Supplementary Material for Diffusion Models as Masked Autoencoder

Appendix

In the Appendix, we first provide implementation details in Appendix A, and then provide more qualitative results in Appendix B.

A. Implementation Details

A.1. ImageNet Experiments

Noise schedule with ρ. We introduce a hyper-parameter ρ to control the noise level of training inputs. Specifically, we use ρ to exponentiate each variance β_t to β_t^ρ, enlarging these noise variance. Recall that the training samples can be reparameterized to $\boldsymbol{x}_t^m = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0^m + \sqrt{1-\bar{\alpha}_t}\epsilon$, where $\alpha_t = 1-\beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$. In Fig. 1, we plot how the values of the data coefficient $\bar{\alpha}_t$ progress with different ρ. $\rho = 1.0$ represents the default linear schedule introduced in DDPM [9], where the forward process variances β_t increase linearly from 10^{-4} to 0.02. With $\rho = 0.8$ and $\rho = 0.6$, the data coefficients $\bar{\alpha}_t$ are lower at each timestep t, and the amount of noise is therefore amplified.

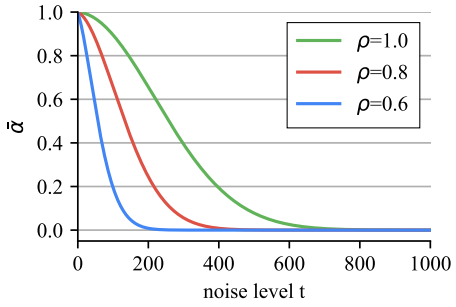

Figure 1: $\bar{\alpha}_t$ throughout the diffusion process with different ρ.

Architecture. We use the standard ViT architecture [5] in base, large and huge sizes for the encoder. The encoder is followed by Layer Normalization [1]. There is a linear projection layer after the layer normalization to match the dimension of the encoder to that of the decoder. We add sinusoidal positional embeddings to both the encoder and decoder inputs for pre-training. We do not use either relative positional embedding [16] or layer scale [2]. The encoder and the decoder use two different linear projections to handle the clean and the noised (masked) inputs, respectively.

config	ImageNet	Kinetics
optimizer	AdamW [14]	
optimizer momentum	β_1, β_2=0.9, 0.95	
weight decay	0.05	
learning rate schedule	cosine decay [13]	
warmup epochs [7]	40	
augmentation	hflip, RandomResizedCrop	
drop path [11]	0.0	
base lr	1.5e-4	8.0e-4
batch size	4096	512
epochs	1600	400
gradient clipping	-	0.02
repeated aug. [10]	-	4

(a) Pre-training setting.

	ImageNet			Kinetics
config	ViT-B	ViT-L	ViT-H	ViT-L
optimizer	AdamW [14]			
optimizer momentum	β_1, β_2=0.9, 0.999			
weight decay	0.05			
learning rate schedule	cosine decay [13]			
warmup epochs [7]	5			
augmentation	`RandAug` (9, 0.5) [3]			
mixup [22]	0.8			
cutmix [21]	1.0			
label smoothing [18]	0.1			
end lr	1.0e-6			
batch size	1024			128
base lr	5.0e-4	1.0e-3	1.0e-3	3.2e-3
layer decay [2]	0.65	0.7	0.75	0.75
base lr (w/ CLIP)	2.0e-4			8.0e-4
layer decay (w/ CLIP)	0.65	0.75	0.8	0.8
training epochs	100	50	50	50
drop path [11]	0.1	0.1	0.3	0.2
drop out [17]	-			0.5
repeated aug. [10]	-			2

(b) Fine-tuning setting.

Table 1: **Configurations on IN-1K and Kinetics-400.** In terms of learning rate (*lr*), we use the linear scaling rule introduced in [7]: *lr* = *base_lr*×*batch_size* / 256. When using repeated augmentation, the number of epochs and batch size count the original samples without repeating.

config	from-scratch	fine-tuning
optimizer	AdamW [14]	
optimizer momentum	β_1, β_2=0.9, 0.999	
weight decay	0.02	
learning rate schedule	cosine decay [13]	
warmup epochs [7]	10	
augmentation	`RandAug`(9, 0.5)	
mixup [22]	0.8	
cutmix [21]	1.0	
label smoothing [18]	0.1	
batch size	512	
epochs	200	
base lr	1.0e-3	
layer decay [2]	-	0.8
drop path [11]	0.1	0.2

Table 2: **Configurations of fine-tuning ADM [4] on IN-1K.** In terms of learning rate (*lr*), we use the linear scaling rule introduced in [7]: *lr* = *base_lr*×*batch_size* / 256. For fine-tuning, we use ADM's unconditional 256^2 model trained on IN-1K.

