Channel Selection for Faster Deep Learning-based Gaze Estimation in the Frequency Domain

A frequency domain approach to reducing latency in deep learning gaze estimation

Thijs Penning[1]

Supervisors: Dr. G. (Guohao) Lan[1]**, Lingyu Du**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Thijs Penning
Final project course: CSE3000 Research Project
Thesis committee: Dr. G. (Guohao) Lan, Dr. Xucong Zhang, Lingyu Du

Abstract

Gaze estimation is an important area of research used in a wide range of applications. However, existing models trained for gaze estimation often suffer from high computational costs. In this study, frequency domain channel selection techniques were explored to decrease these costs by reducing the size of the input data. The main research objective was to investigate the impact of channel selection on the latency and accuracy of frequency domain gaze estimation. Channel selection methods used in related research were adapted and applied to the domain of gaze estimation. The evaluation was conducted on two popular network architectures used in this field, namely the AlexNet and ResNet-18. Multiple channel selection models were designed for each architecture and compared to a traditional RGB approach with the same network structure. Experimental results showed significant speedups during training, calibration, and inference with marginal accuracy loss. The specific speedups that the top-performing models of both the architectures achieves were 3.3, 4.0, and 1.35 for the AlexNet, and 1.5, 1.7, and 1.35 for the ResNet-18. Accompanying these speedups the AlexNet model error only increased by 0.08 degrees compared to a traditional RGB approach, while the ResNet-18 model lost around 0.44 degrees. All the code used in this research is publicly available on GitHub[1].

1 Introduction

Background There have been numerous studies in the field of computer vision focused on estimating and applying human gaze. Human gaze is an important non-verbal cue that communicates valuable information about someone's desires and emotions [1]. This information has applications in various fields of study within and beyond the scope of computer science. In cognitive science, for example, gaze estimation is applied to better understand visual perception [2], while in human-robot interaction, gaze information can be provided to the robots to improve interactions [3]. Additionally, human gaze is utilized in everyday technologies such as smartphones, tablets and webcams to gain useful insights and enhance user experiences [4–6]. As indicated by these examples, human gaze estimation has a wide range of applications across various disciplines.

The field of gaze estimation has changed over the years with the introduction of new technologies and methods. Previously, gaze estimation was performed using machine learning algorithms. Over the last decade, developments in the field of deep learning have paved the way for deep learning gaze estimation. The deep learning techniques allowed for different approaches, resulting in better feature extraction and higher accuracies [7, 8]. Today, gaze estimation is still centered around deep learning convolutional neural networks (CNN). These complex networks can derive the information required to accurately estimate gaze from a subject's eyes or their full face [9].

Related work While significant progress has been made in gaze estimation, challenges remain due to the latency caused by the high computational costs of the complex CNN needed to complete the task. Most popular network architectures utilized in gaze estimation consist of a few million to over a hundred million parameters [10–15]. Additionally, these networks typically require input images of size 224×224, which is relatively small for images, but they still contain over 150 thousand data points each. Combining the computational complexity of these networks with the relatively "large" image sizes results in high computational costs, leading to latency in calibration and interference when used in real-world applications. This latency will, in most cases, decrease user experience [16], and in fields where fast responsiveness is required [17] it could render the model completely useless.

An obvious solution to decrease computational costs is to reduce the input data. The typical approach already applies this approach by cropping and resizing larger images to the previously mentioned 224×224 size. While this is effective for reducing the input size, valuable information can be lost. This trade-off is acceptable for the 224×224 size images, but becomes problematic when attempting to go smaller. As a result, the images in the current RGB format can not be reduced further without substantial data loss. Therefore, a different data representation is required. By converting the RGB images to the frequency domain and applying channel selection, less crucial and noisy information can be removed, while retaining the important data and effectively reducing the overall data size. In their work, Xu et al. [18] showed that CNN taking RGB input can easily be modified to utilize the frequency domain, after which up to 87.5% of the frequency channels could be removed with marginal compromise to accuracy. Other studies applying this technique for various tasks and methods have shown similar results: a speedup of the model with marginal loss of accuracy, or even a gain with certain setups [19, 20]. These previous studies focused on various aspects, image classification, detection and segmentation tasks using a complex channel selection method [18], unsupervised gaze estimation [19], and ImageNet[2] trained models with JPEG frequency domain conversion [20]. However, the impact of using channel selection in the frequency domain for supervised deep learning gaze estimation remains largely unexplored.

Research question and main contributions This research addressed the information gap by investigating the impact of channel selection on the latency and accuracy of frequency domain gaze estimation. Specifically, the focus was on finding channel selections that provide an optimal trade-off for maximal speedup with minimal accuracy loss. In short, this research converted the conventional RGB images to the frequency domain, studied multiple ways to apply channels selection of the frequency domain converted data, and analyzed their impact on both latency and accuracy on

[1] https://github.com/tpenning/DLFDFaceGazeEstimation

[2] https://www.image-net.org/

various model architectures.

Paper Outline The methodology of the research is discussed in Chapter 2. The experimental setup exploring various ideas, along with the achieved results are presented in Chapter 3. The responsible research practices with regards to scientific integrity and AI usage are further elaborated in Chapter 4. Finally, Chapter 5 concludes this paper with a final overview of the work and mentions the shortcomings and possible future extensions.

