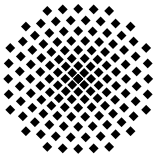


University of Stuttgart
Institute for Signal Processing and System Theory
Professor Dr.-Ing. B. Yang

D1448

AI Assisted Semi-Automatic Labeling of Eye Contact Data Using Active Learning Approach

Author: Srijay Kolvekar

Date of work begin: 15/05/2022
Date of submission: 15/11/2022

Supervisor: Dr. Mihai Bace, Mr. Martin Schwartz

Keywords: Gaze Estimation, Eye Contact Detection, Head Pose Estimation, Face Detection

Abstract

One of the important non-verbal cues to understanding human interactions and intentions is through analyzing facial features. Eye contact is one of the most important feature of facial attributes that signify important non-verbal traits of human communication. With the advent of deep learning and computer vision, many approaches have been developed over the past few years to extract and learn non-verbal cues through gaze estimation, and eye contact detection. While most methods for gaze estimation and head pose detection have improved over the years, the problem persists in detecting eye contact or user attention on smaller objects and with long-range gaze estimation as evident in mobile usage scenarios. In addition to aforementioned shortcomings, a substantial amount of annotated data is required for successful and generalized supervised learning, which is a costly and time-consuming operation. In this work, we propose a novel method of deep learning-based semi-automatic labeling of eye contact data using an active learning approach to label unseen eye contact data. To achieve robust and accurate performance, we employ a contrastive learning framework to improve feature space embedding by further pushing the similar and dissimilar eye contact distribution clusters. In addition to contrastive learning, we use uncertainty estimation to monitor the misclassification and investigate the uncertainty in the feature space embedding of eye contact distribution. Uncertainty information about classes is further used in sampling the data in the active learning approach to make robust labeling decisions.

Contents

1. Introduction

In this chapter, we discuss the motivation for undertaking this task to improve the eye contact detection task. Furthermore, we also discuss the contributions we made in our work to achieve the objectives.

Eye contact is one of the most important non-verbal cues that describe the users intention and interaction in social and human-machine scenarios. As a result, eye contact detection has become a crucial tool for comprehending interaction between human and machine interfaces. This information can be pivotal in designing new applications for attentive user interfaces and improving the interaction between human and machine engagement. Eye contact detection is the task of determining whether a person is looking at a target object or not. However, the eye contact detection algorithm is not a trivial task, as there are a few challenges involved in successful eye contact detection as described in section 1.1. Furthermore, in the mobile use case scenario, due to various factors like the size of the target object, which is a mobile phone, different illumination and facial features affect the overall performance of the detector. Moreover, the target object determines the line between eye contact and non-eye contact in a given visual plane. In the case of mobile usage, the size of each device has a different boundary. For learning-based eye contact detection, the algorithm must first determine the target object's size and placement in relation to the camera. It also needs specialized training data to train the eye contact detector specifically for the target. Training a generic eye contact detector—one that is effective even for very small target sizes and locations close to the camera is a challenging task to solve.

This literature of this work is majorly organised into seven parts. In the first section, we described the introduction to the eye contact detection and further discuss the motivation to undertake the task. We also describe the objective and contributions to this work. In the next chapter, we describe in detail the theory of our work, which cover the description of all fundamental modules used in our work to accomplish our objective. The third section of this work is dedicated to explain our novel framework and all the approaches we developed in our thesis which includes the application and investigation of contrastive learning framework and multitask task framework on the gaze estimation task. In the next section, we go though the results obtained from our different approaches and experiments. And in the last two chapter, we finally discuss the outcome of our work and also provide conclusion with future scope of improvements.

TO BE WRITTEN

1.1. Motivation

With multiple devices at our disposal and frequent notifications, studying visual attention has become a pivotal area of research in the human-computer interaction (HCI) domain. A

crucial characteristic for understanding non-verbal cues is by investigating facial features during human-machine interaction. Facial attribute information can be extracted from the gaze of a person, head pose, facial landmarks, or eye contact data, to name a few. These facial characteristics yield [1] important findings in determining the user's attention, which can be used in the fields of psychology study, web search optimization [2], and in general HCI use cases. Furthermore, due to the influence of neurological activity on visual information processing, eye tracking and eye contact detection have become two of the most useful methods for studying neuroscience. Moreover, estimated eye movements and eye gaze patterns can aid attentional investigations into applications like object-search processes, neurological functions during non-cognitive visceral decision making, and medical diagnoses such as post-concussive syndrome, autism, and schizophrenia [3].

Hence, eye contact detection a very important area of research to explore. Since the advent of deep learning and computer vision, various methods have been described to solve the problem of eye contact detection. Although the methods for eye contact detection achieve good performance, they have a few serious limitations which are considered with stationary settings of mobile use case. Moreover, very few methods are employed on mobile device scenarios, which, as described, involve users in highly dynamic environments like trains, homes, or offices. In addition to environmental conditions, there is variance in the placement of the mobile device with respect to the face, which is non-trivial for gaze estimation and eye contact detection task and is challenging problem to solve. Despite the state-of-the-art performance of appearance-based deep learning gaze estimation models and eye contact detection, due to the high variability in the appearance of human facial features and different use case environments in mobile usage setup, it becomes difficult to generalize the gaze estimation models. Moreover, the performance of gaze estimation is not sufficient in detecting small eye contact as is the case in hand-held devices.

In addition to the above-mentioned drawbacks, successful eye contact detection requires huge amounts of annotated data, which poses serious limitations. Moreover, precise and speedy annotation in the machine learning area is a critical and cumbersome process. Annotating/labeling objects with crucial information with the goal of knowledge transfer has become an ever-growing requirement in the supervised/semi-supervised domain.

Hence, in this work, we investigate methods like contrastive learning, active learning, and uncertainty estimation to improve the task of gaze estimation and eye contact detection. In addition, we are working on developing a labeling framework to label the eye contact data and running inference on the new dataset.

1.2. Objectives

The key objective of this work is to develop a robust eye contact detection algorithm provided it works in extreme cases of mobile usage environments with better generalization for all the domain shifts in the data. We investigate gaze estimation algorithms in depth to make the eye contact detection task more robust.In order to have a robust performance on gaze predictions in adverse conditions, we investigate contrastive learning approaches, both supervised and self-supervised. We also aim to explore the multitask task approach in the case of contrastive learning and with multiple tasks for gaze and head pose predictions. In our work, we pro-

pose a novel semi-supervised labeling technique and a PyQt5-based GUI that facilitates the process of labeling of gaze data and eye contact data, thus mitigating the aforementioned drawbacks. We also intend to use an active learning-based approach based on uncertainty estimation to improve the overall performance of the eye contact detection task.

1.3. Contributions

The main contributions to this work are focused on three areas. First, we investigate the performance of the Feature Pyramid Network (FPN) and Resnet on the gaze estimation task. We compare and quantify the FPN and Resnet architectures with and without pre-trained weights, as well as their effects on gaze estimation task performance. We then explore in depth the gaze estimation task and described three approaches and investigate the possibility and outcomes on using contrastive learning framework on both self-supervised and supervised setting. To train the gaze estimation task on supervised contrastive learning, we use the binning method to convert continuous variables to discrete class labels. As a third approach, we investigate the impact of using a multitask model with and without contrastive settings on the gaze estimation task. The novel aspect of our work is that we investigated the contrastive learning framework on a regression task and evaluated the performance.

1.4. Related Work

1.4.1. Eye Contact Detection

To solve the problem of eye contact detection, many approaches have been demonstrated that use an eye tracking device to track the position of eye movements. In the method described in [4], a viewpointer is used with a wearable IR sensor to detect the user's attention towards the object. However, this requires a dedicated hardware setup, which is expensive and requires a high level of calibration. With increasing research in the deep learning and computer vision domains over the past few years, a lot of emphasis has been shed on developing strong eye contact detection algorithms using an appearance-based gaze estimation method by regressing gaze direction.

In this work[5] GazeLocking, a supervised classification method is used to detect eye contact with an appearance-based gaze estimation approach using a convolution neural network (CNN). This work demonstrated the method of using second-person perspective using head-mounted cameras for eye contact detection using supervised learning. The method described in this work [6] demonstrated gaze estimation method using an unsupervised learning approach with estimated gaze direction clustering for eye contact detection. In the work [7], facial landmarks, along with head pose information, are taken into account for appearance-based gaze estimation on CNN. A clustering-based approach is used to infer gaze location wherein the data point closest to the cluster center is labeled with positive and negative eye contact labels. Furthermore, Support Vector Machine (SVM) is used to train a classifier based on the labels and image features to classify eye detection.

Our work is the extension of [7], in which we investigate the contrastive learning framework and multitask learning in conjunction with active learning to improve the robustness of eye contact detection in mobile use case scenarios.

1.4.2. Gaze Estimation

Gaze estimation is a method of estimating the direction in which a human subject is looking while understanding the underlying human attention. In recent years, with the traction in deep learning and computer vision, many CNN based supervised [8],[**?**] and unsupervised methods [9],[10] have been proposed that excel at high and low feature extraction while also learning non-linear mapping of appearance to gaze. Over the past few decades, many gaze estimation approaches have been proposed. In general, there are three methods of gaze estimation, namely the 3D eye model recovery-based method; the 2D eye feature regression-based method, and the appearance-based method [11]. Methods based on 3D eye model recovery produce a geometric 3D eye model to determine gaze directions. Due to the diversity in appearance of human eyes, the 3D eye model is highly person-specific and cannot be used as a generic approach. As a consequence, personalized calibration is generally required to recover person-specific data like iris radius and kappa angle. Device specifications for 2D eye feature regression-based methods are typically the same as for 3D eye model recovery-based approaches. To regress the point of gaze (PoG), the approaches utilize the observed geometrical features such as pupil center and glint. Geometric calibration is not required when converting gaze directions to PoG. In contrast to the two methods mentioned above, appearance-based approaches don't require any special equipment or calibration setup; instead, they leverage cameras to capture color images of human eye appearance and regress gaze from them. This is possible by simple intuition based on eye appearance, where a change in iris location leads to a change in gaze direction. The early methods of non-intrusive appearance-based gaze estimation were proposed over a decade ago, which used simple neural networks for gaze estimation using eye region pixel extraction [12]. In this work [13], local linear interpolation regression method was used to regress the gaze. These methods described are limited to subject-specific and constrained use cases.

With increasing research in the deep learning and computer vision domains, many state-of-the-art methods based on (CNN) have been proposed which are capable of extracting low-level and high-level features from the full face or eye patch image to regress 2D or 3D gaze angles. Due to robust feature extraction methods, the deep learning-based methods outperform most conventional appearance-based estimation methods, which use classical appearance-based methods. Furthermore, CNN-based methods have been shown to be resistant to changes in visual appearance and lighting conditions, as well as capable of dealing with person-independent use case scenarios. One of the first methods using LeNet, a deep learning based architecture for gaze estimation, was described in the work [8], which used eye patch images to regress 2D gaze angles (yaw and pitch). A large-scale dataset was used to train a generic model. Successive to the work mentioned, many methods have been described with further improvements in the methodologies with robust performance. In [14], the eye image patch for gaze estimation with an hourglass model to estimate pictorial gaze segmentation maps of the eye ball and iris. In [15], a person-specific gaze estimation method is described, which uses an encoder and decoder setup to disentangle the data for robust rotation-aware latent representation with a meta-learning [16] approach for further calibra-

tion. In [17]. a multitask model is employed to learn the feature embeddings of the left eye, right eye and head pose from independent VGG-16 models for gaze estimation tasks. MANet [18],2 stage network, which uses original and augmented images in two stages, later self-supervised loss is used to minimize similarity loss. L2CSNet [19] uses classification and regression losses. The classification loss is calculated on binned values and regression on fine grained value with individual losses for pitch and yaw. The backpropagation is carried out for each individual loss.

However, many challenges hinder the performance of classical appearance-based gaze estimation, such as head pose and subject heterogeneity, especially in the mobile usage environment. These elements pose a serious issue, hence appearance-based gaze estimation is unable to tackle extreme cases.

1.4.3. Head Pose Estimation

Head pose estimation plays a pivotal role in various human-machine interface applications in determining the behavior of human. In head pose estimation given a facial image of a human, the algorithm predicts the viewing direction or 3D head pose. The outcome of head pose estimation provides yaw, pitch, and roll in 3D space. In the context of eye contact detection, a robust algorithm for head pose estimation becomes an important prepossessing step to normalize the face image to normalized space with fixed camera parameters as described in the section **??**.

The head pose estimation task can be majorly divided into two types, namely facial landmark-based approaches and landmark-free approaches. With the landmark-free approach, the head pose is computed without using additional facial landmarks. In QuatNet [20] the model adapts the cross-entropy paradigm but divides classification and regression onto distinct network branches. One branch is used to categorize Euler angles, and the other is used to regress the posture in quaternion form. Similar to this, HPE [21] treats classification and regression independently before averaging the results to create a pose regression subtask. Similarly, in HopeNet [22], the target angles were binned with specific binning windows. A multi-loss approach was used with binned pose along with cross-entropy loss and Euler pose angle with mean squared error loss.

Landmark-based approaches primarily take into account the facial landmarks, then establish correspondence between these landmarks and a 3D head model to recover the 3D head posture. A more widely used method to predict the head pose is through solving the perspective-n-point (PnP) [23] pose computation. In PnP, the projection error is minimized while computing object pose given a set of object points, corresponding image projection, and intrinsic and extrinsic camera parameters. To compute the pose using the PnP method, we need 2D coordinate points of the face image, which can be retrieved from face landmarks, and the 3D location of points from a generic 3D model, which corresponds to an arbitrary reference frame. In addition to 2D coordinates from image and 3D coordinate from 3D model, we also need intrinsic camera parameters as described in section 5.3.1 FILL, which is assumed to be calibrated with known focal length, optical center and radial distortion parameters. However, the intrinsic parameters for different hardware setups cannot be generalized, hence approximated parameters are chosen. Furthermore, all the parameters such as rotational, translation, and intrinsic parameters as described above are used to convert from 3D points

in world coordinates to 3D points in image coordinates and, subsequently, to 2D points in image coordinates.

In the work described in [24], an encoder-decoder CNN with residual and skip connection like Unet [25] is used to predict the head pose estimation along with facial landmarks while describing the important correspondence between both the tasks. Although this method can produce quite accurate results, it is largely dependent on the proper landmark positions being predicted. As a result, an inaccurate head pose estimate can be hampered by poor landmark localization brought on by occlusion and excessive rotation. In the work [7], [26], a PnP method is used to compute head pose, and Levenberg-Marquardt optimization is used to further minimize re-projection error.