During fine-tuning, we extract features from the encoder. We use global average pooling to gather the patch features, followed by a layer normalization and a linear classification head. Both layer normalization and the linear head are randomly initialized. Particularly, the linear head is initialized with a very small standard deviation 2^{-5}, which enhances stability of fine-tuning.

Training recipes. The default settings for pre-training and fine-tuning are in Table 1. We use a different base learning rate and layer decay when fine-tuning CLIP-aided models.

A.2. Kinetics Experiments

Architecture. Given a video clip, we first divide it into non-overlapping patches in spacetime. Positional embeddings are added to the embedded patches. The spacetime patch size is $2 \times 16 \times 16$ for ViT-L/16 and $2 \times 14 \times 14$ for ViT-L/14. The target of our DiffMAE is a single time slice of the patch (16×16 or 14×14), and so are the corresponding noisy inputs to the decoder [6]. Similar to the image setting, the encoder and the decoder use two different linear projections to handle the clean and the noisy (masked) inputs, respectively. We use 90% random masking sampling on the spacetime patches [6].

We extract features from the encoder outputs for fine-tuning. We use global average pooling to gather the patch features, followed by a layer normalization and a linear head. The linear head is initialized with a very small standard deviation 2^{-5}, the same as the image setting. To further enhance the results, we fine-tune the 16×224^2 Kinetics-400 model to a longer duration 32 and a larger resolution 280^2 for a short schedule of 30 epochs without repeated augmentation.

Training recipes. The default settings for pre-training and fine-tuning are in Table 1. Note that many hyper-parameters are shared by the image and the video models, showing that DiffMAE is general across different domains. We search for the best base learning rate and layer decay when fine-tuning CLIP-aided models.

A.3. Fine-Tuning ADM

We fine-tune the pre-trained ADM [4] model to evaluate the recognition ability of this well-designed diffusion model. Specifically, we take its IN-1K unconditional 256^2 version and fine-tune the model at resolution 224^2 on IN-1K classification for a fair comparison to other methods.

The ADM model uses a U-Net [15] architecture for dense prediction. It consists of ResNet [8] blocks and self-attention layers [20]. We fine-tune the input blocks and the middle block, which are followed by a global average pooling, a layer normalization, and a linear classification head that projects the global averaged feature to classification logits. The layer normalization and the linear head are randomly initialized. Regarding the timestep input specifying the noise level for diffusion generation, we simply fix the timestep to 999 for classification fine-tuning, while other numbers that are inside the range of the noise schedule, *i.e.*, from 0 to 999, give similar training curves and results. We also train the same model from scratch as the baseline to show the effectiveness of diffusion generative pre-training.

Training recipes. We include the training recipes of fine-turning and from-scratch training of ADM in Table 2. We carefully tune the optimization hyper-parameters of both the fine-tuning and the from-scratch training. The recipes are based on sophisticated modern training techniques [19, 12], and we tune base learning rate, layer-wise decay [2], and drop path rate for each case.

B. Additional Qualitative Results

We provide more qualitative results of image generation using ImageNet-1K validation images. Figs. 2 and 3 are samples with 75% random masking. Figs. 4 and 5 are samples with the center block masking.

In Fig. 6, we provide visualizations on DiffMAE for video generation on Kinetics-400 validation videos. For a $16 \times 224 \times 224$ video clip, we visualize the generated frames at stride two on the temporal dimension, which makes eight frames for each sample.

