Visual Prompt Tuning for Generative Transfer Learning

Kihyuk Sohn, Yuan Hao, José Lezama, Luisa Polania,
Huiwen Chang, Han Zhang, Irfan Essa, Lu Jiang
Google Research

Abstract

Transferring knowledge from an image synthesis model trained on a large dataset is a promising direction for learning generative image models from various domains efficiently. While previous works have studied GAN models, we present a recipe for learning vision transformers by generative knowledge transfer. We base our framework on state-of-the-art generative vision transformers that represent an image as a sequence of visual tokens to the autoregressive or non-autoregressive transformers. To adapt to a new domain, we employ prompt tuning, which prepends learnable tokens called prompt to the image token sequence, and introduce a new prompt design for our task. We study on a variety of visual domains, including visual task adaptation benchmark [71], with varying amount of training images, and show effectiveness of knowledge transfer and a significantly better image generation quality over existing works.

1. Introduction

Image synthesis has achieved tremendous progress recently with the improvement of deep generative models [2, 11, 18, 60, 62]. The goal of image synthesis is to generate diverse and plausible scenes resembling the training images. A good image synthesis system can capture the appearance of objects and model their interactions to generalize and create novel scenes. However, the generalization ability is usually determined by the amount of training images. Without sufficient data, the synthesis results are often unsatisfactory.

Transfer learning, a cornerstone invention in deep learning, has been proving its indispensable role across a broad array of computer vision tasks, including classification [31], object detection [16, 17], image segmentation [20, 21], *etc*. However, transfer learning is not yet a *de facto* technique for image synthesis. While recent efforts have shown success in transferring knowledge from pretrained Generative Adversarial Network (GAN) models [42, 53, 64, 68], these demonstrations are limited by narrow visual domains, *e.g*., faces or cars [42, 68], as illustrated in Fig. 1, or requiring a non-trivial amount of training data [53, 64] to transfer to an off-manifold distribution.

In fact, some recent works [56, 73] have found that, even when the training data is limited in quantity, learning GANs from scratch with advanced techniques outperforms GAN transfer approaches, implying that the transfer learning may not even be necessary for generative modeling. Such observation is in direct contrast to the essential role of transfer learning for discriminative models,[1] which suggests transfer learning for image synthesis remains under-exploited.

In this work, we approach the transfer learning for image synthesis using generative vision transformers, an emerging class of image synthesis models, such as DALL·E [47], Taming Transformer [14], MaskGIT [6], CogView [12], NÜWA [67], or Parti [70], that excel in several image synthesis tasks. We closely follow the recipe of transfer learning for image classification [31], in which a source model is first trained on a large dataset (*e.g*., ImageNet) and then transferred to a diverse collection of downstream tasks, except in our setting the input and output are reversed and the model generates images from a class label. Our study employs the visual task adaptation benchmark (or VTAB) [71], a standard and challenging benchmark for studying transfer learning. VTAB consists of 19 visual recognition tasks and compiles images from diverse and distinctly different visual domains, such as natural (*e.g*., flowers, scenes), specialized (*e.g*., satellite, medical), or structured (*e.g*., road scenes).

We present a transfer learning framework using *prompt tuning* [34, 36]. While the technique has been used for transfer learning of discriminative models for vision tasks [1, 26], to our knowledge, this work appears to be the first to adopt prompt tuning for transfer learning of image synthesis. Moreover, we propose two technical innovations. First, a parameter efficient design of prompt token generator that admits condition variables (*e.g*., class, instance), a key for controllable image synthesis yet often neglected in prompt tuning for discriminative transfer [26, 34]. Second, a marquee header prompt that engineers (*e.g*., composes and interpolates) learned prompts to enhance generation diversity.

We conduct a large-scale empirical study to understand the mechanics of generative transfer learning for autoregres-

[1] Here discriminative models refer to a board of machine learning models that directly model the conditional distribution of the target variables.

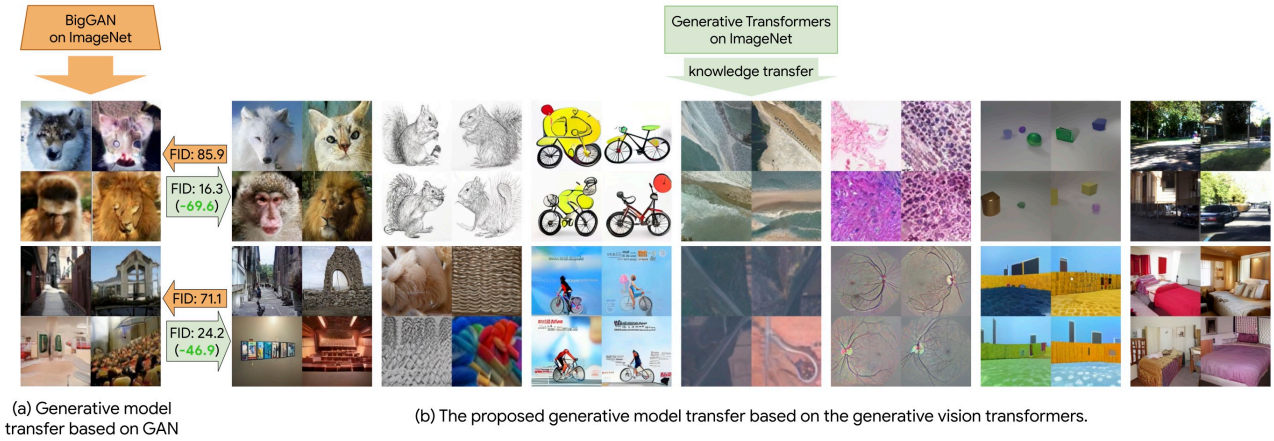


Figure 1. Image synthesis by knowledge transfer. Unlike previous works using GANs as source model and test transfer on relatively narrow visual domains, we transfer knowledge of generative vision transformers [6,14] to a comprehensive list of visual domains, including natural (*e.g.*, scene, flower), specialized (*e.g.*, satellite, medical), and structured (*e.g.*, road scenes, synthetic, infograph, sketch), as defined by the visual task adaptation benchmark [71], with a few training images (*e.g.*, as low as 2 images per class).

sive [14, 47] and non-autoregressive [6] generative transformers. To this end, we show that generative vision transformers with prompt tuning outperforms the prior state-of-the-art held by GANs [53,64] through a vast margin. Moreover, in contrast to prior works [53, 64] limited to show transfer to a few visual domains, we show the efficacy of knowledge transfer from pretrained ImageNet models to 19 downstream tasks of diverse visual distributions and varying amounts of training data in VTAB. Fig. 1 compares visual domains, showing the great expansion on the varieties of downstream tasks to what is achieved by previous works. On the on-manifold distributions on which previous studies mainly focused, our method slashes the prior state-of-the-art in FID from 71 to 24 on Places [74] and 86 to 16 on Animal Face [54] datasets. Moreover, the proposed method is used to demonstrate the few-shot generative transfer capabilities (Sec. 4.2), showing extreme data efficiency while being able to generate images that are realistic and diverse, while following the target distribution.

