Research Article

# CAT: Content-Adaptive Image Tokenization

**Junhong Shen[1,2], Kushal Tirumala[2], Michihiro Yasunaga[2], Ishan Misra[2], Luke Zettlemoyer[2], Lili Yu[2], Chunting Zhou[1,2]**

1. Carnegie Mellon University, United States; 2. Meta

**Most existing image tokenizers encode images into a fixed number of tokens or patches, overlooking the inherent variability in image complexity. To address this, we introduce Content-Adaptive Tokenizer (CAT), which dynamically adjusts representation capacity based on the image content and encodes simpler images into fewer tokens. We design a caption-based evaluation system that leverages large language models (LLMs) to predict content complexity and determine the optimal compression ratio for a given image, taking into account factors critical to human perception. Trained on images with diverse compression ratios, CAT demonstrates robust performance in image reconstruction. We also utilize its variable-length latent representations to train Diffusion Transformers (DiTs) for ImageNet generation. By optimizing token allocation, CAT improves the FID score over fixed-ratio baselines trained with the same flops and boosts the inference throughput by 18.5%.**

## 1. Introduction

Image tokenizers compress high-resolution images into low-dimensional latent features to generate compact and meaningful representations[1][2][3][4][5][6][7].

Despite their effectiveness, most existing tokenizers use a fixed compression ratio, encoding images into feature vectors of exactly the same dimensions, regardless of their content. However, different images contain varying levels of detail, which suggests that a one-size-fits-all approach to compression may not be optimal. Indeed, traditional codecs like JPEG[8] typically produce different file sizes based on the spatial frequency of the images, even when set to the same quality level.

Moreover, using the same representation capacity for all images can compromise both the quality and the efficiency of the tokenizer. Over-compressing complex images may result in the loss of important visual details, while under-compressing simple images can lead to inefficiencies in training downstream models, as additional compute is wasted on processing redundant information. Several recent studies have proposed to adjust the number of tokens used at inference time based on the compute budget[9]. However, these methods overlook the intrinsic complexity of images when training the tokenizers. Besides, they do not account for the downstream use cases in the tokenizer design. For example, image tokenizers are often used to produce inputs for latent diffusion models (LDMs)[10] and perform text-to-image generation, where only the user's text prompt is available at inference time. Nonetheless, existing work all require image inputs to perform adaptive tokenization.

 doi.org/10.32388/WCBNQ2 

In this work, we present **C**ontent-**A**daptive **T**okenizer (CAT), which dynamically allocates representation capacity based on image complexity to improve both compression quality and computational efficiency. To achieve this, we propose a text-based image complexity evaluation system that leverages large language models (LLMs) to predict the optimal compression ratio given the image description. Then, we train a single unified variational autoencoder to generate latent features of variable shapes (Figure 1).

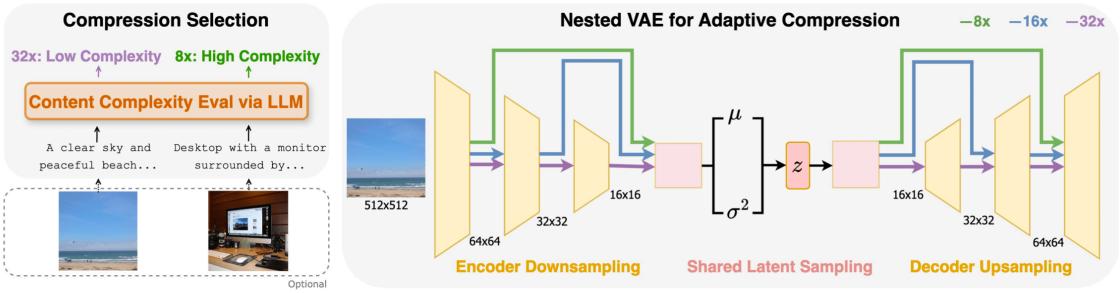



**Figure 1. Content-Adaptive Tokenization.** CAT uses an LLM to evaluate the content complexity and determine the optimal compression ratio based on the image's text description. The image is processed by a nested VAE architecture that dynamically routes the input according to the selected compression ratio. The resulting latent representations thus have varying spatial dimensions. Images shown in the figure are taken from COCO 2014[11].

Our complexity evaluation system is designed to accurately reflect the content complexity, while being compatible with diverse downstream tasks, including text-to-image generation with LDMs. Specifically, we use the text description of an image to prompt an LLM and generate a complexity score. The text description includes the image caption and answers to a set of perception-focused queries, such as "*are there human faces/text*", which are designed to help identify elements sensitive to human perceptions. Based on the complexity score, the image is classified into one of 8x, 16x, or 32x compression. A higher ratio means that we can compress a simpler image more aggressively.

Then, we develop a nested variational autoencoder (VAE) architecture that can perform multiple levels of compression within a single model. This is achieved by routing the intermediate outputs from the encoder downsampling blocks to a shared middle block to generate variable-dimensional Gaussian distributions. From these, we can sample latent features of different spatial resolutions.

We train the nested VAE on images with diverse complexity, specifically using the compression ratios produced by our LLM evaluator. We analyze its reconstruction performance on a variety of datasets, including natural scenes (COCO[11], ImageNet[12]), human faces (CelebA[13]), and text-heavy images (ChartQA[14]). On complex images featuring human faces or text, CAT substantially improves the reconstruction quality, reducing the rFID by 12% on CelebA and 39% on ChartQA relative to fixed-ratio baselines. On natural images like ImageNet, CAT maintains the reconstruction quality while using 16% fewer tokens.

We further validate the effectiveness of CAT in image generation by training Latent Diffusion Transformers (DiTs)[15]. Due to its content-adaptive representation, CAT more effectively captures both high-level and low-level information

in image datasets compared to fixed-ratio baselines, hence accelerating the diffusion model learning process. We demonstrate that CAT achieves an FID of 4.56 on class-conditional ImageNet generation, outperforming all fixed-ratio baselines trained with the same flops. Additionally, CAT improves inference throughput by 18.5%. Beyond the quality and speed improvements, we show that CAT enables controllable generation at various complexity levels, allowing users to specify the number of tokens to represent the images based on practical needs.

To summarize, we introduce CAT, an image tokenizer that enables: (1) **Adaptive Compression**: It compresses images into variable-length latent representations based on content complexity, leveraging an LLM evaluator and a nested VAE model; (2) **Faster Generative Learning**: It boosts the efficiency of learning latent generative models by effectively representing both high-level and low-level image information; (3) **Controllable Generation**: It enables generation at various complexity levels based on user specifications. Overall, CAT represents a crucial step towards efficient and effective image modeling, with promising potential for extension to other visual modalities, such as video.

## 2. Related Work

**Visual Tokenization.** Existing visual tokenizers use diverse architectures and encoding schemes. Continuous tokenizers map images into a continuous latent space, often utilizing the VAE architecture[2] to generate Gaussian distributions for sampling latent features. Discrete tokenizers like VQ-VAE[16] and FSQ[7] use quantization techniques to convert latent representations into discrete tokens. While our experiments focus on the continuous latent space, the proposed adaptive image encoding method is compatible with both continuous and discrete tokenizers.

**Adaptive Compression.** Traditional codecs, such as JPEG[8] for images and H.264[17] for videos, apply varying levels of compression based on the input media and the desired quality, resulting in files of different sizes. In the field of deep learning, a line of work studies adaptive patching for Vision Transformers[18] via patch dropout or merging[19][20][21][22]. [23] use mixed-resolution patches to obtain variable-length token sequences. However, these methods are tailored for visual understanding tasks and cannot be used to generate images.

Developing adaptive tokenizers capable of image generation remains underexplored. ElasticTok[9], a concurrent work to ours, employs a random masking strategy to drop the tail tokens of an image when training the tokenizer. This allows for using an arbitrary number of tokens to represent an image at inference time. However, by assigning random token lengths to training images, ElasticTok overlooks the inherent complexity of the visual content. Another concurrent work, ALIT[24], iteratively distills 2D image tokens into 1D latent tokens to reduce the token count. Unlike ALIT, CAT compresses images based on complexity predicted from captions. Our approach enables adaptive allocation of representation capacity using only text descriptions, without directly observing the images.

**Multi-Scale Feature Extraction.** A final line of relevant research involves designing neural networks that effectively extract multi-scale features. CAT builds upon VAE and adds skip connections inspired by U-Net[25] and Matryoshka representation learning[26][27][28]. In parallel, transformer-based multi-scale feature extractors have also been explored in[29][30][31][32][33][34]. We opt for a convolutional tokenizer architecture due to its strong empirical performance.

