Topic H - Referring Image Segmentation via Text-to-Image Diffusion Models

January 22, 2024

Imane SI SALAH Timothée SELOI

Abstract

Diffusion Models (DMs) are a popular type of generative model known for their ability of conditional synthesis, such as text-to-image generation. In this paper, we analyze a method for extracting representations from a pretrained text-to-image Diffusion Model and fine-tuning it for the referring image segmentation task by training a task-specific head. The method is called Visual Perception with pretrained Diffusion models (VPD) and comes from Zhao et al. *[3]. It is a framework that exploits the semantic information of a pretrained text-to-image diffusion model in visual perception tasks. Specifically, the pretrained diffusion model is a latent diffusion model (LDM) as described in Stable Diffusion [2]. Cross-attention maps are utilized between the visual features and the text features to provide the task-specific head with explicit guidance. We complement the VPD framework by allowing users to add noise to the latent representations of the images. Even though the VPD model provided by [3] attains 73.5% oIoU on RefCOCO, our best VPD model only reaches 16.7% oIoU because of our limited GPU access and simplified training configuration.*

1. Introduction

Extensive text-to-image diffusion models [1, 2] have shown remarkable capability in producing a wide range of detailed and realistic images while offering substantial flexibility. This observation implies that large-scale text-to-image diffusion models may inherently learn a broad range of visual concepts, encompassing both high-level and low-level features. Although image generation and visual perception are not straightforwardly related, [3] showed that it is possible to extract the visual knowledge learned by large diffusion models for visual perception task such as depth estimation, semantic segmentation and referring image segmentation.

Our objective was to reimplement and evaluate the Visual Perception with pretrained Diffusion models (VPD) [3] for the specific task of referring image segmentation. Referring image segmentation involves segmenting specific parts of an image based on a textual prompted description. It is challenging due to the need for precise understanding and alignment of visual and textual data.

2. VPD model architecture

The VPD framework comprises the steps explained in this section, as represented in Figure 1.

2.1. Image encoding and diffusion process

First, the image $\mathbf{x}$ is encoded into the latent space by a pretrained VQGAN encoder $\mathbf{z}_0 = \mathcal{E}(\mathbf{x})$ taken from Stable Diffusion [2]. Then, noise is added to this latent representation in a controlled, gradual manner through a diffusion model. We note that the original work by [3] did not include an implementation of the diffusion process. Consequently, as a contribution, we coded and integrated our own solution to address this aspect.

Diffusion models are a family of generative models that can reconstruct the distribution of data by gradually denoising a normally-distributed random variable, which corresponds to learning the reverse process of a fixed Markov Chain of length T. Denoting $\mathbf{z}_t$ as the random variable at t^{th} timestep, the diffusion process is modeled as a Markov Chain:

$$\mathbf{z}_t \sim \mathcal{N}(\sqrt{\alpha_t \mathbf{z}_{t-1}, (1-\alpha_t)\mathbf{I}}) \tag{1}$$

where $\{\alpha_t\}_t$ are fixed coefficients that determine the noise schedule. The above definition leads to a simple close form for $\mathbf{z}_t$:

$$\begin{aligned} \mathbf{z}_t &= \sqrt{\bar{\alpha}_t}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_t}\mathbf{N}, \\ \bar{\alpha}_t &= \prod_{s=1}^{t} \alpha_s \text{ and } \mathbf{N} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \end{aligned} \tag{2}$$

2.2. Text features

The text inputs are short sentences that describe the referred object (e.g. 'table below plate'). Each word corresponds to a token, allowing a numerical representation of each sentence $\mathcal{T}$. By default, every sentence is of a length equal to token_length $= 40$ tokens, commencing with a `SOS` token and concluding with a sufficient number of `EOS` tokens to achieve the desired length of 40 tokens. Text features $\mathcal{C}$ are extracted by a pretrained CLIP encoder from Stable Diffusion [2]. In the original VPD framework [3], a text adapter implemented as a MLP was used to refine the text features obtained by the CLIP encoder, but the instructions indicated not to use it. To sum up, the text features are computed as

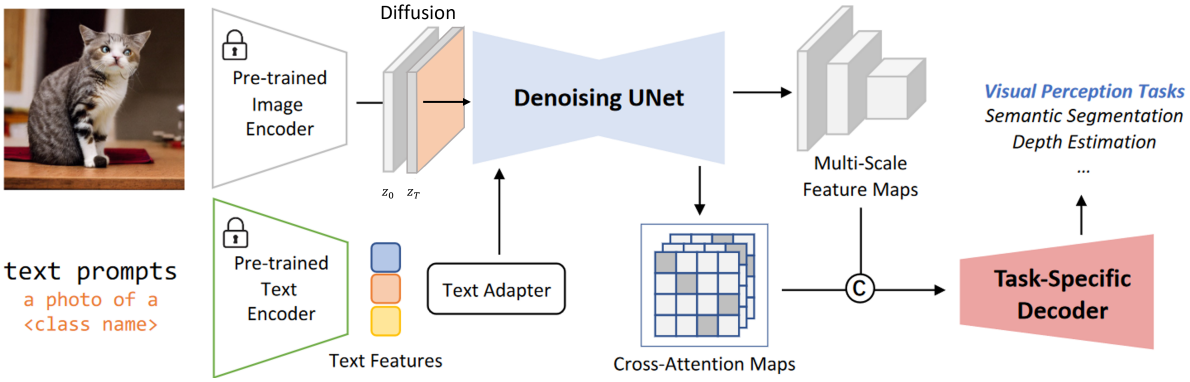

Figure 1. VPD architecture

follows:

$$\mathcal{C} = \text{CLIPTextEncoder}(\mathcal{T}) \tag{3}$$

$$\mathcal{C} = \mathcal{C} + \gamma\text{MLP}(\mathcal{C}) \tag{4}$$

2.3. Denoising UNet

The latent image representation and the text features are fed to a pretrained denoising UNet ϵ_θ obtained from Stable Diffusion [2]. This allows to extract hierarchichal feature maps $\mathcal{F} = \epsilon_\theta(\mathbf{z}_T, T, \mathcal{C})$. Since $\mathcal{C}$ contains information from the natural language, the denoising UNet needs to capture the cross-domain interactions between vision and text.

Typically, the input image $\mathbf{x}$ has size $H_\mathbf{x} = W_\mathbf{x} = 512$ while $\mathcal{F}$ contains four feature maps F_1, F_2, F_3 and F_4 with sizes $H_i = W_i = 2^{2+i}, i \in \{1, 2, 3, 4\}$.

2.4. Semantic guidance via cross-attention maps

Cross-attention maps are extracted from the UNet. They could be used at each level of resolution but the authors of [3] found that at the lowest level of resolution, i.e. $H_1 = W_1 = 8$, these maps are not very accurate. Therefore, we only use them for $i \in \{2, 3, 4\}$. At the i^{th} resolution, the averaged map A_i is obtained by averaging all the cross-attention maps belonging to the resolution. Since the cross-attention maps are computed by using the conditioning inputs $\mathcal{C}$ as the key and value, the averaged attention map has the shape of token_length $\times H_i \times W_i$. We can then concatenate the averaged cross-attention maps with the original hierarchical feature maps and feed the results to the prediction head. In summary, the final feature maps are computed as follows:

$$F_i = [F_i, A_i] \tag{5}$$

Finally, the predicted segmentation is obtained by passing the feature maps to a semantic feature pyramid network (FPN), consisting of several convolutional and upsampling layers. This network incorporates relatively fewer parameters compared to the denoising UNet since the UNet already exhibits capacity to perform downstream vision tasks. In our study, the prediction head is the only model that we train: by freezing all other models, we fine-tune the prediction head for referring image segmentation task.

3. Experiments and results

The following experiments are based on the RefCOCO benchmark which contains in total $142,210$ expressions, associated with $50,000$ reference objects across $19,994$ images. The original training split used by [3] contains $42,404$ reference objects. However, due to limitations discussed later, we trained our models on a subset of $5,000$ objects with validation on 100 objects. As in [3], the performance is assessed using the overall Intersection over Union (oIoU) on the validation set, containing $3,811$ reference objects.

3.1. Codebase adaptation

Our initial step involved adapting the existing VPD framework from Zhao *et al.*'s implementation [3]. We adjusted the libraries and corrected some code to ensure compatibility with our computational setup.

3.2. Evaluation of the pretrained checkpoint

We performed quantitative and qualitative evaluation of a pretrained checkpoint VPD model provided by the authors of [3]. This model involves no diffusion process ($T = 0$). Our evaluation showed an overall IoU of 73.46%, slightly higher than the original 73.25% oIoU reported in the paper, this can be due to differences in computational setup. Further results are presented in Table 1 and Appendix A.1.

3.3. Training models for different noise scales

We performed feature extraction after removing the MLP text adapter and freezing the entire SD model. The hierarchical features occupied 10 Mo space per referred object so storing all of them would have required at least 450 Go for the complete training and test sets. Therefore, we decided

to directly train the semantic FPN head without storing the multi-scale features.

As mentioned earlier, we also fine-tuned the task-specific head after adding various noise scales $T \in \{0, 5, 100, 200, 500\}$ to the latent representations of the images. Introducing noise to the latent representations prevents the model from overfitting to the training data. It encourages the model to learn more robust and generalized features that are not overly specific to the training set, ultimately improving its performance on unseen data.

We trained the models with an AdamW optimizer comprising a $5 \cdot 10^{-5}$ learning rate and a 0.01 weight decay. The total batch size was set to 8. We limited the number of epochs to 20 while authors of [3] trained for 40 epochs.

3.4. Evaluation of our trained models

The Table 1 summarizes the quantitative evaluation of the models on the validation set, composed of $3,811$ referred objects. The main performance indicator is the oIoU, but we also give the precision at different IoU levels. For instance, if a model achieves a 83 score for p@0.6, it means that 83.0% of the objects referred by the model have an IoU greater than or equal to 60%. Table 1 indicates that models with a large noise scale ($T \geq 100$) better adapt to unseen data than models with a low noise scale ($T \leq 5$). Qualitative results are presented in Appendix A.1 and A.2.

