Gaze-LLE: Gaze Target Estimation via Large-Scale Learned Encoders

Supplementary Material

Table of Contents

6. Integration of DINOv2 into Existing Methods

In this section, we provide further details on our experiments in Tab. 1, which integrate DINOv2 into three existing methods: Chong et al. [9], Miao et al. [43], and Gupta et al. [23].

Chong et al. Chong et al. [9]'s method employs separate head and scene encoders, each of which is composed of a ResNet50 + 1 additional ResNet layer. The input to the head branch is a 224×224 crop of the head and the input to the scene branch is the 224×224 scene image concatenated channel-wise with a binary map of the person's head bounding box position. The output of each encoder is a $1024 \times 7 \times 7$ feature map (channels $\times$ height $\times$ width). For our experiments, we replace the scene encoder with a ViT-Base DINOv2 encoder. Because the DINOv2 encoder produces a $768 \times 16 \times 16$ feature map, we apply average pooling with kernel size=3 and stride=2 followed by a convolutional layer with kernel size=1 and stride=1 to transform the feature map to the model's expected size of $1024 \times 7 \times 7$. We follow the rest of the existing method, which consists of an attention mechanism to re-weight the scene features based on the head features and head position, concatenation of the head and scene features, 2 convolutional encoding layers, and a 4-layer convolutional decoder. We consider 3 learning settings for the DINOv2 encoder:

1. **Frozen**: We simply replace the scene encoder with the DINOv2 encoder and freeze it during training. Because the DINOv2 takes in a 3-channel RGB image, we do not concatenate the head position map to the input as in the original method.
2. **Frozen + proj**: We alter the DINOv2 encoder's patch projection layer to take in 4 channels so that the input to the scene encoder is the concatenated RGB image and head position map like in the original method. We freeze all weights of the DINOv2 during training *except* the patch projection layer.
3. **Trained + proj**: We include the altered 4-channel patch projection layer and train the full DINOv2 encoder during training.

DINOv2 Training	Learning rate	AUC ↑	Avg L2 ↓	Min L2 ↓
Original Method	2.5e-4	**0.921**	**0.137**	**0.077**
Frozen	2.5e-4	0.858	0.196	0.133
	1.0e-4	0.857	0.201	0.145
	1.0e-5	0.808	0.230	0.166
	1.0e-6	0.726	0.287	0.218
Frozen + proj	2.5e-4	0.875	0.191	0.125
	1.0e-4	0.872	0.198	0.129
	1.0e-5	0.850	0.212	0.143
	1.0e-6	0.766	0.282	0.208
Trained + proj	2.5e-4	0.876	0.185	0.120
	1.0e-4	<u>0.908</u>	<u>0.167</u>	<u>0.101</u>
	1.0e-5	0.870	0.199	0.132
	1.0e-6	0.805	0.260	0.187

Table 9. Comparison of integrating DINOv2 into Chong et al. [9] with different training configurations (DINOv2 encoder learning strategy & learning rate) on GazeFollow.

Tab. 9 shows our results from trying different training strategies for the DINOv2 encoder and different learning rates. We see that learning the projection layer to integrate head position as an input to the scene encoder has a significant performance gain over using the DINOv2 with RGB-only inputs, and that training the DINOv2 fully performs best. Importantly, we do not observe overfitting - the trained results are better than using the frozen DINOv2. For this method, regular training outperforms LoRA. However, all results using a DINOv2 encoder in place of the ResNet50-based scene encoder perform worse than the original method.

Miao et al. Miao et al. [43] is a more recent work that expands upon Chong et al.'s architecture by integrating estimated depth into the scene encoding and feature fusion, a global attention mechanism over the scene prior to decoding, and an additional patch-level training objective. Similar to Chong et al., Miao et al. employ head and scene encoders composed of a ResNet50 + 1 additional ResNet layer. The input to the scene branch is a 5-channel concatenation of the RGB scene image, the binary head position map, and an estimated depth map from MiDaS [50]. Like with Chong et al., we replace the scene encoder with a DINOv2 encoder, and use average pooling and a convolutional layer to transform the scene feature map to size $1024 \times 7 \times 7$. We consider

DINOv2 Training	Learning rate	AUC ↑	Avg L2 ↓	Min L2 ↓
Original Method	2.5e-4	**0.934**	**0.123**	**0.065**
Frozen	2.5e-4	0.858	0.207	0.141
	1.0e-4	0.859	0.203	0.138
	1.0e-5	0.807	0.236	0.169
	1.0e-6	0.702	0.297	0.228
Frozen + proj	2.5e-4	0.892	0.173	0.109
	1.0e-4	0.887	0.176	0.113
	1.0e-5	0.859	0.203	0.137
	1.0e-6	0.761	0.286	0.213
Trained + proj	2.5e-4	0.899	0.165	0.103
	1.0e-4	0.910	0.152	0.093
	1.0e-5	0.900	0.161	0.098
	1.0e-6	0.847	0.220	0.149

Table 10. Comparison of integrating DINOv2 into Miao et al. [43] with different training configurations (DINOv2 encoder learning strategy & learning rate) on GazeFollow.

