MSc Elective Project
Supervisor: : Fabricio Batista Narcizo

Head pose estimation using Deep Learning

Authored by:
Benjamin Ahrentløv Olsen (beao@itu.dk)

Contents

Abstract

Estimating the head pose of a person is a significant task with various applications such as helping with fitting 3D objects to the face, user interaction in computer systems or applications, and tracking the driver's direction of view. There are two practical approaches to estimating the head pose based on an image of a human head. One is to estimate the landmarks of the human face and then solve the 2D to 3D correspondence problem with a 3D human head model, also called a landmark-based approach. Another method is to train a Deep Convolutional Neural Network to predict the head pose directly from the image of a human head, also called an appearance-based approach. An example of an appearance-based approach is the architecture called HopeNet, which was presented in "Fine-Grained Head Pose Estimation Without Keypoints" and produced state-of-the-art results when compared to commonly used landmark based approaches. However, this method was trained on synthetic datasets and only tested on one dataset with actual data (BIWI) due to the lack of other real datasets of human heads corresponding to Euler angles. The 2018 iPad Pro has a TrueDepth sensor, which, together with the Apple Neural Engine, can capture detailed information from the human face and extract the head pose from this. We present a new approach to create a custom dataset with images of heads and corresponding intrinsic Euler angles (pitch, yaw, roll) and 3D position by recording videos with a 2018 iPad Pro and collecting the frames with the corresponding Euler angles. We show that this data collection method and the HopeNet architecture show great potential for head pose estimation by producing prediction errors similar to what the 2018 iPad Pro produces.
Keywords: *Head pose estimation*, *Euler angles*, *Facial Landmarks*, *Appearance-based approach*, *Synthetic data*, *Actual data*

Introduction

Head pose estimation research in Computer Vision focuses on predicting the pose of a human head in an image, predicting the Euler angles, which describe the 3D rotation of the human head. Euler angles consist of three separate components, namely: pitch, yaw, and roll. These angles correspond to the head rotations around the X-, Y-, and Z-axis of an object in 3D space [7]. By predicting these three rotation values given an image of a human head, we can obtain the head's direction with high accuracy. This approach can support various tasks, such as manipulating multimedia content, user interaction, tracking whether a driver is focusing on the road, among others.

Head pose estimation uses two primary approaches when given an image. The first one is to estimate the facial landmarks, establishing the correspondence with a 3D head model, and performing alignment to obtain the head pose. We also need the camera parameters to project the 3D points onto the image plane for this approach. However, this method has multiple steps that can introduce errors. For example, making an inaccurate prediction of the facial landmarks, using a 3D head model does not work well for the task, or use a flawed method for performing face alignment. The second approach is to predict the Euler angles directly from images without taking facial landmarks into account. This approach has the potential of being more robust, faster, and more accurate because there are fewer steps that can fail before a prediction. Ruiz, Chong, and Rehg [19] introduced the network architecture called HopeNet, which showed great potential for appearance-based approaches by producing state-of-the-art results on several challenging datasets.

Previous research studies have mainly used synthetic datasets of different head poses due to the very few datasets with actual data [19, 22, 5]. The 2018 iPad Pro has a depth-sensing sensor called *TrueDepth* and is capable of estimating the head pose accurately, by passing the images obtained with the TrueDepth sensor to the Apple Neural Engine. In this research, we show our custom dataset using the 2018 iPad Pro to record several videos and save the corresponding Euler angles for each frame. The proposed data collection method has a lot of potential, since the creation of a large and diverse dataset for head pose estimation require less resources compared to other methods and is easy to make a public task.

This master project aims to (1) determine if the appearance-based approach HopeNet can produce similar accuracy as a 2018 iPad Pro, and (2) determine if the trained model can make robust predictions on low resolution images.

The remaining report is organized as follows. Section 3 gives background and theoretical information on head pose estimation, Euler angles, related work, feed forward neural networks, convolutional neural networks, and gradient descent. Section 4 describes the implementation and experimental setup. Section 5 presents the conducted experiments and the main analysis methods, where the results from this section will be compared with related work in Section 6 which also suggests future research. Finally, Section 7 concludes the report.

Background and Motivation

3.1 Head Pose Estimation

Head Pose Estimation is the process of inferring the orientation of the head from a 2D image. The 3D orientation of a human head consists of three Degrees Of Freedom, also called the Euler angles, which are typically inferred relative to the view of the camera in the context of computer vision [16].

There are different ways of estimating the head pose of an image of a human head. One is to find landmarks on the human head and afterwards calculating the Euler angles from the landmarks, another way is to ignore landmarks and use a Deep Convolutional Neural Network to predict the Euler angles based on appearance.

3.1.1 Euler angles

The three Euler angles are characterized by: pitch, yaw, and roll, which can be described in terms of rotations around the X-, Y-, and Z-axes, respectively.

The three rotations are defined as three 3×3 matrices to rotate a 3D object around the given axis (i.e., the human head in this case) [7]:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where ϕ is the pitch angle, θ is the yaw angle, and ψ is the roll angle.

Generally the rotation of a 3D object is described with a single rotation matrix:

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}.$$

This overall rotation matrix is generated by multiplying the individual rotation matrices in different ways, e.g., $R_z(\psi)R_y(\theta)R_x(\phi)$.

Given a rotation matrix with numerical values at each position in the matrix, we can calculate the Euler angles (ϕ, θ, and ψ) and from that derive which direction the 3D object (human head) is pointing.

For example, given a XYZ rotation matrix (meaning that the first rotation was around the X-axis, then the Y-axis, and at last the Z-axis):

$$\mathrm{R}_x(\phi)R_y(\theta)R_z(\psi) = \begin{bmatrix} cos(\theta)cos(\psi) & -cos(\theta)sin(\psi) & sin(\theta) \\ cos(\psi)sin(\phi)sin(\theta) + cos(\phi)sin(\psi) & cos(\phi)cos(\psi) - sin(\phi)sin(\theta)sin(\psi) & -cos(\theta)sin(\phi) \\ -cos(\phi)cos(\psi)sin(\theta) + sin(\phi)sin(\psi) & cos(\psi)sin(\phi) + cos(\phi)sin(\theta)sin(\psi) & cos(\phi)cos(\theta) \end{bmatrix}$$

From this we can obtain the three Euler angles as such:

$$\begin{gathered} \theta = asin(sin(\theta)) \\ \phi = atan2(-cos(\theta)sin(\phi), cos(\phi)cos(\theta)) \\ \psi = atan2(-cos(\theta)sin(\psi), cos(\theta)cos(\psi)) \end{gathered}$$

In terms of the human head, these rotations are controlled by the muscles in the neck. The pitch angle corresponds to moving the head up or down (*extension* and *flexion*), the yaw angle corresponds to moving the head from side to side (*vertical rotation*), and the roll angle corresponds to moving the head down to each shoulder (*lateral bending*) [1], as it can be seen in Figure 3.1.1

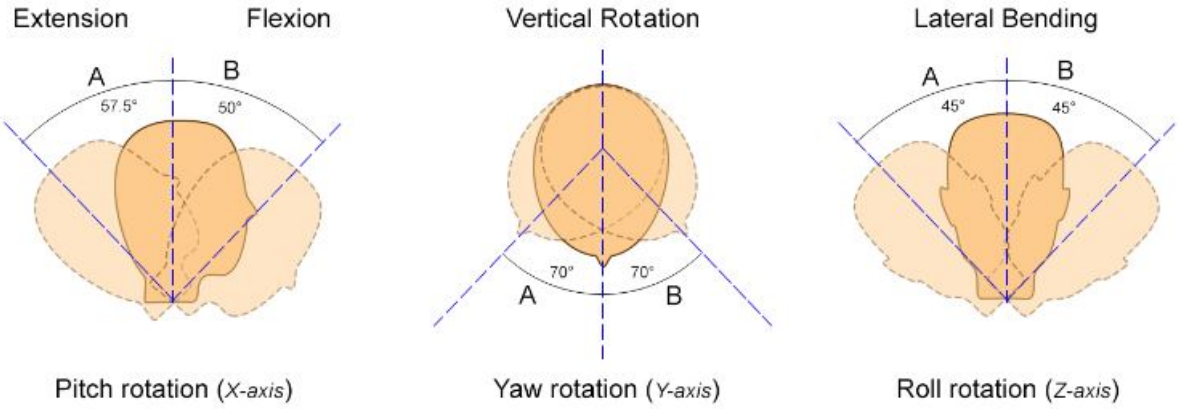


Figure 3.1.1: *Human head rotations based on three major neck motions (from [17])*

3.2 Related work

3.2.1 Landmark-based approach

There have been developed robust deep learning methods for extracting the 2D facial landmarks from images in recent years, which can be used for head pose estimation. It is possible to calculate the head pose by establishing correspondence between the landmarks and a 3D head model, then performing alignment [5, 14].

For example, when the Deep Neural Network predicts the 68 labeled 2D facial landmarks, the points can then be matched with a 3D head model by performing facial alignment, resulting in a rotated 3D head model that represents the head pose.

This example is a two-step process because the landmarks have to be estimated first, then the head pose can be calculated by performing 3D face alignment. In a situation where we only need to predict the head pose, it seems unnecessary to predict the landmarks before the head pose, which is the primary motivation for appearance-based approaches.

3.2.2 Appearance-based approach

The primary motivation of this report is the appearance-based head pose estimator called Hopenet [19].

Unlike landmark-based approaches, this model predicts the head pose based only on the appearance of the head in a given image. The advantage of this is that it is not dependent on finding facial landmarks before predicting the head pose. Therefore, it can always output a prediction since there is no step before that can fail.

They used ResNet (see Section 3.4.1) as a backbone structure to handle the input image and gave the output of ResNet to three different fully-connected layers (one for each Euler angle), which allows the network to adjust itself during training based on three other signals (losses). The model is a multi-loss Convolutional Neural Network, which combines two different loss functions to calculate the loss for each iteration. The authors decided to use Categorical Cross-Entropy loss for binned pose classification and Mean Squared Error as regression loss for predictions of the Euler angles, combined for each fully-connected layer each Euler angle classification and prediction has an effect of the adjustment of weights in the network. The final loss for each Euler angle is:

$$angle\_loss = H(y, \hat{y}) + \alpha \cdot MSE(y, \hat{y}),$$

where H is the Categorical Cross-Entropy loss, α is the regression coefficient, MSE is the Mean Squared Error, y are the ground truth values for the given angle, and $\hat{y}$ are the predicted values for the given angle. Figure 3.2.1 shows the final network structure:

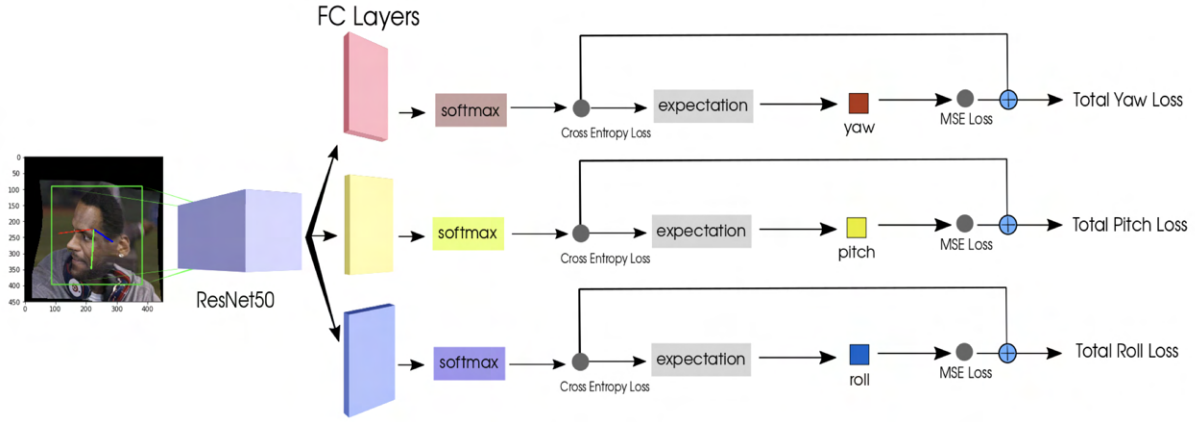


Figure 3.2.1: *Illustration of Hopenet (from [19])*

This master project uses the Hopenet architecture and compares this model's accuracy with the ground truth data from a 2018 iPad Pro.

3.3 Feedforward Neural Networks

Feedforward Neural Networks (FFNN) are multi-layered perceptrons, focusing on solving challenging tasks by finding complex patterns in the task-related information. They are called feedforward because the data is passed in the forward direction from the first layer to the last [21].

The first layer of the FFNN is called the input layer. It contains the information that the FFNN should analyze to solve the given task. It can have one or more hidden layers that will help find more complex patterns in the data from the input layer. At last, an output layer outputs values that seek to solve the given task for the FFNN.

Each node has an activation function that introduces non-linearity to the Neural Network. They often determine how vital that particular neuron is to solve the given task by outputting a numbered value. Different activation functions can be applied depending on the task that should be solved, for example:

Function	Equation
Softmax	$f(x_i) = \frac{e_i^x}{\sum_{j=0}^{n} e_j^x}$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
ReLU	$f(x) = max(0, x)$

The activation function of one neuron is passed a numbered value (x) which consists of weights (w) from all neurons in the previous layer multiplied by their activation function's output (y) plus some bias (b):

$$x = (\sum_{i=0}^{n} y_i w_i) + b$$

Assuming that the selected activation functions are the best fit for the model, it is the weights between all neurons that have the most significant impact on the accuracy of the model, which motivates the search for the most optimal weights.

