Attention Chains - Supplementary Material

A Implementation Details

A.1 ImageNet Zero-Shot Segmentation

To establish fair comparisons, we followed the exact procedure described by Helbling et al. (2025) using the code from `https://github.com/helblazer811/ConceptAttention`. We simplify all categories to a single word, allowing methods conditioned on text to select the corresponding token. We use the timestep-distilled Flux-Schnell DiT (Labs, 2024) and add Gaussian noise to each of the images to the normalized timestep $t = 0.5$, resulting in two backward steps for denoising the image. We use the last 10 layers of the multi-stream attention blocks. The attention matrices from all aforementioned layers and timesteps are averaged. For the heads, we use the λ_2 weighting for averaging. For multi-bounce attention, we use the column-select formulation, i.e. we replace A with A_l and transpose as described in Section 4.2 of the paper.

A.2 Linear Probing of TokenRank

We generate attention maps for all layers, heads, and images for the ImageNet (Deng et al., 2009) subset *Imagenette* (Howard, 2019) for DINOv1, DINOv2, and CLIP with a VIT/B model. Then, we compute attention visualizations with column sum, TokenRank, row-selecting the center token, and row-selecting the CLS token. Finally, we train a linear classifier with the commonly used train and test split and with the attention visualizations of all layers and heads as input, optimized with the SGD optimizer with a set of learning rates ($\{1e-5, 2e-5, 5e-5, 1e-4, 2e-4, 5e-4, 1e-3\}$)) for 20 epochs, and we choose the best performing model, similar to Oquab et al. (2023). We resize and center crop the images to get 905×905 (901×901) attention maps for DINOv2 with registers (other models). We exclude the CLS token column for classification.

A.3 Improving Self Attention Guidance (SAG)

For the SAG experiments, we build on top of the code of (Hong et al., 2023) provided in `https://github.com/SusungHong/Self-Attention-Guidance`. We use SD1.5 (Rombach et al., 2022) as the foundational model and use the 2nd last decoder block's last self-attention layer for masking. We empirically observed it consistently resulted in the best results for both SAG and SAG+TokenRank, as opposed to the usage of the bottleneck layer in the comparison experiments in the original SAG, which is in line with the observation by Hong et al. (2023).

We compute the IS (Salimans et al., 2016), FID (Heusel et al., 2017), and KID (Bińkowski et al., 2018) metrics over 50K generated images, where the ground truth dataset is the LAION-Aesthetics V2 dataset (Schuhmann et al., 2022) - originally used to train SD1.5. We used the $score > 6.5$ subset consisting of 625K images, and filtered it for $score > 6.65$ and images with width and height larger than 512, resulting in 72495 raw links, out of which roughly 50K were valid. After truncating to 50K, these images were used for computing the metrics. The selected metrics offer complementary information in terms of quality (IS) and diversity and resemblance to ground-truth distribution (FID / KID).

We use the unconditional branch only, and allow up to 40% of the attention matrix to be masked, following the original implementation.

Preprint. Under review.

A.4 Masking Most Influential Tokens

For all transformers, we do not allow masking the CLS token in our evaluations, as it consistently ranks as the top most important token for almost all heads and layers, and discarding it yields an extreme drop in accuracy, which does not allow for comparisons of the alternative approaches. For the same reason, we also only use the first half of the available attention layers for each model. This allows for not disturbing the output space too much and as a result accuracy performance degrades gracefully. Refer to Appendix F.2 for further justification of this choice. Moreover, we mask out the same number of tokens for all heads simultaneously. For DINO family type, results in the main paper are averaged over the ViT-B/16, ViT-B/32, ViT-S/16, and ViT-S/32 architectures. For DINOv2 family type, we use the w/ registers variation, and average over the ViT-B/14, ViT-S/14 and ViT-L/14 architectures. For both DINO and DINOv2 family types, we use pretrained linear classifier heads provided with the original publications. We average ViT/B and ViT/L for the transformer models trained in a supervised way. For CLIP family type, we average over the ViT-B/32, ViT-L/14 and ViT-L/14/336 architectures. We use the template prompt "A photo of a <class>" to compute per-class text features. Then, we compute a dot-product with the image features and these per-class features to determine the classification result. Masking is performed by setting the entire column corresponding to a token to $-\infty$ prior to softmax. For a more in-depth view of individual model accuracy degradation, see Appendix F.

