Michael Pranter, BSc

Improving Robustness and Predictive Performance of Eye-Gaze Estimation via Auxiliary Learning

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl-Ing. Dr.techn. Robert Legenstein

Institute of Theoretical Computer Science
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Robert Legenstein

Graz, September 2023

Abstract

Eyes can reveal a large amount of information about the attention, intention and also health of a human. Therefore, eye gaze estimation is a valuable tool in all areas where aforementioned information is relevant. In this thesis, we developed a framework to train convolutional neural networks which estimate the eye gaze from remote eye images. To train and validate the models, we used computer-generated datasets, which feature a wide variety of head poses, eye gazes and other factors like lighting conditions or facial expressions. Aside from the gaze, the models simultaneously learned a number of other eye-related tasks. Multi-task learning showed to improve eye gaze estimation performance while also introducing new optimization challenges, namely negative transfer. We incorporated techniques to mitigate negative transfer, where experiments disclosed the importance of the balance between tasks. It is vital that loss functions of tasks are compatible with each other in terms of scale throughout the training. Additionally, we experimented with contrastive learning methods. Robustness tests revealed that contrastive learning can be a powerful tool to make models more resistant to corruptions of the input images. Finally, all models were benchmarked on the Comlumbia Gaze and MPIIGaze datasets. These benchmarks showed that there is a notable domain gap between the synthetic and real datasets, although the results are competitive. Improvements observed on the synthetic datasets could generally not be reproduced when testing on real data.

Kurzfassung

Augen können zahlreiche Informationen über Aufmerksamkeit, Absicht und auch Gesundheit eines Menschen preisgeben. Daher ist die Blickschätzung ein wertvolles Werkzeug in allen Bereichen, in denen diese Informationen nützlich sind. In dieser Arbeit haben wir ein Framework entwickelt, mit welchem Convolutional Neural Networks trainiert werden können, die in der Lage sind, den Blick auf Basis von Augenbildern aus der Entfernung zu schätzen. Für das Training und die Validierung der Modelle wurden computergenerierte Datensets verwendet, welche eine hohe Variation an Kopfdrehungen, Blickrichtungen und andere Faktoren wie Belichtungen oder Gesichtsausdrücke beinhalten. Neben der Blickrichtung wurden die Modelle gleichzeitig auch auf andere Aufgaben, welche die Augen betreffen, trainiert. Es hat sich gezeigt, dass daraus Synergien entstehen können, welche die Blickschätzung verbessern. Andererseits können sich die Aufgaben auch gegenseitig behindern, was als negativer Transfer bezeichnet wird. Es wurden mehrere Techniken getestet, um diesen negativen Transfer zu minimieren. Dabei hat sich herausgestellt, dass es äußerst wichtig ist, die Aufgaben richtig zu gewichten. Für die Verlustfunktionen der Aufgaben ist elementar, dass sie untereinander kompatibel sind. Sie sollten über die Dauer des Trainings ähnlich skalieren. Weiters haben wir mehrere Methoden in Bezug auf Contrastive Learning untersucht. Robustheitstests haben ergeben, dass die Modelle durch Contrastive Learning widerstandsfähiger gegenüber beschädigten Inputbildern geworden sind. Abschließend wurden alle Modelle einem Benchmark auf echten Daten unterzogen. Hierfür wurden die Columbia Gaze und MPIIGaze Datensets verwendet. Die Benchmarks haben gezeigt, dass es eine beachtliche Domänenlücke zwischen den synthetischen und echten Daten gibt, obwohl die Ergebnisse konkurrenzfähig sind. Verbesserungen, die auf computergenerierten Daten beobachtet wurden, konnten generell nicht mit den Tests auf echten Daten reproduziert werden.

Contents

List of Figures

List of Tables

List of Acronyms and Symbols

AR Augmented reality

CGaze Columbia Gaze

CL Contrastive learning

CLF Compatible loss functions

CNN Convolutional neural network

DIW Dense image warp

GradNorm Gradient normalization

HCI Human-computer interaction

MAE Mean absolute error

MAngE Mean angular error

MLP Multi-layer perceptron

MPE Mean pixel error

MTL Multi-task learning

NT-Xent Normalized temperature-scaled cross-entropy

PCA Principal component analysis

PCGrad Project Conflicting Gradients

RGB Red, green and blue

SimCLR A Simple Framework for Contrastive Learning of Visual Representations

STL Single-task learning

SupCon Supervised contrastive

TL Transfer learning

VR Virtual reality

1 Introduction

Eye gaze estimation is a task that plays a vital role in many application domains in computer science, including driver monitoring systems [1], patient diagnosis [2] and treatment [3], human-computer interaction (HCI) [4] and augmented / virtual reality (AR / VR) [5]. For most use cases it is crucial to estimate the eye gaze accurately and in real-time in order to assess intentions, cognitive states and attention of humans. Eye gaze estimation is a very challenging task and the performance relies on various different aspects such as the hardware that is used or the lighting conditions. In addition there is a notable variance in the anatomy of the eyes of each human [6, 7].

Convolutional neural networks (CNNs) have proven to be a suitable tool for numerous computer vision tasks in the recent years. Object detection [8], image classification [9], semantic segmentation [10] and many other vision-based problems can be approached by utilizing CNNs. Also eye gaze estimation is a task that CNNs can handle well [11]. Transfer learning (TL) can massively shorten computation cost and simultaneously improve the results of model training [12]. With transfer learning, a model that has seen millions of images can be repurposed to learn a similar or different desired task, while still preserving the capability to understand images very well.

Data synthetization can massively aid in machine learning tasks where the acquisition of real data is demanding. The real-world data might not be there, not easy to gather or it can be very time-consuming to label real-world data correctly. Also privacy may play a role if real people are involved. With synthetic data, all of these problems can be mitigated [13]. On the downside, it is in general not easy to generate data that is indistinguishable from real data. Methods that help to close this domain gap are actively researched [14, 15].

Researchers have also explored the potential of multi-task learning (MTL), which involves training a single model to perform multiple related tasks at the same time. MTL in itself is a challenging research field which continously evolves. Many factors influence the effectiveness of MTL like the choice of the model architecture, loss functions, task affinities etc. This technique has been successfully applied in various forms for problems within and outside of the computer vision scope. [16, 17]

In this thesis, we develop a multi-task learning framework for eye gaze estimation. Various auxiliary tasks are introduced in order to improve the performance of the CNN models by leveraging additional information that could not be learned via the eye gaze alone. The models are trained with eye images that were generated by 3D software. By doing so, it is possible to achieve a high variation of head poses and eye gazes. Furthermore, countless imaginary individuals with randomly sampled skin color, iris color, hair style etc. are featured in the synthetic dataset. This diversity would hardly be attainable when relying on real-world data.

We show that learning auxiliary tasks can indeed be beneficial for eye gaze estimation models. Techniques to reduce negative transfer between tasks were evaluated. Contrastive learning proved to be a good tool to increase model robustness on corrupted images. Furthermore, the models achieved competitive cross-dataset validation results on the Columbia Gaze and MPIIGaze datasets.

The thesis is organized as follows. Chapter 2 provides an overview of related work in the fields of multi-task learning, contrastive learning, data synthetization and also a review of important properties of the anatomy of the human eye. In Chapter 3, all datasets that were used for this work are presented along with the data preprocessing, augmentation and postprocessing pipeline. All relevant information concerning the model architecture is featured in Chapter 4. The specifics of how the models were trained are summarized in Chapter 5. All experiments and their validation results can be found in Chapter 6. The robustness tests and benchmarks on real-world data are featured in Chapter 7. Finally, in Chapter 8, the results are summarized and open questions for future research are discussed.

2 Theory

2.1 Multi-task learning

Multi-task learning is a machine learning process in which multiple tasks are considered at the same time using the same input data. By having more information about the domain at hand, a network can make use of the differing as well as correlating attributes of the various tasks. In the thesis titled "Multitask learning", Caruana suggests that the learning of one task can be improved by factoring in the data of other related tasks during training. He states that multi-task learning can be applied successfully in many different problem domains and in conjunction with many machine learning algorithms [18]. In some more recent work, Romera-Paredes et al. showed that also learning unrelated tasks can have a positive impact on the performance of a task [19].

Using multi-task networks in favor of single-task networks can also be desireable in terms of speed, namely in use-cases where multiple parameters need to be estimated. Deploying a model for each task is too computationally costly, especially if inference is run on embedded systems.

In general, multi-task models are more difficult to train, due to the increased complexity of the input data. The different tasks have to be balanced in order to reach a robust and performant model that generalizes well. Thus, tuning of the architecture and training regime is necessary to obtain optimal results.

2.1.1 Parameter sharing

In a multi-task network, all tasks share parameters in the neural network. The shared parameters can be represented in different ways. In general, we can distinguish between hard parameter sharing and soft parameter sharing. Both types of architectures are illustrated in Figure 2.1.

Hard parameter sharing

With hard parameter sharing, all tasks share the same stem / hidden layers of the network, but each task has its own specific output layers. In 1997, Baxter showed that the more tasks a network learns simultaneously, the less training examples are needed for the shared parameters to achieve good generalization for all tasks. In this context, he also noted that good generalization does not necessarily improve performance of the model [20].

Soft parameter sharing

With soft parameter sharing, each task is solved by a distinct network. Knowlegde is transferred between the hidden layers of the networks. Usually, this is achieved by penalizing the distance of the parameters of one task to the corresponding parameters of other tasks. The amount of parameters a model of this architecture features scales with the number of tasks T.

2.1.2 Tackling negative transfer

As gradients of different tasks point in different directions, there can also be negative transfer while training a multi-task model. Negative transfer describes the negative effect of a training signal of one task to the training of the other task, thus decreasing its performance.

Many different techniques have been researched towards eliminating negative interference. These methods include the measurement of task similarity

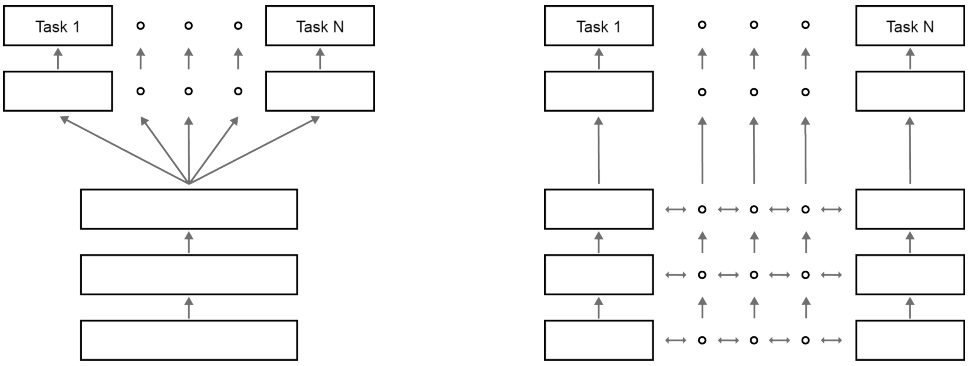

Figure 2.1: Left: multi-task network featuring hard parameter sharing. Right: multi-task network featuring soft parameter sharing.

(a priori or during training [21]) and the direct modification of gradients [22][23]. Some of these techniques will be applied in this work.

Project Conflicting Gradients (PCGrad)

PCGrad is a gradient modification method developed by Yu et al. which aims to reduce negative transfer [22]. The authors identified three major problems concerning MTL: conflicting gradients, dominating gradients and high curvature.

The first issue corresponds to the fact that gradients of different tasks usually point in different directions. This can be measured by calculating the inner product of the gradients, where a negative inner product means that the gradients point away from each other.

Yu et al. argue that this could be mitigated by simply averaging the gradients, but then there remains the concern that gradients with higher magnitude will dominate the average gradient.

Furthermore, if the curvature is high along the directions of the gradients, the positive/negative impact on the learning of tasks may be over- or underestimated.

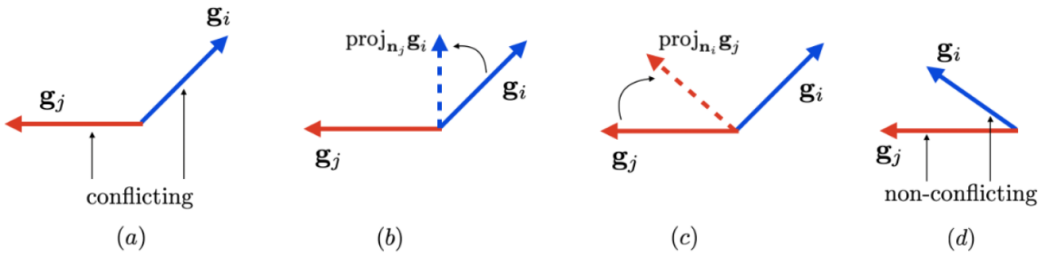

Figure 2.2: Example of gradients of two distinct tasks i and j. (a) Two conflicting gradients. (b) Gradient i projected onto the normal vector of gradient j (c) Same as (b), but vice-versa. (d) Two non-conflicting gradients. The image was taken from the paper in which PCGrad was proposed [22].

The authors showed that they could substantially improve efficiency and performance in various multi-task reinforcement learning scenarios by applying PCGrad. At a high level, the PCGrad algorithm works as follows:

The cosine similarity between gradients g_i and g_j is given by

$$\Phi(g_i, g_j) = \frac{g_i \cdot g_j}{\|g_i\| \|g_j\|} \,. \tag{2.1}$$

If two gradients have negative similarity, i.e.

$$\Phi(g_i, g_j) < 0, \tag{2.2}$$

one will be projected onto the normal vector of the other (see Figure 2.2):

$$g_i = g_i - \frac{g_i \cdot g_j}{\|g_j\|^2} g_j \,. \tag{2.3}$$

The projection of task gradients is executed in random order to avoid that the gradients of only some tasks are projected over and over.

Gradient Normalization (GradNorm)

GradNorm was proposed in "Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks" by Chen et al. [23]. It is a technique

which aims to balance the gradient magnitudes of tasks. In their paper, Chen et al. showed that they could increase accuracy and reduce overfitting by applying GradNorm during training of their multi-task models. They also stated that they observed improvements on both synthetic and real datasets.

Normally, weights for the task losses are chosen once for each task and stay the same for the whole duration of the training. With GradNorm, the weights are learned throughout the training and updated after each step. The algorithm works as follows:

At first, $W \in \mathcal{W}$ has to be chosen, where $\mathcal{W}$ is the set of all weights of the network. The gradient magnitudes which we want to balance are later calculated with respect to W, so it only makes sense to select weights from the shared layers. In this work, W was set to be the weights of the last layer of the stem. This choice was also suggested in the paper.

Let T be the number of tasks. Then, for each training step s and task $t \in \{1, ..., T\}$ the L_2 norm of the gradient of the weighted task loss is calculated with respect to W:

$$G_W^t(s) = \|\nabla_W w_t(s) L_t(s)\|_2 \ . \tag{2.4}$$

Now the average gradient norm of all tasks can be calculated:

$$G_W(s) = \frac{1}{T} \sum_t G_W^t(s) \ . \tag{2.5}$$

Furthermore, some training rates need to be defined for task t:

$$i_t(s) = \frac{L_t(s)}{L_t(0)}, \text{ the inverse training rate of t,} \tag{2.6}$$

$$r_t(s) = \frac{i_t(s)}{\frac{1}{T}\sum_t i_t(s)}, \text{ the relative inverse training rate of t.} \tag{2.7}$$

The loss function that is differentiated to update task weight $w_t(s)$ is defined as

$$L_{\text{GradNorm}}(s; w_t(s)) = \sum_t \left| G_W^t(s) - G_W(s) \times [r_t(s)]^\alpha \right|_1 \ . \tag{2.8}$$

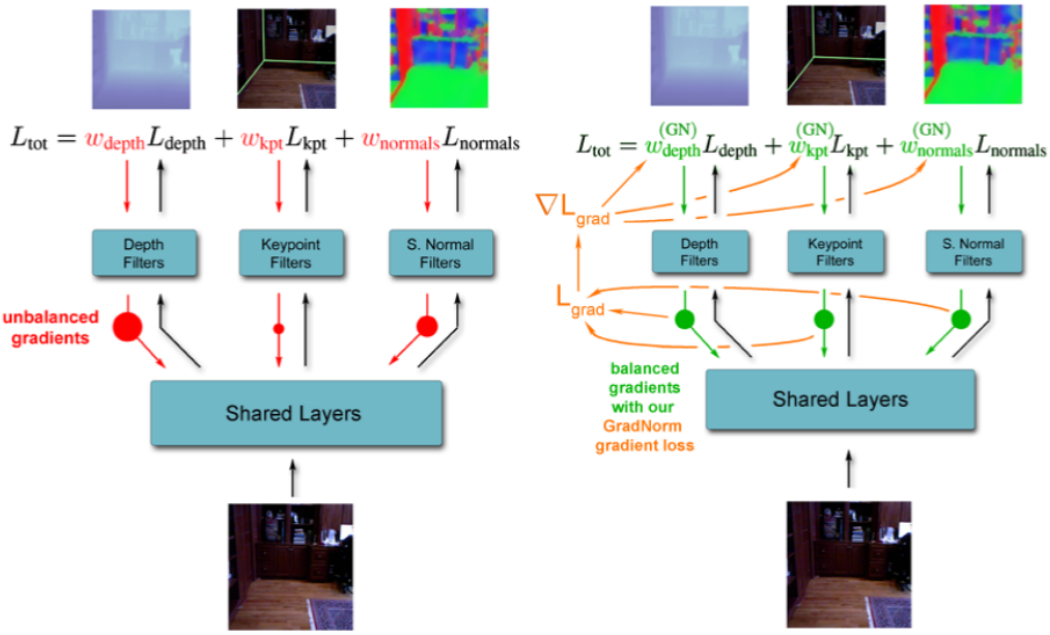

Figure 2.3: A comparison of normal multi-task learning with fixed task weights (left) and GradNorm enhanced learning with adaptive task weigths (right). The image was taken from the paper in which GradNorm was proposed [23].

