FARMER: Flow AutoRegressive Transformer over Pixels

Supplementary Material

6. Related Work

6.1. Continuous AR

A common paradigm in autoregressive image generation is to quantize images into discrete tokens [5, 38, 56, 57, 64, 73, 83] and train autoregressive models on them, as exemplified by LlamaGen [64], Janus-Pro [8], and SimpleAR [75]. However, this design suffers from a key bottleneck: quantization inevitably introduces information loss, which limits the fidelity of the generated images [21, 40, 69].

To address this issue, GIVT [69] uses continuous latents obtained from a VAE to encode images and trains an AR model to predict GMM parameters for approximating token distributions. ARINAR [88] further predicts GMM parameters of each token in a Gaussian-to-Gaussian paradigm. Since GMMs have limited expressive power, Tschannen et al. further introduce a NF to transform GMM samples into tokens, thereby improving generation quality. Jetformer [70] goes one step further by discarding the VAE and directly training AR and NF models in pixel space.

Another line of work explores continuous token modeling by combining AR with diffusion models. In MAR [40], the AR backbone first outputs a conditioning vector for each token, and the diffusion head then generates the next tokens conditioned on that vector. Building on this idea, several other continuous-token approaches have been proposed [10, 11, 22, 30, 42, 46, 53, 58, 59, 67, 79, 80, 82, 89]. For example, FlowAR [58] employs a VAR [68] backbone with flow matching as the generative head; Hi-MAR [89] pivots on low-resolution image tokens to trigger hierarchical autoregressive modeling in a multi-phase manner; xAR [59] autoregressively generates the next groups of tokens through flow matching. Although diffusion-based methods are effective at sampling continuous tokens, they require iterative noise-to-token denoising, which limits the model's ability to perceive and understand images. In contrast, our model directly fits the token distribution without relying on noise sampling.

6.2. Autoregressive Normalizing Flow

Normalizing flows (NFs) [14–17, 19, 32, 35, 47, 48, 51, 60, 72] provide a powerful framework for density estimation, visual generation, and text generation [87], via invertible transformations, which enable exact likelihood computation and efficient sampling. However, the representational capacity of NFs is limited by the expressive power of these invertible transformations. To address this limitation, autoregressive normalizing flows have been proposed, where each token is transformed conditioned on the previous tokens. While AFs significantly improve expressiveness, this autoregressive dependency necessitates a sequential reverse process, leading to substantial latency during inference. Different from previous efforts that focus on model compression—such as distilling a deep NF into a shallow one [74]—we propose a one-step distillation method specifically tailored for AF architectures. Specifically, we distill the slow, sequential reverse process of an AF into a fast, parallel forward pass, achieving efficient sampling while maintaining comparable generation quality. There has been a long line of work on autoregressive normalizing flows, with representative approaches including IAF [34], MAF [50], neural autoregressive flows [29], and T-NAF [54]. More recently, the resurgence of NFs has attracted renewed research interest. TARFlow [86] leverages causal Transformers and simplifies the log-determinant term in the loss function, leading to notable improvements in generation quality. STARFlow extends TARFlow to the VAE latent space and demonstrates that continuous AR flows can deliver competitive generative performance. Meanwhile, JetFormer [70] integrates Jet [36] to enable fully end-to-end continuous AR modeling directly over raw image pixels.

7. Additional Experiments

Qualitative Results. Fig. 10 shows the qualitative results generated by FARMER. In Figs. 11 to 13, we provide additional qualitative results generated by FARMER.

Balancing Precision and Diversity in Unguided Generation. Since the pixel space is vast and the optimization objective encourages broad probability-mass coverage, naive sampling is more likely to hit low-probability regions when sampling without Classifier-Free Guidance (CFG), leading to degraded sample quality. To mitigate this, we propose a decoupled sampling mechanism for GMMs. Specifically, we reduce the Gaussian variance scale s to concentrate sampling on the high-probability regions of individual components, which enhances the precision and local fidelity of samples. Concurrently, we increase the mixture weight temperature T to flatten the categorical distribution; this prevents the model from collapsing into a few dominant modes and preserves sufficient generative diversity. By employing this dual-adjustment strategy ($s = 0.9, T = 1.3$), FARMER significantly improves the unguided FID from 44.56 (as reported in the ablation study, Tab. 2) to 23.29. Tab. 5 summarizes the unguided FID of FARMER across various model scales. The sensitivity analyses of the Gaussian variance scale s and the mixture weight temperature T are visualized in Fig. 7.