This is accomplished by training the FFNN, which can be achieved with Gradient Descent through Back-propagation (see Section 3.5).

3.4 Convolutional Neural Networks

CNN is a Deep Learning Algorithm has the ability to identify and classify objects in a given image [20], as shown in Figure 3.4.1.

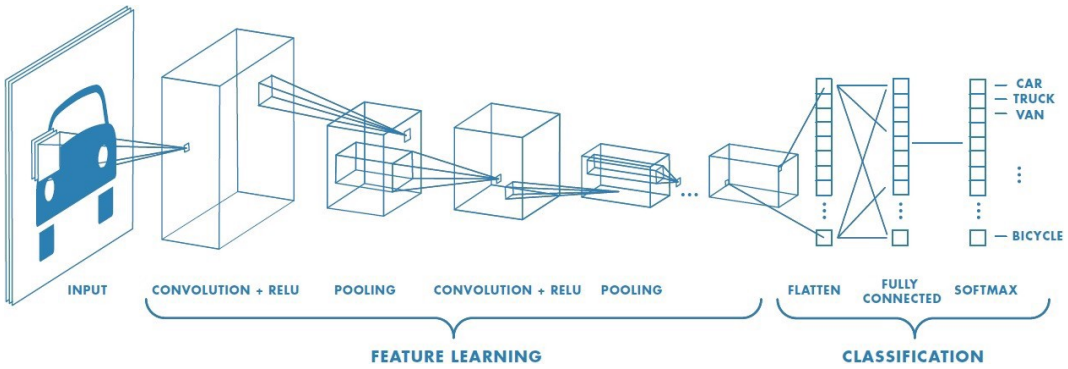


Figure 3.4.1: *Illustration of CNN [20]*

A CNN consists of Convolutional and Pooling layers followed by a fully connected layer.

The first step is to consider the input image. The task becomes much more computationally intensive since the input images can be quite large. This is the purpose of the CNN, to reduce the size of the image without sacrificing the most important features. With optimal filters, a CNN can accurately identify or classify objects by capturing significant features in the image. These filters are also known as *kernels*. The CNN can then learn the kernel values that maximize accuracy.

Kernels are initialized with random values at the first iteration and will be updated through gradient descent back-propagation. The dimensions of the kernels, however, are fixed throughout the training of the CNN.

A kernel of size $n \times n$ is applied to the image by performing matrix multiplication on the selected $n \times n$ sections of the given image. The image sections are determined by setting a stride with a number, which moves the kernel along the X-axis and Y-axis of the image with a step equal to the given stride value. If the stride is equal to one, then all $n \times n$ sections of the image will be processed with the given kernel. When the kernel gets to the maximum x coordinate of the image, it will move in the y-direction and start from 0 on the X-axis. The output of the Convolutional layer will then be a smaller image with the most significant features of the given image.

The Convolutional layer can then pass its output to the next layer, a Pooling layer. Instead of extracting features like the Convolutional layer, it aims to reduce the size of the image while preserving the current most important features. This is done by applying a kernel on the given image, which then applies a function to all values in the current section of the image. The function could, for example, extract the maximum value of all the values in the given matrix, thus extracting the most important feature of the current section in the image.

It is typical to add a fully connected layer to learn the high-level features captured by the Convolutional layers, in order to solve the given task.

3.4.1 ResNet

ResNet is a Deep Convolutional Neural Network made to address losing performance when adding too many layers to Neural Networks [10]. As they mentioned in the paper, theoretically, when a Neural Network has more layers than it needs, it should learn the identity function on the redundant layers. However, as the authors showed, this is not a trivial task for standard Neural Networks.

The authors presented a solution to this issue by creating shortcut connections that skip one or more layers and perform an identity mapping where one layers output is added to the outputs of the stacked layers, as shown in Figure 3.4.2. They also called this "residual blocks":

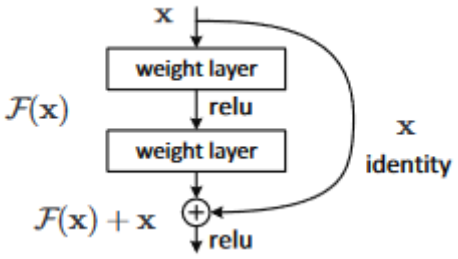


Figure 3.4.2: *Illustration of a residual block [10]*

This allowed them to create much deeper networks that increase in performance rather than decrease in performance.

ResNet was used as a backbone structure for HopeNet to capture features in the given images. The three fully connected layers that the authors of HopeNet created could predict the Euler angles based on the learned features from ResNet.

3.5 Gradient Descent

A significant optimization algorithm for machine learning is Gradient Descent. Gradient Descent is used to find the parameter values of a function $f(X)$ which minimizes a loss function. When the parameters can't be calculated analytically with the help of functions such as linear algebra, it has to be found using an optimization algorithm. Since a Neural Network is often seen as a black box, it is necessary to implement an algorithm that optimizes the NN without analyzing the subject of the given task.

Gradient descent is an iterative method, starting with a set of random values for the model parameters (weights), that gradually improves them such that the model performs better on the given task. To improve the weights, the value of a loss function must be calculated and reduced as much as possible during training. Repeating this multiple times will continue to minimize the loss function, resulting in a more optimized function and higher model accuracy for the given task [4].

The following loss functions are be used to train the Neural Network used in this master project.

3.5.1 Mean Squared Error (MSE)

When we have a regression model with continuous labels, we want to better predict the ground truth values in the training data. We achieve this by e.g., using a loss function called Mean Squared Error, which is defined as [18]:

$$loss\_MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

where n is the number of ground truth values, y_i is the i^{th} ground truth value, and $\hat{y}_i$ is the i^{th} predicted value. Using this, we will, over many iterations, achieve better prediction values with minor errors on average for all of our training data input.

3.5.2 Categorical Cross Entropy Loss (CCEL)

Consider a multi-class classification problem, which we want to solve with a Neural Network. To compare the output (logits) of the network to the one-hot encoded label, which defines the correct class, we have to transform the logits. This is typically done by calculating the softmax of the logits and then calculate loss between the probabilities from the softmax and the one-hot encoded label of the ground truth.

If we use MSE for this problem, there can be situations where the gradient will be small even though the predicted probability for the correct class was far from the ground truth. An example of this is when the model predicts probabilities close to zero for all classes, which will result in a small error because only one of the classes were incorrectly predicted and MSE will return the average error.

For this reason, we use Categorical Cross-Entropy Loss for problems such as this since it penalizes more than MSE. Cross-Entropy is defined as [12]:

$$loss\_CEL = -\sum_{i=1}^{c} y_i \cdot log(\hat{y}_i)$$

where c is the number of classes, y_i is the ground truth label for the i^{th} class, and $\hat{y}_i$ is the predicted softmax probability for the i^{th} class.

Categorical Cross-Entropy Loss is defined identically as Cross-Entropy. The only difference is that Categorical Cross-Entropy has one-hot encoded ground truth labels. In contrast, Cross-Entropy has standard numerical ground truth labels.

While Cross-Entropy works well for multi-class classification problems and often better than MSE, MSE works better for regression problems with continuous labels, which will also be seen later in this report.

Method

This chapter describes the implementation of the previously mentioned theory and how it works together to form a Deep Neural Network. The developed source-code is available on Github repository: https://github.itu.dk/beao/Head-pose-estimation-using-Deep-Learning.git

4.1 Data collection

This master project aims to create a Neural Network, which can achieve similar results in predicting the head pose as a 2018 iPad Pro with a TrueDepth sensor. Thus, Apple ARKit API provides the data used to train, evaluate, and test the model. The perfect model has a prediction error of 0° after training because the data from the 2018 iPad Pro is considered the *ground truth*.

The TrueDepth sensor on the 2018 iPad Pro contains an infrared dot projector, which projects 30,000 dots onto the user's face and creates a digital model of the face [2]. The Apple Neural Engine can then transform the depth map and infrared image of the face into a mathematical representation from which the head pose can be extracted [3].

Apple provides an app called *Reality Composer*, which we use to record videos and augmented reality ARKit data. The created dataset contains approximately 40 videos of resolution 1440×1080 and in different lighting conditions. *XCode* allows us to replay the dataset, and retrieves the face data from each frame, saves the frame in full resolution, and adds the Euler angles and 3D position of the face mid-point to a CSV file. The trained model uses approximately 50,000 frames with corresponding Euler angles from separate CSV files. The entire dataset used for training/testing contains head poses in the range of:

- Pitch: $-72°$ to $+45°$
- Yaw: $-76°$ to $+77°$
- Roll: $-83°$ to $+77°$

A Dlib CNN face detector was responsible for cropping the user's face in each analyzed frame. On the other hand, a face tracker allows us to make the process faster and still capture faces in extreme poses that would be too difficult for the face detector.

The benefit of training on a dataset with actual data instead of synthetic data is that the model should theoretically perform better on unseen actual data. Therefore, it is the same type of data that the trained model will predict in real-world use cases.

The biggest challenge of creating a new dataset is to control the angles of the head poses and replicating the many edge cases covered in current synthetic datasets with actual data, mainly related to different lighting conditions. It is easy for the model to adapt to the given lighting conditions and then lose performance when introduced to a new lighting condition that it has not seen during training. For this reason, the model uses two distinct datasets—one dataset for standard train and validation steps and a second one for testing the model performance. The second dataset contains seen and unseen lighting conditions.

A second challenge is that the dataset will only consist of one individual's head in different lighting conditions and with different rotations. It could theoretically cause the model only to learn how to solve the given task with one individual. However, it is not in the scope of this report to create a custom dataset with many different individuals. Therefore, the results in the next chapter show the potential of this approach. It does not set definite statements for real-world applications due to only having one individual in the used dataset.

4.2 Dlib face detector and face tracker

The pre-trained Dlib CNN (MMOD) face detector retrieves the bounding box around the face in the image of the entire dataset [11]. This detector performs well on detecting faces in extreme poses and different lighting

conditions while being trained with a small dataset. MMOD stands for Max-Margin Object Detection and works by maximizing the margin between the distances of the correct classes (faces) and incorrect classes (anything other than a face).

Suppose the face detector could not find a face. In that case, it discards the current frame and corresponding Euler angles in training/testing. The image is cropped loosely around the detected face's bounding box to keep most of the head in the image. Therefore, the cropping is used to remove most of the background since we are not interested in analyzing that when trying to predict the Euler angles of the head pose.

The Dlib face detector plays an essential role in cropping the images. When attempting to do this while replaying in *Xcode*, the renderer either skipped too many frames when also detecting faces or had inconsistent croppings. This method also prevents changing the croppings after getting the full resolution images and Euler angle label. The entire replay step would have to be re-executed again.

Since the dataset consists of the full resolution images, it could be part of the future work to crop the faces with a different face detector (e.g., a deep learning-based face detector). It might slightly reduce the number of undetected faces and, therefore, keep some extra frames for training/testing.

Due to the limited resources, we use the Dlib object-tracker [6] to track the bounding box of the detected face from frame to frame. It decreases the time to crop all images in the dataset to the face. Otherwise, it takes too long to run the CNN face detector on every image when considering that the dataset was constantly updated and modified during the project. Another reason for using the object-tracker is that detectors are likely to miss some frames, such as images with motion blur. The object-tracker will most likely still include these images.

4.3 Image augmentation

Similarly to HopeNet [19], the image augmentation randomly blurs, randomly flips the image and Euler angles, resizes, and normalizes the images during training before passing them to the model. The image augmentation also randomly downsamples and upsamples the dataset to make it more diverse regarding image resolution. The model should not react poorly to new image resolutions and should not rely too much on detailed features in the image because of this. We randomly downsample each image during training by one of the following factors 1, 6, 11, or 16.

As mentioned before, since the experiments in this report use a custom dataset, the data augmentation methods prevent the model from overfitting on the given lighting conditions. Besides adding more data to the dataset with different lighting conditions, we added lighting noise to the images before normalization as proposed in the AlexNet paper [13].

All these augmentations make sure that the model does not perform poorly on unseen data. It is crucial to augment the data when using a small dataset in a deep learning context, like the one used in this master project.

4.4 Model

The starting point for this master project is the model described in [19]. It is a mixture of binned pose classification and pose prediction. The authors explain how this performs better than only predicting the angles and only using the regression loss. Their final model had a Cross-Entropy Loss function for the binned pose classification and a Mean Squared Error Loss function for the predicted Euler angles. When calculating the Cross-Entropy Loss, each fully connected layer outputs 66 bins compared with the ground truth bins. The bins give support to calculate the corresponding angle, which is the angle prediction used for the regression loss.

The amount of bins is 66 because only images with every Euler angle between $-99°$ and $+99°$ are used for training and evaluation (like the authors of HopeNet did). The angles can then be binned with a step of 3 such that three different angles can end up in the same bin. It makes the network capable of classifying the neighborhood of the pose and makes it easier for the model to predict the exact angle. Since the classification has made the range smaller, the prediction can be in-between.

The purpose of the regression loss is to improve fine-grained predictions. The model then gets better at predicting the angle in the classified range (bin).

4.5 K-Fold Cross Validation

The final experiment uses K-Fold Cross-Validation to ensure that the given model performs well on unseen data and does not suffer from overfitting [15]. K is constant throughout all runs and is set to 10 because

this value of k generally results in low bias and modest variance for training and testing.