## 3. Method

In this section, we introduce CAT for adaptive image tokenization. We begin by discussing how to measure and predict image complexity. Then, we introduce the CAT architecture for performing compression at different ratios.

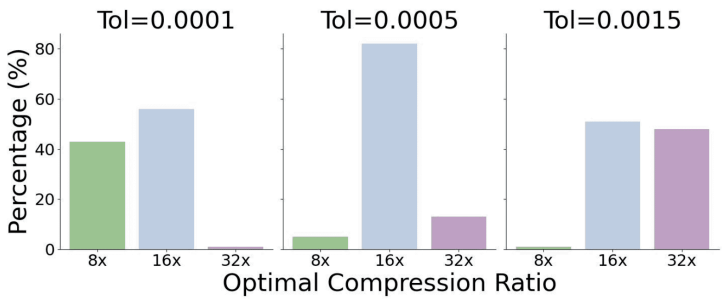
### *3.1. Proof of Concept*

#### *3.1.1. How Much Can We Actually Compress?*

A key question in this work is to determine how much an image can be compressed without significant loss of quality. To explore this, we analyze the reconstruction performance of existing tokenizers with various compression ratios. We take the open-source image tokenizers from LDM[1][10] with 8x, 16x and 32x compression ratios and compute their reconstruction mean squared error (MSE) on 41K $512 \times 512$ images from the COCO 2014 test set[11]. Our analysis reveals that for 28.3% of the images, 32x compression results in less than a 0.001 MSE increase compared to 8x compression, while reducing the token count by a factor of 16. We also compute the best MSE among all compression ratios for each image and determine the maximum acceptable compression ratio under a tolerance $\tau$. That is, denote the compression ratio as $f$, we want to find

$$\text{argmax}_{f \in \{8,16,32\}} (MSE_f - \min_{f' \in \{8,16,32\}} MSE_{f'}) < \tau. \tag{1}$$

Figure 2 shows that 56% of the images can be compressed at least to 16x with negligible (0.0001) increase in MSE[2]. A large portion of natural images can be compressed more aggressively while maintaining the same quality level as a fixed 8x tokenizer.




| Metric | Pearson $r$ |
|---|---|
| JPEG | 0.31 |
| MSE | 0.36 |
| LPIPS | 0.23 |
| Caption | **0.55** |

**Figure 2. Left:** Maximum acceptable compression ratios for COCO images under different error tolerance. We can compress most images more aggressively without compromising reconstruction quality. **Right:** Pearson correlation between various metrics and max acceptable compression ratio with tolerance 0.0015.

On the other hand, our visual inspection reveals that images with fine-grained elements like text have much better reconstruction quality at 8x compression compared to 32x (see for example row 3 and 4 in Figure 3). This suggests that more tokens are required to accurately reconstruct low-level details in such images. The above results provide strong

motivation for developing a tokenizer with adaptive compression ratios. Accordingly, we set the target ratios for CAT to be 8, 16, and 32.

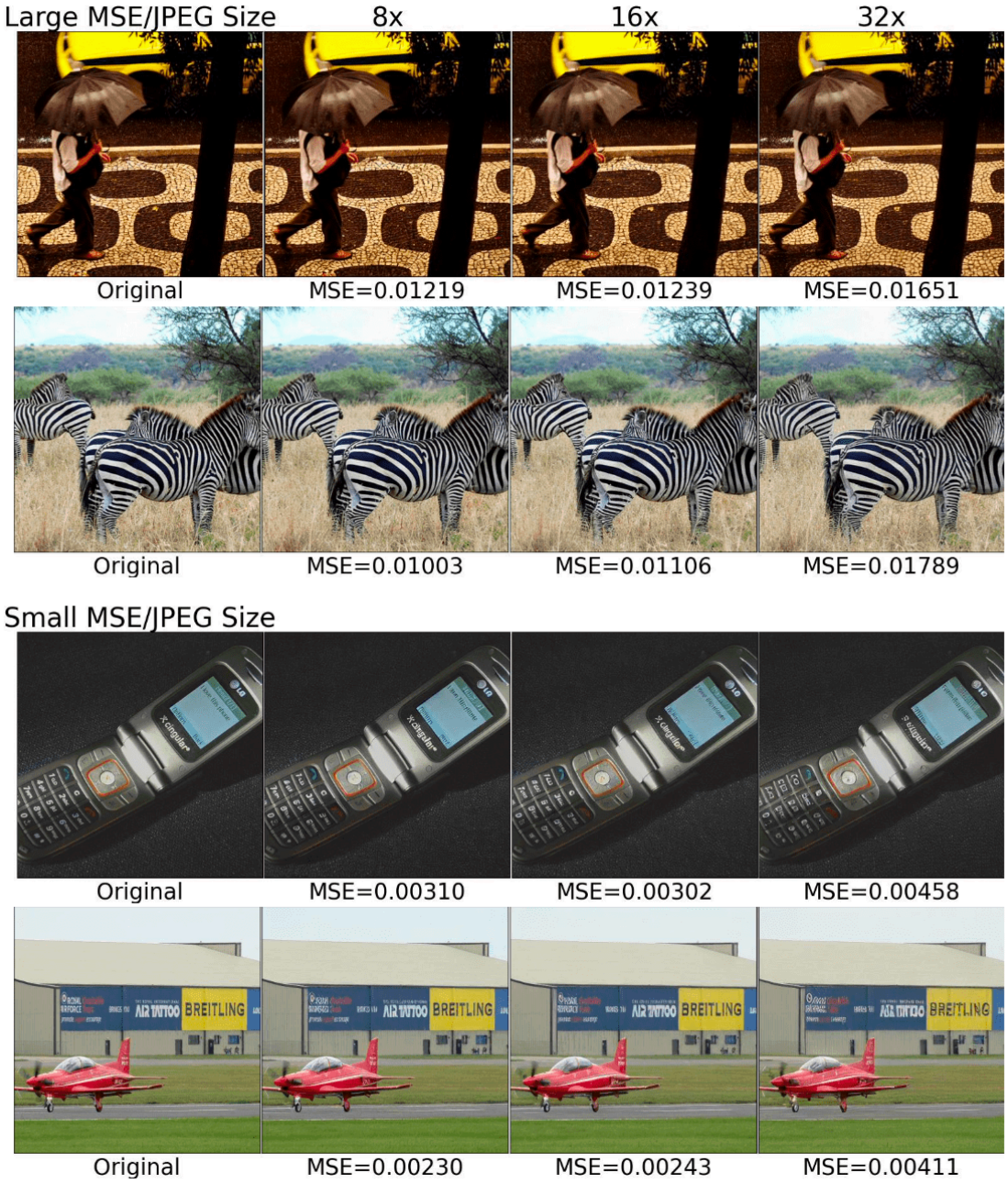



**Figure 3. Existing metrics can misjudge image complexity.** Metrics like JPEG size, MSE, and LPIPS consider images with high contrast and repetitive patterns as complex but underestimate the complexity of text-heavy images that are more challenging for human perception (note the distortion in the bottom two rows). Images shown in the figure are taken from COCO 2014[11].


doi.org/10.32388/WCBNQ2

### 3.1.2. Limitations of Existing Complexity Metrics

Next, we want to identify a metric for predicting the optimal compression ratio given an image. We explore some existing options, categorized into two groups: (1) metrics produced by traditional codecs, i.e., the JPEG file size; (2) metrics based on pretrained VAEs[3], including reconstruction MSE and LPIPS[35], which measures the L2 distance of VGG Net[36] activations between the original and reconstructed images. We first compute these metrics on the COCO dataset and analyze their correlation with the maximum acceptable compression ratio under $\tau = 0.0015$. However, Table 2 shows that the Pearson correlations are relatively low.

After that, we manually inspect the images with large JPEG sizes and MSEs. We note that images featuring repetitive patterns, such as grass, forests, and animals like giraffes and zebras consistently show high complexity metrics. Indeed, JPEG compression can be inefficient for images with sharp edges and high contrast. A single-pixel shift in a zebra image can toggle pixel values between black and white, significantly increasing the pixel-wise MSE. However, as the top rows in Figure 3 show, large MSEs do not always notably affect visual quality. For example, we can easily recognize the zebra and may not perceive the differences resulting from various compression ratios.

On the contrary, we find that many images with low considered metrics in fact have low fidelity. These images often contain visual elements like human faces or text, where even slight distortions can degrade visual quality (Figure 3, bottom rows). Despite this, these images have low MSEs possibly because the critical elements occupy only small portions of the images. Thus, metrics like JPEG size, MSE, and LPIPS might not effectively capture details crucial to human perception. Contrary to the predicted complexity, we actually want to use a small compression ratio for text-heavy images, and a large compression ratio for the zebra.

