GazeTransformer: Gaze Forecasting for Virtual Reality using Transformer Networks

Tim Rolff[1,2][0000−0001−9038−3196], H. Matthias Harms, and Frank Steinicke[2][0000−0001−9879−7414] Simone Frintrop[1][0000−0002−9475−3593]

[1] Universität Hamburg, Computer Vision Group, Hamburg, Germany
[2] Universität Hamburg, Human Computer Interaction Group, Hamburg, Germany
{Tim.Rolff, Frank.Steinicke, Simone.Frintrop}@uni-hamburg.de
harmmatthias.harms@gmail.com

Abstract. In this paper, we propose *GazeTransformer*, a transformer architecture for forecasting egocentric gaze points in virtual environments (VEs) presented on immersive head-mounted displays (HMDs). In contrast to previous architectures, we do not rely on information that depends on the application state, but rather focus on data modalities provided by the eye-tracker or sent to the HMD. GazeTransformer allows to forecast multiple types of eye-movements, including saccades and fixations, by creating two unfiltered datasets, using the raw gaze from the eye-tracker for forecasting. Moreover, we analyze six different image encoding backends in their quality to forecast gaze positions. To evaluate the performance of our model, we compared all architectures on the generated datasets. The results show that our architecture with all chosen backends outperforms the current state-of-the-art approaches in forecasting egocentric gaze points in VEs.

Keywords: Virtual reality · Transformers · Gaze Forecasting

1 Introduction

With the recent increase in interest in virtual reality (VR), in particular, driven by hype around the metaverse [9, 23, 53], as training tool for medical experts and manufacturing jobs [22, 49], or as a platform for social interactions and meetings [58, 61], gaze and eye movements of users has shown to be an integral part for several VR technologies and applications [15, 21, 39, 42, 52, 57]. For example, it is still challenging to provide highly realistic and immersive dynamic virtual environments (VE) on mobile VR platforms [35, 68] due to limited GPU performance and battery capacity [68]. Utilizing gaze data from an eye-tracker can be a potential solution for this problem, as the eye has only a small area, the fovea, in which humans perceive a sharp image [26]. With foveated rendering [18, 51, 57] a high-resolution image is only rendered in the foveal region, whereas the rest of the image is rendered with lower resolution and quality, resulting in reduced requirements on GPU and battery performance. Besides foveated rendering, there

are several other proposed methods that try to solve reoccurring challenges of VR using information provided through gaze data, such as, redirected walking by performing changes to the VE during saccades or eye blinks [42, 67], gaze-contingent rendering [39, 52], gaze behavior analysis [64], or content compression [64]. Many of these solutions rely on knowledge about the human gaze to alter the VE without user detecting such manipulations. For instance, the duration of saccadic movements range from 30 to 80 ms [26] where the visual input is usually suppressed for >100ms [11], such that the user will not notice slight changes to the VE. However, it may take up to 50ms to detect saccades [66], the possibility to predict them would greatly improve applicability of suppression techniques. Hence, knowledge about the user's eye gaze has enormous advantages, as it can be utilized for the aforementioned algorithms.

A commonly utilized method to capture the gaze for the previously listed methods is by utilizing a video-based eye-tracker built in into the head-mounted displays (HMD). These eye-trackers, however, can only report historic gaze data, as the output of them often has a latency of several milliseconds due to the required preprocessing [1, 66]. This latency is often due to the eye-tracker itself or due to applied algorithms when estimating the gaze. While there are recent attempts to reduce latency through the proposal of hardware-based solutions [2, 43], the latency of eye-trackers in widespread commercially available HMD's often hinders the direct usage of gaze data for VR applications, especially for fast eye-movements such as saccades or blinks. Therefore, directly utilizing the gaze data in downstream tasks might miss such eye movements or requiring unnatural actions, such as intentional blinking [42] or long saccades [1, 67].

As a software-based solution, Hu et al. [27, 29] proposed a neural network for forecasting the gaze of individual users. They define gaze prediction as a multi-modal time series prediction problem, utilizing past gaze, head velocities, object positions, and saliency, to predict gaze positions. While [29] focuses more on the prediction of current gaze positions under the assumption that no information on past gaze data is available, they additionally perform ablation studies, showing that their method can forecast gaze positions when given information about the user past gaze. In [27], they extend this work by predicting future fixation points using a pre-filtered dataset, but do not evaluate on other common eye-movements, such as saccades or smooth pursuits.

We build upon this idea by proposing a different architecture for gaze prediction. In our case, we do not only focus on gaze fixations, but rather include further important types of eye-movements: saccades, fixations and smooth pursuits [26]. For the prediction, we propose the utilization of transformer networks that have shown recent success in multiple tasks, such as natural language processing [4, 12, 13, 70] or image classification [7, 14, 38]. As Hu et al. [27, 29] only analyzed saliency as an additional data modality, we will evaluate six different image modalities through pretrained networks, including saliency, RGB and grayscale images, or the output of ResNet.

To summarize, our work addresses the following contributions:

- We propose *GazeTransformer*, a state-of-the-art transformer architecture for egocentric gaze forecasting in VR handling different eye-movements, like fixations or saccades from raw gaze data.
- We analyze six different image processing techniques and backends, such as saliency, grayscale and RGB images, DINO [7] or ResNet [24] on their ability to predict the task at hand. We optimize *GazeTransformer* for each backend independently to find a set of good hyperparameters.

The paper is structured such that we will first explain related work in the upcoming section. Afterwards, Sec. 3 will describe our method in detail, containing explanations on input representation and architecture. Next, we will go over our results along with the utilized datasets and metric in Sec. 4. At last, we will discuss those results in Sec. 5 along with limitations and future work.

2 Related Work

In this section, we will provide an overview about previous work on visual attention and visual saliency, as well as gaze prediction and gaze forecasting.

2.1 Visual Attention & Visual Saliency

Human gaze is generally considered to be controlled by mechanisms of visual attention, which drive the human visual system to focus on regions of general interest [56]. This attention can computationally be modelled through saliency methods, which are generally divided into two categories of *bottom-up* and *top-down* attention.

Bottom-up refers to the visual attention based on low-level image features, such as color, shape or contrast [10]. In fact, earlier computation saliency models compute several feature maps from low-level features and fuse them together for the final saliency prediction [19, 32]. Some of these models are based on work by Treisman and Gelade [69], who theorized that visual features are registered in parallel early in the vision process, whereas objects are formed in later stages from the collected features. More recent approaches rely on modeling fixations through the help of deep-learning [3] by capturing the datasets either through the use of eye-trackers [5, 72] or by recording mouse input [34]. A recent trend for these deep-learning-based saliency predictors is the utilization of transfer learning [33, 46] that employing pretrained backend networks, mostly trained for image classification, such as ResNet [24], NASNet [74] or VGG [63].