The gradients $\nabla w_t L_{\text{GradNorm}}$ are then subtracted from the task weights using a learning rate, following the standard gradient descent procedure. After every weight update step, the weights are also renormalized such that the sum of the weights equals the number of tasks:

$$\sum_t w_t(s) = T \, . \tag{2.9}$$

This is done in order to minimize the impact of GradNorm on the global learning rate.

2.2 Contrastive learning

Contrastive learning is a means to teach a network which input data is similar and which is not. The goal is to achieve similar encodings in latent space for objects that are related.

While empirical data shows that contrastive learning is a very promising technique, it is still an open research question how much it helps models to generalize better and what factors influence this generalization ability. Huang et al. identified three key factors that play a role in the performance: how the positve samples are aligned, the divergence of the class centers (cosine distance) and how the augmented data is concentrated (a measure of how well data is augmented) [24].

The first two factors are properties of a trained model, while the last one is determined by the augmentation pipeline. Assume a contrastive learning method which pulls together various augmentations of the same sample. If data is not augmented at all, then the perfect alignment of positive samples does not have much value since the samples are identical. This shows the importance of data augmentation in the assessment of contrastive learning methods.

There are multiple contrastive learning methods. In this work, three of them were utilized for the experiments: a simple contrastive loss, SimCLR [25] and SupCon [26]. For more information on these techniques see Section 5.1.

2.3 Eye gaze estimation

2.3.1 Problem definition

The purpose of eye gaze estimation is to retrieve the target or direction of an eye's gaze by analyzing e.g. a picture of the eye. The gaze can either be estimated in 2D (for example as a target coordinate on a screen) or in 3D

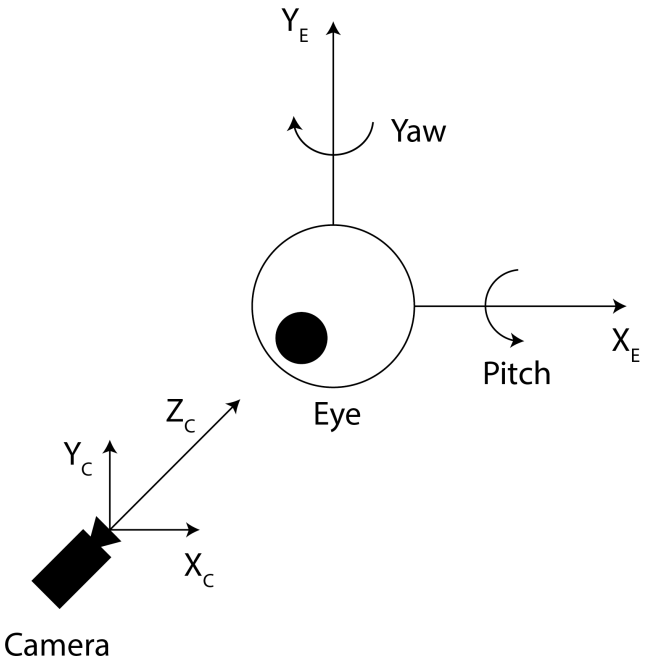

Figure 2.4: Visualization of the eye in relation to the camera. The rotation around the X axis of the eye denotes the pitch and the rotation around the Y axis the yaw of the eye. For the training and validation set, the eye ball center is located on the forward axis of the camera ($X_C = 0$, $Y_C = 0$). For the real-world datasets, this is usually not the case. That means that the eyes of the latter are not necessarily centered in the image.

(as a vector, a coordinate or as pitch and yaw angles, for example). In this thesis, the pitch and yaw angles of the gaze will be estimated. An overview of the setup featuring the camera and the eye can be seen in Figure 2.4.

2.3.2 Optical and visual gaze

Due to the anatomy of the human eye, there is not only one, but two directions which should be considered: the optical axis, which passes from the back of the eyeball through the center of the pupil and the visual axis, which passes from the center of the fovea through the center of the curvature of

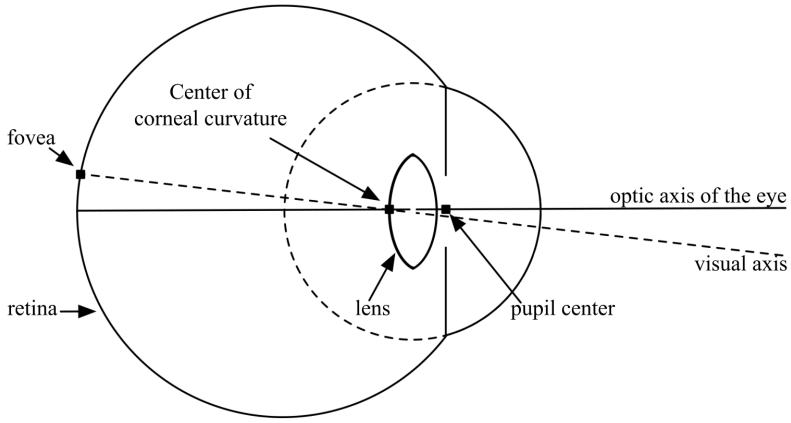

Figure 2.5: Standard geometric model of the human eye by Wu et al. [28].

the cornea.

The optical axis represents the orientation of the eyeball. The visual axis is the actual direction a person focuses on when gazing at a point. The exact angles between visual and optical axes vary from person to person and between left and right eye. On average, the horizontal angle is around 5° for the left and $-5°$ for the right eye, and the vertical angle around 1.5° for both eyes [27].

To estimate a more accurate angular offset of the visual axes of a person and to thereby reduce the base error introduced by these personal anatomic differences, user calibration or similar approaches can be performed before the gaze estimation. In this thesis, no such method was applied. The dataset which was used for training the network only includes the optical gaze. The average angle offsets that are stated above were added to the optical gaze estimation in order to obtain a visual gaze estimation for the benchmarks on real-world datasets (see Section 7.1).

2.3.3 Remote and near-eye images

Eye gaze estimation can be based on remote or near-eye images. This thesis focuses on remote eye images as there are far more use cases due to the fact that images can be taken with smartphones or webcams. Otherwise, special purpose hardware (e.g a head-mounted camera) would be required in near-eye scenarios.

When using images of objects further away from the camera (i.e. remote), some additional factors have to be taken into account when synthethising the data and training the network, such as lighting conditions, camera perspective or other facial features like the nose. The head pose, for example, can be included in training, which can improve eye gaze estimation. That images of the eyes alone do not include all information that is useful for eye gaze estimation was shown by Zhang et al. They could improve gaze estimation performance by using the full face image as input for their CNN [29]. Also for humans it is harder to estimate the gaze of a person when features around the eye are omitted [30].

2.4 Data synthetization

Data synthetization is the process in which a dataset is generated by a program. In general, it makes sense to train a model with generated data, because the labelling of the training samples can be automated and determined very accurately. Especially in the case of eye gaze estimation, where it is not a trivial task to set the correct labels manually, it makes sense to utilize synthesized data.

In previous work, researchers successfully trained convolutional neural networks with simulated data in order to estimate the gaze vector of input images.

However, there are drawbacks when using synthesized images. First and foremost, the underlying domain gap to real world images poses a challenge. It is rather non-trivial to generate perfectly photorealistic images of

eye regions. Lighting, shadows, refractions, hair, skin, textures, wrinkles, etc. have to be simulated and tweaked for a realistic result. Additionally, rendering performance in terms of speed has to be kept in mind as well. The more accurate rendering approaches are usually also more resource-intensive. This makes techniques like ray-tracing impracticable to use, as it may take days or months to render a dataset with hundreds of thousands of images.

2.4.1 UnityEyes

In some of the most relevant work, UnityEyes was used to generate photorealistic remote eye images along with labels [31]. Wood et al. trained a network with the generated data and tested it on real images given by the MPIIGaze dataset, resulting in an average angular error of 9.95°.

This result can likely be improved by increasing the variation of the generated input data, e.g. by reproducing more scenarios which occur in real images. In "Learning an appearance-based gaze estimator from one million synthesised images" [31], it is mentioned that the most obvious false predictions can be directly related to missing cases which are not available in the training data. Wood et al. used a nearest-neighbor approach to estimate the gaze. Some of the worst performing predictions were caused by examples of people wearing makeup/eyeshadow. Features like makeup or eyelashes result in high contrast in various areas around the eye, which can confuse the estimator.

3 Datasets and data pipeline

All datasets feature left eye images. If right eye images existed in a benchmark dataset, they were mirrored to appear like left eye images.

3.1 Vertify dataset

This dataset was provided by Vertify GmbH and features synthesized eye images. The scenes were generated using multiple head scans and offer great variation in lighting conditions, camera field of view and distance, facial expressions, skin and eye textures, hair and other details.

For each image, the eye was positioned at the center to avoid camera distortion as much as possible. The images were generated with uniformly sampled head pose / eye gaze configurations. For each configuration there are 8 samples with the same head pose and gaze vector. These 8 samples vary in lighting conditions and head texture / geometry. This means that for each of the samples, a new character was generated in an individual scene while keeping the ground truth of all samples the same. For 4 of the 8 samples, the facial expressions are the same. These samples are referred to as the simple variations. For the other half, also the facial expressions were resampled, altering the ground truth of keypoints. These are referred to as the complex variations.

This dataset enables a range of experiments to utilize contrastive learning. Having multiple different scenes with the same ground truth makes it easier to teach the network which images should be seen as similar or different.

Figure 3.1: Example images of the training set. Each row represents one head pose / eye gaze configuration with 4 complex and 4 simple variations

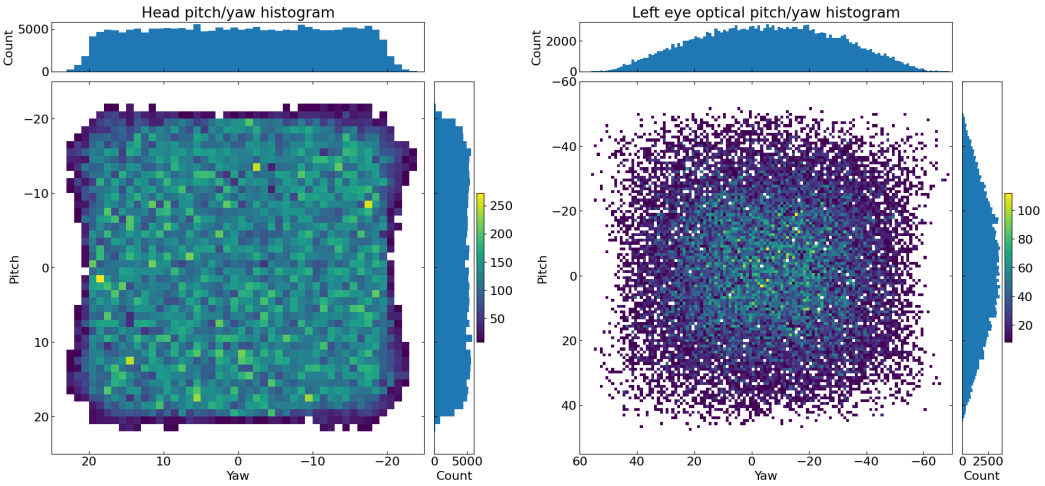

Figure 3.2: Distribution of the training set. Left: head pose. Right: optical gaze. Since the visual gaze is sampled with a mean of zero during synthetization, the optical gaze is shifted.

3.1.1 Training set

The training set features 25,408 head pose / eye gaze configurations, resulting in 203,264 synthesized eye images. Examples are shown in Figure 3.1.

3.1.2 Validation set

The validation set features 1,360 head pose / eye gaze configurations, resulting in 10,880 synthesized eye images. The validation set was generated using exclusive head scans that were not used for the training set.

3.1.3 Corruption set

The corruption dataset comprises augmented versions of 1,000 random samples of the validation set. This set helps to measure the robustness of the trained models. The following corruptions are available and can be

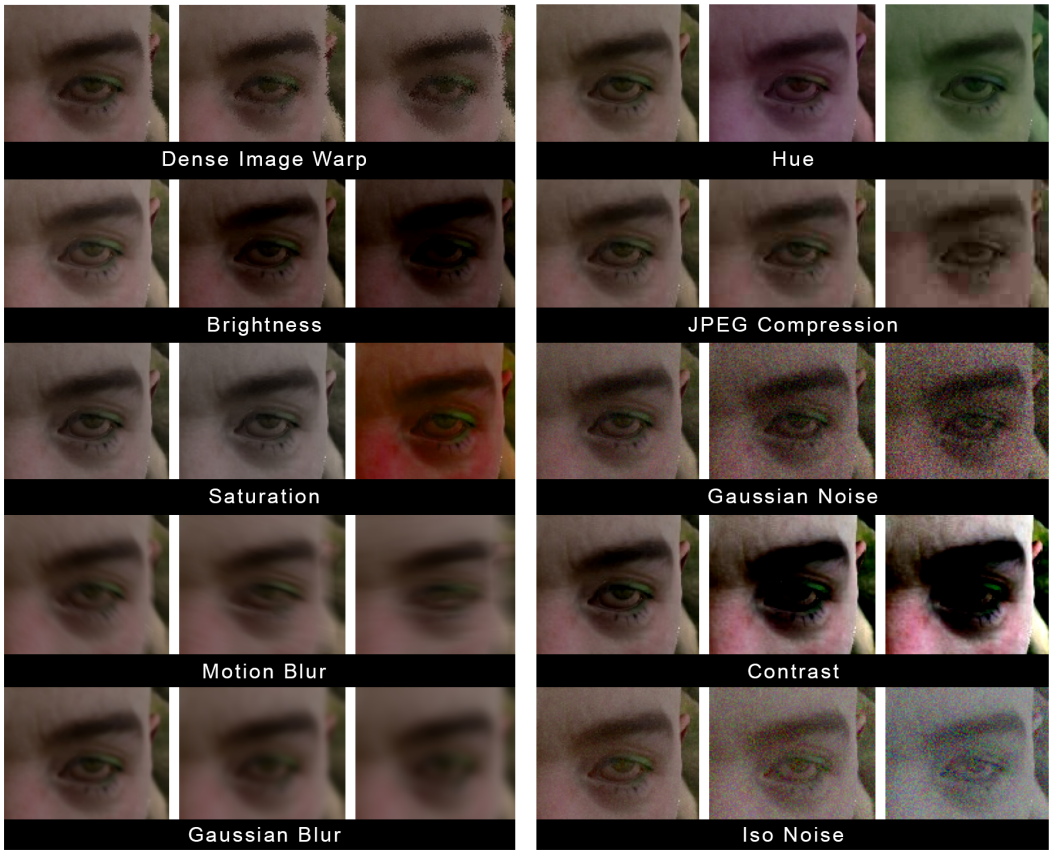

Figure 3.3: Example images of the corruption sets. For each corruption, there are 3 levels of severity.

seen in Figure 3.3: dense image warp, brightness, saturation, motion blur, Gaussian blur, hue, JPEG compression, Gaussian noise, contrast, iso noise. For each corruption, there are 3 levels of severity. This results in a total of 30 corruption subsets.

3.2 Benchmark datasets

3.2.1 Columbia Gaze

The Columbia Gaze dataset was released in 2013. It was produced by Smith et al. at Columbia University and features 5,880 labelled eye images of 56 different participants [32]. 21 of the subjects wore glasses, 24 were female and 32 were male.

The images were taken in a controlled environment. For each participant, there are 105 configurations of head poses and gaze directions:
Vertical head pose: 0°
Horizontal head poses: -30°, -15°, 0°, 15°, 30°
Vertical gaze directions: -10°, 0°, 10°
Horizontal gaze directions: -15°, -10°, -5°, 0°, 5°, 10°, 15°.

Example images of the dataset are shown in Figure 3.4. The distribution of head poses and eye gazes are shown in Figure 3.5.

