Rethinking 360° Image Visual Attention Modelling with Unsupervised Learning.

Yasser Abdelaziz Dahou Djilali[*], Tarun Krishna[*], Kevin McGuinness and Noel E. O'Connor
Insight Centre for Data Analytics, Dublin City University (DCU)
{yasser.dahoudjilali2, tarun.krishna2}@mail.dcu.ie

A. Architecture Details

A.1. VGG16

- **Encoder**: VGG16 encoder is derived from SalGAN [3][1].
- **Global** module: It comprises of a convolution block i.e Conv2D $\rightarrow$ BatchNorm2D $\rightarrow$ ReLU $\rightarrow$ MaxPool2D] followed by two linear layers with rectified linear units to get a global representation.
- **Local** module: Consists of $2\times$[Conv2D $\rightarrow$ ReLU] followed by a BatchNorm2D layer to final local representation.

A.2. ResNet50

This sub-section gives a detailed overview of the ResNet50 based encoder and its corresponding global/local module. The naming convention used is similar to [1].

- **Encoder**: Figure 1 depicts the ResNet50 architecture used up to the average pooling layer ignoring the last fully connected and consequently softmax layer. The encoder f_θ is composed of a ResNet50 up to `conv4_x` and an extra set of convolution layers denoted by Γ_η. This was done to keep the channel dimension fixed to 512 so as to be consistent with the VGG based model.
- **Global** module Σ_σ is composed of H_μ and `conv5_x` up to the `average pooling layer` followed by two linear layers with rectified linear units. Again, H_μ was added to increase the channel dimensions to 1024 to make it compatible with input channel dimension of `conv5_x` i.e., the original pipeline of ResNet (as we extend `conv4_x` to have 512 dimensions) to have same channel dimension of 1024.
- **Local** module is exactly same as VGG16 (local module).

$\boldsymbol{\mu}$ and $\boldsymbol{\eta}$ are parameters to sub-function H and Γ respectively.

Global module can also be consider as a projection layer as used in most of the self-supervised models but in this case it is applied in an asymmetric way.

A.3. Self-Attention

Listing 1 depicts the PyTorch styled implementation of the self-attention module. This takes as input the source ($\Lambda_{\mathbf{x}} = f_\theta(\mathbf{x})$) and augmented ($\Lambda_{\mathbf{x}_t} = f_\theta(\mathbf{x}_t)$) local latent representations, which has dimension $512\times10\times20$. The objective of this module is to provide a mechanism to perform feature selection between the feature maps of two views. To achieve this, the latent representations are transformed into `Q` (query) and `K` (key) feature spaces through weight matrices $\mathbf{W}_\mathrm{Q}$ and $\mathbf{W}_\mathrm{K}$. Attention is calculated through dot product between the two representations i.e $\mathbf{W}_\mathrm{Q}(\Lambda_{\mathbf{x}})$ and $\mathbf{W}_\mathrm{K}(\Lambda_{\mathbf{x}_t})$. This dot product is calculated between each location "vector" (i.e. for each location $\{(:,i,j)\Big| : i \in \{1,\ldots,10\}, j \in \{1,\ldots,20\}\}$) across two views ($\mathbf{W}_\mathrm{Q}(\Lambda_{\mathbf{x}})$ and $\mathbf{W}_\mathrm{K}(\Lambda_{\mathbf{x}_t})$), resulting in attention weights of size 200×50[2]. In Listing 1, `K` is max-pooled to reduced for spatial resolution which results in an attention matrix of size 200×50 (instead of 200×200). Intuitively, this means performing the dot product of each location vector (`Q`) with a patch (in `K`) because each pixel has a receptive field of 4 in K due to the max-pooling. Once we have these attention weights we could now perform feature selection through `V` (value) i.e $\mathbf{W}_\mathrm{V}(\Lambda_{\mathbf{x}_t})$. Further `O` is multiplied by γ and added to $\Lambda_{\mathbf{x}_t}$. This residual connection modulates the value of γ in terms of the extra contribution. In our case γ achieves a value of 8.0 after 250 epochs. The choice to reduce the channel dimensions and applying pooling was done following [6][3].

```
in_channel = 512
K = 8
γ = 0 #trainable parameter

#64:=512/K,256:=512/2
W_Q = nn.Conv2d(512, 64, (1,1))
W_K = nn.Conv2d(512, 64, (1,1))
```