Top-down is, in contrast to *bottom-up*, driven through high-level features, such as task-specific information [10]. Here, multiple works have shown that gaze is heavily influenced by task or context information that drives the attention. This also includes prior knowledge or instructions, with a study by Yarbus [73] showing considerably different eye-movements depending on the information about the same scene supplied beforehand.

2.2 Gaze Prediction & Gaze forecasting

Gaze prediction describes the process of predicting the next gaze positions, based on a set of data points. These data points can either be task information [27], mouse input [41], hand positions [44], or close up images of the eye [50]. This problem closely relates to visual saliency. However, although gaze and saliency prediction are similar problems, they both differ in that gaze prediction describes the process of predicting the gaze point of individuals given a set of input features, whereas saliency maps tries to model the general distribution of fixation points. Gaze prediction has been researched for sometime on a variety of different applications, such as in the use of action recognition [44], in 360° videos [54, 71], or in video games [41]. However, the research on gaze prediction in VR is fairly recent, with prior work focusing on multiple aspects, like dataset collection, architecture proposal, and data analysis [29, 30].

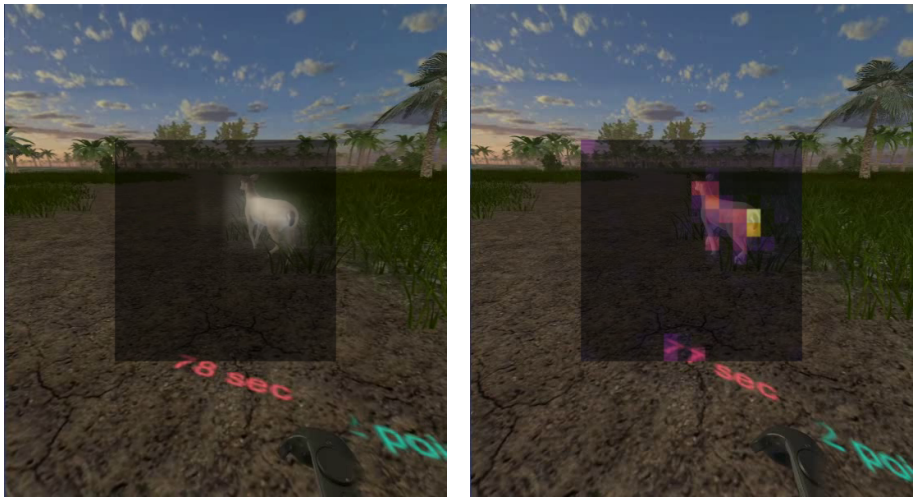
Gaze forecasting can be interpreted as an extension of gaze prediction. While gaze forecasting still outputs a gaze point like gaze prediction, the gaze points should be predicted several milliseconds into the future. This forecasting allows mitigating often found challenges with eye-trackers in commercial HMD's that often only support low frequency update rates [66]. This lead to other research studies relying on low latency gaze data in VR to mitigate those challenges by intentional blinking [42] or relying on long saccade durations [1, 67]. Another recently proposed approach to mitigate latency was proposed by Rolff et al. [59]. They estimate when future gaze shifts will occur by performing time-to-event analysis on each captured sample, rather than predicting the gaze points directly. However, this approach does not report gaze points that can potentially be used in other downstream tasks.

3 GazeTransformer

In this section, we will give an overview of our *GazeTransformer*. We start with an overview of the input representation (Sec. 3.1) and present afterwards the proposed architecture for the gaze forecasting transformer (Sec. 3.2).

3.1 Input Representation

As gaze datasets often provide different data modalities, such as gaze positions, head orientations, IMU data, videos, depth, EEG, or task data [17, 20, 25, 27, 29, 36, 40, 45], our model should be able to handle those different data modalities. Therefore, it is required to represent the input, such that the model can interpret it. Fortunately, most of the listed data modalities can either be represented as images or as a temporal sequence, containing an individual feature vector for each time step. Since the aforementioned datasets provide different representations, we choose to represent each sample point as a concatenation of different data modalities, with each modality having a fixed length. As most modalities like gaze or IMU data are already captured as a sequence, we can directly utilize those



(a) Saliency map, generated through the saliency backend described in Sec. 3.1.

(b) Attention scores generated through the DINO backend described in Sec. 3.1.

Fig. 1: Visualization of the different backends described in Sec. 3.1 overlaid on the original image. These are used as input for the transformer model explained in Sec. 3.2. Note that we only use the center cropped part for saliency and attention score prediction.

for our model. Here, we focus on four different modalities: the horizontal and vertical gaze, the head velocity, and depending on the dataset, the current task and the last rendered frames. For image data, we employ an additional feature extraction network that processes a sequence of frames to extract 1-dimensional vectors first. We explicitly choose these data modalities as they are provided from or, in case of frames, to the HMD without requiring information about the internal application state, like object positions [29], The only exception is task information, but recent work [28] has shown that it is possible to infer the task information from the eye-in-head, gaze-in-world and head data without requiring the application state. In general, we provide the network with the last 400ms of input samples to forecast the sample 150ms ahead. As a result, we can describe our input I as follows:

$$I = \begin{pmatrix} d_{t-39} \\ d_{t-38} \\ \vdots \\ d_t, \end{pmatrix}, \text{ with } d_{t-i} = \left(\text{gaze}_{t-i}, \text{head}_{t-i}, \text{task}_{t-i}, \text{frame}_{t-i}\right), \tag{1}$$

where $i \in [0, \ldots, 39]$, and t denotes the index of the last captured sample. To extract 1-dimensional feature representations from the input frames, we will an-

alyze the following image-to-sequence backends:

Grayscale: As a baseline image method, we flatten a grayscale image of the input into a 1-dimensional vector. This restricts the input to images of the same size, as otherwise spatial information between image pixels are lost. Furthermore, if the chosen image size is too large, we hypothesize that the network might not pick up information on the other modalities. Thus, we transform the input frames to 32×32 grayscale images, resulting in 1024 features for each sample. Here, we concatenate the full 1-dimensional flattened vector onto each step in the sequence, as formulated in Eq. 1, differing from previous approaches, like [8], as they use the individual pixel values of the image as the input sequence.

Patch: To preserve the color information of the image, we transform the image into a singular 64×64 patch, similar to Dosovitskiy et al. [14]. In [14], they generate input vectors for their transformer architecture by dividing the original image into multiple patches. Afterwards, each patch is flattened and processed as part of multiple patches. However, due to the chosen representation of the input, as formulated in Eq. 1, this is not feasible. Instead, we resize the input image down to a singular 64×64 pixel image patch, conserving the color information, resulting in a 12288-dimensional feature representation of the input.