Each fold will produce a Mean Absolute Error on the current test set for each Euler angle. It evaluates the mean of these three errors to show how well this fold performs in general for all angles.

We shuffled the entire dataset before random distribution. If all folds can't contain precisely the same number of elements, then each fold from 0-9 will get an extra component until there are no more left. It means that some of the folds will have one more element than the other folds.

Experiments

This chapter describes the experiments of this master project, the details of their executions, and the results they produced. Please find the detailed descriptions of the model structure in the previous chapters.

5.1 Experiment analysis

The experiments have been executed with a single run of 10-fold cross-validation due to limited resources. Each fold has produced the Mean Absolute Error for all angles and only the results of the best performing were used for further analysis.

The most interesting aspects to analyze are:

- Comparing the error difference between using a pre-trained ResNet structure and a non-pre-trained ResNet Structure
- How the best fold performs on an unseen final test dataset
- How the best fold performs on the final test dataset, where all the images have been downsampled

5.2 Euler angle prediction

Both models will perform head pose estimation on all images in the final test dataset by predicting the Euler angles. Each prediction will be measured and evaluated on the Mean Absolute Error from the ground truth Euler angles, provided by the 2018 iPad Pro.

5.2.1 Hypothesis

Due to the impressive results produced by HopeNet [19], we believe that it will produce similar results on the custom dataset obtained with a 2018 iPad Pro.

5.2.2 Results

As shown in Figure 5.2.1, the Mean Absolute Error produced by each model on several folds, where the leftmost plot is the mean of Mean Absolute Errors and the remaining three plots only show the results on pitch, yaw, and roll, respectively. In Figure 5.2.2, the loss produced during training of the best performing folds for each model are presented and in Table 5.2.1, the results produced by the best folds for each model are presented.

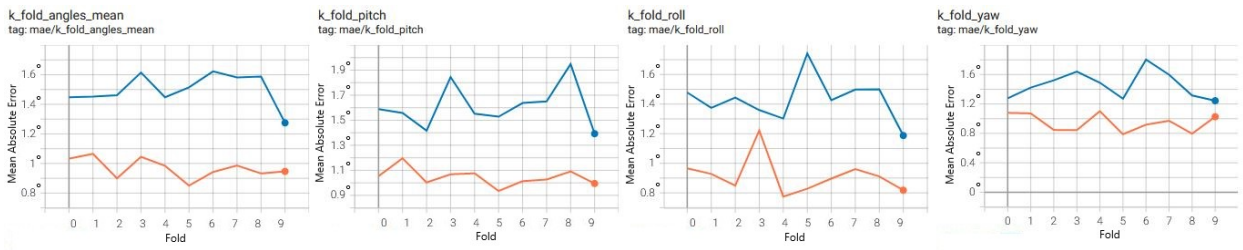


Figure 5.2.1: *10-fold cross-validation with pre-trained ResNet on ImageNet (orange line) and another without a pre-trained ResNet (blue line). Both lines are plotted based on the Mean Absolute Error for the given Euler Angle. The plot on the far left is the mean of Mean Absolute Errors, followed by pitch, roll, and yaw*

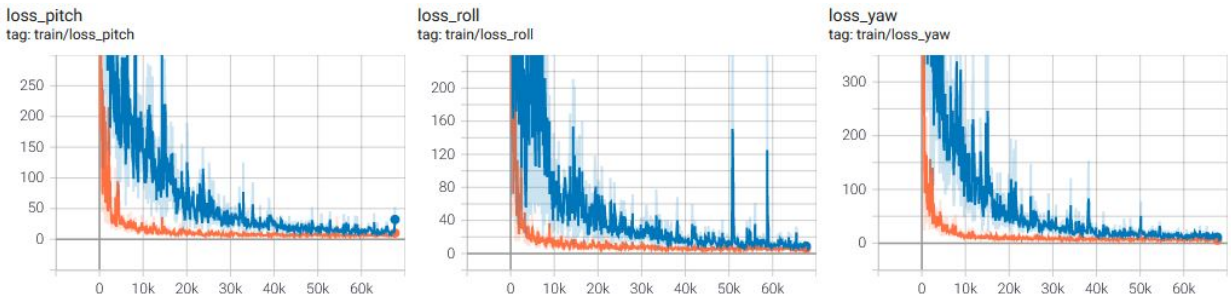


Figure 5.2.2: *The loss of the best run from both 10-fold cross-validations, evaluated based on the mean of all angles*

Model	Pitch	Yaw	Roll	MAE
pre-trained ResNet	2.971°	2.589°	1.619°	2.393°
Not pre-trained ResNet	4.167°	3.717°	2.855°	3.579°

Table 5.2.1: *Mean Absolute Errors on each Euler angle when testing on the final test dataset*

5.2.3 Discussion

As shown in Figure 5.2.1, there is a slight difference between the two different runs. The model with a pre-trained ResNet gets a lower Mean Absolute Error on the testing data for every fold and a lower loss during training when comparing the best fold of each model. The final model in [19] also used a pre-trained ResNet on ImageNet to achieve the best performance. Therefore, initializing the model with pre-trained parameters can speed up convergence since the starting point of the model might be closer to the most optimal solution instead of using random initialization of the weights [9], which in this case improves the performance for the final task. We also do not freeze ResNet after loading the pre-trained parameters. We want the model to fine-tune the weights in the entire model (ResNet and fully connected layers) during training on the custom dataset that only contains different head poses. We took the best-performing fold of each model and tested it on our final test dataset consisting of approximately 10,000 frames from 9 videos, which the model has not seen during training. The best fold on the model with a pre-trained ResNet is the 5^{th} fold, and the best fold on the model without a pre-trained ResNet is the 9^{th} fold. The final test dataset contains head poses in the range of:

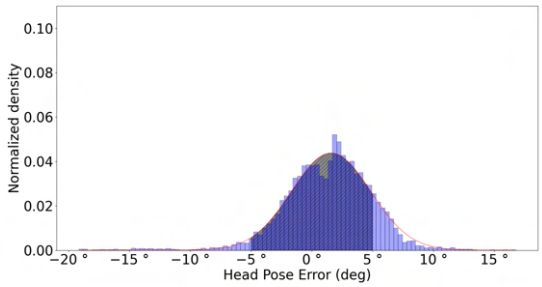
- Pitch: $-73°$ to $+40°$
- Yaw: $-73°$ to $+72°$
- Roll: $-50°$ to $+50°$

The HopeNet paper used the BIWI dataset [8] with a pitch range of $\pm 60°$, a yaw range of $\pm 75°$, and a roll range of $\pm 50°$, as part of their testing. We could not replicate the pitch range when collecting data with the 2018 iPad Pro since it was unable to capture head poses above 40° and stopped tracking when going beyond that.

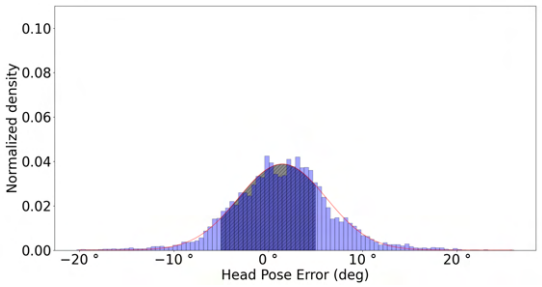
The results when running the models on the final test dataset are presented in Table 5.2.1, which again shows the increase in performance obtained from using a pre-trained ResNet as a backbone structure instead of not using a pre-trained ResNet, because it has to learn the most optimal weights from scratch. Both models use a regression coefficient of 2.0 and L2 regularization (weight decay) with coefficient 0.6 because preliminary testing showed that these values produced the best results with the given final test dataset.

Compared to other research studies [19, 22, 5], this Mean Absolute Error is quite good. So, it would be interesting to see how many of the frames have a prediction error close to the mean. We would expect to approximate the Gaussian distribution of all predictions made on the entire final test dataset.

Figure 5.2.3, 5.2.4, and 5.2.5 shows the Gaussian distribution of the predictions made on each Euler angle by each model, where the highlighted part of the plot are the predictions within $\pm 5°$ of error because in [16] they proposed that an accurate head pose estimator should as a maximum produce a Mean Absolute Error of 5°:

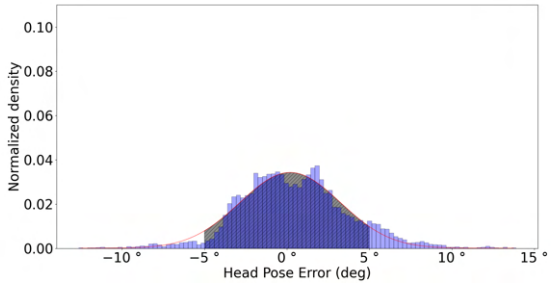


(a) *Pre-trained ResNet model.*
Mean: 1.152°, *std:* 3.690°

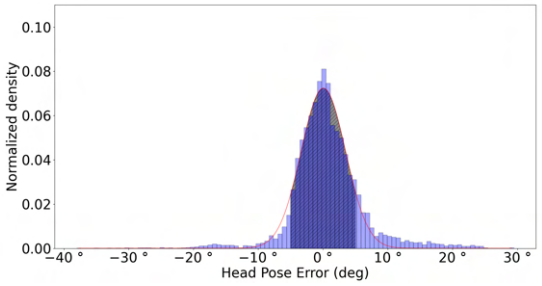


(b) *Non-pre-trained ResNet model.*
Mean: 1.278°, *std:* 5.269°

Figure 5.2.3: *Pitch Gaussian distribution with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the pitch angle on all frames in the final test dataset*

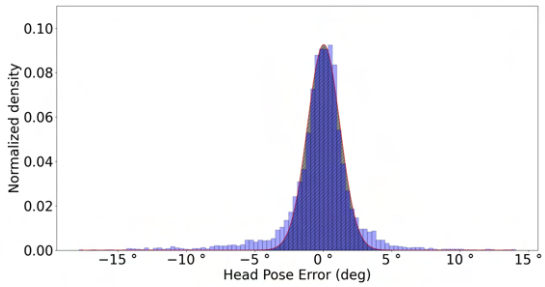


(a) *Pre-trained ResNet model.*
Mean: 0.340°, *std:* 3.319°

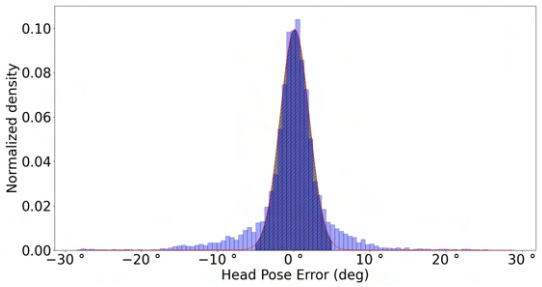


(b) *Non-pre-trained ResNet model.*
Mean: 0.372°, *std:* 5.533°

Figure 5.2.4: *Yaw Gaussian distribution with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the yaw angle on all frames in the final test dataset*



(a) *Pre-trained ResNet model.*
Mean: −0.385°, *std:* 2.637°



(b) *Non-pre-trained ResNet model.*
Mean: −0.406°, *std:* 4.607°

Figure 5.2.5: *Roll Gaussian distribution with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the roll angle on all frames in the final test dataset*

As shown in Figure 5.2.3, 5.2.4, and 5.2.5, the pitch prediction errors for both models seem to approximate the Gaussian distribution better than the other Euler angle prediction errors, which deviate more from the Gaussian distribution.

Another thing to notice is that the standard deviation slightly increases from the model with a pre-trained ResNet to the model without a pre-trained ResNet. This indicates that the model without a pre-trained ResNet produces slightly more of the higher error predictions, which is especially visible on the Gaussian distribution of the yaw and roll error. The yaw prediction error has more centered predictions, but the range of errors has increased from approximately ±15° on the model with a pre-trained ResNet to approximately

be between $-40°$ up to $+30°$ of error. The roll prediction error seems to have the same distribution as the model with a pre-trained ResNet. However, the range of errors ranges from approximately between $-17°$ and $+15°$ to be approximately between $\pm 30°$ of error.

Table 5.2.2 shows the percentage of predictions within $\pm 5°$ error for each Euler angle and model:

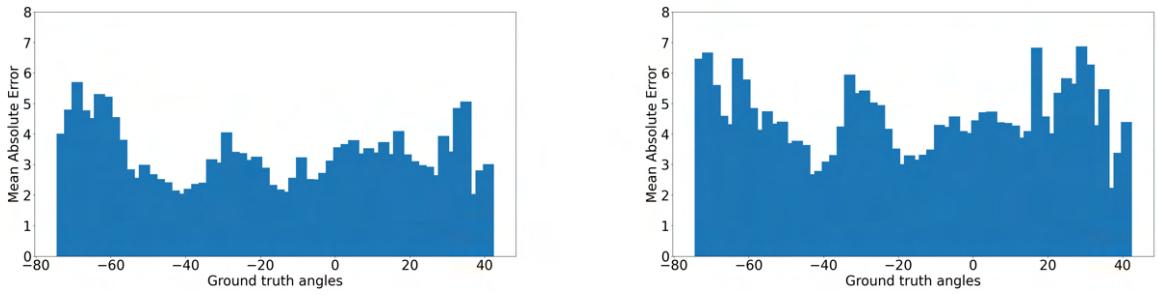
Model	Pre-trained ResNet	Not pre-trained ResNet
Pitch	80.36%	64.32%
Yaw	86.59%	63.26%
Roll	93.92%	72.03%

Table 5.2.2: *Percentage of predictions within $\pm 5°$ error*

Both Figure 5.2.3, 5.2.4, and 5.2.5 and Table 5.2.2 show that the model with a pre-trained ResNet makes more predictions within $\pm 5°$ error than the model without a pre-trained ResNet and also that both models perform best on the roll angle. Naturally, the amount of predictions that fall within $\pm 5°$ error is directly connected to the mean and standard deviation of each Gaussian distribution, which is observed in Figure 5.2.3, 5.2.4, and 5.2.5 where the distributions with a smaller mean and standard deviation produce more predictions within $\pm 5°$ error.