B Ablation and Hyperparameters for Zero-Shot Segmentation

See Table 1 for zero-shot ImageNet segmentation results using the same experimental setup as in the main paper, with different hyper parameter choices. The bottom of Table 1 shows "outgoing ($n = 2$)*", corresponding to the result shown in the main paper, leveraging the 2nd bounce from a one-hot column vector and utilizing λ_2 weighting. "row-select ($n = 1$)" and "column-select ($n = 1$)" are the naive first bounce operation also shown in the main paper. "single-channel" uses the single channel attention blocks in FLUX rather than the dual-channel blocks. "incoming ($n = 2$)" uses the incoming attention formulation of multi-bounce-attention (i.e. 2nd bounce from a one-hot row vector). "incoming ($n = 3$)" is the same but with 3 bounces. "outgoing ($n = 3$)" uses a one-hot column vector with 3 bounces. "w/o λ_2" does not use the weighted λ_2 scheme. Notice the performance for outgoing attention degrades rapidly with more bounces ("outgoing ($n = 2$)*", "outgoing ($n = 3$)") compared to incoming attention ("incoming ($n = 2$)", "incoming ($n = 3$)"). This is because for outgoing attention, the steady-state eventually converges into emphasizing important "hubs", which usually tends to be background information. However, in early bounces ($n = 1, 2$) it out-performs the incoming attention, and we speculate this is because it captures better the target of this task: we are interested in finding out which image tokens "attend-to" the text token of interest (i.e. a many-to-one relationship).

Table 1: **ImageNet segmentation.** Ablation and hyper-parameter choice.

Method	Acc ↑	mIoU ↑	mAP ↑
row-select ($n = 1$)	73.96	54.65	82.64
column-select ($n = 1$)	80.55	64.02	87.20
single-channel	69.63	51.73	80.50
incoming ($n = 2$)	83.73	69.58	93.68
incoming ($n = 3$)	82.93	68.81	94.01
outgoing ($n = 3$)	79.25	63.32	89.80
w/o λ_2	84.00	70.02	94.28
outgoing ($n = 2$)*	**84.12**	**70.20**	**94.29**

C Used Compute

The image segmentation experiments with FLUX required around 1500 GPU hours, where one experiments takes around 50 GPU hours. Extracting all attention map visualizations for the linear probing experiment and training the linear classifier took around 300 GPU hours. We performed

around 40 experiments for the masking experiment, where each experiment required around 30 GPU hours, resulting in around 1200 GPU hours in total. Finally, the SAG experiment took around 1000 GPU hours. In total, we estimate the use of around 4000 GPU hours for the whole study. For all computations, we used an internal GPU cluster consisting of NVIDIA A40, A100, H100, and RTX 8000 GPUs.

D More Segmentation Results

See Figure 7 for more zero-shot segmentation results on ImageNet with comparisons to the state-of-the-art work ConceptAttention (Helbling et al., 2025). We argue that our raw segmentation masks are cleaner and more precise, which can be qualitatively measured by the threshold-agnostic mAP metric presented in the main paper. Additionally, in Figure 1 we show our multi-bounce attention can be used for segmenting different parts of an image by setting the initial one-hot vector to the corresponding text token. The text prompt used to create the image (FLUX-Schnell) was "cute black cat standing up wearing red boots and a bow tie, photorealistic, masterpiece, macro wildlife photography, dewy grass, closeup low angle, wildflowers, sun, shadows, depth of field, desaturated, film grain, low quality".

Figure 1: **Part segmentation.** We can segment different parts of the image by setting the initial one-hot vector to the corresponding text token.

E Illustration of Bouncing as Consolidation Mechanism

Figure 2 illustrates that iterating over the Markov chain results in an iterative refinement of the map, consolidating meta-stable states. The steady state eventually visualizes the global incoming attention. This effect cannot be observed for an input image with random pixels (Figure 2, bottom). Instead, the Markov chain converges very quickly to a dispersed visualization after one step for a noisy input, as the attention matrix in not structured. We compute the visualizations by passing the depicted images through DINOv1 and extracting the attention map in the 9th layer and 3rd head. Then, we compute the first bounce ($n = 1$) with a uniform initial vector, the 2nd bounce, and the steady state. The same effect can be seen in Figure 7, where the second bounce (*Ours*) is significantly cleaner and better adheres to the depicted objects compared to the first bounce (*Column Select*).