Saliency: Since saliency is used by multiple gaze prediction and forecasting methods [27, 29–31, 71], we also analyze the use of saliency as input modality. To generate the saliency map of each frame, we employ the approach proposed by Jia and Bruce [33], acquiring the information on the fixation distribution of the input image. Other work by Einhäuser and Nuthmann [16] has shown that the usage of saliency data is beneficial as they correlate with fixation durations. This correlation might potentially be valuable in case of an upcoming saccade, as it might contain information on the duration about the currently performed fixation. However, as we only provide saliency, the latter network layers might have no awareness of the original content of the image, therefore, losing information on color, intensity, or shape. As shown in Fig. 1a, we follow the approach of Hu et al. [27, 29] and generate the saliency maps from the center-cropped region of each frame and resizing the output to 24×24 pixels.

ResNet: A different approach for the generation of sequence data from images was explored by Carion et al. [6]. They evaluated multiple ResNet [24] architectures by prepending the ResNet module in front of a transformer for feature extraction. In our work, we extend their idea by utilizing a ResNet50 architecture that was pretrained on ImageNet [60]. First, we extract the image features for each input image and pass the extracted features into the transformer architecture. However, to avoid output that fully resembles the class distribution of in the input image, we discard the final classification layer, resulting in a feature vector of 2048.

DINO: As the last evaluated backend, we utilize a vision transformer that is trained through DINO [7], a recently proposed self-supervised training approach. They interpret self-supervised learning as a form of self-distilation. They train a student and a teacher model, where the teacher model is provided with the full

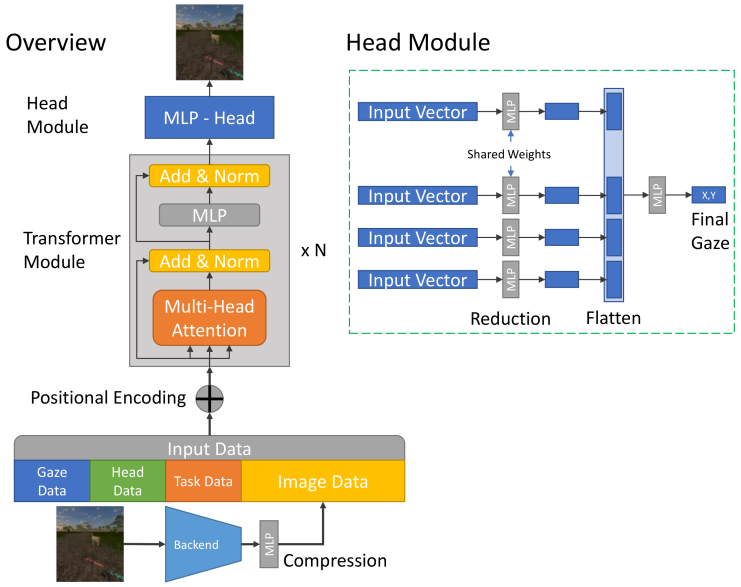


Fig. 2: Overview of our *GazeTransformer* architecture. A backend network extracts a 1D feature vector from the input frames and concatenates it with other data. Positional encoding is concatenated, and the data is fed into multiple (N) transformer encoder layers. A head module, consisting of multiple feedforward layers, reduces the dimensionality of the features and performs the final prediction of gaze points (see Sec. 3 for details).

image and the student model with a randomly augmented and cropped region of the same image. To perform gradient back propagation, they compute the loss as the negative log likelihood between the student and the centered output of the teacher model and propagate the error back through the student network. Then the exponential moving average of the students gradients is used to update the weights of the teacher network. They have shown that the attention weights of the transformer architecture closely correlates with semantic segmentation. Hence, we use the inner attention state, depicted in Fig. 1b, of a pretrained model as the input to our transformer, resulting in feature vectors of size 376.

3.2 Transformer Model

We base our architecture on the popular transformer architecture proposed by Vaswani et al. [70]. For clarity, we divide our proposed architecture into three modules, see Fig. 2 for an overview. The first module is the input encoder module that constructs the input representation for later modules (cf. Sec. 3.1). Next, the transformer encoder module utilizes the transformer architecture of [70] by stacking multiple transformer layers. At last, the head module is used for dimensionality reduction and the final gaze forecasting.

Table 1: Configurations used to report final results. *Heads* and *layers* refer to the number of heads and layers in the transformer layers. *Compress* refers to the number of units in the linear layer that are employed for the compression of backend features. *Reduce* refers to the number of units in the reduction layer of the head module. The name of the model refers to the used backend.

DGaze [29] dataset

Backend	Grayscale	Patch	Saliency	ResNet	DINO	No Backend
Heads – Layers	4 – 1	4 – 1	2 – 1	4 – 1	6 – 1	6 – 4
Compress – Reduce	16 – 128	16 – 64	16 – 128	16 – 128	16 – 128	/ – 64

FixationNet [27] dataset

Backend	Grayscale	Patch	Saliency	ResNet	DINO	No Backend
Heads – Layers	8 – 1	1 – 1	6 – 1	4 – 1	6 – 1	4 – 4
Compress – Reduce	256 – 256	32 – 256	32 – 256	128 – 256	32 – 128	/ – 256

Before providing the data to the transformer model, we first process frames through one of the backend networks listed in Sec. 3.1. These features are further compressed using a feedforward layer followed by a ReLU [55] and a dropout layer [65]. These result in 1-dimensional feature representations of the processed images that we concatenate with the other data samples. Here, we provide the network with the last 400ms of previous gaze positions, head velocities and task-related object information, following the previous approach of [27]. This leads to an input of 40 1-dimensional feature vectors. After the construction of the input data from the different modalities, we perform positional encoding, allowing the transformer model to distinguish between different time-steps of the input.

As we expect the different input data modalities to have different sampling rates, the implementation of our positional encoding is based on [37]. They propose a trainable model-agnostic representation of time that also depends on the input and which is concatenated with a scalar interpretation of time. The entire encoded sequence is fed into the encoder model consisting of multiple stacked transformer layers. In addition, we follow previous work [4, 14] which has shown that a decoder model is not required. Note that the number of utilized layers and transformer heads depend on the backend, see Tab. 1 for more details.