Table 5. Unguided generation performance of FARMER with the decoupled sampling strategy.

Model	Epochs	FID↓	
		(w. CFG)	(w/o. CFG)
LlamaGen [64]	300	3.62	19.07
JetFormer 1.4B impl [70]	300	7.40	44.70
FARMER 1.1B patch16	320	5.40	23.07
FARMER 1.1B patch8	320	5.02	19.76
FARMER 1.9B patch16	320	3.96	17.86
FARMER 1.9B patch8	320	**3.60**	**16.13**

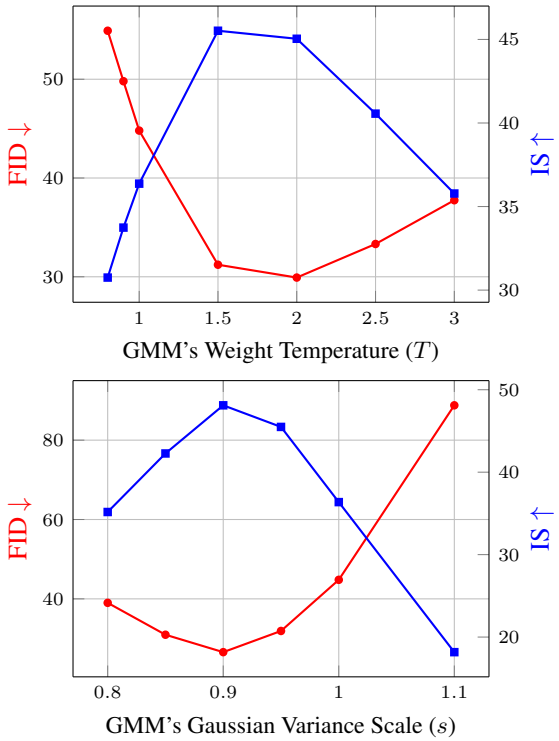

Figure 7. Unguided performance (FID and IS) across varying GMM weight temperatures (T) and Gaussian variance scales (s).

Likelihood Evaluation. To assess the density estimation capabilities of FARMER, we evaluate the Bits-Per-Dimension (BPD) on ImageNet 64×64, following TARFlow [86]. To ensure a fair comparison with TARFlow, we instantiate a comparable variant, FARMER-556M. The architecture comprises an Autoregressive Flow (AF) component and an Autoregressive (AR) backbone. Specifically, the AF part utilizes a patch size of 2 and consists of 16 autoregressive flow blocks, each containing 4 attention layers. The AR backbone is configured with 12 layers. During training, we apply uniform dequantization noise $\mathcal{U}(0, 1/128)$ during training to the image data. We report the BPD results in Tab. 6. FARMER achieves a superior BPD compared to recent state-of-the-art methods such as TARFlow [86] and FractalMAR [41], demonstrating its enhanced data-fitting capability in likelihood terms.

Table 6. **Bits per dim evaluation** on unconditional ImageNet 64x64 test set.

Model	Type	BPD ↓
Glow [32]	Flow	3.81
Flow++ [25]	Flow	3.69
PixelCNN [71]	AR	3.83
Sparse Transformer [9]	AR	3.44
MegaByte [84]	AR	3.40
FractalAR [41]	AR	3.14
FractalMAR [41]	AR	3.15
Improved DDPM [49]	Diff/FM	3.54
VDM [31]	Diff/FM	3.40
Flow Matching [43]	Diff/FM	3.31
NFDM [3]	Diff/FM	3.20
TARFlow [86]	NF	2.99
FARMER-556M	NF+AR	**2.61**

Visualizing the Separation Process of Information in AF. To further examine whether the Autoregressive Flow (AF) component in FARMER—trained end-to-end via the self-supervised objective in Eq. (9)—can effectively disentangle different types of information into two channel groups, we conducted a targeted visualization experiment.