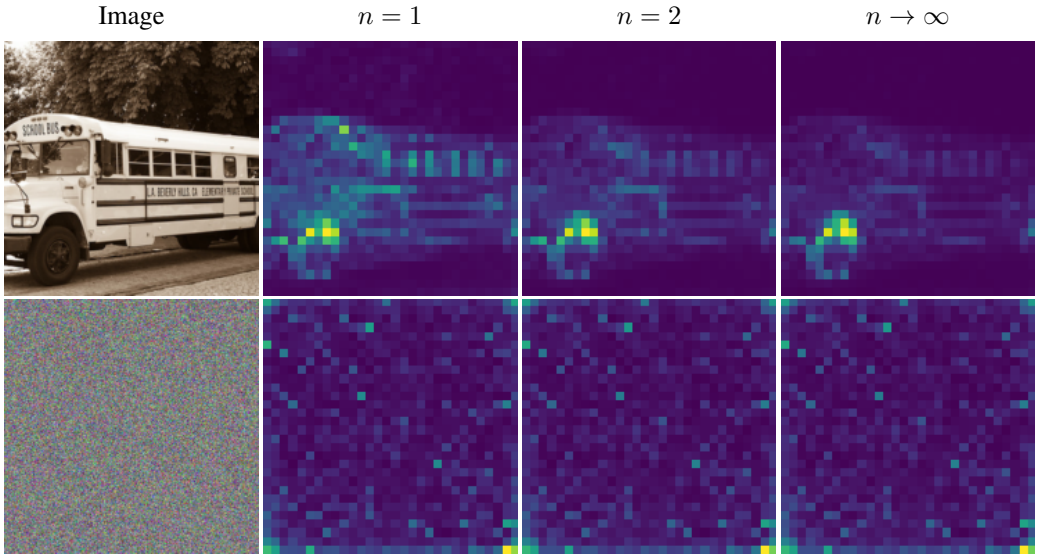

Figure 2: **Illustration of bouncing.** Bouncing the attention signal sharpens the DINOv1 attention map for a realistic input image, supporting the existence of meta-stable states. On the contrary, a random input image results in a fast converging Markov chain, where intermediate bounces to not differ clearly from the first bounce and disperse rather than cluster. This observation is also reflected in the second eigenvalues: $\lambda_2 = 0.44$ for the real image and $\lambda_2 = 0.16$ for the random image, where a smaller second eigenvalue corresponds to a faster convergence.

F Additional Results for Masking Most Influential Tokens

F.1 Per-Model Results

In Figure 3 we show the results of performing the most-influential token masking experiment described in the main paper for individual models. The area-under-curve (AUC) metric in the main paper is computed over the curves.

F.2 Per-Layer Results

In Figure 4, we show the result of choosing progressively more number of layers to mask for the most-influential token masking experiment conducted in the main paper. "6 (Ours)" indicates what we used in the paper. Layers are selected starting from the furthest from the output. The results are shown for DINOv2 (ViT-S/14) w/ registers over the same dataset used in the main paper. However, similar trends were observed for all architectures. We opted for masking tokens from only the first 50% of the layers to avoid distorting the output space and a too steep performance drop (which happens for $n > 6$), while still having an effect on the classification results for higher token mask percentage (which does not occur for $n < 6$).

G λ_2 Weighting for Masking Most Influential Tokens Experiment

We performed another token masking experiment similar to the one conducted in of the paper, only using TokenRank for determining the masking order. In this experiment, we compare using uniform weights per head (as in main paper), random head weights, and using λ_2 weighting. Results can be seen in Figure 5. Perhaps unsurprisingly, randomly weighting the heads performs better than uniform weights, as the transformer is more sensitive to individual heads being fully or almost fully masked out, rather than gradually masking out all heads together. Importantly, results indicate that λ_2 weighting can be used for increasing the degradation in accuracy, suggesting it can serve as a useful tool for determining importance of heads. This was also observed to improve zero-shot segmentation results (see Appendix B).