To gain better insight into which head poses cause the higher error for the given Euler angle, we will create a different histogram that shows the head pose on the X-axis and the Mean Absolute Error for each head pose on the Y-axis. Ideally, the histogram should be flat with an error equal to the mean. The Mean Absolute Error for each head pose would then be the same, which would mean that the model performs similarly for all head poses. However, we expect that the histogram will form a parabola centered at $0°$, which means that the model performs best on standard head poses and worse on extreme head poses because extreme head poses are known to be more difficult to predict correctly.

Figure 5.2.6 shows that it does not follow the expected shape of a parabola. The Mean Absolute Error seems to be higher when nearing the minimum head pose of $-73°$ and on head poses just below the maximum of $+40°$. However, there are still some spikes in between. This holds for both models, and it seems that they have a somewhat similar distribution, even though the model without a pre-trained ResNet generally has a higher Mean Absolute Error on the same head poses.

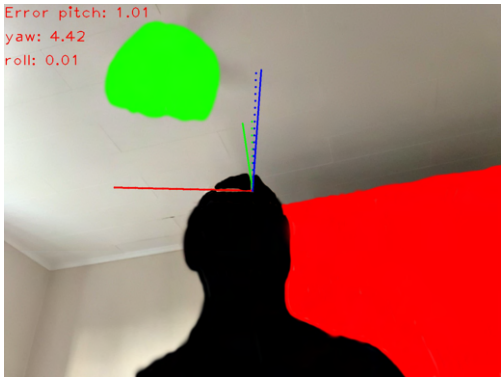


(a) *Pre-trained ResNet model* **(b)** *Non-pre-trained ResNet model*

Figure 5.2.6: *Pitch MAE grouped on similar pose ranges with pre-trained ResNet model and non-pre-trained ResNet model, both produced by predicting the pitch angle on all frames in the final test dataset*

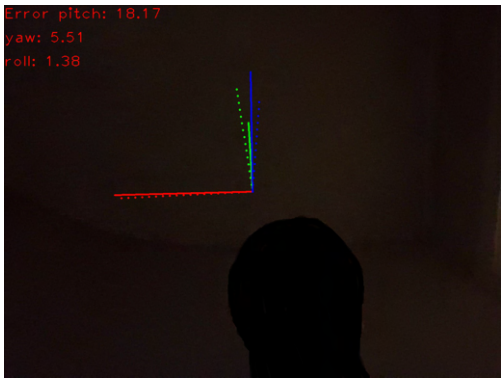
Figure 5.2.7 shows some examples of head poses with a pitch angle of around $-60°$, where the complete (connected) lines represent the ground truth from the 2018 iPad Pro recording and the dotted lines represent estimated data. The images show three separate axes, which represents the head rotation, but they do not show the translation of the head and the axes are therefore placed at the principal point of the image. The red line represents the rotated pitch angle, the green line represents the rotated yaw angle, and the blue line represents the rotated roll angle. Figures 5.2.7 (a) and (b) show that there does not seem to be a very big difference between the two models, but the actual error shows that Figure 5.2.7 (a) is $5°$ from the ground truth on the pitch angle and Figure 5.2.7 (b) is only $1°$ off in pitch but $4.4°$ off on the yaw angle. However, in more tough conditions such as Figures 5.2.7 (c) and (d), the model with a pre-trained ResNet has an error on the pitch angle of $9.9°$ and this almost doubles when using the model without a pre-trained ResNet to $18.1°$.

(a) *Head pose 1 - Pre-trained ResNet*



(b) *Head pose 1 - Non-pre-trained ResNet*

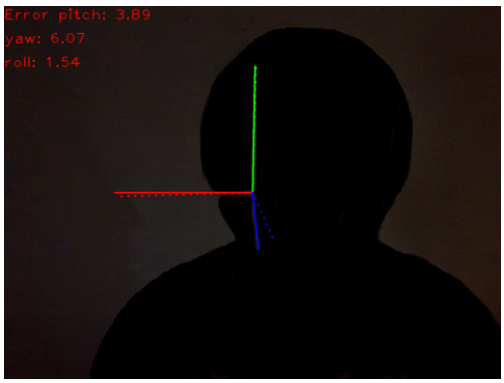
(c) *Head pose 2 - Pre-trained ResNet*



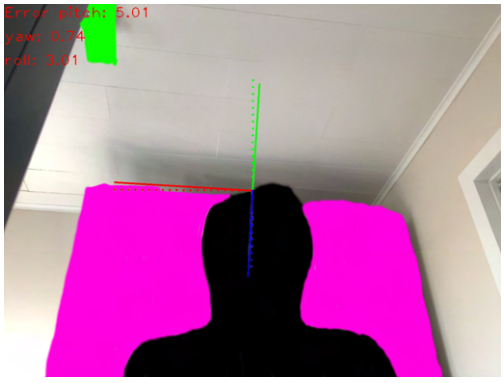
(d) *Head pose 2 - Non-pre-trained ResNet*

Figure 5.2.7: *Images where head pose is around* $-60°$ *on pitch. The normal complete lines represent the ground truth head pose and the dotted lines represent the prediction of the model*

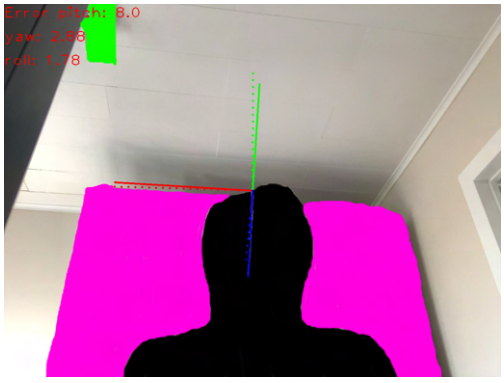
(a) *Head pose 3 - Pre-trained ResNet*



(b) *Head pose 3 - Non-pre-trained ResNet*



(c) *Head pose 4 - Pre-trained ResNet*

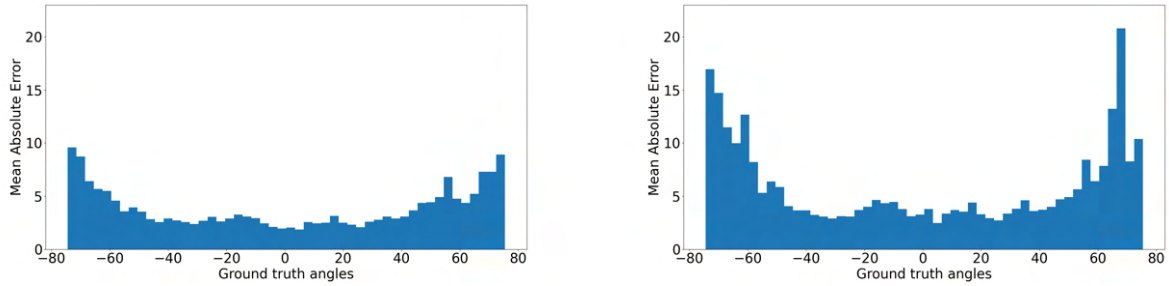


(d) *Head pose 4 - Non-pre-trained ResNet*

Figure 5.2.8: *Images where head pose is* $+20°$ *or above on pitch*

When nearing the +40°, the same error as for angles around −60° is observed. However, we expect that the model will increase in error when going beyond the +40° similarly to what we saw when nearing the minimum angle. Again, the model with the pre-trained ResNet outperforms the model without it by producing a pitch error of 1° as shown in Figure 5.2.8 (a), and an error of 5° as shown in Figure 5.2.8 (c). While the model without a pre-trained ResNet gets a pitch error of 3.8° as shown in Figure 5.2.8 (b), and an error of 8° as shown in Figure 5.2.8 (d). The errors are not necessarily higher on the angle that is on the end of the spectrum. As shown in Figures 5.2.8 (a) and (b), both models produce a higher error on the yaw angle (5.1° and 6° error) even though the head pose is at a higher pitch angle and the yaw angle has almost no rotation.

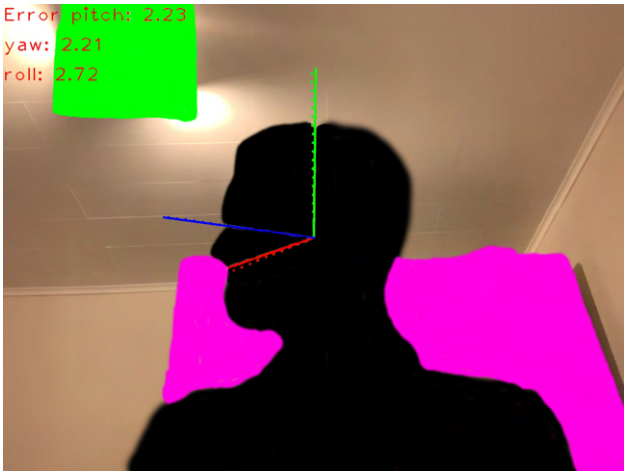
Figure 5.2.9 shows the expected distribution that approximates the shape of a parabola for both models. It shows that head poses near the minimum of −73° or the maximum of +72° have a higher prediction error on average than head poses in a smaller range. It is right around −60° and +60°, the prediction errors start to increase and then keep increasing until the minimum and maximum head poses are reached. It can also be observed that the model without a pre-trained ResNet has increased in Mean Absolute Error for all head poses and even more for the head pose around +65°.



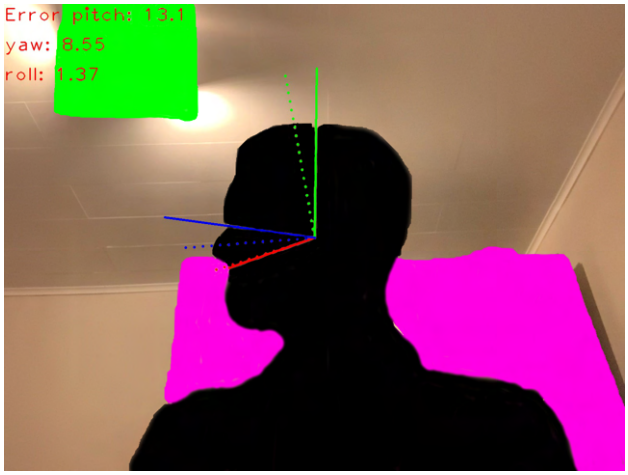
(a) *Pre-trained ResNet model* **(b)** *Non-pre-trained ResNet model*

Figure 5.2.9: *Yaw MAE grouped on similar pose ranges with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the yaw angle on all frames in the final test dataset*

Figures 5.2.10 (a) and (b) show head poses around −60° yaw. They show that the model with a pre-trained ResNet gets a relatively low error by only having 2.2° of error on pitch and yaw and 2.7° on roll, while the model without a pre-trained ResNet performs much worse, increasing the error to 13.1° on pitch and 8.5° on yaw. When predicting on more extreme poses around −70° yaw, as shown in Figures 5.2.10 (c) and (d), the error increases for both models. The model with a pre-trained ResNet gets a higher yaw error of 9°, and the model without a pre-trained ResNet gets an even higher yaw error of 16.7°. This is shown in Figure 5.2.9, where the Mean Absolute Error increased when getting closer to the most extreme poses.