In summary, our contributions are as follows:

- We present a generative visual transfer learning framework for vision transformers with prompt tuning [34], proposing a novel prompt token generator design and a prompt engineering method for image synthesis.
- We conduct a large-scale empirical study for generative transfer learning to validate our method on a variety of visual domains (*e.g.*, VTAB [71]) and scenarios (*e.g.*, few-shot). To this end, we show state-of-the-art image synthesis performance.
- To our knowledge, we are the first to employ prompt tuning for transfer learning of image synthesis, and provide one-of-the-first substantial empirical evidence on the necessity of knowledge transfer for data and compute efficient generative image modeling using the standard visual transfer learning benchmark.

2. Preliminary

2.1. Generative Vision Transformers

This paper uses generative vision transformers to denote the vision transformer models for image synthesis. Generally, there are two types of generative vision transformers: *AutoRegressive (AR)* and *Non-AutoRegressive (NAR)* transformers, both consisting of two stages [14,47]: image quantization and decoding. The first stage is the same between the two types of models in which the image is quantized into a grid of discrete tokens by a Vector-Quantized (VQ) auto-encoder [14, 48, 60, 69]. The VQ encoder quantizes image patches into integer indices (or tokens) in a codebook. The 2D image is then flattened into a 1D sequence to which a special token indicating its class label is prepended.

AR and NAR transformers differ in the second stage of decoding. AR transformers [7], including DALL·E [47], Taming Transformer [14], NÜWA [67], CogView [12], and Parti [70], are inspired by the AR language model [3, 39]. They learn an AR decoder on the flattened token sequence to generate image tokens sequentially based on the previously generated tokens. As illustrated in Fig. 2, the generation follows a raster scan ordering, generating tokens from left to right, line-by-line. Finally, the generated tokens are mapped to the pixel space using the VQ decoder.

On the other hand, NAR transformers [15,19,33], which are originally proposed for machine translation, are recently extended to improve the AR image decoding [6, 35, 72]. Unlike their AR counterpart, NAR transformers (*e.g.*, MaskGIT [6], Token-Critic [35], BLT [32]) are bidirectional and are trained on the masked modeling proxy task of BERT [10]. During inference, the model adopts a non-

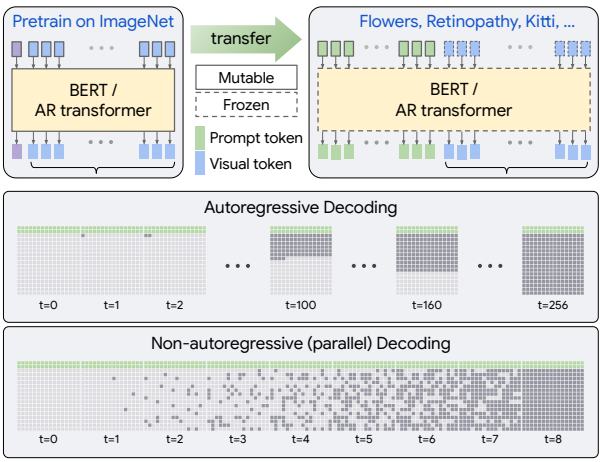


Figure 2. Our method transfers knowledge from generative vision transformers (*e.g.*, autoregressive [14] or non-autoregressive [6]) trained on a large dataset to various visual domains by prepending learnable prompt tokens (green) to visual tokens (blue).

autoregressive decoding method to synthesize an image in a few steps [6, 19, 32, 35]. As shown in Fig. 2, the NAR transformer starts from a blank canvas with all tokens masked out, and generate an image in 8 steps or so. In each step, it predicts all tokens in parallel while retaining ones with the highest prediction scores. The remaining tokens are masked out and will be predicted in the next iteration until all tokens are generated. NAR transformers [6, 35] show faster inference than their AR counterparts (*e.g.*, [14]) while offering on-par or superior fidelity and diversity.

2.2. Prompt Tuning

Prompt tuning [34, 36] has been introduced recently in natural language processing as a way of efficiently adapting pretrained large language models to downstream tasks. Here, prompt is a sequence of additional tokens prepended to a token sequence. In prompt engineering [4], their values are often chosen by heuristic. On the other hand, in prompt tuning [34, 36], tokens are parameterized by learnable parameters and their parameters are updated via a gradient descent to adopt transformers to the downstream tasks.

Due to its simplicity and as transformers getting popular, prompt tuning has been also applied to some vision tasks for knowledge transfer, *e.g.*, image classification [1, 26], detection and segmentation [41]. To our knowledge, we appear to be the first to use prompt tuning for image synthesis.

3. Visual Prompt for Generative Transfer

Our goal is to design a transfer learning framework for image synthesis using vision transformers. Starting from a generative vision transformer pretrained on a large dataset (*e.g.*, ImageNet), we discuss a method to adapt transformers on various target domains (*e.g.*, VTAB). Sec. 3.1 presents a visual prompt tuning for AR and NAR transformers. Then, in Sec. 3.2, we propose a novel prompt, named marquee header prompt, tailored to NAR transformers to trade-off generation fidelity and diversity.

3.1. Building and Learning Visual Prompt

Fig. 2 overviews the proposed generative transfer learning framework. We aim at transferring a generative prior, parameterized by generative vision transformers, while utilizing the same VQ encoder and decoder trained from the large source dataset. We employ a prompt tuning [26,34,36] that uses a sequence of learnable tokens (*e.g.*, green blocks with a solid line in Fig. 2) to adapt to target distributions, while leaving transformer parameters frozen. In the following sections, we discuss how to learn (Sec. 3.1.1) a prompt token generator designed for a conditional image generation (Sec. 3.1.2) and use them for image synthesis (Sec. 3.2).

3.1.1 Learning Visual Prompt

A sequence of prompt tokens is prepended to visual tokens to guide the pretrained transformer models to the target distribution. Prompt tuning, learning parameters of token generator, is done by gradient descent with respective loss functions, while fixing parameters of pretrained transformers. To be specific, let $\mathcal{Z} = \{z_i\}_{i=1}^{H \times W}$ be a sequence of visual tokens (*i.e.*, an output of VQ encoder followed by the vectorization) and $\mathcal{P}_\phi = \{p_{s;\phi}\}_{s=1}^{S}$ be a sequence of prompt tokens. For AR transformer, the loss is given as follows:

$$\mathcal{L}_{\text{AR}} = \mathbb{E}_{x \sim P_{\mathcal{X}}} \big[-\log P_\theta(\mathcal{Z}|\mathcal{P}_\phi) \big] \tag{1}$$

$$P_\theta(\mathcal{Z}|\mathcal{P}_\phi) = \prod_{i=1}^{H \times W} P_\theta(z_i|z_{<i}, \mathcal{P}_\phi) \tag{2}$$

For NAR transformer, we follow the loss of MaskGIT [6]:

$$\mathcal{L}_{\text{NAR}} = \mathbb{E}_{x \sim P_{\mathcal{X}}, M \sim P_{\mathcal{M}}} \big[-\log P_\theta(\mathcal{Z}_M|\mathcal{Z}_{\overline{M}}, \mathcal{P}_\phi) \big] \tag{3}$$