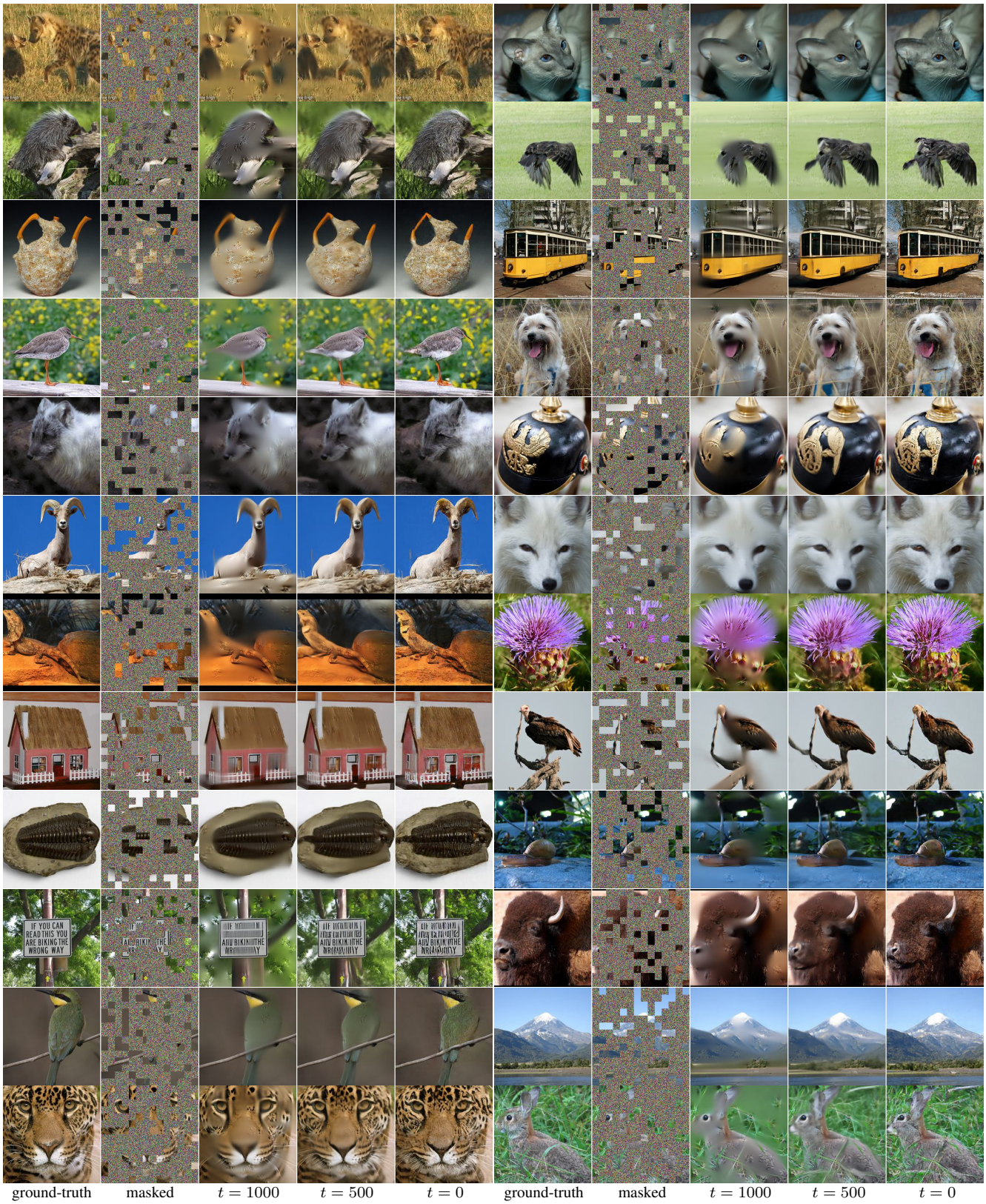

Figure 2: **Visualizations of DiffMAE generation with 75% random masking.** The images are from IN-1K validation set with size 224^2. We show the reverse diffusion at $t = 1000$, 500, and 0. $t = 0$ is the final output. The model is ViT-L. Best viewed in color with zoom.