(a) *Head pose 5 - Pre-trained ResNet*



(b) *Head pose 5 - Non-pre-trained ResNet*

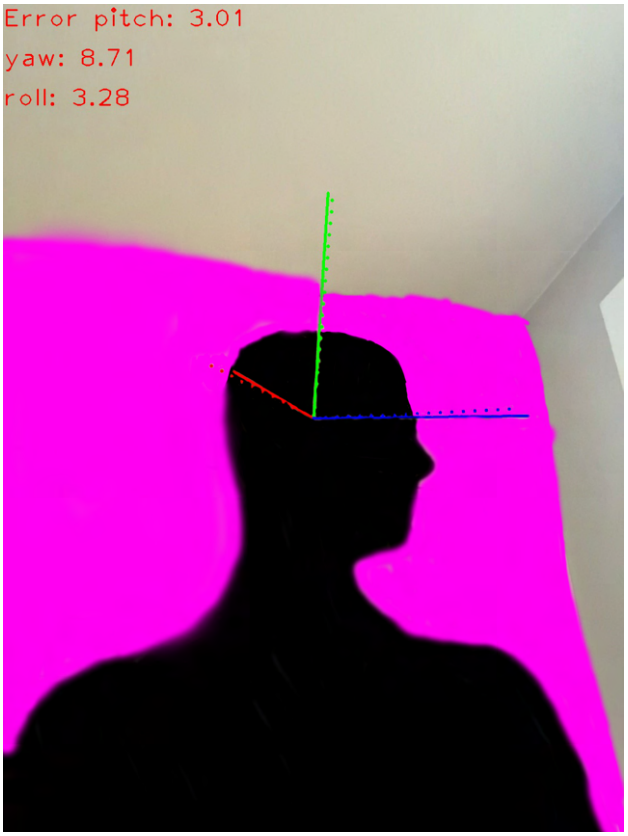
(c) *Head pose 6 - Pre-trained ResNet*

(d) *Head pose 6 - Non-pre-trained ResNet*

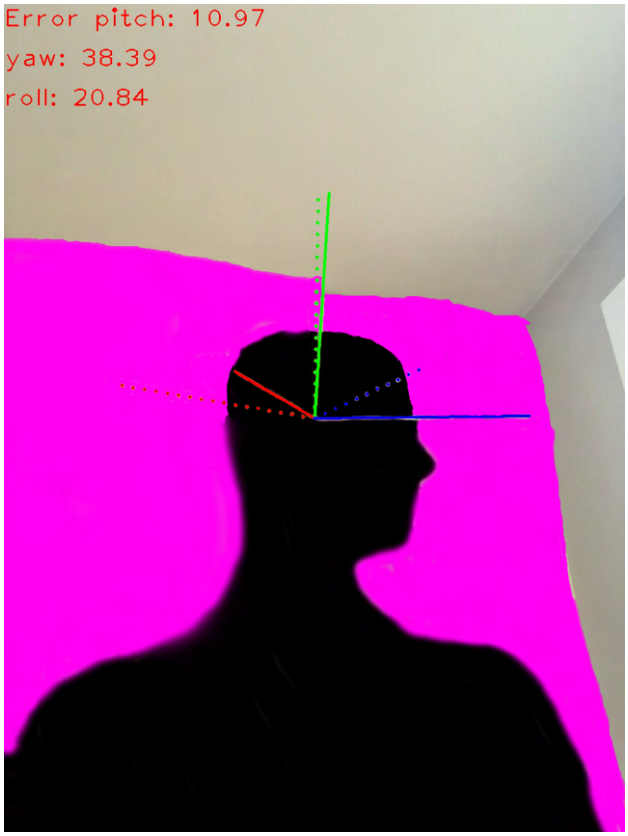
Figure 5.2.10: *Images where head pose is* $-60°$ *or below on Yaw*

Figures 5.2.11 (a) and (b) show head poses around $+65°$ on the yaw angle also seems to be a difficult task for both models since the model with a pre-trained ResNet produces a yaw error of 8.7°. The model without a pre-trained ResNet produces a yaw error of 38.3°. Figure 5.2.11 (b) shows that because the dotted lines deviate a lot from the ground truth lines and it is head poses like this that make the Mean Absolute Error increase in Figure 5.2.9.

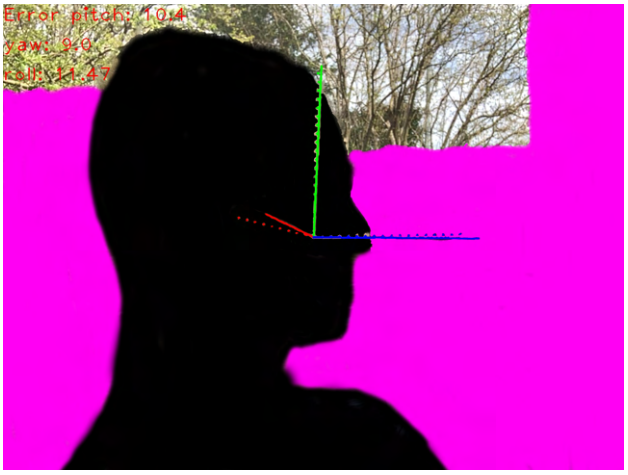
Figures 5.2.11 (c) and (d) show slightly more extremes head poses around $+70°$. However, in these cases, the two models perform similarly on the yaw angle and differently on the other angles. The model with a pre-trained ResNet gets an error of 10.4°, 9°, and 11.4° on pitch, yaw, and roll, respectively. On the other hand, the model without a pre-trained ResNet gets an error of 5.6°, 10.6°, and 7.8° on pitch, yaw, and roll, respectively. So, this shows that in this case, the model with a pre-trained ResNet has a harder time at predicting all angles correctly. In contrast, the model without a pre-trained ResNet gets a similar error on yaw but has learned to keep the other angles more stable. This example again shows that the error for more extreme head poses are high in general on both models but also that the model without a pre-trained performs even worse at around $+65°$ as shown in Figure 5.2.9.



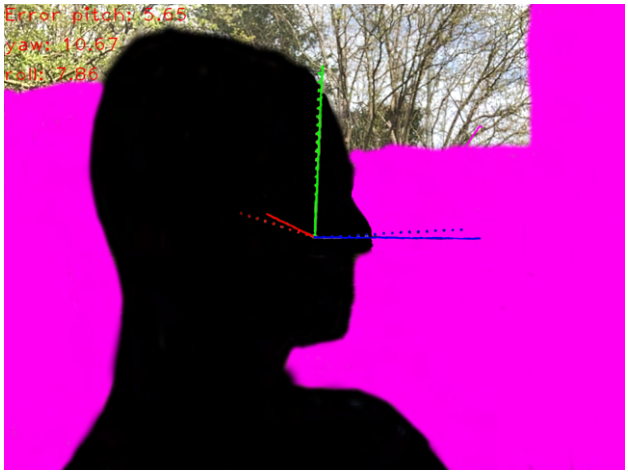
(a) *Head pose 7 - Pre-trained ResNet*



(b) *Head pose 7 - Non-pre-trained ResNet*



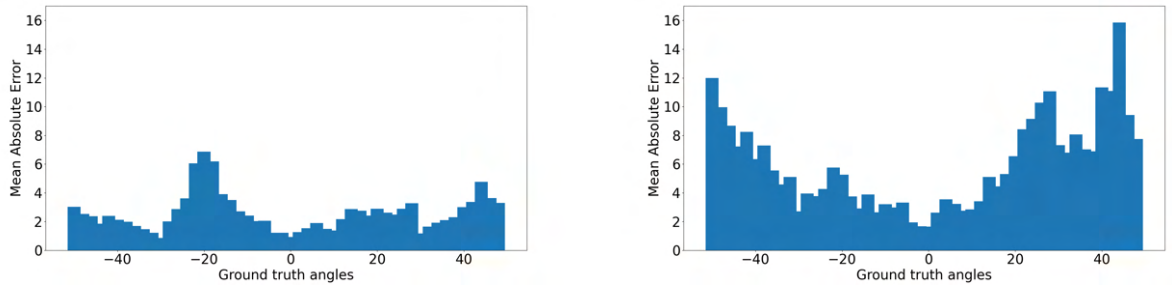
(c) *Head pose 8 - Pre-trained ResNet*



(d) *Head pose 8 - Non-pre-trained ResNet*

Figure 5.2.11: *Images where head pose is* $+60°$ *or above on Yaw*

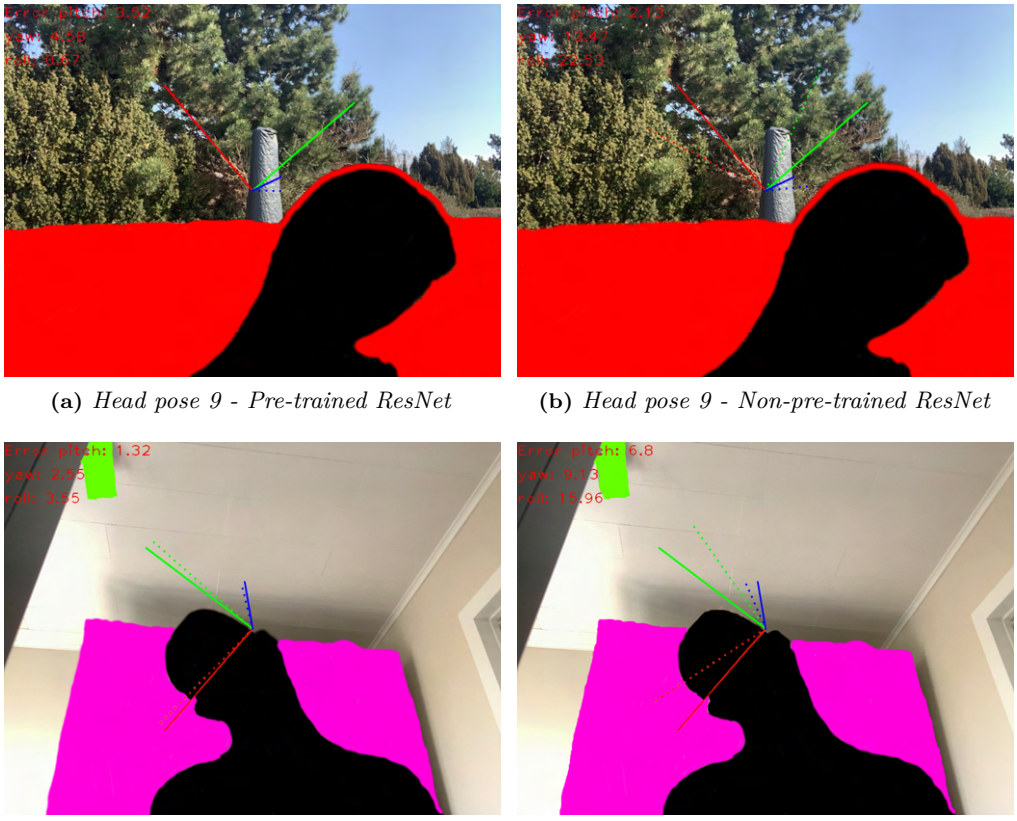
Figure 5.2.12 shows a different pattern than for the two other Euler angles. With the model that uses a pre-trained ResNet, the highest error is no longer at the maximum and minimum head poses. It is around $-20°$, and the model that does not use a pre-trained ResNet produces close to what we expected with a higher error when nearing the maximum and minimum head poses and smaller error around $0°$ head poses. However, the model without a pre-trained ResNet model increases in Mean Absolute Error for all head poses and has almost the same error around $-20°$ as the model with a pre-trained ResNet.



(a) *Pre-trained ResNet model* **(b)** *Non-pre-trained ResNet model*

Figure 5.2.12: *Roll MAE grouped on similar pose ranges with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the roll angle on all frames in the final test dataset*

The two examples in Figures 5.2.13 (a) and (b) show the most extreme poses in the dataset for the roll angle. Figure 5.2.12 shows that the model without a pre-trained ResNet produces the highest errors on these extremes, which is also observed in the example images. Figure 5.2.13 (b) has a very high error of 22.5° on roll, while Figure 5.2.13 (a) has an error of 0.6° on roll. But as mentioned before, the highest error is not necessarily on the extreme angle since the model with a pre-trained ResNet produces a pitch and yaw error of 3.5° and 4.5° in Figure 5.2.13 (a). Figure 5.2.13 (d) shows that the error for the model without a pre-trained ResNet on extreme roll head poses is quite high at 15.9° on roll and that the model with a pre-trained ResNet is better at estimating the correct angles by only producing 3.5° error on roll and reflects the observation made in Figure 5.2.12.



(a) *Head pose 9 - Pre-trained ResNet* **(b)** *Head pose 9 - Non-pre-trained ResNet*

(c) *Head pose 10 - Pre-trained ResNet* **(d)** *Head pose 10 - Non-pre-trained ResNet*

Figure 5.2.13: *Figures (a) and (b) have a head pose around −50° and figures (c) and (d) have a head pose around +50° on the roll angle*

As mentioned before, the model with a pre-trained ResNet does not show the same parabola shape on its error distribution over different roll head pose angles. It shows an increased error for roll angles around −20°, which Figure 5.2.14 is an example of, where we would expect the highest error to be on the yaw angle since the head pose is mostly rotated around the yaw axis. However, the error for this image is 12.7°, 8.8°,

and 13° for pitch, yaw, and roll, respectively, using the model with a pre-trained ResNet. The head poses also have a pitch rotation of −30° and a yaw rotation of 60°, which seems to make it more difficult for the models to predict each of the angles with low error because the angle mostly in focus (yaw) is not the only rotation that is happening.

(a) *Head pose 11 - Pre-trained ResNet* **(b)** *Head pose 11 - Non-pre-trained ResNet*

Figure 5.2.14: *An example of high error at −20° roll*

5.2.4 The effect of low-resolution

This experiment aims to determine how robust the models are when given low-resolution images. As mentioned in Section 4.3, the training process has used downsampled images with a factor of 1, 6, 11, and 16. This analysis downsample all images in the final test dataset with a factor of 16 to discover how the trained models react to a familiar downsampling factor on all images and with a new factor of 21 to discover how the models react to even lower resolution images, which they have not seen during training. Due to the reduced image size, fewer features are available for the models to analyze. So, it is expected that the models will decrease in accuracy from not using downsampling to downsampling by a factor of 16 and decrease even more when downsampling by a factor of 21.

Tables 5.2.3 and 5.2.4 present the results of the model with a pre-trained ResNet and the model without a pre-trained ResNet, when downsampling all images in the final test dataset by a factor of 16 and 21.