2 Methodology

Various deep learning gaze estimation models based on different data formats focused on the frequency domain were tested. These different formats were achieved through a linear, multi-step data preparation process, where each step built upon the previous one resulting in additional data formats. The images were first preprocessed by cropping and resizing them to the commonly used 224×224 size. After that, the images were transformed into the YCbCr color space (Section 2.1). Once in the YCbCr color space, the images could be converted into the frequency domain (Section 2.2). Finally, channel selection techniques were applied to the frequency domain data (Section 2.3).

2.1 Transforming into the YCbCr Color Space

The images were transformed to the YCbCr color space, as this color representation is commonly used in popular compression standards such as JPEG. The YCbCr color space consists of three information channels like RGB, but these channels store the luminance and chrominance information of an image instead of the regular colors. The Y channel is the luma component, and contains the information about brightness and color intensities. The Cb and Cr channels are the chroma components, and store the color variation in the image, indicating how the actual color displayed should differ from the luminance color of the Y channel. The channels are illustrated in Figure 1 showing the information stored in each channel and highlighting their focuses.

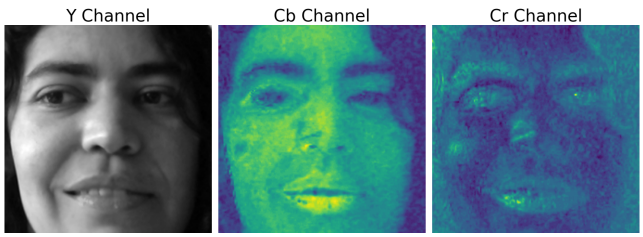


Figure 1: YCbCr color space components shown individually.

The luma component (Y) showing brightness is displayed in grayscale. The chroma components (Cb and Cr) are displayed in a color mapping to highlight the focus of both channels for color variation.

2.2 Converting to the Frequency Domain

To convert the data from YCbCr to the frequency domain, the discrete cosine transform (DCT) was applied to the images. The following steps were applied separately to the Y, Cb, and Cr channels, resulting in three channels of frequency domain information for each image. First, the channels (each with a size of 224×224) were divided into non-overlapping blocks of 8×8 resulting in 28×28 blocks per channel per image. After this, the DCT was applied to each block, generating 64 frequency channels that store the frequency information. These channels contain all the necessary information to reconstruct the original block they originated from. The distribution of this information across the channels varies per image, though generally the distribution is similarly skewed due to the nature of the DCT [21]. Only a (small) subset of the channels contain the majority of the information, allowing the other channels to be removed without substantial data loss. By applying channel selection techniques, this effect could be exploited, potentially allowing significant data size reductions while retaining the important information.

2.3 Channel Selection

Performing channel selection optimally is a critical step when working with the frequency domain. The goal is to take advantage of the significant data reduction possible while keeping enough information to provide accurate results. Therefore, selecting the channels to keep must be done carefully. In this section, the distribution of the information contained in the frequency channel is thoroughly analyzed, and the applied channel selection methods are discussed.

2.3.1 Channel information analysis

To analyze the information stored in the frequency channels, the effects of individual channels were examined using image reconstruction. An image can be converted to the frequency domain using the steps explained in Section 2.2. Following these steps in reverse, and with the inverse discrete cosine transform (IDCT), the original image can be reconstructed from frequency domain channels.

To analyze the value of certain frequency channels, images were transformed to the frequency domain and reconstructed with the respective channels information set to zero. Similarly, the information from individual channels could be isolated by setting all other channels to zero. A side effect of this approach is that the ”blank” reconstructed image is green due to the zero values. However, while other ”blank” image colors such as black or white can be reached by using non-zero values as the ”removed” values in the lowest frequency, these could potentially have more unintended side effects that could affect the data shown and were therefore not used. Additionally, the green background provided better contrast with the information displayed improving the visibility during analysis. To visualize the information in the channels, all 64 frequencies channels were isolated for each of the YCbCr components and an extra insight was made by showing each frequency channel together for all the components. These visualizations are shown in Figure 2.

Upon analyzing Figure 2, several interesting observations can be made, with three main observations that will be focused on. Firstly, the lower frequency channels (located in the top left of the images) appeared to contain the critical information of the image. This was expected due to the work-

ings of DCT [21]. In comparison, the high frequency channels (situated in the bottom right of the images), contained minimal information and are generally considered as noise [22]. Secondly, in addition to the lower frequencies being more vital, the lowest frequency showed to contain the most information. Already shown from their impact on the background color, these channels also provided a relatively accurate reconstruction of the original image far clearer to the human eye than the results from other channels. Thirdly, the luma component (Y channel) seemed to contain more information than the chroma components (Cb and Cr channels). These findings align with established knowledge about DCT [21, 22], and are consistent with similar research conducted by Xu et al. [18].

2.3.2 Static channel selection

The channel selection has an important influence on the model design. To keep the design simple, an approach that would keep the channel selection consistent throughout all images was preferred. Static channel selection fits this description well, as with this method the channel selection is made manually and does not change depending on the image. Since this selection had to be applied to all images it was important that it fitted to all the subjects. To determine which channels generally contained the most information, images from all training subjects were analyzed using the approach illustrated in Figure 2. The results obtained from these analyses generally aligned with the findings from Figure 2, highlighting the importance of the low frequency channels even further. The range of important channels did vary across subjects though, indicating that to ensure a good general model performance a broader selection would probably be better. Utilizing this information, multiple models with different channel selections were designed and experimented with, as detailed in Section 3.2.2.