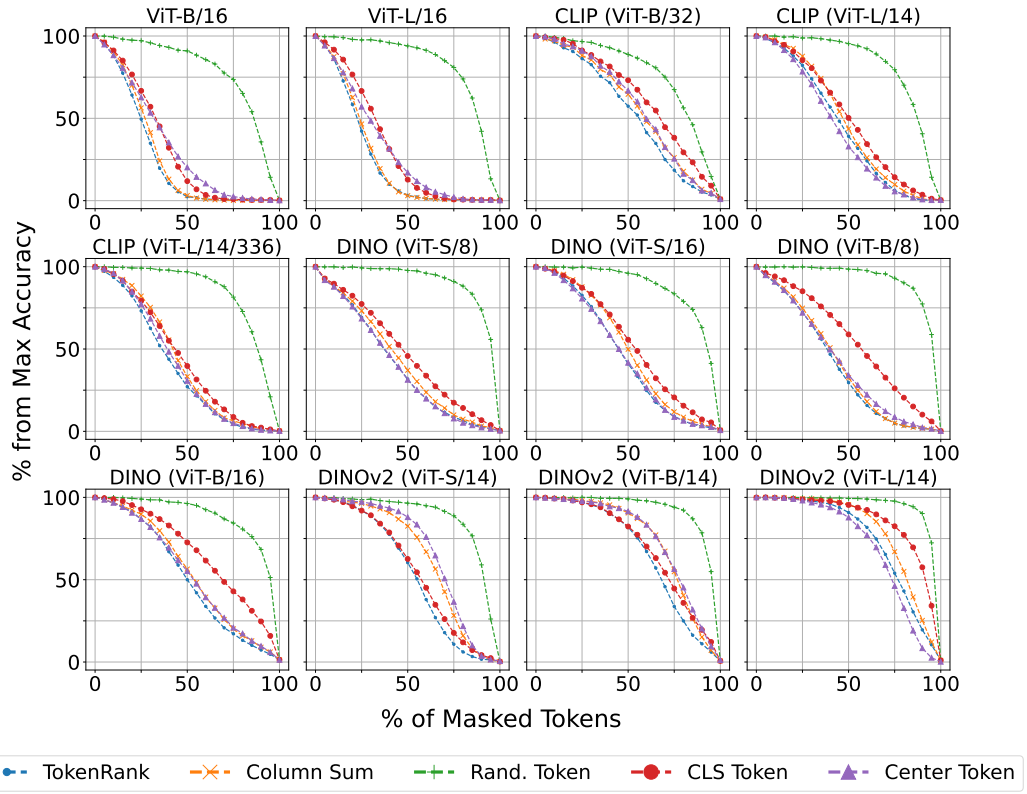

Figure 3: **Token masking.** Using TokenRank to mask out tokens better degrades performance on average across multiple architectures and training types.

Linear Probing Experiment Additionally, we also train a linear classifier on top of the TokenRank vectors (incoming attention) per layer, where we use different weighting schemes for the head dimension. For this, we first compute the weighting coefficients per head and average using them, followed by computing TokenRank for the aggregated result. We compare uniform weighting (simple average), random weighting, and λ_2 weighting. Results are presented in Table 2: While random weighting is significantly worse, λ_2 weighting performs better than simple averaging, particularly for transformers with a well-structured latent space, such as DINOv2 with registers.

Table 2: **Linear probing of steady state vectors after aggregating heads.** λ_2 weighting is better than uniform weighting.

Method	uniform	random	λ_2
CLIP	**49.07**	44.07	**49.07**
DINOv1	53.50	47.21	**53.55**
DINOv2	71.62	50.29	**72.36**

H More SAG Results

In Figure 6 we show more examplary results comparing vanilla SD1.5 (Rombach et al., 2022), SAG (Hong et al., 2023) and SAG+TokenRank.

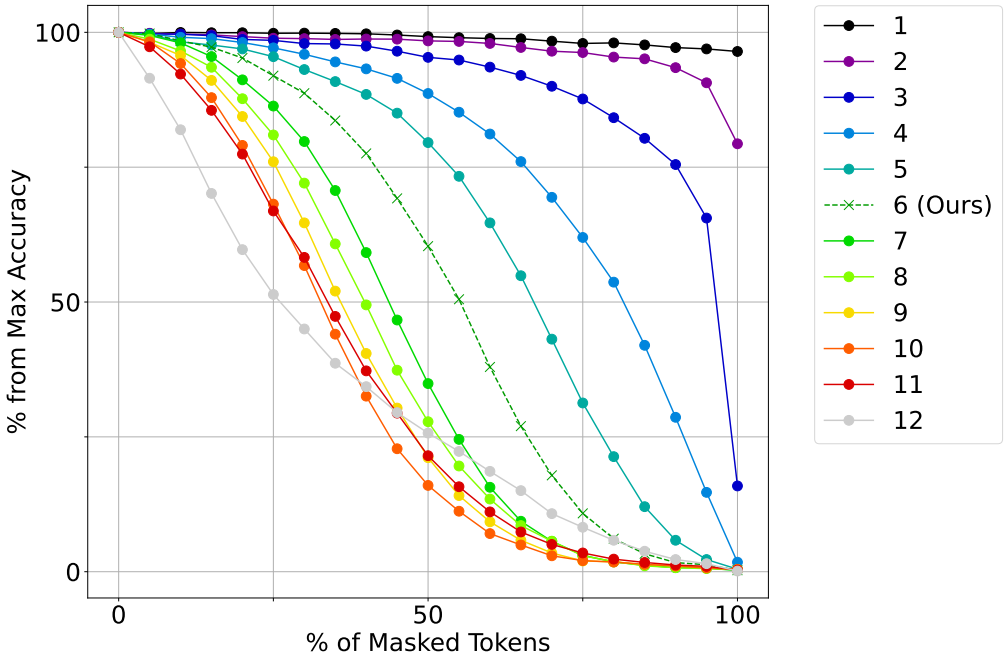

Figure 4: **Layer selection for DINOv2 (ViT-S/14).** Numbers indicate the number of attention layers being masked starting from the closest to the input. Masking fewer layers introduces noise into measurements, while masking more of them results in a faster drop in accuracy allowing less room for comparisons. We chose to mask 50% of the layers for all experiments; in this case "6 (Ours)" variation was used.