Experimental procedure. First, an image is passed through the trained AF, and intermediate latent representations are extracted from different AF blocks. These latents are then interpolated with Gaussian noise, and the perturbed representations are mapped back to pixel space using the corresponding blocks' inverse transformations.

Experimental settings. We consider two interpolation strategies: (i) Partial Interpolation: Only the final 640 latent channels—designated as redundant dimensions—are interpolated with noise. (ii) Full Interpolation: All latent channels are interpolated with noise.

Observations. As shown in Fig. 8, under the Partial Interpolation setting, structural information increasingly concentrates in the informative channels as the depth of the AF latents increases. Consequently, adding noise to the redundant channels has a diminishing effect on the contour and overall structure of the reconstructed images. In contrast, the Full Interpolation setting (right column in Fig. 8) severely corrupts the contour and structure, especially at high noise levels.

Conclusion. These visualizations indicate that the AF in FARMER successfully learns to separate contour and structural information from color and fine details, allocating them to distinct channel groups.

Impact of Logdet. As defined in the training objective (see Eq. (9)), the log-determinant (logdet) loss term quantifies the volume change induced by the transformation from

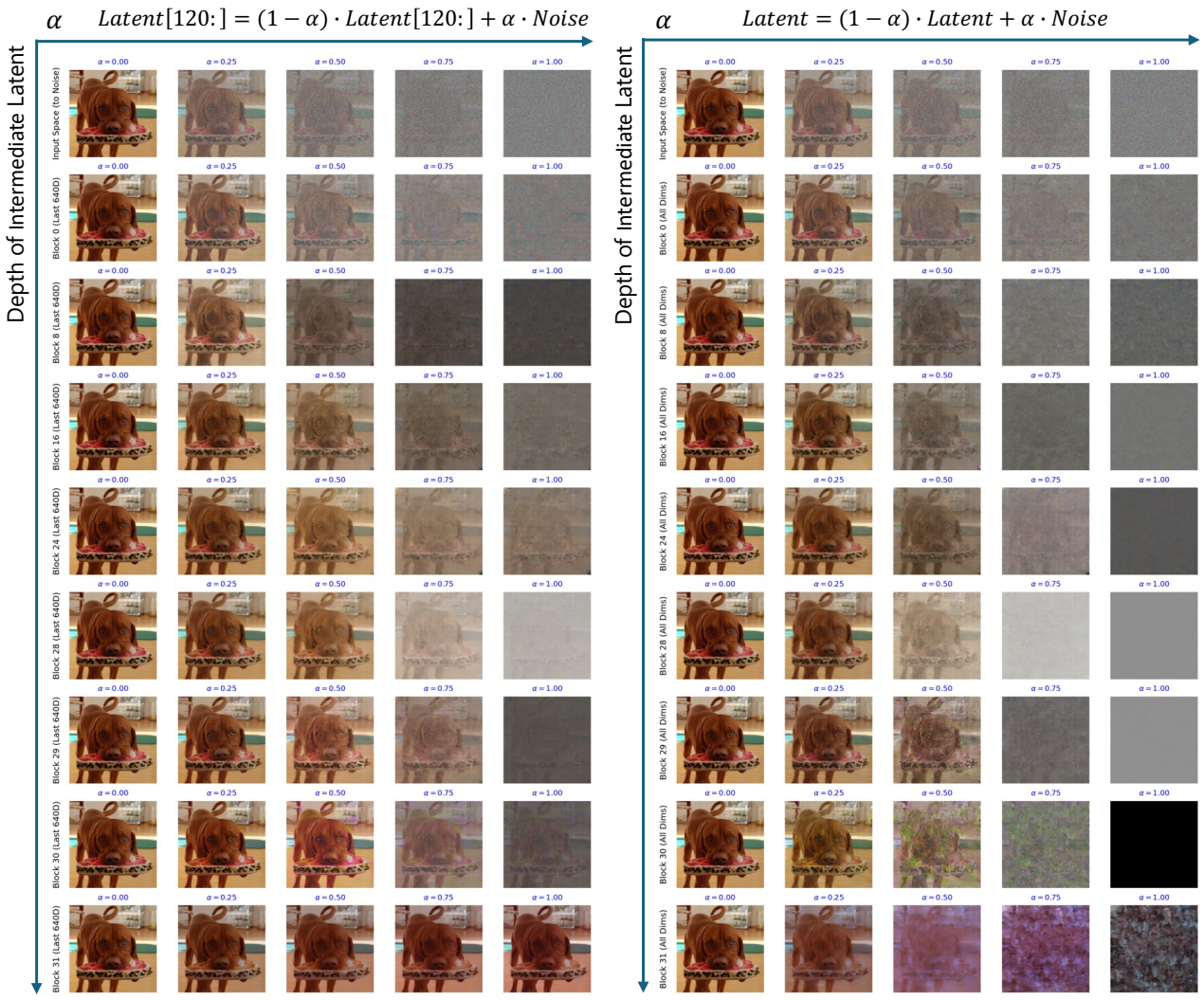

Figure 8. **The separation process of information in the Autoregressive Flow component**.

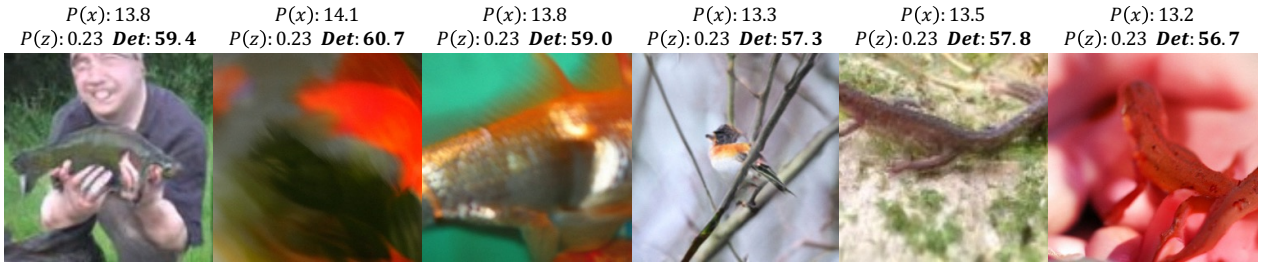

Figure 9. **The sample images with abnormal log-determinant values**. High logdet values cause strong compression in parts of the data space, leading to blurred textures and missing fine-scale details in the generated images.