3.2.2 MPIIGaze

The MPIIGaze dataset was released in 2015. It was produced by Zhang, Sugano et al. at Max Planck Institute and features 213,659 labelled eye images of 15 different participants [33]. 5 of the subjects wore glasses, 6 were female and 9 were male. The labels include the visual gaze vectors and head poses, among others.

All images of the dataset were taken with the participants in front of a laptop, where the built-in webcam captured the images. Zhang et al. chose this setting, because gaze estimation applications often rely on images taken from webcams or smartphone cameras. The nature of this data acquisition process implies that the range of eye gazes and head poses in this dataset is limited. No large eye movements are needed to look at every point on the screen and participants usually keep their head frontal to the camera.

Figure 3.4: Example images of the benchmark datasets. Left: Columbia Gaze. Right: MPI-IGaze.

Example images of the dataset are shown in Figure 3.4. The distribution of head poses and eye gazes are shown in Figure 3.6.

3.3 Data pipeline

The training data was preprocessed before each training step in the following order of events:

At first, the samples got shuffled to achieve a random composition of batches. Then the images were cropped such that the eye is in the center of the image, with a fixed-size padding to the left and right side of the eye. The padding on each side was set to 0.7 times the eye width (corner to corner).

After that, a random zoom was performed. The zoom factor was sampled randomly for each image between 1.0 and 1.5 and the resulting zoom box was offset randomly within the image.

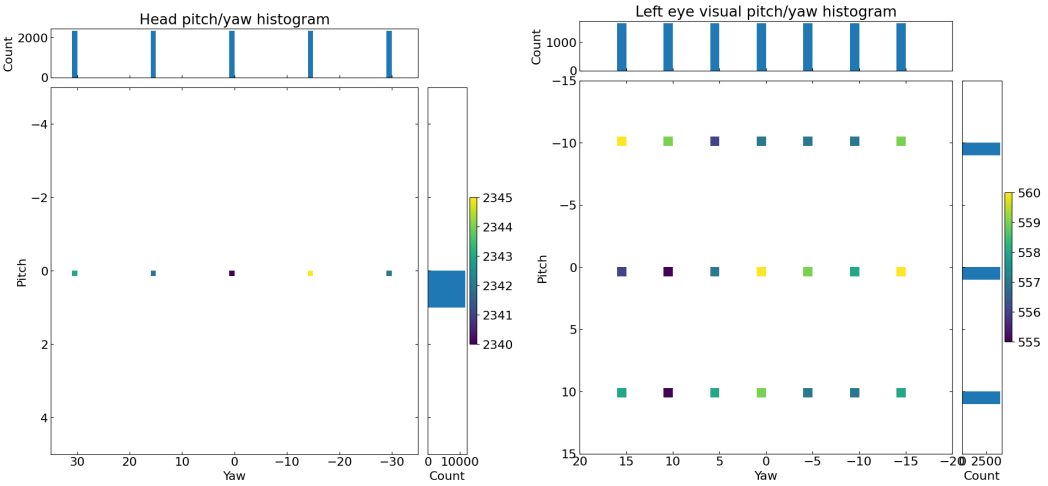

Figure 3.5: Distribution of the Columbia Gaze set. Left: head pose. Right: visual gaze.

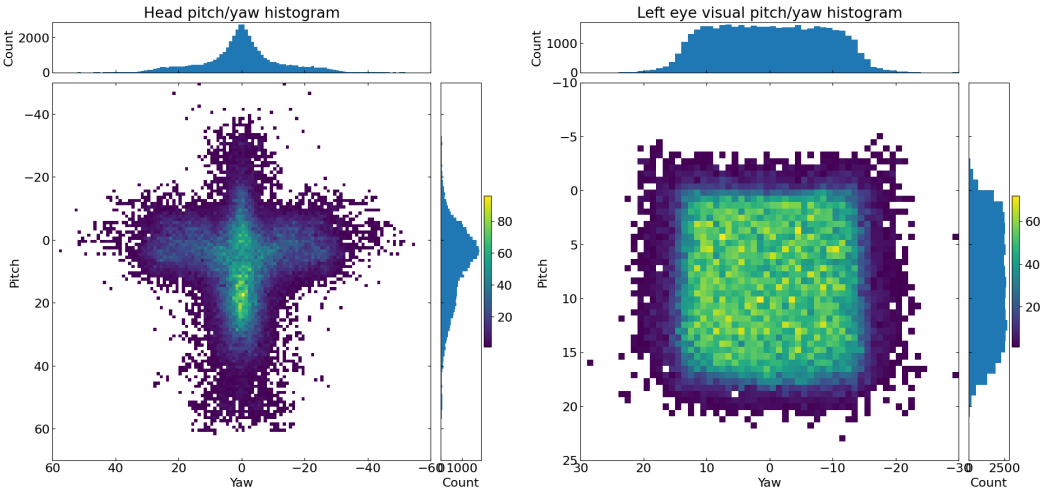

Figure 3.6: Distribution of the MPIIGaze set. Left: head pose. Right: visual gaze.

Afterwards, a color jitter was applied which changed the saturation (random between 0.5 and 1.5), brightness (random delta between -0.125 and 0.275), contrast (random between 0.5 and 1.5) and hue (random delta between -0.04 and 0.04) of the images[1].

Furthermore, with a chance of 50% (about every second image), Gaussian noise was added. The standard deviation σ was sampled between 0.01 and 0.1 for each of the candidate images and the mean was set to $\mu = 0$. Then, for each color value (96 x 96 x 3 values) a randomly sampled value from the Gaussian distribution was added.

After augmentation, the images and their corresponding keypoint labels were normalized to float values between a minimum of -1 and a maximum of 1. The gaze pitch and yaw angles were not normalized, since they were stored as radians and ranged from about -1.047 to 1.047 (-60° to 60°) in the training set. Also the iris visibilities ranged from 0 to 1 and were not normalized.

The training set was filtered before training as follows: for training the gaze and iris keypoint tasks, batches were filtered by iris visibility. Only samples with an iris visibility ≥ 0.1 were utilized for training. For lid keypoints, the mean lid keypoint visibility of samples had to be ≥ 0.5.

[1]The utilized image augmentation functions are shipped with TensorFlow [34]: tf.image.adjust_saturation, tf.image.adjust_brightness, tf.image.adjust_contrast and tf.image.adjust_hue.

4 Model architecture

An overview of the model architecture that was used for the experiments can be seen in Figure 4.1. Each model receives RGB images with a resolution of 96 x 96 pixels as input. Then, the stem of the model extracts features from the image and passes those features to the various heads. The heads then transform the features into estimates, depending on their task.

The models utilize batch normalization and for all activations, the Rectified Linear Unit (ReLU) [35] function was chosen.

4.1 Stem

All models contain a stem which acts as a feature extractor. For this work, a ResNet50 [36] featuring the adaptions suggested in [37] was utilized. The two last layers, namely the global average pooling layer and the dense layer, were removed. Hence the output of the last Residual Unit (see details below) of the ResNet was used as the stem output.

The main building block of a ResNet, namely the Residual Unit, is recapped in Figure 4.2. The architecture of the ResNet50 stem is shown in Table 4.1. The stem was pre-trained with the ImageNet dataset [38]. The pre-trained model is shipped with Tensorflow [34] / Keras. [1]

[1] The model was imported from: keras.applications.resnet_v2.ResNet50V2

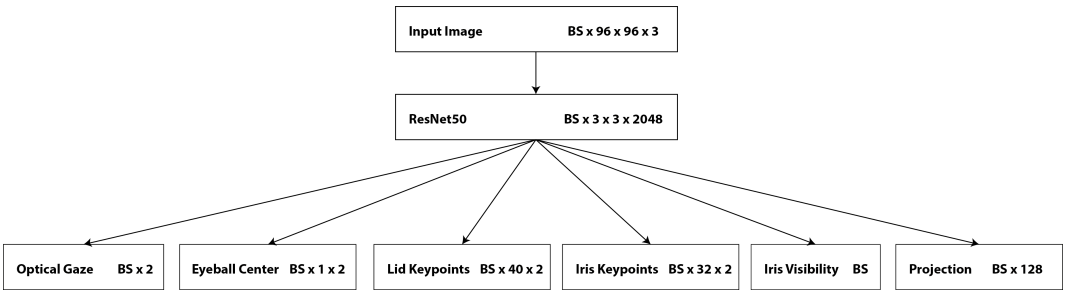

Figure 4.1: An overview of the multi-task model architecture. All tasks share the same output of the stem (hard parameter sharing). For each block, the name is shown along with the output shape of the block. BS is the batch size.

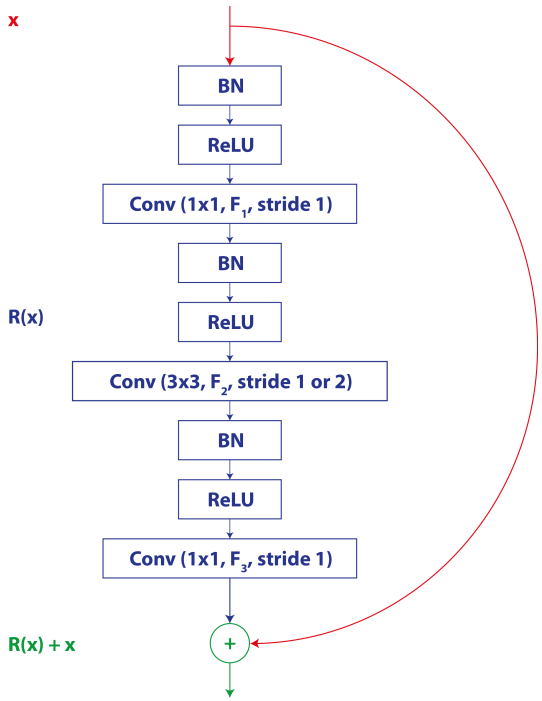

Figure 4.2: A Bottleneck Residual Unit as defined by He et al. [36, 37]. The residual $R(x)$ is calculated on one path, and the identity is passed as a skip-connection on the other path. The output is given by the sum of the residual and the input. F_1, F_2 and F_3 denote the number of filters of the convolutional layers. The 3x3 convolution may downsample (i.e. when stride is 2).

Layer name	Output size	Layer info
conv1	48 x 48 x 64	7 x 7, 64, stride 2
pool1	24 x 24 x 64	3 x 3 max pool, stride 2
conv2_x	12 x 12 x 256	$\begin{bmatrix} 1\,x\,1, & 64 \\ 3\,x\,3, & 64 \\ 1\,x\,1, & 256 \end{bmatrix}$ x 3
conv3_x	6 x 6 x 512	$\begin{bmatrix} 1\,x\,1, & 128 \\ 3\,x\,3, & 128 \\ 1\,x\,1, & 512 \end{bmatrix}$ x 4
conv4_x	3 x 3 x 1024	$\begin{bmatrix} 1\,x\,1, & 256 \\ 3\,x\,3, & 256 \\ 1\,x\,1, & 1024 \end{bmatrix}$ x 6
conv5_x	3 x 3 x 2048	$\begin{bmatrix} 1\,x\,1, & 512 \\ 3\,x\,3, & 512 \\ 1\,x\,1, & 2048 \end{bmatrix}$ x 3

Table 4.1: ResNet50 architecture for inputs with shape 96 x 96 x 3. The last layers are omitted since we did not use them. The properties for the Bottleneck Residual Units are shown in brackets along with the number of repetitions of each unit. In each of the 4 unit repetitions, the last 3x3 convolutional layer downsamples with a stride of 2.

4.2 Task heads

The stem output is fed into the various heads. For each task, one corresponding head is appended to the stem. The heads deliver the outputs which are the estimations for each task. Table 4.2 shows the architecture of the gaze head, Table 4.3 features information on the keypoint task heads and Table 4.4 shows the architecture of the iris visibility head, respectively.

Layer name	Output size	Layer info
gaze_conv1	3 x 3 x 256	1 x 1, 256, stride 1
gaze_conv2_x	2 x 2 x 1024	$\begin{bmatrix} 1\,x\,1, & 256 \\ 3\,x\,3, & 256 \\ 1\,x\,1, & 1024 \end{bmatrix}$ x 2
gaze_conv3	1 x 1 x 256	1 x 1, 256, stride 2
gaze_conv4	1 x 1 x 2	1 x 1, 256, stride 1
gaze_reshape	2	

Table 4.2: Architecture of the head for the optical gaze task. The properties for the Bottleneck Residual Units are shown in brackets along with the number of repetitions of each unit.

Layer name	Output size	Layer info
keypoint_conv1	3 x 3 x 128	1 x 1, 128, stride 1
keypoint_conv2_x	3 x 3 x 512	$\begin{bmatrix} 1\,x\,1, & 128 \\ 3\,x\,3, & 128 \\ 1\,x\,1, & 512 \end{bmatrix}$ x 2
keypoint_conv3	2 x 2 x 128	1 x 1, 128, stride 2
keypoint_conv4	2 x 2 x 32	1 x 1, 32, stride 1
keypoint_conv5	1 x 1 x $2K_t$	2 x 2, $2K_t$, stride 1
keypoint_reshape	K_t x 2	

Table 4.3: Architecture of the head for the keypoint tasks. K_t is the number of keypoints for task $t \in \{1, ..., T\}$, where T is the number of keypoint tasks. The properties for the Bottleneck Residual Units are shown in brackets along with the number of repetitions of each unit.

4.3 Projection head

For contrastive learning (see Section 2.2), a projection head is appended to the model which acts like another task. Zhang et al. used this projection head to map the stem output to a representation to which their SimCLR loss is applied on [25]. The head is a multi-layer perceptron (MLP) with

Layer name	Output size	Layer info
visibility_conv1	3 x 3 x 128	1 x 1, 128, stride 1
visibility_conv2_x	3 x 3 x 512	$\begin{bmatrix} 1\,x\,1, & 128 \\ 3\,x\,3, & 128 \\ 1\,x\,1, & 512 \end{bmatrix}$ x 2
visibility_conv3	2 x 2 x 128	1 x 1, 128, stride 2
visibility_avgpool	1 x 1 x 128	
visibility_conv4	1 x 1 x 1	1 x 1, 1, stride 1
visibility_sigmoid	1 x 1 x 1	
visibility_reshape	1	

Table 4.4: Architecture of the head for the iris visibility task. The properties for the Bottleneck Residual Units are shown in brackets along with the number of repetitions of each unit.

one hidden layer and a ReLU activation (see Table 4.5). The authors showed that applying the loss to a nonlinear projection head improves performance greatly compared to directly calculating the contrastive loss from the stem output.

Layer name	Output size
projection_global_avgpool	2048
projection_fc1	512
projection_fc2	128

Table 4.5: Architecture of the projection head.

5 Training specification

5.1 Loss functions

The different types of tasks require different loss functions. In the following equations we denote B as the batch size and T as the number of tasks.

Iris visibility

Let $x_1, ..., x_B$ be a batch of estimated iris visibilities and $y_1, ..., y_B$ be the labels for that batch. Then the mean absolute error loss is given by

$$L_{\text{MAE}} = \frac{1}{B} \sum_{i=1}^{B} |x_i - y_i| \ . \tag{5.1}$$

Gaze

Let $\boldsymbol{g} = \begin{pmatrix} g_{\text{pitch}} \\ g_{\text{yaw}} \end{pmatrix}$ be a gaze in pitch / yaw representation. Then the corresponding gaze in direction vector representation is

$$\text{vec}(\boldsymbol{g}) = \begin{pmatrix} \cos(g_{\text{pitch}}) \sin(g_{\text{yaw}}) \\ -\sin(g_{\text{pitch}}) \\ \cos(g_{\text{pitch}}) \cos(g_{\text{yaw}}) \end{pmatrix} . \tag{5.2}$$

The angle between two gazes $\boldsymbol{g}$ and $\boldsymbol{h}$ is

$$\text{angle}(\boldsymbol{g}, \boldsymbol{h}) = \arccos \left(\frac{\sum_{i=1}^{3} \text{vec}(\boldsymbol{g})_i \, \text{vec}(\boldsymbol{h})_i}{\|\text{vec}(\boldsymbol{g})\|_2 \, \|\text{vec}(\boldsymbol{h})\|_2} \right) . \tag{5.3}$$

Let $\boldsymbol{x}_1, ..., \boldsymbol{x}_B$ be a batch of estimated gazes in pitch / yaw representation and $\boldsymbol{y}_1, ..., \boldsymbol{y}_B$ be the labels for that batch. Then the mean angular error loss is

$$L_{\text{MAngE}} = \frac{1}{B} \sum_{i=1}^{B} \text{angle}(\boldsymbol{x}_i, \boldsymbol{y}_i) \ . \tag{5.4}$$

Keypoints

Let K_t be the number of keypoints of task t, $k \in \{1, ..., K_t\}$ and $i \in \{1, ..., B\}$. Let $\boldsymbol{p}_k = \begin{pmatrix} p_{k_1} \\ p_{k_2} \end{pmatrix}$ be a keypoint and $\boldsymbol{x}_i = \begin{pmatrix} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_{K_t} \end{pmatrix}$ be all keypoints of task t in sample i. Then $\boldsymbol{x}_1, ..., \boldsymbol{x}_B$ is a batch of estimated keypoints and $\boldsymbol{y}_1, ..., \boldsymbol{y}_B$ are the labels for that batch. Then the mean squared error loss is given by

$$L_{\text{MSE}} = \frac{1}{B} \sum_{i=1}^{B} \left(\frac{1}{K_t} \sum_{k=1}^{K_t} (\boldsymbol{x}_{i_k} - \boldsymbol{y}_{i_k})^2 \right) . \tag{5.5}$$

The Euclidean distance between two keypoints $\boldsymbol{p}$ and $\boldsymbol{q}$ is defined as

$$\text{dist}(\boldsymbol{p}, \boldsymbol{q}) = \sqrt[2]{(\boldsymbol{p}_1 - \boldsymbol{q}_1)^2 + (\boldsymbol{p}_2 - \boldsymbol{q}_2)^2} \tag{5.6}$$

and the mean Euclidean distance loss is then

$$L_{\text{MED}} = \frac{1}{B} \sum_{i=1}^{B} \left(\frac{1}{K_t} \sum_{k=1}^{K_t} \text{dist}(\boldsymbol{x}_{i_k}, \boldsymbol{y}_{i_k}) \right) . \tag{5.7}$$

Note that for evaluation we use the mean pixel error (MPE) as a metric. The MPE is the mean Euclidean distance between estimated keypoints and ground truth keypoints as well, but on the scale of the input images[1]. This makes the values more intuitive to interpret.