Lastly, the considered metrics all require images as input and cannot be used to measure complexity for text-to-image generation tasks, where no image is available at inference time. Given all these limitations of existing metrics, we seek a new method that is independent of pixel data and aligns with human perception to predict image complexity.

## 3.2. Complexity Evaluation via Captions and LLMs

Image generation typically involves users providing a prompt that describes the desired image content. To better align with such real-world use cases, we leverage the text description of an image to measure its content complexity.

We propose a three-stage complexity evaluation system: (1) obtaining the text description, (2) prompting an LLM to output a complexity score, and (3) classifying the score into a compression ratio. The text description consists of both the image caption and responses to a pre-defined set of perception-focused questions *"Are there [obj]?"* where $obj \in \{human\ faces, text\}$. This set can be expanded to accommodate different needs. When images are available, we use InstructBlip[37] to generate the caption and the responses. Otherwise, users need to provide the required description in text.

In stage 2, the text description is processed by an LLM to assess complexity. We use Llama 3 70B Instruct[38] in this work. To ensure consistency in scoring, we design a detailed prompt consisting of the evaluation instructions; the output scale, i.e., an integer score ranging from 1 to 9, where higher scores indicate greater complexity; important

factors for scoring, such as semantic complexity (objects, scenes), visual complexity (color, lighting, texture), and perceptual complexity (presence of faces and text); and lastly, specific examples for each score as demonstrations. We provide the prompt we use in Appendix 7.

We divide the scores into three intervals: $[1, a]$, $(a, b]$, and $(b, 9]$, where $a < b \in \mathbb{Z}^+$. After obtaining the score from the LLM, we classify it into one of 8x, 16x, and 32x compression ratios, with higher complexity scores corresponding to lower compression ratios. The threshold points $a$ and $b$ are selected to achieve an average compression ratio of approximately 16x across all training data, allowing us to make a fair comparison with fixed 16x baselines.

Formally, denote the training distribution as $\mathcal{X}$, input resolution as $r$, the compression ratio of an image $x \in \mathcal{X}$ as $f(x) \in \{f_1, f_2, f_3 | f_1 = 8, f_2 = 16, f_3 = 32\}$, and the target average compression ratio as $\bar{f} := 16$. After collecting the complexity scores for all training images, we set $a, b$ to meet the target compression ratio:

$$\mathbb{E}_{x \in \mathcal{X}} \left[ \frac{r^2}{f(x)^2} \right] \approx \sum_{x \in \mathcal{D}} p(f(x)) \frac{r^2}{f(x)^2} \approx \frac{r^2}{\bar{f}^2} \tag{2}$$

There could be multiple sets of thresholds that achieve the target compression ratio. We show in Section 4.3 that a more diverse distribution of compression ratios leads to better empirical performance. We discuss the exact training data we use and the threshold selection in Section 4.1.

Finally, we verify the proposed caption complexity indeed provides a good estimation of the optimal compression ratio. We compute the correlation between our complexity score and the maximum acceptable compression ratio for COCO images and find that it surpasses all existing metrics (Table 2). Meanwhile, the compression ratio selected by our caption score achieves an exact agreement of 62.39% with the maximum acceptable compression ratio. We also manually inspect the images and confirm that perceptually challenging images are assigned high caption complexity.

### *3.3. Nested VAE for Adaptive Compression*

To reduce training and storage costs, we introduce a nested structure to the standard VAE architecture[2] to enable multiple compression ratios within a single model. In the standard VAE architecture, the encoder consists of multiple downsampling blocks followed by an attention-based middle block. The decoder consists of an attention-based middle block followed by upsampling blocks. This symmetrical design is reminiscent of U-Net[25] and Matryoshka networks[26] for multi-scale feature extraction. Inspired by these works, we leverage the intermediate outputs of the downsampling blocks to enable adaptive compression. We describe the proposed architecture below. See Figure 1 for illustration.

**Skip Connection with Channel Matching.** Denote the feature shape under the largest compression ratio as $(c_3, \frac{r}{f_3}, \frac{r}{f_3})$, where $c_3$ is the channel dimension. We observe that, in the standard VAE encoder, the spatial dimension of the intermediate outputs from the downsampling blocks decreases by a factor of 2 with each additional block. This means that the output of the second-to-last downsampling block naturally has shape $(c_2, \frac{r}{f_2}, \frac{r}{f_2})$, and the output of the third-to-last downsampling block has shape $(c_1, \frac{r}{f_1}, \frac{r}{f_1})$. An immediate thought is to directly route these intermediate outputs to the middle block to generate latent features. However, since the channel dimensions of these intermediate

outputs vary, we leverage ResNet blocks[39] for channel matching. Let the latent channel dimension of the VAE be $c$. Applying channel matching enables us to transform intermediate features of shape $(c_n, \frac{r}{f_n}, \frac{r}{f_n})$ to $(c, \frac{r}{f_n}, \frac{r}{f_n})$ for $n = 1, 2, 3$. This will be the shape of the latent parameters.

For the decoder, similarly, we add skip connection with channel matching and route the output from the decoder's middle block to the corresponding upsampling block. For the compression ratio $f_n$, we bypass the first $n-1$ upsampling blocks to ensure the decoder output has the same resolution as the original image.

**Shared mean/variance parametrization.** In the encoder, features after channel matching are directed to the middle block to generate the parameters of the latent distribution. For the CAT architecture, we share the middle block for all compression ratios to maintain scale consistency of the parameterized mean and variance. The convolutional design of the middle block allows it to process inputs of varying spatial dimensions, as long as the channel dimension is aligned. Thus, for all $n \in \{1, 2, 3\}$, the mean $\mu_n$, variance $\sigma_n^2$, and sample $z_n$ of the Gaussian distribution all have shape $(c, \frac{r}{f_n}, \frac{r}{f_n})$, which is the original input compressed at ratio $f_n$.

**Increasing parameter allocation for shared modules.** Images assigned smaller compression ratios do not go through the later downsampling blocks and are directed straight to the middle block. The middle block is thus tasked with handling multi-scale features. To improve its capacity, we allocate more parameters to the middle block by increasing the number of attention layers.

**Training.** While existing adaptive tokenizers like ElasticTok[9] do not consider the different complexity levels within the training data, we explicitly incorporate content complexity into the training process to learn feature extraction at different granularity. For each training example, we first obtain the compression ratio from the LLM evaluation system. Then, the image is processed only by the layers dedicated to its compression ratio.

Similar to prior works[2][1], we use a joint objective that minimizes reconstruction error, Kullback-Leibler (KL) divergence, and perceptual loss. Specifically, we use $L_1$ loss for pixel-wise reconstruction. To encourage the encoder output $z$ towards a normal distribution, KL-regularization is added: $\mathcal{L}_{\text{KL}}(z) := \mathbb{KL}\left(q_\theta(z|x) \parallel p(z)\right)$, where $\theta$ is the encoder parameters and $p(z) \sim \mathcal{N}(0, \mathbf{I})$. The perceptual loss consists of the LPIPS similarity[35] and a loss based on the internal features of the MoCo v2 model[40]. Beyond these, we train our tokenizer in an adversarial manner[41] using a patch-based discriminator $\psi$. This leads to an additional GAN loss $\mathcal{L}_{\text{GAN}}(x, \hat{x}, \psi)$. Thus, our overall objective is:

$$\begin{aligned}\mathcal{L} = \min_{\theta} \max_{\psi} \mathbb{E}_{x \in \mathcal{X}} [\mathcal{L}_{\text{rec}}(x, \hat{x}) + \beta \mathcal{L}_{\text{KL}}(z) \\ + \gamma \mathcal{L}_{\text{perc}}(\hat{x}) + \delta \mathcal{L}_{\text{GAN}}(x, \hat{x}, \psi)],\end{aligned} \tag{3}$$

where $\beta, \gamma, \delta$ are the weights for each loss term. To simplify implementation, we first sample a compression ratio for each GPU and ensure a batch of training data contains images with the same compression ratio. However, different GPUs can have different compression ratios.

## 4. Image Reconstruction

We first evaluate CAT on image reconstruction. We will present downstream generation results in Section 5.

### *4.1. Setup*

**Model and Training.** We use a nested VAE architecture with six downsampling blocks; the output channels are 64, 128, 256, 256, 512, 512. We use 8 attention layers for the middle block. The latent channel $c$ is 16 for experiments in Table 1, but we study its effect as an ablation study in Table 5. The total number of parameters is 187M.