the data space to the target latent space. As illustrated in Fig. 9, samples with outlier logdet values tend to exhibit a blurred appearance and lack fine-grained details. Excessively large logdet values indicate that certain regions of the latent space are heavily compressed in the data space, which can lead to significant errors when inverting the transformation and reconstructing the data. This suggests that maintaining stable logdet values is crucial for high-fidelity and detail-preserving generation.

8. Training Details

8.1. Experiment Details

Network Architectures. We develop two model scales, namely FARMER-1.1B and FARMER-1.9B. The number of invertible AF blocks is set to 28 for FARMER-1.1B and 32 for FARMER-1.9B. Each AF block contains 4/6 Transformer layers for FARMER-1.1B/1.9B. For the AR Transformer module, the number of Transformer blocks is 12 and 24 respectively. Table 7 summarizes the detailed architectural configurations.

Table 7. The architecture configurations of FARMER in two different scales (i.e., 1.1B and 1.9B).

Model	Autoregressive Transformer			Invertible Autoregressive Flow				Params
	Layers	Hidden size	Params	AF Blocks	Layers	Hidden size	Params	
FARMER-1.1B	12	768	295M	28	4	768	828M	1.1B
FARMER-1.9B	24	1024	498M	32	6	768	1.4B	1.9B

Training Setup. We train the models using the AdamW optimizer ($\beta_1 = 0.9, \beta_2 = 0.95$) with a weight decay of 0.03 for 320 epochs. A cosine learning rate schedule is applied, starting at 1×10^{-4} and decaying to 1×10^{-6}, with a 5,000-step linear warmup. During training, Gaussian noise with a cosine decay from 0.1 to 0.005 is added to the raw image.

8.2. Resample-based CFG Details

While the main text (Sec. 3.4) outlines the proposed Resampling-based Classifier-Free Guidance (CFG) method, we further elaborate on additional tunable parameters that enhance the generation quality and control the diversity. Each of the three stages—*Propose*, *Weigh*, and *Resample*—introduces dedicated temperature coefficients and sampling numbers that can be adjusted.