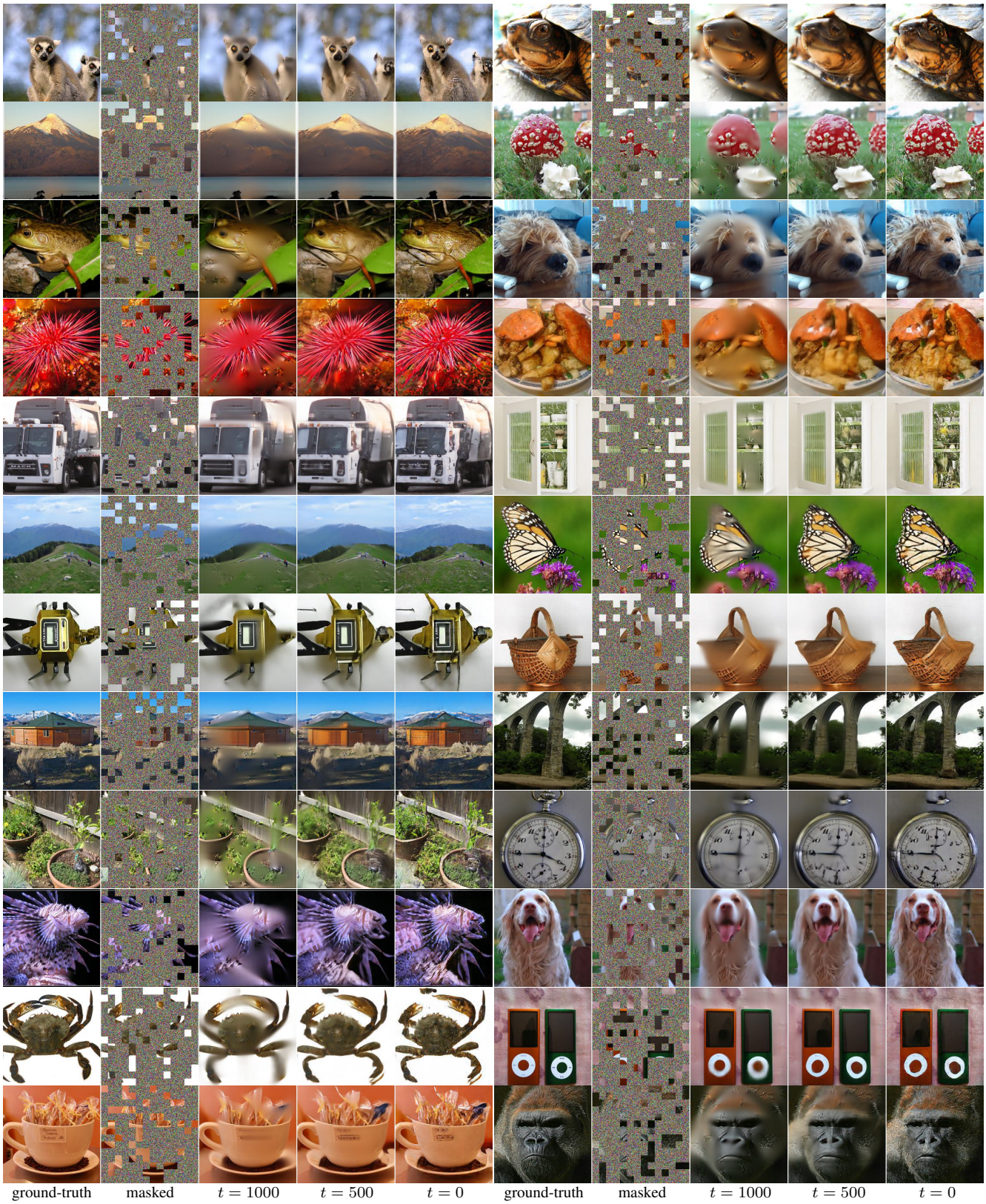

Figure 3: **Visualizations of DiffMAE generation with 75% random masking.** The images are from IN-1K validation set with size 224^2. We show the reverse diffusion at $t = 1000$, 500, and 0. $t = 0$ is the final output. The model is ViT-L. Best viewed in color with zoom.