Model	Pitch	Yaw	Roll	MAE
Pre-trained ResNet	2.909°	2.670°	1.669°	2.416°
Not pre-trained ResNet	4.470°	4.085°	2.977°	3.844°

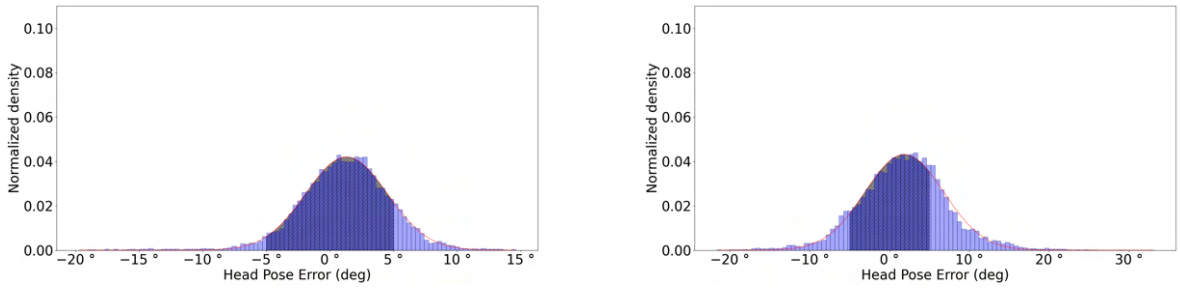
Table 5.2.3: *Mean Absolute Errors on each Euler angle when testing on the final test dataset that has been downscaled by a factor of 16. Has an MAE increase of 0.023 on the pre-trained ResNet model and 0.265 on the Not pre-trained ResNet model*

Model	Pitch	Yaw	Roll	MAE
pre-trained ResNet	2.964°	2.727°	1.800°	2.497°
Not pre-trained ResNet	4.838°	4.285°	3.112°	4.078°

Table 5.2.4: *Mean Absolute Errors on each Euler angle when testing on the final test dataset that has been downsampled by a factor of 21. Has an MAE increase of 0.104 on the pre-trained ResNet model and 0.499 on the Not pre-trained ResNet model*

Tables 5.2.3 and 5.2.4 show that the error slightly increases on the model with a pre-trained ResNet when downsampling all images by a factor of 16 and 21, where the factor of 21 produces a slightly higher error than 16. They also show that the model without a pre-trained ResNet increases more in Mean Absolute Error and generally on all the angles. In contrast, the model with a pre-trained ResNet only increases on yaw and roll but surprisingly decreases slightly on pitch.

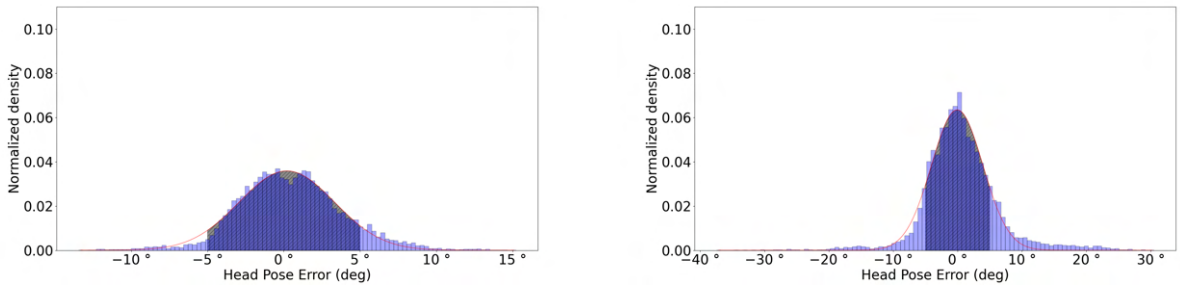
Figures 5.2.15, 5.2.16, and 5.2.17 show the Gaussian distribution of the prediction errors produced by the model with a pre-trained ResNet and the model without a pre-trained ResNet, when downsampling all images in the final test dataset by a factor of 16.



(a) *Pre-trained ResNet model. Mean:* 0.970° *and std:* 3.684°

(b) *Non-pre-trained ResNet model. Mean:* 1.520° *and std:* 5.571°

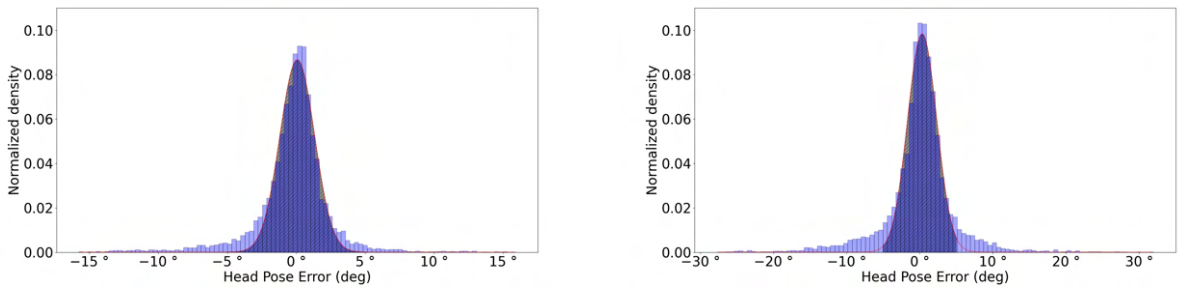
Figure 5.2.15: *Pitch Gaussian distribution with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the pitch angle on all frames in the final test dataset downsampled by a factor of 16*



(a) *Pre-trained ResNet model. Mean:* 0.308° *and std:* 3.437°

(b) *Non-pre-trained ResNet model. Mean:* 0.223° *and std:* 5.906°

Figure 5.2.16: *Yaw Gaussian distribution with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the yaw angle on all frames in the final test dataset downsampled by a factor of 16*



(a) *Pre-trained ResNet model. Mean:* −0.393° *and std:* 2.645°

(b) *Non-pre-trained ResNet model. Mean:* −0.355° *and std:* 4.750°

Figure 5.2.17: *Roll Gaussian distribution with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the roll angle on all frames in the final test dataset downsampled by a factor of 16*

Figures 5.2.15, 5.2.16, and 5.2.17 show that there is a similar pattern as with the increase in Mean Absolute Error from using images that haven't been downsampled. The Gaussian distribution has fewer outliers on the model with a pre-trained ResNet, which did not increase in pitch Mean Absolute Error. It also has a slightly lower mean and standard deviation. For the other two Euler angles, the Mean Absolute Error increased and that is primarily observed on the standard deviation, which has increased slightly from the Gaussian distribution on the final test dataset without downsampling.

The model without a pre-trained ResNet performed worse on all Euler angles. As a result, it got a higher standard deviation on all three Gaussian distributions as well.

With both models, it also appears that a slight difference in Mean Absolute Error on a particular Euler angle results in a slight change in standard deviation for the same angle. In contrast, a more significant difference in Mean Absolute Error for a specific Euler angle results in a more substantial change in standard deviation.

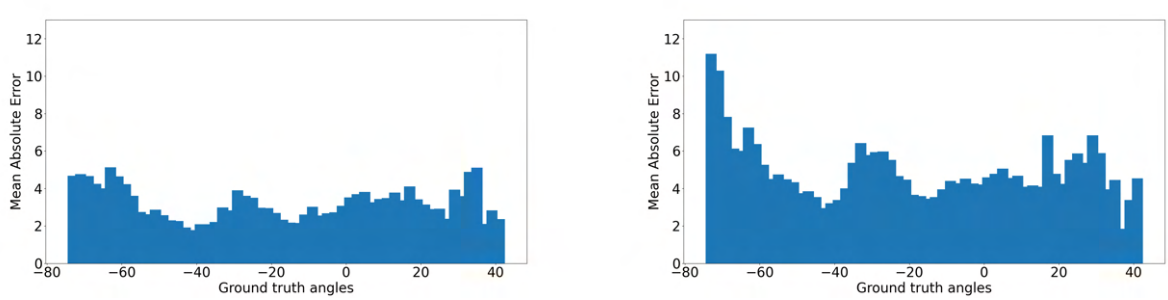
Table 5.2.5 shows the percentage of predictions within ±5° error for each Euler angle and model when the images are downsampled by a factor of 16.

Model	Pre-trained ResNet	Not pre-trained ResNet
Pitch	81.03%	61.29%
Yaw	85.25%	60.23%
Roll	93.84%	70.61%

Table 5.2.5: *Percentage of predictions, on downsampled images by a factor of 16, within ±5° error*

The difference between Table 5.2.5 and Table 5.2.2 without downsampling is minimal, much like the slight difference in Mean Absolute Error. As mentioned before, the predictions using the model with a pre-trained ResNet on the pitch angle slightly decreased in Mean Absolute Error, which is also observed in Table 5.2.5 where the percentage of predictions within ±5° error has slightly increased. On the other angles, the Mean Absolute Error increased, which is again also visible in Table 5.2.5, where the percentage of predictions within ±5° error has slightly decreased. The model without a pre-trained ResNet shows the largest decrease of the two models, in terms of the percentage of predictions within ±5° error, since it has decreased by approximately 3% on the pitch and yaw angle and approximately 2% on the roll angle. Given that the number of predictions within ±5° error has slightly decreased, it means that the models make slightly more predictions that produce a higher error.

Figure 5.2.18 shows the Mean Absolute Error of the predictions made on different pitch angles by the model with a pre-trained ResNet and the model without a pre-trained ResNet, when downsampling all images in the final test dataset by a factor of 16.

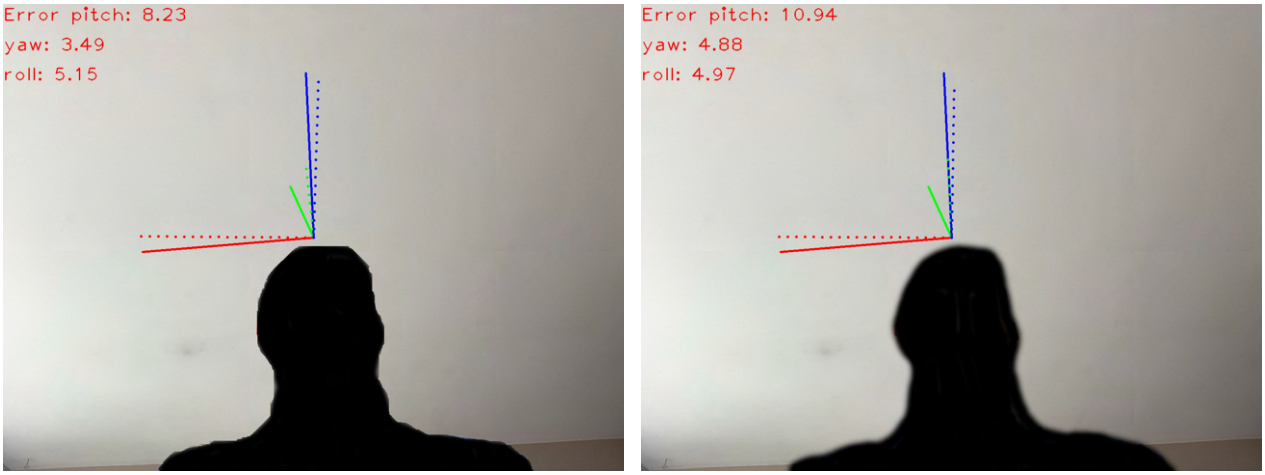


(a) *Pre-trained ResNet model* **(b)** *Non-pre-trained ResNet model*

Figure 5.2.18: *Pitch MAE grouped on similar pose ranges with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the pitch angle on all frames in the final test dataset with downsampling by a factor of 16*

Figure 5.2.18 shows that the Mean Absolute Error for different head poses on the pitch angle follows the pattern in the overall performance change on the pitch angle. The model with a pre-trained ResNet gets a slightly lower error and has an overall slight decrease in Mean Absolute Error for the different angles. The model without a pre-trained ResNet produces a higher error, which seems to be because of the higher Mean Absolute Error on the most extreme head pose of around −70° pitch.

An example of the performance difference between not downsampling and downsampling by a factor of 16 using the model without a pre-trained ResNet is presented in Figure 5.2.19. This example has a pitch angle around −70°, and it is observed that the error increases by downsampling the image by a factor of 16 from 8.2° error on pitch to 10.9° error.

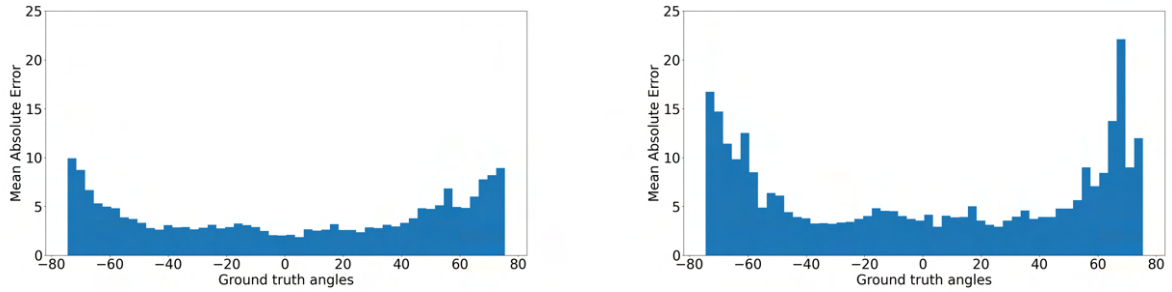


(a) *Head pose 12 - Not pre-trained ResNet model without downsampling*

(b) *Head pose 12 - Not pre-trained ResNet model with downsampling by factor of 16*

Figure 5.2.19: *Example of head pose with around −70° pitch angle*

Figure 5.2.20 shows the Mean Absolute Error of the predictions made on different yaw angles by the model with a pre-trained ResNet and the model without a pre-trained ResNet, when downsampling all images in the final test dataset by a factor of 16.