Model	oIoU	p@0.5	p@0.6	p@0.7	p@0.8	p@0.9
Authors	73.5	85.5	83.0	78.5	68.5	36.3
$T = 0$	11.54	4.4	2.29	0.94	0.31	0.0
$T = 5$	12.39	4.58	2.17	0.91	0.22	0.0
$T = 100$	**16.7**	7.1	3.9	1.8	**0.7**	**0.1**
$T = 200$	15.56	6.48	3.75	1.85	0.63	**0.1**
$T = 500$	16.38	**7.46**	**4.12**	**1.98**	0.61	0.03

Table 1. Performance Metrics of the VPD models

3.5. Visualization of cross-attention weights

As described in the subsection 2.4, each token is associated with an average cross-attention map of shape $H_i \times W_i, i \in \{2, 3, 4\}$. This makes their visualization easy to interpret.

Analyzing averaged cross-attention maps enhances our understanding of the model's decision-making. The Appendix A.3 contains examples illustrating this. In the Figure 15, attention maps align well with the input image, while in the Figure 16, attention maps fail to locate the referenced object (the man wearing a black T-shirt), resulting in an inaccurate segmentation.

The Figure 2 illustrates A_3 for the word 'slice' after applying a threshold (where 80% of the lowest values are zeroed out for visualization). This map is of shape 32×32.The visualization of the attention map explains why the model included the center slice but excluded the two bottom slices.

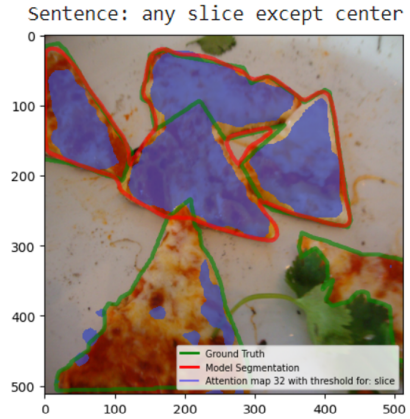

Figure 2. Cross-attention map 32 with threshold for 'slice'

4. Discussion

Throughout the experiments, we encountered a series of challenges for which we had to come up with solutions in order to effectively train and evaluate the VPD model.

4.1. Challenges

- **Limited storage space:** The necessity to store the multi-scale features, requiring around 450 Go for the 45k training/test referred objects, posed an important challenge given our limited storage capacity.
- **Limited GPU access:** The authors of the paper recommended using 8 GPUs '32G NVIDIA V100'. However, such resources were not accessible to us.
- **Limited time:** Training the task specific head was time-intensive. With 45k referred object, training on one epoch would take 13h, translating to 22 days for the 40 epochs.
- **Training multiple noise-scale models:** The need to train and evaluate the model across a range of noise scales $T \in \{0, 5, 100, 200, 500\}$ added complexity to our experimental setup. Initially, we considered unfreezing the UNet backbone for joint training with the decoder head. However, due to limited GPU RAM, this approach was not feasible. As a result, we solely trained the decoder head, excluding the diffusion process.

4.2. Solutions:

- **Direct training of Task-Specific Head:** To manage the storage limitations, we trained the task specific decoder directly without storing the features.
- **Utilizing available GPU resources:** We leveraged 2 GPUs '16G NVIDIA T4' and also utilised our free Colab accounts. This required $150 Google Cloud credits and at least 7 days to complete training and evaluation.
- **Training on a subset of data:** we opted to train on a subset of 5k referred objects (instead of 42k) for 20 epochs. This approach still required substantial time, with 16 hours of training on 2 GPUs and 32 hours on 1 GPU.

References

[1] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv*, 2022. 1

[2] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. *CVPR*, 2022. 1, 2

[3] Wenliang Zhao, Yongming Rao, Zuyan Liu, Benlin Liu, Jie Zhou, and Jiwen Lu. Unleashing text-to-image diffusion models for visual perception. *CVPR*, 2023. 1, 2, 3

A. Qualitative Evaluation Results

In this appendix section, we display all the qualitative results we obtained after performing all our experiments.

A.1. Authors' checkpoint

The Green contour is the ground truth mask while the Red one is the model prediction.

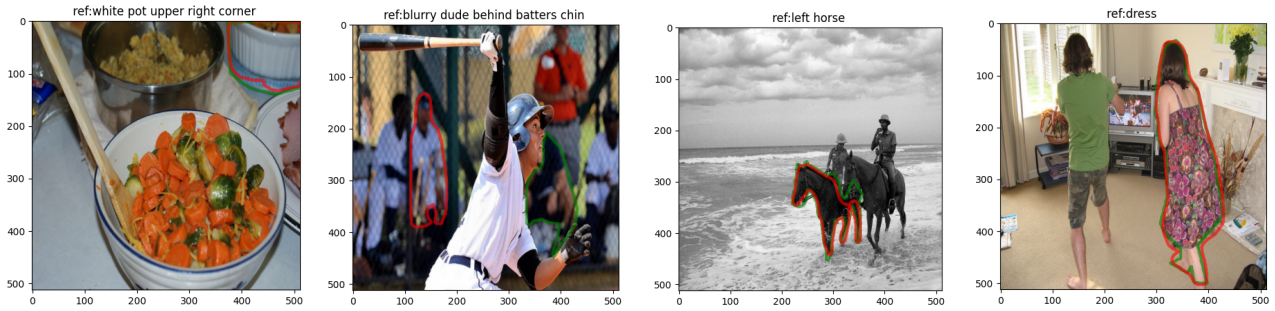

Figure 3. Qualitative evaluation of authors' checkpoint

A.2. Our training results

Here are displayed the resulting the results we obtained for different noise scales. The Green contour is the ground truth mask while the Red one is the model prediction.

