point of 400 images. Again, we extracted heatmaps and then filtered out the empty ones, which left us with 215 of them to examine. To establish a baseline we ran the detection with the same settings as before, so that's a confidence threshold of 0.4 and an IoU threshold of 0.6.

CAM Model	Initial Heatmaps	Total Detections	Remaining
GradCAM++ (avg)	215	28	187
HiresCAM (avg)	215	112	103
GradCAM++ (10 class)	215	0	215
HiresCAM (10 class)	215	35	180

Table 9: Total initial count of non-empty heatmaps, total detections and number of remaining heatmaps without detections for the 4 different CAM models on Hand Over Face

We inspected the detection results for the average models and located 2 errors in GradCAM++ model where the model had detected a face on the bottom right part of the image. Given the dataset, we expect the face to cover a small area around the center to top of the image and looking at the original image the model had indeed failed to detect a face there. In both average and 10 class based models, HiresCAM has a superior detection rate. Specifically for the average models, it has four times more detections than GradCAM++. The 10 class GradCAM++ model fails to make any detections, while the 10 class HiresCAM detects 35 objects as faces. After inspecting the coordinates, they seem to be correctly localizing the image around the center of the frame.

Figure 21 shows the original image, the GradCAM++ and the HiresCAM average models' prediction.

Figure 21: Original image, GradCAM++ and HiresCAM heatmaps and face detections on an image from HandOverFace

4.6 Application on SLAPI data

In order to address the previous research question, the proposed implementation was evaluated using frames extracted from videos of preterm infants. The selection of frames was based on four categories of occlusions. Each video was thoroughly examined, and a frame was chosen when there was noticeable variability among the frames, such as when the infant started moving. It should be noted that the infants were mainly asleep during the majority of the recording.

To assess the performance of the models, the selected frames were manually annotated with a reliable estimation of the bounding box encompassing the infant's face. The annotations took into consideration four types of occlusions.

By conducting this evaluation and annotation process, we aimed to provide a comprehensive evaluation of the proposed implementation's performance in detecting and handling different occlusions in the context of preterm infants.

- No Occlusions: the infant has no tubes around the face (rare)
- Easy: There are occlusions such as hat or a slim tube on one side of the face, but the face shape is clear, good lighting conditions, small amount of rotation.
- Medium: There are possibly heavier occlusions around the face, movement from the infant, a change in lighting conditions, blurry frame.
- Hard: the infant is heavily occluded from its blanket and hat, can have a bigger device attached to it that blocks the view, fully covers its head with its hand, or the room is darker and the frame is blurry.

Table 10 presents different metrics on the various occlusions. The RGB model is tested only on the whole face class, for an IoU of 0.6 and default confidence of 0.001 as according to the authors a higher value affects the mAP. The precision, recall and F1 scores are calculated based on the best confidence value that gives their best combination.

Table 11 illustrates the total number of correct detections for the whole face class and the lowest confidence score for the correct detections for the images.

Set	Count	P	R	F1	mAP50	mAP50_95
No Occl	10	0.989	1	0.994	0.995	0.454
Easy	64	0.970	0.750	0.850	0.775	0.446
Medium	203	0.862	0.485	0.621	0.543	0.228
Hard	210	0.667	0.220	0.331	0.270	0.107

Table 10: Precision, Recall, mAP on the SLAPI dataset categories for the RGB model on the face class

Set	Total	Correct	Lowest conf	Remaining	Non-empty heatmaps
No Occl	10	10	0.570	0	0
Easy	64	49	0.400	15	3
Medium	203	130	0.400	72	25
Hard	210	68	0.400	142	32

Table 11: Total count, total correct detections and lowest confidence of correct detections for the 4 different occlusion sets of SLAPI

CAM Model	Set	P	R	F1	mAP50	mAP 50_95
Grad++ (avg)	Easy	0.477	0.330	0.418	0.372	0.074
	Medium	0.185	0.115	0.142	0.101	0.031
	Hard	0.917	0.062	0.117	0.097	0.029
Hires (avg)	Easy	0.954	0.667	0.785	0.697	0.166
	Medium	0.491	0.462	0.476	0.364	0.090
	Hard	0.341	0.312	0.326	0.211	0.048
Grad ++ (10 class)	Easy	0.987	1	0.993	0.995	0.233
	Medium	0.585	0.192	0.289	0.201	0.052
	Hard	0.344	0.125	0.183	0.093	0.031
Hires (10 class)	Easy	0.939	0.667	0.780	0.675	0.372
	Medium	0.565	0.500	0.531	0.444	0.182
	Hard	0.601	0.500	0.546	0.456	0.126

Table 12: Precision, Recall, mean Average Precision on different levels of occlusions for the SLAPI dataset for the GradCAM++ model and the HiresCAM based models. Heatmaps are produced by averaging over all predicted classes (avg) or by stacking the heatmap for each of the classes into a tif

There are very few frames where the infant doesn't have devices or a hat, and the face is fully visible. For these frames, the model has a recall equal to 1, meaning zero false negatives. As the difficulty level of the occlusions rises, the precision, recall, F1, and mAP scores naturally drop. For the hardest occlusions it can be observed that the precision is at 66.7% while the recall is at 22.0%. This is interpreted as the model having a high level of accuracy when detecting true positives, while having lower sensitivity. This means a significant number of faces remains undetected. Given the selection of frames, this makes sense as the faces can be quite challenging to find and the image quality is lower in some cases. It's a positive fact that when a face is detected, in most cases it is truly a face. For the frames where the face failed to be detected, we can apply the second part of the pipeline to examine whether it helps predict faces that were previously missed. Thus, we will examine the four CAM models with the easy, medium and hard occlusion test sets. However, not all frames have detections and thus have empty heatmaps so we apply the CAM object detection only on non-empty heatmaps.

4.6.1 *Validation results on SLAPI*

Examining Table 12 there are various conclusions we can draw. First of all the Easy dataset contains only four images so it is not a significant value from which to draw definite conclusions but we will try to make comparisons.

The *GradCAM++ (avg)* model overall exhibits a mediocre performance and the mAP@50-95 is significantly low for the three cases as the IoU threshold increases. What's interesting is that for the Hard occlusions the model displays a very high precision (91.7%) meaning its positive predictions are mostly correct however it misses a large number of faces as the recall value is at 6.2 %.

The *HiresCAM (avg)* model has a more balanced expected performance. For the Easy occlusions the precision is very high and the recall is good so that leads to a robust F1 score. Going to Medium occlusions there's an expected drop in performance and the precision and recall have similar values. Moving to the Hard occlusions, the performance declines but the overall F1 score is still higher than the average GradCAM++ method.