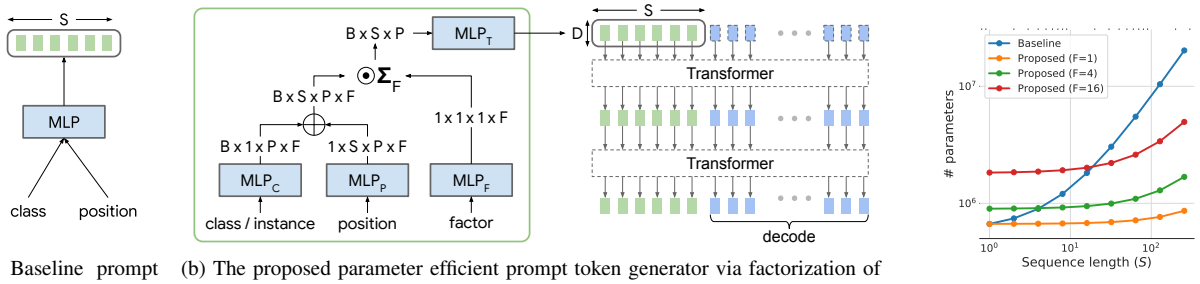
$$P_\theta(\mathcal{Z}_M|\mathcal{Z}_{\overline{M}}, \mathcal{P}_\phi) = \prod_{i \in M} P_\theta(z_i|\mathcal{Z}_{\overline{M}}, \mathcal{P}_\phi) \tag{4}$$

where $M \subset \{1, ..., H \times W\}$ is a set of visual token indices sampled from a masking schedule distribution $P_{\mathcal{M}}$, $\overline{M}$ is its complement, and $\mathcal{Z}_M = \{z_i\}_{i \in M}$. Prompt tuning proceeds by minimizing respective loss functions with respect to the prompt parameters ϕ while fixing transformer parameters θ:

$$\phi^* = \arg\min_\phi \mathcal{L}_{\text{AR/NAR}} \tag{5}$$

While our focus is at the prompt tuning due to its effectiveness and compute-efficiency for large source models, we note that the proposed learning framework is amenable with other transfer learning methods, such as adapter [25] or fine-tuning [31], with learnable prompts, as shown in Sec. 5.4.

After prompt tuning, we generate visual tokens for image synthesis by iterative decoding. For AR transformer,



(a) Baseline prompt token generators of length S conditioned on class.

(b) The proposed parameter efficient prompt token generator via factorization of class / instance and position. $\oplus$ is an element-wise sum, $\odot$ is an element-wise product, Σ_F is a sum over F dimension. S: sequence length, B: batch size, P: feature dimension, D: token dimension.

(c) Number of parameters with respect to the sequence length and different number of factors F.

Figure 3. Prompt token generators and their use in transformer. (a) a straightforward extension of baseline prompt token generators [26, 34, 36] with a class condition. When using an MLP with a single dense layer of P units, the number of trainable parameters is $P{\cdot}(C{\cdot}S{+}D)$. (b) The proposed parameter efficient prompt token generators that factorizes data dependent conditions (e.g., class, instance) and token position. Under a similar design choice as baseline models, the number of trainable parameters is $P{\cdot}(F{\cdot}(C{+}S){+}D)$, which could be significantly fewer when $F{\ll}\min(C,S)$. (c) Number of parameters for prompt token generators with respect to the sequence length (S), while setting $P{=}768$, $D{=}768$, and $C{=}100$ with different number of factors F.

1: **for** $i \leftarrow 1$ to $H \times W$ **do**
2: $\quad \hat{z}_i \sim P_\theta(z_i|\hat{z}_{<i}, \mathcal{P}_\phi)$
3: **end for**

For NAR model, scheduled parallel decoding [6] is used:

Require: $\overline{M} = \{\}, T, \{n_1, ..., n_T\}, \sum_{t=1}^{T} n_t = H \times W$
1: **for** $t \leftarrow 1$ to T **do**
2: $\quad \hat{z}_i \sim P_\theta(z_i|\widehat{\mathcal{Z}}_{\overline{M}}, \mathcal{P}_\phi), \forall i \in M$
3: $\quad \overline{M} \leftarrow \overline{M} \cup \{\arg\text{topk}_{i \in M}(P_\theta(z_i|\widehat{\mathcal{Z}}_{\overline{M}}, \mathcal{P}_\phi), k = n_t)\}$
4: **end for**

where $\{n_1, ..., n_T\}$ is a masking schedule that decides the number of tokens to decode at each decoding step. We refer to [6] for details on decoding for NAR transformer. Illustrations of decoding steps for both models are in Fig. 2.

3.1.2 Prompt Token Generator Design

For discriminative transfer learning, prompts are designed without condition variables [26]. For generation, it is beneficial to have condition variables (*e.g.*, class, attribute) for better control in generation. We accomplish this with rather a straightforward extension of existing prompt designs using a class-condition, $\mathcal{P}_\phi(c)$, as in Fig. 3a.

One caveat of the baseline token generator design is that the number of learnable parameters increases as the product of three factors: the number of classes C, the prompt sequence length S and the feature dimension P. For example, when using a prompt of length S=128, hidden P=768 and embedding dimension D=768, the token generator would introduce 10.4M parameters for C=100 class conditions, as in Fig. 3c. The bottleneck occurs at the 3d weight tensor of size $C{\times}S{\times}P$. To make it parameter efficient, we propose a factorized token generator, as in Fig. 3b. Specifically, we encode class and sequence position index via MLP_C and MLP_P with F factors, respectively. The MLP outputs are element-wise summed, multiplied by an 1d factor vector from MLP_F, and reduced along the factor dimension. The output is then fed to MLP_T to produce a prompt of length S. As in Fig. 3c, the number of parameters of the proposed architecture is greatly reduced, requiring only 0.76M parameters, down from 10.4M, for a prompt of length 128 when $F{=}1$.[2] An implementation of the proposed token generator in Flax [22] is in Fig. 13 of Appendix. We empirically find that $F{=}1$ is sufficient for NAR transformers, demonstrating extreme parameter efficiency. For AR transformers, we need extra capacity and use $F{=}16$.

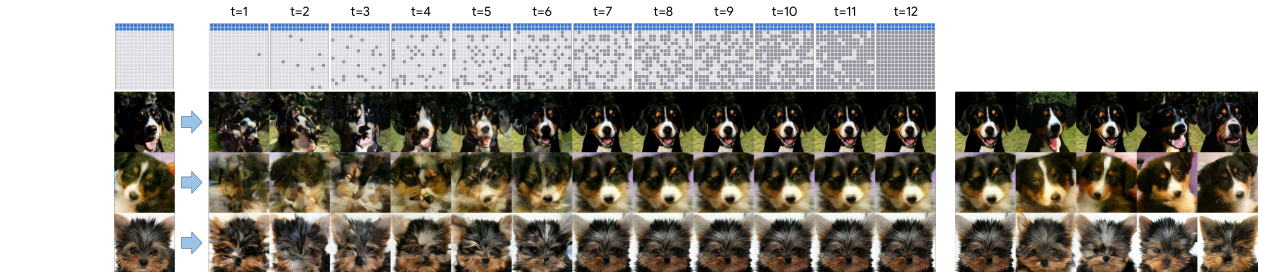
Moreover, we build a new type of prompt tokens conditioned on individual data instances, inspired by the instance-conditioned GAN [5]. We assign each data a unique index and map it into a distinct embedding via MLP_C. When both class label and instance index are used, instance index is simply treated as an extra class, indexed from C. To train the model, we sample between class label and instance index. As we explain below in Sec. 3.2, instance conditioned prompts add more fine-grained control on generation.