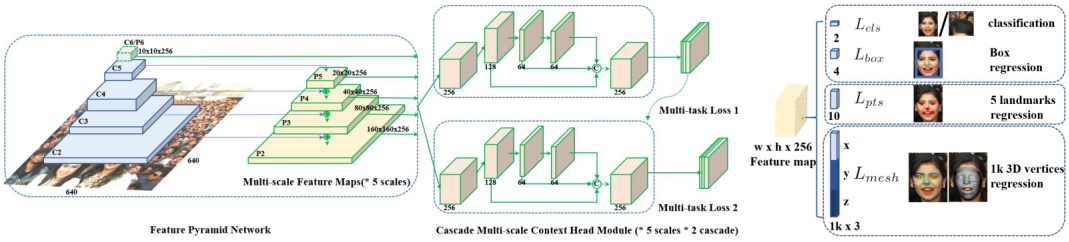
For our work, we used landmarks based PnP method to compute the head pose of the image for further normalization procedure as described in details in **??**.

1.4.4. Face Detection

For successful gaze detection task and subsequently for eye contact detection task, robust face detection is become a pivotal preprocessing step of the eye contact detection pipeline. Images captured from the front-facing camera in mobile use case scenarios in various environmental conditions typically have a large variation in head posture and may only show a portion of the face [27].

In recent years, face detection algorithms have evolved significantly, with a focus on constantly improving performance and efficiency. The human face contains a complex structure with huge variability in features such as: skin texture, color, facial hair, non-verbal cues, and pose, to name a few. All the above mentioned features pose a difficulty in face detection, making the task non-trivial. Face detection algorithms take input data in image or video format. The face in the image is regraded as the foreground or area of interest, and the non-face as the background, making a clear distinction. Facial attributes are extracted using a feature extractor to then locate and detect the face. The face detection algorithms can be classified into single-stage and two-stage detectors. Much work recently has been focusing on single-stage detectors due to their lower computational complexity and lower inference time. In [28], the authors described a single-stage face detector with a headless network and scale-invariant architecture working on multiple scales at a single forward pass without a pyramid network. [29] described a method to propose a scale-equitable framework to handle faces of different sizes while improving the anchor scales with an effective receptive field. In [30], a context anchor was used to monitor the high-level facial attributes with a semi-supervised method known as the pyramid anchor. In addition, a second feature pyramid was used to learn low-level facial features with high semantic information.

In this work, we use a state-of-the-art single-stage face detection algorithm called Retina-Face face detector [31]. Retina-Face is based on the Feature Pyramid Network (FPN) architecture with multi-task loss, which carries out the operations of classification, bounding box regression for face localization, landmarks regression, and 3D vertices regression as described in the loss function 1.1. The Retina-Face contains Multi-level Face localization which employs FPN for multiple level feature maps. In the context of multilevel face localization, a multitask task loss is minimized for given training anchors from multiple scales of the input image.



(a) Network Structure (b) Multi-task Loss

Figure 1.1.: An overview of FPN based single shot retina face network structure

$$\mathcal{L} = \mathcal{L}_{cls}\left(p_i, p_i^*\right) + \lambda_1 p_i^* \mathcal{L}_{box}\left(t_i, t_i^*\right) + \lambda_2 p_i^* \mathcal{L}_{pts}\left(l_i, l_i^*\right) + \lambda_3 p_i^* \mathcal{L}_{mesh}\left(v_i, v_i^*\right) \tag{1.1}$$

where t_i, l_i, v_i are box, five landmarks and 1k vertices predictions, t_i^*, l_i^*, v_i^* are corresponding ground-truth. p_i is objectness score of anchor i which signifies positive anchor and negative anchor. L_{cls} is softmax loss for binary classes signifying face and not face. For computing the bounding box, a positive anchor with center coordinates x^a_{center}, y^a_{center} and scale s^a.

The framework of the proposed single-shot, multi-level face localization strategy is shown in Figure 1.1. The network consists of three major parts: the context head module, the cascade multi-task loss, and the feature pyramid network. The feature pyramid network first receives the input face images before producing five feature maps of various scales. The context head module then obtains a scale-specific feature map and computes the multi-task loss. The first context head module generates the regressed anchor, which is used by the second context head module to forecast a bounding box that is more precise than the one produced by the first context head module. Retina Face uses FPN from level P2 to P6, with P2 to P5 computed from the output of the Resnet residual stage through C2 to C5. P6 is calculated using a 3×3 convolution on C5 with a stride of 2. Furthermore, the context module is for all feature pyramid levels to improve the receptive field and increase context modelling power. The anchors for the face localization are generated from FPN from layer P2 to P6, where P2 is a high resolution feature map for capturing small objects and P6 is to capture larger features from feature maps. The anchors can cover scales from 16×16 to 406×406 across the feature pyramid levels with an input image size of 640×640. P2 accounts for 75% percent of the total of 102,300 anchors. When the intersection over union (IoU) for the first head module is greater than 0.7 and less than 0.3, the background is used to match the anchors. When IoU is greater than 0.5 for the second head module, anchors are matched to a ground-truth box; otherwise, when IoU is less than 0.4, anchors are matched to the background. During training, unmatched anchors are disregarded.

2. Theory

2.1. Architecture for Gaze Estimation

In this work, we investigate the Feature Pyramid Network (FPN) architecture for gaze estimation tasks, thus comparing the performance of the generic resnet architecture with that of the resnet-based FPN architecture. In most of the gaze estimation tasks [32], [33], [9], as described in these works, a generic resnet is used as a feature extractor. Although the generic resnet feature extractor performs well, there hasn't been much research carried out in exploring the FPN-based architecture for gaze estimation tasks.

2.1.1. ResNet

Feature extraction is crucial in the deep learning domain to fully comprehend the subtle features of the data and use them for the subsequent tasks of classification, regression, or generation. Many deep convolutional neural network algorithms are reported with higher performance in light of current advances in image classification. Furthermore, the depth of the neural network, aids in learning low-level and high-level aspects of the data.

Deep residual learning, often known as Resnet [34], is one of the most effective feature extraction techniques due to its successful applications around various domains. Resnet makes use of the ability of skip connections to create deeper networks without running into the issue of vanishing/exploding gradient. When layers are stacked on top of one another, the issue of vanishing or extending gradients develops, leading to an ill-conditioned convergence problem. The Resnet employs a skip connection or shortcut connection to add mapping for a sequence of stacked non-linear layers, thereby adding extra information from the prior levels to avoid the issue of vanishing gradient as described in figure 2.6. There is no additional computational cost or parameter increase due to the identity mapping from the preceding layer. The skip connection additionally aids in resolving the vanishing/exploding gradient issue in deeper networks, which also solves the problem of convergence.

In the formulation described in [34], $H(x)$ is considered as underlying mapping from few non-linear stacked layers, with x as an input to these layers. Assuming input and output as same dimensions, we can approximate the residual function as $H(x)x$. Therefore, we explicitly allow these layers to approach a residual function $F(x) := H(x) + x$ rather than assuming that they will approximate $H(x)$ thus, the initial function becomes 2.1.

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{2.1}$$

Two approaches—bottleneck and non-bottleneck residual function—are outlined in Resnet for the skip architecture of skip connection. A network that is relatively shallower is employed with the non-bottleneck residual function. As seen in figure 5.3.1 left, this module

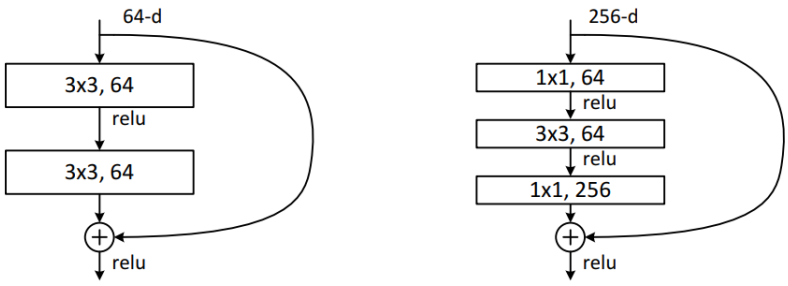


Figure 2.1.: Left: Non Bottleneck residual block. Right: Bottleneck residual block with 1×1 convolutions

has two $3x3$ convolutional networks with a skip connection. A bottleneck residual module is utilized to deal with deeper networks. Unlike the non-bottleneck module, this module has a stack of 3 convolutional layers rather than 2. The kernel sizes of the three layers are 1×1, 3×3, and 1×1, respectively. The 1×1 layer is used as a bottleneck, which is predominantly used to reduce the computational complexity of the network by first reducing the feature maps in the first layer and then increasing the feature maps in the third layer. The middle layer's input and output dimensions are reduced to 3×3 via this layer map procedure. It also includes a skip link from the preceding tier. For problems involving gaze estimation, we have used Resnet as our basis feature extractor in our work.

2.1.2. Feature Pyramid Network (FPN)

Both low-level and high-level features from the complete face or eye patch image are used to regress 2D or 3D gaze angles in the context of an appearance-based gaze estimation task. Understanding both high-resolution semantically weak features and low-resolution semantically strong features is necessary for the fine-grained regression task of gaze estimation. The Resnet backbone-based FPN [35] architecture with a bottom-up pathway and a top-down pathway with lateral connection is used to carry out the operation as previously defined.

In this subsection, we describe the general FPN architecture. The FPN contains CNN backbone with feedforward operation, which computes various feature hierarchy levels with correspondingly varying scaling steps starting at a step of two, is described by the bottom-up pathway. The feature pyramid has one level defined for each stage. To obtain the reference feature maps, resnet-based CNN's final residual block output is employed. For conv2, conv3, conv4, and conv5 outputs, the output from the residual block is defined as C2, C3, C4, C5, and their strides from the input image are 4, 8, 16, 32 pixels. Due to its significant memory footprint, the initial conv layer is not taken into consideration.

The top-down pathway uses feature maps from higher pyramid levels that have stronger semantic associations to illustrate higher resolution features with spatially coarser features. These features are then enhanced with features from the bottom-up pathway through lateral connections. Each lateral link combines the same-sized feature maps from the top-down and bottom-up pathways. Although the bottom-up feature map was subsampled less frequently, its activations are more precisely localized despite having lower-level semantics. To decrease

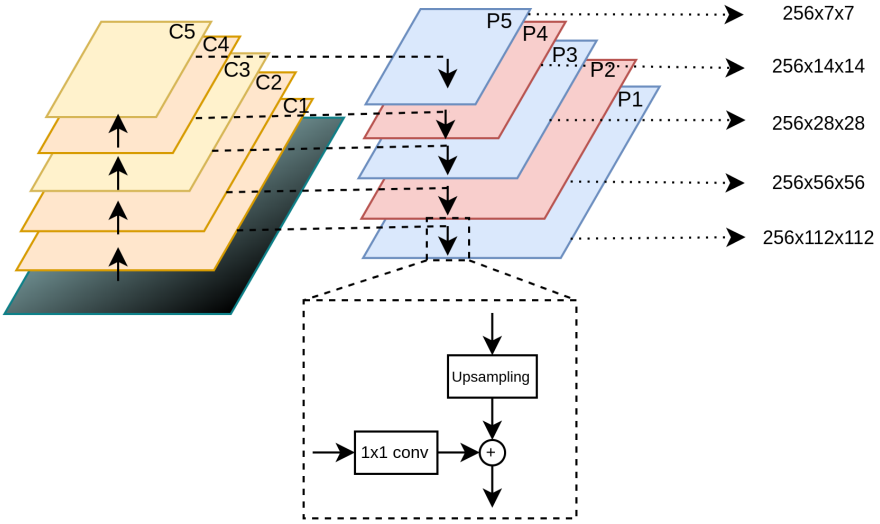


Figure 2.2.: FPN architecture with multiple levels with top-down and bottom-up pathway for feature extraction

channel dimensions, top-down feature maps and their matching bottom-up feature maps are combined with a 1×1 convolutional layer. Up till the highest resolution map is obtained, this process is iterated. In addition, C5 is given a 1×1 convolutional layer to start the iteration and produce the lowest resolution map. P2, P3, P4, and P5 are the final feature maps from FPN, and they correspond to C2, C3, C4, and C5. Downsampled feature maps can be found in P6. According to the figure 2.2 the C1 and C2 levels each have 64 feature maps, while the C3, C4, and C5 levels have 128, 256, and 512 feature maps, respectively.

2.2. Multitask Learning

In the multi-task learning approach, machine-learning algorithms are trained utilizing data from several tasks at once, employing shared representations to discover the commonalities among a set of related tasks [36]. These common representations enhance data effectiveness and could hasten learning. Multi-task learning more closely resembles how humans learn than particular task learning because collaborative learning across domains is an essential part of human cognition. A cyclist needs develop broad motor skills that rely on intuitive physics and abstract concepts of balance in order to learn to ride a bike or control a bicycle with its legs and hands. These motor skills and cognitive concepts can be learned, used, and eventually developed for more challenging tasks. Every time a human makes an effort to learn anything new, they carry a tremendous quantity of prior knowledge from a variety of experiences. It is obvious why neural networks need so many training samples and so much computing time given that each task must be thoroughly learned from scratch.

Humans are able to learn new things rapidly with a few instances because of the process of acquiring generalizable concepts that can be applied in a variety of circumstances and applying these concepts for speedy learning. Multitask learning is crucial for grasping a task by achieving many goals. Utilizing generalized knowledge, multitask learning distributes

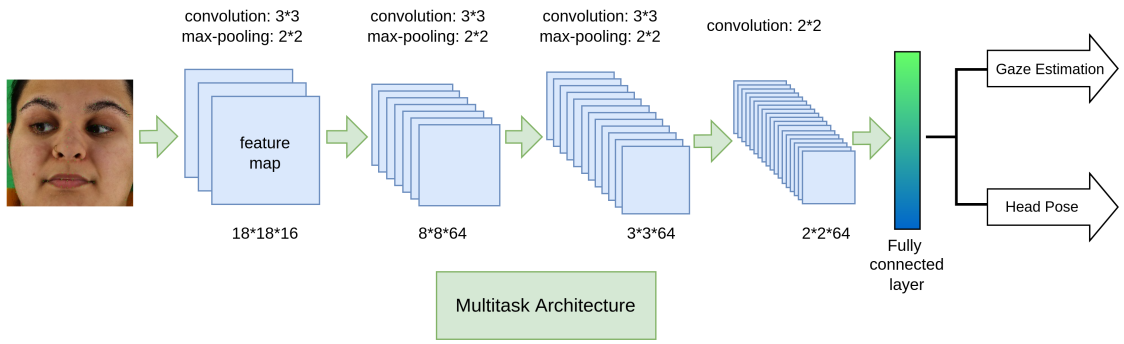


Figure 2.3.: An example of multitask architecture with visualization of intermediate feature extraction CNN layer and respective heads for gaze and head pose estimation.

feature representations or embeddings among related tasks. The context of object location and detection in an object detection task is a good illustration of multitask learning. In this use instance, the model anticipates that the learned representation will include data on both the type of object location relative to the image coordinate and the object to which it belongs.