DINOv2 Training	Learning rate	AUC ↑	Avg L2 ↓	Min L2 ↓
Original Method	2.5e-4	**0.933**	**0.134**	**0.071**
Frozen + proj	2.5e-4	0.893	0.180	0.113
	1.0e-3	0.894	0.184	0.116
	1.0e-4	0.897	0.175	0.108
	1.0e-5	0.874	0.199	0.129
	1.0e-6	0.818	0.228	0.161
Trained + proj	2.5e-4	0.908	0.165	0.099
	1.0e-3	0.912	0.155	0.091
	1.0e-4	0.911	0.159	0.095
	1.0e-5	0.899	0.167	0.101
	1.0e-6	0.842	0.219	0.149

Table 11. Comparison of integrating DINOv2 into Gupta et al. [23] (Image-only variant) with different training configurations (DINOv2 encoder learning strategy & learning rate) on GazeFollow.

the same training configurations as we did with Chong et al., however we change the learned patch projection to have 5 input channels to account for Miao et al.'s inclusion of depth as input. As shown in Tab. 10, we achieve the best results by fully training the DINOv2. However, all configurations still perform worse than the original method with the ResNet50 backbone.

Gupta et al. Gupta et al. [23]'s approach consists of of a head-centric module, scene-centric module, and heatmap decoder. The head-centric module is a ResNet18 encoder which is supervised to predict 3D gaze from the head crop. This 3D gaze prediction is processed along with the head location into spatial gaze cone, which is passed to the scene-centric module along with the image. The scene-centric module consists of a separately trained EfficientNet encoder from different scene modalities: image, predicted depth, or predicted pose. Optionally, the encoders for the different modalities may be used together with a learned weighted attention module for fusion. As the training process calls for separately training each modality, we consider the image-only variant for our DINOv2 integration experiments. We replace the EfficientNet-B1 image encoder with DINOv2, and add an additional learned projection layer to reduce the dimension from DINOv2's output dimension of 768 to the model's internal dimension of 64. We consider both training the full encoder and freezing the encoder (with the exception of the input projection, which must accept the extra gaze cone channel). We report performance in Tab. 11. Like the other methods, training the DINOv2 performs better than freezing it, but still underperforms compared to the original method.

Method	Input size	AUC	Avg L2	Min L2
Chong et al. - Original	224	0.921	0.137	0.077
Chong et al. - Original	448	0.923	0.138	0.076
Chong et al. - Trained DINOv2	224	0.908	0.170	0.101
Chong et al. - Trained DINOv2	448	0.897	0.169	0.105
Miao et al. - Original	224	0.934	0.123	0.065
Miao et al. - Original	448	0.923	0.151	0.086
Miao et al. - Trained DINOv2	224	0.910	0.152	0.093
Miao et al. - Trained DINOv2	448	0.908	0.154	0.094
Gupta et al. - Original	224	0.943	0.114	0.056
Gupta et al. - Original	448	0.939	0.108	0.052
Gupta et al. - Trained DINOv2	224	0.912	0.155	0.091
Gupta et al. - Trained DINOv2	448	0.908	0.170	0.103

Table 12. Effect of increasing the input scene image size for Chong et al., Miao et al., and Gupta et al.'s original methods and best variants with DINOv2. We do not observe clear gains from using a larger input size.

Input Size Because we do not include a separate head branch that operates on a higher-resolution crop of the head in our main method, we use an input size of 448×448 instead of 224×224 like these prior works. To validate that our method's gains are not only a result of the larger input to the scene encoder, we retrain Chong et al., Miao et al.'s, and Gupta et al.'s original methods as well as the best variant with a DINOv2 scene encoder with scene input size 448×448 in Tab. 12. For Gupta et al.'s original method, we use their full multimodal model. We perform average pooling on the resultant scene feature maps when necessary to reduce the spatial dimensions to the expected shape for compatibility with the rest of the model. For Chong et al.'s method, the results are largely the same between using 224 vs. 448, while for Miao et al., using 448 actually decreases performance. For Gupta et al.'s architecture, increasing the resolution to 448 results in worse AUC, which is the primary metric on GazeFollow, but achieves slight gains on the L2 metrics. We thus *do not* see clear improvements from using an increased input size, illustrating that a larger scene input size is not necessary when a high-resolution head crop is already provided to the model.

Transformer Decoder
Linear ($d \rightarrow 256$)
Trans. Layer (dim=256, heads=8, mlp_dim=1024)
ConvT($256 \rightarrow 256$, k=2, s=2)
Conv($256 \rightarrow 1$, k=1, s=1)
Sigmoid

Conv Decoder
Conv ($d \rightarrow 768$, k=1, s=1)
Conv($768 \rightarrow 384$, k=1, s=1)
Conv($384 \rightarrow 192$, k=2, s=2)
ConvT($192 \rightarrow 96$, k=2, s=2)
ConvT($96 \rightarrow 1$, k=2, s=2)
Conv($1 \rightarrow 1$, k=1, s=1)
Sigmoid

Table 13. Architecture details for Transformer Decoder and Convolutional Decoder for experiments in Section 3.1

7. Experiment Details for Section 3.2

In this section, we provide further details about our experiments in Sec. 3.2 that investigate early vs. late head position integration, transformer vs. convolutional decoding, and head & scene branch vs. scene-branch only design.

Scene & Head Backbones We use a frozen DINOv2 ViT-Base backbone for both the scene branch and the head branch. For the scene branch, we use input size 448×448, yielding a feature map $x_{\text{scene}} \in \mathbb{R}^{768\times32\times32}$. Because the head occupies a smaller portion of the full-resolution image, we use input size 224×224 for the head branch and upsample the resulting feature map to $x_{\text{head}} \in \mathbb{R}^{768\times32\times32}$ so it can be concatenated with x_{scene}. We concatenate x_{scene} and x_{head} channel-wise to form the combined features $x \in \mathbb{R}^{1536\times32\times32}$. For the scene-only variant, we set $x = x_{\text{scene}} \in \mathbb{R}^{768\times32\times32}$.