Lastly, the head module consists of a multi-layer perceptron network that predicts the final visual angle from the screen center. As the transformer architecture does not reduce the sequence length, it outputs the same number of 1-dimensional feature vectors that are input into the architecture. Thus, we reduce the dimensionality through a feedforward layer in the head module with the number of units listed in Tab. 1. Afterwards, we reduce the number of features to two, resulting in an output of 40×2. To apply a non-linearity, we additionally add a ReLU and a dropout layer after the output of both linear layers. After-

wards, we flatten the outputted matrix into a 1-dimensional vector that we then utilize to forecast the final horizontal and vertical gaze position.

For training the network, we follow the approach of Hu et al. [27], using the angular error as the loss function, which is defined as the mean angular difference between the line of sight of the ground truth and the forecasted gaze position. For the hyperparameters and optimizer, we choose to employ the AdamW [48] with a learning rate of 0.001, a batch size of 256, dropout of 0.1, and the weight decay set to 0.01. While training, we monitor the loss on the validation set and use early stopping after three epochs, if the loss has not decreased. To find a set of good of hyperparameters for layers, transformer heads, compression, and reduction units for each backend separately, we evaluate our proposed architecture on multiple configurations that can be found in the ablation studies, found in the supplementary. We did not perform additional hyperparameter optimization for other parameters, such as batch size or weight decay. Then we utilize the best performing configuration as our parameters, which are reported in Tab. 1. For further details on these results and the evaluated parameters, see the supplementary material.

4 Evaluation

4.1 Metric

For the error metrics, we follow Hu et al. [27], by using the angular error as our primary metric that describes the angular distance between the ground truth line of sight and the predicted line of sight, with a smaller value indicating better performance.

4.2 Datasets

We evaluate our model on two different egocentric datasets fully captured in VR, namely the *DGaze* [29] and *FixationNet* [27] dataset. For capturing the first-mentioned *DGaze* dataset, each participant was asked to freely explore 2 out of 5 randomly assigned VEs containing different dynamic visual distractors in the form of animals. For the recording, each participant was instructed to record at least 3 minutes of data without any further instructions on a task or explanation of the environment. In total, the dataset contains 86 samples from 43 participants with an average sequence length of over 20,000 data points per session. The *FixationNet* dataset captures the same data modalities as the *DGaze* dataset, but additionally provides information on the current task of each participant. Here, all participants were instructed to solve a specific search task, pointing the VR controller onto the target. In total, the *FixationNet* dataset contains 162 samples from 27 participants containing on average 12,000 head and gaze points per trial.

However, as the evaluated task of Hu et al. [27] is the prediction of fixation points, the authors provide a pre-processed dataset with filtered gaze positions

Fig. 3: Sequence of frames showing a prediction of *GazeTransformer* (red dot) on the FixationNet dataset [27] along with the ground truth (green dot) and the last captured gaze position (blue dot). The frames are cropped to the relevant region. As participants were tasked to search for targets, the left animal disappears due to its correct classification.

that correspond to the fixation points of the participants. Therefore, we generate a custom dataset from the raw data of both datasets that do not compute fixation points and therefore account for other eye-movements. We also follow the same methodology for the *DGaze* dataset. Both datasets contain videos, gaze, head velocities captured through the IMU of the HMD, and object positions of the nearest objects. In these generated datasets, we made sure not to utilize any internal information only known to the running application, such as object positions, but rather focused on the information already provided from or to the HMD, with the only exception being the task information. As both datasets do not contain gaze events, we follow the methodology of [59] to generate fixation and saccade events, using the I-VT algorithm [62].

4.3 Results

Tab. 2 shows the result of our proposed *GazeTransformer* architecture with all backends introduced in Sec. 3.1. As described in Sec. 4.2, we evaluate our approach against the state-of-the-art *DGaze* and *FixationNet* dataset. As the baseline, we choose the last known gaze position that is lagging the target by 150ms. For the evaluation, we choose a cross validation approach by validating across users and scenes. For computational reasons, we only evaluate across the data of 3 participants as the test sets and used the rest either as our training or validation data and compute the final results as the mean over all folds.

Overall, Tab. 2 shows that all selected backends outperform the baseline, as well as the methods introduced by Hu et al. [27, 29]. When utilizing the image data provided by a backend network, we found that the *Patch* backend shows the

Table 2: Results for the FixationNet [27] and DGaze datasets [29] using the raw gaze as input. We name the models of *GazeTransformer* according to their image-to-sequence backend. The error metric is computed through the angular error (*Mean*: mean angular error; *Std.*: mean standard deviation over different folds). Best results in **blue bold**.

	DGaze dataset				FixationNet dataset			
	Cross-User		Cross-Scene		Cross-User		Cross-Scene	
Model	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
Baseline								
Current Gaze	5.12°	8.00°	5.85°	9.77°	3.67°	7.17°	3.66°	7.01°
State-of-the-art								
DGaze [29]	9.58°	6.87°	10.24°	7.65°	8.66°	6.89°	8.76°	6.80°
FixationNet [27]	9.58°	7.37°	10.57°	7.70°	8.49°	6.71°	8.56°	6.76°
Gaze Transformer								
Grayscale	4.91°	6.80°	5.27°	7.76°	3.63°	6.07°	3.55°	6.06°
Patch	4.75°	**6.75°**	5.09°	7.77°	3.49°	**6.05°**	3.49°	6.09°
Saliency	4.85°	6.77°	5.20°	7.75°	3.49°	6.07°	3.51°	**6.04°**
ResNet	4.84°	6.76°	5.16°	7.77°	3.58°	6.06°	3.49°	6.09°
DINO	4.81°	6.76°	5.15°	7.76°	3.55°	6.06°	3.61°	**6.04°**
No Backend	**4.71°**	**6.75°**	**5.07°**	**6.89°**	**3.47°**	6.10°	**3.44°**	6.39°

best performance. Surprisingly, we discovered that utilizing image data does not improve the predictive performance, as the model that does not use frame data performs the best on all datasets. The reported mean standard deviation across all methods is similar, regardless of the employed backend. Surprisingly, the standard deviation of the best performing *GazeTransformer* is also not the most optimal among all our *GazeTransformers*. As mentioned in Sec. 3, we performed multiple ablation studies to find a good set of parameters for our architecture, dependent on the used backend. For more information on these results, see the supplementary material. Tab. 3 also shows the separate errors for fixations and saccades between the current state-of-the-art and our *No Backend* model. Here, we compute the error metrics on saccades and fixations separately using the estimate gaze event label.

Additionally, we performed a qualitative evaluation on the output of *GazeTransformer* and FixationNet. Here, we discovered that the performance of FixationNet is due to its design choice to only predict fixations and often shifts towards salient regions. It is unable to handle strong gaze shifts, even though the network was retrained on our dataset. In contrast, although not instantaneous with the target, our model can perform these shifts by reacting faster than the baseline, with an example shown in Fig. 3.