For training data, we use a collection of 380M licensed Shutterstock images with input resolution 512. After obtaining the complexity scores, we find that two sets of threshold points, $(a, b) \in \{(4, 7), (2, 8)\}$, both achieve an average compression ratio of approximately 16x. However, since $(4, 7)$ leads to a more diverse distribution and better emprical performance (see Table 3 and ablation studies in Section 4.3), we use it in the final setup of CAT. All models including the baselines are trained using a global batch size of 512 on 64 NVIDIA A100 GPUs for 1M steps. Further architecture and training details (e.g., loss weights, optimizer, and learning rate schedule) can be found in Appendix 8.

**Baselines.** We compare CAT against fixed compression ratio baselines that use the same VAE architecture but without the nested structure. To study the effect of caption-based complexity, we train another nested VAE using the JPEG file size of the image as the complexity metric. We ensure all models have average 16x compression. See Appendix 8.3 for more baseline details.

**Evaluation Datasets and Metrics.** We evaluate the reconstruction performance on four datasets: COCO[11] and ImageNet[12], representing natural images; CelebA[13] and ChartQA[14], representing perceptually challenging images. We report reconstruction FID (rFID), LPIPS, and PSNR[42] as the performance metrics.

### *4.2. Main Results*

Table 1 presents the image reconstruction results of CAT and various baselines. For fixed compression methods, the 8x compression ratio achieves substantially better performance than the 16x and 32x compression ratios, which shows that reducing the compression ratio is an effective strategy to improve reconstruction at the cost of increased computational expense. Then, we compare our method with the fixed 16x baseline. On COCO and ImageNet, CAT generally outperforms the baseline, with only a slight drop in rFID on ImageNet. However, the average dimension of CAT latent features is 31.87 for COCO and 29.32 for ImageNet, both of which are smaller than the baseline dimension of 32 (Table 2). This shows CAT can effectively learn compact representations for natural images. On CelebA and ChartQA, CAT significantly outperforms the baselines on all metrics. On ChartQA, CAT even surpasses the fixed 8x baseline, proving its efficacy in capturing visual details.

| Average Compression | | | COCO | | | ImageNet | | | CelebA | | | ChartQA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | rFID ↓ | LPIPS ↓ | PSNR ↑ | rFID ↓ | LPIPS ↓ | PSNR ↑ | rFID ↓ | LPIPS ↓ | PSNR ↑ | rFID ↓ | LPIPS ↓ | PSNR ↑ |
| 8 | Fixed | 8x | 0.48 | 0.10 | 30.95 | 0.24 | 0.095 | 33.86 | 1.86 | 0.028 | 45.36 | 8.21 | 0.019 | 36.98 |
| 16 | Fixed | 16x | 0.66 | 0.16 | 29.79 | **0.38** | **0.15** | 30.45 | 2.25 | 0.059 | 41.84 | 8.67 | 0.029 | 33.48 |
| | Adaptive | JPEG | 0.72 | 0.17 | 30.11 | 0.51 | 0.16 | 30.61 | 6.57 | 0.14 | 36.47 | 10.17 | 0.048 | 31.54 |
| | Adaptive | CAT (**Ours**) | **0.65** | **0.15** | **30.19** | 0.46 | **0.15** | **30.62** | **1.97** | **0.051** | **42.43** | **5.27** | **0.021** | **36.45** |
| 32 | Fixed | 32x | 1.18 | 0.26 | 26.93 | 0.81 | 0.25 | 27.48 | 6.10 | 0.16 | 36.35 | 10.79 | 0.045 | 30.99 |

**Table 1. Reconstruction results.** All models have latent channel $c = 16$. CAT outperforms fixed 16x and JPEG baselines on most metrics.

We also compare CAT with training the same adaptive architecture but using JPEG size as the complexity metric. Across all datasets, CAT achieves better rFID, LPIPS, and PSNR. While we ensure both tokenizers have the same trainingcompression ratio distribution, the compression ratio distribution of the evaluation datasets varies significantly (Table 2). Notably, since JPEG size often cannot capture perceptually important factors (see discussion in Section 3.1.2), nearly all images in CelebA and ChartQA are assigned the highest 32x compression ratio. Thus, CAT significantly outperforms JPEG on these two datasets, showing the effectiveness of caption-based metric and LLM evaluation in determining an image's intrinsic complexity.


doi.org/10.32388/WCBNQ2

| Eval Dataset | Compression Method | Eval Distribution | | | Latent Dim |
|---|---|---|---|---|---|
| | | 8x | 16x | 32x | |
| COCO | CAT | 9% | 54% | 37% | 31.87 |
| | JPEG | 10% | 54% | 36% | 32.43 |
| ImageNet | CAT | 6% | 49% | 45% | 29.32 |
| | JPEG | 9% | 49% | 42% | 31.24 |
| CelebA | CAT | 17% | 83% | 0% | 39.29 |
| | JPEG | 0% | 0% | 100% | 16 |
| ChartQA | CAT | 96% | 4% | 0% | 63.02 |
| | JPEG | 0% | 3% | 97% | 16.61 |

**Table 2.** Test data distribution and average spatial dimension ($\frac{r}{f}$) of the latent features. The numbers denote the proportion of images for each dataset. Compared to fixed 16x baseline, which has a latent dimension of $\frac{512}{16} = 32$, CAT uses smaller latents for natural images and larger latents for CelebA and ChartQA.

Figure 4 shows qualitative examples of progressive reconstruction quality using the learned CAT VAE as we manually reduce the compression ratio and use more tokens to represent each image. We highlight the compression ratio selected by our caption metric in red. Different visual inputs have different optimal compression ratios. Natural images with fewer objects and simpler patterns can be accurately reconstructed at 32x, whereas complex images with visual details require lower compression. Thus, the caption-based CAT reconstruction has comparable quality to the fixed 16x baseline on natural images but surpasses it on text-heavy images. These results further demonstrate the effectiveness of CAT. We include more visualization and comparison with LDM VAEs in Appendix 9.4.

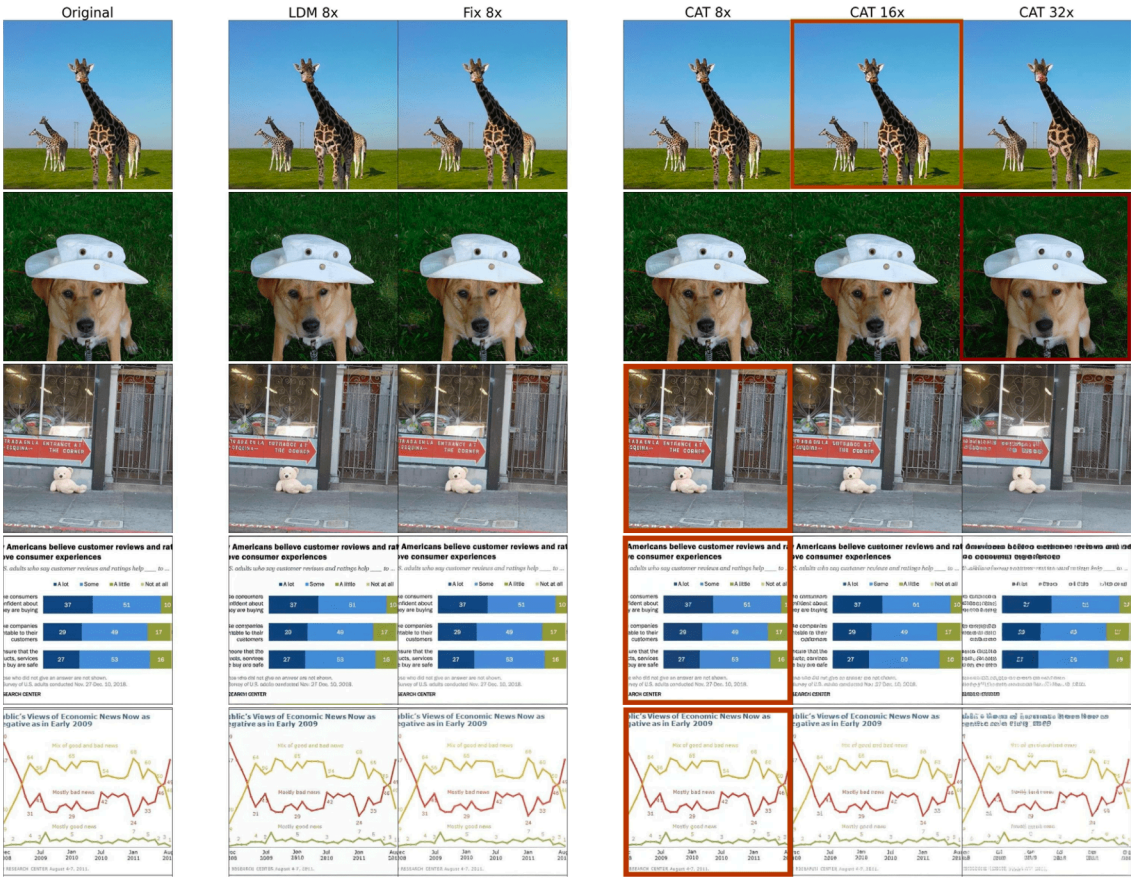



**Figure 4.** We highlight the compression ratio selected by our proposed caption complexity in red. On simpler images (top two rows), adjusting the CAT compression ratio does not significantly affect quality. On more complex images (bottom three rows), the impact is substantial. Also note that CAT's text reconstruction is comparable with fixed 8x baseline and better than pretrained LDM VAE. Images shown in the figure are taken from COCO 2014[11] and ChartQA[14].