However, different task learning does create challenges that are absent from single task learning [36]. It is especially conceivable for different tasks to have requirements that conflict with one another. When a model performs better on one task but worse on another with different criteria, this is referred to as adverse transfer or destructive interference. Minimizing negative propagation is one of the main goals of multitask learning strategies. Hard parameter sharing and soft parameter sharing are the two main categories of frameworks used in multitask learning. In a method known as "hard parameter sharing," model weights are divided across numerous tasks and each weight is trained to jointly minimize a number of loss functions. Because the model learns numerous task-specific feature representations, overfitting is less likely in hard parameter sharing. Under soft parameter sharing, each task has its own task-specific models with distinct weights or parameters, but the distance between each model parameter is regularized to the combined objective function for similarity. Even when there isn't a clear parameter exchange, the task-specific models are nevertheless driven to have comparable parameters.

With the growing popularity of multitask learning among deep learning researchers, many approaches have been described that leverage the potential of multitask learning in solving complex problems. In the work [37], multilinear relationship network is described which discovers the task relationship with the normal prior task-specific layer in deep convolutional networks. It exploits transferable features and multilinear relationships between tasks and features to overcome the problem of negative transfer. In [38], a bottom-up technique is proposed, starting with a narrow network and gradually widening it greedy throughout training using a criterion that encourages grouping of related jobs. While assigning each branch to only one job prevents the model from learning more complicated relationships between tasks, the greedy approach might not be able to find a model that is globally optimal.

In our work, we used a multitask learning approach to improve the feature embeddings of the gaze estimation task. We explored the possibility of using multitask learning in a contrastive learning setup. We also used the method in training the gaze estimation model by training the model for both the gaze estimation task and the head pose estimation task. These different setups will be explained in detail in the approach section 3.4.

2.3. Uncertainty Estimation

In decision making, strong reasoning becomes necessary for unbiased and rational decisions. With advancements in deep learning, there is wide usage of DL models in safety-critical applications like autonomous driving and the medical domain. Despite the remarkable performance, there still exists the notion of black box paradox when understanding DL models. For example, in automatic cancer detection using CT scans, it becomes of utmost priority and essential for models to be certain while not infusing any bias in decision making by experts. Furthermore, an out of distribution image can result in an uncertain model prediction. Moreover, the models only yield point estimates of parameters and predictions, thus creating doubts about the credibility of the predictions. Bayesian models provide a probabilistic framework, which makes them more dependable and trustworthy than DL models. We do uncertainty estimation on DL models to overcome the uncertainty issue. This provides specific insights into the operation of the models as well as the degree of confidence and uncertainty in the forecasts.

Considered broadly, altetoric and epistemic uncertainty are the two categories of uncertainty in DL models. Aleatoric uncertainty, also known as dice player's uncertainty, is caused by noise during the data measurement process. This might also be caused by data uncertainty. For instance, even a human annotator may find an image to be blurry or hazy when classifying it, which could increase uncertainty. Due to the stochastic nature of the data, alteoric uncertainty is often referred to as irreducible uncertainty. Epistemic uncertainty, also known as knowledge uncertainty, is specifically brought on by ignorance of or uncertainty over model parameters. As the name implies, this can be diminished by training the model with more relevant and diverse data while reducing the domain shift.

In our work, we compute epistemic uncertainty or model uncertainty of gaze estimation task to find the amount of uncertainty in the gaze predictions. In this method [39], we apply Monte Carlo dropout to the pre-trained gaze estimation model. A statistical technique called Monte-Carlo uses a random sample of variables to determine the predicted distribution. This technique simulates many models by applying dropout at the network's last layer. Multiple predictions are made for the same input sample during each forward pass as a result of some nodes at the last layer being eliminated due to the dropout layer having a specific dropout probability. The prediction uncertainty of a model defining the mean and the related uncertain distribution is successfully captured by this method.

2.4. Support Vector Machine

SVM is a supervised machine learning algorithm is used in tasks such as classification, regression, and function approximation.SVM has gained huge success due to its applicability in a wide range of applications to solve linear and non-linear practical problems.

Considering a binary classification problem with labels, SVM creates a decision boundary or hyperplane between two classes given the input data and its ground truth labels. There are numerous hyperplanes that could be used to separate the two classes of data points. The goal of SVM is to find a plane with the greatest margin, or the greatest distance between data points from both classes. Maximizing the margin distance provides some reinforcement,

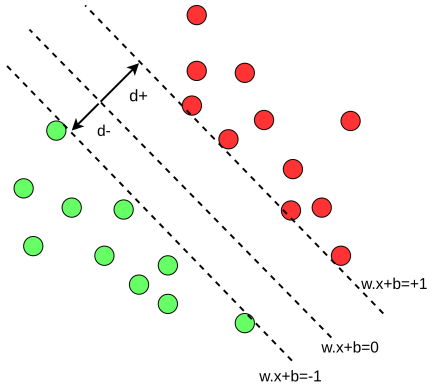


Figure 2.4.: An example with support vectors and hyperplane decision boundary separating two classes

allowing future data points to be classified with greater certainty. Hyperplanes are decision boundaries that aid in the classification of data points. Data points on either side of the hyperplane can belong to different classes as described in the figure 2.4. Furthermore, the size of the hyperplane is determined by the number of features. Support vectors are training data samples that run along hyperplanes near the class boundary, and the margin is the distance between the support vectors and the class boundary hyperplanes. d+ and d- describes the shortest distance of the closest positive and negative point respectively. $w.x + b = +1$ is the positive hyperplane, $w.x + b = -1$ is the negative hyperplane and $w.x + b = 0$ is the median hyperplane.

$$\min_{w} \lambda \|w\|^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+ \tag{2.2}$$

Where, λ is weight regularization term, x_i is input and y_i is corresponding label to that input. w is set of weights. The goal is to maximize the margin between the data points and the hyperplane in the SVM algorithm. A hinge loss is a loss function that aids in margin maximization. If the predicted and actual values have the same sign, the cost is zero. In addition, a regularization parameter is added to the cost function to balance margin maximization and loss.

However, when the classes are not linearly separable, the SVM performs a kernel trick. In a kernel trick, the low-dimensional input is converted into high-dimensional space while converting non-linearly separable problems into linearly separable problems in high dimensional space using kernels like polynomial or radial basis functions. SVM is majorly divided into two types, namely soft margin and hard margin, based on how the outliers are handled. To maximize the margin, SVM tolerates a few outliers or misclassified data samples. The default behaviour of SVM is hard margin, which does not tolerate any misclassification to happen and works well when the data is linearly separable.

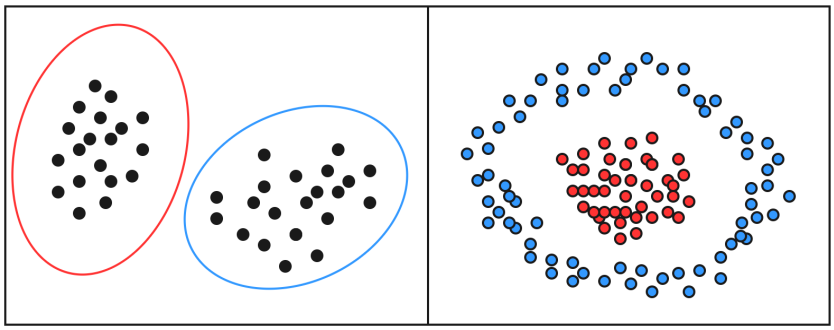


Figure 2.5.: An example of type of clusters

2.5. Clustering

A clustering algorithm is an unsupervised learning method with the objective of organizing database objects into a set of useful categories. Clustering is used to gain insight into the distribution of a dataset and to focus subsequent data mining and processing. A clustering algorithm can be used on its own or as a preprocessing step for other algorithms that operate on the identified clusters. The goal of cluster analysis is to help the user understand the underlying grouping or pattern in a data collection. Clustering algorithms are classified into three types: hierarchical, partitioned, and density-based. The hierarchical clustering algorithm divides a large pool of data into smaller tree-based clusters with several levels of nested clustering as described in figure 2.5. Furthermore, every node in this tree represents a cluster. In partitioned, the data is divided into cluster sets, with objects in the same cluster belonging to the same family, as opposed to samples with different clusters. When the cluster is uncluttered and divided, both hierarchical and partitioned systems work best. When the clusters are cluttered and arbitrary in shape the density-based clustering techniques are more appropriate and efficient.

In this work, we use a density-based clustering algorithm called Ordering Points to identify the clustering structure(OPTICS) [40] which is based on density-based spatial clustering of applications with noise (DBSCAN) [41]. The samples in OPTICS are clustered based on a few important parameters such as eps, minPts, core distance, and reachability distance. The distance between neighborhoods is defined by eps in this case. The data points are said to be neighbors if the distance between them is less than or equal to eps. The minPts value represents the smallest number of samples that make up a cluster. The data samples are classified as core point or border point based on the eps and minPts. A sample is considered a core point if there are minPts in its vicinity with a radius of eps, and a border point if it is near a core point and there are lower minPts nearby.The core distance is the minimum radius to categorize the sample as a core point. Lastly, the reachability distance represents the distance between two points and allows for different densities of extraction.

In the eye contact detection pipeline, we obtain the 2D gaze points from the gaze estimation algorithm. The obtained 2D points are then used as sudo-labels for the final task of detecting eye contact. The sudo-labels, on the other hand, are simply gaze points projected on a 2D plane. We use the OPTICS clustering algorithm to convert the gaze points into actual sudo-labels.

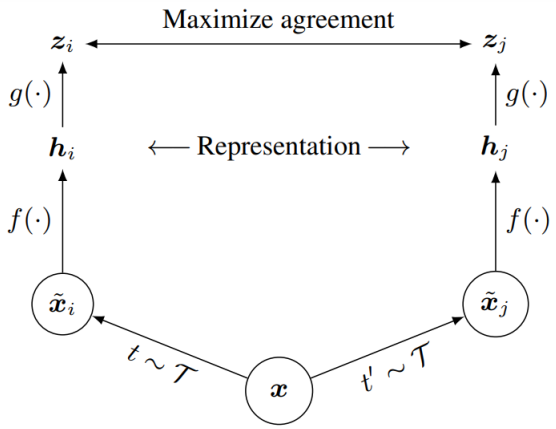


Figure 2.6.: [42], A representation of contrastive learning framework to maximize agreement between similar images

2.6. Contrastive Learning

In this section, we investigate the constastive learning framework on gaze estimation regression tasks, both supervised and semi-supervised, and investigate the change in model performance. Contrastive learning is a learning paradigm that learns rich feature representations of raw data in order to bring similar features closer together and push dissimilar features away while maximizing mutual information and minimizing conditional cross entropy.

The quality of the data plays an important role in determining the performance and learning capability of a machine learning algorithm. In general, data has a set of important and useful features that, in a nutshell, represent the data's complete context and influence the downstream task of classification or regression. Thus, representing distinctive features of a dataset forces an algorithm to learn better feature representation, which helps map a raw input to a rich feature embedding. In the machine learning domain, feature representation learning is widely distinguished into two types, namely generative and discriminative modelling. In both the feature learning tasks, a common hypothesis is that the underlying elements that account for fluctuations in the data will be captured by a decent representation. However, these factors vary depending on how these representations are learned. In the context of generative modelling, a model reproduces a realistic data sample from the distribution. In discriminative modelling, a model learns a conditional distribution given the output variable, which helps in the mapping of input to downstream tasks.

[43] described an early method for comparing related and unrelated data points in order to learn better invariant feature representations by maximizing mutual information by delineating the same data sample from different perspectives. In the work "Siamese Network" described in [44], two identical weight-sharing models based on time-delayed neural networks is used with a metric learning setup. The feature vector from two different data samples is used to compute the angle between two vectors using the cosine metric to find the distance value. In [45], the contrastive pair loss is described for discriminative models to train an invariant mapping, laying the groundwork for the contrastive learning framework. The goal is to learn a function that converts input patterns into a target space such that the L/sub 1/norm in the target space is close to the "semantic" distance in the input space. [46] introduces a

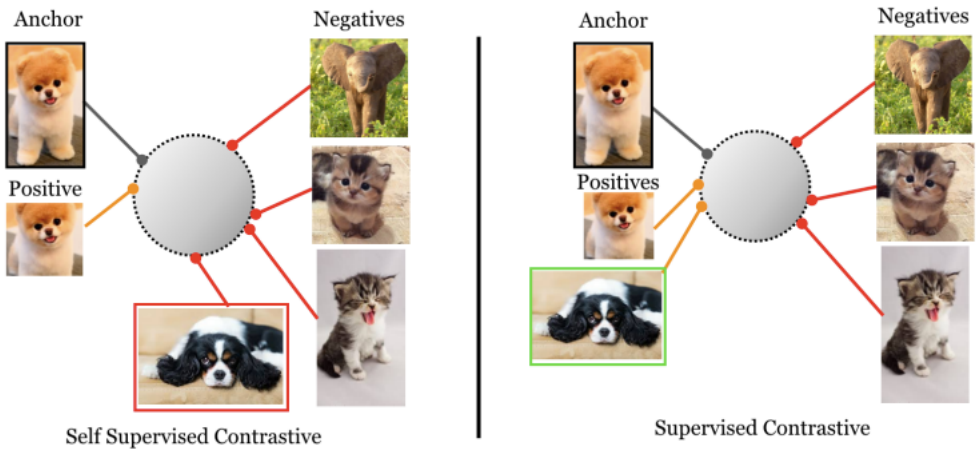


Figure 2.7.: [42], Image representing fifference between the self-supervised and supervised contrastive learning methods

concept of noise contrastive estimation, which shows the relationship between data and noise distribution to estimate the unnormalized statical model.

There are various losses defined in the contrastive learning framework with the goal of learning strong feature embedding. The loss function defined in [45] used input samples and their corresponding labels. The goal is to enforce the learning where samples with the same class labels are placed in similar embedding space while samples from different classes are represented by dissimilar embeddings space. In the work described in [47], triplet loss is introduced to learn the different poses and angles of the face. In this loss, given a anchor input $\mathbf{x}$, a positive sample is defined $\mathbf{x}^+$ which belonged to same class as anchor and negative sample $\mathbf{x}^-$ from different class as represented in equation 2.3. where ϵ is configured for setting up the distance. The objective of triplet loss was to minimize the distance between the anchor and the positive sample while maximizing the distance between the anchor and the negative sample.

$$\mathcal{L}_{\text{triplet}}\left(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-\right) = \sum_{\mathbf{x}\in\mathcal{X}} \max\left(0, \left\|f(\mathbf{x}) - f\left(\mathbf{x}^+\right)\right\|_2^2 - \left\|f(\mathbf{x}) - f\left(\mathbf{x}^-\right)\right\|_2^2 + \epsilon\right) \tag{2.3}$$

In the next section, we discuss self-supervised and supervised contrastive learning in detail. In self-supervised learning, we explore the work described in [48], simple framework for contrastive learning for visual representation (SimCLR). In supervised contrastive learning, we employ the work described [42] in the context of a gaze estimation task. Also, we discuss the loss functions used in both self-supervised and supervised contrastive setups.