When running in inference, we achieve real-time performance running at approximately 329 predictions per second when using the *No Backend* model on

Table 3: Individual results for saccades and fixations measured on the FixationNet [27] and DGaze datasets [29] using the raw gaze as input. The error metric is computed through the angular error seperatly across all saccades and fixations (*Mean*: mean angular error; *Std.*: mean standard deviation over different folds). Best results in **blue bold**.

		DGaze dataset				FixationNet dataset			
		Cross-User		Cross-Scene		Cross-User		Cross-Scene	
	Model	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
Saccades	*State-of-the-art*								
	DGaze [29]	12.81°	10.94°	14.81°	**11.07°**	12.86°	**9.86°**	12.96°	9.96°
	FixationNet [27]	14.45°	**9.56°**	15.06°	11.17°	12.91°	9.91°	13.03°	**9.93°**
	Gaze Transformer								
	No Backend	**10.80°**	10.21°	**11.77°**	12.26°	**9.96°**	10.57°	**9.93°**	10.57°
Fixations	*State-of-the-art*								
	DGaze [29]	9.17°	5.45°	9.00°	5.35°	7.85°	5.80°	7.94°	5.84°
	FixationNet [27]	9.18°	5.43°	9.20°	5.46°	7.67°	5.56°	7.72°	5.58°
	Gaze Transformer								
	No Backend	**3.02°**	**4.01°**	**2.96°**	**3.80°**	**2.23°**	**3.64°**	**2.20°**	**3.57°**

the GPU. To measure the timings, we use a system equipped with an AMD Ryzen Threadripper 3960X with 128 GB RAM and an NVIDIA RTX 3090 and performed the execution on the GPU. The other timings, including CPU performance metrics, can be found in the supplementary. When estimating the run-time, we included the full backend and predict image features twice per second.

5 Conclusion and Discussion

In this paper, we proposed *GazeTransformer* a state-of-the-art transformer architecture for forecasting gaze points in VR using the raw gaze data and IMU data provided by the HMD's with built-in eye-trackers. For the prediction of future gaze points, we only utilize data provided from or send to the HMD, without dependencies on the internal state of the application through object locations except for task data, which itself has shown to be predictable without the application state [28]. We forecast the gaze positions 150ms into the future, allowing us to compare our method against existing literature [27, 29]. To evaluate our proposed architecture, we analyzed its ability on two state-of-the-art VR datasets. Furthermore, we analyzed multiple image backends, such as grayscale or RGB images, saliency, attention weights of DINO, and ResNet. Overall, we discovered that our approach, regardless of the backend, significantly outper-

formed all previous state-of-the-art methods when using the raw gaze as input to the network.

Surprisingly, we found that the best performing approach did not utilize image data. This contrasts with previous literature [27, 29], which found that utilizing the saliency directly does improve the final performance of the network, or with other work heavily relying on RGB data of captured frames [44]. There are several potential reasons for this finding. First, some utilized backends, like ResNet, might not provide meaningful information to the transformer layers and head module. This is since ResNet was pretrained for image classification, and therefore high-level features will most likely contain information on the estimated class. Another reason might be that only two frames are used over the entire input sequence. As the input of *GazeTransformer* is split by time instead of splitting by features, like [29] or [27], this causes the duplication of frames that might potentially result in over weighting the image features, as the size of the other modalities is significantly smaller than the image feature space. Moreover, due to memory requirements and analysis on the different backends, we did not train the backend networks in combination with the transformer architecture. This might have further impacted the performance of the image-dependent architectures. At last, this may also be due to the rather small dataset size, as transformers rely on huge datasets when trained from scratch to capture meaningful correlations between features [47]. Given this, we also expect that the need to use different hyperparameters for each data set to achieve optimal performance will be eliminated if the data set is large enough. However, we also suspect that conducting a final calibration phase for each user before using a pre-trained model is a worthwhile research direction, as it could lead to a more optimal model.

Investigating these observations maybe valuable directions for future work, as we expect the image data to have a positive impact on the final performance of the architecture, even though we could not confirm this in our paper. Here, training these backend networks may be a good initial step on verifying if performance can further be improved when image modalities are used. This, however, might be bound by the memory of the GPU. Therefore, another direction might be to analyze other virtual and real-world datasets that are similar to the data in the datasets commonly used for pre-training, such as ImageNet, to make better use of the pre-trained networks. With those, it would also be possible to explore additional image modalities, for example depth data or EEG input. Besides, extending the architecture to directly work with image data that is split by time as well as by feature would remove the need for duplicated input frames. Moreover, adding multi-horizontal forecasting, to predict multiple future short-term and long-term gaze points would be helpful, as we expect the network to perform better on shorter forecast durations. Besides the surprising results on image modalities, we found that *GazeTransformer* significantly outperforms the state-of-the-art regardless of the backend utilized.

6 Ablation Studies

As mentioned in our paper, we optimize the general architecture of *GazeTransformer* to find reasonable hyperparameters for each model, further described in Sec. 3 of our Paper, we perform multiple ablation studies on both the DGaze [29] and the FixationNet [27] dataset. To avoid biasing GazeTransformer for a specific input modality, we perform the hyperparameter optimization on all image backends: *Grayscale*, *Patch*, *Saliency*, *ResNet*, *DINO*, as well as reference network without any backend (*No Backend*). We use the angular error metric as in our paper to report the results below. Since, the different modules depend on each other, we decided to estimate the parameters in the following order: First the estimation of units in the reduction layer (cf. Tab. 4 and Tab. 5). Note that we cannot report results for the *No Backend* model, as it does not contain a *Compression* module. Afterwards, number of units in the compression layer (cf. Tab. 6 and Tab. 7). At last, the number of transformer heads and layers (cf. Tab 8 and Tab. 9). To construct the final networks, reported in the paper, we take the best performing configuration of each analyzed module and retrain the architecture as explained in the paper. We also perform additional evaluations on the optimal input resolution when using the *Flatten* and *Patch* backend (cf. Tab. 10). Tab. 11 shows the measured run-time of our model on CPU and GPU using the system described in the paper. Showing that our model is able to run in real-time when run on the CPU, as the GPU is might be highly utilized due with the rendering process.

Table 4: Results for the number of hidden features in the *Compression* layer, measured on the DGaze dataset. Best results in **blue bold**.

Compression Units	Grayscale	Patch	Saliency	ResNet	DINO	No Backend
16	**5.59°**	**5.49°**	**5.56°**	**5.66°**	**5.51°**	–
32	5.73°	5.56°	5.65°	5.74°	5.56°	–
64	5.84°	5.55°	5.62°	5.69°	5.59°	–
128	5.78°	5.59°	5.63°	5.67°	5.52°	–
256	5.67°	5.64°	5.68°	5.68°	5.75°	–

Table 5: Results for the number of hidden features in the *Compression* layer, measured on the FixationNet dataset. Best results in **blue bold**.