3.2. Engineering Learned Prompts

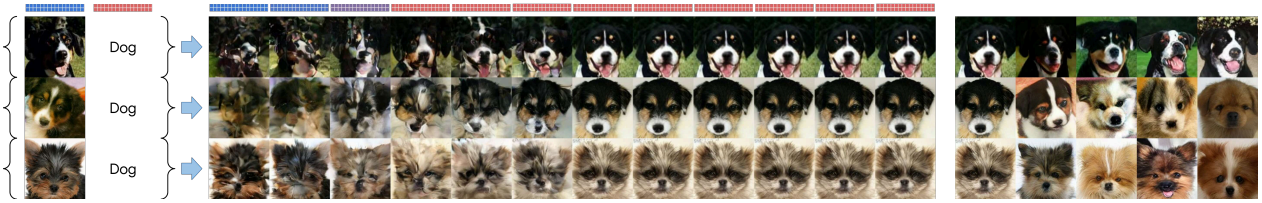
An interesting aspect of generative transformers in contrast to GANs is their iterative decoding. For example, as illustrated in Fig. 2, AR transformers [14] decode tokens sequentially given previously decoded tokens, and NAR transformers [6] use a scheduled parallel decoding.

Given the wealth of learned prompts conditioned on the class and instance proposed in Sec. 3.1, we propose a novel prompt engineering strategy, a "Marquee Header" prompt, of the iterative transformer decoding, for enhancing the generation diversity. The idea is simple – similarly to the latent

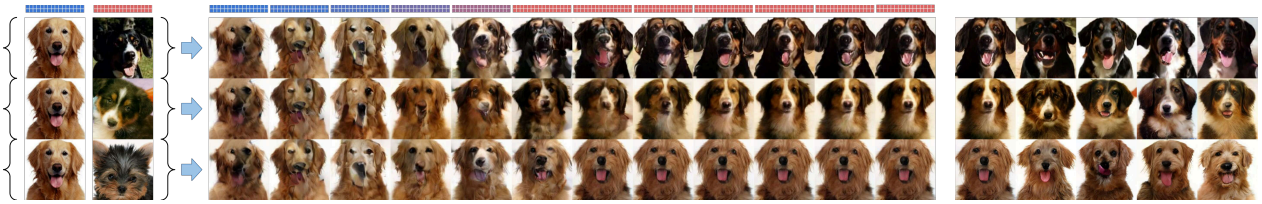
[2]The proposed factorization can be extended to incorporate the "depth" position of deep visual prompt [26] to reduce the number of parameters.



(a) Image synthesis using instance-conditioned prompts.



(b) Image synthesis using a marquee header prompt between instance (blue) and class (red) conditioned prompts.



(c) Image synthesis using a marquee header prompt between instance-conditioned prompts (blue and red).

Figure 4. Iterative decoding of NAR transformers. (4a) instance prompts generate images of high-fidelity but with low diversity. Marquee header prompts enhance generation diversity by interpolating (4b) from instance to class prompts or (4c) between instance prompts.

variable interpolation of GANs, we interpolate the learned prompt representations (*e.g.*, outputs of MLP_C). Yet, due to the iterative decoding, the interpolation between prompts is carried out over multiple decoding steps. This is illustrated in Fig. 4b, where we start the decoding process using instance-conditioned prompts (blue header) but gradually transition to a class-conditioned prompt (red header) over decoding steps. Compared to the generation in Fig. 4a where we use instance-conditioned prompts all along, the proposed prompt engineering strategy enhances the generation diversity while being controlled in that synthesized images follow certain characteristics (*e.g.*, pose, color pattern, hairiness) of reference instances. In addition, it is also plausible to construct a marquee header prompt between instance-conditioned prompts, as in Fig. 4c.

We provide a marquee header prompt formulation:

$$\text{PMT}(t) = (1-w_t)\text{PMT}_1 + w_t\text{PMT}_2 \quad (6)$$

$$w_t = \min\left\{\left(\frac{t-1}{T_\text{cutoff}-1}\right)^2, 1\right\} \quad (7)$$

where $t=1,...,T$ is a decoding step, $T_\text{cutoff} \leq T$ is a cutoff step, and PMT_i is a prompt representation (*e.g.*, an output of MLP_C). The schedule in Eq. (7) makes a smooth transition of prompts from PMT_1 to PMT_2. Note that there could be various marquee header prompt formulations, which we leave their investigations as a future work.

4. Experiments

We evaluate the efficacy of the generative transfer learning on diverse visual domains and varying amounts of training data and compare with existing methods. In Sec. 4.1, we test on visual task adaptation benchmark (VTAB) [71] and demonstrate state-of-the-art image generation performance with knowledge transfer. In Sec. 4.2, we verify our method on diverse few-shot transfer learning tasks.

4.1. Generative Transfer on VTAB

Dataset. Towards developing a generative transfer method generalizable across domains and distributions, we employ the visual task adaptation benchmark (VTAB) [71] – a suite of 19 visual recognition tasks based on 16 datasets. It covers diverse image domains (*e.g.*, natural, structured, and specialized such as medical or satellite imagery) and tasks (*e.g.*, object and scene recognition, distance classification, counting), making it a valuable asset not only for discriminative, but also for generative transfer learning. The dataset information is provided in Appendix B.1.1.

Model		(# tr params)	Mean	C101	Flowers	Pet	DTD	Kitti	SUN	EuroSAT	Resisc
MineGAN [64]		(88M)	151.5	102.4	132.1	130.1	87.4	117.9	77.5	111.5	81.0
cGANTransfer [53]		(105M)	85.1	89.6	61.6	48.6	70.3	48.9	31.1	45.6	50.3
Non-Autoregressive	Prompt ($S=1$)	(0.67M)	53.7	13.5	13.8	11.9	**25.8**	32.3	**7.3**	45.9	28.5
	Prompt ($S=16$)	(0.68M)	39.9	12.7	**13.2**	11.1	26.0	30.0	7.4	**35.8**	24.9
	Prompt ($S=128$)	(0.76M)	**36.4**	**12.9**	13.4	**10.9**	**25.9**	**29.9**	7.7	38.4	**24.8**
	Scratch	(172M)	42.7	72.7	57.2	70.3	66.1	33.8	9.2	39.5	32.0
Autoregressive	Prompt ($S=1$)	(0.86M)	58.4	45.5	28.9	42.4	37.1	66.9	18.9	37.3	35.1
	Prompt ($S=16$)	(0.88M)	45.8	41.4	19.6	36.6	33.4	41.4	16.4	32.6	28.8
	Prompt ($S=256$)	(1.06M)	39.0	39.6	17.3	34.9	32.5	37.1	15.0	29.6	26.7
	Prompt ($S=256, F=16$)	(5.16M)	**36.9**	**27.2**	**14.1**	**27.2**	**30.0**	34.6	**12.8**	26.4	**22.2**
	Scratch	(306M)	39.6	76.0	56.1	52.5	92.7	**31.6**	13.5	**19.4**	29.5

Table 1. FIDs (lower the better) of image generation models on VTAB tasks. The number of trainable parameters (second column) are computed assuming 100 classes. The mean FID over 19 VTAB tasks (third column) and those for dataset with a small to mid-scale training data are reported. Complete results are in Appendix B.1.3. The **best** and the second best results are highlighted in each column.

(a) SUN397 (FID=7.7; NAR + Prompt) (b) DTD (FID=25.9; NAR + Prompt) (c) Resisc (FID=24.8; NAR + Prompt)

(d) SUN397 (FID=12.8; AR + Prompt) (e) DTD (FID=30.0; AR + Prompt) (f) Resisc: (FID=22.2; AR + Prompt)

Figure 5. Class conditional generation using NAR (top; S=128) and AR (bottom; S=256, F=16) transformers with prompt tuning.