Head Position Integration For "early" integration of the head position, we change the patch projection layer of the DINOv2 scene branch to have 4 input channels (RGB + binary head position map) instead of 3. During training, we learn this patch projection layer while keeping the rest of the DINOv2 frozen. For "late" integration, we do not alter or train the projection layer. Instead, we downsample the binary head position map map to size $1 \times 32 \times 32$ and concatenate it with x to form $x' \in \mathbb{R}^{d\times32\times32}$ where $d = 1537$ or $d = 769$ depending on the inclusion of the head branch. For "early" integration, we do not concatenate the head position map, so $x' = x \in \mathbb{R}^{d\times32\times32}$ where $d = 1536$ (head & scene branch) or $d = 768$ (scene branch only).

Decoder We provide architecture details for the transformer and convolutional heatmap decoders in Tab. 13. Each produce a 64×64 gaze heatmap from x'. The convolutional decoder is based on the network design used by Chong et al. [9] and several subsequent methods, consisting of 6 convolutional layers (each followed by batch normalization and a ReLU activation) to progressively project the feature map to a smaller dimension while upscaling it to the output heatmap size. The transformer decoder consists of a single transformer layer of dimension 256 followed by 2 shallow convolutional layers. Both decoders have approximately the same number of learned parameters (1.85M for the scene-branch only model with late head position integration).

Training Details We train the models on GazeFollow for 15 epochs using the Adam optimizer, cosine scheduling with initial learning rate 1e-3, and batch size 60. We use the same data augmentations during training that we use in our main experiments (random crop, flip, and bounding box jitter).

8. Comparison to Detection Methods

A set of recent works formulate gaze target estimation as a set detection problem, jointly predicting a set of head bounding boxes and their corresponding gaze locations [65–67]. We exclude these works from our main comparisons in Sec. 4.1 due to differences in the evaluation setting, as these methods perform bipartite matching *using the ground truth gaze targets at test time*. In this section, we provide further details about the difference in evaluation setting, and provide quantitative comparison with Tonini et al. [65] in our setting using their open source codebase.

Formulation Tu et al. [66] provided the first set detection formulation for joint head and gaze target detection by proposing HGTTR, a DETR [3]-based transformer detection framework. Given an image $x_{\text{img}} \in \mathbb{R}^{3\times H_{\text{in}}\times W_{\text{in}}}$, HGTTR predicts a fixed number of N human-gaze *instances*, where each instance y is composed of a head bounding box prediction $y_{\text{bbox}} \in [0,1]^4$, a binary classification score $y_{\text{class}} \in [0,1]$ indicating the probability that the instance is indeed a head, a prediction of if the gaze is in or out of frame $y_{\text{in/out}} \in [0,1]$, and a gaze heatmap $y_{\text{heatmap}} \in [0,1]^{H_{\text{out}}\times W_{\text{out}}}$. Notably, the difference between this setting and the traditional problem formulation (which we follow) is that a head bounding box is not given as input. Instead, the model predicts all head bounding boxes along with their associated gaze target as output.

Matching Algorithm Like DETR, HGTTR uses the Hungarian algorithm [34] to determine a one-to-one mapping between the predicted instances and the ground truth instances in order to calculate loss at train time. The optimal matching is found by considering all possible mappings w between the predicted instances and ground truth instances and selecting the one that minimizes

$$\mathcal{L}_{\text{cost}} = \sum_{i=1}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{w(i)}) \tag{4}$$

where $\mathcal{L}_{\text{match}}(y_i, \hat{y}_{w(i)})$ is a pairwise matching cost function between the i-th ground truth instance y_i, and the predicted instance with index $w(i)$, $\hat{y}_{w(i)}$. In HGTTR, $\mathcal{L}_{\text{match}}$ is defined as a weighted sum of loss functions:

$$\lambda_1 \mathcal{L}_{\text{bbox}} + \lambda_2 \mathcal{L}_{\text{class}} + \lambda_3 \mathcal{L}_{\text{in/out}} + \lambda_4 \mathcal{L}_{\text{heatmap}} \quad (5)$$

where $\mathcal{L}_{\text{bbox}}$ is an IoU loss on the predicted bounding box, $\mathcal{L}_{\text{class}}$ is a binary classification loss on predicting if the instance is a head or not, $\mathcal{L}_{\text{in/out}}$ is a binary classification loss on predicting if the gaze is in or out of frame, and $\mathcal{L}_{\text{heatmap}}$ is heatmap loss between the predicted and ground truth gaze target. The model always predicts N instances, with N being chosen to exceed the number of ground truth instances present in each image in the dataset (all existing methods use $N = 20$, which is significantly larger than the typical number of people present in a single image in GazeFollow). Because there are always less ground truth instances than predicted instances, the ground truth instance list is padded with $\emptyset$ so that it is length N. Predicted instances mapped to $\emptyset$ are excluded from cost and loss calculation. See Tu et al. [66] and DETR (Carion et al.) [3] for further details on matching.

Tonini et al. [65] expand upon this formulation, training their model to also predict all objects in the scene as an auxiliary training objective, and including depth as an input. They also add a term y_{vector} to each instance, which is a predicted gaze vector, and use this as auxiliary supervision. Their matching cost is defined as:

$$\lambda_1 \mathcal{L}_{\text{bbox}} + \lambda_2 \mathcal{L}_{\text{class}} + \lambda_3 \mathcal{L}_{\text{in/out}} + \lambda_4 \mathcal{L}_{\text{heatmap}} + \lambda_5 \mathcal{L}_{\text{vector}} \quad (6)$$

For all approaches, the mapping between the ground truth and predicted instances is determined by finding the closest subset of predicted instances to the ground truth based on bounding box, class, and gaze.