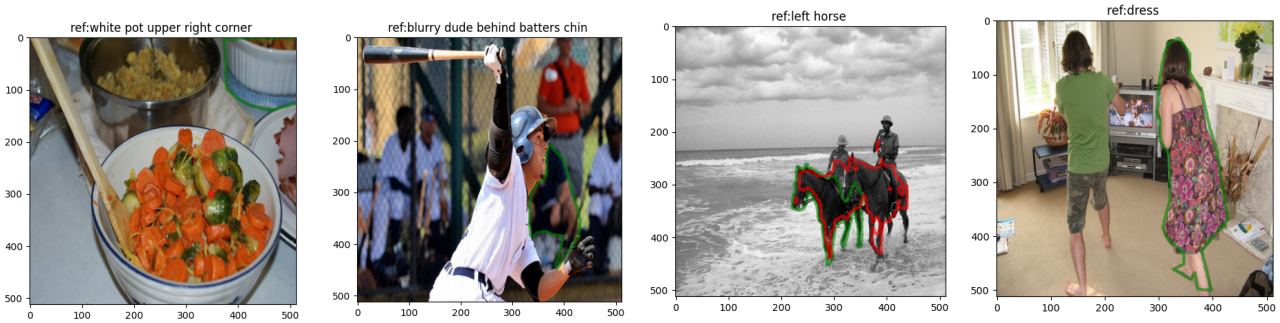

Figure 4. Qualitative evaluation of our checkpoint for $T = 0$

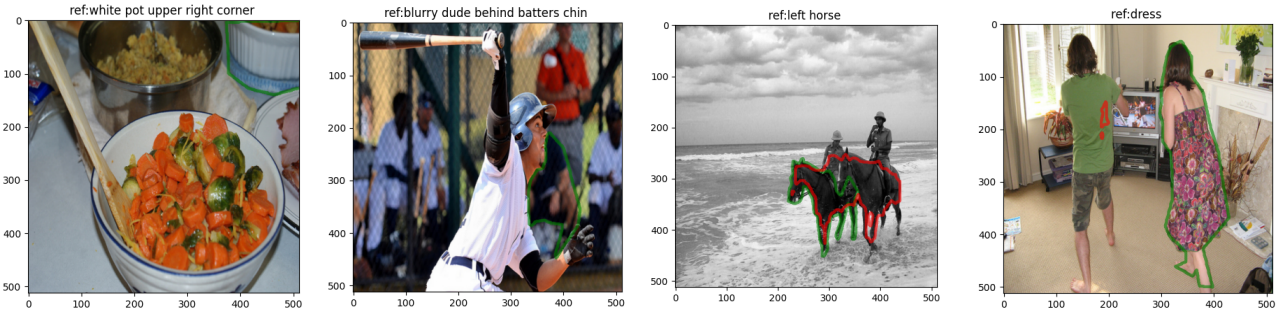

Figure 5. Qualitative evaluation of our checkpoint for $T = 5$

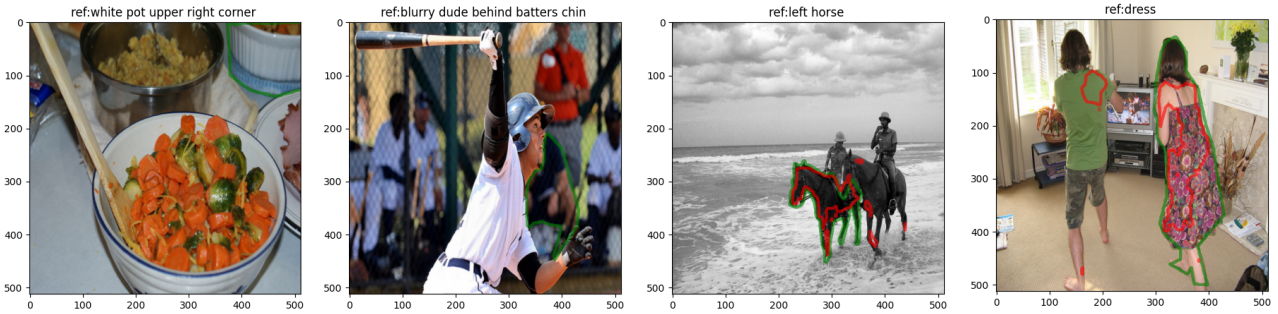

Figure 6. Qualitative evaluation of our checkpoint for $T = 100$

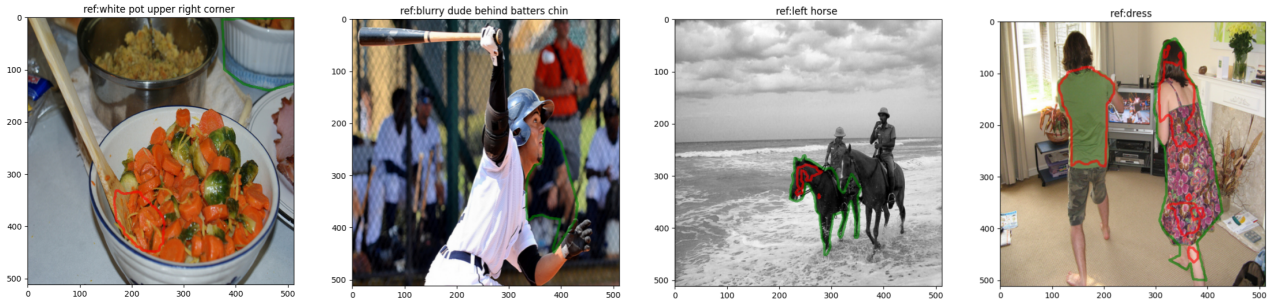

Figure 7. Qualitative evaluation of our checkpoint for $T = 200$

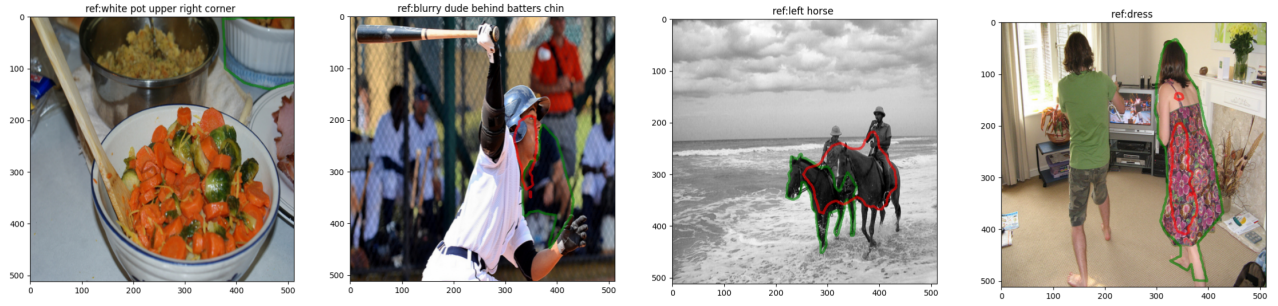

Figure 8. Qualitative evaluation of our checkpoint for $T = 500$

A.3. Cross-attention weights at different noise scales

In this subsection, we visualize the cross-attention maps for two different images. The Green contour is the ground truth mask while the Red one is the model prediction. The represented maps are the A_2 and A_3 averaged cross-attention weights. For each token, their shapes are 16×16 and 32×32 respectively.

A.3.1 Image associated with the sentence 'white pot upper right corner'