Setting. We study class-conditional image generation models on the VTAB (full) tasks. Class-conditional prompts are trained on the "train" split, using the same hyperparameters across tasks as provided in Appendix B.1.2.

We investigate generative transfer of AR and NAR transformers using class-conditional Taming Transformer [14] and MaskGIT [6], respectively, trained on 256×256 images of ImageNet dataset as source models. Both models contain 24 transformer layers, comprised of 306M and 172M model parameters, respectively.

Baselines. We compare our method with GAN-based generative transfer learning methods, including MineGAN [64] and cGANTransfer [53]. Note that both of these algorithms use a BigGAN [2] trained on ImageNet as a source. It is worth noting that the BigGAN model is trained on 128×128 images and its validation FID on ImageNet is 7.4. This is better than that of our pretrained AR transformer (18.7) and almost on par with that of NAR transformer (6.2).

We further compare with generative transformers trained from scratch on VTAB. To highlight the compute efficiency, models are trained with a comparable compute budget (*e.g.*, same number of train epochs) to transfer learning models. Hyperparameters are provided in Appendix B.1.2. We provide more in-depth analysis without compute budget restrictions in Sec. 5.4.

Evaluation. We use Frechet Inception Distance (FID) [24] as a quantitative metric. We generate $20k$ images from each model and compare with images from a respective dataset. We sample $20k$ images if the dataset is larger than $20k$.

Results. We report FIDs of models trained and evaluated

on VTAB tasks in Tab. 1 averaged over 3 runs. Due to limited space, we report results on tasks with small to mid-scale train set in addition to the mean FID over 19 datasets. Complete results are given in Appendix B.1.3. In addition, we provide images generated by various transfer learning methods for thorough visual inspection. See Appendix B.1 for evaluation details and more extensive comparison. We see that prompt tuning is effective for both AR and NAR generative transformers, especially when the number of training images is small (*e.g.*, $\leq 10k$). Between AR and NAR transformers, we find that NAR model transfers better than the AR counterpart. Nevertheless, both generative transformers with class-conditional prompt tuning show significant gain in performance when compared to GAN-based baselines.

We see that the prompt tuning of generative transformers benefits greatly from a long prompt, reducing mean FID from 53.7 to 36.4 by increasing the length from 1 to 128. This is achieved by only adding less than 0.1M parameters, thanks to our parameter-efficient design of the prompt token generator. Nevertheless, this comes at an increased cost at generation time due to increased sequence length. Empirically, we find that using 128 tokens for the prompt increases the overall generation time by 25%, as shown in Tab. 4.

AR transformers also benefit from the longer prompt. On the other hand, AR transformers generally requires prompts with more learnable parameters, which is achieved by increasing the number of factors. The performance is still on par with that achievable with the baseline prompt, while using significantly less number of parameters (5.6M instead of 20.5M), as shown in Sec. 5.3.

In Fig. 5, we show generated images using 128 prompt tokens for NAR transformers and 256 prompt tokens (with $F = 16$) for AR transformers on a few VTAB tasks. More generated images are in Appendix B.1.4. Despite learning less than 0.5% of the transformer parameters, the learned prompts are able to change the generation process of pre-trained generative transformers to follow the target distribution.

4.2. Few-shot Generative Transfer

After validation on VTAB, we delve deeper into a few-shot generative transfer, where the number of training images is further reduced. We limit our study to transfer of an NAR transformer, *i.e.*, MaskGIT [6], but with more comparisons to existing few-shot image generation models, either with [53, 64] or without [56, 73] knowledge transfer.

Dataset. We study few-shot generative transfer learning on Places [74], ImageNet [9], and Animal Face [54]. Following [53, 64], for Places and ImageNet, we select 5 classes[3] and use 500 images per class for training. For Animal Face, we consider two scenarios – following [53], we use 100 images per class for training from 20 classes (denoted as "Animal Face" in Tab. 2); alternatively, following [56, 73], we use all images of dog (389) and cat (160) classes (denoted as "dog face" and "cat face" in Tab. 2) for training.

[3]Cock, Tape player, Broccoli, Fire engine, Harvester for ImageNet, and Alley, Arch, Art gallery, Auditorium, Ballroom for Places.

Dataset (shot)	ImageNet (500)	Places (500)	Animal Face (100)	Dog Face (389)	Cat Face (160)
MineGAN [64]	61.8†	82.3†	–	93.0*	54.5*
cGANTransfer [53]	–	71.1‡	85.9‡	–	–
DiffAug [73]	–	–	–	58.5*	42.4*
LeCam GAN [56]	–	–	–	54.9*	34.2*
Ours (class)	**16.9**	24.2	16.3	65.4	40.2
Ours (instance)	19.6	**19.5**	**13.3**	**26.0**	**31.2**

Table 2. FIDs of image generation models on few-shot benchmark. Numbers with †, ‡, ∗ are from [64], [53], [56], respectively.

Moreover, we test our methods to more challenging off-manifold target tasks on DomainNet [45] Infograph and Clipart (345 classes), and ImageNet sketch (1000 classes) [63] with as low as 2 training images per class.

Setting. We study a class-and-instance conditional generative transfer as in Sec. 3.1.2. Class-and-instance conditional prompts are particularly suitable for few-shot scenarios as there are only a limited number of training images.

Baselines. GAN-based generative transfer learning methods, *e.g.*, MineGAN [64] and cGANTransfer [53], are used as baselines. Moreover, we compare to few-shot image generation models, *e.g.*, DiffAug [73] and LeCam GAN [56].

Evaluation. We report FIDs using $10k$ generated images, except for experiments on dog and cat faces, where we generate $5k$ images following [73]. For Places, ImageNet, and Animal Face, we use an entire training data (*i.e.*, 2500 for Places and ImageNet, 2000 for Animal Face, 389 and 160 for dog and cat faces, respectively) for the reference distribution. We sample $10k$ images for the reference distribution to compute FID for DomainNet and ImageNet sketch.

Results. In Tab. 2, we report FIDs of our proposed method using prompts of $S = 128$. When conditioned on the class, our method improves FIDs upon existing generative transfer learning methods. When comparing with few-shot generation methods on dog and cat face datasets, our method with a class condition slightly under-performs, likely due to that dataset having one class. When conditioned on instances, our models outperform all GAN-based few-shot generation models. We provide visualizations in Appendix B.2.1.

We visualize generated images conditioned on the class by our models in Fig. 6. We show 2 (and only) training data for each class in red boxes. We observe that, though images in these datasets are highly artificial and their distributions are different from the source dataset, our method is able to synthesize images from respective target distributions well. Moreover, as clearly seen from Fig. 6, our models do more

(a) DomainNet Clipart (2 shot; FID=22.4) (b) DomainNet Infograph (2 shot; FID=20.6) (c) ImageNet Sketch (2 shot; FID=14.4)

Figure 6. Class conditional generation of few-shot transfer models. Images in red boxes are two training images of each class.

than simply memorizing the training data.

Data Efficiency. We conduct experiments with less training images to investigate the data efficiency. We train models on 5, 10, 50 and 100 images per class for ImageNet, Places and Animal Face datasets. We use a class-condition for image generation. The same number of images is used for the reference set to make FIDs comparable across settings.