Compression Units	Grayscale Patch		Saliency ResNet		DINO	No Backend
16	**3.47°**	3.40°	3.40°	3.48°	3.44°	–
32	3.48°	**3.38°**	**3.37°**	**3.43°**	**3.38°**	–
64	**3.47°**	3.45°	3.40°	3.44°	3.40°	–
128	3.51°	3.40°	3.40°	3.56°	3.44°	–
256	3.47°	3.57°	3.42°	3.58°	3.43°	–

Table 6: Results for the number of hidden features in the *Reduction* layer, measured on the DGaze dataset. Best results in **blue bold**.

Reduction Units	Grayscale Patch		Saliency ResNet		DINO	No Backend
16	5.81°	5.61°	5.65°	5.83°	5.61°	5.54°
32	5.87°	5.62°	5.63°	5.75°	5.71°	5.55°
64	5.76°	**5.50°**	5.61°	5.82°	5.64°	**5.52°**
128	**5.71°**	5.54°	**5.56°**	**5.67°**	**5.51°**	5.59°
256	5.72°	5.59°	**5.56°**	10.58°	5.66°	5.64°

Table 7: Results for the number of hidden features in the *Reduction* layer, measured on the FixationNet dataset. Best results in **blue bold**.

Reduction Units	Flatten Patch		Saliency ResNet		DINO	No Backend
16	3.50°	3.43°	3.46°	3.50°	3.46°	3.34°
32	3.48°	3.44°	3.49°	3.58°	**3.39°**	3.34°
64	3.48°	3.41°	3.41°	3.49°	3.49°	3.44°
128	3.51°	3.41°	3.43°	3.43°	3.44°	3.37°
256	**3.39°**	**3.40°**	**3.40°**	**3.41°**	**3.39°**	**3.31°**

Table 8: Results on the DGaze dataset using different layer and transformer head configurations. *Heads* describes the number of attention heads, while *Layers* describes the number of stacked transformer encoder layers. Invalid configurations are marked with "-", as the embedding dimension must be divisible by the number of heads. Best results in **blue bold**.

Heads – Layers	Grayscale	Patch	Saliency	ResNet	DINO	No Backend
1 - 1	6.73°	5.86°	5.75°	8.08°	6.30°	5.63°
1 - 2	8.07°	8.16°	7.63°	10.60°	6.08°	5.59°
1 - 4	10.61°	12.19°	10.57°	11.97°	12.18°	5.53°
2 - 1	10.55°	5.98°	**5.69°**	7.71°	6.02°	5.62°
2 - 2	7.90°	7.69°	5.89°	10.57°	6.04°	5.61°
2 - 4	10.57°	12.33°	10.56°	12.18°	10.56°	5.51°
4 - 1	**6.15°**	**5.76°**	5.89°	**7.31°**	5.96°	5.65°
4 - 4	12.11°	10.57°	10.57°	12.09°	7.66°	5.48°
4 - 6	10.55°	10.58°	12.29°	10.61°	10.60°	**5.46°**
6 - 1	—	5.89°	5.91°	—	**5.93°**	5.57°
6 - 4	—	12.06°	12.02°	—	7.37°	**5.46°**
6 - 6	—	10.56°	12.17°	—	12.14°	5.50°

Table 9: Results on the FixationNet dataset using different layer and transformer head configurations. *Heads* describes the number of attention heads, while *Layers* describes the number of stacked transformer encoder layers. Invalid configurations are marked with "-", as the embedding dimension must be divisible by the number of heads. Best results in blue bold.

Heads – Layers	Grayscale	Patch	Saliency	ResNet	DINO	No Backend
1 - 1	7.21°	**3.47°**	3.67°	7.42°	3.73°	3.39°
1 - 2	4.94°	5.06°	5.05°	8.50°	7.65°	3.40°
1 - 4	8.49°	8.50°	8.49°	8.53°	8.48°	3.32°
2 - 1	4.78°	3.54°	3.69°	7.38°	4.20°	3.39°
2 - 2	7.42°	4.88°	4.79°	8.48°	4.17°	3.36°
2 - 4	8.48°	8.50°	8.50°	8.48°	8.47°	**3.29°**
4 - 1	4.32°	3.97°	3.77°	**4.62°**	4.09°	3.39°
4 - 4	8.47°	8.49°	8.50°	8.50°	8.49°	3.31°
4 - 6	8.50°	8.49°	8.49°	8.52°	8.50°	3.37°
6 - 1	—	3.70°	**3.55°**	—	**3.69°**	3.49°
6 - 4	—	8.49°	8.51°	—	6.26°	3.34°
6 - 6	—	8.49°	8.49°	—	8.50°	3.34°
8 - 1	**4.22°**	8.48°	3.73°	**4.62°**	3.92°	3.39°
8 - 4	8.53°	8.51°	8.52°	8.51°	8.51°	3.36°
8 - 6	—	8.50°	8.52°	8.51°	8.49°	3.32°

Table 10: Results for different resolutions when using the *Flatten* and *Patches* backends evaluated on the DGaze and the FixationNet datasets. *Resolution* describes the number of pixels per axis. *OOM* stands for *out of memory*, as our system had not enough space to handle a sufficient batch size. Best results in **blue bold**.

	DGaze dataset		FixationNet dataset	
Resolution	Grayscale	Patch	Grayscale	Patch
16×16	5.81°	5.84°	**3.49°**	3.53°
32×32	**5.68°**	5.80°	3.50°	3.52°
64×64	5.70°	**5.60°**	3.50°	**3.48°**
128×128	5.74°	OOM	3.60°	OOM

Table 11: Measured run-time of our proposed architecture on all backends using the system described in Sec. 4.3 of the paper. The arrows indicate if a higher or lower value is desired. Best results in **blue bold**.