Propose stage. In the proposal step, candidate samples are drawn from the conditional GMM $p_c(z_i)$ and the unconditional GMM $p_u(z_i)$. To control the diversity at this stage, we introduce two distinct temperature coefficients:

- *Weight temperature* T_π: applied multiplicatively to the mixture weights $\pi_k(z_{<i})$ of the GMM components before normalization. This modulates the relative selection probability among Gaussian components.
- *Variance temperature* T_σ: applied multiplicatively to the variance $\sigma_k(z_{<i})$ of each Gaussian component, scaling the spread of proposals.

Additionally, the number of samples drawn from p_c and p_u can differ; we denote these by s_c and s_u. This allows for striking a balance between strongly conditioned proposals and broader unconditional exploration.

Weigh stage. Given candidate samples $z_{i,j}$, their importance weights are computed as:

$$\log \omega_{i,j} = w \cdot (\log p_c(z_{i,j}; T_{\pi,v}, T_{\sigma,v}) - \log p_u(z_{i,j}; T_{\pi,v}, T_{\sigma,v})),$$

where $T_{\pi,v}$ and $T_{\sigma,v}$ are temperature coefficients for the evaluation distributions in this stage (which need not be equal to those used in the *Propose* stage). These temperatures control the sharpness or smoothness of the scoring in the log-probability space.

Table 8. Temperature and sampling parameters for Resampling-based CFG in different models.

Model	T_π	T_σ	s_c	s_u	$T_{\pi,v}$	$T_{\sigma,v}$	T_s	CFG
FARMER 1.1B (patch 16)	1.0	0.9	5	5	0.2	0.9	1.1	2.5
FARMER 1.1B (patch 8)	1.0	1.0	5	5	0.2	0.9	1.1	2.0
FARMER 1.9B (patch 16)	1.0	0.9	5	5	0.2	0.9	1.1	3.5
FARMER 1.9B (patch 8)	1.0	1.0	5	5	0.1	1.0	1.1	1.5

Resample stage. Finally, the normalized weights $\omega_{i,j}$ define a categorical distribution. To further modulate selection sharpness, we introduce a *resampling temperature* T_s applied uniformly to all log-weights before normalization:

$$p_{\text{final}}(z_{i,j}) \propto \exp(\log \omega_{i,j} * T_s).$$

A higher T_s emphasizes high-weight proposals, whereas a lower T_s encourages diversity.

Summary table of parameter choices. Table 8 summarizes the temperature and sampling configurations used for different model variants evaluated in this work.

Algorithm 1 Resampling-based CFG method

Require: AR model P_θ and AF model F_θ
Require: Guidance scale w
for $i \in [0, ..., N+1]$ **do** ▷ Sampel tokens
  `# step1:Propose candidates`
  $G_{c,i} = P_\theta(z^u_{<i}; c)$ ▷ Predict GMM$_c$
  $G_{u,i} = P_\theta(z^u_{<i}; \emptyset)$ ▷ Predict GMM$_u$
  $z_{i,j} \sim G_{c,i},\ j \in [0, .., s]$ ▷ Sample from $p_c(z)$
  $z_{i,j} \sim G_{u,i},\ j \in [s+1, .., s+s']$ ▷ from $p_u(z)$
  `# step2:Weigh candidates`
  if $j \in [0, ..., s]$ **then** ▷ Calculate weights
    $\pi_j = w \cdot (\log G_{c,i}(z_{i,j}) - \log G_{u,i}(z_{i,j}))$
  else
    $\pi_j = (w+1) \cdot (\log G_{c,i}(z_{i,j}) - \log G_{u,i}(z_{i,j}))$
  end if
  $\pi_1, ..., \pi_{s+s'} =$ **logsoftmax**$(\pi_1, ..., \pi_{s+s'})$
  `# step3:Resample from candidates`
  if $i \leq N$ **then** ▷ For informative tokens
    $idx \sim$ **Categorical**$(\pi_1, ..., \pi_{s+s'})$
    $z^u_i := z_{i,idx}$
  else ▷ For redundant tokens
    for $k \in [0, ..., N]$ **do** ▷ s, s' larger in here
      $idx_k \sim$ **Categorical**$(\pi_1, ..., \pi_{s+s'})$
      $z^d_k := z_{idx_k}$
    end for
  end if
end for
$z =$ **concat**$[[z^u_1, z^d_i], ..., [z^u_N, z^d_N]]$
$x = F^{-1}_\theta(z)$ ▷ Reverse to data

8.3. Algorithm Details

We present the details of our resampling-based CFG method and one-step sampling distillation method in Algorithm 1 and Algorithm 2, respectively.