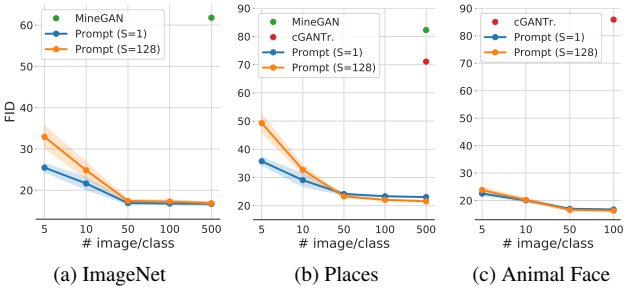
Results are in Fig. 7. Our method shows far superior data efficiency, achieving substantially lower FIDs with only 5 training images per class, to GAN-based transfer learning methods trained with 20 or 100 times more images per class. We find that using long prompts is not favorable when the number of training images is too small (*e.g.*, less than 10 images per class for ImageNet and Places, 50 in total), as models start to overfit to a few images in the train set. When the total number of images is larger than 250, we find that using a long prompt is still beneficial.

Enhancing Generation Diversity via Prompt Engineering. As in Sec. 3.2 and Figs. 4b and 4c, our model offers a way to enhance generation diversity by composing prompts. We report quantitative metrics to support our claim.

We conduct experiments on the dog and cat faces dataset using marquee header prompts with different T_{cutoff} values. For the fidelity metric, we compute the FID. To measure the diversity, we follow [42] and report a intra-cluster pairwise LPIPS distance, where we generate $5k$ samples and map them into one of training images.[4]

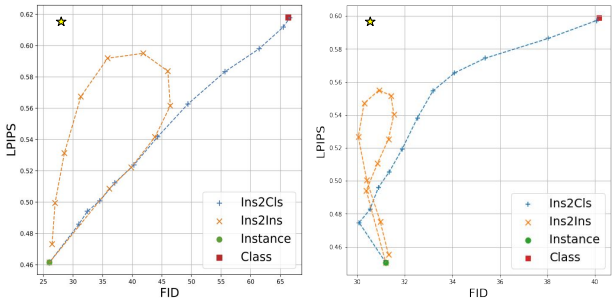
Results are shown in Fig. 8. Ideally, we expect a model with low FID and high intra-cluster LPIPS scores (*e.g.*, yellow star at top-left corner). When generating samples using a class-condition (red square), we generate diverse images, but with relatively poor fidelity. On the other hand, when conditioned on data instances (green dot), we improve the FID by a large margin, but at the cost of reduced diversity. Instance to class Marquee header prompts (blue) allow to control the generation diversity and fidelity. Moreover, instance to instance Marquee header prompts, which interpolates an instance prompt to another randomly selected instance prompt, shows much better tradeoff between fidelity and diversity.

[4]We use a pixel-wise L2 distance for computation efficiency instead of LPIPS distance in [42].



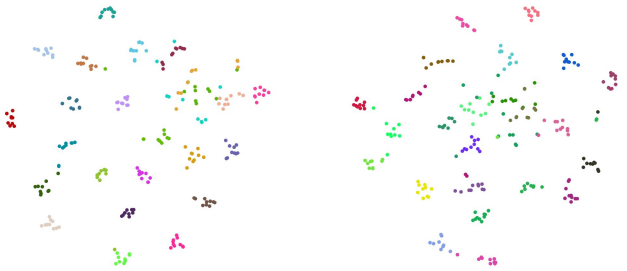
(a) ImageNet (b) Places (c) Animal Face

Figure 7. FIDs for models trained with varying numbers of images per class for class-conditional few-shot generative transfer.



(a) Dog Faces (b) Cat Faces

Figure 8. Marquee header prompt shows clear tradeoff between fidelity (FID) and diversity (LPIPS) when interpolating from instance to class (blue). It shows a better tradeoff when interpolating between instances (orange), achieving low FID and high LPIPS.



(a) $S=1$ (NMI=0.848) (b) $S=128$ (NMI=0.800)

Figure 9. t-SNE plots of instance-conditioned prompt representations on flowers dataset. Points of the same color are from the same class. We also report normalized mutual information (NMI) score by clustering prompt representations using KMeans.

5. Analysis and Discussion

With a successful demonstration of the power of prompt tuning for generative transfer learning, we further study to understand prompt representations (Sec. 5.1, Sec. 5.2) and conduct an ablation study regarding design choices of prompt token generator (Sec. 5.3) and transfer learning (Sec. 5.4).

5.1. What does the Prompt Learn?

To understand what the prompt has learned, we study some properties of learned prompt representations. For this study, we train instance conditioned prompt models on flowers dataset of VTAB, with $S=1$ and 128. Note that no class information is used for training in this experiment.

We draw t-SNE plots [61] of prompts in Fig. 9. Here, we opt to use an output of an MLP_C as a prompt representation instead of a token sequence (*e.g.*, output of an MLP_T) due to its low dimensionality. We see in Fig. 9a that points of the same color (*i.e.*, same class) are grouped together, implying that the prompt representations learn discriminative class information. While we see a similar trend in Fig. 9b, there are clusters crowded with points of various colors. We quantify our observation using a normalized mutual information (NMI) computed by clustering prompts. Clustering is more consistent with the ground-truth class labels with higher NMIs. The model with $S=1$ achieves 0.848 and the one with $S=128$ gets 0.800. Note that these are even better results than the number obtained using an embedding from ImageNet pretrained ResNet-50 [21] (NMI=0.734).

5.2. Adaptation-Diversity Trade-Off

We study prompts with various lengths, but on a *single* image. We show generated images of models with different lengths in Fig. 10. With short prompts, the model produces diverse but less detailed images. On the other hand, a long prompt model generates images of a higher quality, more faithful to the training image, but less diverse. This implies that the short prompt learns concepts, while the long prompt learns fine details of training data. This is in line with our results in Sec. 5.1 where short prompts learn more discriminative information than long prompts.

In Fig. 11, we visualize images generated by models of Sec. 5.1. Compared to images in Fig. 11a whose model is trained with $S=1$, we clearly see in Fig. 11b that the model trained with a long prompt generates images that are more consistent with training instances.

Figure 10. A single training image in red box and those generated by models using prompts of various lengths from 1 to 128.

NAR		# params	Small	Medium	Large	Natural	Struct.	Spec.
S=16	baseline	1.81M	18.6	34.6	89.1	23.8	50.9	41.7
	F=1	0.68M	18.6	36.1	89.5	25.2	51.9	41.5
	F=4	0.95M	18.6	35.5	88.4	24.4	51.5	41.4
	F=16	2.02M	18.5	35.0	86.8	24.3	50.8	40.4
S=128	baseline	10.4M	18.2	30.8	86.4	22.0	46.9	39.9
	F=1	0.76M	18.5	30.6	88.9	22.5	47.1	40.5
	F=4	1.30M	18.1	31.5	88.0	23.3	48.2	38.0
	F=16	3.39M	17.9	30.8	86.5	22.6	47.4	37.7
AR		# params	Small	Medium	Large	Natural	Struct.	Spec.
S=16	baseline	2.02M	30.5	41.9	82.7	28.5	61.9	41.7
	F=1	0.88M	34.5	43.3	83.9	32.3	62.9	42.9
	F=4	1.14M	31.9	42.3	82.7	29.9	62.0	42.0
	F=16	2.21M	31.2	41.9	82.6	28.9	61.9	41.6
S=256	baseline	20.4M	25.7	32.7	71.6	23.7	52.1	35.9
	F=1	1.06M	32.3	33.5	70.5	29.0	49.1	36.4
	F=4	1.88M	31.2	41.9	82.6	28.9	61.9	41.6
	F=16	5.16M	26.6	32.6	69.9	24.5	48.9	34.6

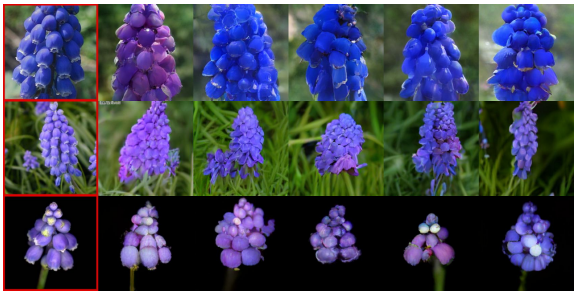
Table 3. Ablation on prompt token generators for (top) NAR and (bottom) AR transformers on VTAB. We report FIDs averaged by different categorizations of tasks.

