

CAT: Content-Adaptive Image Tokenization

Supplementary Material

8. Prompt for LLM Scorer

Our caption complexity pipeline works as follows:

Step 1: Use a VLM to generate image description, with the following prompts:

- What's in the image? → Caption
- Are there text or numbers in the image? → Yes/No.
- Are there faces in the image? → Yes/No.

Step 2: Use the same VLM or a separate LLM to generate the complexity score with the prompt:

Given the description of a 512px image, determine its complexity based on the following factors:

1. Number of distinct objects
2. Color variance
3. Texture complexity
4. Foreground and background
5. Symmetry and repetition
6. Human perception factors, like the presence of human faces or text

You will be given the caption, whether there are text or numbers, and whether there are faces in the image. Assign a complexity score such that a higher number means the image is more complex. Note that text and facial details are intrinsically complex because they are crucial to human perception. Here are some examples for scoring:

- Score 1: A plane in a sky
- Score 2: A t-shirt with a emoji on it
- Score 3: A dog lying on the grass
- Score 4: A woman skiing in the snow
- Score 5: Two kids walking on the beach
- Score 6: A dinning table full of food
- Score 7: A close-up shot of a old man
- Score 8: Many people gathering in the stadium
- Score 9: Newspapers or graphs with text and numbers

Now determine the complexity for the caption:

[Insert caption here]

[Insert one of the following based on the Yes/No questions:

- There are text visible in the image. There are also facial details.
- There are text visible in the image, but there is no human face.
- There is no obvious text in the image, but there are facial details.
- There is no text or human face in the image.]

Respond with "Score: ? out of 9", where "?" is a number between 1 and 9. Then provide explanations.

Note these two steps can be combined into a single inference call.

9. Compression Ratio Distributions with Different LLMs

930

Scoring Model		8x	16x	32x	Avg Rate
ImageNet	InstructBLIP + Llama3.1-8B-Instruct	6%	49%	45%	17.43
	InstructBLIP + Qwen2.5-7B-Instruct	2%	70%	28%	17.35
	LLaVA1.5 7B	2%	65%	33%	17.75
CelebA	InstructBLIP + Llama3.1-8B-Instruct	17%	83%	0%	13.02
	InstructBLIP + Qwen2.5-7B-Instruct	15%	70%	15%	13.83
	LLaVA1.5 7B	5%	95%	0%	14.92
DTD	InstructBLIP + Llama3.1-8B-Instruct	0%	40%	60%	21.57
	InstructBLIP + Qwen2.5-7B-Instruct	0%	38%	62%	21.87
	LLaVA1.5 7B	0%	41%	59%	21.42
EuroSAT	InstructBLIP + Llama3.1-8B-Instruct	3%	30%	67%	20.87
	InstructBLIP + Qwen2.5-7B-Instruct	3%	25%	72%	21.57
	LLaVA1.5 7B	2%	22%	76%	22.85
SUN397	InstructBLIP + Llama3.1-8B-Instruct	6%	73%	21%	15.82
	InstructBLIP + Qwen2.5-7B-Instruct	2%	78%	20%	16.77
	LLaVA1.5 7B	3%	80%	17%	16.30

Table 9. To study whether our caption score is robust to LLM choice, we test multiple captioners and LLMs for scoring. The fact that these combinations generate similar score and compression ratio distributions show that our scoring method is robust.

10. Reconstruction Experiments

931

10.1. Architecture

932

We implement the nested VAE similar to the `AutoencoderKL` implementation in the `diffusers` library. The network configuration is:

933

934

- `sample_size`: 512
- `in_channels`: 3
- `out_channels`: 3
- `down_block_types`: `[DownEncoderBlock2D] × 6`
- `up_block_types`: `[UpDecoderBlock2D] × 6`
- `block_out_channels`: `[64, 128, 256, 256, 512, 512]`
- `layers_per_block`: 2
- `act_fn`: `silu`
- `latent_channels`: `4/8/16`
- `norm_num_groups`: 32
- `mid_block_attention_head_dim`: 1
- `num_layers`: 8

935

936

937

938

939

940

941

942

943

944

945

946

The model sizes for different latent channels are shown below. For the discriminator, we use the pretrained StyleGAN [64].