The *GradCAM++ (10 class)* model has almost perfect scores for the metrics in the Easy case (recall = 1), but displays significant drops in the Medium and Hard cases, with the recall in the Hard case being notably low. This indicates that the model does not perform as expected and is unreliable.

The *HiresCAM ++ (10 class)* model has overall good performance. The Easy occlusions have very high scores on the precision and recall. The Medium occlusions display moderate performance but the recall value is higher than for the GradCAM++ 10 class model. This is also the case for the Hard occlusions, the precision is higher than the Medium occlusions, the recall score is the same 50% for the Medium and Hard occlusions but still higher than the GradCAM++ model.

Overall, the HiresCAM models show a more balanced performance, doing well in the Easy case and dropping as the occlusion level is higher. Comparing the average heatmap model and the 10 class based model, the latter shows a higher performance for the more difficult cases which contain more samples, so it can be a promising choice for improving face detection. The GradCAM++ based methods have more highlighted regions in their heatmaps, making them cover more areas that could play a role in the detection. This does not mean they are precise, and the more defined heatmaps of HiresCAM make the detection easier.

4.6.2 *Detection results on SLAPI*

When running the validation on labelled data, a very low confidence threshold is used for the computation of metrics. In practise, our data will not be labelled so we examine the performance with a higher confidence threshold. The F1-confidence curve from the validation can point to the optimal confidence value that maximizes precision and recall. Table 13 presents the number of correct detections for every model, within the different occlusion levels

when setting the optimal confidence value. We use the same IoU threshold as before which is 0.6.

CAM Model	Set	Conf	F1	Total Preds	Precision (Test Set)	Coverage (Test Set))
Grad++ (avg)	Easy	0.099	0.450	0	0%	0%
	Medium	0.021	0.150	6/25	33.33%	24%
	Hard	0.181	0.120	3/32	66.67%	9.37%
Hires (avg)	Easy	0.592	0.790	1/3	100%	33.33%
	Medium	0.364	0.490	21/25	100%	84%
	Hard	0.188	0.330	31/32	87.1%	96.87%
Grad++ (10class)	Easy	0.377	0.990	0/3	0%	0%
	Medium	0.325	0.300	0/25	0%	0%
	Hard	0.339	0.190	0/32	0%	0%
Hires (10class)	Easy	0.545	0.760	1/3	100%	33.33%
	Medium	0.294	0.560	10/25	90%	40%
	Hard	0.204	0.550	9/32	77.78%	28.12%

Table 13: Number of correct detections when testing the models for the optimal confidence value coming from the respective F1-confidence curve obtained during validation

From 13 we see that *GradCAM++ (avg)* struggled with coverage but showed decent precision for the predictions made in the Medium (33.33%) and Hard (66.67%) occlusion sets. Meanwhile it failed to make any detections for the Easy occlusion set.

The *HiresCAM (avg)* model has the most heatmaps for the Medium and Hard occlusions and also detects more correct faces. For the Easy occlusions it only makes one detection which is correct. For the Medium occluded faces, 21/25 images have a detection present and all of them are correct so the model indeed detects the faces for these images. For the Hard occlusions the model makes 31 detections for the 32 images, and 27 of them are correct (87.1%).

It's counterpart 10 class model has a lower performance, but also manages to detect 1 face correctly for 1 out of the 3 heatmaps that represent a face with easy occlusions. For the Medium occlusions it makes 10 detections and 90% of them is correct, while for the Hard occlusions it makes 9 detections and 77.78% are correct. The GradCAM++ model for the 10 classes fails to make any detections while the average model still makes fewer detections compared to HiresCAM.

Interestingly, we lowered the detection confidence threshold back to 0.001 and the model was able to make some correct detections. For the GradCAM++(avg) Easy set there are 2 correct detections. For the GradCAM++ (10class) model there were again 0 correct detections for the Easy set, 1 correct detection for the Medium set and 4 correct for the Hard set. These detections are low confidence so not very reliable, but the confidence threshold issue should be further examined to see if there is an internal flaw in the calculations in YOLOv5. In any case these numbers do not show any significant increase in performance and cannot be used to make any comparisons.

In summary, HiresCAM models seem to perform much better and are a promising method to guide face detection under occlusions as they provide clear heatmaps highlighting pixels around the face. HiresCAM(avg) performed admirably across all occlusion levels, with very high precision and coverage, especially for Medium and Hard occlusions. Even if the trained model fails to make a detection, it can still be possible to localize the face by drawing a bounding box around the highlighted regions and comparing the boxes coordinated with the coordinates of other boxes to estimate whether the highlighted region makes sense. Training under different model configurations could improve the results of the HiresCAM models and further enhance their outcome.

5 Conclusions & Future Work

This section addresses the research questions and conclusions we reached, and discusses potential future improvements.

5.1 Conclusions

We will address the research questions one by one in this section.

Which facial parts can we split the face in and what is the models' performance?

The answer to the first part of this question relies on the available annotated datasets we could find and access. They followed the 68 landmark iBUG annotation protocol meaning they had landmarks for the face contour, eyes, nose, mouth and most of the images had faces in various poses. Thus, the most meaningful classes corresponded to each distinct facial part e.g. eyes, nose, and a combination of these such as right face, left face, eyes nose etc. We trained a YOLOv5 object detector model on a mixed dataset consisting mainly of frontal and partially rotated faces with various occlusions, face sizes and backgrounds. Our model overall has a very high performance, with balanced precision and recall values for the two test sets that vary in difficulty. For the MERL-RAV test set compromised of mostly clear frontal faces, the F1-score is at 93% and for the WFLW dataset containing more random images and variations, the F1-score is at 77.6%. Overall we can conclude the model has promising performance.

Which methods produce the most useful heatmaps and what is their performance on the same datasets as above?

For the next part we explored various available explainability methods, and decided to wok with GradCAM++, an improvement of GradCAM and HiresCAM, a method that produces very precise heatmaps. There are two ways to extract heatmaps, one is by averaging the heatmaps produced for every class into one grayscale image, and the second is by stacking the separate heatmaps into a 10 channel grayscale image. The aim of the second method is to overcome the occlusion problem so even if one class has no detection, a heatmap can still be produces and give an estimate of the location of the face. We trained four YOLOv5 based models on these heatmaps with only the whole face as ground truth in this case. We evaluate the performance on the same data as the above model. The performance is very good for all the cases, and naturally the WFLW dataset is the most complex one so we observe a slight drop in the performance. Since SLAPI data is also complex, we think that the 10 class variants seem to handle this data better as the F1 scores for WFLW are higher than their counterparts.

How does the first face detector perform on frames of infants with various occlusion levels?