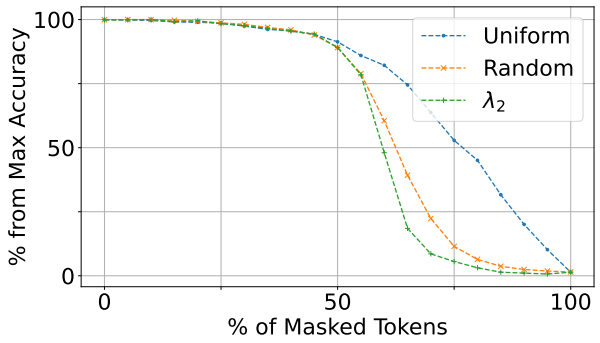

Figure 5: **Aggregating heads.** We used TokenRank to mask tokens while using different weighting schemes for the head dimension. λ_2 yields a larger drop in accuracy compared to uniform or random weighting.

I Per-Head TokenRank Visualizations

We show more visualizations for TokenRank, column sum, Center Patch, and CLS token for various heads and layers in Figure 8 and Figure 9.

J Visualization Across Generation Timesteps

In Figure 10, we show generated image results of using FLUX-Dev with 50 timesteps. We computed the TokenRank (incoming attention) and average over every 5 consecutive timesteps shown in every

column (first entries correspond to noisier timesteps, while last entries are more noise-free). The TokenRank visualizations remain stable through the timestep dimension. It can be observed how attention maps are getting sharper during the denoising process, illustrating that TokenRank can serve to visualize and analyze generative models.

Figure 6: **Self Attention Guidance.** Using TokenRank produces less artifacts and more structured images.