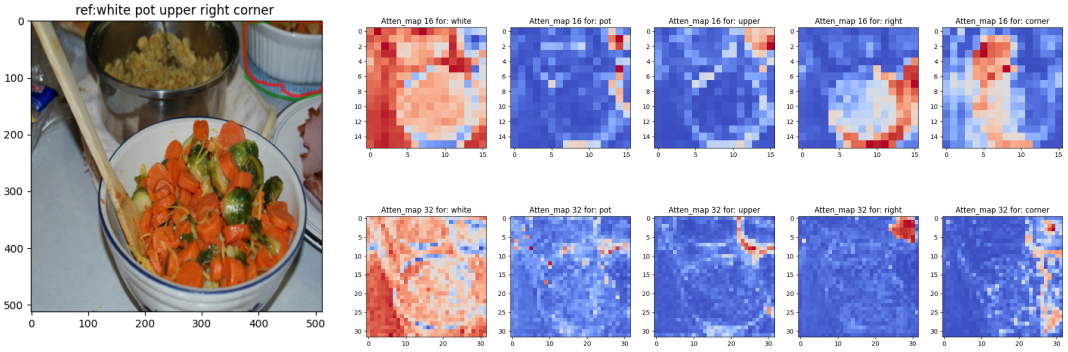

Figure 9. Attention maps 16 and 32 per token for authors' checkpoint

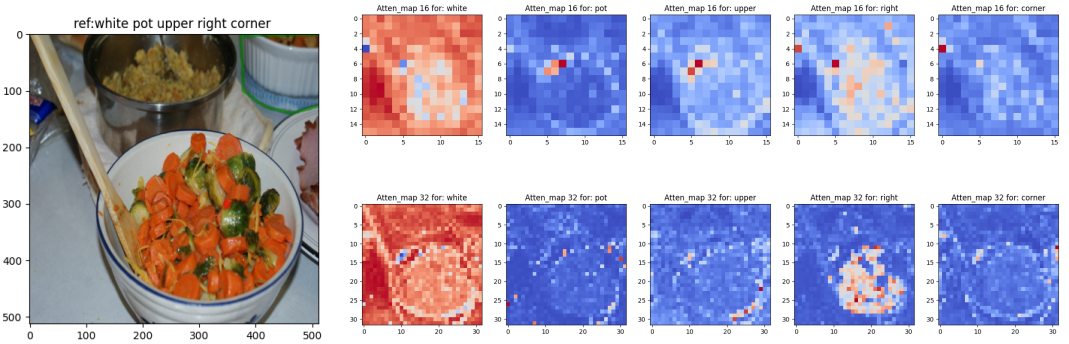

Figure 10. Attention maps 16 and 32 per token for our checkpoint with $T = 0$

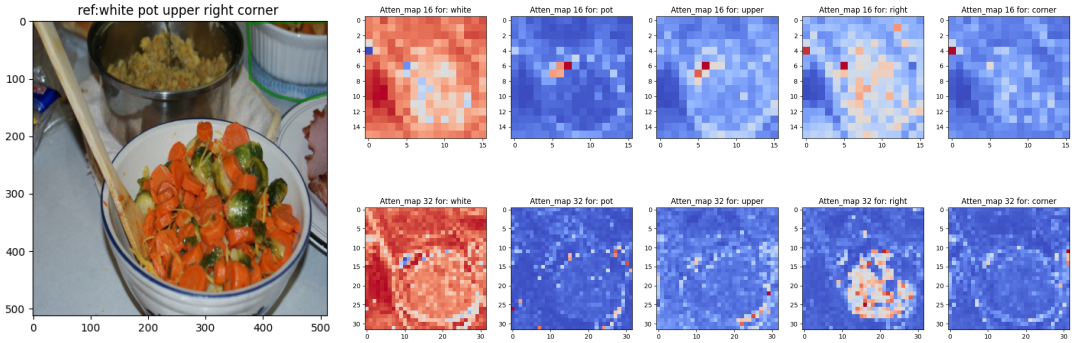

Figure 11. Attention maps 16 and 32 per token for our checkpoint with $T = 5$

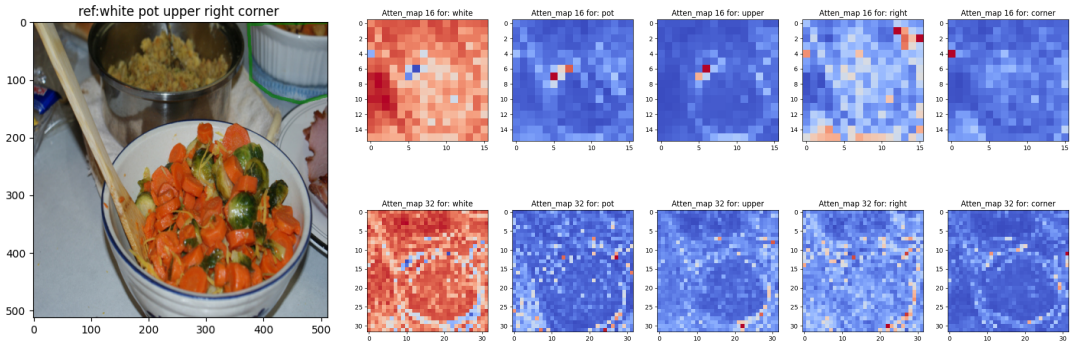