2.6.1. Self-supervised Contrastive Learning

SimCLR describes the method which leverages self-supervised contrastive learning to learn the visual representation. In SimCLR, there are four main important components which represent the complete working of contrastive learning in a self-supervised setting. In this

section, we discuss in detail the components and workings of the self-supervised contrastive learning framework.

In the contrastive learning paradigm, data augmentation plays a pivotal role in creating different views of the same image by applying geometric affine and color transforms. This results in two images from the same sample with correlated views. We denote two images, namely x_i and x_j which are considered a positive pair in a given batch, while all the other samples and their correlated views are considered negative samples. This results in 2N total samples, which contain 2 positive samples and 2(N-1) negative samples, where N is the batch size. The augmentation methods vary depending on the use case, but affine transforms such as random cropping, resizing, rotation, and translation are commonly used. In color transform methods like gaussian noise, color distort is employed by tuning parameters like saturation, hue, contrast, and brightness. Data augmentation techniques are important and vary based on the downstream task. In the case of the regression task, the affine transform distorts the geometry of the gaze labels, thus distorting the ground truth labels. Thus, in our use case, we only use the color transforms.

The second component of the contrastive framework is the strong feature extractor, or base encoder, which learns the representation of input data to vector representation from agumented samples. In this work, we used FPN as described in 2.2 as a feature extractor to obtain the latent vectors. Furthermore, to the base feature extractor, the third component of the framework, a projection head is attached, which further maps the latent vector to an l2 normalized hypersphere where contrastive loss is applied. The projection head consists of a multi-layer preceptron (MLP) with one hidden layer to get the final projection vector with a length of 128 dimensions. The intention behind this step is to ensure that all the feature vectors are mapped to normalized hypersphere while achieving uniformity and preserving maximal information. This also makes sure the underlying classes are mapped onto well-clustered spherical space with a well-conditioned feature distribution. Moreover, with the projection head, the information loss is reduced, which is introduced by contrastive loss. The information loss effects important features like color or orientation of objects, which might be important for corresponding downstream tasks.

The last component is the loss function, which is applied to the projection head through the training process. The loss function is termed normalized temperature scaled cross-entropy loss (NT-Xent) which uses positive and negative samples to compute the similarity. In training step, minibatch with N samples are selected with its corresponding augmented versions resulting in total $2N$ samples in total. Here, one sample with its corresponding augmented version of a sample is treated as a positive pair, and the remaining $2(N-1)$ samples are considered as negative samples. Both positive pairs are fed to the network to get the $l2$ normalized feature vector namely u and v of 128 dimensions. Furthermore, cosine similarity or dot product is computed between two vectors sim(u, v). The loss for positive pairs is defined in the equation 2.4

$$\mathcal{L}^{\text{self}} = \sum_{i \in I} \mathcal{L}_i^{\text{self}} = -\sum_{i \in I} \log \frac{\exp\left(z_i \cdot z_{j(i)}/\tau\right)}{\sum_{a \in A(i)} \exp\left(z_i \cdot z_a/\tau\right)} \tag{2.4}$$

In the given loss function, the numerator represents the positive pairs and the denominator represents the sum of all negative pairs in the given minibatch. τ denotes a temperature parameter. τ determines the strength of penalties on hard negative samples [49]. The local

structure of each sample tends to be more separated, and the embedding distribution is probably more uniform, because contrastive loss with small temperature tends to penalize the hardest negative samples much more. The hardness-aware property vanishes as the temperature approaches positive and the contrastive loss with increasing temperature is less sensitive to the hard negative samples. Finally, the final loss is computed by summing all positive pairs and dividing by $2 \times N = views \times batch\_size$.

2.6.2. Supervised Contrastive Learning

Self-supervised contrastive learning has proved to be a significantly superior method in representation learning when compared to standard cross-entropy loss in the computer vision domain [42]. As explained in the last section, the goal is to learn the intricate features of the image by comparing it to the augmented version, resulting in better representation learning and performance. In this process, the mutual information between different views of the same image sample is improved. However, in the case where the labels are present, the self-supervised method does not leverage the label information.

In the approach described in the [42], supervised contrastive learning takes into account the label information, where normalized feature embeddings from the same class are pulled closer in contrast to the different class samples being pushed apart. In addition to using the data augmentation method to create positive samples as in the self-supervised contrastive learning method, in supervised contrastive learning, all the samples that belong to the same class are considered positive from a given batch, and the rest are considered negative samples as described in the figure 2.7. Apart from considering labels into account with loss function, the rest of the modules like data augmentation, encoder and projection head remain similar to the self-supervised method as explained in the section 2.6.1.

$$\mathcal{L}_{in}^{sup} = \sum_{i \in I} \mathcal{L}_{in,i}^{sup} = \sum_{i \in I} -\log \left\{ \frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp\left(z_i \cdot z_p / \tau\right)}{\sum_{a \in A(i)} \exp\left(z_i \cdot z_a / \tau\right)} \right\} \tag{2.5}$$

In supervised learning, the loss function incorporates the labels with a set of indices of all positive samples in a batch denoted by P(i) $\equiv \left\{ p \in A(i) : \tilde{\mathbf{y}}_p = \tilde{\mathbf{y}}_i \right\}$. In equation 2.5, the numerator describes all positive samples from the same class in a multi-viewed batch which contains both the augmented and non-augmented versions. The denominator represents all the samples which do not belong to the anchor class in the given batch. In this setting, a more robust clustering of the representation space is produced by the supervised losses, which drive the encoder to give closely matched representations to all entries belonging to the same class. z_i and z_a are l2 normalized feature vectors which are obtained from the projection head. τ denotes a temperature parameter.

$$z_i \cdot z_a = \left|\vec{z}_i\right| \left|\vec{z}_a\right| \cos\theta \tag{2.6}$$

$$\cos\theta = \frac{z_i \cdot z_a}{\left|\vec{z}_i\right| \left|\vec{z}_a\right|} \tag{2.7}$$

In both the supervised and self-supervised contrastive loss frameworks, the following explanation for the loss function holds true. In the equation 2.7, two vectors z_i and z_a are separated by angle θ. If the angle between two vectors is increased, the dot product of the two vectors decreases, and the dot product increases when the angle is decreased. In the context of bringing together and pushing apart the vectors in a contrastive setting, the similar features have a lower dot product when compared to the dissimilar feature vectors from different samples. This creates the differentiation between different samples in feature space, achieving the goal of contrastive learning.

In the loss function, we try to achieve maximization of the log term while attempting to reduce the loss function. The exponential of the dot product between images belonging to the same class is found in the numerator of the log term in the loss function, whereas the dot product between images belonging to different classes is found in the denominator. The numerator inside the log function will be raised, and the denominator will be lowered, to maximize the log term. To put it another way, the exponential of the dot product of photos within the same class is maximized, but the exponential of the dot product of image within separate classes is minimized. In order to reduce the loss, we ultimately strive to spread out vectors from different classes and bring those from the same class closer together.

3. Eye Contact Detection Framework

3.1. Eye Contact Detection Pipeline

Understanding the intention and interactions with the target object depend heavily on eye contact. It encompasses nonverbal cues that disclose important details about a user's attention span and behavioral patterns. Eye contact detection also becomes a crucial method to assess fragmented attention span and its impact on mobile device usage as the number of gadgets at our disposal increases. Robust eye contact recognition is a challenging problem because of the numerous differences in the look of face features, illumination, and domain gap. To accomplish the task of eye contact detection, the pipeline contains numerous components. In this section, we discuss in detail the training and testing pipelines with all their components to robustly achieve the task of eye contact detection as represented in figure 3.1. Furthermore, we also discuss the modules for uncertainty estimation and their application in active learning.

Training Pipeline

The gaze estimator is one of the most crucial components of the eye contact detection task. The sudo labels are generated by the gaze estimation module, and are subsequently used in the eye contact task. We investigate a contrastive learning and multitask strategy together with the gaze estimator algorithm training to create a robust multitask gaze estimator model in the training pipeline. A resnet backbone facilitates the FPN-based design of the multitask feature extractor for gaze estimation. The feature extractor also has multiple heads that is used to train contrastive loss as well as perform tasks like head posture estimation and gaze estimation. We have investigated three approaches as described in details in section 3.4, for the task of gaze estimation and compared the performance of these approaches.

The feature extractor is trained on the ETH-XGaze dataset, which contains the ground truth information of 2D gaze angle in yaw and pitch and 3D head pose angles. We optimize the complete model based on the l1 loss function for head pose and gaze estimation and normalized temperature scaled cross-entropy loss for training contrastive learning head. The multitask feature extractor performs gaze estimation and head pose estimation tasks on the input image, which provides 2D yaw and pitch angle for the gaze and 3D yaw, pitch, and roll for head pose.

Testing Pipeline

Based on approaches for gaze estimation as described in 3.4, we employ the pretrained FPN based feature extractor in the testing and inference pipeline. The image normalization module is used in the initial stage of the testing process in addition to the feature extractor which contains the face detector as defined in 1.1 and also contains image normalization procedure as described in 3.2 . Image normalization module detects and locates the face in an image. We use only one face in the image, if the image contains multiple faces. The confidence

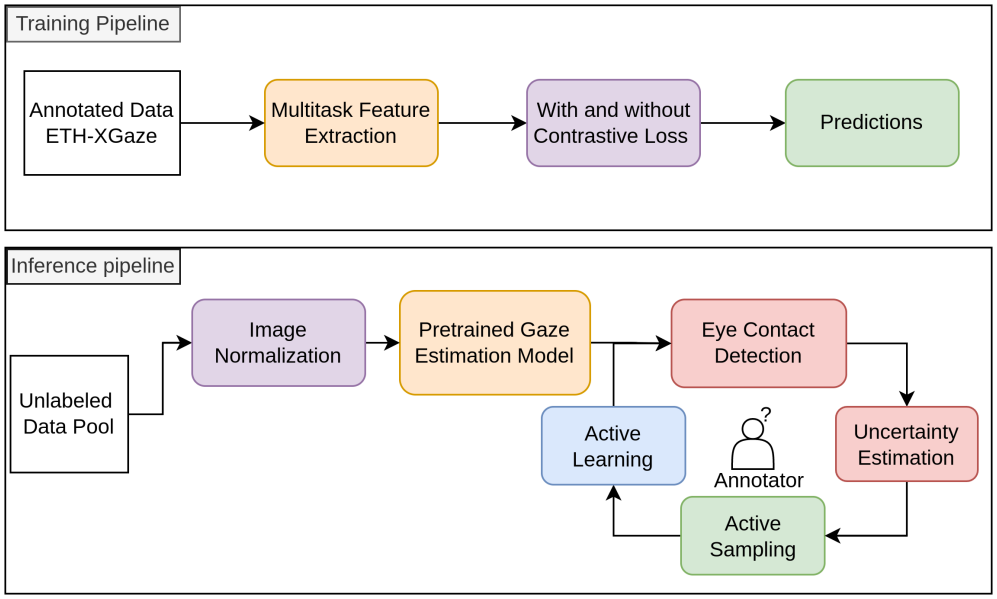


Figure 3.1.: An overview of eye contact detection training and testing pipeline with respective modules

of the face detector is set to 85%. To further process the face image in the normalization procedure as mentioned in the section 3.2 the face localization information in the form of a bounding box is utilized to crop just the face image and then normalization procedure is performed.

The normalized image is then sent to the pre-trained feature extraction module, where we obtain the 2D gaze information and its corresponding uncertainty values of each gaze point based on dropout method as mentioned in 2.3. In the next part, this 2D gaze information along with its uncertainty information is used to create the sudo labels using a clustering operation. Multiple experiments are conducted while eliminating specific percentage of the gaze estimates with the highest uncertainty which not taken into consideration during the clustering operation. The gaze points, which lie at the center of the plane and their corresponding points within a certain cluster, are considered as positive labels denoting eye contact, while points outside the cluster are considered to have no eye contact.

The user is given a list of the images that are most uncertain in the final stage of inference, which is based on the uncertainty determined by the gaze estimation algorithm. In order to improve the performance of the final eye contact detection model, the user can retrain the SVM-based eye contact detection model using updated labels. In this effort, we also created a straightforward PyQt5 GUI for tagging eye contact data.

3.2. Image Normalization

Appearance-based gaze estimation has gained a lot of traction due to advances in deep learning and computer vision domains. Moreover, appearance-based gaze estimation also holds huge potential in HMI and medical applications. In the appearance-based gaze estimation method, a full-face image is mapped to gaze direction by learning underlying features for

accurate and robust gaze detection and estimation. However, the major problem that arises in the appearance-based gaze estimation method is the possibility of covering all head poses and gaze directions for the training robust gaze estimator. This is possible only when the dataset contains all the diverse and possible combinations. Moreover, in real-world cases, the collection of huge amounts of data is non-trivial. In this work, we use the image normalization procedure only during the inference where the ground truth gaze label information is not present.

To solve the abovementioned problem, data normalization is introduced in [50] to make the training more efficient, thus avoiding the task of collecting huge, diverse datasets. Furthermore, the normalization procedure described in [50] also reduces variability caused by head rotation and translation. In addition to the head pose variability, the distance between the virtual camera and the face also affects the appearance of the image. The normalization procedure is applied before the gaze estimator. The main concept is to use camera rotation and scaling to uniformly translate and rotate images between the camera and face coordinate systems.

The head pose of the face image is computed before applying normalization in order to use rotation and translation information in the image normalization procedure. The head pose of face in the image is computed by solving the perspective-n-point problem (PnP) as described in 1.4.3, given the facial landmarks and a generic 3D face model, as well as its extrinsic and intrinsic camera parameters. Extrinsic camera parameters, such as aperture and focal length, are independent of internal camera parameters and are affected by the camera's position and orientation. The image retrieval process is influenced by camera-specific intrinsic factors. The intrinsic parameters that affect the intrinsic matrix of a camera model include focal length, aperture, field-of-view, resolution, and others. Extrinsic parameters are used to convert world coordinates in 3D to camera coordinates in 3D, whereas intrinsic parameters are used to convert camera coordinates in 3D to pixel coordinates in 2D.