5.3. Ablation on Prompt Token Generators

One of our technical novelties is the parameter-efficient design of the prompt token generator as in Fig. 3b. We provide in-depth study on different prompt token generators.

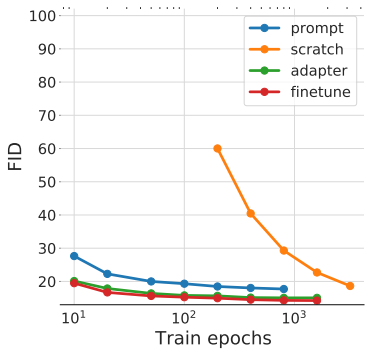
Tab. 3 summarizes results. The key takeaway is that the performance, measured in FIDs, for models using prompts with the proposed factorization closely matches those using the baseline, non-factorized prompts. This is particularly true for NAR transformers. On the other hand, AR transformers still prefers prompt generators with more parameters. Nevertheless, we achieve on par results with the baseline using less than 30% of parameters.

(a) Oxford Flowers, "Grape hyacinth" ($S = 1$)

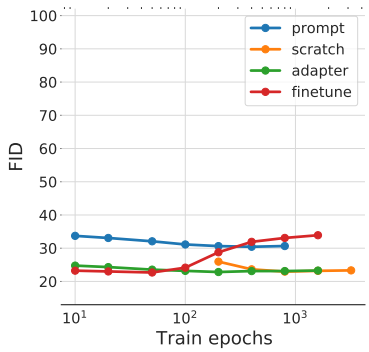


(b) Oxford Flowers, "Grape hyacinth" ($S = 128$)

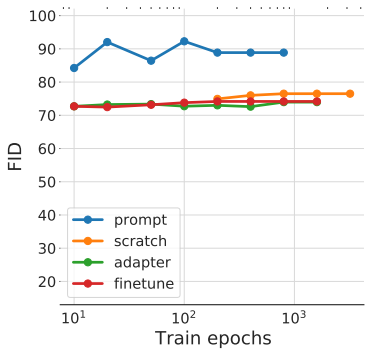
Figure 11. Instance-conditioned generation. For each row, leftmost image in red box is a training image and next five images are generated. When instance conditioned, generated images follow finer-grained details of the reference training image, such as color, shape, or background, beyond class information. Adaptation and diversity could be further controlled by the prompt length.



(a) VTAB small ($<10k$)



(b) VTAB medium ($<100k$)



(c) VTAB large ($>100k$)

Figure 12. FID vs the number of train epochs for various learning methods for transformer-based sequence models. Knowledge transfer is essential for faster convergence when training data is small.

5.4. Beyond Prompt Tuning for Generative Transfer

We have studied applying a prompt tuning to learn generative vision transformers via knowledge transfer. We have seen promising results, *e.g.*, excelling state-of-the-art GAN-based transfer learning methods at generative modeling. We also demonstrate the importance of knowledge transfer for fast and efficient learning of generative models from small training data. Despite the success, prompt tuning is not the only method for learning transformer-based sequence models. For the completeness, we conduct an extended study for various learning methods of generative vision transformers.

To that end, we evaluate adapter tuning and fine-tuning in addition to the prompt tuning and learning from scratch. Adapter tuning [25] introduces learnable adapter modules to each transformer block. Fine-tuning unfreezes pretrained weights and updates them. All models are trained using the same loss (*e.g.*, masked visual token model loss [6] for NAR transformer). As we are interested in class-conditional generative models, we also introduce class-conditional prompts of length 1 that are randomly initialized for adapter tuning and fine-tuning.

For experiments, we vary the number of training epochs from 10 to 3200,[5] as training efficiency is one of the key differentiating factors across various learning strategies. For prompt tuning, we use 128 prompt tokens with a single factor. For adapter tuning, we use 64 hidden units for adapter modules. We report the number of trainable parameters assuming 100 classes, train time per step and generation time comparisons in Tab. 4. Prompt tuning shows the best parameter and train time efficiency, where the number of trainable parameters is less than 0.5% of those of fine-tuning and learning from scratch. On the other hand, due to the longer sequence, it takes more time for generation than those models with a single class token. Adapter tuning, together with a tunable class-conditional prompt, turns out to be a method with a good balance, with relatively few trainable parameters and efficiency at both train and test time.

Fig. 12 compares the generation performance in FID on VTAB. We see that models with a knowledge transfer converge faster than the ones without a transfer. For example, it requires almost 800 epochs for models learned from scratch to reach FIDs of the prompt tuning models trained for 10 epochs for tasks with a small data. Fine-tuning also adapts

[5] We limit the maximum number of training steps to $500K$ to finish model training within a reasonable time window.

	# params	train / step	generation
Prompt tuning ($S=128$)	0.76M	1×	1×
Adapter tuning	5.43M	1.04×	0.84×
Fine-tuning, Scratch	172M	1.67×	0.80×

Table 4. Qualitative comparison (*e.g.*, number of trainable parameters, train and generation time) among various learning strategies based on NAR transformers.

to new data distributions quickly, though it takes more time per step for model training. As in Fig. 12a, for tasks with a small training data, fine-tuning shows the best FIDs. On the other hand, we find that fine-tuning behaves unstable on some datasets (*e.g.*, smallnorb), and the performance diverges as training goes, as in Fig. 12b. Complete FID results are in Tab. 7 of Appendix. To our surprise, learning from scratch performs well even for tasks with a small training data when given sufficient compute resources and time.

Finally, we'd like to note that there is no single method that wins against the rest as each method has its own advantage. For example, for applications where the small number of parameter is critical, prompt tuning should be preferred despite slightly worse generation quality. Also, prompt and adapter tuning are preferred when there are many datasets and tasks as transformer parameters are shared across tasks.

6. Related Work

Transfer learning [43, 55, 66, 75] is a method for improving the performance of downstream tasks using knowledge from the source domain and task. It is shown to be particularly effective when the amount of training data is limited for downstream tasks. Knowledge transfer of deep neural networks has been realized in various forms, such as linear probing [8, 23], fine-tuning [31, 46], or adapter [25, 49, 50]. Recently, prompt tuning [34, 36, 37] has emerged as a powerful tool for transfer learning of transformer-based large language models in NLP. Since the introduction of Vision Transformer [13], such approach has been studied for vision tasks as well [1, 26]. While previous works have shown effectiveness of prompt tuning for discriminative tasks (e.g., classification), we apply the technique for image synthesis.