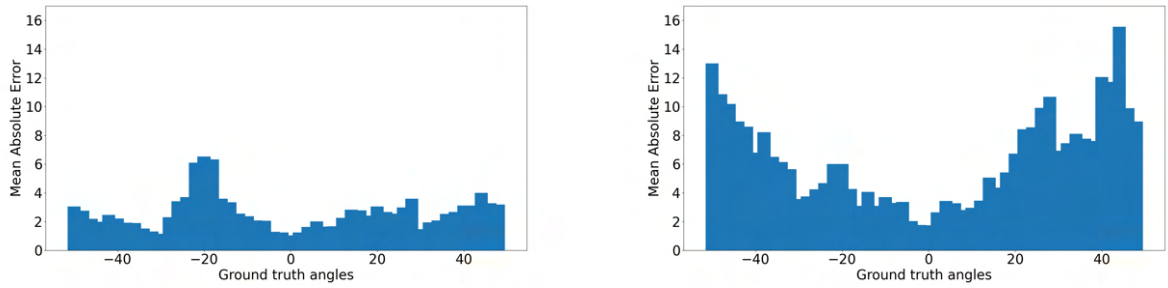
(a) *Pre-trained ResNet model*

(b) *Non-pre-trained ResNet model*

Figure 5.2.20: *Yaw MAE grouped on similar pose ranges with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the yaw angle on all frames in the final test dataset with downsampling by a factor of 16*

With the yaw angle in Figure 5.2.20, it shows a very similar shape with both models between downsampling by a factor of 16 and without downsampling from Figure 5.2.9. The model with a pre-trained ResNet has a very slight overall increase in Mean Absolute Error. In contrast, the model without a pre-trained ResNet performs slightly worse, mostly on what it had difficulty with predicting correctly before at around +65° yaw angle.

Figure 5.2.21 shows the Mean Absolute Error of the predictions made on different roll angles by the model with a pre-trained ResNet and the model without a pre-trained ResNet, when downsampling all images in the final test dataset by a factor of 16.



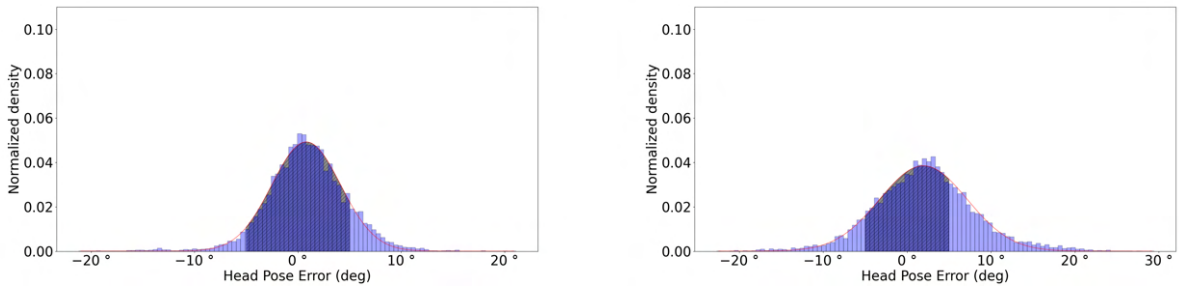
(a) *Pre-trained ResNet model*

(b) *Non-pre-trained ResNet model*

Figure 5.2.21: *Roll MAE grouped on similar pose ranges with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the roll angle on all frames in the final test dataset with downsampling by a factor of 16*

The difference in the Mean Absolute Error on the roll angles between downsampling and not downsampling the final test dataset is smaller than for the other Euler angles, reflected in Figure 5.2.21. The shape of the histograms are almost identical to the ones produced when not downsampling in Figure 5.2.12, where there are produced slightly higher prediction error in some cases and slightly lower in others.

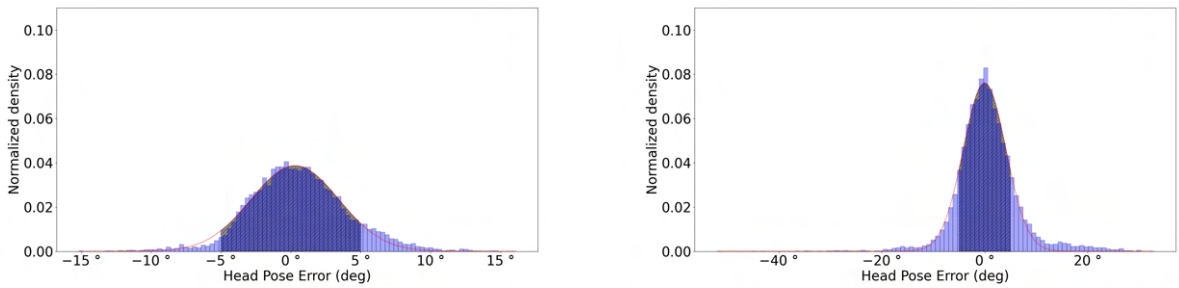
Figures 5.2.22, 5.2.23, and 5.2.24 show the Gaussian distribution of the prediction errors produced by the model with a pre-trained ResNet and the model without a pre-trained ResNet, when downsampling all images in the final test dataset by a factor of 21.



(a) *Pre-trained ResNet model.*
$Mean : 0.656°$ *and std:* $3.887°$

(b) *Non-pre-trained ResNet model.*
$Mean : 1.893°$ *and std* $: 6.016°$

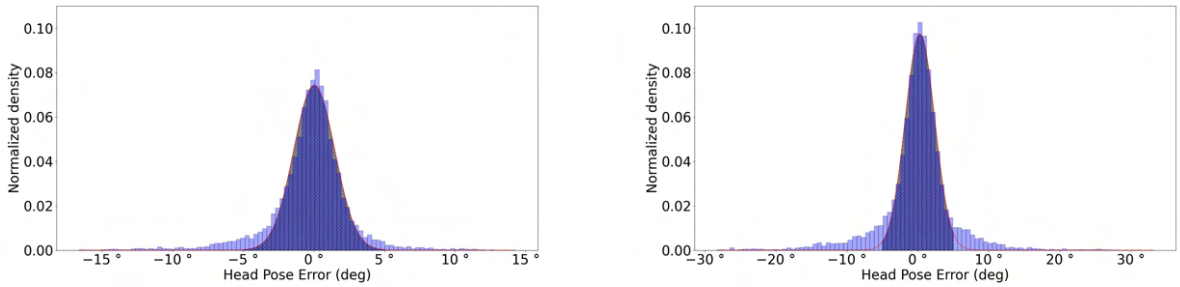
Figure 5.2.22: *Pitch Gaussian distribution with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the pitch angle on all frames in the final test dataset downsampled by a factor of 21*



(a) *Pre-trained ResNet model.*
$Mean : 0.406°$ *and std:* $3.534°$

(b) *Non-pre-trained ResNet model.*
$Mean : 0.114°$ *std:* $6.222°$

Figure 5.2.23: *Yaw Gaussian distribution with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the yaw angle on all frames in the final test dataset downsampled by a factor of 21*



(a) *Pre-trained ResNet model.*
$Mean: -0.436°$ *and std:* $2.736°$

(b) *Non-pre-trained ResNet model.*
$Mean: -0.360°$ *and std:* $4.986°$

Figure 5.2.24: *Roll Gaussian distribution with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the roll angle on all frames in the final test dataset downsampled by a factor of 21*

When going from downsampling all images in the final test dataset by a factor of 16 to a factor of 21, we observe an overall increase in Mean Absolute Error for all Euler angles in Table 5.2.4. The model with a pre-trained ResNet still has a slightly lower Mean Absolute Error on the pitch angle compared to not downsampling the dataset. However, it has increased slightly more on the two other Euler angles.

The Gaussian distributions in Figure 5.2.22, 5.2.23, and 5.2.24, continue to (slightly) rise in standard deviation on both models for all Euler angles when compared to the the Gaussian distributions produced without downsampling.

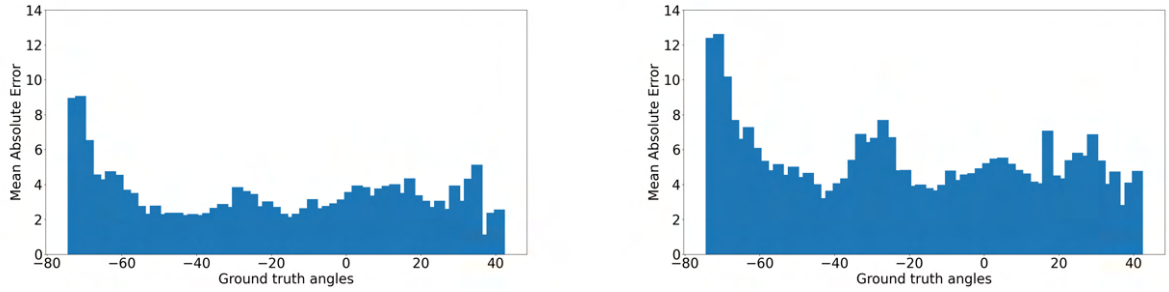
Table 5.2.6 shows the percentage of predictions within $\pm 5°$ error for each Euler angle and model when the images are downsampled by a factor of 21.

Model	Pre-trained ResNet	Not pre-trained ResNet
Pitch	79.52%	57.12%
Yaw	84.00%	57.82%
Roll	92.88%	68.27%

Table 5.2.6: *Percentage of predictions, on downsampled images by a factor of 21, within* $\pm 5°$ *error*

It is shown in Table 5.2.6 that the larger downsampling once again makes the percentage of predictions within $\pm 5°$ error decrease. The decrease from downsampling by a factor of 16 is quite close to the decrease from not downsampling to downsampling by a factor of 16. The total decrease from not downsampling to downsampling by a factor of 21 is approximately 1% on pitch, 2% on yaw, and 1% on roll for the model with a pre-trained ResNet. For the model without a pre-trained ResNet, the total decrease from not downsampling to downsampling by a factor of 21 is approximately 7% on pitch, 6% on yaw, and 4% on roll. From this, we observe that both models are relatively robust towards downsampling the data and that the model with a pre-trained ResNet is more robust than the model without a pre-trained ResNet when it comes to maintaining the number of accurate predictions for lower resolution images. But since both models show a slight decrease in the percentage of predictions that fall within $\pm 5°$ error, it once again means that there are slightly more predictions that produce a higher error.

Figure 5.2.25 shows the Mean Absolute Error of the predictions made on different pitch angles by the model with a pre-trained ResNet and the model without a pre-trained ResNet, when downsampling all images in the final test dataset by a factor of 21.



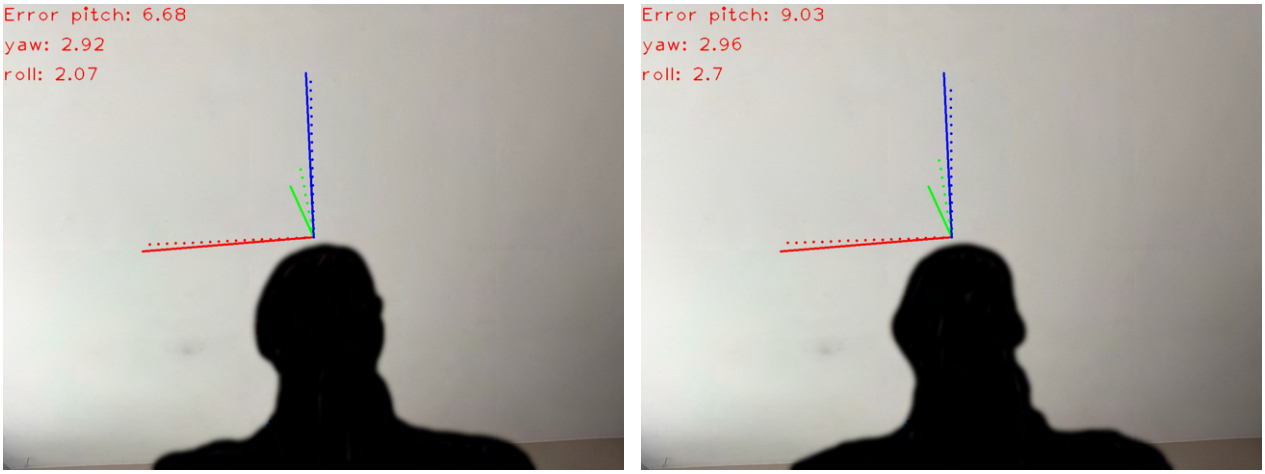
(a) *Pre-trained ResNet model*

(b) *Non-pre-trained ResNet model*

Figure 5.2.25: *Pitch MAE grouped on similar pose ranges with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the pitch angle on all frames in the final test dataset with downsampling by a factor of 21*

Given that the Mean Absolute Error increases on the pitch angle from downsampling by a factor of 16, we also observe a slight overall increase in Figure 5.2.25 for both models. However, when downsampling by a factor of 21 we observe that the model with a pre-trained ResNet start to follow the same shape in the histogram as the model without a pre-trained ResNet. The Mean Absolute Error increases on the end at around $-70°$ for both models now, which only increased on the model without a pre-trained ResNet when downsampling by a factor of 16.

This is also visible in the same example image as before, with a pitch angle of around $-70°$.