[1] L_{MED} is calculated on the normalized keypoints which yields very small numbers

Contrastive losses

One of the earliest contrastive losses was presented by Hadsell et al. in 2006 [39]. The aim of the loss is to learn embeddings which have a small Euclidean distance for similar samples and a high distance for samles that are very different. The loss is used for classification tasks, so in order to utilize it we formed triplets in each batch. Every triplet features one base sample (anchor), one similar (positive) and one dissimilar (negative) sample.

Let $B \bmod 3 = 0$ and $i \in \{1, ..., \frac{B}{3}\}$. Then we can divide the batch into 3 equally sized chunks of samples. We then compare each sample x_i in the first chunk with the corresponding sample y_i of the second chunk and also the corresponding sample z_i of the third chunk:

$$x_i^{\text{anchor}} = x_i, \tag{5.8}$$

$$x_i^{\text{pos}} = \begin{cases} y_i & \text{if angle}(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i) < \text{angle}(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{z}}_i) \\ z_i & \text{else} \end{cases}, \tag{5.9}$$

$$x_i^{\text{neg}} = \begin{cases} y_i & \text{if angle}(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i) \geq \text{angle}(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{z}}_i) \\ z_i & \text{else} \end{cases}, \tag{5.10}$$

where $\hat{\boldsymbol{x}}_i$, $\hat{\boldsymbol{y}}_i$ and $\hat{\boldsymbol{z}}_i$ denote the gaze labels of the samples x_i, y_i and z_i.

Let $f(\cdot)$ be the output of the projection head. Then the distance between features given by the projection head can be calculated as follows:

$$d_i^{\text{pos}} = \|f(x_i^{\text{anchor}}) - f(x_i^{\text{pos}})\|_2, \tag{5.11}$$

$$d_i^{\text{neg}} = \|f(x_i^{\text{anchor}}) - f(x_i^{\text{neg}})\|_2 \,. \tag{5.12}$$

The contrastive loss is then given by

$$L_{\text{Contrastive}} = \frac{3}{B} \sum_i \left((d_i^{\text{pos}})^2 + \max(m - d_i^{\text{neg}}, 0)^2 \right), \tag{5.13}$$

where m denotes the margin. This hyperparameter specifies how much positive and negative samples get separated.

For SimCLR [25], all samples are augmented a second time. This gives a positive pair (denoted by indices i and j) for each sample in the batch. In their work, Chen et al. used the so-called normalized temperature-scaled cross-entropy loss (NT-Xent), which was previously introduced by Sohn et al. [40].

The cosine similarity is defined as

$$\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|} \,. \tag{5.14}$$

Then the NT-Xent loss is given by

$$L_{\text{NT-Xent}}(i, j) = -\log \frac{\exp(\text{sim}(f(x_i), f(x_j)) \,/\, \tau)}{\sum_{k=1}^{2B} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(f(x_i), f(x_k)) \,/\, \tau)} \,, \tag{5.15}$$

where $\mathbb{1}_{[k \neq i]} = \begin{cases} 1 & \text{if } k \neq i \\ 0 & \text{else} \end{cases}$ and τ is the temperature hyperparameter.

If we append the second augmentations of all samples to the batch, the SimCLR loss is given by

$$L_{\text{SimCLR}} = \frac{1}{2B} \sum_{i=1}^{B} \left(L_{\text{NT-Xent}}(i, B+i) + L_{\text{NT-Xent}}(B+i, i) \right) \,. \tag{5.16}$$

Supervised contrastive learning (SupCon) can be described as the fully supervised version of the self-supervised SimCLR. With SupCon, one can not only feed the loss with augmented versions (views) of one sample, but also with other multi-augmented samples that all share the same label [26].

Let A be the number of augmentations per sample. Then the SupCon loss is defined as follows:

$$L_{\text{SupCon}_i} = \frac{1}{A \cdot B_{\hat{y}_i}} \sum_{j=1}^{A \cdot B} \left(\mathbb{1}_{[i \neq j]} \cdot \mathbb{1}_{[\hat{y}_i = \hat{y}_j]} \cdot L_{\text{NT-Xent}}(i, j) \right) , \tag{5.17}$$

$$L_{\text{SupCon}} = \sum_{i=1}^{A \cdot B} L_{\text{SupCon}_i} \,. \tag{5.18}$$

Total loss

The task losses can be weighted in order to equalize the scale of the losses. Let w_t be the task weight and L_t be the unweighted loss function for task $t \in \{1, ..., T\}$.
Then the total loss that is minimized is the sum of all weighted task losses:

$$L_{\text{Total}} = \sum_t w_t L_t \,. \tag{5.19}$$

5.2 Optimizer

To minimize the loss, the Adam optimizer [41] was used. The implementation that was used is the one shipped with Keras[2]. All parameters except for the learning rate were left unchanged. The default parameters are:

$$\begin{gathered} \beta_1 = 0.9, \\ \beta_2 = 0.999, \\ \epsilon = 10^{-7}, \\ \text{amsgrad} = \text{False}. \end{gathered}$$

5.3 Training stages

The training is divided into three stages. The learning rate η_t is interpolated for step $t \in \{t_{\text{start}}, ..., t_{\text{end}}\}$ as follows:

$$\alpha_t = \frac{t - t_{\text{start}}}{t_{\text{end}} - t_{\text{start}}}, \tag{5.20}$$

[2]The optimizer was imported from: keras.optimizers.Adam

$$\eta_t = (1 - \alpha_t) \cdot \eta_{t_{\text{start}}} + \alpha_t \cdot \eta_{t_{\text{end}}} \ . \tag{5.21}$$

5.3.1 First stage

In the first stage, the weights of the pre-trained stem are frozen. This should help the heads to adapt to the pre-trained features. Furthermore there is a warmup phase for the learning rate. The phase reduces numerical instabilities which can occur at the beginning of the training. The warmup period comprises the first 5000 steps ($\approx$ 3 epochs; $t_{\text{start}} = 1$ and $t_{\text{end}} = 5000$) of the stage in which the learning rate is slowly scaled up from a very low value to the target learning rate of 10^{-4}. When the target learning rate is reached, it is kept at that level for the rest of the stage. The first training stage lasts for 10 epochs.

5.3.2 Second stage

In the second stage, the pre-trained stem weights are unfrozen and trained as well. The learning rate slowly decays linearly for the whole duration of the stage (t_{start} is the first step of the stage and t_{end} is the last step of the stage). The end learning rate is one tenth of the initial learning rate. The second training stage lasts for 40 epochs.

5.3.3 Third stage

In the third stage, the pre-trained stem weights are still unfrozen. The learning rate slowly decays from the end learning rate of stage 2 down to one hundredth of that. This is done in order to fine-tune the model. The third training stage lasts for 40 epochs.

The learning rate schedule is visualized in Figure 5.1.

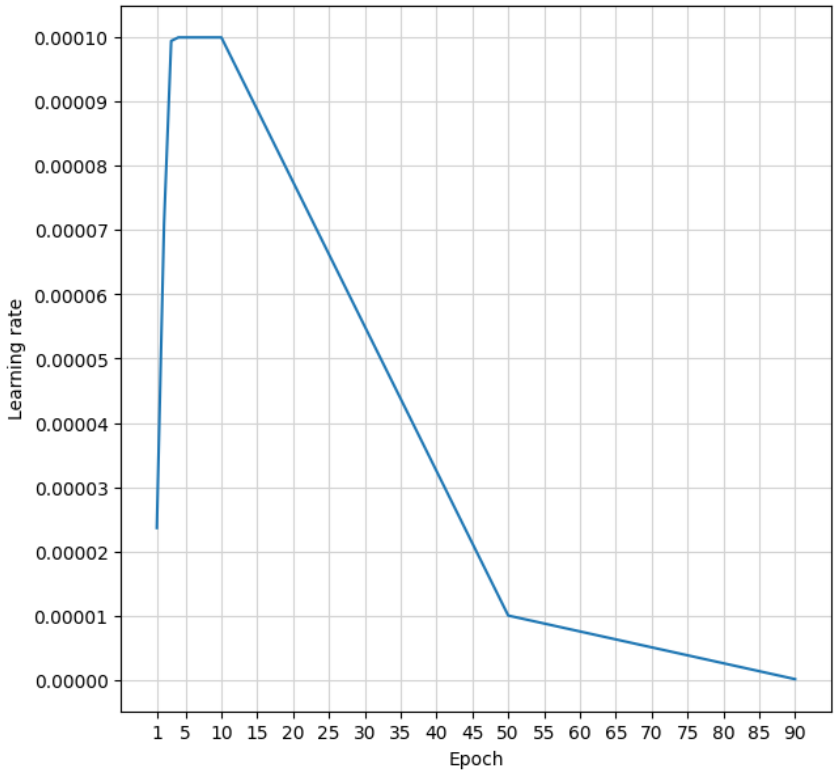

Figure 5.1: The learning rate schedule. The learning rate is slowly warmed up in the first ~ 3 epochs up to 10^{-4}. Then it stays constant until epoch 10, from which point it slowly decays to 10^{-5} in epoch 50. In the last stage, the learning rate decreases faster towards the end learning rate of 10^{-7} in epoch 90.

5.4 Mixed precision

The models are trained in mixed precision mode since all layers use float16 precision except for the output layers which use float32 precision. This has the advantage of reduced memory usage of the model and a higher training speed on recent NVIDIA GPUs.

Due to the lower memory usage of the model, there is more memory left on the GPU. Thus, the batch size can be raised, resulting in even more training duration speedup.

6 Experiments

6.1 Environment

The software to define, train and evaluate the models was written in Python using Tensorflow 2.9.3 with Keras [34]. All models were trained on Google Cloud's AI Platform. The machine type that was chosen is n1-standard-8, which features a 8-core CPU with 30GB RAM and a NVIDIA Tesla T4 GPU.

6.2 Procedure

All models in the experiments were trained using the Vertify training set (see Section 3.1.1) and then evaluated using the Vertify validation set (see Section 3.1.2). To evaluate the statistical significance of the results, a t-test was performed in Section 6.8. In Chapter 7 the resulting models were tested on other datasets featuring real images. Furthermore, the robustness of the models was assessed.

In order to find a suitable learning rate, the baseline multi-task experiment was trained three times with candidates $\eta_{5000_{\text{start}}} \in \{10^{-3}, 10^{-4}, 10^{-5}\}$[1]. The most promising learning rate in terms of the task performances was $\eta_{5000_{\text{start}}} = 10^{-4}$. This learning rate was used as a starting point for all experiments and was scaled throughout training as described in Section 5.3.

[1] The learning rate after 5000 warmup steps

6.3 Baselines

6.3.1 Single-task learning

The first experiment set a baseline for the multi-task experiments. In this experiment, the model featured only one head, namely the one for the optical gaze. The loss function used for the optical gaze was L_{MAE}.

The mean angular error for the optical gaze was 3.6910°. The progression of the mean angular error over epochs is shown in Figure 6.1.

6.3.2 Multi-task learning

The first multi-task learning experiment also set a baseline. Besides the optical gaze, this model featured the following additional tasks which were also included in the subsequent experiments: eyeball center, eyelid and iris keypoints as well as iris visibility estimation. The loss function used for the optical gaze and iris visibility was L_{MAE}, while for the eyeball center and the keypoints L_{MSE} was chosen.

The mean angular error for the optical gaze was 3.3728°. For eyeball center localization, the mean pixel error was 1.7344. The lid keypoints were estimated with a mean pixel error of 1.5173. In terms of iris keypoint localization, the mean pixel error was 1.2412. Lastly, the mean absolute error for iris visibility was 0.03145.

The progression of the mean angular error over epochs is shown in Figure 6.1. The progression of the other task metrics is shown in Figure 6.2.

The baseline results show that the model architecture seems to be suitable for the given task(s). The multi-task model performed better with a mean angular error of 3.3728°, decreasing the error of the single-task model by 0.3182°.

In the multi-task experiment the model might start to overfit on some

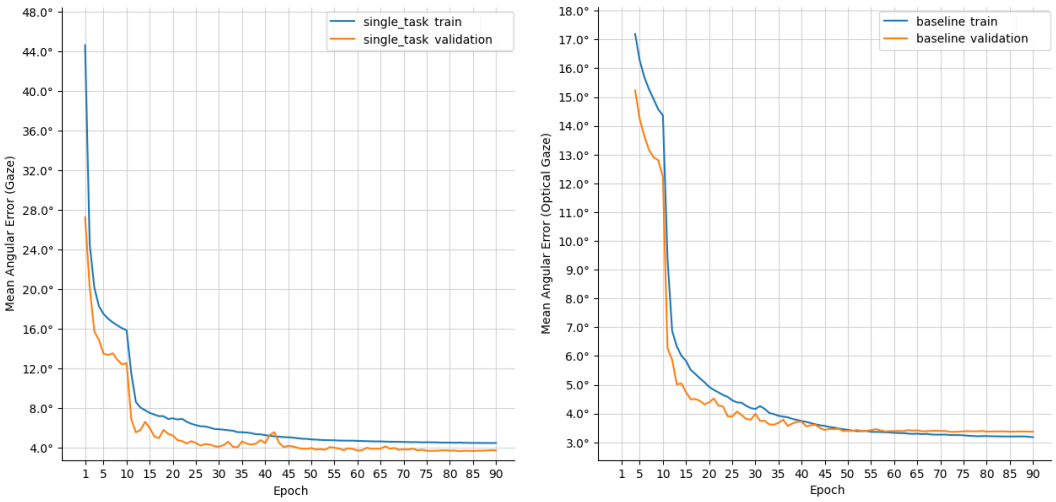

Figure 6.1: The mean angular error over the training epochs of the baselines. Left: single-task model, right: multi-task model.

tasks after 40 to 60 epochs since the training metric still decreases while the validation metric does not change much.

Figure 6.3 shows 25 example images from the validation set along with estimations from the model in this experiment.

As demonstrated in Figure 6.4, gaze estimation is more difficult when a person looks downwards (denoted by a high pitch). In Figure 6.5 it can be seen that variations in gaze yaw do not have a big impact on the difficulty. In Figure 6.6 it is shown that the model has a harder time to correctly estimate the gaze when the iris visibility is low.

6.4 Tackling negative transfer

Multi-task networks usually suffer from negative transfer between tasks. The experiments in this section are a continuation on top of the baseline multi-task learning experiment of the previous section.