In the normalization procedure, the head coordinates are defined by points connecting the eyes and the end of the mouth. The x-axis is defined by a line connecting both the centers of the eyes, and the y-axis is represented as a line at right angle to the x-axis inside the triangle plain from the eye to the mouth. And the z-axis is perpendicular to the triangle with back projection into the plane as described in the figure 3.2. For simple notations, the midpoint of the right eye serves as the origin for the head coordinate system. The rotation and translation from the camera coordinate system to the head coordinate system are denoted by e_r and R_r. There are three main conditions that are necessary to meet the image normalization procedure. In the first condition, the center of the eye is situated at the center of the normalized image as the normalized camera faces the origin of the head coordinate system. In the second condition, the x-axis of the head coordinate system appears as a horizontal line in the normalized image because the x-axes of the head and camera coordinate systems are in the same plane. And the last goal is to locate the virtual camera at a fixed distance d_n from the eye center using focal length as steps represented in the figure 3.2.

The aim is to calculate the conversion matrix M = SR from the input image I and the location of the reference point x. R is the inverse of the rotation matrix that rotates the camera to face the reference point and aligns the x-axes of the head coordinate system with the camera coordinate system. Furthermore, the camera coordinate system's rotated z-axis z_c must be e_r. For second condition to meet, the y-axis is defined as $y_c = z_c \times x_r$, where x_r is x-axis of the

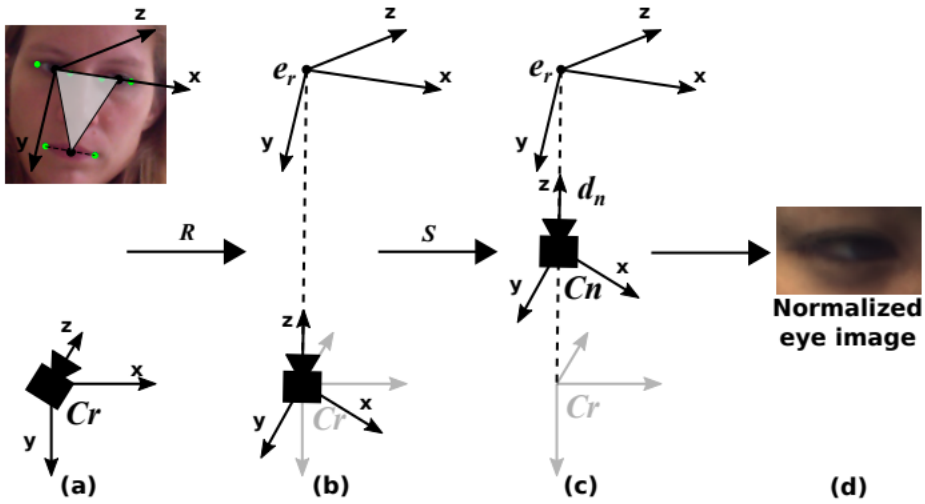


Figure 3.2.: [50]

head coordinate system and y-axis of rotated camera is modified to be perpendicular to both z_c and x_r. In the next step, the x-axis of the rotated camera is defined as $x_c = x_c \times z_c$. By taking into consideration all the vectors, namely x_c, y_c and z_c; a rotation matrix is obtained as defined in equation 3.1. The scaling matrix to meet third condition and for computing overall transformation matrix $M = SR$ is computed as given in equation 3.2.

$$\boldsymbol{R} = \left[\frac{\boldsymbol{x}_c}{\|\boldsymbol{x}_c\|} \cdot \frac{\boldsymbol{y}_c}{\|\boldsymbol{y}_c\|} \cdot \frac{\boldsymbol{z}_c}{\|\boldsymbol{z}_c\|} \right] \tag{3.1}$$

$$\text{diag}\left(1, 1, \frac{d_n}{\|\boldsymbol{e}_r\|}\right) \tag{3.2}$$

The conversion matrix M rotates and scales any 3D points in the input camera coordinate system to the normalized coordinate system. The same transformation is applied for the given input image I to meet the above mentioned conditions of normalization with perspective wrapping $W = C_s \times M \times C_r^{-1}$. A predefined parameter named Cs defines the camera projection matrix in the normalized space. Cr is the projection matrix matching the input image obtained from a camera calibration.-

3.3. Gaze Estimation Network

In this task, we used Resnet and Resnet based Feature Pyramid Network to train the feature extractor for the task of gaze estimation. In FPN, we used three approaches which included self-supervised and supervised contrastive learning and Multitask task approach. For all the three approaches as mentioned, we used same FPN based feature extractor with similar hyperparamerter, model parameter and training procedure.

Architecture

The neural network learns the intricate details of the data to map the input N-dimensional input into an M-dimensional output. In the case of appearance-based gaze estimation, the task is to regress the input face image to 2D gaze with yaw and pitch angles while also learning features that contribute to the task. The architecture in general contains multiple layers, and each layer is composed of multiple neurons and activation functions, the basic elements in a neural network.

In the case of convolutional neural network based architecture, the layers are stacked with the input layer, set of convolution layers followed by pooling layer which are also called hidden layers and output layer for final downstream task. The input layer receives the images in batches with a size of 224x244 pixels to process many images in one batch to minimize the variance of parameter updates and maximize the use of highly optimized matrix optimizations during training. The convolutional layers have a kernel which performs feature extraction, which extracts all the low-level and high-level features from the input image while storing the information in feature map channels. The pooling layer is responsible for the down-sampling operation where we use either max pooling or average pooling to reduce the spatial size of the image. The output layer is composed of a multi-layer perceptron or linear layer with an activation function that finally performs the downstream task of regression or classification. For our task, the final output layer in gaze estimation heads contains two neurons for yaw and pitch estimation, and the head pose estimation head contains three neurons for yaw, pitch, and roll predictions. For the contrastive loss projection head, the output is a 128-dimensional latent feature vector.

As explained in the section 2.1, we used Resnet and FPN architecture to perform the gaze estimation task, while also comparing the performance of both the architecture.

Regularization

Regularization is an important technique used in the machine learning domain to overcome the problem of overfitting while training the neural network model. Overfitting is caused due to over-parameterization of a model where in the model is too big or the data is too small, so it memorizes the data while not really learning the underlying features of the data, leading to poor generalization. To avoid the problem of overfitting, various methods like dropout, weight decay, data augmentation, and early stopping are used.

We used data augmentation techniques like random blur, changing the color correction methods like saturation, hue, and contrast. We did not use any affine transformation method as it would result in a change in the underlying gaze and pose geometry, which would affect the gaze and pose estimation tasks. Data augmentation techniques are also a preprocessing step while training contrastive loss.

As a part of the architecture, batch normalization is also used in the hidden layers of both resnet and FPN based on resnet to tackle the problem of internal covariant shift. This is caused when the distribution of the data changes when it flows from one layer to another. This can cause the training to slow down, resulting in a very long time to reach or converge to a global minimum. The batch normalizes the incoming data into zero mean and unit variance with two learning parameters which normalize the data.

We also use a dropout layer at the last layer of the network before feeding the logits to the respective heads for gaze and head pose estimation. It is one of the important but effective regularization approaches for handling overfitting, making it widely employed in the deep learning field. The dropout method, in our case, is used for two reasons. The first reason is to prevent overfitting of the model as a regularization method. The second reason is to use the Monte Carlo dropout method [39] for uncertainty estimation. In dropout, the nodes or neurons of a particular layer are turned off or masked with certain dropout percentage. As a consequence, in the regularization case, it reduces the codependency of the neurons during the training, making the network more robust. In the case of uncertainty estimation, the dropout is switched on during inference, which forces the network to behave like a different network while acting as an ensemble of networks. This provides the method to learn the predictive distribution as explained in the section 2.3.

Loss Functions

Selection of loss function is crucial step in training the neural network model. This primarily affects the learning and accuracy of a model. In supervised learning methods, gradient descent is used to train a model based on the backpropagation approach in order to achieve convergence to the global minimum under ideal circumstances. The loss function or error function is used to evaluate the predictions of the network by comparing it to the ground truth information. This provides the amount of error in the prediction of a models describing how far the true value is from the prediction which can then be reduces using optimization. The smaller loss represents the better model which is performing good on training data and vice versa. To train the models, we used l1 loss or MAE for gaze and head pose estimation defined in 3.3

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.3}$$

Where N is the batch size, y_i is ground truth and $\hat{y}_i$ is the prediction of the model.

To train contrastive learning models, we used normalized temperature scaled cross entropy loss for self-supervised and supervised methods as described in 3.4 and 3.5 respectively which is explained in details in sections 2.6.

$$\mathcal{L}^{\text{self}} = \sum_{i\in I}\mathcal{L}_i^{\text{self}} = -\sum_{i\in I}\log\frac{\exp\left(z_i\cdot z_{j(i)}/\tau\right)}{\sum_{a\in A(i)}\exp\left(z_i\cdot z_a/\tau\right)} \tag{3.4}$$

$$\mathcal{L}_{in}^{sup} = \sum_{i\in I}\mathcal{L}_{in,i}^{sup} = \sum_{i\in I}-\log\left\{\frac{1}{|P(i)|}\sum_{p\in P(i)}\frac{\exp\left(z_i\cdot z_p/\tau\right)}{\sum_{a\in A(i)}\exp\left(z_i\cdot z_a/\tau\right)}\right\} \tag{3.5}$$

Optimization

Optimization is rudimentary task in deep learning to find the optimum value in order to minimize the loss function. The most commonly used optimization algorithm in deep learning is

gradient-based approach. In gradient based approach, the gradient is computed of a loss function 3.6 with respect to a variable which signifies the small change in the value of the variable which is required to achieve the minimum error. This small steps are taken to descend a loss curve in negative slope direction 3.7.

$$L(\theta) = \frac{1}{N} \sum_{n=1}^{N} L((x(n), y(n)); \theta) \tag{3.6}$$

$$\theta^{t+1} = \theta^{t} - \gamma^{t} \nabla L(\theta^{t}) \tag{3.7}$$

where θ is the unknown parameter with weights W and bias b. t is the iteration index, γ is the learning parameter. Finding the global minimum in a convex function and minimizing the cost function are the basic goals of gradient descent. However, with real-world data sets, locating a global minimum is challenging. Moreover, algorithms like gradient descent and stochastic gradient descent (SGD) lack few functionality to deal with the ill conditioned problems. Thus, numerous new optimization techniques with significant improvement over gradient descent and SGD are applied to address this difficulty. We use Adam optimizer for gaze estimation task and use LARS while training the contrastive learning methods.

3.4. Approaches

In this work, we investigate multitask and contrastive learning approaches in order to achieve the robust performance of the gaze estimation algorithm for the eye contact detection task. In contrastive learning, we apply the contrastive learning framework to the regression task of gaze estimation using both supervised and self-supervised contrastive learning approaches.

Metrics for Performance Evaluation of Gaze Estimation and Eye Contact Detection

To evaluate the gaze estimation models, we use gaze angular error which is computed via cosine similarity given by equation 3.8.

$$\theta = \cos \frac{a \cdot b}{|\vec{a}| \, |\vec{b}|} \tag{3.8}$$

For evaluating the eye contact detection binary classification model based on SVM, a confusion matrix is employed which is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one, in the field of machine learning. It consist of 4 elements as follows. True positive (TP), number of actual positive class correctly classified as positive, True Negative (TN), number of actual negative class correctly classified as negative, False positive (FP), number of actual negative class incorrectly classified as positive and False Negative (FN), number of actual positive class incorrectly classified as negative.

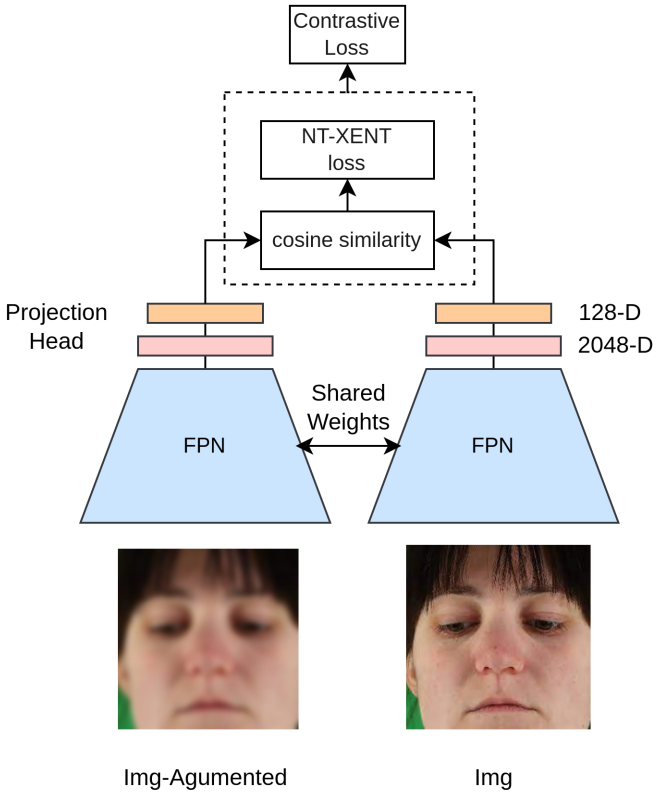


Figure 3.3.: Self-supervised contrastive framework with projection head and NT-XENT loss

Precision is defined as the proportion of true positives among all detected positives, or simply TP/(TP+FP). Recall is the number of correctly classified true positives TP/(TP+FN). F1 score is harmonic mean of precision and recall.

Specificity (True negative rate) refers to the likelihood of a negative test if it is truly negative.

Mattews-Correlation Coefficient (MCC) is defined by equation 3.9. The MCC score is a popular performance metric for binary classification problems. The MCC provides more information than other metrics because it considers the balance ratios of the four confusion matrix categories (TP, TN, FP, FN). A MCC of +1.0 represents perfect predictions, -1.0 represents total disagreement between predictions and observations, and 0 represents random guessing.

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TN} + \mathrm{FN})}} \tag{3.9}$$

3.4.1. Gaze Estimation on Self-Supervised Contrastive Learning

In this approach, we trained a multitask model on a gaze estimation regression task using a self-supervised contrastive strategy. In the classical contrastive learning method, a first-stage feature encoder is trained using multi-view samples, where an image with its augmented version is used to train a network along with its projection head to get the latent feature representation of both the images, as described in detail in section 2.6. However, in our

implementation, we used both the two-stage contrastive learning approach and also investigated the possibility of using a multitask model to train the contrastive loss along with the gaze estimation task with a single-stage method.

To train the contrastive loss in the conventional two-stage approach, we employed FPN based on the resnet backbone as the feature extractor. As described in the figure 3.3, two image samples are used, one with the original view and the second image as an augmented version of the first image. In the augmented versions of the image samples, we avoid applying any geometric alteration that could possibly alter the geometry and alignment of the eyes in order to retain the gaze direction. With a batch size of N, we sample one image and its augmented version, considering the positive pair and all the reset 2(N-1) images as negative samples. These pairs of images are fed to an FPN-based feature extractor. The logits obtained from the concatenation of feature maps from multiple levels of FPN from both the image samples are further fed to the projection head. The projection head is a multi-layer preceptron (mlp) layer with an output dimension of 128 where we obtain the final latent representation of both the image samples, which are further confined to a unit hypershpere by using l1 normalization. These latent representations are further used to compute the contrastive loss, which in our implementation was normalized temperature scaled cross-entropy loss. Later, the self-supervised trained weights were used to further train the downstream gaze estimation regression task to train the model.