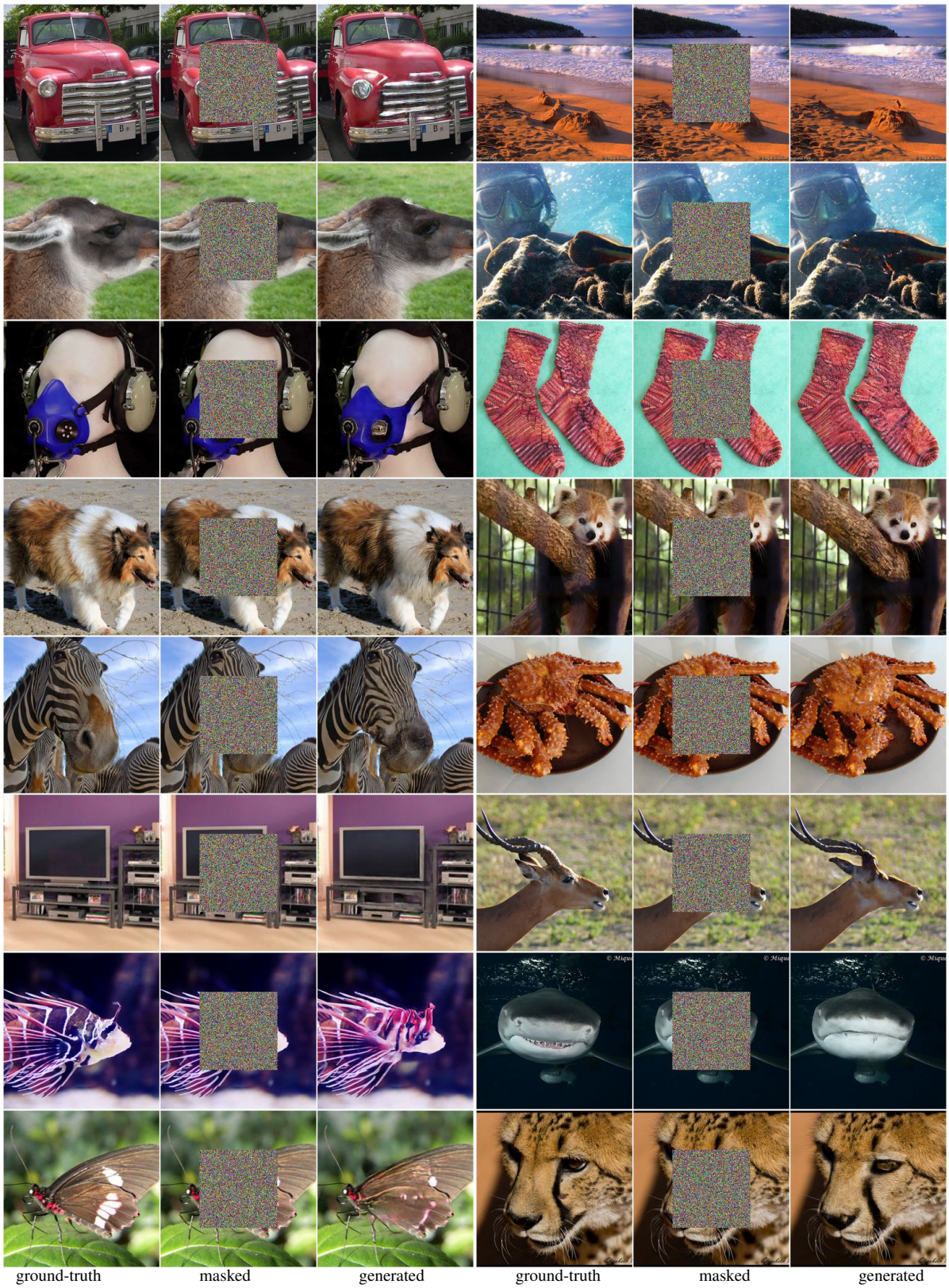

Figure 4: **Visualizations of DiffMAE generation with center masking.** The images are from IN-1K validation set. The input images are of size 256^2, with the center 128^2 block masked. The model is ViT-L. Best viewed in color with zoom.