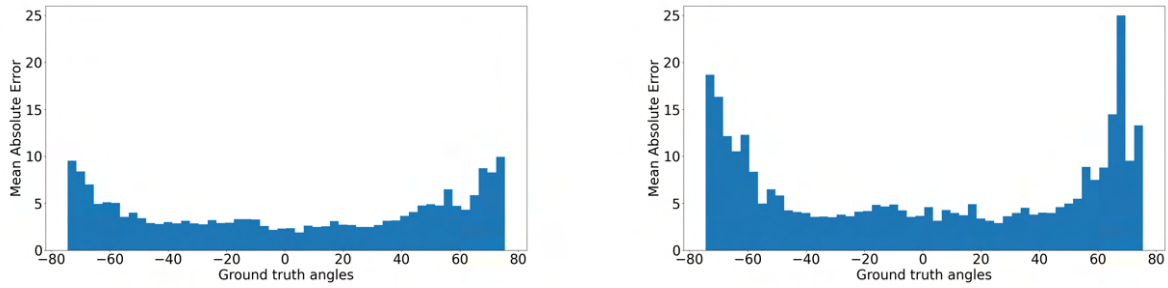
(a) *Head pose 12 - Pre-trained ResNet model without downsampling*

(b) *Head pose 12 - Pre-trained ResNet model with downsampling by factor of 21*

Figure 5.2.26: *Example of head pose with around $-70°$ pitch angle*

Since the model with a pre-trained ResNet forms a similar histogram shape as the model without a pre-trained ResNet, the image in Figure 5.2.26 also shows an increased error from 6.6° pitch error to 9° pitch error when downsampling by a factor of 21. The model without a pre-trained ResNet follows the same pattern as before and increases the error even more for pitch angles at around $-70°$.

Figure 5.2.27 shows the Mean Absolute Error of the predictions made on different yaw angles by the model with a pre-trained ResNet and the model without a pre-trained ResNet, when downsampling all images in the final test dataset by a factor of 21.



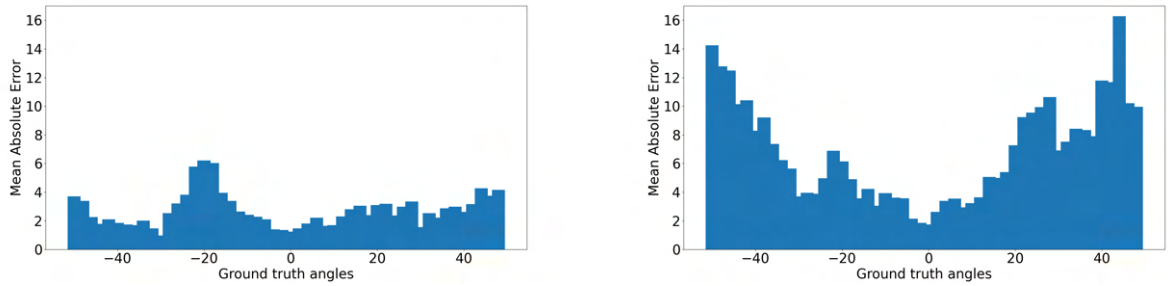
(a) *Pre-trained ResNet model* **(b)** *Non-pre-trained ResNet model*

Figure 5.2.27: *Yaw MAE grouped on similar pose ranges with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the yaw angle on all frames in the final test dataset with downsampling by a factor of 21*

Figure 5.2.27 shows the yaw angle which the model with a pre-trained ResNet increases slightly in error mostly on extreme yaw angles, when compared to downsampling by a factor of 16, and that the model without a pre-trained ResNet mostly increases in error on the extreme ends of the spectrum and especially at $+65°$.

Much like the error on the pitch angle, downsampling by a factor of 21 seems to make it more difficult for the models to predict the correct angles on the extreme head poses, which can be interpreted as the more difficult images.

Figure 5.2.28 shows the Mean Absolute Error of the predictions made on different roll angles by the model with a pre-trained ResNet and the model without a pre-trained ResNet, when downsampling all images in the final test dataset by a factor of 21.



(a) *Pre-trained ResNet model* **(b)** *Non-pre-trained ResNet model*

Figure 5.2.28: *Roll MAE grouped on similar pose ranges with pre-trained ResNet model and Non-pre-trained ResNet model, both produced by predicting the roll angle on all frames in the final test dataset with downsampling by a factor of 21*

Both models produce a similar histogram on the roll angle when downsampling by a factor of 21 as when downsampling by a factor of 16, as shown in Figure 5.2.28. The only difference is that the Mean Absolute Errors continue to increase on the angles, which the given model found difficult without downsampling as well. Given that the overall error slightly increased when downsampling by a factor of 16 and more with a factor of 21, we believe that the error will continue to increase when using larger downsampling factors.

Comparison and Future Work

6.1 Comparison with related work

Given that the custom dataset proposed in this master project consists of actual data and not synthetic data, the most comparable dataset is the BIWI dataset [8]. Looking at the results presented in [19], this dataset produces a lower error of 2.393 compared to 4.895 on the BIWI dataset when using HopeNet. However, it still falls behind 3D Morphable Model+ (3DMM+) Online [22], which got an error of 2.066 on the BIWI dataset.

The BIWI dataset contains 5.000 more images, has a larger pitch maximum angle, and contains images of more people of different genders, which should make it more difficult to find a general solution than only having one individual in the dataset.

6.2 Expand the dataset

The custom dataset with 60,000 images of different head poses in different lighting conditions was only made by one individual (white male), which makes the possibility of this model performing well on actual data with different people of different genders less likely. To make any factual claims about the real-world performance of this model and be able to better compare this dataset with other datasets, a more diverse dataset needs to be created with many different people. Luckily, it is relatively easy to record a video and can easily be made a public task to gather more data by anyone with an iPad Pro with a TrueDepth sensor.

Furthermore, our dataset is somewhat small for a deep learning task, and we expect the model to generalize better for all angles if more images are added. Specifically for the head pose angles, which were shown to be difficult for both models in this master project, we believe that this error could be reduced by creating more images in these head pose ranges.

6.3 Comparison with other methods

Another different method that would be interesting to compare this method with, is the Google MediaPipe Face Mesh estimator [1]. It would first have to be determined if the head pose can be obtained directly from the Face Mesh, or if facial landmarks have to be extracted from the Face Mesh first, before comparing the head pose estimations made by Google MediaPipe with the two models presented in this master project.

6.4 Predicting head pose and depth

Given that the data collected with the 2018 iPad Pro contains the Euler angles and the 3D position of the head pose, an obvious extension to the base model is to enable 3D position prediction while also predicting the Euler angles.

This is achieved by adding another fully connected layer with three outputs that are simply evaluated with MSE regression loss, which is then added to the total loss. The final loss is then:

$$total\_loss = angle\_loss_{pitch} + angle\_loss_{yaw} + angle\_loss_{roll} + \alpha \cdot MSE(y_{position}, \hat{y_{position}}).$$

The motivation for this is that the model will then wholly replace depth-sensing cameras for head pose estimation. Given that input images for the model can have different resolutions and the used device can have different camera parameters, it is not optimal to use the pixel coordinates to represent the ground truth position. For this reason, the ground truth position values are the world coordinates since we calculate them into pixel coordinates by using the camera matrix.

[1] https://google.github.io/mediapipe/solutions/face_mesh

To get the pixel coordinates from the world coordinates, we first have to multiply the world coordinates by the projection matrix:

$$P = K[R|t]$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix},$$

where P is the projection matrix, K are the intrinsic parameters of the camera (focal length, shear, and principal point), $[R|t]$ are the extrinsic parameters of the camera (rotation and position in the world coordinate space), (X, Y, Z) is the 3D point in the world coordinate system and (x, y, z) is the image point (with depth). Then we simply need to remove the depth dimension by dividing x and y by z.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x/z \\ y/z \\ z/z \end{bmatrix}$$

We believe that this approach will predict the head pose in Euler angles and the 3D position of the head with high accuracy, which is why we intend to publish a paper with this method.

Conclusion

In this work, we have created a custom dataset of approximately 60,000 images (50,000 for 10-fold cross-validation and 10,000 for final testing) with various head poses in different lighting conditions and have tested it using the multi-loss deep neural network called HopeNet. This network architecture has shown impressive results in previous research. The results presented in this report suggest that it gets close to the exact predictions made by the 2018 iPad Pro with a TrueDepth sensor.

The results presented in this report also showed that the architecture performs better in general for all head poses when the ResNet backbone structure has been pre-trained on ImageNet and gets fine-tuned during training on the custom dataset made with the 2018 iPad Pro. The model that initializes the ResNet backbone structure with random weights has been shown to produce higher error in general on most head poses but does not necessarily perform worse only on the difficult head poses for the model that uses a pre-trained ResNet.

When downsampling the final test dataset by a factor of 16 (largest downsampling seen during training) and by a factor of 21 (unseen during training), we showed that both models got slightly worse for each downsampling increment. However, the most robust model was once again shown to be the model with a pre-trained ResNet. The model without a pre-trained ResNet was shown to be the most sensitive model out of the two towards downsampled images. It was observed that both models primarily increased slightly in prediction error on the head poses, which were already challenging for the given model and that the difficult head poses for pitch and yaw were extreme head poses. At the same time, the roll angle also showed difficult head poses that were not necessarily extreme. But all things considered, both models still showed to be relatively robust towards lower resolution images.

To further develop this project, we suggest increasing the dataset by recording more videos with an iPad Pro equipped with the TrueDepth sensor and include more individuals in this dataset. Furthermore, we believe that a more complex architecture can be built by training the model to predict the 3D position of the head, which is given as information by the 2018 iPad Pro alongside the Euler angles of the head. We intend to publish a paper in a conference with this method.

References

[1] S. Alfayad et al. 'HYDROïD Humanoid Robot Head with Perception and Emotion Capabilities: Modeling, Design, and Experimental Results'. In: (2016). URL: https://www.frontiersin.org/articles/10.3389/frobt.2016.00015/full.

[2] Apple. 'About Face ID advanced technology'. In: (2020). URL: https://support.apple.com/en-us/HT208108.

[3] Apple. 'Tracking and Visualizing Faces'. In: (). URL: https://developer.apple.com/documentation/arkit/content_anchors/tracking_and_visualizing_faces.

[4] Jason Brownlee. 'Gradient Descent For Machine Learning'. In: (2016). URL: https://machinelearningmastery.com/gradient-descent-for-machine-learning/.

[5] Adrian Bulat and Georgios Tzimiropoulos. 'How far are we from solving the 2D 3D Face Alignment problem? (and a dataset of 230,000 3D facial landmarks)'. In: (2017). URL: https://www.adrianbulat.com/downloads/FaceAlignment/FaceAlignment.pdf.

[6] Martin Danelljan et al. 'Accurate Scale Estimation for Robust Visual Tracking'. In: (2014). URL: http://www.bmva.org/bmvc/2014/files/paper038.pdf.

[7] David Eberly. 'Euler Angle Formulas'. In: (). URL: https://www.geometrictools.com/Documentation/EulerAngles.pdf.

[8] Gabriele Fanelli et al. 'Random forests for real time 3D face analysis'. In: (2013). URL: https://pages.iai.uni-bonn.de/gall_juergen/download/jgall_RFdepthFace_ijcv12.pdf.

[9] Kaiming He, Ross Girshick and Piotr Dollár. 'Rethinking ImageNet Pre-training'. In: (2018). URL: https://arxiv.org/pdf/1811.08883.pdf.

[10] Kaiming He et al. 'Deep Residual Learning for Image Recognition'. In: (2015). URL: https://arxiv.org/pdf/1512.03385.pdf.

[11] Davis E. King. 'Max-Margin Object Detection'. In: (2015). URL: https://arxiv.org/pdf/1502.00046.pdf.

[12] Kiprono Elijah Koech. 'Cross-Entropy Loss Function'. In: (2020). URL: https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e.

[13] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. 'ImageNet Classification with Deep Convolutional Neural Networks'. In: (2012). URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[14] Satya Mallick. 'Head Pose Estimation using OpenCV and Dlib'. In: (2016). URL: https://learnopencv.com/head-pose-estimation-using-opencv-and-dlib/.

[15] Siladittya Manna. 'K-Fold Cross Validation for Deep Learning Models using Keras'. In: (2020). URL: https://medium.com/the-owl/k-fold-cross-validation-in-keras-3ec4a3a00538.

[16] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. 'Head Pose Estimation in Computer Vision: A Survey'. In: (2009). URL: http://cvrr.ucsd.edu/publications/2009/MurphyChutorian_Trivedi_PAMI09.pdf.

[17] Fabricio Batista Narcizo. 'Using Priors to Improve Head-Mounted EyeTrackers in Sports'. In: (2017). URL: https://pure.itu.dk/portal/files/83177849/PhD_Thesis_Final_Version_Fabricio_Batista_Narcizo.pdf#cite.Alfayad2016.

[18] Hossein Pishro-Nik. *Introduction to Probability, Statistics, and Random Processes*. 2014. URL: https://www.probabilitycourse.com/chapter9/9_1_5_mean_squared_error_MSE.php.

[19] Nataniel Ruiz, Eunji Chong and James M. Rehg. 'Fine-Grained Head Pose Estimation Without Keypoints'. In: (2018). URL: https://arxiv.org/pdf/1710.00925.pdf.

[20] Sumit Saha. 'A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way'. In: (2018). URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[21] Yash Upadhyay. 'Introduction to FeedForward Neural Networks'. In: (2019). URL: https://towardsdatascience.com/feed-forward-neural-networks-c503faa46620.

[22] Yu Yu, Kenneth Alberto Funes Mora and Jean-Marc Odobez. 'Robust and Accurate 3D Head Pose Estimationthrough 3DMM and Online Head Model Reconstruction'. In: (2017). URL: http://publications.idiap.ch/downloads/papers/2017/Yu_FG2017_2017.pdf.