In the second case of single-stage contrastive learning implementation, we used a multitask model with two mlp heads for training the contrastive loss and for training the regression task of gaze estimation simultaneously. In this method, we used an almost identical procedure using the same FPN backbone architecture as a feature extractor. However, in the training procedure, we used both the NT-XENT loss along with l1 loss for 2D gaze regression as represented in equation 3.10 to optimize the models for both the heads.

$$L_{total} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| + \sum_{i \in I} -\log \frac{\exp\left(z_i \cdot z_{j(i)}/\tau\right)}{\sum_{a \in A(i)} \exp\left(z_i \cdot z_a/\tau\right)} \tag{3.10}$$

To train the models in both the abovementioned methods, we used a layer-wise adaptive rate scaling (LARS) optimizer with a batch size of 128 for 50 epochs. LARS is used when using a large batch size as expected for contrastive learning frameworks or stable training processes. For each layer rather than for each weight, LARS employs a different learning rate. Second, for better management of training pace, the amount of the update is regulated in relation to the weight norm. We used a learning rate of 0.0001 for a step learning rate decay of 0.1 at 20 epochs. We saved the top 3 best performing checkpoints using the callback function.

3.4.2. Gaze Estimation on Supervised Contrastive Learning

In supervised contrastive framework, we use the same methods as described above 3.4.1 with classical two-stage and single-stage approaches, with a multitask model in the latter case, in the supervised contrastive learning approach for gaze estimation. However, the major difference was converting the continuous variable labels of gaze estimation using the binning method to discrete classes for supervised setting. In this section, we describe the procedure for using supervised contrastive loss in the gaze estimation regression task.

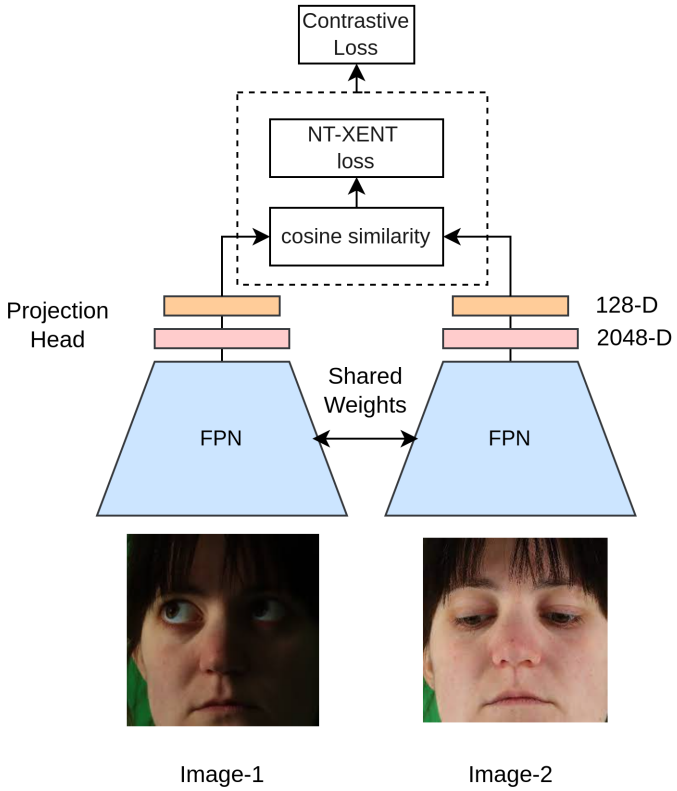


Figure 3.4.: Supervised contrastive learning framework with projection head, leveraging class labels

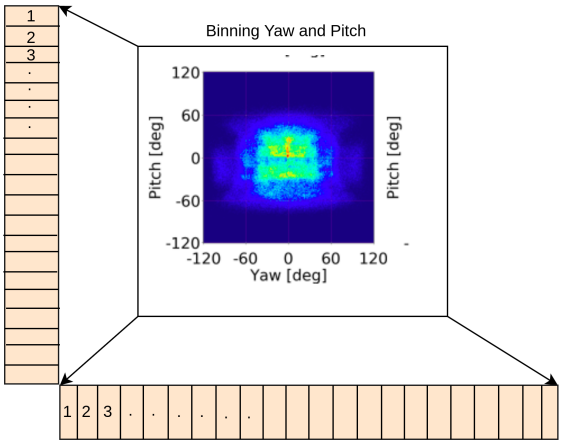


Figure 3.5.: Binning continuous variable into discrete 2D gaze yaw and pitch class labels with bin size of 4 degree

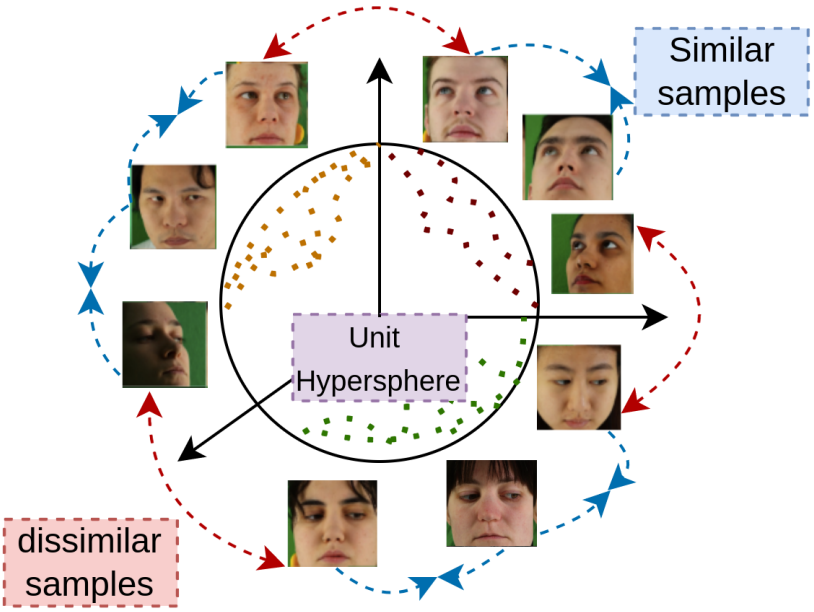


Figure 3.6.: Supervised contrastive learning in feature space embeddings. All the features are enforced in unit hypersphere.

In this method, the goal is to bring the images with similar gaze directions as close as possible in feature representation space while pushing apart the gazes that are not similar. This operation can be depicted in the figure 3.7, where in a given hyperpsphere, the blue arrows directing the image are similar samples which are brought together, and the red arrows denote the dissimilar samples which are to be pushed away in the feature space representation. As a consequence, the feature space should be improved, making the gaze estimation task more robust to out-of-domain data while also improving the overall performance of the gaze estimation task.

In a two-stage setting, we trained the FPN using contrastive loss with the projection head. To convert the continuous yaw and pitch variables into classes, we use the method of binning. In binning, we use the yaw and pitch angles as labels, which vary from -120 to 120 degrees. The corresponding angles are binned into a size of 3 degrees, as described in figure 3.5. In this way, we convert the continuous variable pitch and yaw angle to discrete bins, with each bin of 3 degrees considered a class. This information is further used as a class label to train a supervised contrastive loss. As mentioned earlier, we used the data augmentation method to create the positive and negative pairs. Moreover, in this case, we also take into consideration the class labels. The images that belong to a specific bin are considered as images from the same class; thus, a positive sample and the rest of the images are regarded as negative samples in a given batch 3.4. The normalized latent variable as described earlier is obtained from the projection head with the supervised NT-XENT loss as described in the equation 3.5. In a similar way to the self-supervised setting, the trained weights are further used to train the downstream task of gaze estimation.

In the second method of single stage, we use an exactly similar method as described in the self-supervised setup with multiple heads to train the supervised contrastive loss and gaze estimation tasks represented by equation 3.11, and a similar training procedure is carried out with hyperparameters and model parameters as described in the section above.

$$L_{total} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| + \sum_{i\in I} -\log\left\{\frac{1}{|P(i)|}\sum_{p\in P(i)}\frac{\exp\left(z_i \cdot z_p/\tau\right)}{\sum_{a\in A(i)}\exp\left(z_i \cdot z_a/\tau\right)}\right\} \tag{3.11}$$

3.4.3. Gaze and Head Pose Estimation on Multitask Model

In this section, we describe the hard parameter sharing multitask model for gaze estimation and head pose estimation. The main goal is to learn strong feature embeddings in the feature representation space. We postulate that the gaze and head pose have strong correlation in the final task of gaze estimation. Thus the performance of the gaze estimation task strongly depends on the direction in which the person is facing thus the weights learning contributes to the concept of hard parameter sharing with collaborative learning.

In our approach, the multitask model consist of the same FPN archiceture based on resnet backbone with two heads. One head is responsible for training the head pose estimation task and second head is employed to train the gaze estimation task. The loss function for both 3D head pose estimation and 2D gaze estimation are mean average error loss or l1 loss as described in equation 3.14. Both the loss functions are optimized from the respective heads during the training process. We use similar hyperparameters and model parameters in this approach as described in the above mentioned methods with contrastive learning framework.

$$MAE_{gaze} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.12}$$

$$MAE_{headpose} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.13}$$

$$L_{total} = MAE_{gaze} + MAE_{headpose} \tag{3.14}$$

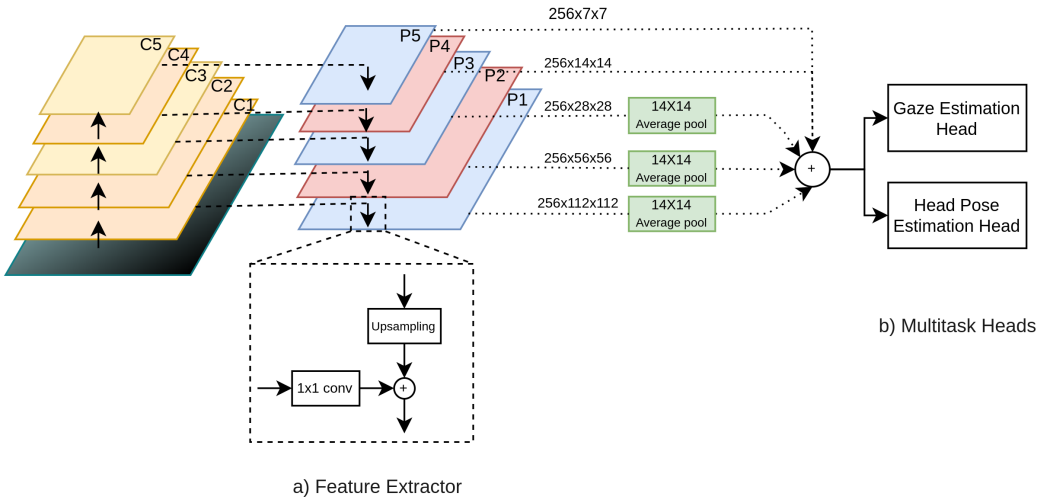


Figure 3.7.: FPN based Multitask model with two heads for 2D gaze and 3D head pose estimation.

4. Active Learning

4.1. Active Learning

Active learning aims to improve the performance of supervised learning neural network models while taking into consideration the cost involved in annotating the samples. Annotation is a non-trivial job, as it requires a huge amount of time and effort. Moreover, it also requires some amount of domain knowledge to annotate the data. With an ever increasing focus on deep learning, a huge amount of effort is put into producing large annotated datasets. As a consequence of large datasets, supervised deep learning models have shown remarkable strides in different domains by achieving state-of-the-art performance. However, deep learning models require a huge amount of high-quality labeled data to achieve superior performance. Moreover, to solve the real world applications, for example in medical or autonomous driving, the models still require application or domain-specific data to train them. Hence, a better and more efficient annotation framework is required. And the problem of data annotation leads to the attention of active learning frameworks, which try to reduce the cost involved in annotation.

Active learning explores the datasets while understanding the importance of each data sample, which contributes to better feature learning while training the model while achieving better performance with the lowest cost spent on annotation. In order to minimize labeling costs while retaining performance, active learning seeks to specifically choose the most useful samples from the unlabeled dataset and pass them along to the human annotator for labeling. Active learning strategies are majorly divided among three categories, namely membership query synthesis [51], stream-based selective sampling [52] and pool-based sampling [53]. In the membership query synthesis method, the learner can ask to query the label of any unlabeled sample in the input space, even the sample that the learner generated. In the stream-based method, an assessment is carried out in a sequential manner to make a query decision. However, in the pool-based method, the query selection is executed based on the evaluation and ranking of all the samples in the dataset. The stream-based method is mainly used in edge devices that require strict timing constraints due to limited computing and strong capability. However, the most common approach is the pool-based approach, which is used in our work. In the pool-based approach, one or more samples are randomly chosen from an unlabeled data pool and provided to a human annotator or oracle to annotate and create the labeled dataset. This dataset is then used to train the initial model. Based on the performance of the trained model and knowledge gained, the next set of samples are queried from the unlabeled pool and submitted to the oracle. This additional labeled dataset is then used to further train the model until the predefined conditions are met with good performance.

In the pool-based method, there are various querying strategies that are used based on the application and use case scenarios. In the density-based querying method, the representation describing the underlying distribution of the entire dataset is taken into consideration

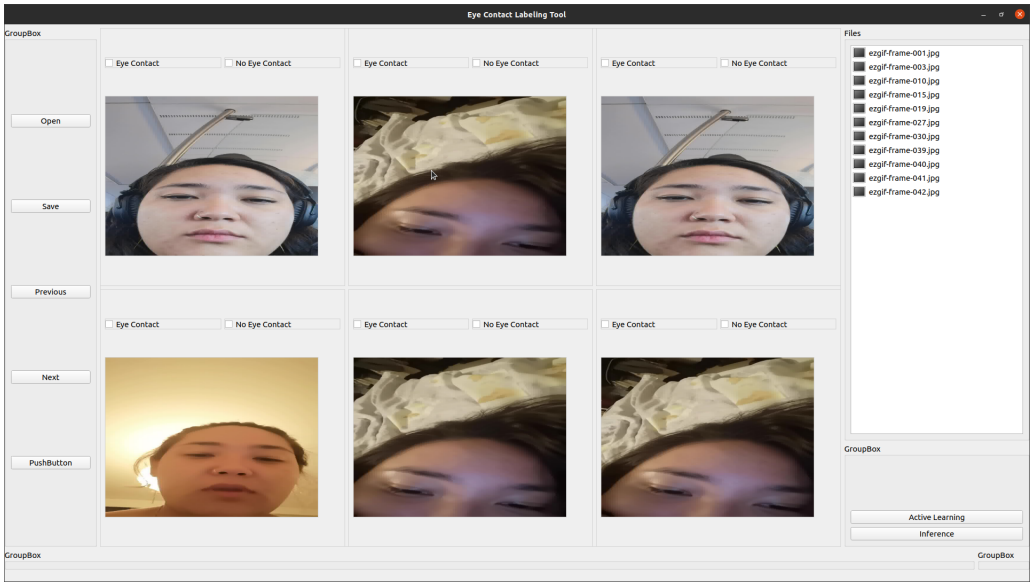


Figure 4.1.: PyQt5 based GUI for eye contact labeling and inference

for sampling the samples. In the deep bayesian method, a bayesian convolutional neural network is used for active learning, which performs Gaussian prior modelling on model weights followed by variational inference.