Figure 12. Attention maps 16 and 32 per token for our checkpoint with $T = 100$

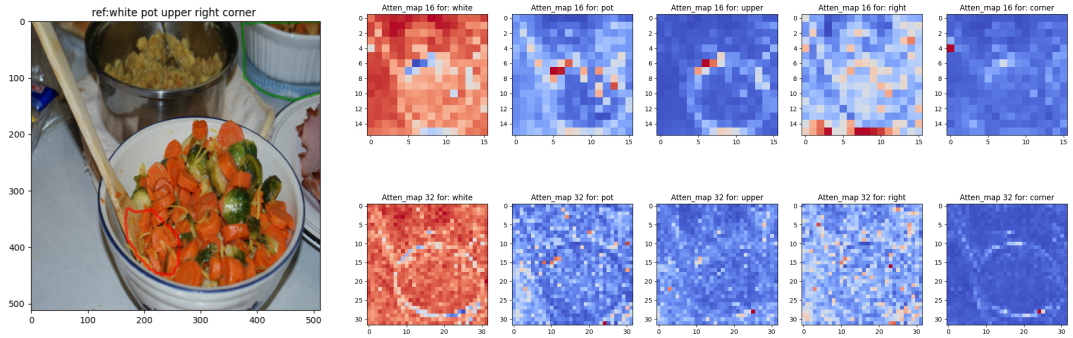

Figure 13. Attention maps 16 and 32 per token for our checkpoint with $T = 200$

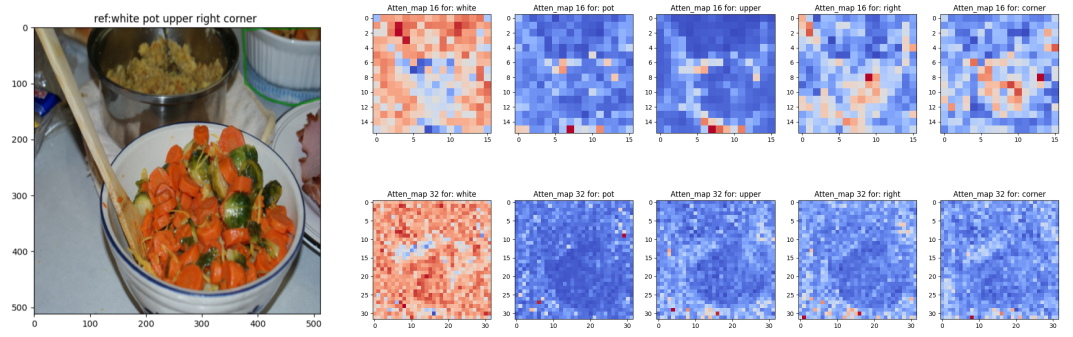

Figure 14. Attention maps 16 and 32 per token for our checkpoint with $T = 500$

A.3.2 Image associated with the sentence 'guy in black'