We test the first model on a subset of SLAPI frames we manually annotated with the whole face. Most of the frames contain medium and hard occlusions as expected such has hands, tubes, hats. The

confidence is set to 0.4 and the IoU threshold to 0.6 for the detection as we aim to avoid low confidence predictions. We see that for the non-occluded faces the model performs flawlessly with a recall value equal to 1 meaning zero false negatives. As the occlusion level increases, the model struggles more and more, having a relatively low F1-score of 33.1% for the 210 highly occluded frames.

Whether the use of explainability based models has a positive effect on detecting more faces.

Next, for the frames where the model failed to detect a face, we extract heatmaps using the four explainability techniques. Still, not all of the frames have detections so this leaves us with fewer valuable heatmaps where we can test the four trained models. To get the optimal confidence threshold per case, we run validation with a very low confidence and observe the F1-confidence plot. We then run detection with that threshold and a stable IoU of 0.6. The results show that the GradCAM based models struggle with lightly-occluded frames probably because of the nature of their heatmaps. The HiresCAM avg model manages to detect more faces for all occlusion levels, most of them correctly compared to the methods. Its 10 class counterpart also displays a good performance, while it manages to detect less faces it also correctly detects some of them.

We can conclude that HiresCAM based models are a promising alternative to detecting faces that the 1st model missed. It is worth it to have more data to draw stronger conclusions, but using explainability methods as a second step in face detection looks like a good option to ensure more faces are detected correctly.

5.2 Future Work

There are many different aspects of future work to do.

When considering the first model, it is always possible to a add more data with more face poses to ensure a larger degree of variability. Also adding another face class such as ears which might be visible in infant videos but not annotated in our training datasets could be a good idea. One aspect to consider is that our training data had one annotated face per image but some images contained more faces. This can affect YOLO's performance as it looks at the entire image and learns contextual information about the classes. We made sure to not include such images in the validation set but finding and creating a whole dataset with only one face per image proved to be quite challenging. One model limitation was the hyperparameter evolution which had to be terminated early due to technical difficulties and limited time. Perhaps running the algorithm for more time could have yielded optimal parameters for our model.

When considering the second model, we extracted the heatmaps from the second to last layer of YOLO, as the last one outputs the detections before NMS. The assumption is that this layer will contain all the relevant detailed features so the heatmaps will correspond more closely to important regions for the prediction. Experimenting with the target layer would be an interesting idea to see how the heatmaps change and ensure they are a meaningful input to the second model. Furthermore, due to limited time and resources we did not run hyperparameter evolution to evaluate different optimizers and learning rate values but instead used default ones from YOLO. In future work, performing hyperparameter evolution could yield valuable insights into optimizing model performance and enhancing the quality of the results. By experimenting with different optimizer and learning rate configurations, we could fine-tune the model for better overall performance, potentially leading to more accurate and reliable face predictions. Apart from that, as new explainability methods are being established in literature, it would be an intriguing prospect to see if a new future method could work better for our case.

Another aspect to consider is the practical application of this setup. To ensure its usability in real-world scenarios, we need to evaluate the efficiency and practicality of this pipeline, considering both its speed and accuracy. Testing with a larger amount of data will also give more valuable insights to the models performance and help choose a suitable model. Knowing that each frame contains a single face, ensuring a good balance of true positives while minimizing false negatives is important. Leveraging the temporal nature of the videos can be used as an advantage in this case. Even if the face is missed in some cases, information from the previous frames can be used to give an estimate of its location. The temporal continuity can improve the overall robustness and confidence of the model even when encountering highly occluded faces. Another idea would be to perform transfer learning or fine-tuning the models to a small labeled SLAPI dataset and evaluate the performance there. By doing so we can enhance the models' accuracy and make it more robust to the specific challenges of our application.

References

[1] D. Triantafyllidou and A. Tefas. "Face detection based on deep convolutional neural networks exploiting incremental facial part learning". en. In: *23rd International Conference on Pattern Recognition (ICPR*. IEEE, 2016.

[2] A. Elmahmudi and H. Ugail. "A framework for facial age progression and regression using exemplar face templates". en. In: *The Visual Computer* 37 (2021).

[3] H.K. Ekenel and R. Stiefelhagen. "Why is facial occlusion a challenging problem?" en. In: *International conference on biometrics*. Berlin, Heidelberg: Springer, 2009.

[4] L. Jiang-Jing et al. "A deep regression architecture with two-stage re-initialization for high performance facial landmark detection". en. In: *IEEE conference on computer vision and pattern recognition*. 2017.

[5] T.-Y. Lin. "Focal loss for dense object detection". en. In: *Proceedings of the IEEE international conference on computer vision*. 2017.

[6] J. Redmon. "You Only Look Once: Unified, Real-Time Object Detection". en. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[7] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: *arXiv preprint arXiv:2207.02696* (2022).

[8] H. Deshpande, A. Singh, and H. Herunde. "Comparative analysis on YOLO object detection with OpenCV". en. In: *International journal of research in industrial engineering* 9.1 (2020), pp. 46–64.

[9] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection". In: *arXiv preprint arXiv:2004.10934* (2020).

[10] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[11] T.-Y. Lin. "Feature pyramid networks for object detection". en. In: *IEEE conference on computer vision and pattern recognition*. 2017.

[12] K. He et al. "Deep residual learning for image recognition". en. In: *IEEE conference on computer vision and pattern recognition*. 2016.

[13] J. Uijlings. "Selective search for object recognition". en. In: *International journal of computer vision* 104 (2013), pp. 154–171.

[14] R. Girschick and al. "Rich feature hierarchies for accurate object detection and semantic segmentation". en. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[15] R. Shaoqing et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". en. In: *Advances in neural information processing systems 28*. 2015.

[16] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[17] J. Huaizu and E. Learned-Miller. "Face Detection with the Faster R-CNN". en. In: *12th IEEE international conference on automatic face gesture recognition*. 2017.

[18] S. Xudong, P. Wu, and S.C. Hoi. "Face detection using deep learning: An improved faster RCNN approach". en. In: *Neurocomputing* (2018), pp. 42–50.

[19] Shervin Minaee et al. "Going deeper into face detection: A survey". In: *arXiv preprint arXiv:2103.14983* (2021).

[20] D. Zeng, R. Veldhuis, and L. Spreeuwers. "A survey of face recognition techniques under occlusion". fr. In: *IET Biometrics* 10.6 (2021), pp. 581–606.

[21] M. Jones and P. Viola. "Rapid object detection using a boosted cascade of simple features". en. In: *IEEE computer society conference on computer vision and pattern recognition*. 2001.

[22] D. Jiankang et al. "Retinaface: Single-shot multi-level face localisation in the wild". en. In: *IEEE/CVF conference on computer vision and pattern recognition*. 2020.