2.3.3 Dynamic channel selection

Models using a more complex dynamic channel selection allow data distribution differences to be handled more efficiently. The frequency channel analysis described in Section 2.3.2 revealed variations in the data distributions among the subjects, with specific channels being more or less utilized. While static channel selection can include all relevant channels in its selection, dynamic channel selection varies its channel usage based on each image. In dynamic channel selection, all channels are used as input. Based on the information in the channels, individual channel are then "turned off" by setting their values to zero. These zeros are computationally faster which results in the speedup for this type of channel selection.

The idea behind the dynamic channel selection implementation was inspired by Xu et al. [18], where they applied it to image classification, detection and segmentation tasks. This paper did not provide any source code of their implementation, therefore it was coded from scratch. Following the description of their approach, together with various testing, a model was designed that had a similar yet slightly modified structure. The dynamic model required an additional mini model in front of the network that decided the selections based on the image, which were then multiplied with this same image to create the input for the rest of the network.

The selection process involved multiple steps that facilitate channel learning and exploration. First an average pool was applied to the original image. This averaged information was input into two different 1×1 convolutional layers, generating two values for each channel. These values were then normalized together to the range $[0, 5]$, representing probabilities of selecting the channel or not. To generate selection values based on these probabilities, the Gumbel Softmax reparameterization trick was used, as sampling a Bernoulli distribution is not differentiable. This trick added noise following a Gumbel distribution to the values and then applied Softmax to them resulting in values in the range $[0, 1]$. Then, to remove and reduce the floating point operations (FLOPS) the lowest values were set to zero. By applying mathematical operations ($[0, 1] * 1.1 - 0.1$) to the range and using a ReLu activation function, all values smaller than 0.09 were effectively removed. Finally, the original image was multiplied with this resulting tensor of selections and input in the rest of the model.

To allow the model to decide on a good trade-off between the amount of channels to use and the accuracy achievable, the loss function had to be adjusted. For the dynamic model, the average amount of channels selected per batch was added from the mini model directly to the full model output. When learning, the loss was then based on two parts: first, the L1 loss on the first two values in the output representing the gaze direction was calculated. Second, the last output value, the number of channels, was averaged over all the batches and multiplied with a regularization parameter. The full loss consisted of these two losses added together.

3 Experiments

This chapter describes the dataset used during the experiments (Section 3.1), and, the experimental setup detailing model design and training is discussed (Section 3.2). Finally, the obtained results are visualized and elaborated upon (Section 3.3).

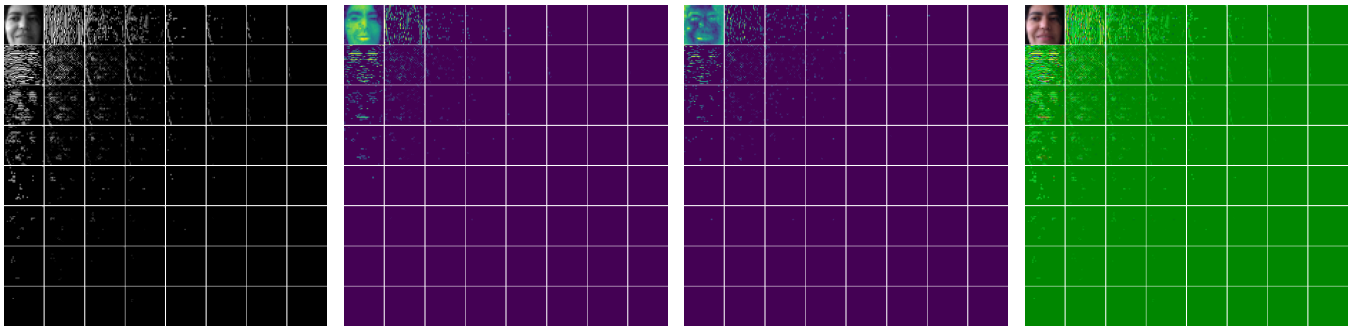
3.1 Dataset

The experiments were performed using the publicly available MPIIFaceGaze dataset[3]. The MPIIFaceGaze dataset has been designed for gaze estimation tasks and consists of 45,000 full-face images of subjects looking in varying directions. The dataset contains information from 15 distinct subjects, allowing the trained models to be more robust and generalize better across different individuals and environments. The images themselves are of size 448×448, suitable for larger models, but also easily resizable to 224×224 as used in this research. Additionally, labels in pitch and yaw describing the 3D gaze direction of the images are provided as part of the dataset.

3.2 Experimental Setup

The experiments performed involved testing various model setups using frequency domain information and comparing

[3] https://perceptualui.org/research/datasets/MPIIFaceGaze/



(a) Y channels (b) Cb channels (c) Cr channels (d) Combined channels

Figure 2: Information stored in the individual frequency channels.

Low frequency channels are in the top left and the high frequencies in the bottom right. (a) is the Y channel displayed in grayscale and (b) and (c) are the Cb and Cr channels displayed with a color mapping. (d) is the combination of the same frequency channel for Y, Cb, and Cr displayed in RGB.

these to models based on RGB images. The following sections explain the models and procedures used in the experiment, including the key components, i.e., network architectures, model design and phases and procedures.