[*]Equal contribution
[1]Code: https://github.com/imatge-upc/salgan
[2]excluding batch-size for simplicity
[3]Implementation is taken from here [6] and adopted to PyTorch

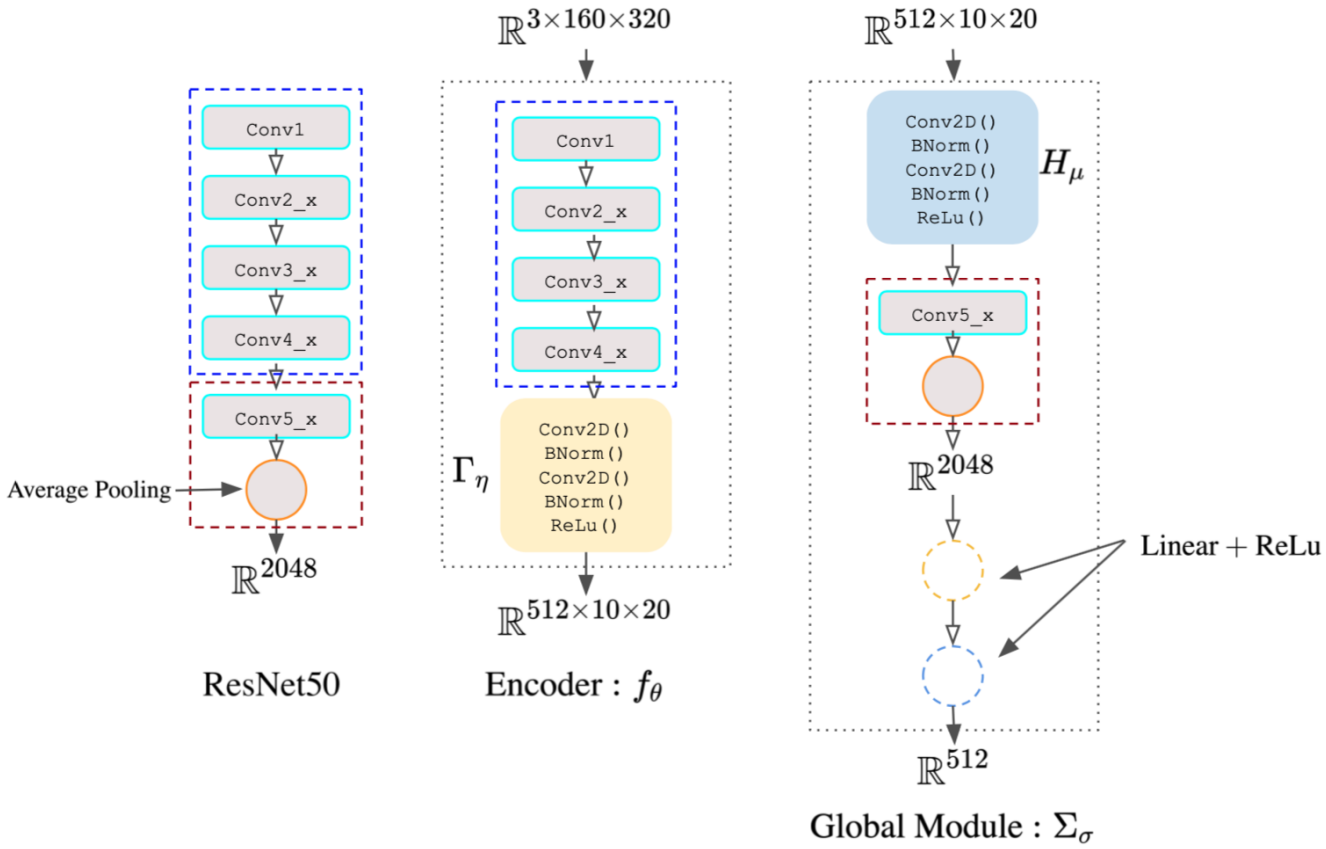

Figure 1: Detailed architecture for ResNet50 based encoder.

```
W_V = nn.Conv2d(512, 256, (1,1))
W_O = nn.conv2d(256, 512, (1,1))

def selfAttention(Λ_x, Λ_{x_t}):
    #Λ_x, Λ_{x_t} latent representations for x and x_t

    #Get Query
    Q = W_Q(Λ_x)#(bsz,64,10,20)
    Q = Q.view(bsz,64,-1).permute(0, 2, 1)#(bsz
    ,200,64)

    #Get Key
    K = W_K(Λ_{x_t})#(bsz,64,10,20)
    K = F.MaxPool2D(K) #over (2,2) patches
    K = K.view(bsz, 64, -1)#Reshape (bsz,64,50)

    #Calculate importance weights
    attn = torch.bmm(Q, K) #(bsz, 200, 50)
    attn = F.softmax(attn, dim=-1) #Softmax
    attn = attn.permute(bsz,2,1)#(bsz, 50, 200)

    #Get Value
    V = W_V(Λ_{x_t})#(bsz,256,10,20)
    V = F.MaxPool2D(V) #over (2,2) patches
    V = V.view(bsz, 256, -1)#reshape (bsz,256,50)

    #Get importance vectors
    attn_val = torch.bmm(v, attn)#(bsz, 256, 200)
    attn_val = attn_val.reshape(bsz, 256, 10,20)

    #Apply another projection
    O = W_O(attn_val)

    return O*γ + Λ_{x_t}
```

Listing 1: PyTorch-based implementation

B. Results

It can be seen in Figure 2 and Figure 3 that the VGG-based model overly biases the saliency to the equator. Since the encoder's weights were fixed when training the decoder on saliency, the learning relies on the latent representations learned from the unsupervised training. The sequential architecture of VGG causes gradient vanishing in the early layers (i.e. they do not change much from the initial initialization). Figure 4 show the normalized Frobenius norm difference between the weights at epochs 250 and 10 per layer; the ResNet-based encoder parameters change by more than 2-fold on average, whereas the VGG weights do not exceed 1.42-fold. We argue that the VGG encoder preserves the information of the pixels, more so than learning higher-level features. Another interesting point is the considerable change in the first layer for both encoders. We suspect this is