[23] Kaipeng Zhang et al. "Joint face detection and alignment using multitask cascaded convolutional networks". In: *IEEE signal processing letters* 23.10 (2016), pp. 1499–1503.

[24] F. Yuantao et al. *Detect Faces Efficiently: A Survey and Evaluations*. fr. 2021.

[25] T.Y. Yin et al. "Focal loss for dense object detection". en. In: *ICCV*. 2017.

[26] Shuo Yang et al. "Wider face: A face detection benchmark". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5525–5533.

[27] R. Anurag et al. "Generating 3D faces using Convolutional Mesh Autoencoders". en. In: *CVPR*. 2018.

[28] Z. Yuxiang et al. "Dense 3D Face Decoding over 2500FPS: Joint Texture Shape Convolutional Mesh Decoders". en. In: *CVPR*. 2019.

[29] D. Jia et al. "ImageNet: A large-scale hierarchical image database". it. In: *IEEE*. 2009.

[30] W. Geeroms et al. *Audio-Visual Active Speaker Identification: A comparison of dense image-based features and sparse facial landmark-based features*. en. in 2022 Sensor Data Fusion: Trends, Solutions, Applications (SDF)., 2022.

[31] F. Khan. *Facial expression recognition using facial landmark detection and feature extraction via neural networks*. en. arXiv, 2018.

[32] K.S. Khabarlak and L.S. Koriashkina. *Fast Facial Landmark Detection and Applications: A Survey*. en. 2022.

[33] M. Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". en. In: *IEEE/CVF Conference*. 2018.

[34] Jianfeng Wang, Ye Yuan, and Gang Yu. "Face attention network: An effective face detector for the occluded faces". In: *arXiv preprint arXiv:1711.07246* (2017).

[35] Shiming Ge et al. "Detecting masked faces in the wild with lle-cnns". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2682–2690.

[36] C. Zhu. "Cms-rcnn: contextual multi-scale region-based cnn for unconstrained face detection". en. In: *Deep learning for biometrics* (2017), pp. 57–79.

[37] M. Zhu et al. "Robust Facial Landmark Detection via Occlusion-adaptive Deep Networks". en. In: *CVPR*. Long Beach, CA, USA, 2019.

[38] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[39] Aditya Chattopadhay et al. "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks". In: *2018 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2018, pp. 839–847.

[40] Rachel Lea Draelos and Lawrence Carin. "Hirescam: Faithful location representation in visual attention for explainable 3d medical image classification". In: *arXiv preprint arXiv:2011.08891* (2020).

[41] Ruigang Fu et al. "Axiom-based grad-cam: Towards accurate visualization and explanation of cnns". In: *arXiv preprint arXiv:2008.02312* (2020).

[42] Haofan Wang et al. "Score-CAM: Score-weighted visual explanations for convolutional neural networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 24–25.

[43] Mohammed Bany Muhammad and Mohammed Yeasin. "Eigen-cam: Class activation map using principal components". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–7.

[44] Timesler. *facenet-pytorch*. https://github.com/timesler/facenet-pytorch. 2021.

[45] Sefik Ilkin Serengil. *RetinaFace*. https://github.com/serengil/retinaface. 2021.

[46] Jiankang Deng et al. *insightface: 2D and 3D Face Analysis Project*. https://github.com/deepinsight/insightface. 2021.

[47] Arnaud Dapogny, Kévin Bailly, and Matthieu Cord. "Deep Entwined Learning Head Pose and Face Alignment Inside an Attentional Cascade with Doubly-Conditional fusion". In: *CoRR* abs/2004.06558 (2020). arXiv: 2004.06558. URL: https://arxiv.org/abs/2004.06558.

[48] A. Kumar et al. "Luvli face alignment: Estimating landmarks' location, uncertainty and visibility likelihood". en. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR*. 2020.

[49] Christos Sagonas et al. "300 faces in-the-wild challenge: Database and results". In: *Image and vision computing* 47 (2016), pp. 3–18.

[50] Sakher Ghanem, Ashiq Imran, and Vassilis Athitsos. "Analysis of hand segmentation on challenging hand over face scenario". In: *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*. 2019, pp. 236–242.

6 Appendix

6.1 Results of Face Detectors

Below are presented the results from running several face detectors on images of preterm infants obtained online. These contain various occlusions and have a large degree of variability.

RetinaFace

Figure 22 illustrates the results of RetinaFace on several images. In cases of heavier occlusion and rotation it can detect part of the face but fails to provide an accurate landmark estimation. The results also depend on the image resolution the pretrained network it uses was trained on. On the top right image where there are are multiple false detections the resolution is 1200 x 675 whereas the one below the resolution is 4032 x 1024.