	Grayscale	Patch	Saliency	ResNet	DINO	No Backend
GPU						
Predictions per second$_\uparrow$	**1137.72**	1086.77	886.60	981.91	388.01	329.65
Processing time in ms$_\downarrow$	**0.88**	0.92	1.13	1.02	2.58	3.03
CPU						
Predictions per second$_\uparrow$	**926.41**	200.42	429.43	350.32	14.95	372.38
Processing time in ms$_\downarrow$	**1.08**	4.99	2.33	2.85	66.87	2.69

Bibliography

[1] Albert, R., Patney, A., Luebke, D., Kim, J.: Latency requirements for foveated rendering in virtual reality. ACM Transactions on Applied Perception (TAP) 14(4), 1–13 (2017)

[2] Angelopoulos, A.N., Martel, J.N., Kohli, A.P., Conradt, J., Wetzstein, G.: Event-based near-eye gaze tracking beyond 10,000 hz. IEEE Transactions on Visualization and Computer Graphics (TVCG) 27(5), 2577–2586 (2021)

[3] Borji, A.: Saliency prediction in the deep learning era: Successes and limitations. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 43(2), 679–700 (2019)

[4] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. Advances in Neural Information Processing Systems (NeurIPS) 33, 1877–1901 (2020)

[5] Bylinskii, Z., Judd, T., Borji, A., Itti, L., Durand, F., Oliva, A., Torralba, A.: Mit saliency benchmark (2015)

[6] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 213–229. Springer (2020)

[7] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9650–9660 (2021)

[8] Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., Sutskever, I.: Generative pretraining from pixels. In: Proceedings of the 37th International Conference on Machine Learning. vol. 119, pp. 1691–1703. PMLR (2020)

[9] Cheng, R., Wu, N., Chen, S., Han, B.: Reality check of metaverse: A first look at commercial social virtual reality platforms. In: IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW). pp. 141–148. IEEE (2022)

[10] Connor, C.E., Egeth, H.E., Yantis, S.: Visual attention: Bottom-up versus top-down. Current Biology 14(19), R850–R852 (2004)

[11] Crevecoeur, F., Kording, K.P.: Saccadic suppression as a perceptual consequence of efficient sensorimotor estimation. eLife 6, e25073 (2017)

[12] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., Salakhutdinov, R.: Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860 (2019)

[13] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
[14] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: 9th International Conference on Learning Representations (ICLR). OpenReview (2021)
[15] Duchowski, A.T.: Gaze-based interaction: A 30 year retrospective. Computers & Graphics 73, 59–69 (2018)
[16] Einhäuser, W., Nuthmann, A.: Salient in space, salient in time: Fixation probability predicts fixation duration during natural scene viewing. Journal of Vision 16(11), 13–13 (2016)
[17] Emery, K.J., Zannoli, M., Warren, J., Xiao, L., Talathi, S.S.: OpenNEEDS: A dataset of gaze, head, hand, and scene signals during exploration in open-ended vr environments. In: ACM Symposium on Eye Tracking Research and Applications (ETRA). ACM, New York, NY, USA (2021)
[18] Franke, L., Fink, L., Martschinke, J., Selgrad, K., Stamminger, M.: Time-warped foveated rendering for virtual reality headsets. In: Computer Graphics Forum. vol. 40, pp. 110–123. Wiley Online Library (2021)
[19] Frintrop, S.: VOCUS: A visual attention system for object detection and goal-directed search, vol. 3899. Springer (2006)
[20] Fuhl, W., Kasneci, G., Kasneci, E.: TEyeD: Over 20 million real-world eye images with pupil, eyelid, and iris 2d and 3d segmentations, 2d and 3d landmarks, 3d eyeball, gaze vector, and eye movement types. In: IEEE International Symposium on Mixed and Augmented Reality (ISMAR). pp. 367–375. IEEE (2021)
[21] Guenter, B., Finch, M., Drucker, S., Tan, D., Snyder, J.: Foveated 3d graphics. ACM Transactions on Graphics (TOG) 31(6), 1–10 (2012)
[22] Gurusamy, K.S., Aggarwal, R., Palanivelu, L., Davidson, B.R.: Virtual reality training for surgical trainees in laparoscopic surgery. Cochrane Database of Systematic Reviews (CDSR) (1) (2009)
[23] Han, D.I.D., Bergs, Y., Moorhouse, N.: Virtual reality consumer experience escapes: preparing for the metaverse. Virtual Reality pp. 1–16 (2022)
[24] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)
[25] Hollenstein, N., Rotsztejn, J., Troendle, M., Pedroni, A., Zhang, C., Langer, N.: ZuCo, a simultaneous eeg and eye-tracking resource for natural sentence reading. Scientific Data 5(1), 1–13 (2018)
[26] Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., Van de Weijer, J.: Eye tracking: A comprehensive guide to methods and measures. OUP Oxford, Oxford, England (2011)
[27] Hu, Z., Bulling, A., Li, S., Wang, G.: FixationNet: Forecasting eye fixations in task-oriented virtual environments. IEEE Transactions on Visualization and Computer Graphics (TVCG) 27(5), 2681–2690 (2021)

[28] Hu, Z., Bulling, A., Li, S., Wang, G.: EHTask: Recognizing user tasks from eye and head movements in immersive virtual reality. IEEE Transactions on Visualization and Computer Graphics (TVCG) (2022)
[29] Hu, Z., Li, S., Zhang, C., Yi, K., Wang, G., Manocha, D.: DGaze: Cnn-based gaze prediction in dynamic scenes. IEEE Transactions on Visualization and Computer Graphics (TVCG) 26(5), 1902–1911 (2020)
[30] Hu, Z., Zhang, C., Li, S., Wang, G., Manocha, D.: SGaze: A data-driven eye-head coordination model for realtime gaze prediction. IEEE Transactions on Visualization and Computer Graphics (TVCG) 25(5), 2002–2010 (2019)
[31] Huang, Y., Cai, M., Li, Z., Lu, F., Sato, Y.: Mutual context network for jointly estimating egocentric gaze and action. IEEE Transactions on Image Processing (TIP) 29, 7795–7806 (2020)
[32] Itti, L., Koch, C.: A saliency-based search mechanism for overt and covert shifts of visual attention. Vision Research 40(10-12), 1489–1506 (2000)
[33] Jia, S., Bruce, N.D.B.: EML-NET: An expandable multi-layer network for saliency prediction. Image and Vision Computing 95 (2020)
[34] Jiang, M., Huang, S., Duan, J., Zhao, Q.: SALICON: Saliency in context. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1072–1080 (2015)
[35] Kanter, D.: Graphics processing requirements for enabling immersive vr. AMD White Paper pp. 1–12 (2015)
[36] Kastrati, A., Plomecka, M.B., Pascual, D., Wolf, L., Gillioz, V., Wattenhofer, R., Langer, N.: EEGEyeNet: a simultaneous electroencephalography and eye-tracking dataset and benchmark for eye movement prediction. In: Proceedings of the Neural Information Processing Systems (NIPS) Track on Datasets and Benchmarks (2021)
[37] Kazemi, S.M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., Brubaker, M.: Time2vec: Learning a vector representation of time. arXiv preprint arXiv:1907.05321 (2019)
[38] Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in vision: A survey. ACM Computing Surveys (2021)
[39] Konrad, R., Angelopoulos, A., Wetzstein, G.: Gaze-contingent ocular parallax rendering for virtual reality. ACM Transactions on Graphics (TOG) 39(2), 1–12 (2020)
[40] Kothari, R., Yang, Z., Kanan, C., Bailey, R., Pelz, J.B., Diaz, G.J.: Gaze-in-wild: A dataset for studying eye and head coordination in everyday activities. Scientific Reports 10(1), 1–18 (2020)
[41] Koulieris, G.A., Drettakis, G., Cunningham, D., Mania, K.: Gaze prediction using machine learning for dynamic stereo manipulation in games. In: IEEE Virtual Reality. pp. 113–120. IEEE (2016)
[42] Langbehn, E., Steinicke, F., Lappe, M., Welch, G.F., Bruder, G.: In the blink of an eye: leveraging blink-induced suppression for imperceptible position and orientation redirection in virtual reality. ACM Transactions on Graphics (TOG) 37(4), 1–11 (2018)
[43] Li, R., Whitmire, E., Stengel, M., Boudaoud, B., Kautz, J., Luebke, D., Patel, S., Akşit, K.: Optical gaze tracking with spatially-sparse single-pixel