Table 1: The standard deviation of all predictions across the validation set.

	Salient360! AUC-J ↑	NSS ↑	CC ↑	SIM ↑	KLD ↓
VGG	0.760±0.04	1.548±0.20	0.538±0.11	0.569±0.05	0.922±0.28
ResNet	0.769±0.03	1.601±0.18	0.584±0.08	0.591±0.03	0.849±0.16

related to the 360 input images; local filters change to adapt to the data representation. Table 2 analyses the 25 latent representations obtained from both VGG and ResNet contrastive encoders. We measured the inter-maps correlation: for a given representation $\Lambda \in \mathbb{R}^{512\times10\times20}$, we fix a target map $\Lambda_\mathbf{i} \in \mathbb{R}^{10\times20}$, and compute the average correlation coefficient among the maps, $j \in \{1,\dots,512\}, j \neq i$. For VGG, the average $CC = 0.31$ while ResNet has $CC = 0.10$. This shows that the latent representations produced by the contrastive ResNet encoder have more variations across the channels, thus embedding more patterns compared to the VGG ones (see Figure 5). Furthermore, training on top of the VGG-encoder appears to force the decoder to learn an average saliency distribution over the training set.

Surprisingly, however, this still achieves reasonable results ($KLD = 0.922; NSS = 1.548$). An explanation could relate to the statistical bias present in HM/EM fixations. Viewers have the tendency to gaze at regions near the equator (termed the equator bias [4, 2]). In contrast, the decoder trained on top of the ResNet-based encoder is able to learn non-obvious saliency distribution, and fixate on critical regions, further improving the model's accuracy. It can be observed that the ResNet-based model is able to correlate well with the ground-truth distribution (see Figure 2, Figure 3). To further show this, Table 3 measures the distance between an equator bias, and the models' predictions; the VGG-based model has a higher similarity with the equator bias (KLD $= 1.867$), whereas the ResNet-based model has lower similarity (KLD $= 2.507$).