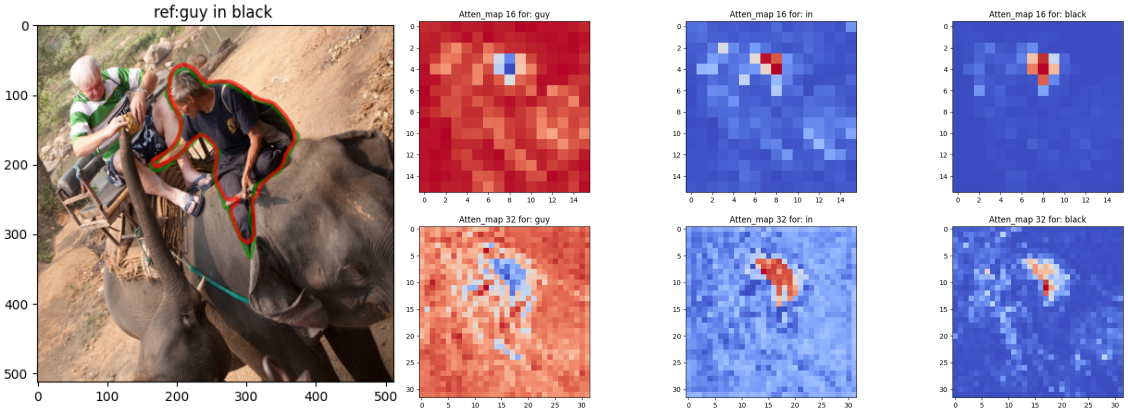

Figure 15. Attention maps 16 and 32 per token for authors' checkpoint

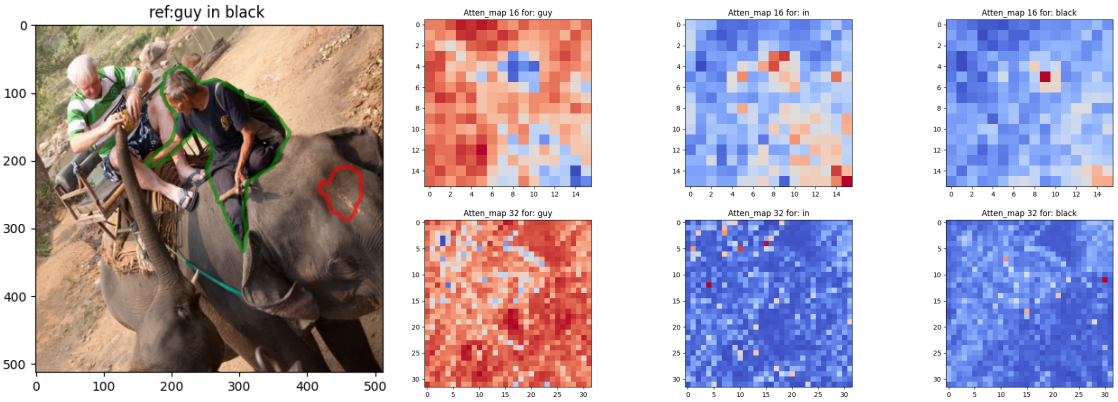

Figure 16. Attention maps 16 and 32 per token for our checkpoint with $T = 0$

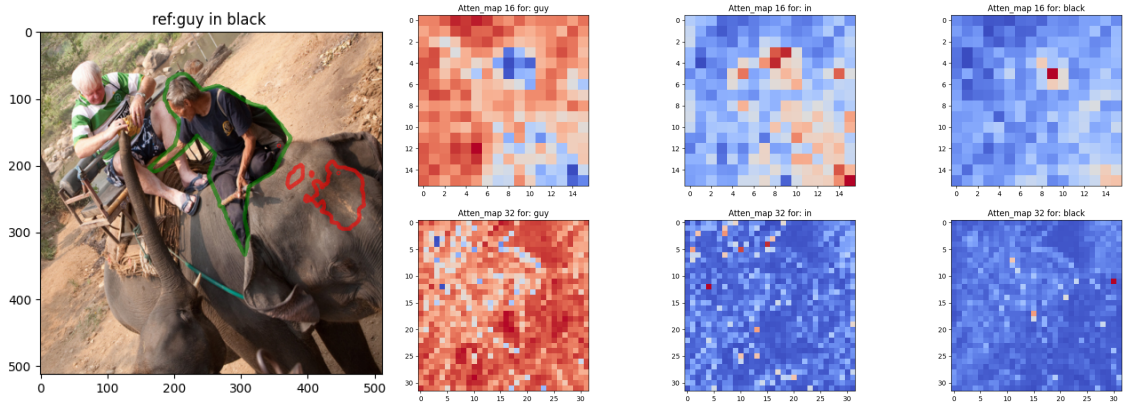

Figure 17. Attention maps 16 and 32 per token for our checkpoint with $T = 5$