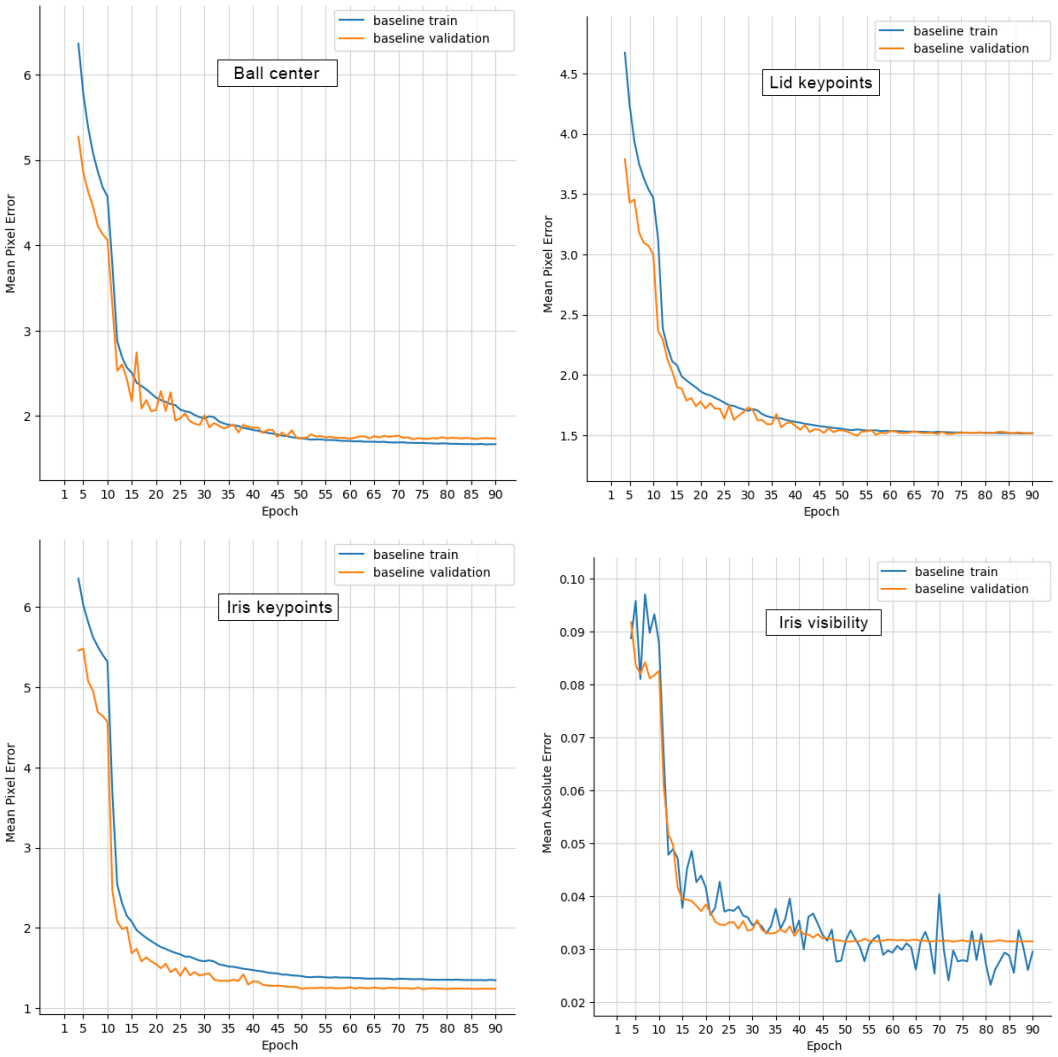

Figure 6.2: Metrics of the auxiliary tasks over time (baseline multi-task model). Top left: the mean pixel error of the ball center. Top right: the mean pixel error of the lid keypoints. Bottom left: the mean pixel error of the iris keypoints. Bottom right: the mean absolute error of the iris visibility.

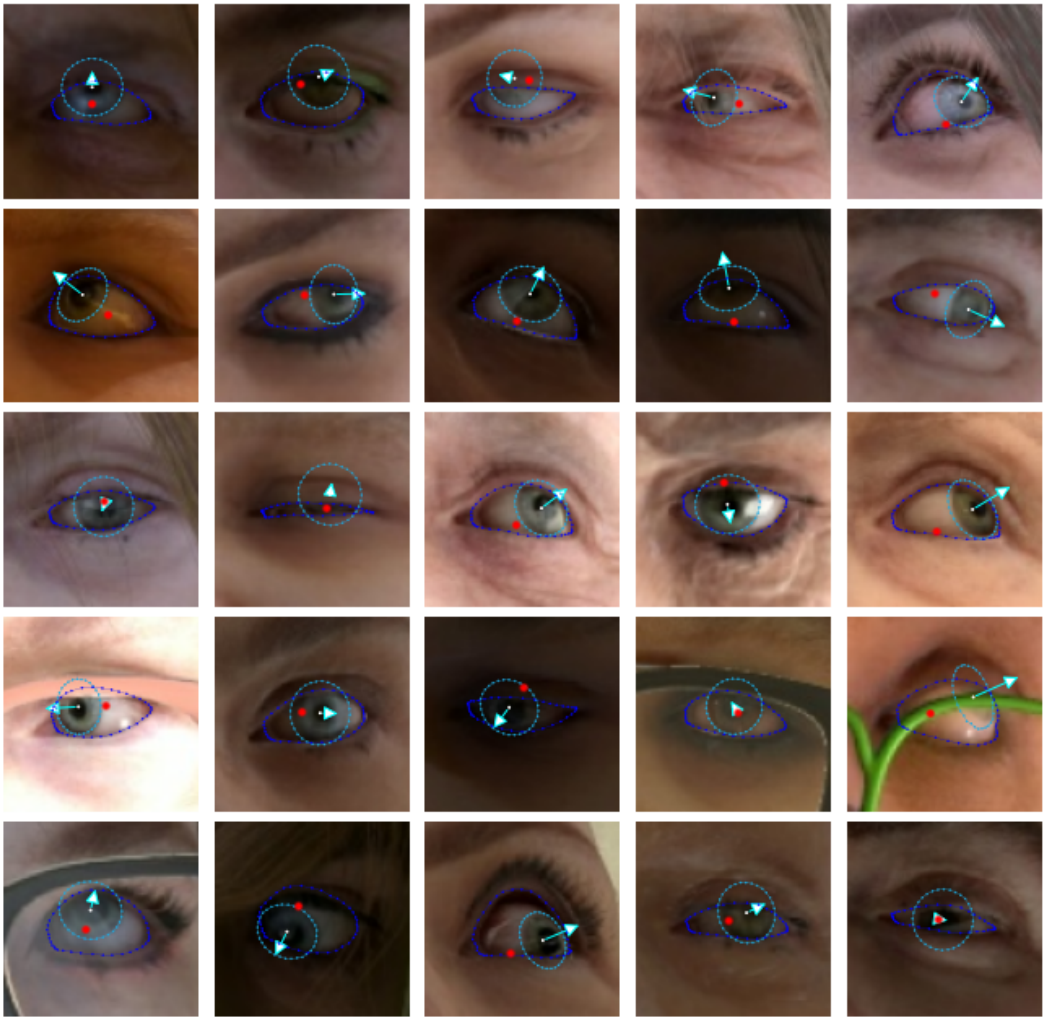

Figure 6.3: Visualization of estimations of the baseline multi-task model on 25 samples of the validation set. The estimated gaze is shown as an arrow in cyan. The iris keypoints are also in cyan and the lid keypoints in blue. The ball center keypoint is drawn in red. The estimated iris visibility is not shown.

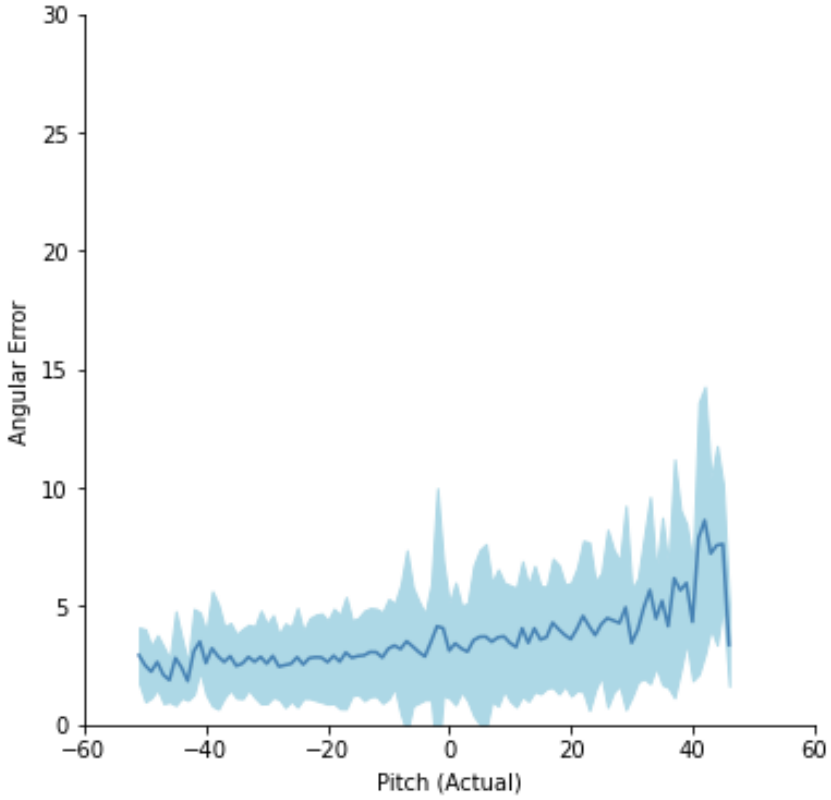

Figure 6.4: The mean angular error of the optical gaze as a function of the gaze pitch ground truth (validation set). This chart indicates that the more the gaze points to the ground (positive pitch), the harder it is to correctly estimate the gaze for the model. Conversely, the model is most confident at estimating gazes of eyes that look upwards. An explanation could be the fact that humans tend to open their eyelids more when looking upwards, which leads to more valuable details in the image.

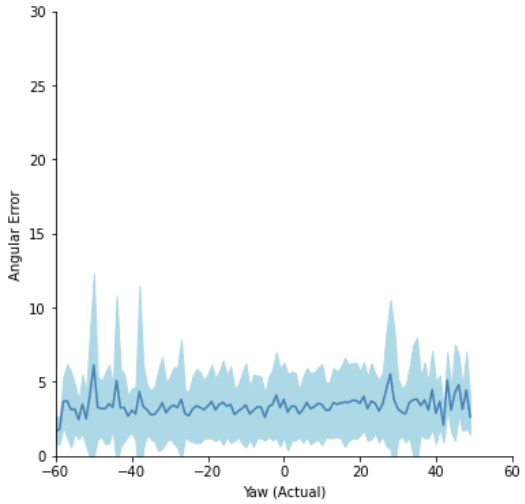

Figure 6.5: The mean angular error of the optical gaze as a function of the gaze yaw ground truth (validation set). This chart shows that the ground truth yaw barely influences the estimation performance of the model.

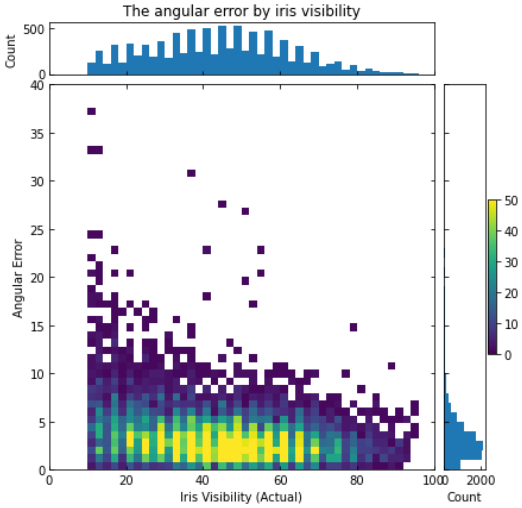

Figure 6.6: In this two-dimensional histogram, the iris visibility ground truth of each sample is binned on the horizontal axis, while the mean angular error of each sample is binned on the vertical axis (validation set). The chart shows that the model performs worse for samples with low iris visibility and vice versa.

6.4.1 PCGrad

In an attempt to decrease negative transfer, PCGrad [22] was applied in this experiment. More details about PCGrad can be found in Section 2.1.2. The loss functions for the tasks were chosen like in the baseline experiment.

The mean angular error for the optical gaze was 3.3694°. For eyeball center localization, the mean pixel error was 1.7287. The lid keypoints were estimated with a mean pixel error of 1.4584. In terms of iris keypoint localization, the mean pixel error was 1.2251. The mean absolute error for iris visibility was 0.03116.

After applying PCGrad, all tasks got estimated better compared to the baseline. Negative transfer shows to play a big role in multi-task model performance.

6.4.2 GradNorm

On examination of the task losses of the baseline multi-task experiment shown in Figure 6.7, it becomes apparent that the losses differ largely in scale. The optical gaze and iris visibility losses are much higher than the ones of the keypoint tasks.

A solution to this problem is task weighting, where for each task a weight is assigned. Then the total loss is the sum of the weighted task losses as defined in Equation 5.19.

Before manually weighting the tasks, GradNorm [23] was applied in this experiment. More details about GradNorm can be found in Section 2.1.2.

For the GradNorm hyperparameters, we set $\alpha = 1$ and the learning rate $\eta_{\text{GradNorm}} = 0.025$. The loss functions for the tasks were chosen like in the baseline experiment.

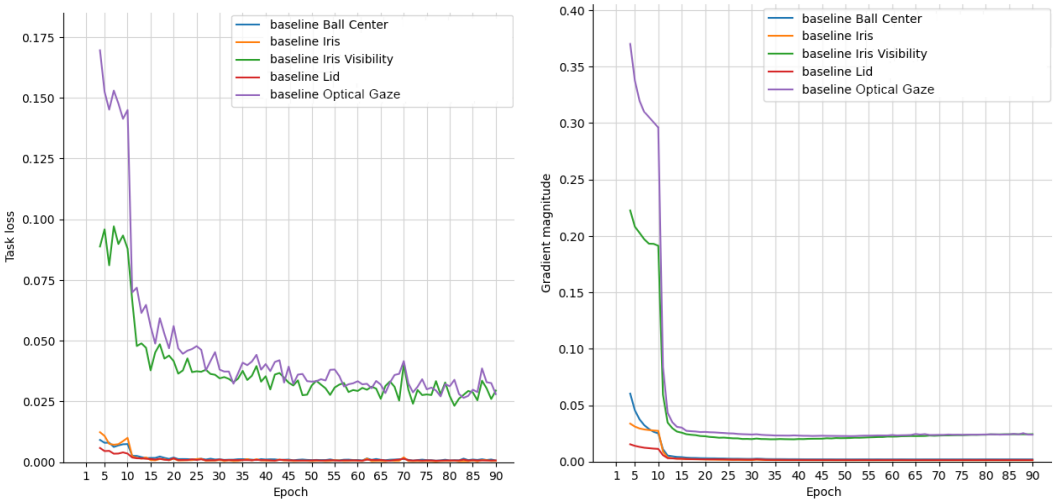

Figure 6.7: Imbalanced training losses / gradients in the previous experiments. Left: the loss for each task over time. Right: the gradient magnitude for each task over time.

The mean angular error for the optical gaze was 3.9884°. For eyeball center localization, the mean pixel error was 2.0207. The lid keypoints were estimated with a mean pixel error of 1.7114. In terms of iris keypoint localization, the mean pixel error was 1.4872. The mean absolute error for iris visibility was 0.03195.

All tasks performed worse than the baseline in this experiment. Since L_{MAE} decreases linearly and L_{MSE} quadratically, the algorithm cannot find a suitable way to balance the tasks. An improvement is proposed in the following section.

6.5 Learning with compatible loss functions (CLF)

As mentioned in the previous section, the loss functions are not compatible with each other in terms of their scale. In order to improve this, we switched to L_{MAngE} for the optical gaze and L_{MED} for the keypoints. The iris visibility loss remained to be L_{MAE}. In all the following experiments these loss

functions were utilized.

6.5.1 Single-task (CLF)

After changing the loss function, the mean angular error of the optical gaze improved from 3.6910° to 3.6399° for the single-task model. This indicates that minimizing the mean absolute error is inferior to minimizing the mean angular error for gaze estimation.

6.5.2 Multi-task baseline (CLF)

Also the multi-task learning model benefitted from the application of the new loss functions since all task metrics reported lower errors. The mean angular error of the optical gaze improved from 3.3728° to 3.2045°. The mean pixel error for the eyball center was 1.6041, for the lid keypoints it was 1.3077 and for the iris keypoints the error got reduced to 1.0938. The mean absolute error of the iris visibility improved to 0.03092.

6.5.3 PCGrad (CLF)

After the application of the new loss functions, PCGrad did not improve the performance on all tasks anymore. While the gaze error decreased to 3.1957° and the mean pixel error of the ball center got reduced to 1.5963, the mean pixel errors for lid and iris keypoints increased to 1.3090 and 1.0942, respectively. Also the iris visibility error got worse and was at 0.03113. The decreased efficacy of PCGrad could be caused by the change of loss functions, since that change probably reduced negative transfer between tasks.

6.5.4 GradNorm (CLF)

After the application of the new loss functions, the application of GradNorm could still not convince. The gaze error decreased to 3.5583° and the mean pixel error of the ball center got reduced to 1.6394. The mean pixel errors for lid and iris keypoints decreased to 1.3041 and 1.1943, respectively. Also the iris visibility error got worse and was at 0.0319.

6.6 Manual task weighting to balance gradients

6.6.1 Manual weighting

The task weights were chosen manually for this experiment. The goal of this weighting was to balance the gradient magnitudes of all tasks. The idea behind this method was inspired by Chen et al. with their GradNorm publication [23].