Finetuning the whole model. As an additional experiments, we finetuned the Resnet based model end-to-end on the saliency dataset using the Adam optimizer, this would adapt the unsupervised weights to the specific task at hand. The results are: (AUC-J: 0.790), (NSS: 1.616), (CC: 0.586), (SIM: 0.607), (KLD: 0.815). This is not particularly surprising as the encoder weights are more specific to the task at hand.

Average pooling based encoder. We replaced the attention with an average pooling on $f(x_t)$ (equivalent to equal attention). This will induce a global attention effect, the results are (AUC-J: 0.728), (NSS: 1.544), (CC: 0.528), (SIM: 0.558), (KLD: 0.901). This shows the importance of the local attention specifically designed for the downstream task.

Failure scenarios. To further analyse the model's expressive power, we have detected some outliers on the Salient360! validation set (i.e. the images with the highest KLD $= 1.51$; the lowest NSS $= 1.09$). Figure 6 illustrates two failure modes identified so far. In the top image, the model misses the object highlighted in the yellow box, which is salient to humans. This object, however, appears to have been blurred in capture due to camera motion, which likely affected the features that were usually being used to detect an object as salient. However, this may be seen as a problem/artifact of the input data., and not a problem due to model capacity. The second scenario appears more challenging. The image was captured between two cabins in a train; the model is able to detect salient regions in the front cabin, but clearly humans localized in the viewing sphere to generate the ground truth were saccading on people standing in the rear cabin. The resolution of this rear cabin is small in the ERP format. Hence, we suspect that detecting saliency at depth is a weakness of the model. A potential solution could be the use of multi-scale attention [5], which has been shown to improve failure modes relating to features that need to be detected at different spatial scales.

Table 2: Statistics about the Salient360! 24 validation images latent representations $\Psi_\mathbf{x} \in \mathbb{R}^{512\times10\times20}$.

	CC
VGG contrastive encoder	0.295
ResNet contrastive encoder	0.107

Model consistency. Averaging the metrics over the validation set biases the evaluation somehow, as models can fit precisely some samples, but give poor predictions for others. Table 1 shows the results on Salient360! validation set, with confidence intervals. It can be seen that the ResNet-based model has less variance across all metrics, demonstrating the consistency of the predictor. The VGG-based model, on the other hand, has a wider confidence interval, which is explained by the fact that a substantial amount of the ground truth saliency has high density in the equator.

C. Projections

There are a wide variety of well known projections from a sphere to a two-dimensional map, and the choice depends on the task at hand. For saliency prediction, the straightforward equi-rectangular projection of the sphere to a 2D plane