### *4.3. Ablation Studies*

We explore several design choices for our tokenizer. First, we study how the distribution of compression ratios affects overall reconstruction. To achieve an average compression ratio of 16, we consider setting the thresholds $(a, b)$ to either $(4, 7)$ or $(2, 8)$. As shown in Table 3, the configuration $(4, 7)$ yields a more diverse distribution, whereas $(2, 8)$ results in a distribution that is more concentrated and similar to a fixed 16x tokenizer—making it a less interesting setting. Table 3 also compares the reconstruction performance of these configurations. The thresholds $(4, 7)$ produce better reconstruction metrics across all datasets, possibly because the diversity in compression ratios ensures that all parts of the model are fully trained. Consequently, we adopt $(4, 7)$ as the thresholds for CAT.

| $(a, b)$ | 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 3.** 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.

| **DiT-XL/2+Tokenizer** | | **FID↓** | **sFID↓** | **IS↑** | **Precision↑** | **Recall↑** | **Eval rFLOPs↓** |
|---|---|---|---|---|---|---|---|
| Fixed | LDM VAE | 10.03 | 16.88 | 114.84 | 0.65 | **0.50** | 1× |
| | Fixed 16x | 4.78 | 11.81 | 187.47 | 0.72 | 0.49 | 1× |
| Adaptive | CAT | **4.56** | **10.55** | **191.09** | **0.75** | 0.49 | **0.82**× |

**Table 4. 512× 512 class-conditional ImageNet generation results after 400K training steps (cfg=1.5).** All tokenizers have average compression ratio $\overline{f} = 16$ and latent channel $c = 16$. "rFLOPs" means relative FLOPs.

We also vary the latent channel dimension $c$ to study its effect on tokenizer performance. As shown in Table 5, a larger $c$ leads to better reconstruction metrics. However, consistent with previous studies[10][43], we observe a reconstruction-generation trade-off: while increasing $c$ improves reconstruction quality of the tokenizer, it does not necessarily result in better second-stage generative performance. We elaborate on this trade-off in the next section.

| 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 5. Larger latent channel $c$ generally improves rFID.**

## 5. Image Generation

In this section, we use CAT to develop image generation models for ImageNet dataset. Given the continuous and adaptive nature of CAT, we use the diffusion transformer (DiT)[15] as the second-stage model, which is capable of handling variable-length token sequences. DiT takes the noised latent features as input, applies patching to further downsample the input, and uses a transformer architecture to predict the added noise.

### *5.1. Setup*

Following[15], we utilize DiT-XL with 431M parameters and a patch size of 2. We work with images of input resolution 512. With a 16x compression during tokenization and an additional 2x compression during patching, the number of patches (referred to as "tokens" hereafter) representing each image is $(\frac{512}{16 \cdot 2})^2 = 256$.

Since the ImageNet dataset does not naturally include text captions, we employ InstructBlip to generate captions for the images individually during training. For inference, we use the caption *"this is an image of [label]"*. We follow our scoring system to determine the target number of tokens to generate—specifically, 64 for 32x decoder, 256 for 16x decoder, and 1024 for 8x decoder.

As for baselines, we consider DiT-XL paired with the open-source 16x LDM VAE and the fixed 16x tokenizer trained in the previous section. We train all models with the same global token batch size of 262,144, which is equivalent to 1,024 images at a 16x compression ratio, and for 400,000 steps on 16 NVIDIA H100 GPUs. Following the original DiT work, we report FID[44], Sliding FID[45], Inception Score[46], precision and recall[47] on 50K images generated with 250 DDPM sampling steps and classifier-free guidance[48]. See Appendix 9 for details.

### *5.2. Results*

Table 4 summarizes the results, showing that CAT achieves the best FID, sFID, IS, and precision among all baselines trained with the same computational resources. We attribute this strong performance to two factors. First, adaptively

allocating representation capacity enables more effective modeling of complex images while reducing noise in simpler ones. Second, using fewer tokens for simpler images improves processing efficiency, allowing for more extensive and diverse training within the same computational budget. Specifically, since ImageNet primarily consists of natural images, only a few classes featuring people or fine-grained text receive high complexity scores. On the training dataset, the average token count per image for DiT-CAT is 197.44, which is 23% lower than the 256 tokens used by DiT with fixed 16x tokenizers. During inference, this average increases to 216, leading to an 18.5% increase in inference throughput (samples per second).

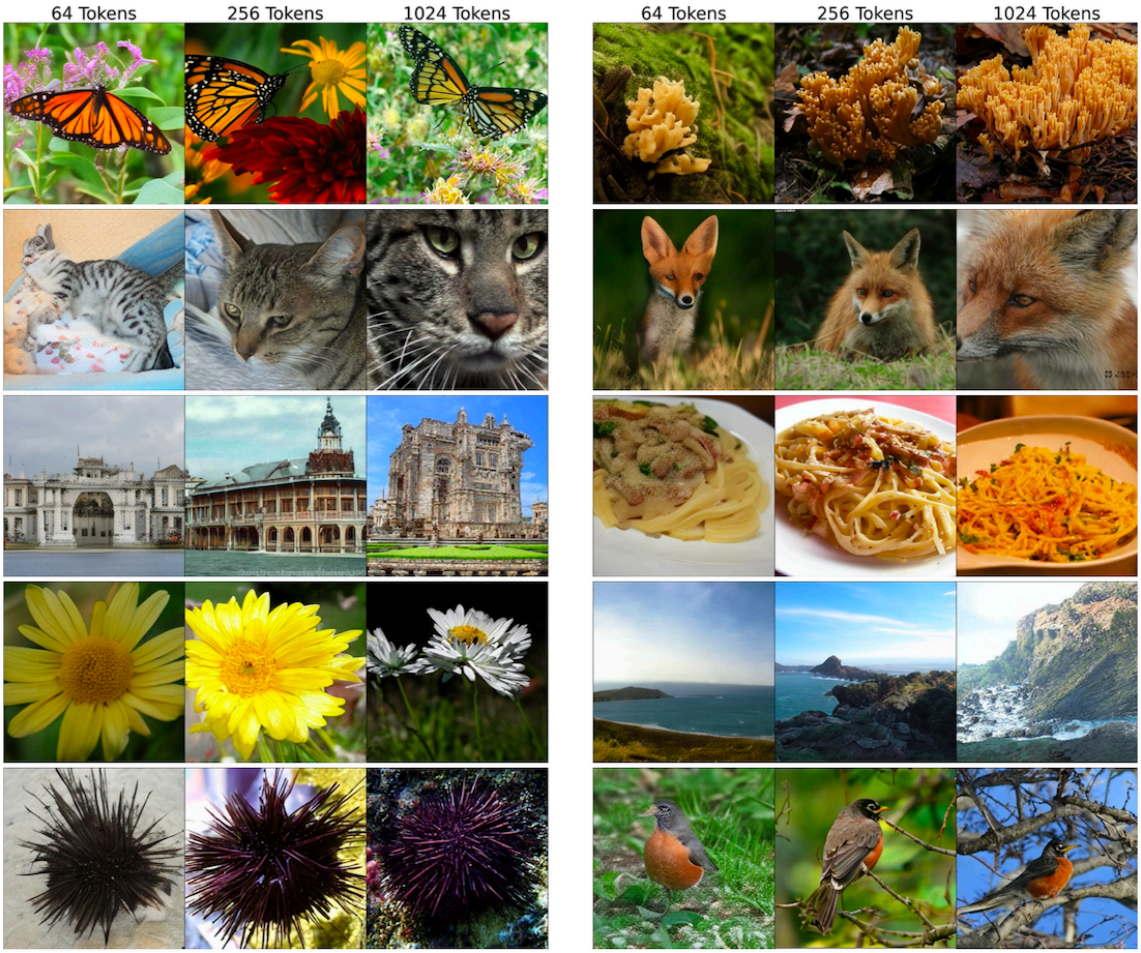



**Figure 5. Increasing token count (left→right) for CAT leads to better image quality and higher complexity.**

We study the effect of manually increasing the number of tokens for DiT-CAT during generation. Table 6 shows that FID score is significantly improved when using more tokens during image generation. We further provide qualitative examples. As Figure 5 shows, utilizing more tokens leads to more complex images, such as featuring more objects and more intricate texture. This highlights a side benefit of adaptive tokenization: it enables complexity-controllable generation at no additional training cost.