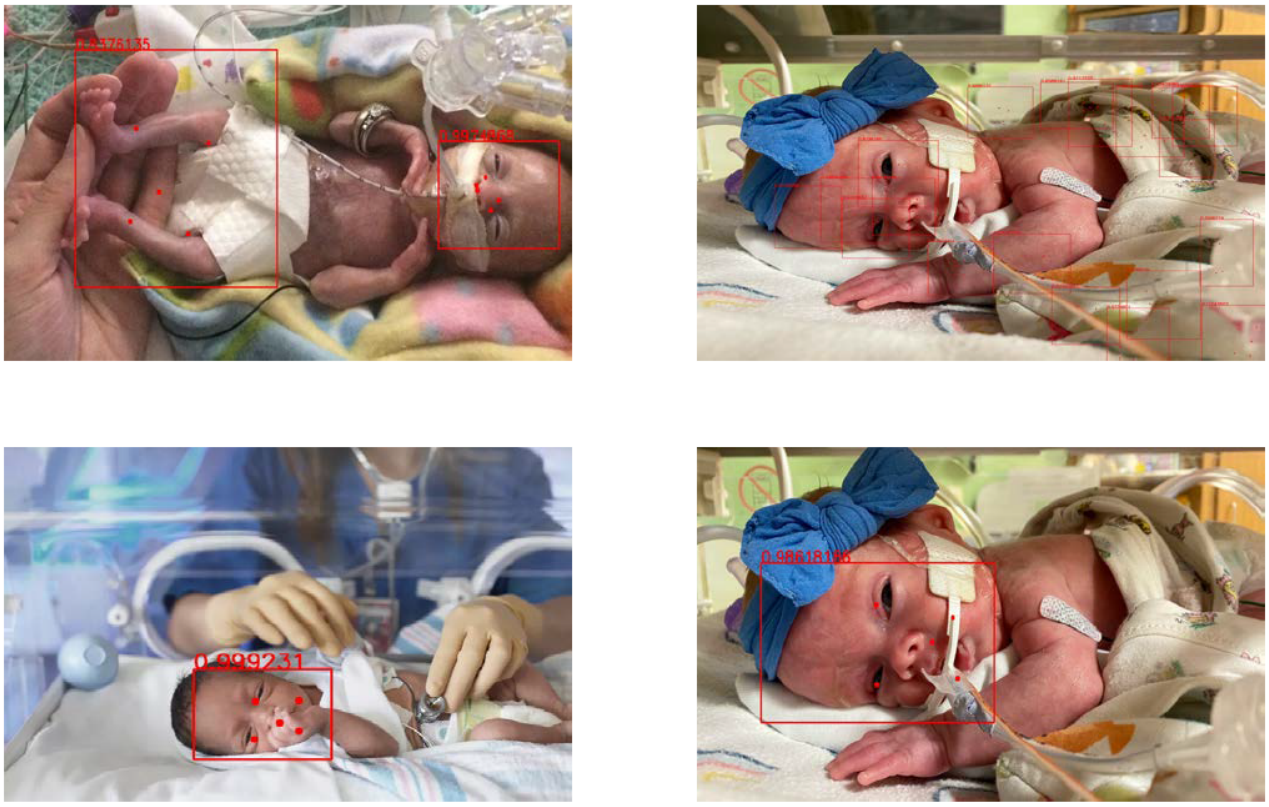

Figure 22: RetinaFace detections on stock photos of infants

MTCNN

Figure 23 is the only one where MTCNN could detect a face and landmarks. Looking closely one can observe 5 landmarks that have been predicted however they do not resemble the correct face parts. The face however is detected rather accurately.

Figure 23: MTCNN detection

6.2 First YOLO Model: Train, Loss, Metrics Plots

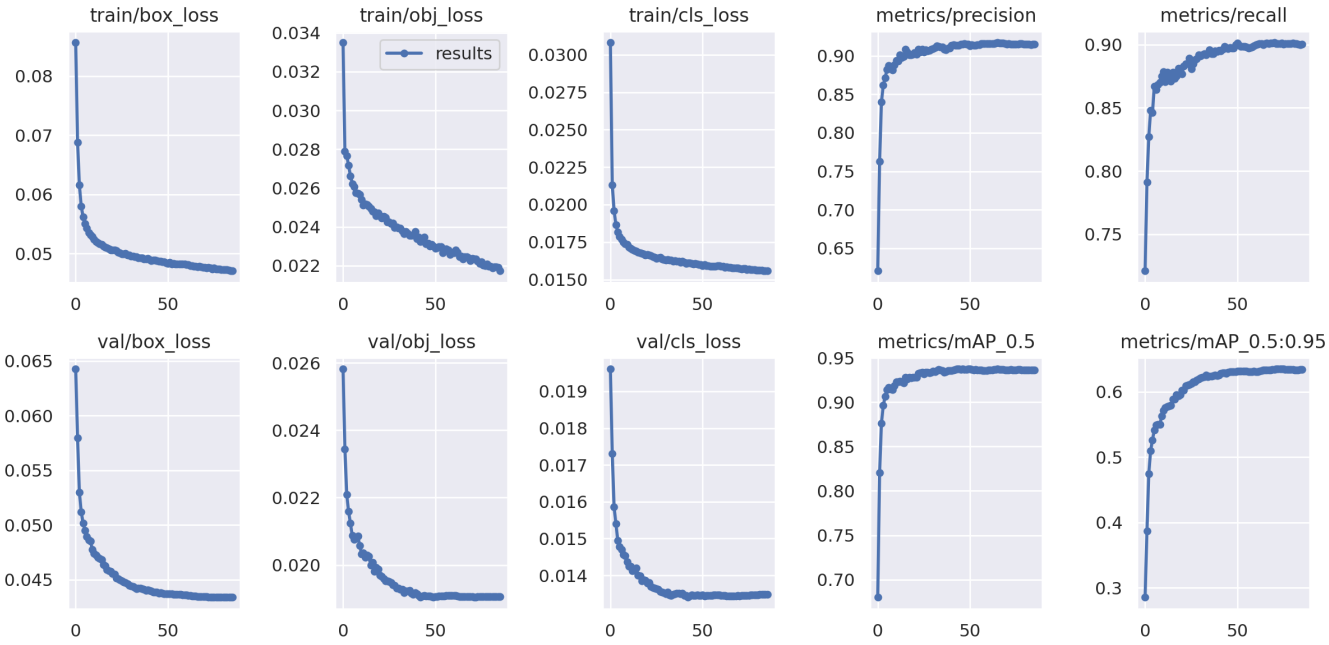

Figure 24: Loss plots and metrics for the 10 class RGB model

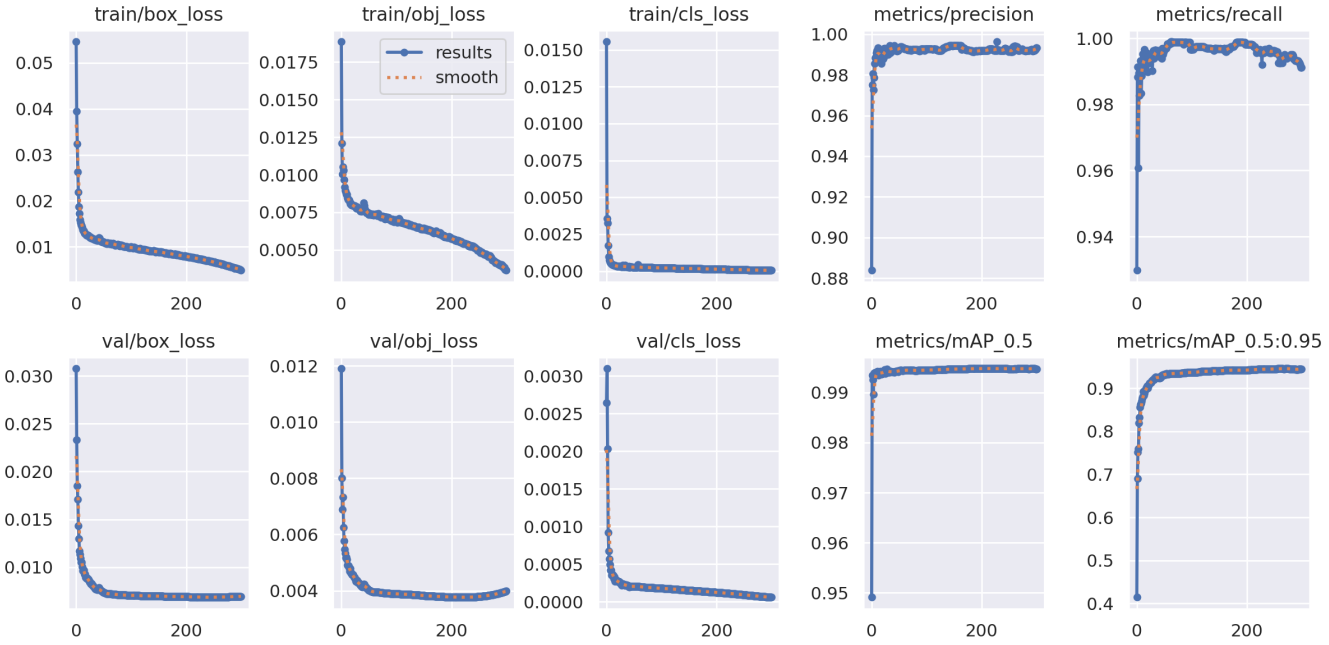

Figure 25: Loss plots and metrics for the 1 class RGB model trained only on the whole face class