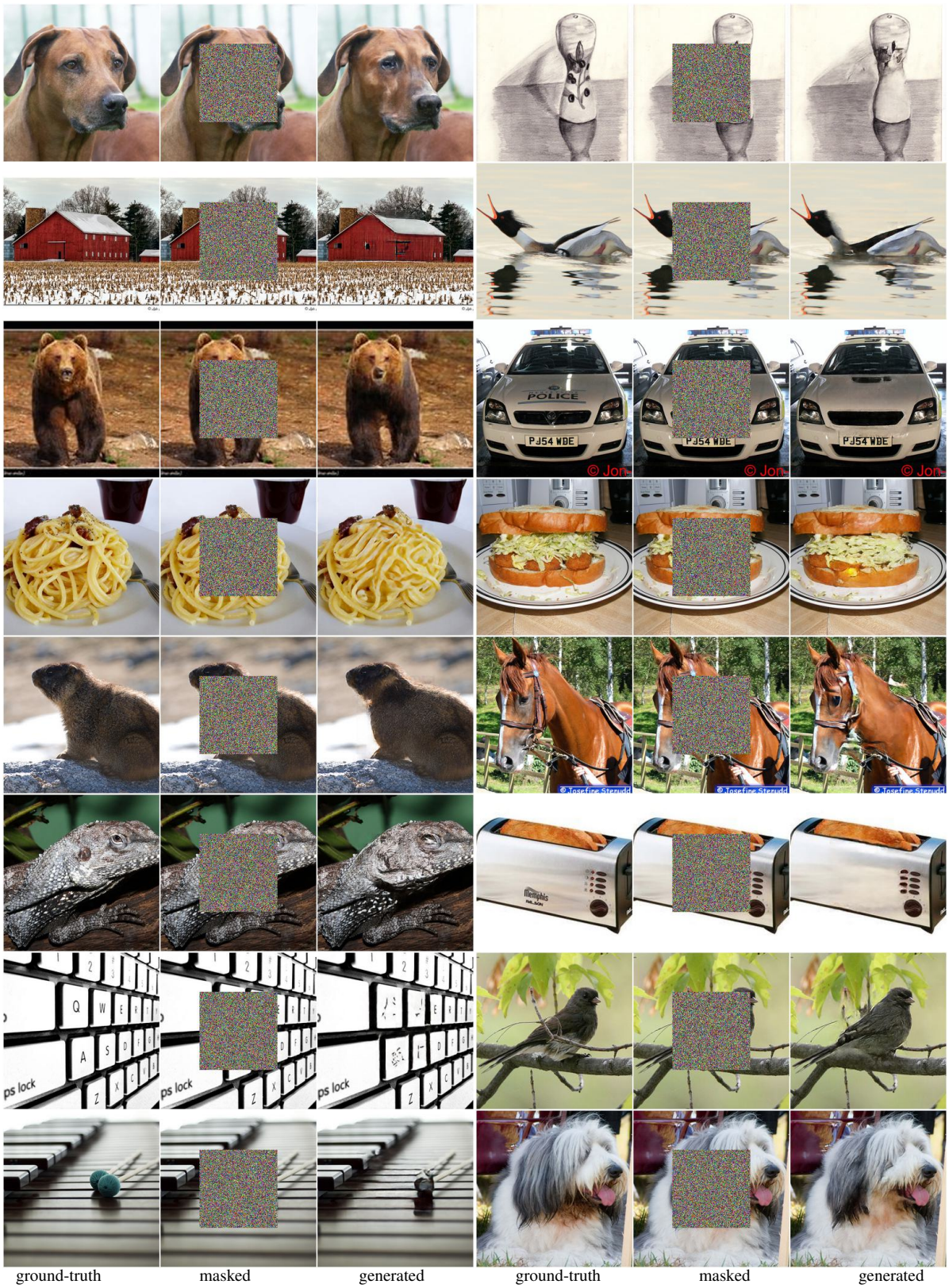

Figure 5: **Visualizations of DiffMAE generation with center masking.** The images are from IN-1K validation set. The input images are of size 256^2, with the center 128^2 block masked. The model is ViT-L. Best viewed in color with zoom.