Evaluation Setting At inference time, these methods use the same matching cost $\mathcal{L}_{\text{cost}}$ to determine which predicted instances are evaluated against which ground truth instances, and use this to calculate the gaze performance metrics (e.g. heatmap AUC, L2 distance). This is inherently a different evaluation setting than ours because *the ground truth gaze labels are used at inference time* to retrieve the predicted instances that are compared against the ground truth instances. Because the fixed number of predicted instances ($N = 20$ for HGTTR) is much higher than the typical number of ground truth instances per image, a model can predict multiple instances with the same head bounding box, but different gaze targets (see Fig. 6 for visual examples of this). In this case, the matching algorithm will match each ground truth instance to the predicted instance with the closest gaze and calculate the gaze metrics between these pairs, discarding the extra incorrect predictions from evaluation. In this way, the gaze performance metrics alone do not penalize overdetection. They characterize recall by assessing the accuracy of the predicted instances that are closest to the ground truth, but do not assess precision by penalizing the model for predicting additional instances with incorrect gaze targets. Additionally, the matching algorithm does not enforce that the ground truth instance is matched to a detection with a similar predicted head bounding box; if the heatmap loss dominates the matching cost, an instance may be selected based only on similarity between the predicted gaze heatmap and ground truth (see Fig 6 for examples). Thus, the model does not need to correctly associate people with their respective gaze targets to achieve high performance. For these reasons, the gaze metrics in this evaluation setting are not a direct comparison against our work and prior methods that follow the traditional problem formulation.

Method	AUC ↑	Avg L2 ↓	Min L2 ↓
with ground truth gaze matching			
Tu et al. [66]	0.917	0.133	0.069
Tu et al. [67]	0.928	0.114	0.057
Tonini et al. [65]	0.922	0.069	0.029
Tonini et al.* [65]	0.924	0.068	0.030
no ground truth gaze matching			
Tonini et al.* [65]	0.767	0.211	0.148
Ours	0.956	0.104	0.045

Table 14. Quantitative comparison with detection-based methods on GazeFollow. The results *with ground truth gaze matching* use the ground truth gaze labels to perform bipartite matching at test time, and thus are not a direct comparison to our method and prior work. The *no ground truth gaze matching* results report our method compared to Tonini et al.'s model evaluated with the altered matching cost function in Equation 7, which excludes ground truth gaze information. (*Results we obtained ourselves by running Tonini et al.'s published code.)

Quantitative Results We show the reported results of the 3 detection-based methods and our results on GazeFollow in Tab. 14. To quantitatively characterize the difference in evaluation setting, we also re-evaluate Tonini et al.'s [65] method on GazeFollow with the ground truth gaze label removed from the matching cost, using their published codebase. We alter the matching cost from Equation 6 to exclude the ground truth gaze label. This altered matching cost is defined as:

$$\mathcal{L}'_{\text{match}} = \lambda_1 \mathcal{L}_{\text{bbox}} + \lambda_2 \mathcal{L}_{\text{class}} \quad (7)$$

With this cost, the model retrieves a prediction for each ground truth instance based only on bounding box overlap and class similarity. This reflects our use case, where the model is used to predict a gaze target for a certain per-

son based on their head location, and does not have knowledge of the ground truth gaze. Without the use of ground truth gaze in the matching cost at inference time, we observe a significant performance drop. This quantitatively indicates the overdetection of gaze instances, as the altered matching cost results in the model selecting detections that have more bounding box overlap and class similarity[2] to the ground truth, but a less accurate gaze target. However, it is important to note that we do not exhaustively attempt to adapt their method to this setting (e.g. by developing a new matching algorithm for training or a non-maximal suppression method). We include this result to quantitatively demonstrate the difference in evaluation setting and use case between our method their method as-is. We note that Tu et al. [66, 67] do not publish code or models so we do not re-evaluate their methods.

Qualitative Results We visualize the output instances of Tonini et al.'s default matching algorithm that uses ground truth gaze as part of the cost function, and our altered matching algorithm that does not use ground truth gaze in Fig. 6. The first two rows demonstrate cases where the matching algorithm chooses an instance with a predicted bounding box that is not associated with the correct person; the heatmap loss dominates the matching cost. With our altered matching function, an instance with a predicted bounding box for the correct person but incorrect gaze heatmap is retrieved. The third row shows an example of overdetection, where multiple instances are predicted with a head bounding box for the correct person, but different gaze targets. With the use of ground truth gaze during matching, the instance with the most correct heatmap is selected. However, without this ground truth information, the model does not select the best instance and produces an incorrect gaze prediction. These examples visually illustrate the difference in evaluation setting: when ground truth gaze information is used at test time, a model can achieve high performance by producing instances that capture different potential gaze targets and relying on the matching algorithm to retrieve the best instances to evaluate with. However, the gaze metrics do not characterize the model's ability to determine which of these instances are indeed gaze targets and associate them with the correct person.