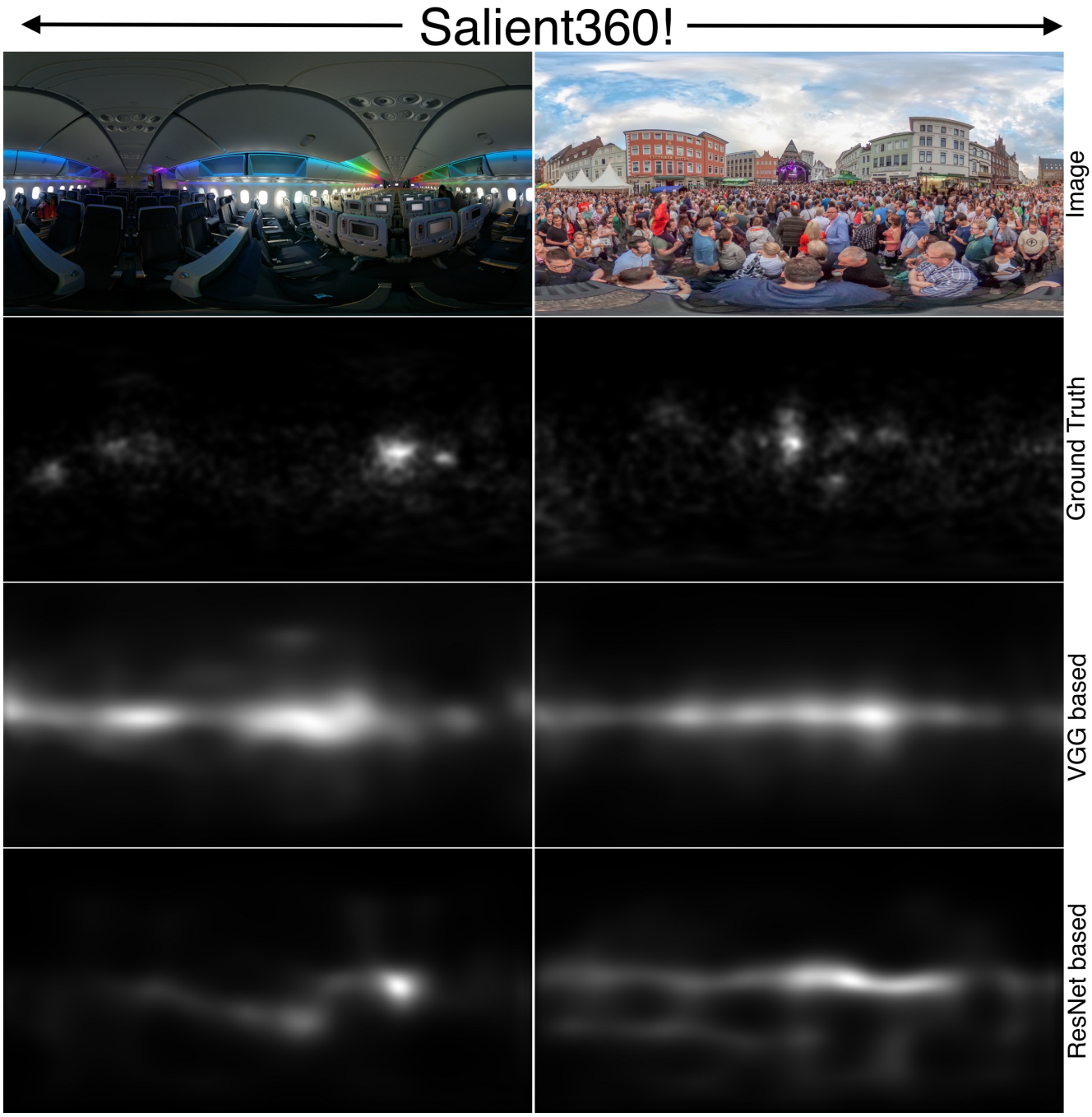

Figure 2: Qualitative results of model variants on sample images from Salient360! dataset.

introduces significant geometric distortions near the poles. Furthermore, the spherical data representation gives more flexibility to design new views for contrastive learning. The main idea is to be able to reconstruct the whole sphere angles through the different views, as shown in Figure 7. The first projection consist of bring top-to-front by rotating the sphere around the horizontal axis, this recovers objects originally localized on the Zenith in the support ERP image. Accordingly, the second projection deals with the bottom-to-front.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778,