The multi-task baseline (CLF) experiment was used to calculate the weights. Let g_t be the gradient magnitude of the last epoch of task t. Then the mean task's gradient magnitude of the last epoch is given by

$$g = \frac{1}{T} \sum_t g_t \tag{6.1}$$

and the weight of a task is

$$w_t = \frac{g_t}{g} \,. \tag{6.2}$$

This resulted in the following weights:

$$\begin{aligned} w_{\text{OpticalGaze}} &= 0.7088, \\ w_{\text{BallCenter}} &= 1.0243, \\ w_{\text{Lid}} &= 1.5583, \\ w_{\text{Iris}} &= 1.4104, \\ w_{\text{IrisVisibility}} &= 0.7922. \end{aligned}$$

The mean angular error for the optical gaze was 3.2351°. For eyeball center localization, the mean pixel error was 1.6152. The lid keypoints were estimated with a mean pixel error of 1.2827. In terms of iris keypoint localization, the mean pixel error was 1.1005. The mean absolute error for the iris visibility was 0.024723. Iris visibility estimation got considerably improved. This task was likely weighted too low in previous runs.

6.6.2 Manual weighting with PCGrad

For this experiment, the weights are recalculated based on the gradient magnitudes of the experiment above. Additionally, the weights are scaled such that the sum of weights equals the sum of the weights of the previous experiment.

This resulted in the following weights:

$$\begin{aligned} w_{\text{OpticalGaze}} &= 0.6270, \\ w_{\text{BallCenter}} &= 0.9963, \\ w_{\text{Lid}} &= 1.7196, \\ w_{\text{Iris}} &= 1.4931, \\ w_{\text{IrisVisibility}} &= 0.6580. \end{aligned}$$

The mean angular error for the optical gaze was 3.1623°. For eyeball center localization, the mean pixel error was 1.586. The lid keypoints were estimated with a mean pixel error of 1.269. In terms of iris keypoint localization, the mean pixel error was 1.0729. The mean absolute error for the iris visibility was 0.020407.

This model achieved the best performance so far. The combination of manual task weighting with PCGrad enhanced the estimation accuracy of all tasks.

6.7 Contrastive Learning

In the experiments of this section, different contrastive learning techniques were applied. The goal behind contrastive learning is to teach a model to identify similarities and differences between samples, which can be achieved by enhancing the latent space of the stem. This should help the model to generalize better and thus improve performance and robustness. Contrastive learning should also be able to help closing the domain gap between synthesized and real data.

The model architecture was altered for the experiments in this section. A projection head was appended to the model, which was used for an auxiliary task which aimed to alter the embedding space of the stem. The projection head is described in more detail in Section 4.3.

All experiments in this section are based on the experiment in Section 6.6.1.

6.7.1 Contrastive loss

In this experiment, the loss function $L_{\text{Contrastive}}$ was applied with the goal to pull together samples in a batch with a similar eye gaze and vice-versa. To achieve this, triplets of samples had to be chosen. One anchor sample, one positive sample to the anchor and one negative sample to the anchor. To make the batch size divisible by 3, it was reduced from 128 to 120 for this experiment.

The first training run was conducted with hyperparameters $m = 1$ and $w_{\text{Projection}} = 1$, where $w_{\text{Projection}}$ is the weight of the contrastive loss. Like the other task weights, this weight helped to balance the gradient magnitude of the contrastive learning task.

Applying the contrastive loss improved all tasks except for the iris visibility. The mean angular error for the optical gaze was 3.1703°. For eyeball center localization, the mean pixel error was 1.5923. The lid keypoints were

estimated with a mean pixel error of 1.2682. In terms of iris keypoint localization, the mean pixel error was 1.071. The mean absolute error for the iris visibility was 0.024631.

A visualization of the latent space of the stem output can be seen in Figure 6.8. The images appear to be separated depending on the gaze. Samples of persons looking to the left (from the camera perspective) are clustered on the left side of the plot and vice versa. Also images of persons looking towards the top and floor are clustered on the top and bottom of the plot, respectively. We conclude that the stem has learned to produce useful representations for eye gaze estimation.

The experiment was repeated with hyperparameters $m = 0.5, w_{\text{Projection}} = 1.5$ and also once with $m = 2.0, w_{\text{Projection}} = 0.15$ to get a notion of the impact of the margin parameter. The errors for these subsequent experiments did not change for the better. The results are shown in Table 6.1.

6.7.2 SimCLR

Since SimCLR [25] works with multiple augmentations per sample, two augmentations were generated per sample in each step as shown in Figure 6.9. Thus, each batch featured 128 images which were sourced from 64 base samples. The loss function for the projection head was L_{SimCLR} and the hyperparameters were chosen as $\tau = 0.5$ and $w_{\text{Projection}} = 0.35$, where τ is the temperature of the contrastive loss.

The performance of all tasks except for the iris keypoint estimation decreased compared to the model without the contrastive learning task. The mean angular error for the optical gaze was 3.2999°. For eyeball center localization, the mean pixel error was 1.648. The lid keypoints were estimated with a mean pixel error of 1.2917. In terms of iris keypoint localization, the mean pixel error was 1.095. The mean absolute error for the iris visibility was 0.02523.

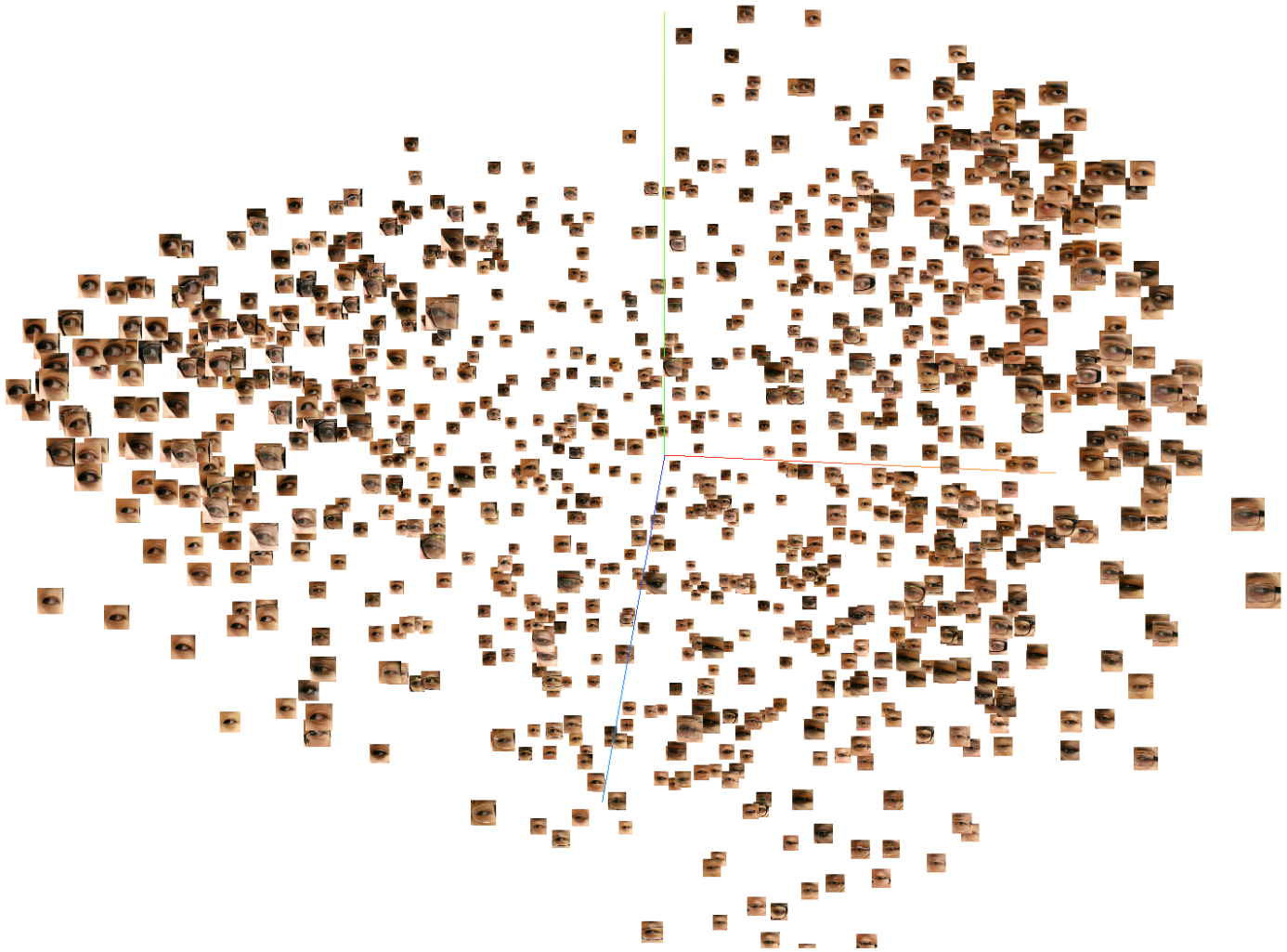

Figure 6.8: Stem embeddings of 1024 samples taken from the Columbia Gaze (CGaze) dataset, visualized by TensorBoard Projector. The embeddings were generated by the model trained in the contrastive loss experiment with $m = 1$. The visualization technique that was chosen is PCA. The red, green and blue axes correspond to the x, y and z components of the PCA.

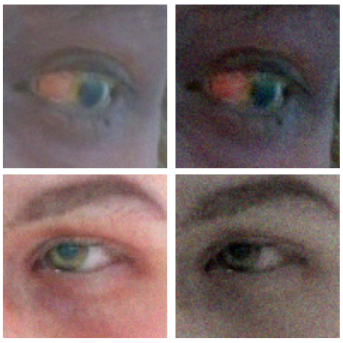

Figure 6.9: Two base samples, both were augmentated twice. SimCLR aims to pull together the corresponding embeddings in latent space.

As the training dataset features tuples of 8 samples with the same head pose and eye gaze, it is possible to provide the contrastive loss with all of these views per sample. A batch then features $16 \cdot 8$ images, i.e. samples from 16 different head pose and eye gaze configurations.

The results worsened with this approach. The mean angular error for the optical gaze was 3.4903°. For eyeball center localization, the mean pixel error was 1.7029. The lid keypoint error stayed the same at 1.2917. The iris keypoint error was 1.1067. The mean absolute error for the iris visibility was 0.025044.

6.7.3 Supervised Contrastive Loss (SupCon)

The loss function for the SupCon [26] experiments was L_{SupCon}. The SupCon loss can be seen as an extension to the SimCLR loss. In addition to providing multiple augmentations per base sample, one can also label them. Thus it is possible to link multi-augmented samples with other ones that share the same gaze. In order to make use of this feature, the 4 complex variations of a head pose / gaze configuration were provided as one multi-augmented sample, as well as the 4 simple variations. That means a batch consisted of $16 \cdot 2 \cdot 4$ images.

The hyperparameters were set to $\tau = 0.5$ and $w_{\text{Projection}} = 0.35$. This approach did not seem to improve the performance either, even though the results were slightly better than in the previous experiment. The mean angular error for the optical gaze was 3.4339°. For eyeball center localization, the mean pixel error was 1.6834. The lid keypoint error stayed the same at 1.2899. The iris keypoint error was 1.0973. The mean absolute error for the iris visibility was 0.024868.

The experiment was repeated with hyperparameters $\tau = 0.1, w_{\text{Projection}} = 1.75$ and also once with $\tau = 2.5, w_{\text{Projection}} = 0.07$ to get a notion of the impact of the temperature parameter. The errors for these subsequent experiments did not significantly change for the better. The results are shown in Table 6.1.

6.8 Statistical significance of results

In this section, we examine whether the improvements are statistically significant. Since it would be too computationally expensive to train each model multiple times, we only trained the Baseline (CLF) and the Contrastive $\text{loss}_{m=1}$ models 5 times each. The validation errors and statistics can be found in Table 6.2. The independent samples t-test results indicate that the errors of the two models do very likely not share the same normal distribution. The improvements are therefore statistically significant.

Experiment	MAngE Optical Gaze	MPE Ball Center	MPE Lid	MPE Iris	MAE Iris Visibility
Single-task	3.6910°				
Multi-task baseline	3.3728°	1.7344	1.5173	1.2412	0.0315
PCGrad	3.3694°	1.7287	1.4584	1.2251	0.0312
GradNorm	3.9884°	2.0207	1.7114	1.4872	0.0320
Single-task (CLF)	3.6399°				
Multi-task baseline (CLF)	3.2045°	1.6041	1.3077	1.0938	0.0309
PCGrad (CLF)	3.1957°	1.5963	1.3090	1.0942	0.0311
GradNorm (CLF)	3.5583°	1.6394	1.3041	1.1943	0.0319
Manual weighting	3.2351°	1.6152	1.2827	1.1005	0.0247
+ PCGrad	**3.1623°**	**1.5860**	1.2690	1.0729	**0.0204**
Contr. loss $_{m=1}$	3.1703°	1.5923	1.2682	**1.0710**	0.0246
+ PCGrad	3.1721°	1.5961	**1.2674**	1.0739	0.0245
Contr. loss $_{m=0.5}$	3.1770°	1.5904	1.2769	1.0753	0.0245
Contr. loss $_{m=2}$	3.1680°	1.5933	1.2695	1.0751	0.0243
SimCLR $_{2\text{ Aug.}}$	3.2999°	1.6480	1.2917	1.0950	0.0252
SimCLR $_{\text{Tuples}}$	3.4903°	1.7029	1.2917	1.1067	0.0250
SupCon $_{\tau=0.5}$	3.4339°	1.6834	1.2899	1.0973	0.0249
SupCon $_{\tau=0.1}$	3.5290°	1.6591	1.2884	1.1170	0.0249
SupCon $_{\tau=2.5}$	3.4416°	1.6511	1.2913	1.0949	0.0246

Table 6.1: The validation errors of the experiments. The lowest error for each task is highlighted. For each experiment, the mean angular error (MAngE) of the optical gaze, the mean pixel error (MPE) of the ball center, lid and iris keypoints and the mean absolute error (MAE) for the iris visibility are shown.

Experiment	MAngE Optical Gaze	MPE Ball Center	MPE Lid	MPE Iris	MAE Iris Visibility
Baseline (CLF) $_1$	3.2045°	1.6041	1.3077	1.0938	0.0309
Baseline (CLF) $_2$	3.2095°	1.6233	1.3209	1.1025	0.0312
Baseline (CLF) $_3$	3.2049°	1.6129	1.3031	1.0987	0.0312
Baseline (CLF) $_4$	3.2061°	1.6094	1.3130	1.0974	0.0311
Baseline (CLF) $_5$	3.2093°	1.6058	1.3142	1.1021	0.0313
Baseline (CLF) $_{\text{Mean} \pm \text{SD}}$	$3.2069° \pm 0.002141°$	1.6111 ± 0.006813	1.3118 ± 0.006042	1.0989 ± 0.003209	0.0311 ± 0.000127
Contr. loss $_{m=1,1}$	3.1703°	1.5923	1.2682	1.0710	0.0246
Contr. loss $_{m=1,2}$	3.1788°	1.6004	1.2738	1.0742	0.0246
Contr. loss $_{m=1,3}$	3.1579°	1.5951	1.2731	1.0793	0.0246
Contr. loss $_{m=1,4}$	3.1424°	1.5823	1.2782	1.0710	0.0246
Contr. loss $_{m=1,5}$	3.1604°	1.5907	1.2765	1.0801	0.0246
Contr. loss $_{m=1,\text{Mean} \pm \text{SD}}$	$3.1620° \pm 0.012285$	1.5922 ± 0.005931	1.2740 ± 0.003417	1.0751 ± 0.003926	0.0246 ± 0.000027

Value	Independent samples t-test Optical Gaze	Ball Center	Lid	Iris	Iris Visibility
T-Score	7.2010	4.1935	10.8977	9.3791	101.1111
P-Value	0.000092	0.003023	0.000004	0.000014	0.000000

Table 6.2: Top: the results for experiments Baseline (CLF) and Contrastive loss $_{m=1}$ with 5 training runs each. The training runs were initialized with different random seeds. For each experiment, the mean and standard deviation are also shown. Bottom: the results of the Student's t-tests for each task. The t-tests indicate that the improved results are reproducible.

7 Benchmarks

7.1 Performance on datasets with real-world data

Since datasets which feature real images capture the visual gaze, the optical gaze output of the model had to be altered. As the eyes of every person are a bit different, this is best done via user calibration. In this work no calibration is done. Therefore the average offset angle between visual and optical gaze was simply added to the optical gaze output. The relationship between optical and visual gaze is described in Section 2.3.2.

The results of the benchmarks can be seen in Table 7.1. There is a notable domain gap between the synthetic datasets and the real-world datasets. In addition to the fact that synthesized images are less realistic, the benchmark datasets largely feature people of asian ethnicity, which are not represented at all in the training set. All models reached a gaze error above 7° for the Columbia Gaze (CGaze) dataset and an error above 10° for the MPIIGaze dataset.

The biases in the table show how far the average predictions were away from the average labels. Table 7.1 also shows the results after bias correction, which are naturally a bit better but still not comparable to results that could be achieved with calibration.