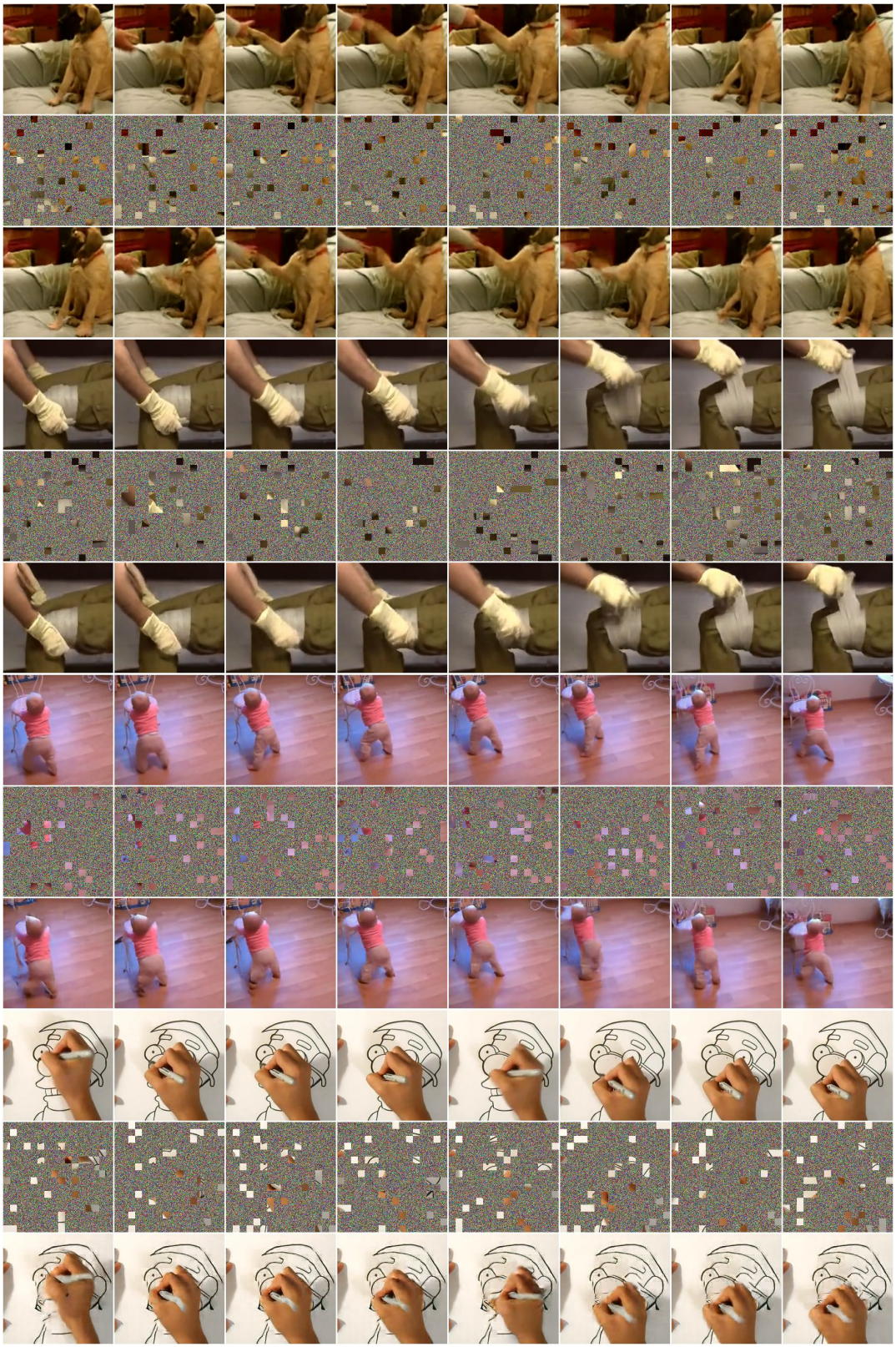

Figure 6: **Visualizations of DiffMAE generation with video.** The videos are from Kinetics-400 validation set with random masking ratio 90%. We show the original video (*top*), masked video (*middle*), and DiffMAE output (*bottom*) for each sample. The model is ViT-L/14.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *NeurIPS*, 2016. 1

[2] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020. 1, 2

[3] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. RandAugment: Practical automated data augmentation with a reduced search space. In *CVPR*, 2020. 1

[4] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 2

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1

[6] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. In *NeurIPS*, 2022. 2

[7] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 1, 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1

[10] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *CVPR*, 2020. 1

[11] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. FractalNet: Ultra-deep neural networks without residuals. In *ICLR*, 2017. 1, 2

[12] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022. 2

[13] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 1, 2

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 1, 2

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2

[16] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL*, 2018. 1

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014. 1

[18] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 1, 2

[19] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 2

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[21] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 1, 2

[22] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 1, 2