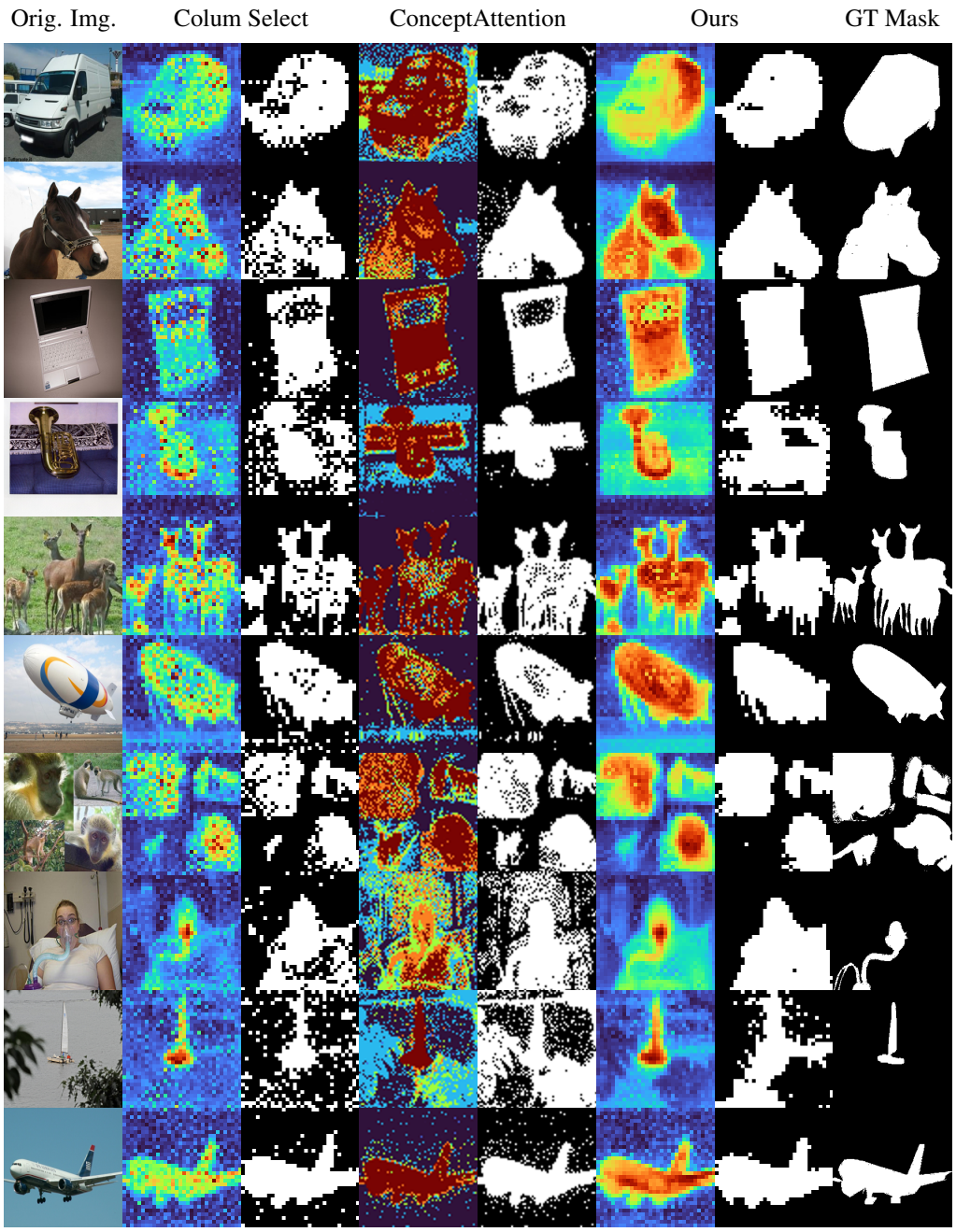

Figure 7: **ImageNet Zero-Shot Segmentation.** Our results are on par with state of the art ConceptAttention (Helbling et al., 2025).

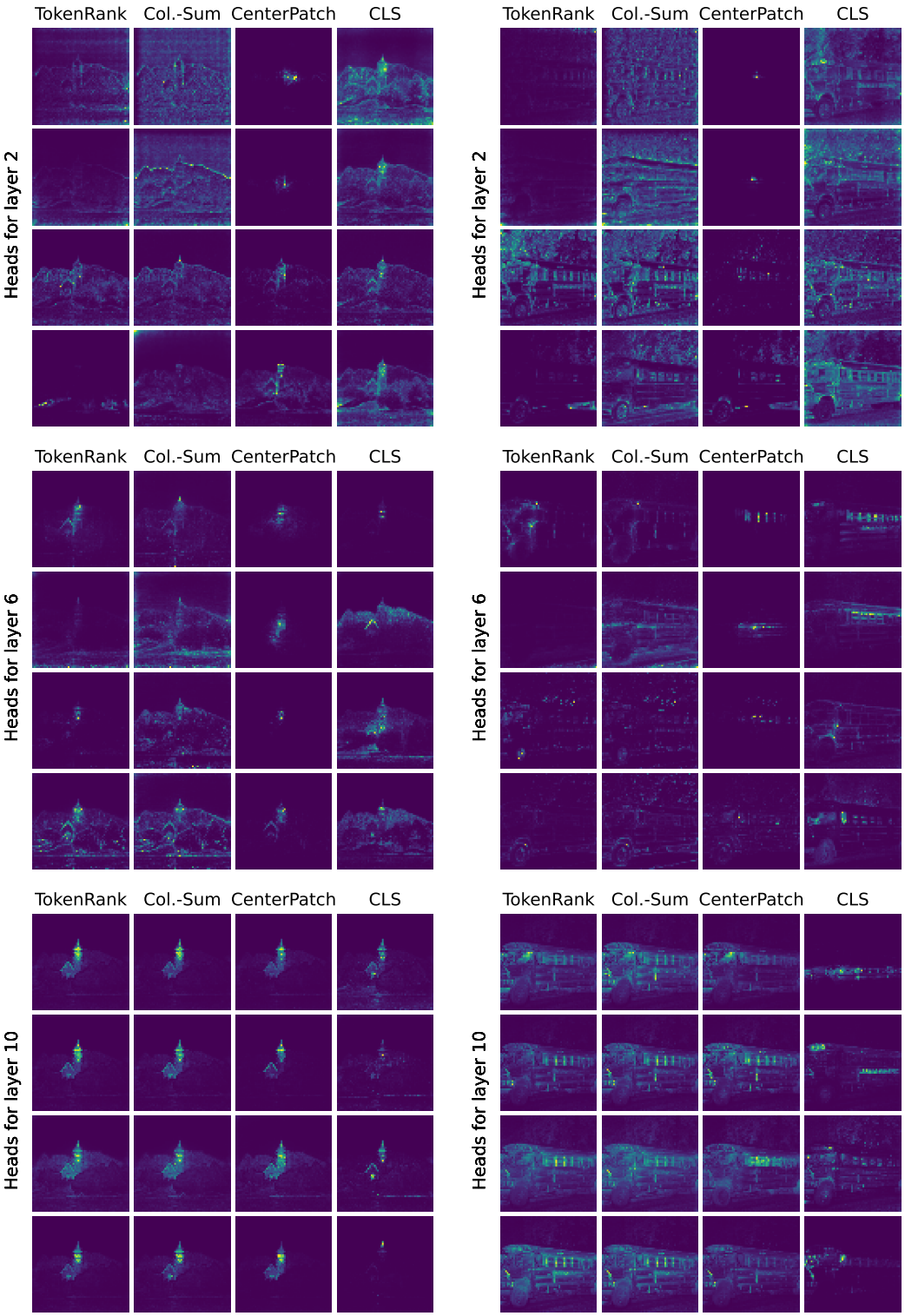

Figure 8: **Per-head visualizations of global incoming attention.** We plot for various layers and the first four heads for DINOv1 (ViT-B/8) with different visualization strategies. Images correspond to the upper row of Figure 4 in the main paper.