Nested VAE	$c = 4$	$c = 8$	$c = 16$
# Params (M)	187.45	187.50	187.61

947

10.2. Training

948

We use the following training configuration:

949

- GPU: 64 NVIDIA A100
- Per-GPU batch size: 8
- Global batch size: 512
- Training steps: 1,000,000
- Optimizer: AdamW
 - `lr`: 0.0001
 - `beta1`: 0.9

950

951

952

953

954

955

956

- beta2: 0.95
- weight_decay: 0.1
- epsilon: 1e-8
- gradient_clip: 5.0
- Scheduler: constant with 10,000 warmup steps
- Loss:
 - recon_loss_weight: 1.0
 - kl_loss_weight: 1e-6
 - perceptual_loss_weight: 1.0
 - moco_loss_weight: 0.2
 - gan_loss_weight: 0.5
 - gan_loss_starting_step: 50,000

The discriminator is trained with the standard GAN loss.

10.3. Baselines

We train fixed compression baselines using the same data, training configuration, and VAE backbone. For smaller compression ratios, e.g., fixed 8x, the last two downsampling blocks and first two upsampling blocks are not used.

For the adaptive JPEG baseline, we use `torchvision.io.encode.jpeg` to transform the images into JPEG file and compute the number of bytes as the complexity metric. Smaller files correspond to larger complexity. To provide a better understanding of this metric, we show in Figure 6 the distribution of JPEG sizes on the COCO 2014 test set, with relevant statistics included in the caption. Then, based on the JPEG sizes of all images in the Shutterstock training dataset, we set the thresholds (a, b) to (38761, 65837) to categorize the file sizes into three compression ratios. This set of thresholds ensure that the JPEG baseline has the same training compression ratio distribution as CAT.

For LDM VAEs, we follow the instructions in their original repository to use the model checkpoints. Note that LDM VAEs are trained on OpenImages dataset [65], which is different from our training data, so it is hard to fairly compare the reconstruction performance. Nonetheless, we present their rFIDs on the evaluation datasets in Table 10.

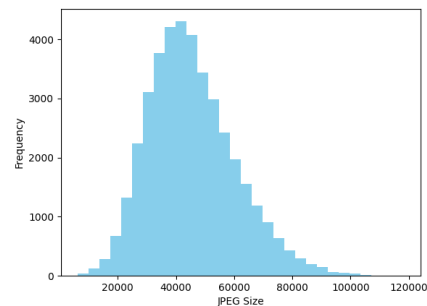


Figure 6. On COCO 2014 test set, the minimum JPEG size is 6128; maximum is 118428; mean is 45474.29; standard deviation is 15037.07.

	COCO	ImageNet	CelebA	ChartQA
CAT	0.65	0.46	1.97	5.27
LDM 8x	0.51	0.33	2.83	8.32
LDM 16x	0.53	0.37	3.07	8.49
LDM 32x	0.90	0.62	5.54	10.35

Table 10. rFIDs for CAT and LDM VAEs.

10.4. More Reconstruction Visualization

See Figure 7 in the next page.

10.5. Fixing Token Count for CAT

We also evaluate the reconstruction performance under fixed compression ratio (token count) for different datasets. Table 11 compares the reconstruction FID for CAT with caption-guided compression ratio vs. fixed 16x compression ratio. We see that adaptive compression based on image complexity outperforms uniform compression using the same architecture in most cases, possibly because error reduction on complex images outweighs the slight error increase on simpler ones.

	CAT rFID	Adaptive	Equalized 16x
COCO		0.65	0.67
ImageNet		0.46	0.40
CelebA		1.97	2.47
ChartQA		5.27	7.27

Table 11. Equalizing test-time token counts.