In our work, we employ the uncertainty-based query strategy. In this method, we use the Monte Carlo-dropout method as described in section 2.3. In the inference stage, we get the uncertainty of each gaze point sample for gaze estimation. For each sample, an uncertainty deviation is computed, which represents by what amount the model is uncertain about the 2D gaze estimation. The uncertainty is computed for both yaw and pitch angles. This information on gaze uncertainty is then used to select the sudo labels while performing the clustering operation. The most uncertain samples are excluded from the computed gaze labels. In this way, we reduce the amount of uncertain data that can influence the binary classification task of eye contact detection. The uncertain samples are further provided to the human annotator where the eye contact labeling is performed using the eye contact labeling tool.

We have created a GUI based on PyQt5 to label the eye contact data as described in the figure 4.1. The GUI provides basic functions to labels to eye contact or no eye contact and assigns binary labels to images. The GUI can also be used to run the inference on images which contains in a certain folder. Our inference pipeline as described in the section 3, provides the 2D gaze predictions with it corresponding uncertainty. The list of images with highest uncertainty is provided to the user. The information can be further use to label the images and retrain the SVM.

5. Results

5.1. Datasets

5.1.1. ETH-XGaze

For training the gaze estimation feature extractor, the ETH-XGaze dataset [54] is used. ETH-XGaze is a large-scale dataset for gaze estimation with more than 1 million samples encompassing 110 people of various ages and genders who participated in the large head posture and gaze survey with consistent label quality and high-resolution photos of their ethnicity. However, we used data from 68 participants for training. Participants' racial and ethnic backgrounds include Caucasians, Africans, East Asians, South Asians, and Middle Easterners.

The ETH-XGaze contains the sample of every individual with 90 gaze points in the dim-lighting situation and 525 gaze points in the full-lighting with an additional fifteen different lighting conditions, with six gaze spots for every one of them. The 18 distinct cameras gathered a total of 18 photos for each look point. The head pose variation for this dataset ranges between -80 to +80 degrees for both yaw and pitch. Moreover, gaze directions range between -120 to +120. The dataset comes with multiple image resolutions ranging from 112x112 to as high as 6000x4000. To train our gaze estimator, we use 224x224 resolution images.

Due to restricted access of ground truth labels for the testing dataset, a subset of training data is used which contains around 200 thousand images of 11 users for testing the gaze estimator.

Figure 5.1.: Images in ETH-XGaze dataset with various headpose and gaze direction. [54]

Figure 5.2.: Images contained in EMVA dataset with different users and variable environment conditions. [55]

5.1.2. Eveyday Mobile Visual Attention (EMVA)

In the inference stage, we use the EMVA dataset [55] to evaluate the performance of the eye contact detection algorithm. The EMVA dataset contains 14,322 video data totalling around the playback time of 472 hours. Each video is captured at a resolution of 720 pixels wide by 1280 pixels hight. The videos were recorded in outdoor and indoor environments with varying illumination conditions. The video data is also associated with meta-data and sensor data of the respective device.

The dataset has a total of 32 participants with different racial and ethnic backgrounds. The majority of the participants were right-handed, with the exception of three left-handed individuals. The majority of them were undergraduate, graduate, and doctoral students, but we also had a few from other professions. These participants also used different devices to record the data.

EMVA also contains around 15 thousand labeled images for evaluating eye contact detection task. Each images corresponds to one of three labels, 0 for no eye contact, 1 eye contact and -1 if the unsure. We use this data to train our eye contact detection SVM model.

In this section, we discuss the results obtained from training and validation of gaze estimation algorithm and also the eye contact detection algorithm with all the approaches used.

5.2. Gaze Estimation

The findings of the various methods to the gaze estimation challenge employing self-supervised, supervised contrastive learning, and multitask models are discussed in depth in this section.

5.2.1. Resnet VS FPN

Architectures play a significant role in feature learning in deep learning, and effective feature learning leads to robust model performance. For the gaze estimation challenge in this study, we used FPN with a ResNet backbone architecture. The FPN's performance was compared to that of the ResNet-50 model. With an appearance-based gaze recognition algorithm, we learn the features of the face and the eyes to translate the face image into the direction of the glance. The fundamental objective and premise of FPN was to more effectively learn the tiny features, such eyes, which can lead to more accurate gaze detection. On labeled data from ETH-XGaze, we trained the FPN model and the resnet-50 model both from scratch and using previously trained image-net weights. We used the gaze angular error metric 3.4 to assess the training models and determine how well the gaze estimation algorithm performed.

The FPN model when trained from scratch with any pre-trained weights performed superior to resnet-50 with FPN achieving the gaze angular error of 6.3 degree when compared to resnet-50 with the 7.923 degree. The gaze angular error for FPN improved by 11.23% over resnet-50. This was a significant increasing in the performance.

Models	Gaze Angular Error
Resnet-50	7.923°
FPN	6.3°

Hence in the all the further experiments with the contrastive learning framework, we used FPN to train the models.

5.2.2. Self-supervised Contrastive Learning

In this part, we go over the outcomes of a self-supervised contrastive loss task used to estimate gaze. In this experiment, we evaluated employing a single stage contrastive learning strategy with a multitask model as well as a two stage classical contrastive learning approach.

In the two stage approach as described in details in the section 5.3.1, the models did not perform well both in the case where we first trained the FPN on self-supervised contrastive loss as described in the section 5.3.1 and froze the weights and second also when we did not freeze the weights in the subsequent downstream task of gaze estimation. The model was unable to converge hence achieving extremely bad gaze angular error results as described in the table 5.3.1.

Two Stage Contrastive Learning	
Models	Gaze Angular Error
FPN (weights frozen)	89.91°
FPN (weights not frozen)	88.48°
Single Stage Contrastive Learning	
FPN (improved)	6.8°

However, when compared to the two stage technique, the model's performance was substantially better when it was trained using a multitask strategy with a single stage, and the validation gaze angular error was 6.8 degrees.

5.2.3. Supervised Contrastive Learning

Similar to the self-supervised method, we used supervised contrastive learning method which leverages the class labels for contrastive learning as described in the section 5.3.1. We used similar two stage and single stage methods also in this case to train the model for gaze estimation task.

The model performance in the first scenario using the two stage procedure was comparable to the self-supervised method with very poor validation gaze angular error when the weights were frozen. The results were improved when the weights were not frozen, but the validation gaze angular error was still too high compared to the vanila FPN performance at 6.917 degree.

Two Stage Contrastive Learning	
Models	Gaze Angular Error
FPN (weights frozen)	89.82°
FPN (weights not frozen)	6.917°
Single Stage Contrastive Learning	
FPN (improved)	6.02°

However, in the case of single stage contrastive learning approach, we acheived slightly better results with validation gaze angular error of 6.02 degree which was better than vanila FPN with 6.323.

5.2.4. Multitask Model for gaze and head pose estimation

This part covers the last method for gaze estimation, where we investigated into the usage of a multitask model rather than a contrastive learning approach.

With this technique, we trained the multitask model to predict the 2D gaze direction with pitch and yaw as well as the 3D head position with pitch, yaw, and roll with two different heads for each task, separately. The model was trained with two l1 losses for gaze estimation and head pose estimation as described in the section 5.3.1.

In comparison to all of the aforementioned approaches, the multitask model produced the best results for the gaze estimation test. When compared to the baseline, the validation gaze angular error for the FPN model trained on the imagenet pretrained weights was 5.577 degrees.

Models	Gaze Angular Error
Multitask FPN (pretrained)	5.577°
Multitask FPN (no pretrained)	5.735°

5.2.5. Feature Maps

In this section, we discuss the feature maps obtained from all the levels of FPN to visualize the feature learning process as represented in the figure 5.3.

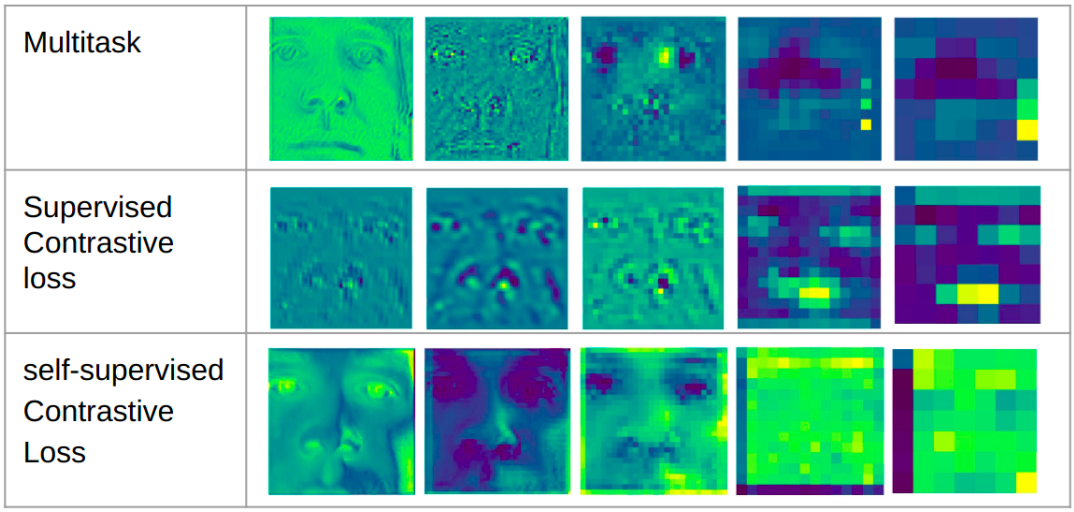


Figure 5.3.: Feature maps obtained from the different levels of FPN model for gaze estimation task

The first column describes the multitask model's feature map, which shows the gradients' focus on the eye area. This represents the focus of feature learning on the eye part from feature maps three to five. The first two feature maps in the case of supervised contrastive learning show the overall features captured from the face image, and the last two feature maps show the focus of the feature learning on the eye and also on the nose part. In the final case of self-supervised contrastive learning, the features in the first two feature maps describe the focus on the overall face, but unlike the previous two methods, the features in the last feature maps are not well defined.

5.3. Eye Contact Detection

In this section, we describe the results achieved from the eye contact detection task. For obtaining the sudo labels, we use the above mentioned approaches for gaze estimation task 3.4 and use the respective gaze estimation models which attained good gaze estimation performance. To train the eye contact detection SVM based model, we use the sudo labels or gaze projection obtained from the previous pre-processing step of gaze estimation model and OPTICS clustering. For inputs to the SVM, we use the feature embeddings of 512 dimension obtained from the face detection model. We quantify quantitative eye contact results using MCC and Confusion matrix metrics to evaluate the performance of the eye contact detection task. For qualitative evaluation, we investigate the 2D gaze projections obtained from gaze estimation algorithm.

We use our above mentioned gaze estimation approaches with its corresponding eye contact detection model. For comparing the results, we also perform evaluation with two state-of-the-art methods namely ETH-XGaze [54] and L2CSNET [19].

5.3.1. Eye Contact With Different Approaches

In the eye contact detection task, we used pre-trained models from all the above mentioned approaches for gaze estimation and clustering operations to obtain sudo labels to train the eye contact detection model. In this subsection, we describe the results achieved from all three approaches and also compare the performance to the additional two state-of-the-art methods.

The figure 5.4 describes the gaze projections obtained from different approaches. The red dot or gaze prediction on the 2D plane represents eye contact and the blue dot represents no eye contact. The size of each point differs based on the amount of uncertainty, which is computed using the MC-dropout method as described in section 2.3. The size of the gaze prediction is larger if the uncertainty is greater and smaller if the uncertainty is less. For reference, the figure 5.3.1 represents the least uncertain and most uncertain images in the case of a multitask model.

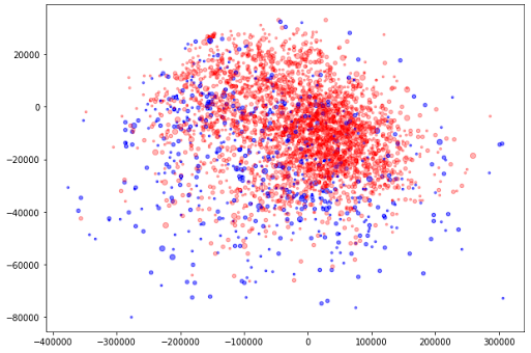
In figure 5.5, the cluster or sudo labels obtained from the OPTICS clustering operation are represented. For each approach, we alter the eps parameter based on the cluster shape. The red projection describe the no eye contact labels denoted by -1 and the green projection describes the eye contact with label 1. These labels, along with the features, are used to train the SVM for eye contact detection. The data is split into train and test data, with 75% of the data used for training and 25% used for testing the SVM model.

As described, we quantify the results in quantitatively using the metrics and qualitatively using the gaze projection and cluster obtained. This table 5.3.1 represents the performance of all the approaches using the ground truth labels **(GT)** and sudo-labels **(sudo)**.