9. Runtime Analysis

Inference Speed Our ViT-Base model runs in 15ms ($\approx$ 66fps) on a single NVIDIA RTX 4090 GPU. We compare the inference time of our model with existing methods in Fig 7a. For Miao et al. [43], we include the auxiliary depth estimation model (DPT-Hybrid[50]) in runtime calculation. Compared to Miao et al., our approach is both faster, and achieves better performance. In fact, the inference time of the DPT-Hybrid depth model (17ms) exceeds the entire inference time of our approach. This result highlights the benefit of using a single encoder, both in inference speed and performance. Chong et al.'s approach [9], which does not use any models for auxiliary modalities like depth, runs faster than our model. However, this comes with a significant drop in performance compared to our method. As shown in Tab. 3, recent convolutional methods all use at least one auxiliary model to augment performance. While these approaches may use faster backbones than a ViT, requiring auxiliary models ultimately increases runtime.

Multi-Person Scaling We also investigate how our model's runtime scales with estimating the gaze for multiple people per image (Fig. 7b). We measure the inference time for 1-10 people per image for both our default method, and our variant that uses a head position token (t_{pos}) and decodes gaze via cross attention and a dot product with the scene features (Tab. 7 configuration 1b 2b). Because the majority of our model's computation can be attributed to the DINOv2 scene encoder (>95% of computation), which is run once regardless of the number of people, our model's runtime does not increase much with addition of more people (15ms for 1 person vs. 19ms for 10 people). The token variant of our model with cross attention scales even better, as it decodes gaze for all people from the same final feature map. However, as shown in Fig. 7a, this is accompanied by a slight performance decrease.

We include Tonini et al.'s [65] detection method for comparison, which is designed to simultaneously predict the gaze and bounding boxes for all people in an image and thus has a constant runtime across different numbers of people. We include both the 2D variant (which does not use depth), and the 3D variant (which uses depth), accounting for the inference time for a DPT-Hybrid depth estimator for the 3D variant. Because our model requires head bounding boxes, we include a YOLOv5 head detector in the displayed runtimes for our model. We observe that our default method is faster than Tonini et al.'s 2D method for up to 7 people, and our token variant with cross attention is faster for all numbers of people. Due to the inclusion of running the depth model and modeling differences to include depth, the 3D version of Tonini et al.'s method is slower than the 2D version and our method.

10. Comparison to ViTGaze

We acknowledge concurrent work ViTGaze [55], which also proposes a single-branch transformer architecture for gaze target estimation based on DINOv2 pretrained weights. In contrast to Gaze-LLE, ViTGaze fully trains its

[2]We observe that matching is mainly based on bounding box overlap. Changing the weight of class similarity in the matching cost has little effect on performance both in the original setting and our altered setting where gaze is not used in matching.

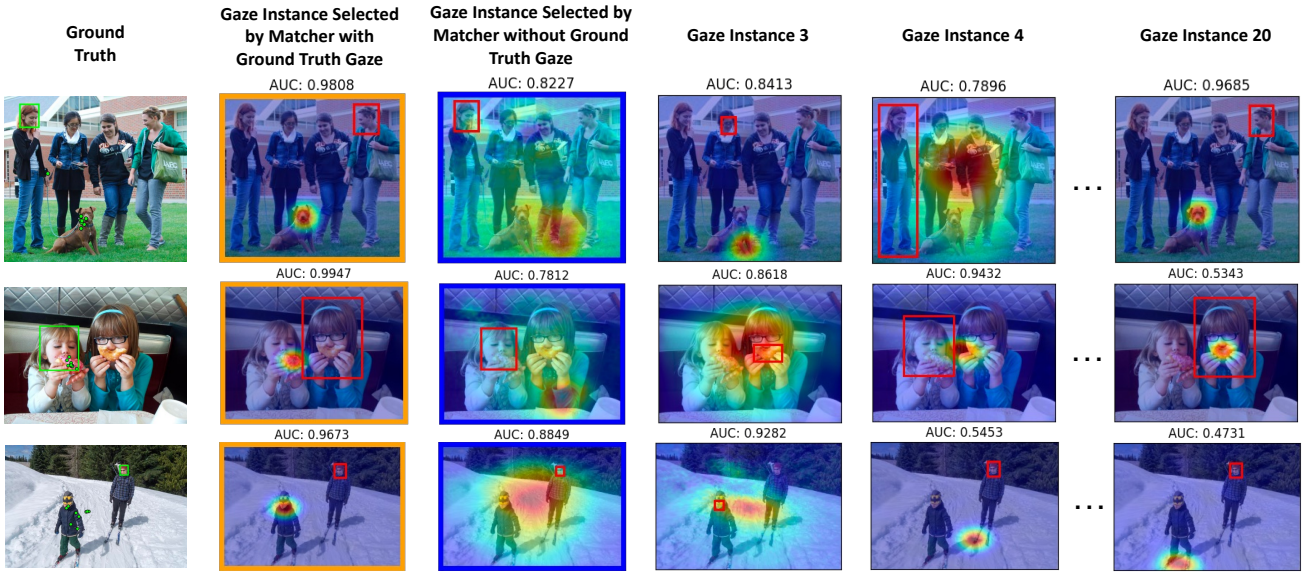


Figure 6. We show the output gaze instances (predicted head bounding box & gaze heatmap) from Tonini et al.'s model [65] for 3 examples. We identify the instances selected by Tonini et al.'s matching cost (which uses the ground truth gaze) and our altered matching cost (which excludes ground truth gaze and instead performs matching based on bounding box overlap). Tonini et al.'s matching algorithm selects the instance with the closest gaze prediction to the ground truth, but the bounding box prediction does not always correspond to the correct person (Rows 1-2). Additionally, we observe *overdetection*, where the algorithm predicts multiple instances for the same person with different gaze heatmaps (Row 3). Without the use of ground truth gaze information, the model cannot determine which of these instances is best.