6.3 Second YOLO model: Heatmaps example & Loss and Metrics plots for the four different models

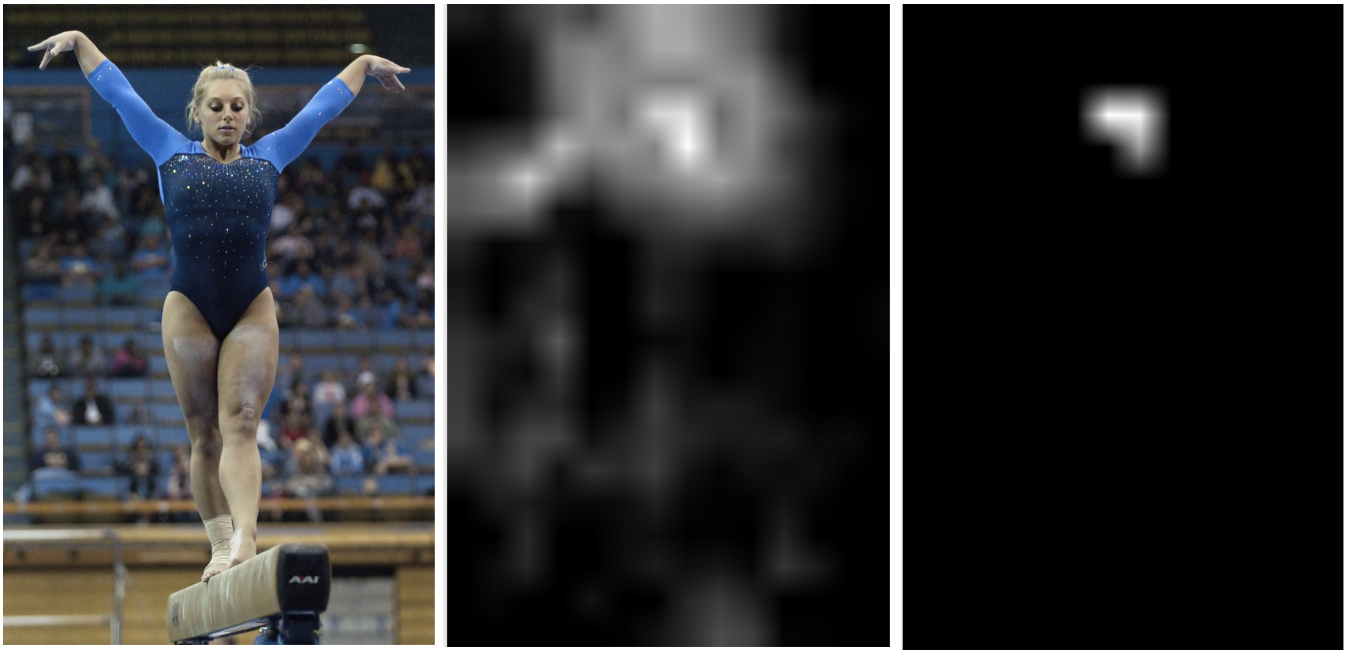

Figure 26: Example image 000023 from the training set. On the left is the original image, in the center the GradCAM++ heatmap and on the right the HiresCAM heatmap.

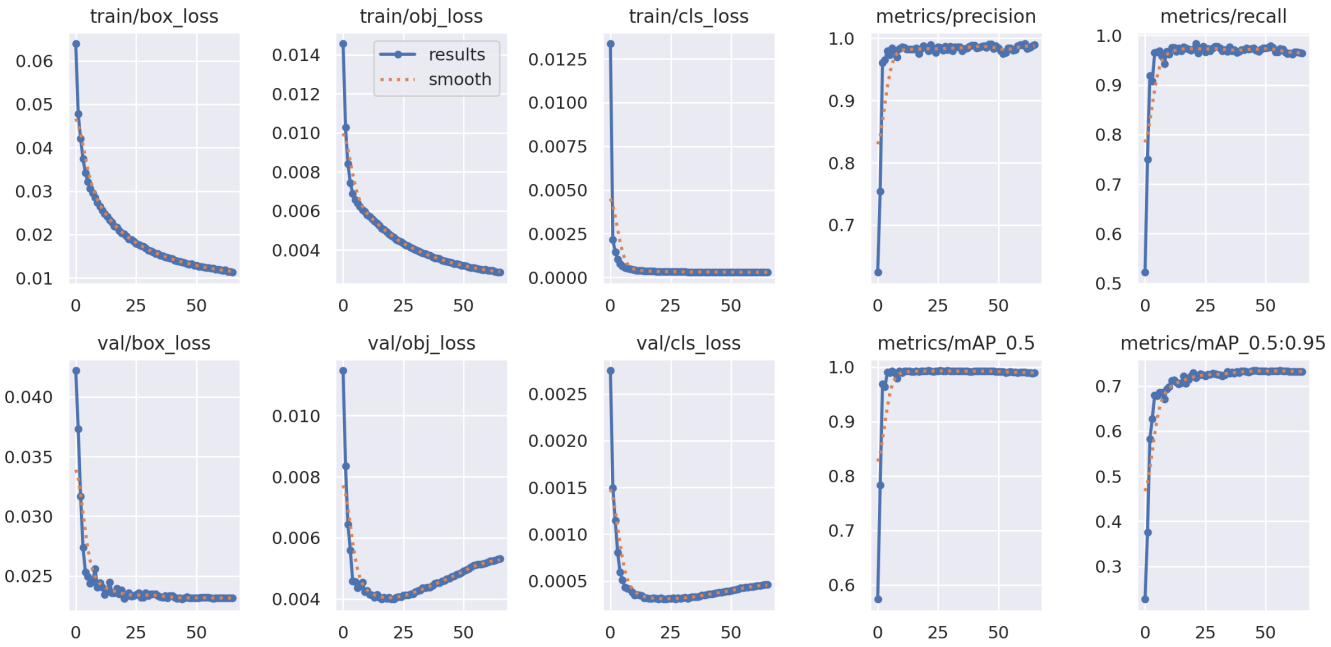

Figure 27: Loss and metrics plots for the GradCAM++ model trained with averaged heatmaps as input