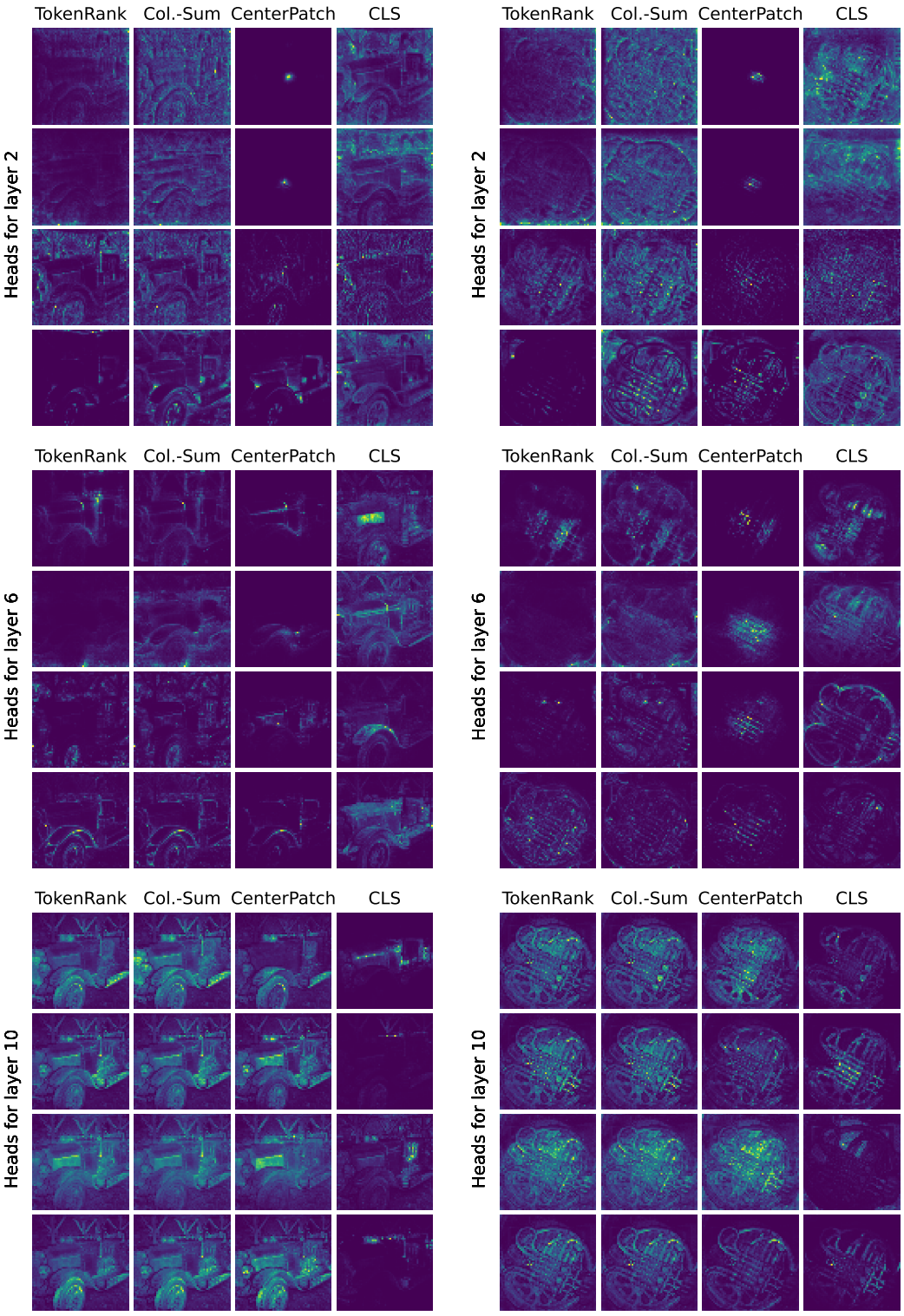

Figure 9: **Per-head visualizations of global incoming attention.** We plot visualization for various layers and the first four heads for DINOv1 (ViT-B/8) with different visualization strategies. Images correspond to the lower row of Figure 4 in the main paper.