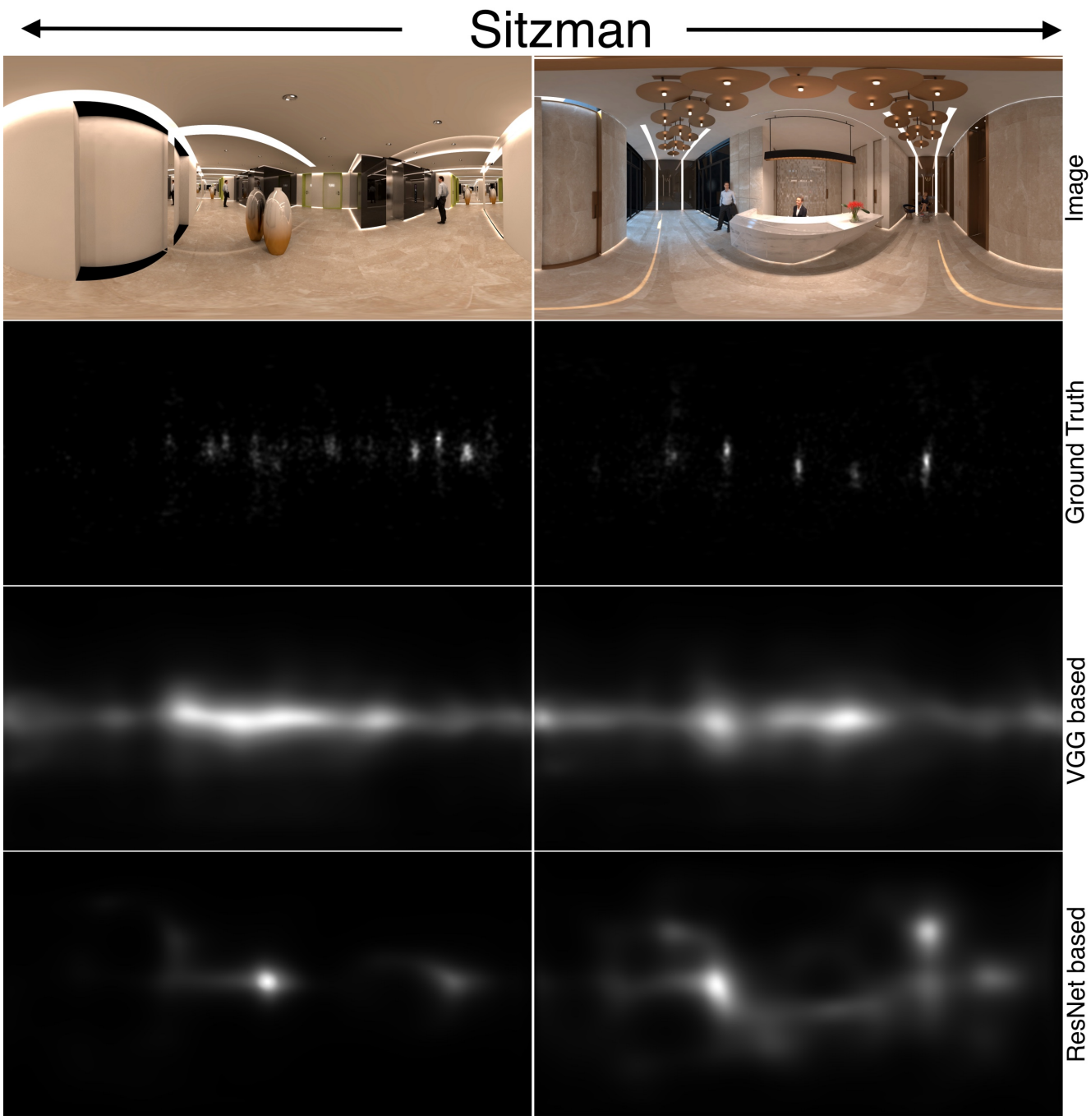

Figure 3: Qualitative results of model variants on sample synthetic images from Sitzman dataset.

2016. 1

[2] P. Lebreton and A. Raake. Gbvs360, bms360, prosal: Extending existing saliency prediction models from 2d to omnidirectional images. *Signal Processing: Image Communication*, 69:69–78, 2018. 3

[3] J. Pan, C. C. Ferrer, K. McGuinness, N. E. O'Connor, J. Torres, E. Sayrol, and X. Giro-i Nieto. Salgan: Visual saliency prediction with generative adversarial networks. *arXiv preprint arXiv:1701.01081*, 2017. 1

[4] V. Sitzmann, A. Serrano, A. Pavel, M. Agrawala, D. Gutierrez, B. Masia, and G. Wetzstein. Saliency in vr: How do people explore virtual environments? *IEEE transactions on visualization and computer graphics*, 24(4):1633–1642, 2018. 3

[5] A. Tao, K. Sapra, and B. Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv preprint arXiv:2005.10821*, 2020. 3