In general, the improvements that could be seen when testing on the validation set were largely not confirmed by the results on real data. Rather, some of the worst performing models on the validation set achieved the best results in this benchmark. This indicates that the models overfit the synthetic data.

The bias-corrected benchmark on the Columbia Gaze (CGaze) dataset indicates that multi-task learning with compatible loss functions (Baseline CLF) did improve eye gaze estimation performance compared to single-task learning. All other experiments (except for $\text{SupCon}_{\tau=0.5}$) could not bring about any further error reductions. For reference, other cross-dataset validation results include 8.7° (GazeML [42]) and 7.1° (GeoGaze [42]) when trained on UT-Multiview and without calibration.

The results on the MPIIGaze dataset do not really seem to correlate with the results on the Columbia Gaze dataset. The best performing model (GradNorm) on the MPIIGaze dataset was simultaneously the worst performing model on the Columbia Gaze dataset. Still, the results are competitive. Wood et al. reported an error of 9.95° (UnityEyes [31], see Section 2.4.1). Baltrusaitis et al. achieved an error of 9.1° (OpenFace 2.0 [43]). Other cross-dataset validation results include 10.8° (GazeNet, [44]), 9.51° (MeNet [45]), 8.8° (GeoGaze [42]) and 8.4° (GazeML [42]) when trained on the UT-Multiview dataset.

7.2 Robustness

All models were also tested for robustness. The corruption set with severity level 2 of 3 was chosen (see Figure 3.3). More information on the corruption set can be found in Section 3.1.3.

Table 7.2 features the results for the optical gaze estimation. The results for eyeball center, lid and iris keypoints are shown in Table 7.3, Table 7.4 and Table 7.5, respectively. For the robustness of iris visibility estimation, the results are shown in Table 7.6.

When comparing the single- and multi-task baseline models, it becomes apparent that the multi-task learning models are more robust to corrupted images. While contrastive learning did not seem to help closing the domain gap in the previous benchmark, it has helped to improve robustness. The experiment where SimCLR with 2 augmentations ($\text{SimCLR}_{2\ \text{Aug.}}$) was applied showed promising results. This model achieved the best results for all tasks.

Especially keypoint estimation was considerably more robust. Furthermore, corruptions that were not seen in training (dense image warp (DIW), JPEG compression (JPEG), Gaussian blur) also had a less detrimental effect on the performance compared to the other experiments. This shows that SimCLR also improved robustness on corruptions that were not featured in the augmentation pipeline.

The tuple dataset approach ($\text{SimCLR}_{\text{Tuples}}$, all SupCon models) did show to be inferior to the multi-augmented sample ($\text{SimCLR}_{2\ \text{Aug.}}$) experiment. The explanation for this might be that the variation in tuple pairs is greater, but also does not change much for the whole duration of the training. For each tuple there are 8 fixed scenarios, where only the augmentation changes in each training step. Since the scenarios never change, the model might excessively minimize the distance for these given scenarios and thus decrease generalization ability.

Experiment	MAngE (Visual)		Pitch / Yaw Bias		Bias-corrected	
	CGaze	MPIIGaze	CGaze	MPIIGaze	CGaze	MPIIGaze
Single-task	7.445°	10.171°	−2.032° / 0.804°	−3.688° / 1.280°	7.186°	9.389°
Baseline	7.591°	10.702°	−2.001° / 0.580°	−4.600° / −0.110°	7.351°	9.786°
PCGrad	7.464°	11.104°	−2.032° / 0.799°	−5.108° / −1.094°	7.187°	9.960°
GradNorm	8.083°	**10.092°**	−2.909° / 0.750°	−4.201° / −0.216°	7.626°	**9.258°**
Single-task (CLF)	7.617°	11.157°	−2.736° / 0.679°	−4.980° / 1.230°	7.187°	10.000°
Baseline (CLF)	**7.348°**	10.677°	−1.715° / 0.775°	−4.577° / −0.224°	7.131°	9.771°
PCGrad (CLF)	7.492°	10.381°	−1.731° / 0.708°	−3.712° / −0.992°	7.287°	9.695°
GradNorm (CLF)	7.830°	10.541°	−2.562° / 0.742°	−3.449° / −1.343°	7.449°	9.878°
Manual weighting	7.464°	10.183°	−1.864° / 0.752°	−2.660° / −1.373°	7.225°	9.744°
+ PCGrad	7.412°	10.817°	−1.855° / 0.856°	−4.444° / 0.242°	7.175°	9.947°
Contr. loss $_{m=1}$	7.422°	10.204°	−1.940° / 0.745°	−3.890° / −0.242°	7.174°	9.487°
+ PCGrad	7.368°	10.584°	−1.821° / 0.723°	−4.109° / −0.233°	7.142°	9.794°
Contr. loss $_{m=0.5}$	7.421°	10.428°	−1.811° / 0.800°	−3.604° / −0.804°	7.209°	9.806°
Contr. loss $_{m=2}$	7.396°	10.612°	−1.886° / 0.776°	−3.714° / −0.699°	7.159°	9.929°
SimCLR $_{\text{2 Aug.}}$	7.582°	12.038°	−2.288° / 0.742°	−7.995° / 4.008°	7.257°	11.221°
SimCLR $_{\text{Tuples}}$	7.483°	10.300°	−1.702° / 1.135°	−3.898° / 0.738°	7.244°	9.491°
SupCon $_{\tau=0.5}$	7.355°	10.547°	−1.702° / 1.120°	−3.503° / 0.950°	**7.120°**	9.891°
SupCon $_{\tau=0.1}$	7.403°	10.393°	−1.513° / 0.891°	−3.579° / −0.644°	7.220°	9.681°
SupCon $_{\tau=2.5}$	7.451°	10.365°	−1.949° / 1.061°	−4.101° / 0.650°	7.167°	9.500°

Table 7.1: The results of the benchmarks on datasets with real images. For each experiment it shows for both benchmark datasets: a) The mean angular error (MAngE) of the visual gazes. The best results are highlighted. b) The bias of the model, which shows how far away the average output of the model was compared to the average label. c) The mean angular error (MAngE) of the visual gazes after bias correction. The best results are highlighted.

Experiment	Mean	Noise		Digital		Color				Blur	
		Gaussian	Iso	DIW	JPEG	Brightness	Contrast	Hue	Saturation	Gaussian	Motion
Single-task	6.103°	5.740°	5.595°	5.787°	4.931°	4.230°	6.010°	4.064°	3.889°	8.726°	12.061°
Baseline	5.783°	5.388°	5.252°	5.460°	4.654°	3.950°	5.621°	3.831°	3.664°	8.406°	11.607°
PCGrad	5.845°	5.446°	5.277°	5.446°	4.641°	4.064°	5.656°	3.860°	3.704°	8.495°	11.856°
GradNorm	6.623°	6.117°	6.120°	6.142°	5.378°	4.773°	6.547°	4.539°	4.460°	9.412°	12.739°
Single-task (CLF)	6.270°	5.621°	5.676°	5.902°	5.023°	4.384°	6.145°	4.329°	4.008°	9.156°	12.452°
Baseline (CLF)	5.595°	5.158°	4.993°	5.356°	4.501°	3.801°	5.333°	3.684°	3.571°	8.098°	11.455°
PCGrad (CLF)	5.589°	5.136°	4.996°	5.234°	4.463°	**3.696°**	5.317°	3.614°	3.534°	8.387°	11.509°
GradNorm (CLF)	6.234°	5.576°	5.443°	5.674°	4.860°	4.176°	5.895°	4.112°	3.905°	9.774°	12.930°
Manual weighting	5.658°	5.127°	5.012°	5.326°	4.527°	3.838°	5.330°	3.688°	3.558°	8.229°	11.940°
+ PCGrad	5.595°	4.996°	**4.824°**	5.231°	4.465°	3.794°	5.250°	3.615°	3.482°	8.230°	12.062°
Contr. loss $_{m=1}$	5.587°	5.047°	4.925°	5.217°	4.426°	3.748°	5.273°	3.614°	**3.400°**	8.058°	12.165°
+ PCGrad	5.549°	4.999°	4.867°	5.205°	4.334°	3.808°	5.320°	3.668°	3.538°	8.219°	11.531°
Contr. loss $_{m=0.5}$	5.555°	5.044°	4.891°	5.317°	4.462°	3.754°	**5.201°**	**3.601°**	3.499°	7.962°	11.819°
Contr. loss $_{m=2}$	5.692°	5.068°	4.980°	5.244°	4.466°	3.776°	5.349°	3.640°	3.531°	8.456°	12.412°
SimCLR $_{\text{2 Aug.}}$	**5.469°**	**4.775°**	4.945°	**5.039°**	**4.241°**	3.960°	5.244°	3.779°	3.627°	**7.649°**	11.434°
SimCLR $_{\text{Tuples}}$	5.808°	5.372°	5.439°	5.502°	4.751°	4.028°	5.726°	3.931°	3.718°	8.259°	11.352°
SupCon $_{\tau=0.5}$	5.667°	5.232°	5.300°	5.385°	4.574°	3.940°	5.500°	3.812°	3.673°	8.243°	**11.008°**
SupCon $_{\tau=0.1}$	5.811°	5.622°	5.620°	5.438°	4.680°	4.064°	5.580°	3.893°	3.725°	8.066°	11.421°
SupCon $_{\tau=2.5}$	5.708°	5.304°	5.219°	5.515°	4.595°	3.965°	5.568°	3.875°	3.655°	8.327°	11.064°

Table 7.2: The robustness test results for optical gaze estimation. The mean angular error is provided for each model / corruption combination. The best results are highlighted.

Experiment	Mean	Noise		Digital		Color				Blur	
		Gaussian	Iso	DIW	JPEG	Brightness	Contrast	Hue	Saturation	Gaussian	Motion
Baseline	1.827	1.606	1.622	1.665	1.500	1.309	1.571	1.294	1.281	2.621	3.800
PCGrad	1.841	1.613	1.622	1.637	1.482	1.292	1.544	1.282	1.269	2.772	3.901
GradNorm	2.034	1.850	1.847	1.882	1.734	1.529	1.813	1.504	1.516	2.813	3.853
Baseline (CLF)	1.763	1.506	1.510	1.595	1.447	1.218	1.440	1.213	1.219	2.708	3.769
PCGrad (CLF)	1.729	1.483	1.492	1.544	1.413	1.191	1.420	1.186	1.190	2.688	3.682
GradNorm (CLF)	1.898	1.612	1.617	1.645	1.533	1.317	1.559	1.320	1.305	3.034	4.040
Manual weighting	1.744	1.492	1.493	1.563	1.418	1.229	1.416	1.212	1.216	2.571	3.832
+ PCGrad	1.748	1.461	1.464	1.532	1.407	1.192	1.420	1.176	**1.171**	2.683	3.976
Contr. loss $_{m=1}$	1.752	1.464	1.472	1.543	1.398	**1.184**	1.410	1.184	1.180	2.681	4.002
+ PCGrad	1.700	1.475	1.461	1.540	1.387	1.198	1.405	**1.175**	1.190	2.514	3.651
Contr. loss $_{m=0.5}$	1.718	1.480	1.476	1.552	1.414	1.207	1.405	1.208	1.193	2.546	3.700
Contr. loss $_{m=2}$	1.779	1.452	1.472	1.532	1.401	1.206	1.389	1.186	1.193	2.793	4.167
SimCLR $_{2 \text{ Aug.}}$	**1.606**	**1.390**	**1.401**	**1.458**	**1.267**	1.189	**1.372**	1.188	1.189	**2.287**	**3.321**
SimCLR $_{\text{Tuples}}$	1.764	1.533	1.562	1.583	1.492	1.267	1.492	1.245	1.250	2.565	3.654
SupCon $_{\tau=0.5}$	1.744	1.504	1.528	1.543	1.450	1.225	1.442	1.217	1.221	2.634	3.674
SupCon $_{\tau=0.1}$	1.773	1.533	1.584	1.564	1.446	1.241	1.450	1.236	1.224	2.640	3.811
SupCon $_{\tau=2.5}$	1.737	1.507	1.511	1.545	1.418	1.222	1.432	1.208	1.204	2.639	3.683

Table 7.3: The robustness test results for eyeball center estimation. The mean pixel error is provided for each model / corruption combination. The best results are highlighted.

Experiment	Mean	Noise		Digital		Color				Blur	
		Gaussian	Iso	DIW	JPEG	Brightness	Contrast	Hue	Saturation	Gaussian	Motion
Baseline	1.404	1.274	1.290	1.279	1.211	1.103	1.295	1.089	1.078	1.835	2.589
PCGrad	1.406	1.270	1.284	1.247	1.186	1.080	1.264	1.059	1.052	1.888	2.725
GradNorm	1.549	1.466	1.469	1.413	1.364	1.254	1.465	1.231	1.221	1.897	2.711
Baseline (CLF)	1.232	1.119	1.136	1.104	1.052	0.949	1.102	0.926	0.927	1.613	2.392
PCGrad (CLF)	1.222	1.101	1.133	1.102	1.045	0.940	1.107	0.928	0.918	1.593	2.354
GradNorm (CLF)	1.276	1.170	1.190	1.141	1.095	0.981	1.161	0.964	0.955	1.688	2.410
Manual weighting	1.224	1.093	1.113	1.080	1.031	0.937	1.111	0.913	0.908	1.620	2.433
+ PCGrad	1.218	1.078	1.109	1.071	1.017	**0.914**	1.082	0.902	**0.893**	1.598	2.512
Contr. loss $_{m=1}$	1.217	1.078	1.113	1.070	1.012	0.916	1.084	**0.899**	0.899	1.569	2.528
+ PCGrad	1.215	1.088	1.110	1.074	1.022	0.925	1.094	0.903	0.902	1.600	2.431
Contr. loss $_{m=0.5}$	1.227	1.094	1.106	1.081	1.032	0.933	1.091	0.913	0.906	1.623	2.485
Contr. loss $_{m=2}$	1.225	1.069	1.108	1.073	1.016	0.921	1.085	0.902	0.897	1.631	2.544
SimCLR $_{2\text{ Aug.}}$	**1.134**	**1.029**	**1.039**	**1.019**	**0.960**	0.924	**1.076**	0.900	0.900	**1.388**	**2.111**
SimCLR $_{\text{Tuples}}$	1.212	1.108	1.149	1.085	1.045	0.936	1.108	0.917	0.913	1.533	2.326
SupCon $_{\tau=0.5}$	1.226	1.099	1.174	1.098	1.048	0.940	1.108	0.920	0.916	1.609	2.347
SupCon $_{\tau=0.1}$	1.236	1.106	1.158	1.090	1.050	0.939	1.115	0.929	0.914	1.605	2.458
SupCon $_{\tau=2.5}$	1.223	1.092	1.121	1.077	1.030	0.942	1.116	0.920	0.911	1.598	2.423

Table 7.4: The robustness test results for lid keypoint estimation. The mean pixel error is provided for each model / corruption combination. The best results are highlighted.

Experiment	Mean	Noise		Digital		Color				Blur	
		Gaussian	Iso	DIW	JPEG	Brightness	Contrast	Hue	Saturation	Gaussian	Motion
Baseline	1.626	1.508	1.463	1.519	1.397	1.176	1.640	1.128	1.084	2.272	3.071
PCGrad	1.643	1.531	1.472	1.518	1.408	1.195	1.664	1.125	1.093	2.320	3.107
GradNorm	1.867	1.776	1.738	1.717	1.663	1.427	1.948	1.360	1.318	2.481	3.245
Baseline (CLF)	1.510	1.363	1.311	1.402	1.292	1.038	1.490	0.990	0.959	2.183	3.070
PCGrad (CLF)	1.509	1.384	1.338	1.394	1.301	1.026	1.486	0.984	0.963	2.180	3.033
GradNorm (CLF)	1.629	1.476	1.452	1.448	1.386	1.120	1.619	1.104	1.048	2.425	3.214
Manual weighting	1.513	1.348	1.358	1.413	1.296	1.036	1.491	0.983	0.946	2.166	3.094
+ PCGrad	1.492	1.330	1.292	1.382	1.273	**1.019**	**1.437**	**0.970**	0.928	2.160	3.130
Contr. loss $_{m=1}$	1.501	1.348	1.309	1.380	1.256	1.021	1.467	0.982	**0.923**	2.128	3.196
+ PCGrad	1.498	1.337	1.291	1.371	1.235	1.037	1.485	0.980	0.933	2.238	3.073
Contr. loss $_{m=0.5}$	1.499	1.368	1.322	1.372	1.253	**1.019**	1.443	0.975	0.933	2.172	3.133
Contr. loss $_{m=2}$	1.506	1.356	1.321	1.385	1.254	1.025	1.490	0.984	0.947	2.174	3.120
SimCLR $_{2 \text{ Aug.}}$	**1.426**	**1.268**	**1.237**	**1.322**	**1.180**	1.064	**1.437**	0.988	0.955	**1.968**	**2.836**
SimCLR $_{\text{Tuples}}$	1.519	1.399	1.403	1.389	1.302	1.033	1.532	1.018	0.948	2.155	3.015
SupCon $_{\tau=0.5}$	1.510	1.365	1.409	1.377	1.300	1.045	1.483	1.004	0.947	2.175	2.996
SupCon $_{\tau=0.1}$	1.516	1.445	1.453	1.402	1.307	1.045	1.525	1.003	0.952	2.101	2.925
SupCon $_{\tau=2.5}$	1.506	1.367	1.351	1.382	1.276	1.045	1.521	1.015	0.960	2.172	2.970

Table 7.5: The robustness test results for iris keypoint estimation. The mean pixel error is provided for each model / corruption combination. The best results are highlighted.