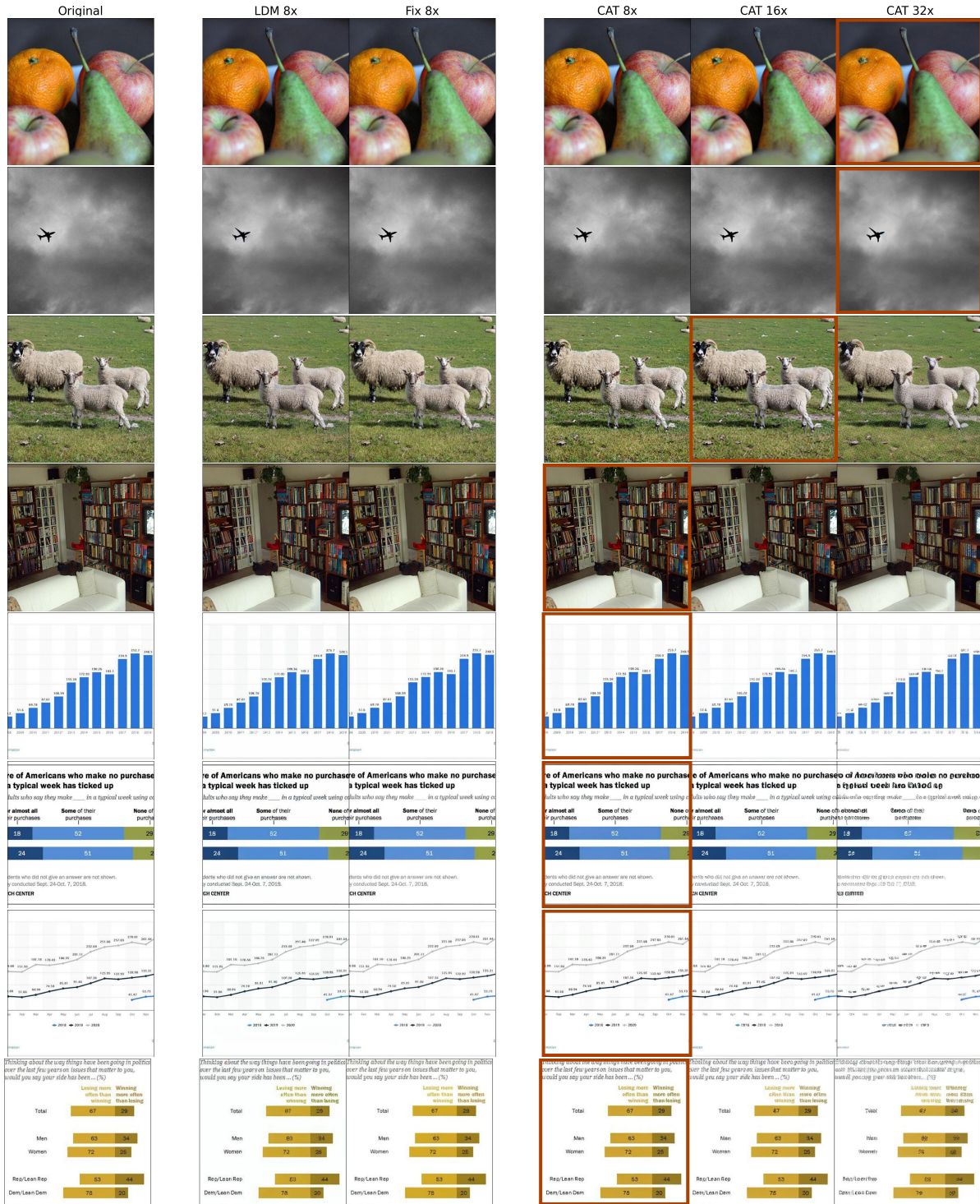


Figure 7. More CAT reconstruction examples. We highlight the compression ratio selected by our caption complexity in red.

10.6. Full Results for Table 3 and Table 4

To complement Table 3, we include the training distribution for different scoring thresholds in Table 12. To complement Table 4, we include the results of fixed-ratio baseline in Table 13.