3.2.1 Network architectures

The goal of this research was to develop faster models with comparable accuracy by utilizing the frequency domain. However, the effects of the frequency domain data vary depending on the network structure. Therefore, two popular architectures with distinct structures were used to show more relevant and diverse results. Ordered by citation count, the following popular gaze estimation architectures were considered: ResNet [10], AlexNet [11], VGGNet [12], GoogLeNet [13], MobileNet [14], and SqueezeNet [15]. Ultimately, the AlexNet and ResNet variant ResNet-18 were selected. Despite both options being among the larger candidates, their significant relevance and clear structural differences guided the selection. The AlexNet consists of only 8 complex layers, totaling around 60 million parameters. On the other hand, ResNet-18 contains 18 "simpler" layers containing around 11 million parameters in total. These architectures allowed us to examine the effects on shorter, more complex networks and longer, less complex ones.

3.2.2 Model design

To find good channel selections and measure their performance, multiple baseline and experimental models were designed for both of the selected architectures. Four baseline models were designed as reference points to compare the experimental models to. The first baseline model was trained on 224×224 RGB images following traditional models, and acted as the main baseline. The second baseline used 224×224 YCbCr images to show any change in accuracy of the different color representation utilized by the frequency domain models. The third baseline utilized the frequency domain without channel selection. This model should, in theory, contain the same information and therefore produce similar results as the previous baselines. The fourth and final baseline again used the frequency domain, this time with channel selection, selecting only the lowest frequency of each YCbCr component acting as a minimal result baseline.

The effects of channel selection were tested in a step-wise manner with six experimental channel selection models, each expanding the range of selected channels. The channel selections made are displayed in Table 1. As indicated by the analysis performed in Section 2.3, all the channel selections focused around the lower frequency channels. Each next model "grew" its selection from the previous one by selecting the next lowest frequencies of Y and/or Cb and Cr channels. There is one exception that does not follow this pattern, selection FD4 is a mutation of selection FD5 based on the dynamic channel selection preferences of [18] testing if the same channel preference would provide good results in gaze estimation. Furthermore, the dynamic channel selection FDD7 stands on its own and is not related to the static channel selections.

Table 1: Channel selections for the experimental models.

The channel selections made shown by their ID, FD0 and FD1 are two of the baseline models, FD2 - FD6 are the static channel selections, and FDD7 is the dynamic model. The Y channel denotes the Y channel selections, while the Cb and Cr components share the same selections displayed under the Cb/Cr channel column. The channel indices layout has zero in the top left and sixty-three in the bottom right. The channels in between are numbered row-wise, for example, the top right channel has index seven.

Selection ID's	Y Channel	Cb/Cr Channel
FD0	all	all
FD1	[0]	[0]
FD2	[0, 1, 8]	[0, 1, 8]
FD3	[0, 1, 2, 8, 9, 16]	[0, 1, 8]
FD4	[0, 1, 2, 3, 8, 9, 10, 16, 17, 24]	[0, 1, 3, 8, 24]
FD5	[0, 1, 2, 3, 8, 9, 10, 16, 17, 24]	[0, 1, 2, 8, 9, 16]
FD6	[0, 1, 2, 3, 4, 8, 9, 10, 11, 16, 17, 18, 24, 25, 32]	[0, 1, 2, 3, 8, 9, 10, 16, 17, 24]
FDD7	dynamic	dynamic

The network architectures required slight modifications to fit the various data formats. All models were changed to produce only two output values (the pitch and yaw), and all

AlexNet models had one fully connected layer removed as it did not improve the accuracy. For the RGB and YCbCr models, no further changes were required as these models used standard 224×224 size images for which these architectures were designed.

Looking at the frequency domain models, changes were necessary due to the input data being too small in size combined with the high channel count. Furthermore, the frequency domain conversion already applied a kernel operation similar to what the neural network would start with, which meant that less kernel operation in the network should not pose an immediate problem. As a result, the models were changed to preserve more channel information and reduce the data size less to keep the parameter count similar to the other models for a fair comparison. Additionally, to fit the high channel count better two versions were made of each model. One model would use the regular architecture, with the other changes made to it. The other model would follow the same structure, but the channels of the data in all the layers was doubled. This double channel model approach was also ran for the non-frequency domain models to provide a fair comparison.

3.2.3 Phases and procedures

The experiment consisted of training, calibrating, and running inference on each of the models. In each phase, the time spent and final error were measured. In the full experiment, each model went through all the phases from scratch ten times. After, the mean and standard deviation of the measurements from all the runs were recorded. As the relative time measurements slightly differ depending on the computing device, it is important to note that this experiment was conducted on the TU Delft DefltBlue supercomputer[4] as of June 2023.

The training process used the data of the first 14 subjects, which was divided into a training and validation set using 90% and 10% of the data respectively. The model was trained for 20 epochs on this data, where in each epoch, first the training data was run and then the error on the validation set was measured. This approach provided a clear overview of the learning curve, and using the validation set the model with the lowest validation error was saved for the next phase. Following this setup, it was opted to measure the entire training time including running the validation set, as it was part of the process, and instead of the final error, the error of the saved model was used. Furthermore, during the training process, the following parameters were used: batch size 32, training epochs 20, and learning rate 0.0001.

The calibration is in practice basically a second training. For this reason a similar setup as the training phase was utilized. This time the data was different as the 15th and last subject was used, and the data was split into 100 images for calibration and 2900 images for the validation set. The process and parameters used were the same as the training phase, with the only differences being that 100 epochs were used for calibration, and the batch normalization layers were frozen.