Experiment	Mean	Noise		Digital		Color				Blur	
		Gaussian	Iso	DIW	JPEG	Brightness	Contrast	Hue	Saturation	Gaussian	Motion
Baseline	0.065	0.050	0.047	0.062	0.048	0.039	0.056	0.038	**0.037**	0.112	0.164
PCGrad	0.066	0.050	0.048	0.060	0.048	0.039	0.055	0.038	0.037	0.117	0.168
GradNorm	0.067	0.051	0.050	0.060	0.050	0.040	0.055	0.039	0.038	0.115	0.169
Baseline (CLF)	0.063	0.050	0.048	0.062	0.049	0.039	0.056	0.039	0.038	0.104	0.148
PCGrad (CLF)	0.062	0.050	0.047	0.060	0.048	**0.038**	0.056	0.038	**0.037**	0.100	0.149
GradNorm (CLF)	0.063	0.050	0.048	0.059	0.048	**0.038**	**0.054**	0.038	**0.037**	0.105	0.148
Manual weighting	0.063	0.051	0.047	0.061	0.048	0.040	0.055	0.039	0.039	0.104	0.151
+ PCGrad	0.064	0.050	0.046	0.061	0.047	0.039	0.055	0.039	**0.037**	0.102	0.157
Contr. loss $_{m=1}$	0.062	0.050	0.047	0.061	0.047	0.039	0.055	**0.037**	**0.037**	0.096	0.149
+ PCGrad	**0.061**	0.050	0.046	0.061	0.048	0.040	**0.054**	0.039	0.038	0.094	**0.142**
Contr. loss $_{m=0.5}$	0.064	0.051	0.048	0.061	0.049	0.039	0.056	0.038	0.038	0.103	0.157
Contr. loss $_{m=2}$	**0.061**	0.050	0.047	0.060	0.047	**0.038**	0.055	0.038	**0.037**	0.093	0.149
SimCLR $_{\text{2 Aug.}}$	**0.061**	**0.048**	**0.045**	**0.056**	**0.044**	0.040	**0.054**	0.038	0.038	0.102	0.144
SimCLR $_{\text{Tuples}}$	0.062	0.052	0.048	0.066	0.049	0.040	0.056	0.039	0.038	**0.090**	0.144
SupCon $_{\tau=0.5}$	0.063	0.051	0.049	0.066	0.050	0.039	0.055	0.039	0.038	0.095	**0.142**
SupCon $_{\tau=0.1}$	0.063	0.052	0.049	0.064	0.049	0.039	0.056	0.039	0.038	0.101	0.146
SupCon $_{\tau=2.5}$	0.063	0.050	0.048	0.063	0.048	0.039	0.057	0.038	0.038	0.101	0.147

Table 7.6: The robustness test results for iris visibility estimation. The mean absolute error is provided for each model / corruption combination. The best results are highlighted.

8 Conclusion

In this section, the thesis is summarized briefly and an overview is given for potential future work.

8.1 Summary

In Chapter 1, an introduction to eye gaze estimation was given, along with techniques which help to build models that are capable to learn this task. The goal of the thesis was to build a framework which is capable to apply multi-task learning methods in order to improve performance and robustness of eye gaze estimation.

Chapter 2 features all the theory that was needed to start working on solutions for the stated problem. At first, we gave an overview of the different ways a multi-task learning architecture can look like and the pros and cons of multi-task models. Especially negative transfer was discussed as the biggest hurdle to deal with, while also providing literature on how to reduce negative transfer (PCGrad, GradNorm). Furthermore, contrastive learning was discussed concisely. Since the goal was to interpret eye images, it was also important to obtain knowledge about the anatomy of the human eye. Most importantly, the fact that the visual axis of the eye can not be determined directly by looking at an eye is a problem that can only really be tackled with user calibration or training on real images. We also stated the difference between remote and near-eye images and that this thesis focuses on remote images. For training and validation, these images were generated, which had previously been shown to be an effective way to gather a reasonable amount of data.

The synthesized datasets were presented in Chapter 3, along with corrupted versions of the validation set for robustness tests and two benchmark datasets which feature eye images of actual persons. At the end of this chapter, the data augmentation pipeline was described.

The model architecture for the experiments in this thesis is depicted in Chapter 4. A ResNet50 was used to extract features of the images which could then be used by the heads to deliver the estimates for their various tasks. Also a projection head was added, which helped to learn useful representations in our contrastive learning experiments.

Chapter 5 covered the mathematical formulas that were required to optimize the models. We defined our loss functions for the tasks and the contrastive learning techniques. Also the learning rate schedule that was used for all experiments was presented.

The main part of this work is featured in Chapter 6, where all experiments are documented. At first, baselines for single-task and multi-task learning were established. These experiments already showed the potential of multi-task learning, since the validation error of the optical gaze improved promisingly. After this step, we tried to reduce negative transfer, where PCGrad showed to be a simple and useful tool to reduce negative effects of the training signals of tasks on each other. When applying GradNorm, it became apparent that the task losses need to be compatible in order to achieve good results. Therefore, we switched to losses that scale similarly as they are minimized, leading to improved results. We then also started to weight the task losses manually, leading to further reduction of the errors. Finally, a few experiments were conducted in which contrastive learning was explored. A simple contrastive loss that minimized the distance of images with similar gazes and vice versa in latent space proved to be a solid approach. On the other hand, our training set that featured tuples of multiple images with the same gaze in different scenes could not bring about any improvements, but instead the models trained using this dataset performed worse.

The models were then tested for performance on real data and also for robustness in Chapter 7. Unfortunately, the improvements that were ob-

served in Chapter 6 did largely not translate to the tests on real images. This showed the significance of the domain gap between the datasets. We had to transform our optical gaze estimations to visual gaze estimations, which is not optimal if no user calibration is performed. Another challenge was that for real images, the eyes are not centered on the camera's principal axis, which results in distorted views of eyes that the models had previously not seen.
The robustness tests showed that models can benefit strongly from contrastive learning. Corruptions definitely had the least detrimental effect on the SimCLR model where we augmented each sample twice and pulled these paired samples together in latent space.

8.2 Future work

For future work, there are many factors which could likely be improved. Some thoughts and pointers are noted in this subsection.

The potential of synthesized datasets is high. The generated eye images could look more realistic, have greater variation (more head scans, ethnicities, lighting conditions, etc.) and arguably the eyes should not always be centered on the principal axis of the camera. One should also carefully choose the distribution of head poses and gazes in the training set. A wider range of gazes, for example, provides more flexibility at the cost of precision. The generated datasets feature gaze yaws of up to 60°, while the real data only ranges to about 20° in both directions (compare Figure 3.2, Figure 3.5 and Figure 3.6). Incorporating information about the face (e.g. head poses) in training generally also leads to better results [29]. Possibly also other tasks (e.g. semantic segmentation) could show synergy effects combined with gaze estimation. Another factor that plays a role is image resolution. Due to perfomance constraints, images were downsampled to $96\,x\,96$ in this work. Higher resolutions may lead to increased performance.

As stated before, adding user calibration would be hugely beneficial. Garde et al. reported improvements of around 50% due to calibration for cases where the number of users was high [46].

Moreover, different model architectures should be tried out. It can also be helpful to calculate affinities between tasks as outlined by Fifty et al. [21]. These affinities help to decide on which tasks should be trained together in a model. Another idea would be to add more layers that are shared between similar tasks. For example, all keypoint tasks could share some additional layers. Evaluating both eyes at the same time also improves accuracy, even if both eye gazes are estimated separately and geometrically merged afterwards [44].

Bibliography

[1] Susmitha Mohan and Manoj Phirke. "Eye Gaze Estimation Invisible and IR Spectrum for Driver Monitoring System." In: *Signal and Image Processing: An International Journal* 11 (Oct. 2020), pp. 1–20. DOI: 10.5121/sipij.2020.11501.

[2] Sheng Wang et al. *Follow My Eye: Using Gaze to Supervise Computer-Aided Diagnosis*. Apr. 2022.

[3] Katarzyna Harezlak and Pawel Kasprowski. "Application of eye tracking in medicine: A survey, research issues and challenges." In: *Computerized Medical Imaging and Graphics* 65 (2018). Advances in Biomedical Image Processing, pp. 176–190. ISSN: 0895-6111. DOI: https://doi.org/10.1016/j.compmedimag.2017.04.006.

[4] Robert Lupu et al. "Eye tracking based communication system for patient with major neoro-locomotor disabilites." In: Jan. 2011, pp. 1–5. ISBN: 978-1-4577-1173-2.

[5] Viviane Clay, Peter König, and Sabine Koenig. "Eye Tracking in Virtual Reality." In: *Journal of Eye Movement Research* 12 (Apr. 2019). DOI: 10.16910/jemr.12.1.3.

[6] Inessa Bekerman, Paul Gottlieb, and Michael Vaiman. "Variations in Eyeball Diameters of the Healthy Adults." In: *Journal of ophthalmology* 2014 (Nov. 2014), p. 503645. DOI: 10.1155/2014/503645.

[7] Hesham Gharieb, Hisham Shalaby, and Ihab Othman. "Distribution of angle lambda and pupil offset as measured by combined Placido Scheimpflug Topography." In: *International Ophthalmology* 43 (July 2022). DOI: 10.1007/s10792-022-02394-3.

[8] Reagan L. Galvez et al. "Object Detection Using Convolutional Neural Networks." In: *TENCON 2018 - 2018 IEEE Region 10 Conference*. 2018, pp. 2023–2027. DOI: 10.1109/TENCON.2018.8650517.

[9] Neha Sharma, Vibhor Jain, and Anju Mishra. "An Analysis Of Convolutional Neural Networks For Image Classification." In: *Procedia Computer Science* 132 (2018). International Conference on Computational Intelligence and Data Science, pp. 377–384. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.05.198. URL: https://www.sciencedirect.com/science/article/pii/S1877050918309335.

[10] Farhana Sultana, Abu Sufian, and Paramartha Dutta. "Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey." In: *Knowledge-Based Systems* 201-202 (Aug. 2020), p. 106062. DOI: 10.1016/j.knosys.2020.106062. URL: https://doi.org/10.1016%2Fj.knosys.2020.106062.

[11] Andronicus A. Akinyelu and Pieter Blignaut. "Convolutional Neural Network-Based Methods for Eye Gaze Estimation: A Survey." In: *IEEE Access* 8 (2020), pp. 142581–142605. DOI: 10.1109/ACCESS.2020.3013540.

[12] Mahbub Hussain, Jordan Bird, and Diego Faria. "A Study on CNN Transfer Learning for Image Classification." In: June 2018.

[13] Alvaro Figueira and Bruno Vaz. "Survey on Synthetic Data Generation, Evaluation Methods and GANs." In: *Mathematics* 10.15 (2022), pp. 1–41.

[14] Benedikt T. Imbusch, Max Schwarz, and Sven Behnke. "Synthetic-to-Real Domain Adaptation using Contrastive Unpaired Translation." In: *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE, Aug. 2022. DOI: 10.1109/case49997.2022.9926640.

[15] Tariq Alkhalifah, Hanchen Wang, and Oleg Ovcharenko. "MLReal: Bridging the gap between training on synthetic data and real data applications in machine learning." In: *Artificial Intelligence in Geosciences* 3 (2022), pp. 101–114. ISSN: 2666-5441. DOI: https://doi.org/10.1016/j.aiig.2022.09.002.

[16] Michael Crawshaw. *Multi-Task Learning with Deep Neural Networks: A Survey*. 2020. arXiv: 2009.09796 [cs.LG].

[17] Michael Crawshaw. *Multi-Task Learning with Deep Neural Networks: A Survey*. 2020. arXiv: 2009.09796 [cs.LG].

[18] Rich Caruana. "Multitask Learning." In: *Machine Learning* 28 (July 1997). DOI: 10.1023/A:1007379606734.

[19] Bernardino Romera-Paredes et al. "Exploiting Unrelated Tasks in Multi-Task Learning." In: *AIS-TATS* 22 (Jan. 2012).

[20] Jonathan Baxter. "A Bayesian/information theoretic model of learning to learn via multiple task sampling." In: *Machine Learning*. 1997, pp. 7–39.

[21] Christopher Fifty et al. *Efficiently Identifying Task Groupings for Multi-Task Learning*. 2021. arXiv: 2109.04617 [cs.LG].

[22] Tianhe Yu et al. *Gradient Surgery for Multi-Task Learning*. 2020. arXiv: 2001.06782 [cs.LG].

[23] Zhao Chen et al. *GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks*. 2018. arXiv: 1711.02257 [cs.CV].

[24] Weiran Huang et al. "Towards the Generalization of Contrastive Self-Supervised Learning." In: *The Eleventh International Conference on Learning Representations*. 2023.

[25] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: 2002.05709 [cs.LG].

[26] Prannay Khosla et al. *Supervised Contrastive Learning*. 2021. arXiv: 2004.11362 [cs.LG].

[27] E. D. Guestrin and M. Eizenman. "General theory of remote gaze estimation using the pupil center and corneal reflections." In: *IEEE Transactions on Biomedical Engineering* 53.6 (2006), pp. 1124–1133. DOI: 10.1109/TBME.2005.863952.

[28] Zhengyang Wu et al. *MagicEyes: A Large Scale Eye Gaze Estimation Dataset for Mixed Reality*. Mar. 2020.

[29] Xucong Zhang et al. "It's Written All Over Your Face: Full-Face Appearance-Based Gaze Estimation." In: *CoRR* abs/1611.08860 (2016). arXiv: 1611.08860.

[30] Stephen Langton, Helen Honeyman, and Emma Tessler. "The influence of head contour and nose angle on the perception of eye-gaze direction." In: *Perception and psychophysics* 66 (Aug. 2004), pp. 752–71. DOI: 10.3758/BF03194970.

[31] Erroll Wood et al. "Learning an Appearance-based Gaze Estimator from One Million Synthesised Images." eng. In: *Proceedings ETRA 2016*. Charleston, SC, USA: ACM, 2016, pp. 131–138. ISBN: 978-1-4503-4125-7. DOI: 10.1145/2857491.2857492.

[32] B.A. Smith et al. "Gaze Locking: Passive Eye Contact Detection for Human-Object Interaction." In: *ACM Symposium on User Interface Software and Technology (UIST)*. Oct. 2013, pp. 271–280.

[33] X. Zhang et al. "Appearance-based gaze estimation in the wild." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 4511–4520. DOI: 10.1109/CVPR.2015.7299081.

[34] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. 2016. arXiv: 1603.04467 [cs.DC].

[35] Abien Fred Agarap. *Deep Learning using Rectified Linear Units (ReLU)*. 2019. arXiv: 1803.08375 [cs.NE].

[36] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[37] Kaiming He et al. *Identity Mappings in Deep Residual Networks*. 2016. arXiv: 1603.05027 [cs.CV].

[38] Jia Deng et al. "ImageNet: A large-scale hierarchical image database." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[39] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping." In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.

[40] Kihyuk Sohn. "Improved Deep Metric Learning with Multi-class N-pair Loss Objective." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016.

[41] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[42] Harsimran Kaur, Swati Jindal, and Roberto Manduchi. "Rethinking Model-Based Gaze Estimation." In: *Proc. ACM Comput. Graph. Interact. Tech.* 5.2 (May 2022). DOI: 10.1145/3530797.

[43] Tadas Baltrusaitis et al. "OpenFace 2.0: Facial Behavior Analysis Toolkit." In: *2018 13th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2018)*. 2018, pp. 59–66. DOI: 10.1109/FG.2018.00019.

[44] Xucong Zhang et al. *MPIIGaze: Real-World Dataset and Deep Appearance-Based Gaze Estimation*. 2017. arXiv: 1711.09017 [cs.CV].

[45] Yunyang Xiong, Hyunwoo Kim, and Vikas Singh. "Mixed Effects Neural Networks (MeNets) With Applications to Gaze Estimation." In: June 2019, pp. 7735–7744. DOI: 10.1109/CVPR.2019.00793.

[46] Gonzalo Garde et al. "Low-Cost Eye Tracking Calibration: A Knowledge-Based Study." In: *Sensors* 21.15 (2021). ISSN: 1424-8220. DOI: 10.3390/s21155109.