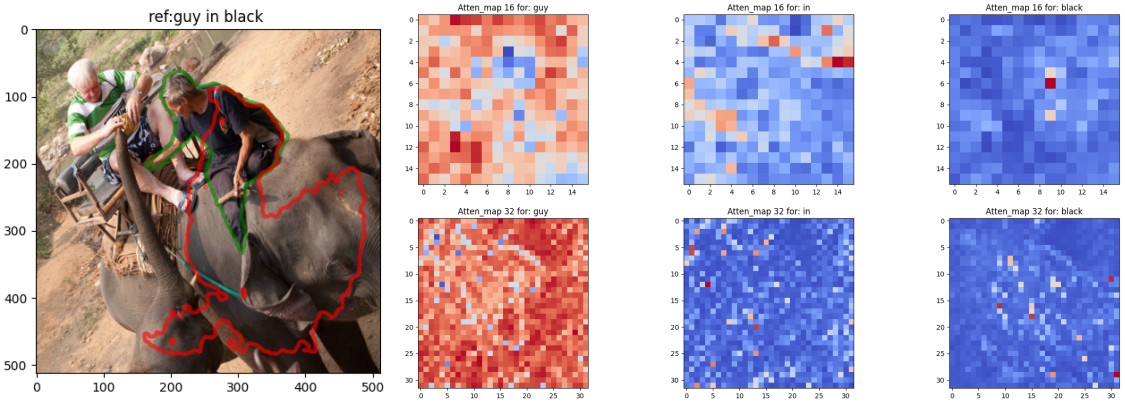

Figure 18. Attention maps 16 and 32 per token for our checkpoint with $T = 100$

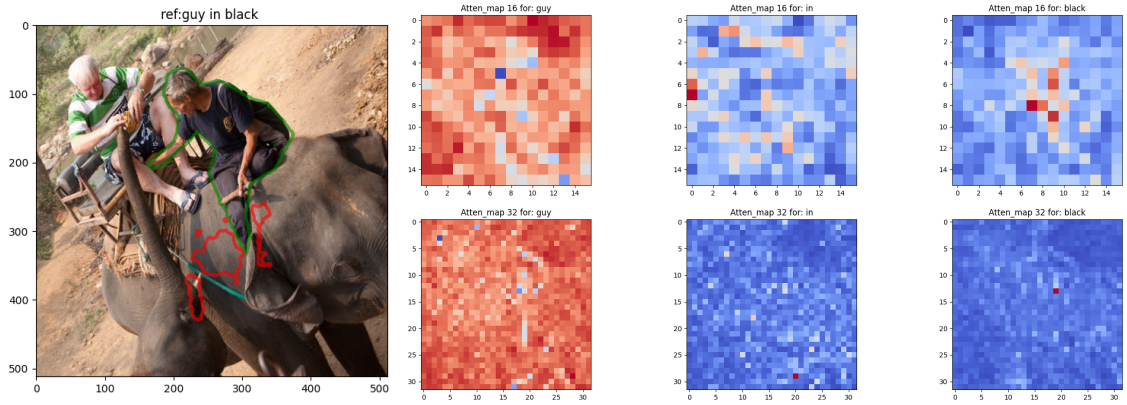

Figure 19. Attention maps 16 and 32 per token for our checkpoint with $T = 200$

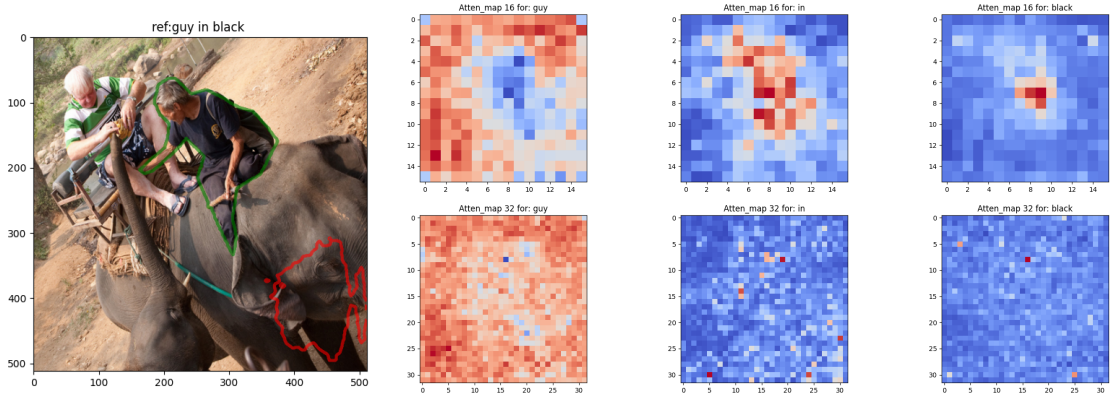

Figure 20. Attention maps 16 and 32 per token for our checkpoint with $T = 500$