The last phase, running inference, tests the most important part of the system, simulating real world use. In this phase, the model was run on all images of the last subject including the 100 images used during calibration. This phase did not use multiple epochs and ran the images one at a time. The measurements consisted of the total time spent and average error over the 3000 images.

[4]https://www.tudelft.nl/dhpc/system

3.3 Results

All the designed models went through all three phases: training, calibration, and inference ten times, as described in Section 3.2. The mean and standard deviation of the measurements over these ten runs are displayed in Table 2. Section 3.3.1 analyzes the effects of the different models and selections on the training and calibration phases. The inference results are further examined in Section 3.3.2. Finally, Section 3.3.3 presents the optimal selections based on the results, along with their corresponding speedup and error changes over the baseline.

3.3.1 Training and calibration

All the frequency domain models were faster during training and calibration than the color (RGB and YCbCr) models (Table 2). The exact speedups achieved varied depending on the architecture and whether the regular or modified version was used. The fastest frequency domain models achieved speedups of around 3.3, 3.3, 1.5, and 1.2 during training and 4.0, 3.2, 1.8, and 1.2 during calibration. Comparing the individual frequency domain models, the differences were marginal. The baseline using all channels and the dynamic channel selection model were slightly slower than the others, although still faster than the color models. The other models (FD1 - FD6), which were the static channel selection models, showed marginal differences. Only the largest selections were slightly slower, and some outliers caused slightly higher means, as indicated by the larger standard deviations.

While the frequency domain models were consistently faster than the color models, the color models performed better, having a lower error in nearly all comparisons. The exact increase in error varied again per architecture and phase, and it even differed between the color baselines. For the four architectures, in general, the error increase in degrees during training were around 0.4, 0.3, 0.6, and 0.5, and during calibration, these were 0.1, 0.05, 0.5, and 0.4. Interestingly, the all channel baseline (FD0) did not match the color baselines. Instead, it was more equal to the other frequency models and was even bested by these during calibration. Additionally, the YCbCr baseline showed slightly worse results in the AlexNet training phases but was equal or better than the RGB model in other areas. These findings indicate that the different color representation and data format used impacted the accuracy of the model besides just the channel selection. Furthermore, the performance of the dynamic model (FDD7) during training was the best among the frequency models for the AlexNet and good with the ResNet-18. However, its performance after calibration was worse than that of the static selection models. This suggests that the dynamic model has the potential to outperform the static selection models, but it requires more data to achieve these results, as the 100 images provided were not sufficient for calibration.

Lastly, the modified double channel models which were designed to better fit the frequency domain data, did result in ac-

Table 2: Experiment results for the various models tested.