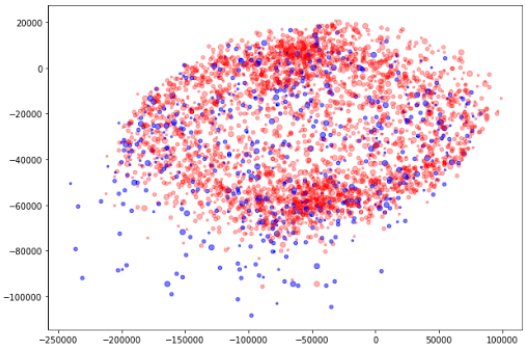
Describe the results HERE

Models	MCC (sudo)	Specificity (sudo)	F1 Eye Contact (sudo)	F1 No Eye Contact (sudo)
ETH-XGaze	0.011	0	0	0
L2CSNET	0.047	0	0	0
Multitask (ours)	-0.006	0	0	0
Self-Supervised (ours)	0.081	0	0	0
Supervised (ours)	0.017	0	0	0

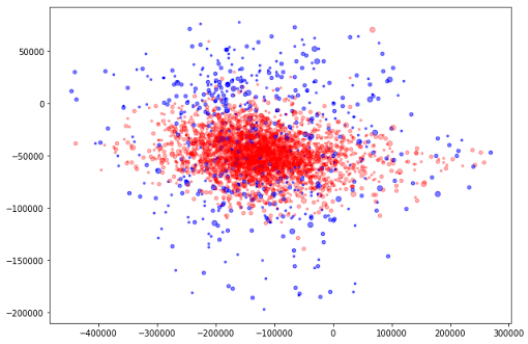
Models	MCC (GT)	Specificity (GT)	F1 Eye Contact (GT)	F1 No Eye Contact (GT)
ETH-XGaze	0.011	0	0	0
L2CSNET	0.047	0	0	0
Multitask (ours)	-0.006	0	0	0
Self-Supervised (ours)	0.081	0	0	0
Supervised (ours)	0.017	0	0	0



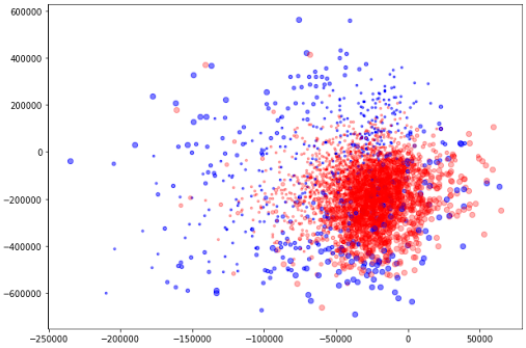
(a) ETH-XGaze



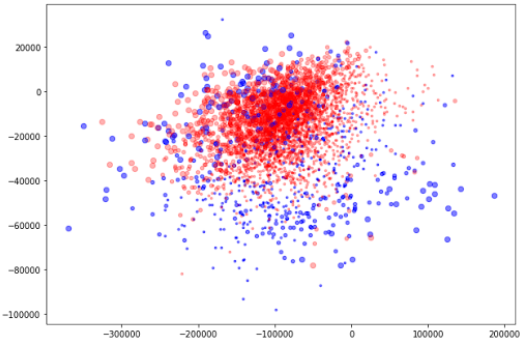
(b) L2CSNET



(c) Multitask

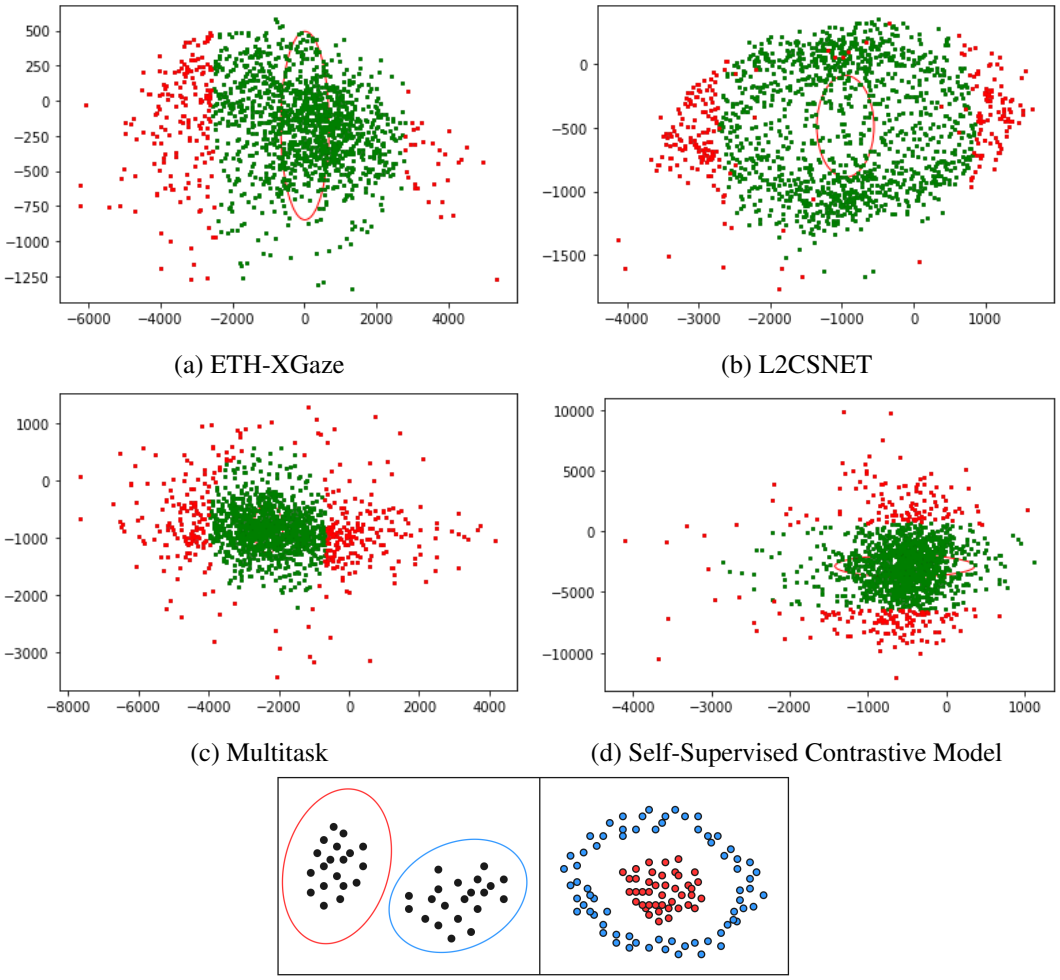


(d) Self-Supervised Contrastive Model



(e) Supervised Contrastive Model

Figure 5.4.: .



(a) ETH-XGaze

(b) L2CSNET

(c) Multitask

(d) Self-Supervised Contrastive Model

(e) Supervised Contrastive Model

Figure 5.5.: .

6. Discussion

test

7. Conclusion

In this section, we conclude our project and discuss the outlook of the project with possible room for future scope.

Conclusion

From the series of experiments carried out and discussed in the previous section. We demonstrated the decomposition of simulated sEMG signals into constituent MU predictions using GRU deep neural network with the labels created by gCKC and the MU activation labels.

Future Scope

sEMG decomposition remains a potential topic of interest. The work of decomposing sEMG can be extended to further improve the robustness of the algorithms.

1. This work can be further explored from the deep learning point of view by investigating more into the model parameter and hyperparameter space of the neural network. Better class weights in the first approach can significantly improve the performance.

A. Acronyms

HMI: Human Machine Interface

HCI: Human Computer Interface

PoG: Point of Gaze

CNN: Convolutional Neural Network

FPN: Feature Pyramid Network

SVM: Support Vector Machine

PnP: Perspective-n-Point

IoU: Intersection Over Union

EMVA: Eveyday Mobile Visual Attention

OPTICS:

DBSCAN: CHECK

NT-XENT: Normalized Temperature Scale Cross Entropy Loss

MCC: Matthews correlation coefficient

List of Figures

List of Tables

Bibliography

[1] S. C. Quax, N. Dijkstra, M. J. van Staveren, S. E. Bosch and M. A. van Gerven, "Eye movements explain decodability during perception and cued attention in meg," *Neuroimage*, vol. 195, pp. 444–453, 2019.

[2] J. H. Goldberg, M. J. Stimson, M. Lewenstein, N. Scott and A. M. Wichansky, "Eye tracking in web search tasks: design implications," in *Proceedings of the 2002 symposium on Eye tracking research & applications*, 2002, pp. 51–58.

[3] K. Wang, H. Su and Q. Ji, "Neuro-inspired eye tracking with eye movement dynamics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9831–9840.

[4] J. D. Smith, R. Vertegaal and C. Sohn, "Viewpointer: lightweight calibration-free eye tracking for ubiquitous handsfree deixis," in *Proceedings of the 18th annual ACM symposium on User interface software and technology*, 2005, pp. 53–61.

[5] B. A. Smith, Q. Yin, S. K. Feiner and S. K. Nayar, "Gaze locking: passive eye contact detection for human-object interaction," in *Proceedings of the 26th annual ACM symposium on User interface software and technology*, 2013, pp. 271–280.

[6] X. Zhang, Y. Sugano and A. Bulling, "Everyday eye contact detection using unsupervised gaze target discovery," in *Proceedings of the 30th annual ACM symposium on user interface software and technology*, 2017, pp. 193–203.

[7] M. Bâce, S. Staal and A. Bulling, "Accurate and robust eye contact detection during everyday mobile device interactions," *arXiv preprint arXiv:1907.11115*, 2019.

[8] X. Zhang, Y. Sugano, M. Fritz and A. Bulling, "Appearance-based gaze estimation in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4511–4520.

[9] Y. Yu and J.-M. Odobez, "Unsupervised representation learning for gaze estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7314–7324.

[10] Z. Guo, Z. Yuan, C. Zhang, W. Chi, Y. Ling and S. Zhang, "Domain adaptation gaze estimation by embedding with prediction consistency," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[11] Y. Cheng, H. Wang, Y. Bao and F. Lu, "Appearance-based gaze estimation with deep learning: A review and benchmark," *arXiv preprint arXiv:2104.12668*, 2021.

[12] S. Baluja and D. Pomerleau, "Non-intrusive gaze tracking using artificial neural networks," *Advances in Neural Information Processing Systems*, vol. 6, 1993.

[13] K.-H. Tan, D. J. Kriegman and N. Ahuja, "Appearance-based eye gaze estimation," in *Sixth IEEE Workshop on Applications of Computer Vision, 2002.(WACV 2002). Proceedings.* IEEE, 2002, pp. 191–195.

[14] S. Park, A. Spurr and O. Hilliges, "Deep pictorial gaze estimation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 721–738.

[15] S. Park, S. D. Mello, P. Molchanov, U. Iqbal, O. Hilliges and J. Kautz, "Few-shot adaptive gaze estimation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9368–9377.

[16] T. Hospedales, A. Antoniou, P. Micaelli and A. Storkey, "Meta-learning in neural networks: A survey," *arXiv preprint arXiv:2004.05439*, 2020.

[17] T. Fischer, H. J. Chang and Y. Demiris, "Rt-gene: Real-time eye gaze estimation in natural environments," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 334–352.

[18] Y. Wu, G. Li, Z. Liu, M. Huang and Y. Wang, "Gaze estimation via modulation-based adaptive network with auxiliary self-learning," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[19] A. A. Abdelrahman, T. Hempel, A. Khalifa and A. Al-Hamadi, "L2cs-net: Fine-grained gaze estimation in unconstrained environments," *arXiv preprint arXiv:2203.03339*, 2022.

[20] D. Dai, W. Wong and Z. Chen, "Rankpose: Learning generalised feature with rank supervision for head pose estimation," *arXiv preprint arXiv:2005.10984*, 2020.

[21] J. Li, J. Wang and F. Ullah, "An end-to-end task-simplified and anchor-guided deep learning framework for image-based head pose estimation," *IEEE Access*, vol. 8, pp. 42 458–42 468, 2020.

[22] B. Doosti, S. Naha, M. Mirbagheri and D. J. Crandall, "Hope-net: A graph-based model for hand-object pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 6608–6617.

[23] V. Lepetit, F. Moreno-Noguer and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.

[24] R. Valle, J. M. Buenaposada and L. Baumela, "Multi-task head pose estimation in-the-wild," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 8, pp. 2874–2881, 2020.

[25] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[26] S. Li, C. Xu and M. Xie, "A robust o (n) solution to the perspective-n-point problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1444–1450, 2012.

[27] M. Khamis, A. Baier, N. Henze, F. Alt and A. Bulling, "Understanding face and eye visibility in front-facing cameras of smartphones used in the wild," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[28] M. Najibi, P. Samangouei, R. Chellappa and L. S. Davis, "Ssh: Single stage headless face detector," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4875–4884.

[29] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang and S. Z. Li, "S3fd: Single shot scale-invariant face detector," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 192–201.

[30] X. Tang, D. K. Du, Z. He and J. Liu, "Pyramidbox: A context-assisted single shot face detector," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 797–813.

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

[32] E. T. Wong, S. Yean, Q. Hu, B. S. Lee, J. Liu and R. Deepu, "Gaze estimation using residual neural network," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2019, pp. 411–414.

[33] Y. Xiong, H. J. Kim and V. Singh, "Mixed effects neural networks (menets) with applications to gaze estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 7743–7752.

[34] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[35] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[36] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," *arXiv preprint arXiv:2009.09796*, 2020.

[37] M. Long, Z. Cao, J. Wang and P. S. Yu, "Learning multiple tasks with multilinear relationship networks," *Advances in neural information processing systems*, vol. 30, 2017.

[38] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi and R. Feris, "Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5334–5343.

[39] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[40] M. Ankerst, M. M. Breunig, H.-P. Kriegel and J. Sander, "Optics: Ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.

[41] H. Bäcklund, A. Hedblom and N. Neijman, "A density-based spatial clustering of application with noise," *Data Mining TNM033*, pp. 11–30, 2011.

[42] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu and D. Krishnan, "Supervised contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 661–18 673, 2020.

[43] S. Becker and G. E. Hinton, "Self-organizing neural network that discovers surfaces in random-dot stereograms," *Nature*, vol. 355, no. 6356, pp. 161–163, 1992.

[44] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger and R. Shah, "Signature verification using a" siamese" time delay neural network," *Advances in neural information processing systems*, vol. 6, 1993.

[45] S. Chopra, R. Hadsell and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 539–546.

[46] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 297–304.

[47] F. Schroff, D. Kalenichenko and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[48] T. Chen, S. Kornblith, M. Norouzi and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[49] F. Wang and H. Liu, "Understanding the behaviour of contrastive loss," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2495–2504.

[50] X. Zhang, Y. Sugano and A. Bulling, "Revisiting data normalization for appearance-based gaze estimation," in *Proceedings of the 2018 ACM symposium on eye tracking research & applications*, 2018, pp. 1–9.

[51] D. Angluin, "Queries and concept learning," *Machine learning*, vol. 2, no. 4, pp. 319–342, 1988.

[52] I. Dagan and S. P. Engelson, "Committee-based sampling for training probabilistic classifiers," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 150–157.

[53] D. D. Lewis, "A sequential algorithm for training text classifiers: Corrigendum and additional data," in *Acm Sigir Forum*, vol. 29, no. 2. ACM New York, NY, USA, 1995, pp. 13–19.

[54] X. Zhang, S. Park, T. Beeler, D. Bradley, S. Tang and O. Hilliges, "Eth-xgaze: A large scale dataset for gaze estimation under extreme head pose and gaze variation," in *European Conference on Computer Vision*. Springer, 2020, pp. 365–381.

[55] M. Bâce, S. Staal and A. Bulling, "Quantification of users' visual attention during everyday mobile device interactions," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–14.

Declaration

Herewith, I declare that I have developed and written the enclosed thesis entirely by myself and that I have not used sources or means except those declared.

This thesis has not been submitted to any other authority to achieve an academic grading and has not been published elsewhere.

Stuttgart, 15/11/2022. Srijay Kolvekar