doi.org/10.32388/WCBNQ2

Lastly, recall that we trained tokenizers with different latent channels in Section 4.3. Table 7 shows the generation performance. While larger $c$ is better for reconstruction, it is not the case for generation. In fact, $c = 8$ leads to better average results for both fixed and adaptive settings, and CAT with $c = 8$ obtains the best FID across all experiments we perform. This observation agrees with existing work [10] and underscores the importance of choosing an appropriate $c$. We leave diving into the dynamic of latent channel dimension and downstream performance as future work.

| | **CAT 8x** | **CAT 16x** | **CAT 32x** |
|---|---|---|---|
| FID-50K | 4.12 | 5.02 | 5.83 |

**Table 6. We manually adjust the inference token count for CAT with $c = 8$ to control the complexity of the generated images.**

| | $c$ | **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 7. Larger channel $c$ is not always better for generation.** Contrary to Table 5, we find that increasing channel dimension does not always result in generation gains.

## 6. Discussion and Conclusion

In this work, we propose an adaptive image tokenizer, CAT, which allocates different number of tokens to represent images based on content complexity derived from the text description of the image. Our experiments show that CAT improves both the quality and efficiency of image reconstruction and generation. We identify several future directions to work on. First, we can apply complexity-driven compression to developing discrete tokenizers and combine CAT with quantization techniques. Besides, experimenting with more downstream tasks beyond class-conditional generation[49][50] and integrating CAT to multi-modal models, such as Chameleon[51] and Transfusion[52], can help strengthen this work. Lastly, extending CAT to video tokenization could be a promising future direction due to the higher inherent redundancies in video clips, especially along the temporal dimension.

## Supplementary Material

# 7. Prompt for LLM Scorer

Our caption complexity pipeline works as follows:

Step 1: Use `Salesforce/instructblip-vicuna-7b` to generate caption, 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 `meta-llama/Meta-Llama-3-70B-Instruct` 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.

# 8. Reconstruction Experiments

### *8.1. Architecture*

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

- `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

The model sizes for different latent channels are shown below. As for the discriminator, we use the pretrained StyleGAN[53] architecture.

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

### *8.2. Training*

We use the following training configuration:

- 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
  - 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.

### *8.3. 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.

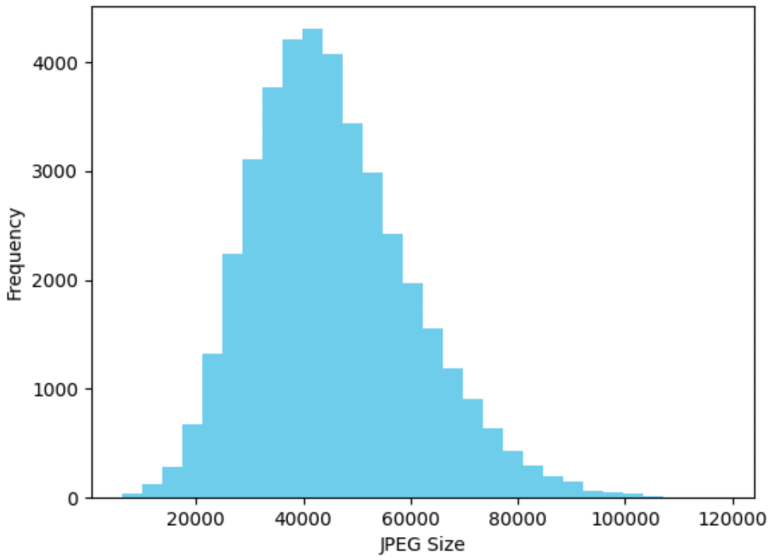



**Figure 6.** On COCO 2014 test set, the minimum JPEG size is 6128; maximum is 118428; mean is 45474.29; standard deviation is 15037.07.

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[54], 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 8.

| | 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 8.** rFIDs for CAT and LDM VAEs.

### *8.4. More Reconstruction Visualization*

See Figure 7 in the end.

## 9. Generation Experiments

### *9.1. 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.

### *9.2. Training & Inference*

The training configuration for DiT is as follows:

- GPU: 16 NVIDIA H100
- Per-GPU token batch size: $4096 \times 4$ (equivalent to 64 images for 16x compression ratio)
- Global token batch size: $4096 \times 64$
- 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

DDPM scheduler (diffusers implementation):

- 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

For 10 % of the time, we remove the image class label from the input and train unconditional image generation. All FID-50K and images generated in this paper are using cfg=1.5.

### *9.3. 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.

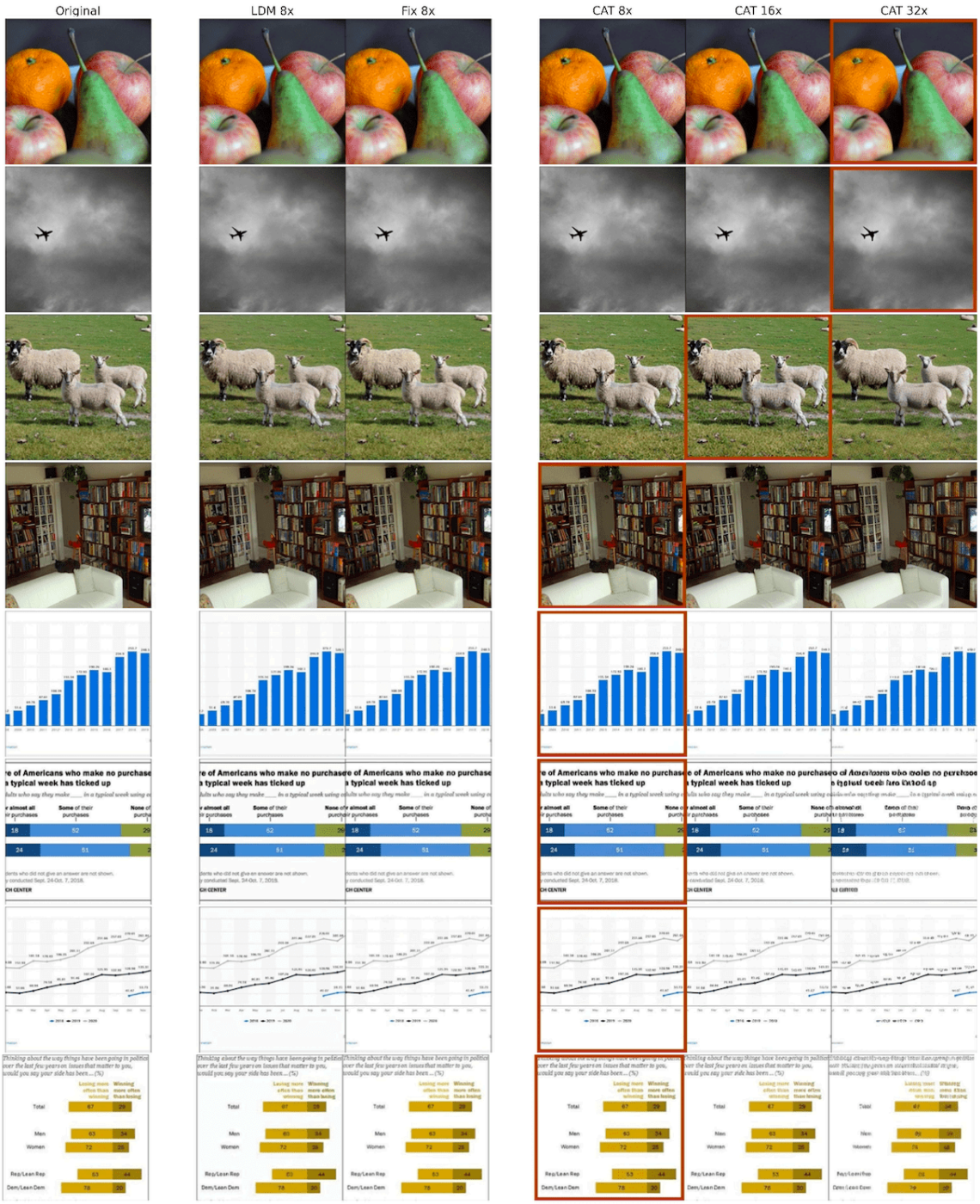



**Figure 7. More CAT reconstruction examples. We highlight the compression ratio selected by our proposed caption complexity in red.** Images shown in the figure are taken from COCO 2014[11] and ChartQA[14].