Model		Training		Calibration		Inference	
Architecture*	**Data**	**Time** (seconds)	**Error** (degrees)	**Time** (seconds)	**Error** (degrees)	**Images** (per second)	**Error** (degrees)
AlexNet LC	RGB	434.1±17.3	2.396±0.033	98.0±4.7	3.361±0.108	771.9±1.8	3.291±0.104
	YCbCr	435.8±11.9	2.580±0.045	104.1±4.4	3.389±0.119	664.7±1.0	3.316±0.115
	FD0	229.2±7.9	2.601±0.043	67.4±6.9	3.585±0.138	772.7±2.6	3.500±0.134
	FD1	133.4±14.8	3.277±0.179	25.1±7.3	3.777±0.224	1078.6±35.9	3.686±0.220
	FD2	134.9±8.2	2.828±0.085	24.4±2.9	3.490±0.110	1050.7±8.8	3.408±0.107
	FD3	130.0±4.4	2.793±0.073	24.3±3.8	3.448±0.153	1037.2±76.8	3.367±0.150
	FD4	142.4±11.6	2.676±0.054	26.8±3.7	3.505±0.136	1090.3±1.5	3.423±0.130
	FD5	145.4±10.1	2.732±0.116	28.9±2.4	3.469±0.102	1035.0±26.6	3.389±0.099
	FD6	155.4±16.1	2.657±0.051	38.7±10.7	3.464±0.129	972.9±5.6	3.384±0.128
	FDD7	279.2±14.4	2.594±0.072	75.9±3.8	3.723±0.089	567.2±0.6	3.634±0.084
AlexNet HC	RGB	910.1±9.9	2.334±0.037	157.1±7.3	3.378±0.086	541.3±0.8	3.304±0.081
	YCbCr	915.9±17.4	2.533±0.046	163.4±12.5	3.332±0.132	491.5±0.7	3.260±0.126
	FD0	382.5±17.2	2.575±0.059	98.3±16.8	3.502±0.080	773.0±1.4	3.422±0.077
	FD1	276.1±6.7	3.280±0.094	50.0±13.6	3.661±0.109	970.6±2.3	3.575±0.105
	FD2	280.5±10.1	2.817±0.068	52.5±11.5	3.508±0.130	960.0±7.3	3.426±0.126
	FD3	281.7±11.2	2.789±0.073	52.5±7.8	3.391±0.102	960.0±1.0	3.311±0.099
	FD4	281.8±4.2	2.668±0.068	49.7±3.6	3.437±0.100	963.7±2.1	3.358±0.098
	FD5	306.1±5.7	2.642±0.057	54.0±5.1	3.452±0.099	930.2±15.6	3.372±0.095
	FD6	294.6±7.5	2.625±0.068	47.1±3.7	3.442±0.112	734.1±2.9	3.364±0.106
	FDD7	425.3±10.5	2.510±0.098	98.8±5.1	3.560±0.162	481.5±1.2	3.477±0.157
ResNet-18 LC	RGB	774.7±3.7	1.998±0.103	131.7±1.3	2.879±0.137	366.1±1.2	2.820±0.134
	YCbCr	776.8±5.2	2.023±0.110	137.7±3.0	2.817±0.065	341.2±1.8	2.757±0.062
	FD0	614.7±5.2	2.677±0.115	122.5±7.9	3.423±0.113	374.2±0.8	3.347±0.109
	FD1	513.5±3.0	2.833±0.064	72.1±1.9	3.528±0.076	486.7±7.6	3.455±0.070
	FD2	519.7±1.4	2.792±0.164	75.1±2.4	3.408±0.081	482.8±8.1	3.336±0.074
	FD3	515.8±3.0	2.634±0.136	75.9±5.6	3.514±0.130	486.9±2.0	3.441±0.129
	FD4	517.8±1.9	2.570±0.086	75.7±3.0	3.338±0.128	495.8±0.5	3.264±0.128
	FD5	539.4±2.7	2.681±0.185	82.9±6.2	3.445±0.073	448.3±19.2	3.370±0.070
	FD6	525.8±1.9	2.606±0.124	79.0±5.5	3.441±0.071	457.9±1.5	3.362±0.066
	FDD7	632.7±2.1	2.608±0.092	123.1±2.1	3.454±0.158	318.8±0.3	3.385±0.151
ResNet-18 HC	RGB	1962.0±7.3	2.005±0.147	290.3±6.8	2.768±0.183	307.2±0.6	2.717±0.175
	YCbCr	1963.3±7.3	1.947±0.115	291.7±5.4	2.700±0.086	290.9±0.2	2.651±0.089
	FD0	1726.9±22.6	2.479±0.051	286.6±19.8	3.203±0.084	322.5±0.3	3.136±0.083
	FD1	1629.8±12.3	2.664±0.122	243.3±17.4	3.276±0.119	368.8±0.5	3.209±0.119
	FD2	1642.4±14.0	2.504±0.114	231.1±8.6	3.200±0.100	366.4±0.2	3.129±0.100
	FD3	1623.0±8.9	2.441±0.143	241.1±14.4	3.212±0.106	367.3±0.2	3.143±0.105
	FD4	1638.8±17.2	2.391±0.080	237.3±9.6	3.109±0.140	368.6±0.3	3.044±0.136
	FD5	1646.5±7.0	2.440±0.110	244.9±30.0	3.174±0.130	347.4±9.2	3.107±0.128
	FD6	1630.9±2.4	2.485±0.124	228.1±2.2	3.137±0.070	330.8±0.1	3.071±0.071
	FDD7	1739.2±7.5	2.546±0.110	274.2±4.7	3.334±0.124	257.8±0.1	3.269±0.116

* The regular architecture is indicated by LC, where the double channel modified version is indicated by HC.

curacy improvements. Most models improved with this modification, except for the color models with the AlexNet architecture. However, the improvements achieved were small and would still require a significant speedup over the unmodified color model to be viable. Due to this, the ResNet-18 models can be disregarded as these require over three times more calibration time. In contrast, the modified AlexNet models allowed the frequency models to perform equally to the regular AlexNet color models while still being about twice as fast. However, the improvement of the regular version of this model is only 0.05 degrees, and this gain in accuracy is not worth the double training and calibration times required.

3.3.2 Inference

The inference results showed a more relevant statistics for applying the various techniques tested. In terms of accuracy the results were rather similar to the calibration as these use the same model and mostly the same images. In contrast, the time results, which were also measured in a different format changed slightly due to the absence of a learning component in this phase. The speedups were reduced to only 1.4, 1.8, 1.3, and 1.2, as the lack of a learning component favored the higher input data size models. Furthermore, two models performed worse compared to the training and calibration timings. The largest static selection (FD6) showed a more significant speed gap compared to the other frequency models due to its notable larger input data size. The dynamic model (FDD7) was also slower than before, it was able to process significantly less images per second than all other models. This is presumably due to the dynamic selection part of the model being quite computationally expensive relative to the other models, this effect is also amplified with the batch size of one used during inference.

The inference results are visualized in Figure 3. This figure emphasizes how the AlexNet and frequency domain models are generally faster but less accurate than the ResNet-18 and color models. In contrast, when comparing the AlexNet models it shows how the frequency models are at the same accuracy height while far further on the right than the color models as they can process more images per second. Furthermore, the differences between the static channel selection model are shown in a clearer way. The simpler selections FD1 and FD2 although fast show a worse angular error. The mid sized selections FD3 and FD4 seem to be fast while remaining accurate, together besting the higher selection models FD5, FD6, and FD0. This effect is probably due to the model struggling to handle the high channel data, where less channels can improve the accuracy as shown. Lastly, the dynamic model is shown to perform the worst across all architectures in this experiment.