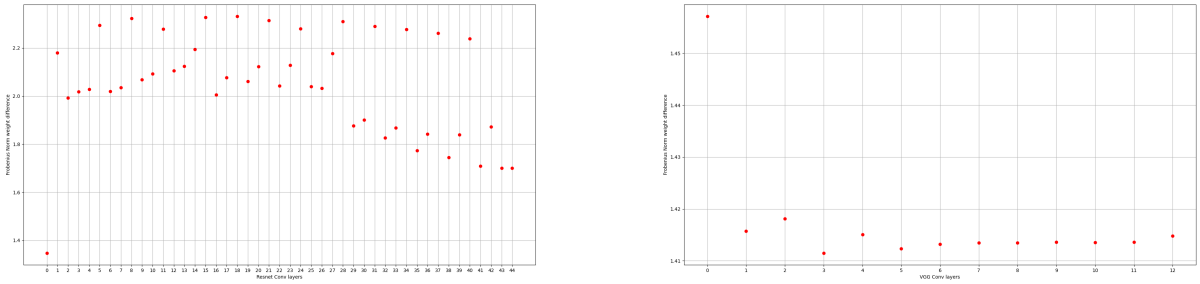

Figure 4: The Frobenius norm of the weights difference layer-wise, between encoders obtained at epochs 250 and 10.

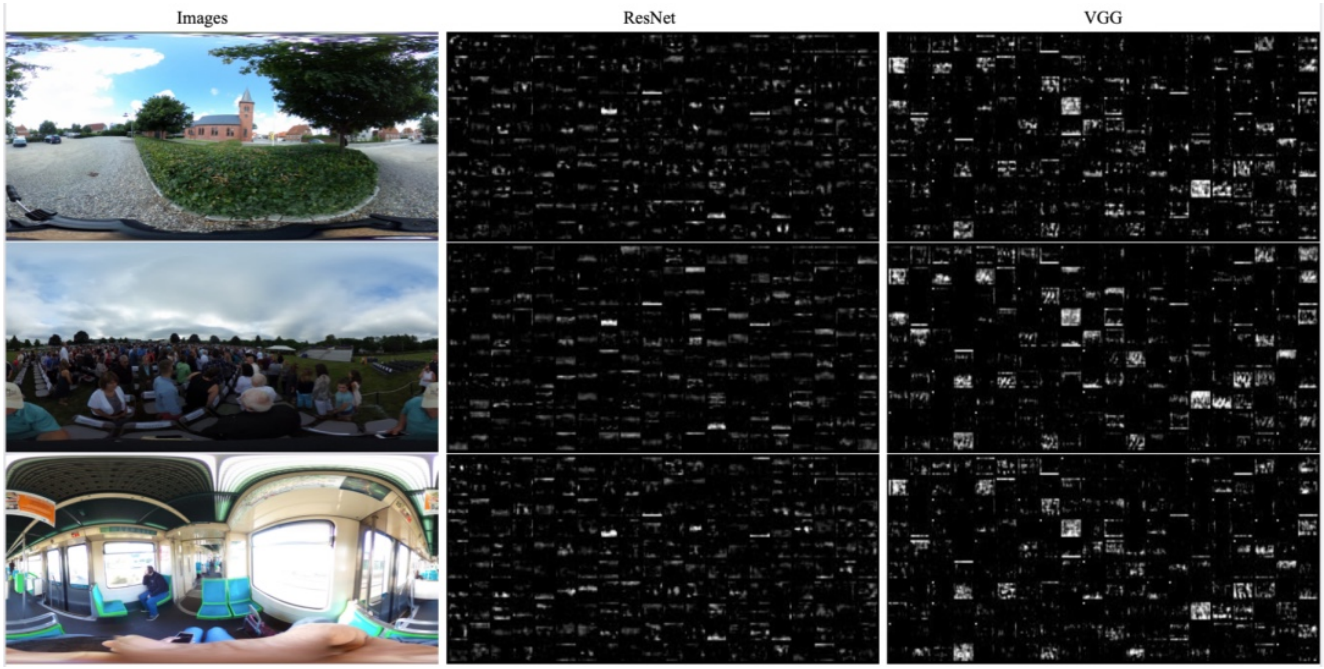

Figure 5: t-SNE visualization of the embeddings on sample images from the Salient360! validation set. ResNet embeddings are marked in red. VGG embeddings are marked in navy blue.