ViT-S backbone (initialized from DINOv2 weights) end-to-end, and uses the attention weights between image patches as its feature representation. Gaze-LLE has the advantage of using the frozen DINOv2 features out-of-the-box, which is ideal for settings where general-purpose features are pre-computed and used for several downstream tasks. With its smaller backbone, ViTGaze is lightweight and may be better suited for on-device applications, where Gaze-LLE's ViT-B or ViT-L backbone may be too large to run. We note that ViTGaze produces predictions for the L2 metric differently than prior gaze methods: while prior work determines the maximal gaze point from a standard-sized 64×64 heatmap, ViTGaze uses additional postprocessing [72] to bypass the limitations of the low resolution of the output heatmap. ViTGaze also uses a higher input resolution (512).

11. Performance with Estimated Head Bounding Boxes

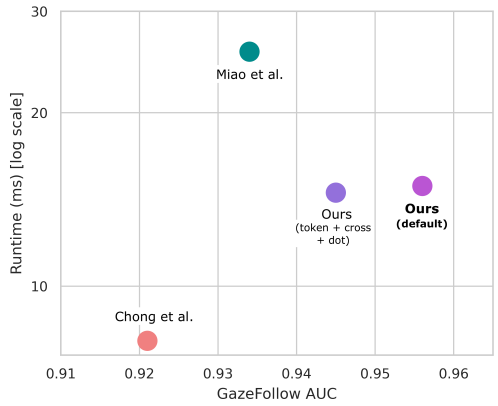
Tu et al. [66] report that 2-stream methods suffer major performance drops when using head bounding boxes from a detector rather than the dataset ground truth. In contrast, we observe almost no performance degradation when pairing our method with a YOLOv5 head detector trained on CrowdHuman [1, 54] (Tab. 15). This result demonstrates that our single-stream design, which uses a coarse, down-sampled head position map, is less dependent on an exact head crop, and works well with out of the box head detections. Given DINOv2's strong performance on tasks such as semantic segmentation with linear probing, future work may explore integrating head detection directly into the pipeline by predicting heads from the same frozen DINOv2 features.

Method	AUC ↑	Avg L2 ↓	Min L2 ↓
ViT-B + GT	0.956	0.104	0.045
ViT-B + YOLO	0.955	0.106	0.047
ViT-L + GT	0.958	0.099	0.041
ViT-L + YOLO	0.958	0.101	0.043

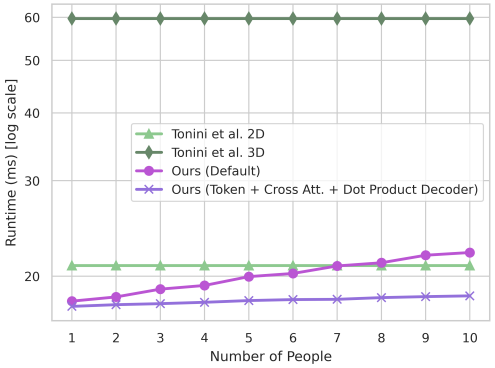
Table 15. Gaze-LLE achieves consistent results when using head detections from an out-of-the-box YOLOv5 detector instead of head ground truth bounding boxes.

12. Reimplementation of Horanyi et al.

We use our own implementation of Horanyi et al. [26] for our main comparison. We choose to reimplement this method because the reported results are outliers among other methods, and there is imbalance between the reported metrics (*e.g.*, 0.932 AUC on GazeFollow, but very low L2 error). Since the method is largely constructed from ele-



(a) Runtime vs. Performance



(b) Runtime scaling for multi-person inference

Figure 7. Runtime analysis of our approach: we show the tradeoff of inference time vs. performance (7a), and analyze how different variants of our approach paired with a head detector scale for multi-person prediction, compared to detection methods (7b). All experiments are performed on a single NVIDIA RTX 4090 GPU.

	GazeFollow			VideoAttentionTarget		
Experiment	AUC ↑	Avg L2 ↓	Min L2 ↓	AUC ↑	L2 ↓	$AP_{in/out}$ ↑
Frozen Aux. Angle	0.869	0.217	0.146	0.802	0.234	0.720
Trained Aux. Angle	**0.896**	**0.196**	**0.127**	**0.832**	**0.199**	**0.800**

Table 16. Experimental results for our implementation of Horanyi et al.[26] on GazeFollow and VideoAttentionTarget. We consider the setting where we freeze the auxiliary 3D gaze angle model vs. where we train it along with the rest of the network.

ments that are present in other works (*e.g.*, constructing a "gaze cone" from estimated 3D gaze and depth [17, 23, 65], providing estimated depth as input to the scene encoder [17, 23], and using a ResNet50-based scene encoder + 4-layer convolutional decoder [9, 43]), it is difficult to identify the source of large reported performance gains. There is not published code for this work. The original paper provides limited implementation details, so we follow some choices from Chong et al.'s codebase [9].