3.3.3 Optimal selections

Based on the analysis of Table 2 and Figure 3 described in Sections 3.3.1 and 3.3.2, optimal selections for the AlexNet and ResNet-18 architectures were determined. For the AlexNet, static channel selection model FD3 was identified as the optimal choice. This model provided the best accuracy with an error increase of only 0.08 degrees compared to the RGB model. It also achieved speedups of 3.3, 4.0, and 1.35 for training, calibration and inference respectively. These speedups were equal with the best frequency model results except inference, although there it showed a higher standard deviation. Regarding the ResNet-18, the optimal selection was found to be the static channel selection model FD4. Similar to FD3 for the AlexNet, this model achieved the best accuracy which was an error increase of 0.44 over the RGB model. Additionally, the speedups of this model were 1.5, 1.7, and 1.35.

4 Responsible Research

During the course of this research, various responsible research practices were considered. While responsible research often revolves around the ethical aspects of how the data was handled, this is not as big a factor in this research. The dataset contained only facial images of which the information was analyzed and the gaze direction estimated. Since no sensitive information could be extracted via this usage there were no ethical concerns. Other areas of responsible research, such as reproducibility and the usage of large language models do require a more detailed explanation, which is provided in the following sections.

4.1 Reproducibility

Ensuring that the results achieved are reproducible is a critical part of research as it provides transparency and improves the reliability of the work. To facilitate this, a GitHub repository containing the entire codebase used in the research was made publicly available[5]. The repository includes all the code for converting the data, analyzing the data and running models along with explanations on the functionality and instructions to use the system. The combination of the provided repository and this paper should contain sufficient information to repeat all the experiments performed and reach similar results. The only unreproducible aspect may reside in the computational device used. In this research a supercomputer was utilized to run the experiment. A high RAM GPU machine is required to run the full experiment, with results only slightly differing depending on the exact device specifications.

4.2 Usage of Large Language Models (LLMs)

Large language models have recently been on the rise and are capable of assisting in various tasks. Therefore, a further explanation of how they were utilized in this research is important. During this research, ChatGPT[6] was used to assist in coding and writing tasks. The practices explained below were carefully applied to improve the overall quality of the research while keeping the authenticity of human work in there.

When coding, ChatGPT was utilized to improve the efficiency and assist when fixing bugs. Very basic components, such as simple plots or filling in values in a long conditional chain were implemented by the AI following given instructions. The AI generated code was checked carefully and modified where necessary to perform the correct task and prevent errors. This approach saved time and improved work efficiency for simple yet time-consuming tasks. Additionally,

[5] https://github.com/tpenning/DLFDFaceGazeEstimation

[6] https://openai.com/blog/chatgpt

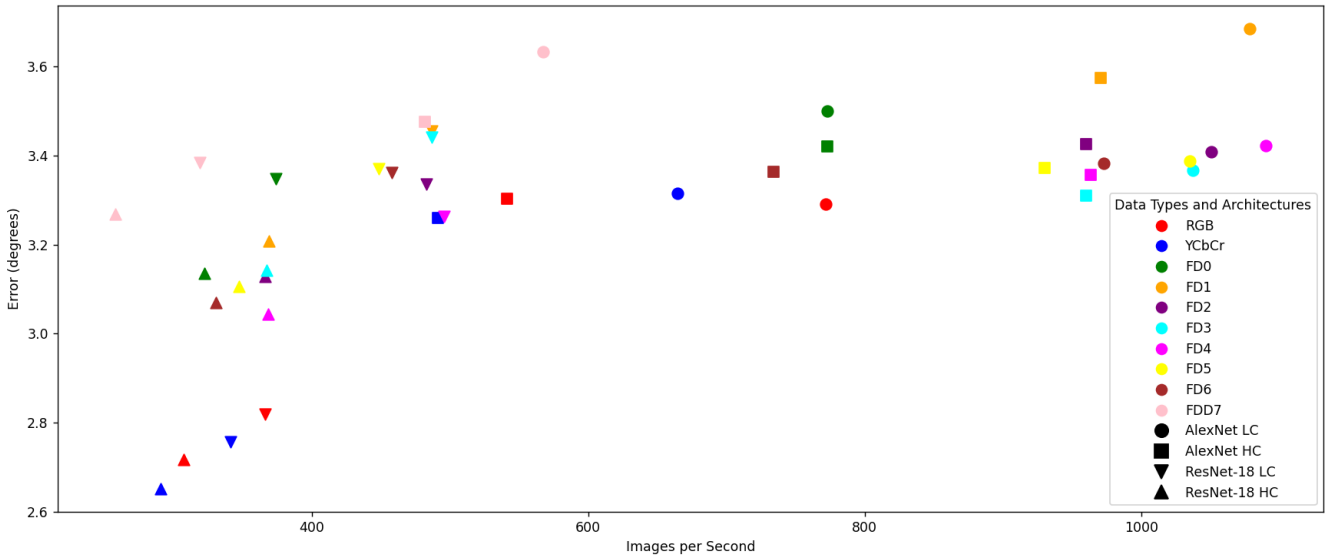


Figure 3: Inference results of the data types combined with the architectures.

during more complex tasks which were coded manually, AI assistance was utilized to analyze error messages and identify the bug causing them in the code.

In terms of writing, ChatGPT was utilized to help adding data visualizations in Overleaf[7] and polish the writing. Initially, the AI was asked to help create the Overleaf code for simple figures and tables. Later figures and tables were made by copying and changing the existing ones, where ChatGPT was consulted in case a very different format was needed. Regarding its help in writing, after sections or chapters were written first, ChatGPT assisted in correcting spelling and punctuation errors. Additionally, it provided suggestions to improve text flow and improve conciseness. The ideas behind these suggestions were selectively applied and modified to fit the style of the writing.