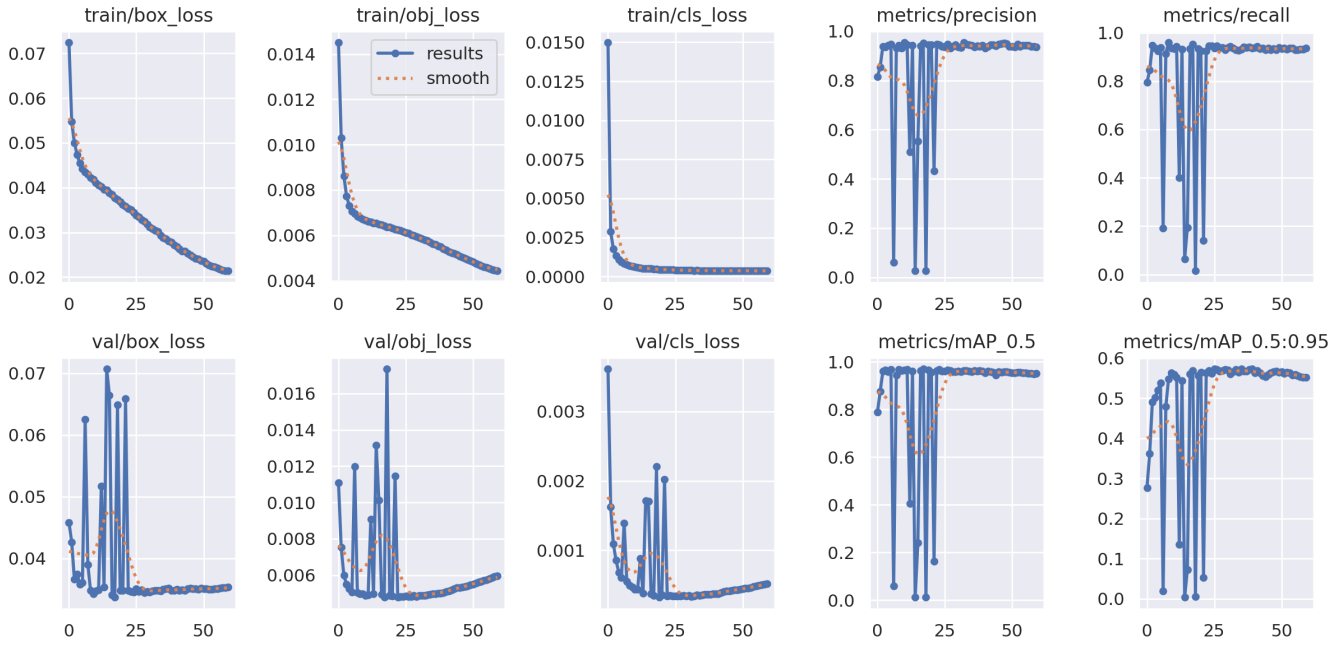

Figure 28: Loss and metrics plots for the HiresCAM model trained with averaged heatmaps as input

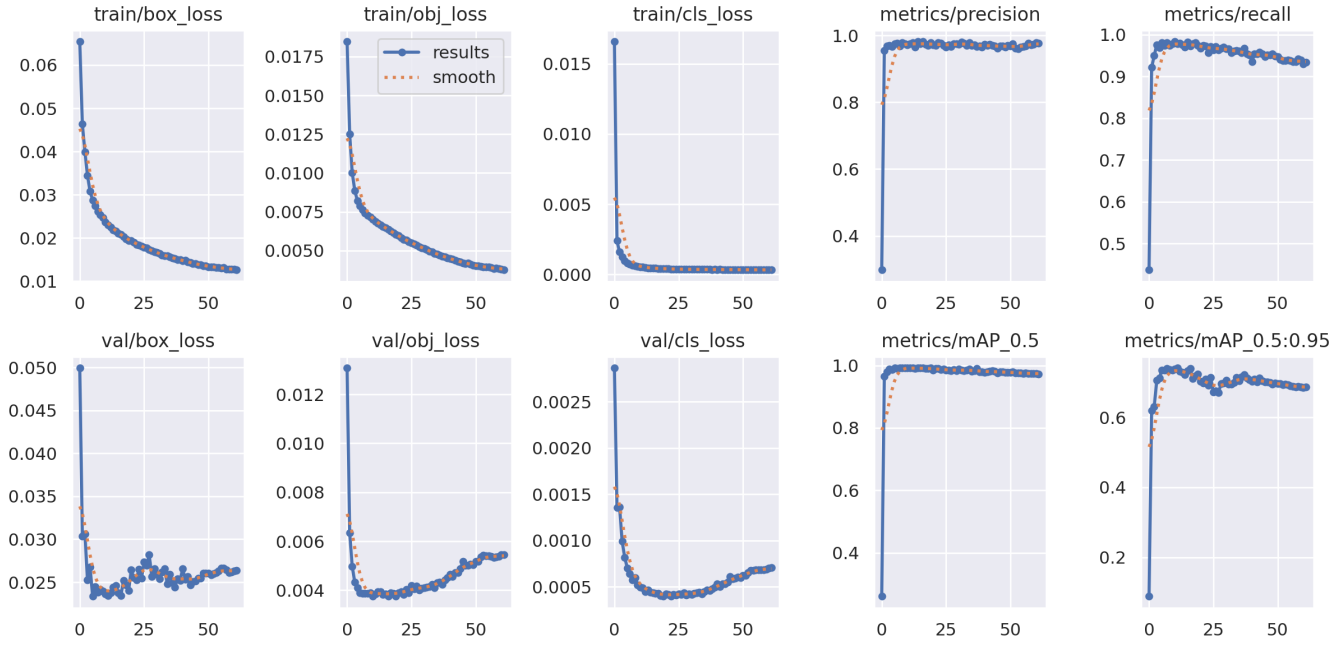

Figure 29: Loss and metrics plots for the GradCAM++ model trained with heatmaps per class