(a, b)	Training Distribution				Reconstruction FID ↓			
	8x	16x	32x	Average	COCO	ImageNet	CelebA	ChartQA
(4, 7)	10%	48%	42%	16.0x	0.65	0.46	1.97	5.27
(2, 8)	0.5%	89.5%	10%	16.5x	0.67	0.43	2.58	7.70

Table 12. **Compression ratio distribution affects learning outcomes.** Both settings have an average compression of $\sim 16x$, but (4, 7) leads to better distribution diversity and empirical results.

rFID↓	c	COCO	ImageNet	CelebA	ChartQA
Fixed 16x	4	1.25	1.32	5.89	9.45
	8	1.10	0.61	4.99	8.19
	16	0.66	0.38	2.25	8.67
CAT	4	1.66	1.10	5.83	9.13
	8	1.03	0.60	4.54	7.95
	16	0.65	0.46	1.97	5.27

Table 13. **Larger latent channel c generally improves rFID.**

11. Classification Experiments

We selected the diverse datasets because they represent out-of-domain distributions where zero-shot models perform poorly [66–68]. Similar to OpenClip [55], we initialize a simple linear classification head that maps from the tokenizer’s maximum latent dimension (under 8x compression) to the number of labels for each task. When the image is compressed with 16x or 32x ratio, we zero-pad the latent features to make the length match with the classification head. We evaluate two settings: (1) linear probing, where we keep the image encoder frozen and only fine-tune the classification head; (2) full fine-tuning, where we update both the encoder and the classification head. We train both settings for 20 epochs with a batch size of 64, learning rate $1e-4$ and a cosine annealing learning rate schedule with 2 warm-up epochs. We use the AdamW optimizer with weight decay 0.1.

12. Generation Experiments

12.1. Architecture

We use DiT-XL architecture with a patchify downsampler and patch size of 2. The model size depends on the latent channel, but is generally around 431M parameters. The model TFLOPs is 22.0.

12.2. Training & Inference

We use LLaVA1.5 7B to generate the complexity score for ImageNet training images. For 10 % of the time, we remove the image class label from the input and train unconditional image generation. The training configuration for DiT is:

- GPU: 16 NVIDIA H100
- Per-GPU token batch size: 4096×4 (equivalent to 64 images for 16x compression ratio)
- Global token batch size: 4096×64
- Training steps: 400,000
- Optimizer: AdamW
 - lr: 0.0001
 - beta1: 0.9
 - beta2: 0.95
 - weight_decay: 0.1
 - epsilon: $1e-8$
 - gradient_clip: 1.0
- Scheduler: Cosine
 - warmup: 4000
 - cosine_theta: 1.0
 - cycle_length: 1.0
 - lr_min_ratio: 0.05

At test time, we use “*this is an image of [label]*” as a standardized prompt and manually provide answers to the queries to enable automated evaluation. Then, we use Llama3.1 to obtain the complexity score. DDPM scheduler (diffusers implementation) configuration is:

- num_train_timesteps: 1000
- beta_start: 0.0001
- beta_end: 0.02
- beta_schedule: squaredcos_cap_v2
- prediction_type: epsilon
- timestep_spacing: leading
- num_inference_steps: 250

All FID-50K and images generated in this paper are using cfg=1.5.

12.3. Baselines

To ensure we train the baseline with the same compute FLOPs, we fix the token batch size and number of training steps for all settings. For pretrained LDM VAE, we use the scaling factor specified in the model configuration to ensure the input scale and noise scale are similar. For CAT, we use a scaling factor of 1.

12.4. More Visualization

See Figure 8 in the end.

12.5. Full Results for Table 8

To complement Table 8, we include the results of fixed-ratio baseline in Table 14.

	c	FID↓	sFID↓	IS↑	Precision↑	Recall↑
Fixed 16x	4	5.11	10.84	158.80	0.75	0.49
	8	4.96	10.39	221.85	0.76	0.51
	16	4.78	11.81	187.47	0.72	0.49
CAT	4	5.12	11.12	152.39	0.72	0.48
	8	4.38	10.31	181.03	0.76	0.48
	16	4.56	10.55	191.09	0.75	0.49

Table 14. **Larger channel c is not always better for generation.** Contrary to Table 4, we find that increasing channel dimension does not always result in generation gains.