5 Conclusions and Future Work

This paper utilizes the frequency domain combined with channel selection to develop faster deep learning-based gaze estimation models. The primary goal of this research was to study the impact of channel selection on the latency and accuracy of frequency domain gaze estimation. Additionally, the focus was to determine channel selections that would provide an optimal trade-off between maximal speedup and minimal accuracy loss. Various models were created to experiment with and answer the research question. A model using traditional RGB images was designed as a baseline for the other models, while the other models used various channel selections based on static and dynamic methods to test their impact on latency and accuracy.

The experimental results achieved different levels of success for the two network architectures used. In the case of AlexNet, the best channel selection resulted in speedups of 3.3, 4.0, and 1.35 during training, calibration, and inference respectively, with only a 0.08 degrees increase in angular error compared to the RGB model. In contrast, the ResNet-18 architecture resulted in more mixed results. The models for this architecture were still able to achieve speedups of 1.5, 1.7, and 1.35, however, the models also experienced a notable decrease in accuracy, with a 0.44 degrees increase in angular error compared to the RGB model. It is important to note that this accuracy loss was already present before channel selection was applied indicating it was a result of the frequency domain format rather than the channel selection. Therefore, the accuracy still matched that of the AlexNet, and so it was still good, but it did not fully utilize the potential of the ResNet-18 architecture.

The findings of this research revealed various interesting insights. The dynamic channel selection model tested displayed promising results during training. Ultimately it failed during calibration and inference as the amount of data used for calibration was not sufficient for its complexity. In the static channel selection the mid sized selection performed better than the smaller and larger selections. This indicated that the models struggled to handle many channels and a precise selection would be important. The main takeaway from the results is that the frequency domain models clearly resulted in faster networks, but the accuracy accompanying the speedup differed per network architecture. Regarding the accuracy loss observed in the ResNet-18 models, a potential explanation lies in how the networks adapt to the high channel input data. The ResNet-18 has fewer channels throughout the layers than the AlexNet, leading to more data reduction in the first few layers, which could potentially have caused valuable information to get lost. Despite the precise cause, this difference in results demonstrates that, in addition to channel selection, the model structure also plays a crucial role in model performance.

These findings highlight the need for further research focusing on how the network structure influences performance.

[7] https://www.overleaf.com/

An important research question would be how popular gaze estimation networks can be successfully transformed to best fit the channel-selected frequency information. Additionally, research into models specifically designed for channel-selected frequency domain information could provide the solution to fully utilizing the potential of this different data format demonstrated by the findings of this paper.

References

[1] Dan Witzner Hansen and Qiang Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):478–500, 2009.

[2] Jiri Najemnik and Wilson S Geisler. Optimal eye movement strategies in visual search. *Nature*, 434(7031):387–391, 2005.

[3] Bilge Mutlu, Toshiyuki Shiwa, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. Footing in human-robot conversations: how robots might shape participant roles using gaze cues. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 61–68, 2009.

[4] Nachiappan Valliappan, Na Dai, Ethan Steinberg, Junfeng He, Kantwon Rogers, Venky Ramachandran, Pingmei Xu, Mina Shojaeizadeh, Li Guo, Kai Kohlhoff, et al. Accelerating eye movement research via accurate and affordable smartphone eye tracking. *Nature communications*, 11(1):4553, 2020.

[5] Erroll Wood and Andreas Bulling. Eyetab: Model-based gaze estimation on unmodified tablet computers. In *Proceedings of the symposium on eye tracking research and applications*, pages 207–210, 2014.

[6] Yusuke Sugano, Xucong Zhang, and Andreas Bulling. Aggregaze: Collective estimation of audience attention on public displays. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 821–831, 2016.

[7] Dario Cazzato, Marco Leo, Cosimo Distante, and Holger Voos. When i look into your eyes: A survey on computer vision contributions for human gaze estimation and tracking. *Sensors*, 20(13):3739, 2020.

[8] Yihua Cheng, Haofei Wang, Yiwei Bao, and Feng Lu. Appearance-based gaze estimation with deep learning: A review and benchmark. *arXiv preprint arXiv:2104.12668*, 2021.

[9] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. It's written all over your face: Full-face appearance-based gaze estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 51–60, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[15] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[16] Polona Caserman, Michelle Martinussen, and Stefan Göbel. Effects of end-to-end latency on user experience and performance in immersive virtual reality applications. In *Entertainment Computing and Serious Games: First IFIP TC 14 Joint International Conference, ICEC-JCSG 2019, Arequipa, Peru, November 11–15, 2019, Proceedings 1*, pages 57–69. Springer, 2019.

[17] Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016.

[18] Kai Xu, Minghai Qin, Fei Sun, Yuhao Wang, Yen-Kuang Chen, and Fengbo Ren. Learning in the frequency domain. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1740–1749, 2020.

[19] Lingyu Du and Guohao Lan. Freegaze: Resource-efficient gaze estimation via frequency domain contrastive learning. *arXiv preprint arXiv:2209.06692*, 2022.

[20] Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu, and Jason Yosinski. Faster neural networks straight from jpeg. *Advances in Neural Information Processing Systems*, 31, 2018.

[21] K Ramamohan Rao and Ping Yip. *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 2014.

[22] Gregory K Wallace. The jpeg still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.