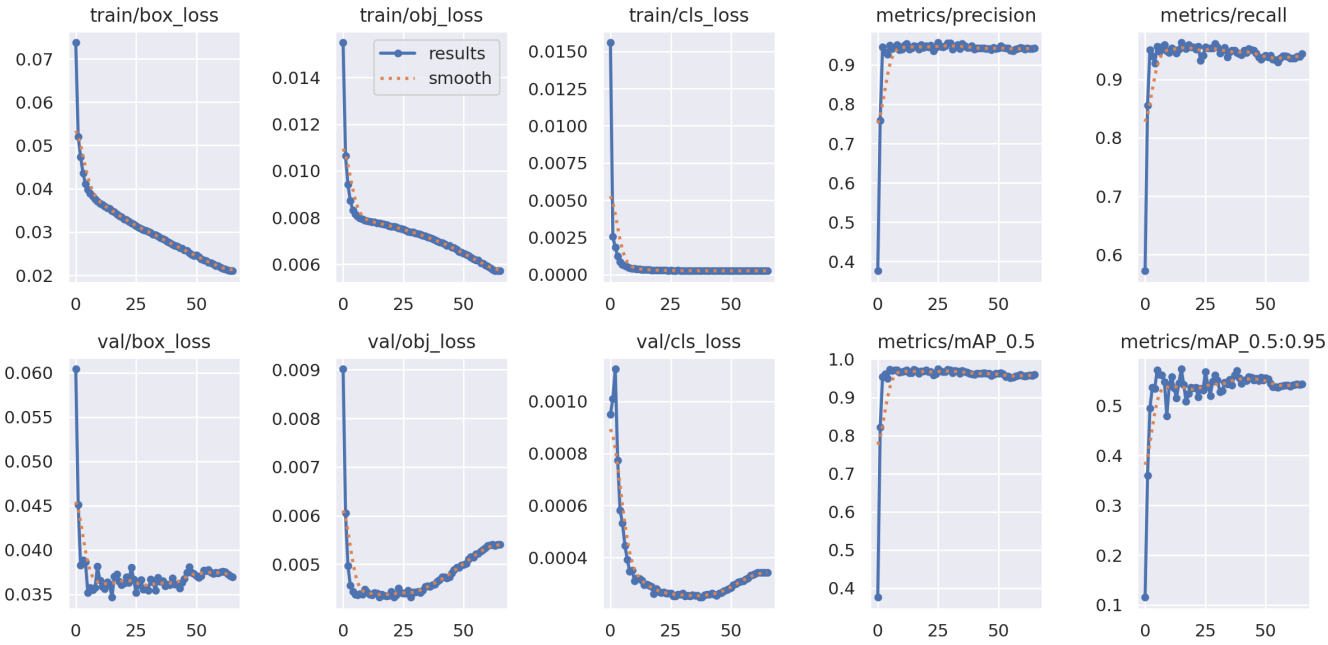

Figure 30: Loss and metrics plots for the HiresCAM model trained with heatmaps per class

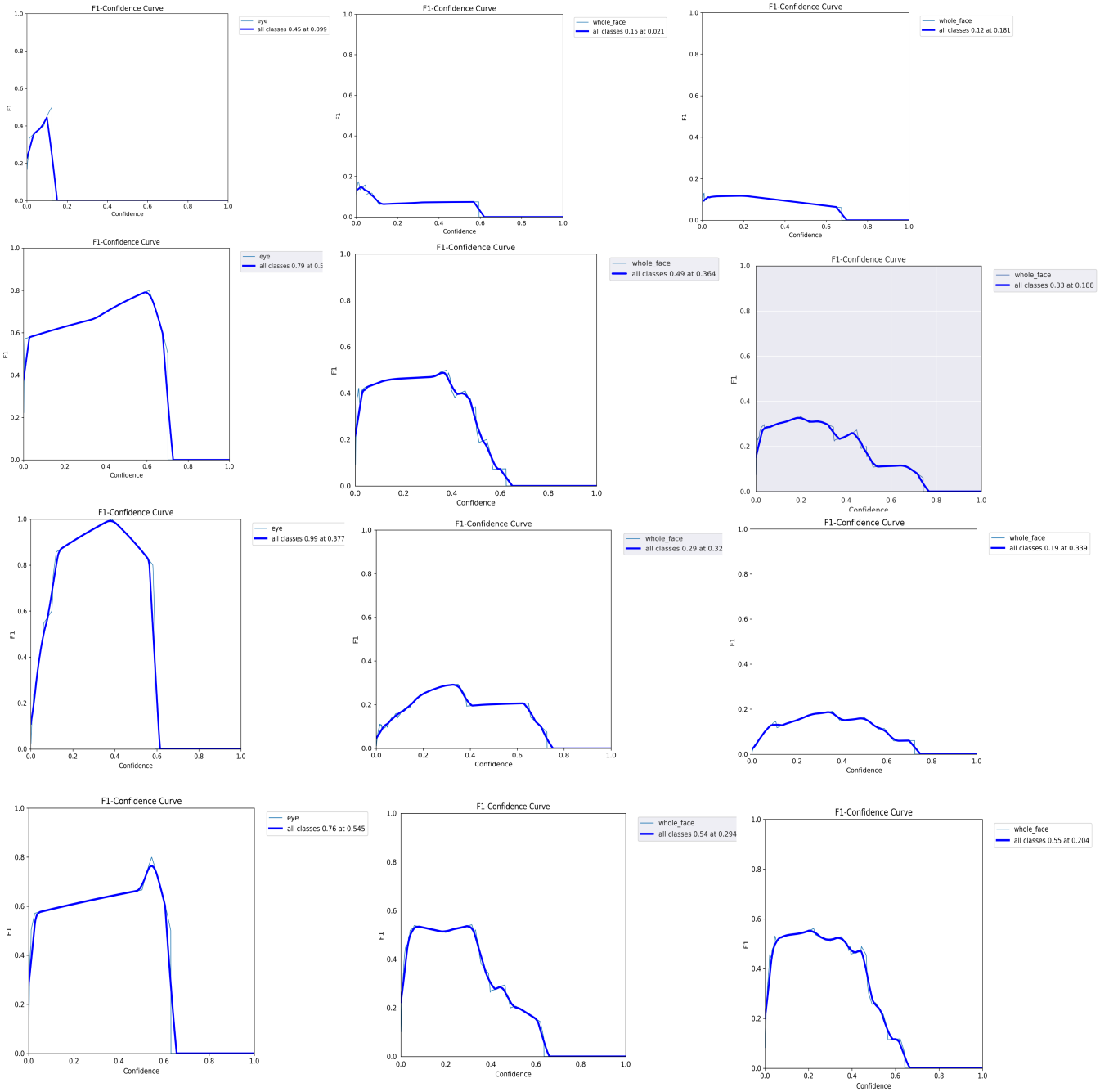

Figure 31: F1-confidence curves for Easy, Medium and Hard occlusion sets of SLAPI. From top to bottom we have Grad-CAM++(avg), HiresCAM(avg), GradCAM++ (10class), HiresCAM(10class)