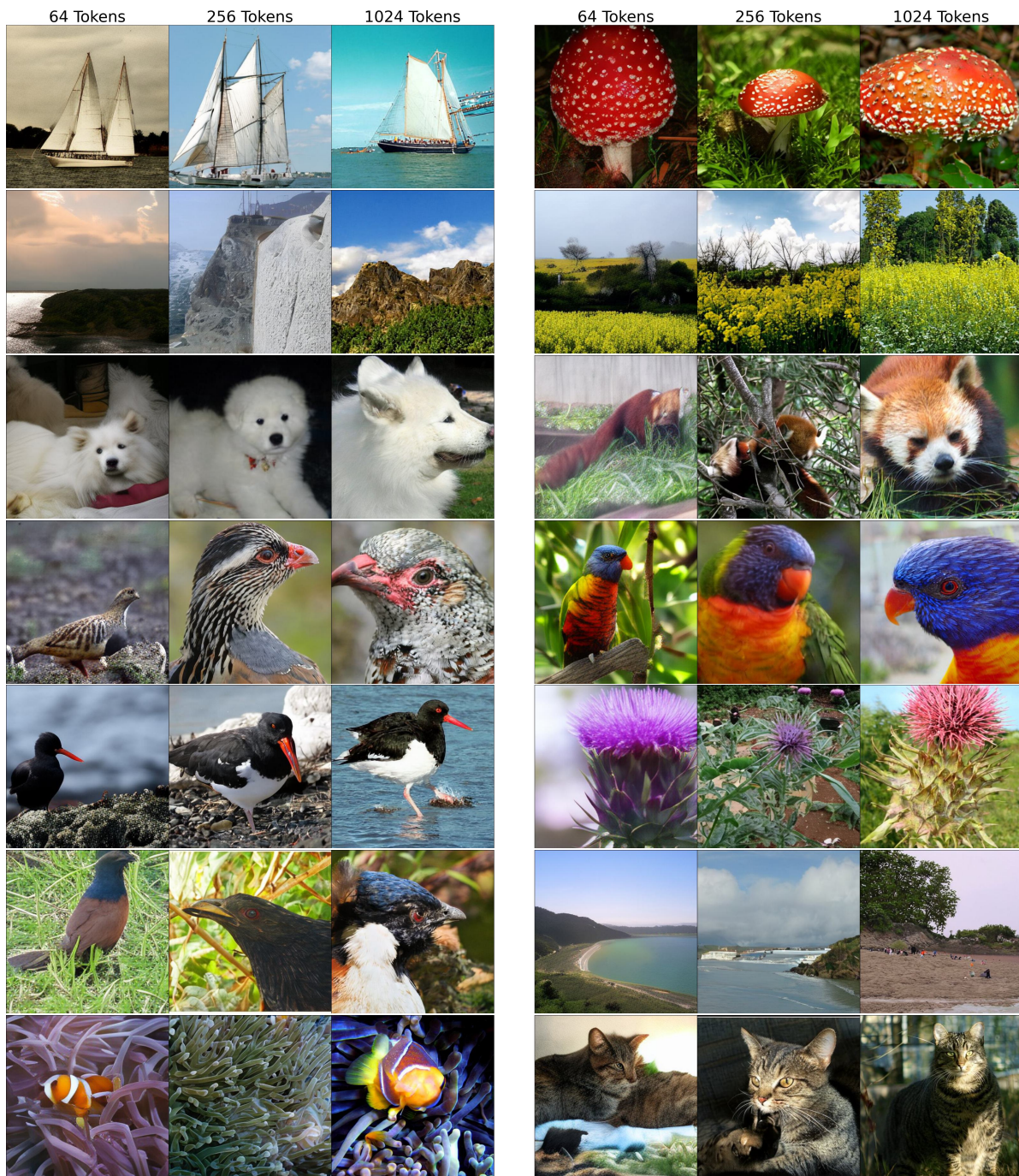


Figure 8. More DiT-CAT generation examples. Increasing token count (left→right) generally leads to better image quality and higher complexity.