## *9.4. More Visualization*

See Figure 8 in the end.

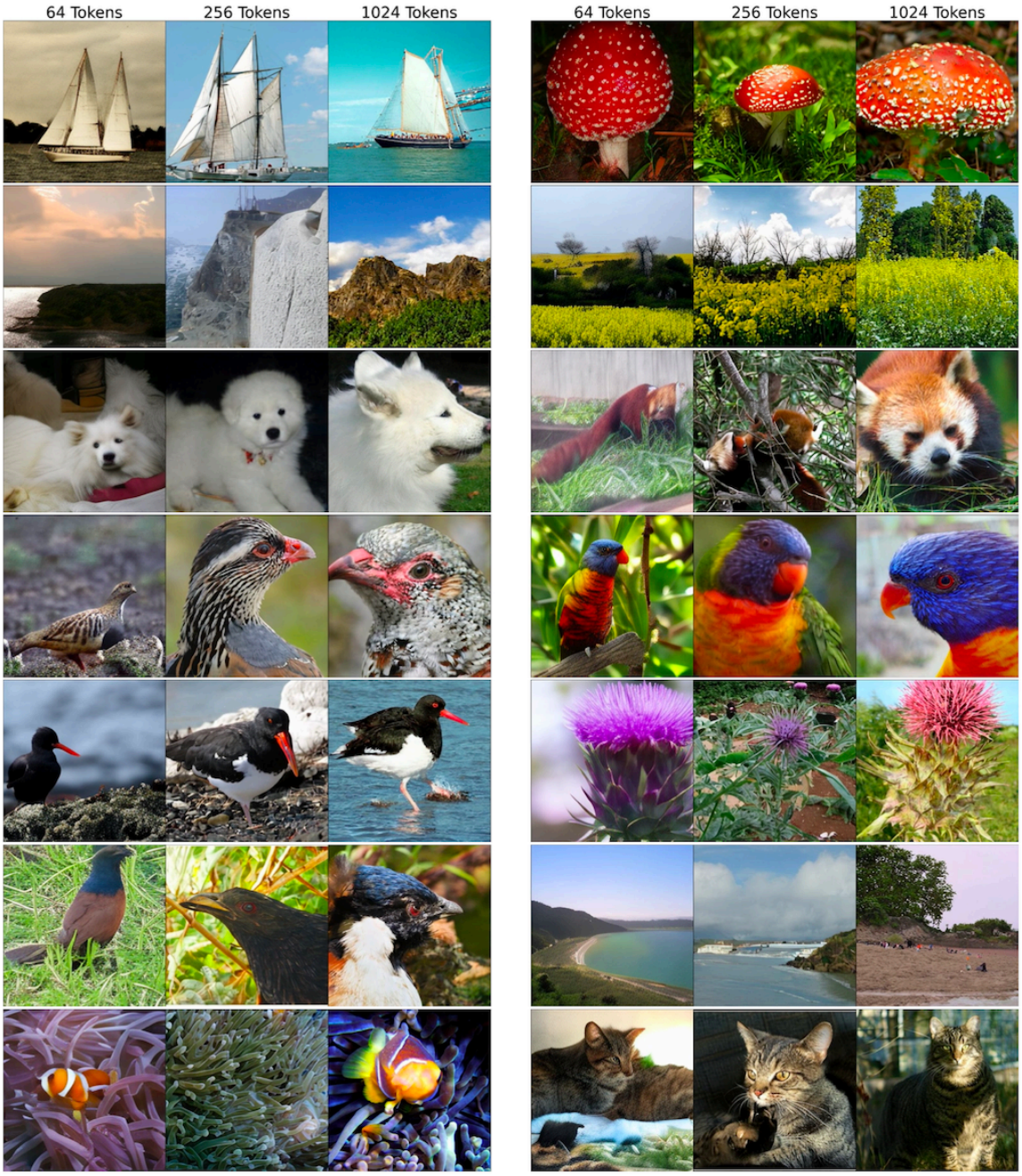



**Figure 8. More DiT-CAT generation examples. Increasing token count (left→right) generally leads to better image quality and higher complexity.**

## Notes

- **Junhong Shen:** Work done during an internship at Meta.
- **Lili Yu:** Joint senior author.
- **Chunting Zhou:** Work done while at Meta.

## Acknowledgments

We would like to thank Omer Levy, Daniel Li, Hu Xu for helpful discussion throughout the project.

## Footnotes

[1] LDM released a series of VAE tokenizers with diverse compression ratios and trained in a controlled setting. Most other tokenizers, such as stabilityai/sd-vae-ft-mse, only have one compressed ratio.

[2] Note that the average MSE across all images for 8x LDM VAE is 0.0039, so a 0.0001 tolerance should be acceptable.

[3] We use stabilityai/sd-vae-ft-mse for this analysis.

## References

1. ^a, ^b*Esser P, Rombach R, Ommer B (2020). "Taming Transformers for High-Resolution Image Synthesis". arXiv. arXiv:2012.09841.*
2. ^a, ^b, ^c, ^d*Kingma DP, Welling M (2014). "Auto-Encoding Variational Bayes". In: ICLR.*
3. ^^*Yu L, Lezama J, Gundavarapu NB, Versari L, Sohn K, Minnen D, Cheng Y, Birodkar V, Gupta A, Gu X, Hauptmann AG, Gong B, Yang MH, Essa I, Ross DA, Jiang L. "Language Model Beats Diffusion -- Tokenizer is Key to Visual Generation". arXiv [cs.CV]. 2024. Available from: https://arxiv.org/abs/2310.05737.*
4. ^^*Yu Q, Weber M, Deng X, Shen X, Cremers D, Chen L (2024). "An Image is Worth 32 Tokens for Reconstruction and Generation". arxiv: 2406.07550.*
5. ^^*Shen J, Khodak M, Talwalkar A (2022). "Efficient architecture search for diverse tasks". Advances in Neural Information Processing Systems (NeurIPS).*
6. ^^*Tu R, Roberts N, Khodak M, Shen J, Sala F, Talwalkar A (2022). "NAS-Bench-360: Benchmarking Neural Architecture Search on Diverse Tasks". In: Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track.*
7. ^a, ^b*Mentzer F, Minnen D, Agustsson E, Tschannen M (2023). "Finite Scalar Quantization: VQ-VAE Made Simple". arXiv. Available from: https://arxiv.org/abs/2309.15505.*
8. ^a, ^b*Wallace GK. "The JPEG still picture compression standard". IEEE Transactions on Consumer Electronics. 38 (1): xviii–xxxiv, 1992. doi:10.1109/30.125072.*
9. ^a, ^b, ^c*Yan W, Zaharia M, Mnih V, Abbeel P, Faust A, Liu H (2024). "ElasticTok: Adaptive Tokenization for Image and Video". arXiv preprint. 2024.*
10. ^a, ^b, ^c, ^d*Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B (2021). "High-Resolution Image Synthesis with Latent Diffusion Models". arXiv. arXiv:2112.10752 [cs.CV].*
11. ^a, ^b, ^c, ^d, ^e, ^f, ^g*Lin TY, Maire M, Belongie S, Bourdev L, Girshick R, Hays J, Perona P, Ramanan D, Zitnick CL, Dollár P (2015). "Microsoft COCO: Common Objects in Context". arXiv. arXiv:1405.0312 [cs.CV].*