The model consists of a 3D Field-of-View (FoV) map construction module, a scene encoder, and a gaze decoder. The FoV module uses an auxiliary depth estimator and 3D gaze angle estimator to produce an FoV heatmap for a person over the scene. The estimated depth, FoV map, and 224×224 map are passed to a ResNet50-based scene encoder and decoded into gaze predictions. Figure 8 illustrates the architecture details for our reimplementation. We use the same auxiliary models used in the original approach [26]: Gaze360 [32] and Monodepth2 [21]. We follow the version of their scene encoder without non-local (NL) blocks. The FoV module uses the construction equation from Horanyi et al. [26]:

$$M_{ind} = \text{min\_max\_scaler}\left(\frac{(i-h_x, j-h_y, k-h_z)\cdot(g_x, g_y, g_z)}{\|i-h_x, j-h_y, k-h_z\|^2 \cdot \|g_x, g_y, g_z\|}\right) \tag{8}$$

We make the assumption that k-coordinate comes from the normalization of the estimated depth map. We follow the high-level architecture described in the text: a ResNet50 trainable scene encoder, two convolutions for encoding, and a 4-layer convolutional decoder. Because details such as hidden dimensions and kernel sizes are not specified, we generally follow Chong et al.'s open-source code [9] since Horanyi et al.'s described architecture mostly matches Chong et al.'s. We conduct experiments in two settings on the Gazefollow and VideoAttentionTarget datasets. The first setting keeps both the auxiliary gaze angle and depth estimation models frozen, as suggested in the text [26]. In the second setting, we train the gaze angle model. For the GazeFollow experiments, we use batch size 128, learning rate 4e-4, and the Adam optimizer. For VideoAttentionTarget, we finetune the GazeFollow-trained model with batch size 32 and learning rate 1e-4. The results of these experiments are shown in Tab. 16. We observe training the auxiliary gaze angle model performs better, so we report these results in the main paper.

13. Additional Ablation Studies

We provide additional ablations for our ViT-Base model on GazeFollow in Tab. 17. We find there is benefit to using a smaller internal dimension for our gaze estimation module, both in performance and reduction of learnable parameters (Tab. 17a). Our model produces competitive results with prior work using only 1 transformer layer (Tab. 17b); however, we achieve sizeable performance gains by increasing the number of layers to 3. Beyond 3 layers, the performance is largely stable. To balance performance with reducing learnable parameters, we select dimension 256 with 3 layers as our default configuration.

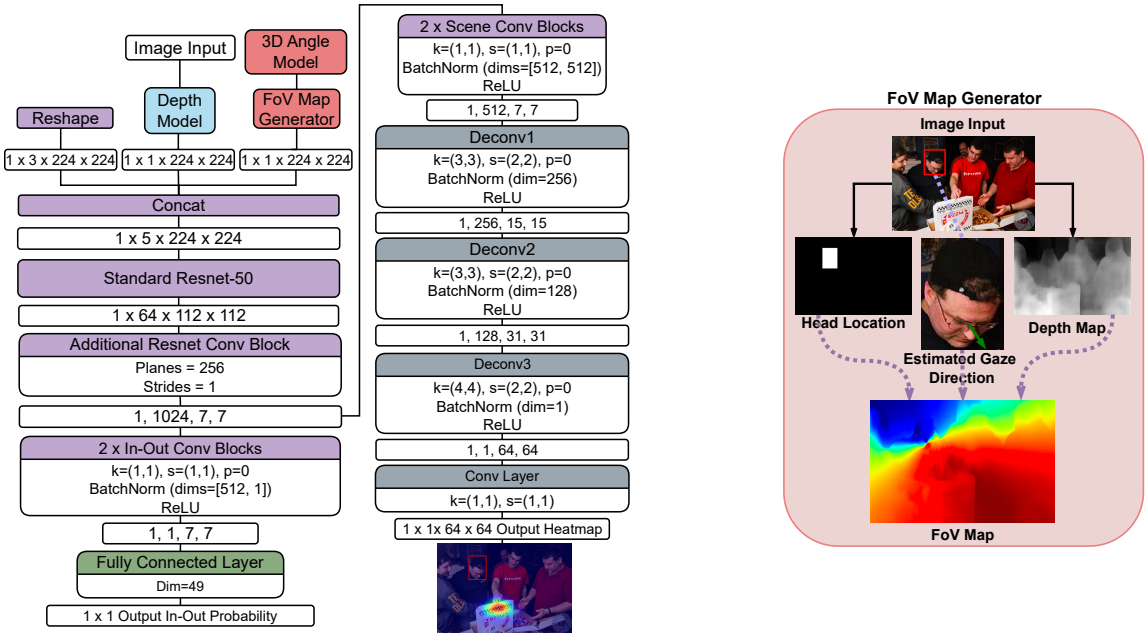


Figure 8. Architecture details for our reimplementation of Horanyi et al.'s model [26]. The model consists of a FoV Map Generator (shown on right), which uses an auxiliary 3D gaze angle estimator and an auxiliary depth model to produce an FoV map for a given person. The FoV map, estimated depth, and image are passed to a ResNet50-based encoder and convolutional decoder to produce a gaze prediction. In our experiments, we consider both freezing vs. training the 3D gaze angle estimator as part of the model.

d_{model}	Params	AUC ↑	Avg L2 ↓	Min L2 ↓
128	1.2M	**0.956**	0.106	0.046
256 (default)	2.8M	**0.956**	**0.104**	**0.045**
384	5.0M	**0.956**	0.105	0.046
512	7.7M	0.953	0.108	0.049
768	14.8M	0.953	0.108	0.049

(a) Dimension of gaze estimation module.

Layers	Params	AUC ↑	Avg L2 ↓	Min L2 ↓
1 layer	1.2M	0.953	0.115	0.054
2 layers	2.0M	0.955	0.108	0.049
3 layers	2.8M	**0.956**	0.104	**0.045**
4 layers	3.6M	**0.956**	**0.103**	**0.045**
5 layers	4.4M	**0.956**	0.104	**0.045**

(b) Number of transformer layers.