Algorithm 2 One-step sampling distillation

Require: Trained teacher AF (frozen) $\mathbf{F}_\eta = f_{\eta_n} \circ f_{\eta_{n-1}} \circ \cdots \circ f_{\eta_1}$
Require: Data set $\mathcal{D}$
Require: Student AF $\mathbf{G}_\theta = g_{\theta_1} \circ g_{\theta_2} \circ \cdots \circ g_{\theta_n}$
for m epochs **do**
  for K iterations **do**
    $x \sim \mathcal{D}$
    $x = \textbf{Patchify}(x)$
    $Z^0 := x$
    for n Teacher AF blocks **do**
      $Z^t, \_ = f_{\eta_t}(Z^{t-1})$ ▷ Teacher Transform
    end for
    $Z := Z_n$
    $\epsilon \sim N(0, I)$ ▷ Sample noise
    $s \sim U[0, 0.3]$ ▷ Sample scale
    $\tilde{Z} = Z + s \cdot \epsilon$ ▷ Add noise to latent
    $\tilde{Z}^n := \tilde{Z}$
    `# Distill a one-step student`
    `# reverse path from the teacher`
    `# forward path`
    for n Student AF reversed blocks **do**
      $\tilde{Z}^{t-1}, \_ = g_{\theta_t}(\tilde{Z}^t)$ ▷ Student Transform
      $L_{\theta_t} = \|\tilde{Z}^{t-1} - Z^{t-1}\|_2^2$ ▷ MSE loss
    end for
    $L_\theta = \frac{1}{n} \sum_t^n L_{\theta_t}$
    $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$
  end for
end for

9. Discussions

9.1. FARMER reduces to AF when mixtures $K = 1$

When the number of components (K) in the Gaussian Mixture Model (GMM) predicted by FARMER is set to one ($K = 1$), FARMER reduces to an Autoregressive Flow (AF). In this case, each token z_i in the sequence is modeled by a conditional Gaussian distribution, where the mean and variance are functions of the preceding tokens $z_{<i}$. The optimization objective for each token becomes:

$$\log p(z_i|z_{<i}) = \log \left(\mathcal{N}(z_i; \mu(z_{<i}), \sigma^2(z_{<i}))\right) \tag{11}$$

This can be further expressed as:

$$\log p(z_i|z_{<i}) = \log \left[\mathcal{N}\left(\frac{z_i - \mu(z_{<i})}{\sigma(z_{<i})}; 0, I_d\right) \left|\frac{\partial \left[z_i - \mu(z_{<i})\right]/\sigma(z_{<i})}{\partial z_i}\right|\right], \tag{12}$$

where $\mathcal{N}(\cdot; 0, I_d)$ denotes the standard normal density, and the second term inside the log represents the change of variables formula (the volume correction by the Jacobian determinant).

Expanding the log yields two components:

$$\log p(z_i|z_{<i}) = \log \mathcal{N}\left(\frac{z_i - \mu(z_{<i})}{\sigma(z_{<i})}; 0, I_d\right) + \log \left|\frac{1}{\sigma(z_{<i})}\right|, \tag{13}$$

z_i is transformed into a new token $\frac{z_i - \mu(z_{<i})}{\sigma(z_{<i})}$ by the predicted results $(\mu(z_{<i}), \sigma(z_{<i}))$ of the AR model conditioned on preceding tokens $z_{<i}$, and this transformation is invertible; the first term is the log-likelihood of the new token z_i' under the standard Gaussian distribution, and the second term is the log-determinant of the Jacobian of the affine transformation. Thus, the AR model can be considered as the last block of Autoregressive Flows.

This confirms that when $K = 1$, FARMER reduces to an Autoregressive Flow.

Figure 10. **Qualitative Results**. Images generated by FARMER on ImageNet 256x256.