Table 3: The probabilistic distance between the equator bias and the models predictions on the Salient360! validation set.

	KLD ↓
VGG-based saliency model	1.867
ResNet-based saliency model	2.507

[6] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019. 1

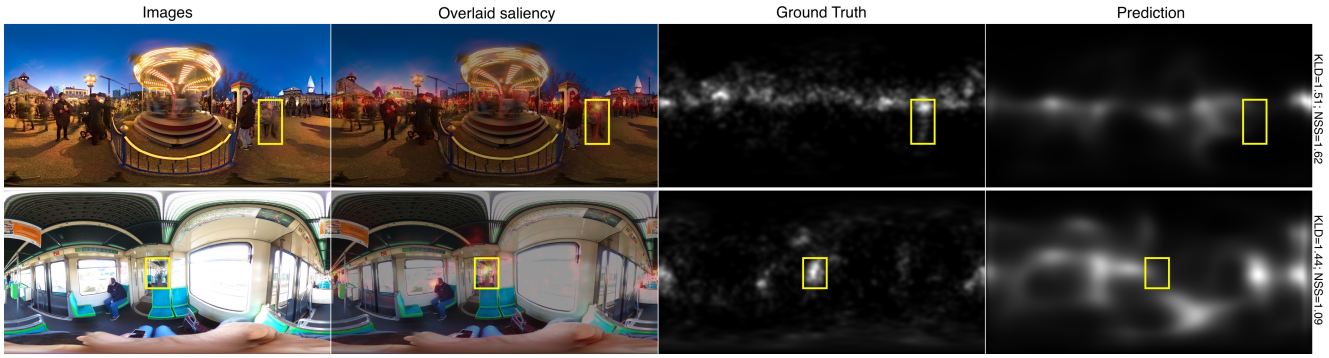

Figure 6: Failure modes of the model.

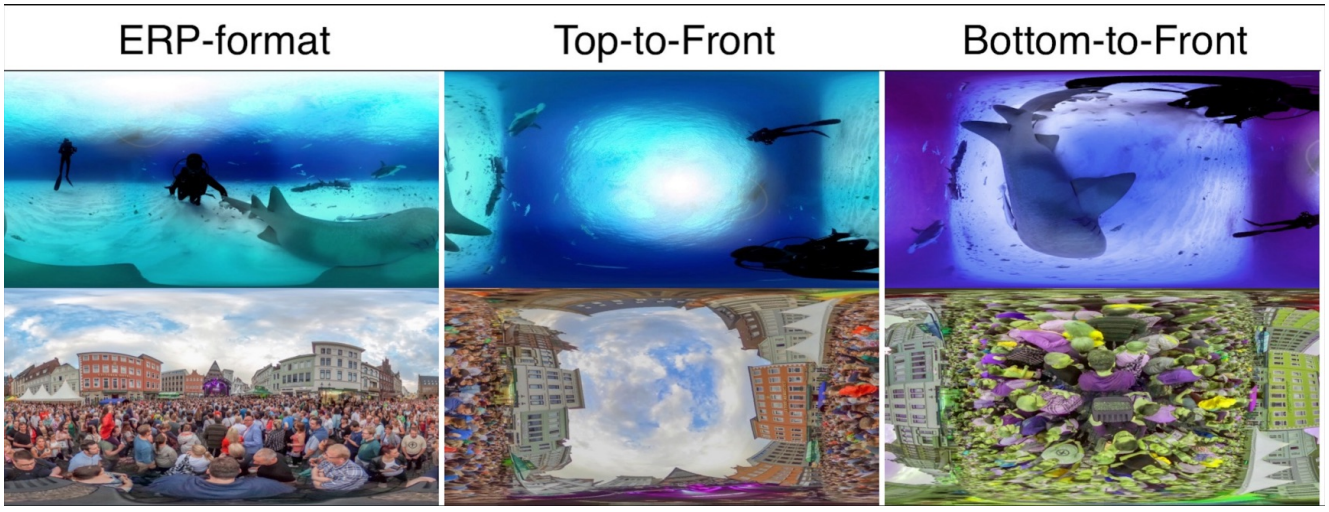

Figure 7: The different views.