Table 17. We investigate the effect of different internal model dimensions and number of transformer layers for our gaze estimation module with a ViT-Base DINOv2 backbone. We observe diminishing returns as we increase the dimension and number of layers. We select $d_{\text{model}} = 256$ with 3 transformer layers as our default configuration.

14. LoRA Backbones

To investigate if training the backbone improves performance, we explore using Low Rank Adaptation (LoRA) [27] on GazeFollow in Tab. 19. LoRA updates the backbone while introducing limited additional learnable parameters by learning weight update matrices as low rank decompositions. We update the query and value projections of the DINOv2 backbone using rank 16. We observe limited improvements, which we attribute to (1) the effectiveness of the frozen encoder's feature representation for our task and (2) that our models with frozen encoders already achieve extremely close performance to the inter-rater performance of the human annotators, which serves as a soft upper bound on the L2 metrics.

Backbone	Params	AUC ↑	Avg L2 ↓	Min L2 ↓
One Human		*0.924*	*0.096*	*0.040*
ViT-B	2.8M	0.956	0.104	0.045
ViT-B + LoRA	3.1M	0.957	0.103	0.045
ViT-L	2.9M	0.958	0.099	0.041
ViT-L + LoRA	3.7M	0.960	0.097	0.040

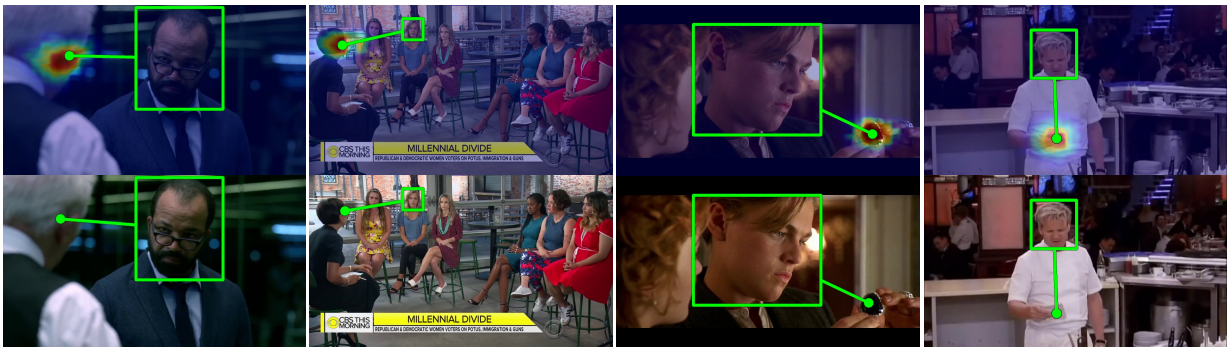
Table 18. LoRA-tuned DINOv2 Backbones

Table 19. Frozen vs. LoRA-tuned DINOv2 backboneswith Gaze-LLE on GazeFollow.

15. Additional Visualizations & Failure Modes

We provide additional visualizations of our ViT-B model's predicted heatmaps on the GazeFollow, VideoAttentionTarget, ChildPlay, and GOO-Real datasets in Figure 9. We

(a) GazeFollow



(b) VideoAttentionTarget

(c) ChildPlay

(d) GOO-Real

Figure 9. Additional qualitative results on the 4 evaluation datasets: For each example, we show our model's predicted heatmap with its maximum point on the top, and the ground truth gaze annotations on the bottom.

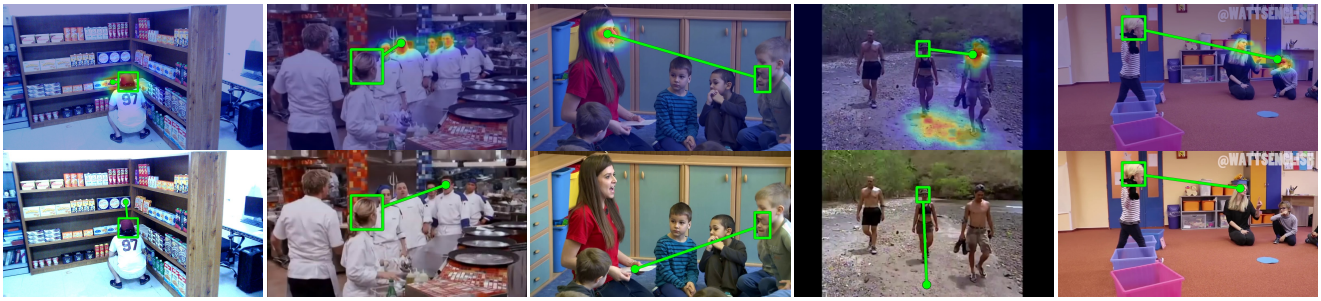


Figure 10. Lower performing cases: we observe errors in some cases where the head is facing away from the camera (examples 1-2), the head is occluded (examples 3), or the face is blurred (examples 4-5).

show examples where our model does not perform as well in Figure 10. These cases are representative of error modes we observe across the evaluation datasets. Our model is more likely to exhibit errors when the person is positioned with the back of their head towards the camera (examples 1-2) or their face is heavily occluded (example 3). In these cases, we observe our model selects potential targets (such as faces) that are broadly in person's field of view, but does not always refine this prediction to the ground truth gaze target. It is not surprising that the model does not perform as well on these cases, as the ground truth is often inherently more ambiguous in such examples. We observe similar errors in cases where the person's face and eyes are blurred (examples 4-5), which is more common in video datasets like VideoAttentionTarget and ChildPlay. Future work may explore using temporal information from surrounding frames to resolve ambiguities in these cases.