Figure 11. **Additional Qualitative Results**. Images generated by FARMER on ImageNet 256x256.

Figure 12. **Additional Qualitative Results**. Images generated by FARMER on ImageNet 256x256.

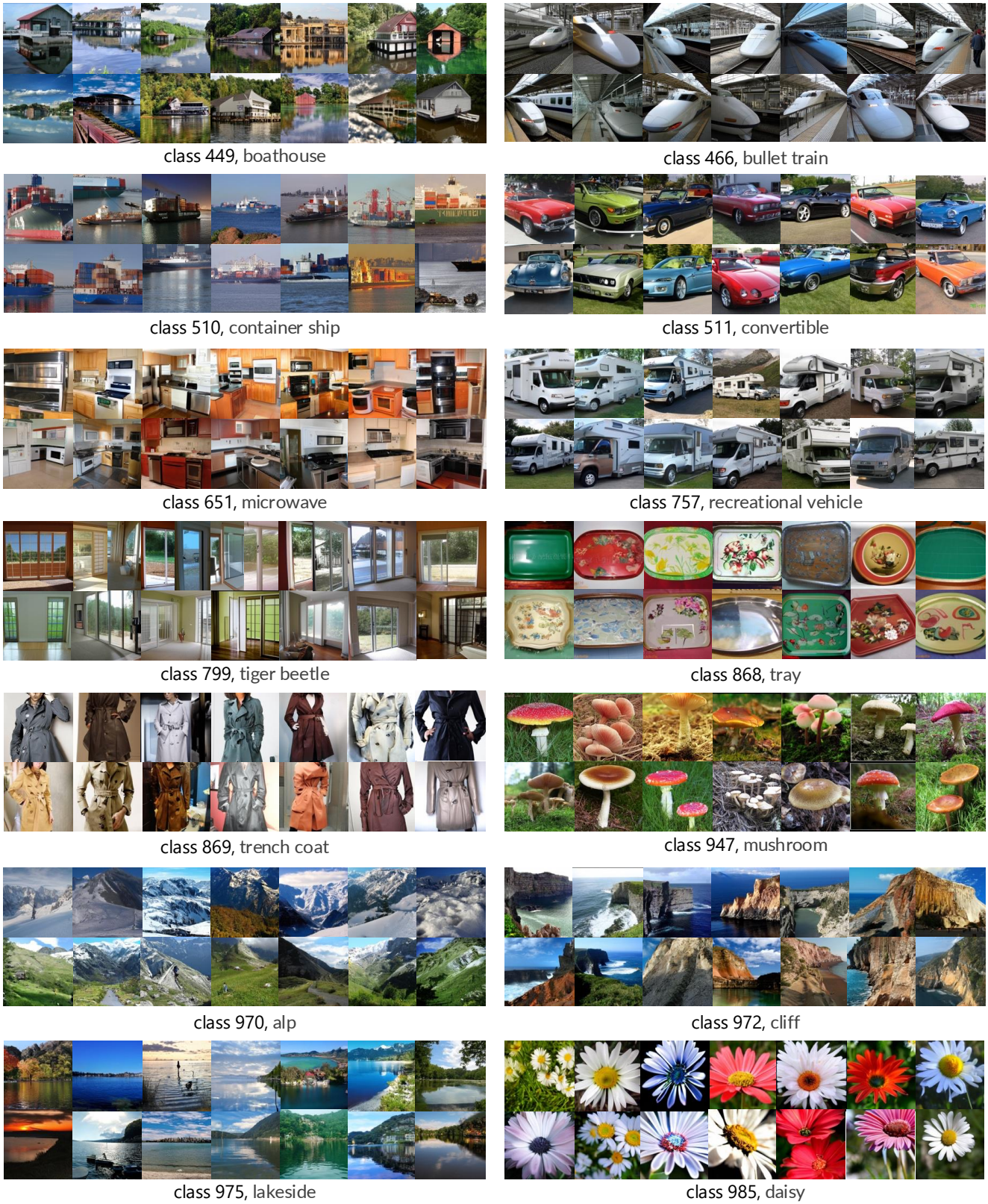

Figure 13. **Additional Qualitative Results**. Images generated by FARMER on ImageNet 256x256.