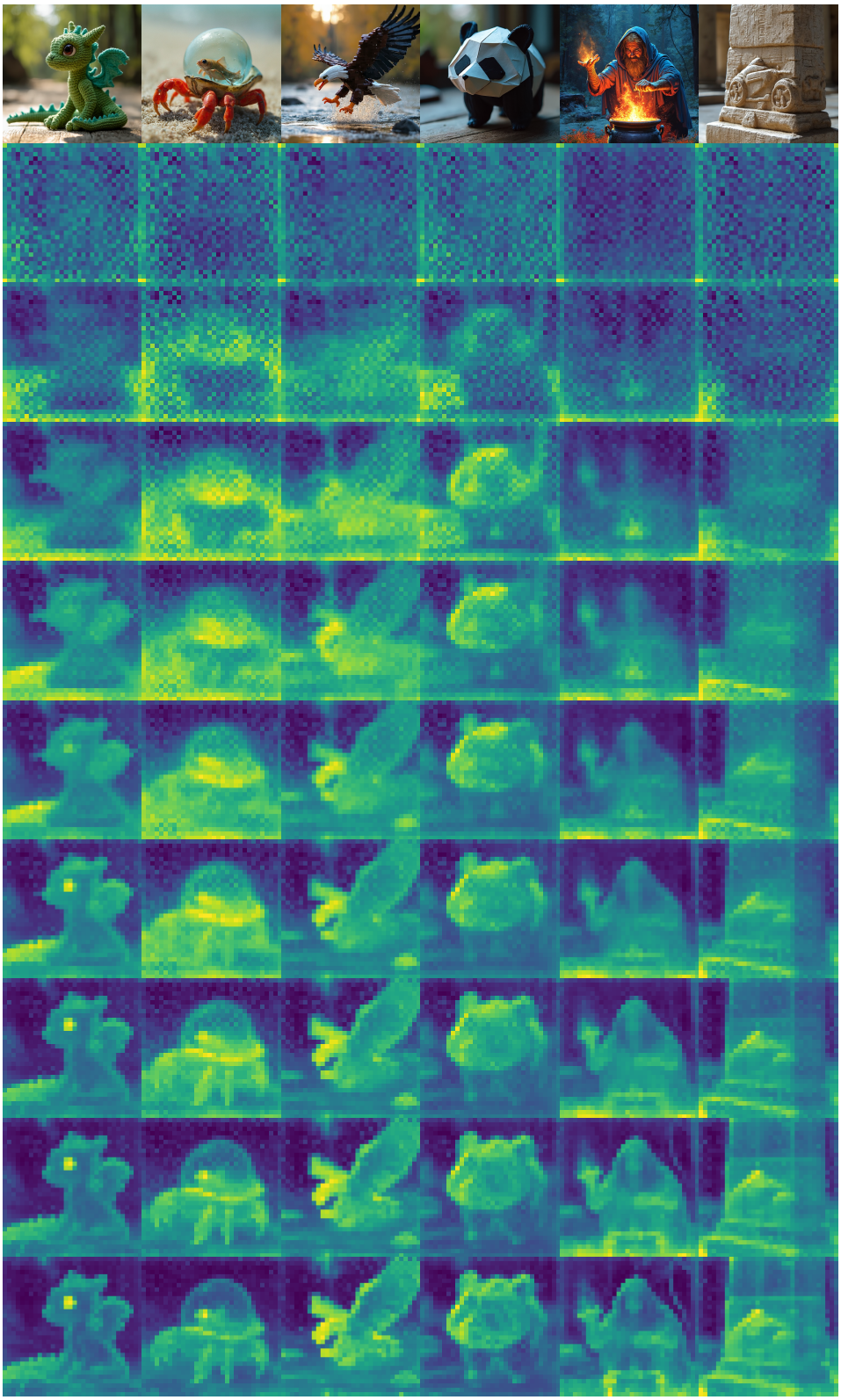

Figure 10: **Timestep Stability.** Top: generated images using FLUX-Dev with 50 timesteps. Each column depicts the TokenRank (incoming attention) for increasingly denoised images during the diffusion process. Rows correspond to decreasing timesteps.

References

Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Alec Helbling, Tuna Han Salih Meral, Ben Hoover, Pinar Yanardag, and Duen Horng Chau. Conceptattention: Diffusion transformers learn highly interpretable features. *arXiv preprint arXiv:2502.04320*, 2025.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Susung Hong, Gyuseong Lee, Wooseok Jang, and Seungryong Kim. Improving sample quality of diffusion models using self-attention guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7462–7471, 2023.

Jeremy Howard. Imagenette: A smaller subset of 10 easily classified classes from imagenet, 2019.

Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35: 25278–25294, 2022.