12. a, b Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: CVPR09. Available from: http://www.image-net.org/papers/imagenet_cvpr09.bib.
13. a, b Liu Z, Luo P, Wang X, Tang X (2015). "Deep Learning Face Attributes in the Wild". In: Proceedings of International Conference on Computer Vision (ICCV), December 2015.
14. a, b, c, d Masry A, Long DX, Tan JQ, Joty S, Hoque E (2022). "ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning". arXiv. arXiv:2203.10244 [cs.CL].
15. a, b, c Peebles W, Xie S (2022). "Scalable Diffusion Models with Transformers". arXiv preprint arXiv:2212.09748. Available from: https://arxiv.org/abs/2212.09748.
16. ^ van den Oord A, Vinyals O, Kavukcuoglu K. Neural Discrete Representation Learning. 2018. Available from: https://arxiv.org/abs/1711.00937.
17. ^ Wiegand T, Sullivan GJ, Bjontegaard G, Luthra A (2003). "Overview of the H.264/AVC video coding standard". IEEE Transactions on Circuits and Systems for Video Technology. **13** (7): 560–576. doi:10.1109/TCSVT.2003.815165.
18. ^ Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N (2020). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". CoRR. **abs/2010.11929**. Available from: https://arxiv.org/abs/2010.11929.
19. ^ Rao Y, Zhao W, Liu B, Lu J, Zhou J, Hsieh C-J (2021). "DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification". In: Advances in Neural Information Processing Systems (NeurIPS).
20. ^ Yin H, Vahdat A, Alvarez J, Mallya A, Kautz J, Molchanov P (2022). "A-ViT: Adaptive Tokens for Efficient Vision Transformer". Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.
21. ^ Bolya D, Fu CY, Dai X, Zhang P, Feichtenhofer C, Hoffman J (2023). "Token Merging: Your {ViT} but Faster". International Conference on Learning Representations.
22. ^ Chen L, Tong Z, Song Y, Wu G, Wang L (2023). "Efficient Video Action Detection with Token Dropout and Context Refinement". arXiv. Available from: https://arxiv.org/abs/2304.08451.
23. ^ Ronen T, Levy O, Golbert A (2023). "Vision Transformers with Mixed-Resolution Tokenization". arXiv. Available from: https://arxiv.org/abs/2304.00287.
24. ^ Duggal S, Isola P, Torralba A, Freeman WT (2024). "Adaptive Length Image Tokenization via Recurrent Allocation". arXiv. arXiv:2411.02393 [cs.CV].
25. a, b Ronneberger O, Fischer P, Brox T (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". arXiv. arXiv:1505.04597 [cs.CV].
26. a, b Kusupati A, Bhatt G, Rege A, Wallingford M, Sinha A, Ramanujan V, Howard-Snyder W, Chen K, Kakade S, Jain P, et al. Matryoshka representation learning. In: Advances in Neural Information Processing Systems; December 2022.
27. ^ Cai M, Yang J, Gao J, Lee YJ (2024). "Matryoshka Multimodal Models". arXiv preprint arXiv:2405.17430.
28. ^ Gu J, Zhai S, Zhang Y, Susskind J, Jaitly N. Matryoshka Diffusion Models. 2024. Available from: https://arxiv.org/abs/2310.15111.

29. ^Nash C, Carreira J, Walker J, Barr I, Jaegle A, Malinowski M, Battaglia P (2022). "Transframer: Arbitrary Frame Predictio n with Generative Models". arXiv. Available from: https://arxiv.org/abs/2203.09494.

30. ^Hu W, Dou ZY, Li LH, Kamath A, Peng N, Chang KW (2024). "Matryoshka Query Transformer for Large Vision-Language Models". arXiv. arXiv:2405.19315 [cs.CV].

31. ^Shen J, Li L, Dery LM, Staten C, Khodak M, Neubig G, Talwalkar A. Cross-modal fine-tuning: align then refine. In: Proce edings of the 40th International Conference on Machine Learning; 2023; Honolulu, Hawaii, USA. p. 1285.

32. ^Shen J, Marwah T, Talwalkar A (2024). "Ups: Towards foundation models for pde solving via cross-modal adaptation". arXiv preprint arXiv:2403.07187.

33. ^Roberts N, Guo S, Xu C, Talwalkar A, Lander D, Tao L, Cai L, Niu S, Heng J, Qin H, Deng M, Hog J, Pfefferle A, Shivakumar SA, Krishnakumar A, Wang Y, Sukthanker RS, Hutter F, Hasanaj E, Le TD, Khodak M, Nevmyvaka Y, Rasul K, Sala F, Schne ider A, Shen J, Sparks ER. AutoML Decathlon: Diverse Tasks, Modern Methods, and Efficiency at Scale. In: Neural Informat ion Processing Systems; 2021. Available from: https://api.semanticscholar.org/CorpusID:265536645.

34. ^Shen J, Tenenholtz N, Hall JB, Alvarez-Melis D, Fusi N (2024). "Tag-LLM: Repurposing General-Purpose LLMs for Speci alized Domains". arXiv. arXiv:2402.05140.

35. a, bZhang R, Isola P, Efros AA, Shechtman E, Wang O (2018). "The Unreasonable Effectiveness of Deep Features as a Perce ptual Metric". In: CVPR.

36. ^Simonyan K, Zisserman A (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". arXiv:1409.1 556 [cs.CV].

37. ^Dai W, Li J, Li D, Tiong AMH, Zhao J, Wang W, Li B, Fung P, Hoi S (2023). "InstructBLIP: Towards General-purpose Visio n-Language Models with Instruction Tuning". arXiv. Available from: https://arxiv.org/abs/2305.06500.

38. ^Dubey A, Jauhri A, Pandey A, Kadian A, Al-Dahle A, Letman A, Mathur A, Schelten A, Yang A, Fan A, et al. The Llama 3 H erd of Models. 2024. Available from: https://arxiv.org/abs/2407.21783.

39. ^He K, Zhang X, Ren S, Sun J (2015). "Deep Residual Learning for Image Recognition". arXiv. arXiv:1512.03385 [cs.CV].

40. ^He K, Fan H, Wu Y, Xie S, Girshick R (2020). "Momentum Contrast for Unsupervised Visual Representation Learning". ar Xiv. arXiv:1911.05722 [cs.CV].

41. ^Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014). "Generative Adve rsarial Networks". arXiv. Available from: https://arxiv.org/abs/1406.2661.

42. ^Horé A, Ziou D (2010). "Image Quality Metrics: PSNR vs. SSIM." In: 2010 20th International Conference on Pattern Recog nition, 2366-2369. doi:10.1109/ICPR.2010.579.

43. ^Dai X, Hou J, Ma CY, Tsai S, Wang J, Wang R, Zhang P, Vandenhende S, Wang X, Dubey A, Yu M, Kadian A, Radenovic F, Mahajan D, Li K, Zhao Y, Petrovic V, Singh MK, Motwani S, Wen Y, Song Y, Sumbaly R, Ramanathan V, He Z, Vajda P, Pari kh D (2023). "Emu: Enhancing Image Generation Models Using Photogenic Needles in a Haystack". arXiv. Available from: https://arxiv.org/abs/2309.15807.

44. ^Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S (2018). "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". arXiv. arXiv:1706.08500 [cs.LG].

45. ^Ding X, Wang Y, Xu Z, Welch WJ, Wang ZJ (2023). "Continuous Conditional Generative Adversarial Networks: Novel Empirical Losses and Label Input Mechanisms". arXiv. Available from: https://arxiv.org/abs/2011.07466.

46. ^Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X (2016). "Improved Techniques for Training GANs". arXiv. arXiv:1606.03498 [cs.LG].

47. ^Kynkäänniemi T, Karras T, Laine S, Lehtinen J, Aila T (2019). "Improved Precision and Recall Metric for Assessing Generative Models". arXiv. Available from: https://arxiv.org/abs/1904.06991.

48. ^Ho J, Salimans T. Classifier-free diffusion guidance, 2022. arXiv. Available from: https://arxiv.org/abs/2207.12598.

49. ^Shen J, Yang LF. "Theoretically Principled Deep RL Acceleration via Nearest Neighbor Function Approximation". Proceedings of the AAAI Conference on Artificial Intelligence. **35** (11): 9558-9566, May 2021. doi:10.1609/aaai.v35i11.17151. https://ojs.aaai.org/index.php/AAAI/article/view/17151.

50. ^Shen J, Jain A, Xiao Z, Amlekar I, Hadji M, Podolny A, Talwalkar A (2024). "ScribeAgent: Towards Specialized Web Agents Using Production-Scale Workflow Data". arXiv:2411.15004 [cs.CL].

51. ^Chameleon Team. Chameleon: Mixed-Modal Early-Fusion Foundation Models. 2024. Available from: https://arxiv.org/abs/2405.09818.

52. ^Zhou C, Yu L, Babu A, Tirumala K, Yasunaga M, Shamis L, Kahn J, Ma X, Zettlemoyer L, Levy O (2024). "Transfusion: Predict the Next Token and Diffuse Images with One Multi-Modal Model". arXiv. arXiv:2408.11039 [cs.AI].

53. ^Karras T, Laine S, Aila T (2019). "A Style-Based Generator Architecture for Generative Adversarial Networks". arXiv. arXiv:1812.04948 [cs.NE].

54. ^Kuznetsova A, Rom H, Alldrin N, Uijlings J, Krasin I, Pont-Tuset J, Kamali S, Popov S, Malloci M, Kolesnikov A, Duerig T, Ferrari V. "The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale." International Journal of Computer Vision. **128**(7):1956–1981, March 2020. doi:10.1007/s11263-020-01316-z.

## Declarations

**Funding:** No specific funding was received for this work.

**Potential competing interests:** No potential competing interests to declare.