detectors. In: IEEE International Symposium on Mixed and Augmented Reality (ISMAR). pp. 117–126. IEEE (2020)
[44] Li, Y., Fathi, A., Rehg, J.M.: Learning to predict gaze in egocentric video. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 3216–3223 (2013)
[45] Li, Y., Liu, M., Rehg, J.M.: In the eye of beholder: Joint learning of gaze and actions in first person video. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 619–635 (2018)
[46] Linardos, A., Kümmerer, M., Press, O., Bethge, M.: DeepGaze IIE: Calibrated prediction in and out-of-domain for state-of-the-art saliency modeling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 12919–12928 (2021)
[47] Liu, Y., Sangineto, E., Bi, W., Sebe, N., Lepri, B., Nadai, M.: Efficient training of visual transformers with small datasets. Advances in Neural Information Processing Systems (NeurIPS) 34 (2021)
[48] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
[49] Matsas, E., Vosniakos, G.C.: Design of a virtual reality training system for human–robot collaboration in manufacturing tasks. International Journal on Interactive Design and Manufacturing (IJIDeM) 11(2), 139–153 (2017)
[50] Mazzeo, P.L., D'Amico, D., Spagnolo, P., Distante, C.: Deep learning based eye gaze estimation and prediction. In: 2021 6th International Conference on Smart and Sustainable Technologies (SpliTech). pp. 1–6. IEEE (2021)
[51] Meng, X., Du, R., Zwicker, M., Varshney, A.: Kernel foveated rendering. Proceedings of the ACM on Computer Graphics and Interactive Techniques (PACMCGIT) 1(1), 1–20 (2018)
[52] Murphy, H.A., Duchowski, A.T.: Gaze-contingent level of detail rendering. In: Eurographics 2001 - Short Presentations. Eurographics Association (2001)
[53] Mystakidis, S.: Metaverse. Encyclopedia 2(1), 486–497 (2022)
[54] Naas, S.A., Jiang, X., Sigg, S., Ji, Y.: Functional gaze prediction in egocentric video. In: Proceedings of the 18th International Conference on Advances in Mobile Computing & Multimedia (MoMM). pp. 40–47. ACM, New York, NY, USA (2020)
[55] Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML). ACM, New York, NY, USA (2010)
[56] Pashler, H.E.: The Psychology of Attention. MIT Press (1999)
[57] Patney, A., Salvi, M., Kim, J., Kaplanyan, A., Wyman, C., Benty, N., Luebke, D., Lefohn, A.: Towards foveated rendering for gaze-tracked virtual reality. ACM Transactions on Graphics (TOG) 35(6), 1–12 (2016)
[58] Perry, T.S.: Virtual reality goes social. IEEE Spectrum 53(1), 56–57 (2015)
[59] Rolff, T., Steinicke, F., Frintrop, S.: When do saccades begin? prediction of saccades as a time-to-event problem. In: ACM Symposium on Eye Tracking Research and Applications. ETRA '22, ACM, New York, NY, USA (2022)

[60] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. International Journal of Computer Vision (IJCV) 115(3), 211–252 (2015)
[61] Rzeszewski, M., Evans, L.: Virtual place during quarantine–a curious case of vrchat. Rozwój Regionalny i Polityka Regionalna (51), 57–75 (2020)
[62] Salvucci, D.D., Goldberg, J.H.: Identifying fixations and saccades in eye-tracking protocols. In: Proceedings of the 2000 Symposium on Eye Tracking Research & Applications. p. 71–78. ETRA '00, Association for Computing Machinery, New York, NY, USA (2000), https://doi.org/10.1145/355017.355028
[63] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations (ICLR) (2015)
[64] Sitzmann, V., Serrano, A., Pavel, A., Agrawala, M., Gutierrez, D., Masia, B., Wetzstein, G.: Saliency in VR: How do people explore virtual environments? IEEE Transactions on Visualization and Computer Graphics (TVCG) 24(4), 1633–1642 (2018)
[65] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1), 1929–1958 (2014)
[66] Stein, N., Niehorster, D.C., Watson, T., Steinicke, F., Rifai, K., Wahl, S., Lappe, M.: A comparison of eye tracking latencies among several commercial head-mounted displays. i-Perception 12(1), 1–16 (2021)
[67] Sun, Q., Patney, A., Wei, L.Y., Shapira, O., Lu, J., Asente, P., Zhu, S., McGuire, M., Luebke, D., Kaufman, A.: Towards virtual reality infinite walking: Dynamic saccadic redirection. ACM Transactions on Graphics (TOG) 37(4), 1–13 (2018)
[68] Sun, Y., Chen, Z., Tao, M., Liu, H.: Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff. IEEE Transactions on Communications 67(11), 7573–7586 (2019)
[69] Treisman, A.M., Gelade, G.: A feature-integration theory of attention. Cognitive Psychology 12(1), 97–136 (1980)
[70] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NIPS). vol. 30. Curran Associates, Inc. (2017)
[71] Xu, Y., Dong, Y., Wu, J., Sun, Z., Shi, Z., Yu, J., Gao, S.: Gaze prediction in dynamic 360 immersive videos. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5333–5342 (2018)
[72] Yang, C., Zhang, L., Lu, H., Ruan, X., Yang, M.H.: Saliency detection via graph-based manifold ranking. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on. pp. 3166–3173. IEEE (2013)
[73] Yarbus, A.L.: Eye movements and vision. Springer (2013)
[74] Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE Confer-

ence on Computer Vision and Pattern Recognition (CVPR). pp. 8697–8710 (2018)