Generative models have been extensively studied for image synthesis, including variational autoencoder [30, 57, 59], diffusion [11, 51] and autoregressive [44, 58, 62] models. A large volume of progress has been made around the generative adversarial network (GAN) [18] thanks to its ability at synthesizing high-fidelity images [2, 27, 28, 52]. As such, generative knowledge transfer has been studied to transfer knowledge of pretrained GAN models. TransferGAN [65], following a usual practice by fine-tune on the target dataset, has demonstrated transferring knowledge from pretraining improves the performance when training with limited data. Freezing a few layers of discriminator [40] further improves while stabilizing the training process. MineGAN [64] introduces a miner, which projects random noise into the embedding space of the pretrained generator, and trains it with discriminator while fixing generator parameters. cGANTransfer [53] makes explicit transfer of knowledge on classes of the source dataset to new classes. Albeit showing improvement, these methods still require careful training (e.g., early stopping) and have evaluated on a few datasets. In our work, we extensively test methods on a wide variety of visual domains (e.g., VTAB) and show improvement by a large margin over existing GAN-based generative transfer methods.

7. Conclusion

We present a method for learning image generation models from diverse data distributions and varying amount of training data via knowledge transfer from the source model trained on a large dataset. A simple modification on prompt token designs allows to learn a parameter and compute efficient class and instance conditional image generation models of autoregressive and non-autoregressive vision transformers. Comprehensive experimental results of image synthesis are provided across diverse visual domains, tasks, and the number of training images. In addition, we show how to use learned prompts for novel image synthesis in the form of marquee header prompts, which is particularly useful when synthesizing images using generative models learned from a few images.

Acknowledgment

We thank Brian Lester for helpful discussion on prompt tuning, Boqing Gong and David Salesin for their feedback on the manuscript.

References

[1] Hyojin Bahng, Ali Jahanian, Swami Sankaranarayanan, and Phillip Isola. Visual prompting: Modifying pixel space to adapt pre-trained models. *arXiv preprint arXiv:2203.17274*, 2022. 1, 3, 11

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018. 1, 6, 11

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020. 2

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 3

[5] Arantxa Casanova, Marlene Careil, Jakob Verbeek, Michal Drozdzal, and Adriana Romero Soriano. Instance-conditioned gan. *Advances in Neural Information Processing Systems*, 34:27517–27529, 2021. 4

[6] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. *arXiv preprint arXiv:2202.04200*, 2022. 1, 2, 3, 4, 6, 7, 10

[7] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020. 2

[8] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2018. 11

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 7

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[11] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 11

[12] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. Cogview: Mastering text-to-image generation via transformers. *Advances in Neural Information Processing Systems*, 34:19822–19835, 2021. 1, 2

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 11

[14] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021. 1, 2, 3, 4, 6

[15] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *EMNLP-IJCNLP*, 2019. 2

[16] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1

[17] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1, 11

[19] Jiatao Gu and Xiang Kong. Fully non-autoregressive neural machine translation: Tricks of the trade. In *Findings of ACL-IJCNLP*, 2021. 2, 3

[20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 1

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 9

[22] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020. 4, 15

[23] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International conference on machine learning*, pages 4182–4192. PMLR, 2020. 11

[24] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 6

[25] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019. 3, 10, 11, 15

[26] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. *arXiv preprint arXiv:2203.12119*, 2022. 1, 3, 4, 11

[27] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 11

[28] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020. 11

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 15

[30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 11

[31] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pages 491–507. Springer, 2020. 1, 3, 11, 15

[32] Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. Blt: Bidirectional layout transformer for controllable layout generation. *arXiv preprint arXiv:2112.05112*, 2021. 2, 3

[33] Xiang Kong, Zhisong Zhang, and Eduard Hovy. Incorporating a local translation mechanism into non-autoregressive translation. *arXiv preprint arXiv:2011.06132*, 2020. 2

[34] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, 2021. 1, 2, 3, 4, 11

[35] Jose Lezama, Huiwen Chang, Lu Jiang, and Irfan Essa. Improved masked image generation with token-critic. In *ECCV*, 2022. 2, 3

[36] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021. 1, 3, 4, 11

[37] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021. 11

[38] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 15

[39] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010. 2

[40] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze the discriminator: a simple baseline for fine-tuning gans. *arXiv preprint arXiv:2002.10964*, 2020. 11

[41] Xing Nie, Bolin Ni, Jianlong Chang, Gaomeng Meng, Chunlei Huo, Zhaoxiang Zhang, Shiming Xiang, Qi Tian, and Chunhong Pan. Pro-tuning: Unified prompt tuning for vision tasks. *arXiv preprint arXiv:2207.14381*, 2022. 3

[42] Utkarsh Ojha, Yijun Li, Jingwan Lu, Alexei A Efros, Yong Jae Lee, Eli Shechtman, and Richard Zhang. Few-shot image generation via cross-domain correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10743–10752, 2021. 1, 8

[43] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009. 11

[44] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pages 4055–4064. PMLR, 2018. 11

[45] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019. 7

[46] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019. 11

[47] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *ICML*, 2021. 1, 2

[48] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019. 2

[49] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017. 11

[50] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127, 2018. 11

[51] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 11

[52] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. *arXiv preprint arXiv:2202.00273*, 1, 2022. 11

[53] Mohamad Shahbazi, Zhiwu Huang, Danda Pani Paudel, Ajad Chhatkuli, and Luc Van Gool. Efficient conditional gan transfer with knowledge propagation across classes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12167–12176, 2021. 1, 2, 6, 7, 11, 15

[54] Zhangzhang Si and Song-Chun Zhu. Learning hybrid image templates (hit) by information projection. *IEEE Transactions on pattern analysis and machine intelligence*, 34(7):1354–1367, 2011. 2, 7

[55] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018. 11

[56] Hung-Yu Tseng, Lu Jiang, Ce Liu, Ming-Hsuan Yang, and Weilong Yang. Regularizing generative adversarial networks under limited data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7921–7931, 2021. 1, 7

[57] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020. 11

[58] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016. 11

[59] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 11

[60] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *NeurIPS*, 2017. 1, 2

[61] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 9

[62] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016. 1, 11

[63] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. *Advances in Neural Information Processing Systems*, 32, 2019. 7

[64] Yaxing Wang, Abel Gonzalez-Garcia, David Berga, Luis Herranz, Fahad Shahbaz Khan, and Joost van de Weijer. Minegan: effective knowledge transfer from gans to target domains with few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9332–9341, 2020. 1, 2, 6, 7, 11, 15

[65] Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring gans: generating images from limited data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 218–234, 2018. 11

[66] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016. 11

[67] Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Daxin Jiang, and Nan Duan. N\" uwa: Visual synthesis pre-training for neural visual world creation. *arXiv preprint arXiv:2111.12417*, 2021. 1, 2

[68] Ceyuan Yang, Yujun Shen, Zhiyi Zhang, Yinghao Xu, Jiapeng Zhu, Zhirong Wu, and Bolei Zhou. One-shot generative domain adaptation. *arXiv preprint arXiv:2111.09876*, 2021. 1

[69] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. *arXiv preprint arXiv:2110.04627*, 2021. 2

[70] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022. 1, 2

[71] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019. 1, 2, 5

[72] Zhu Zhang, Jianxin Ma, Chang Zhou, Rui Men, Zhikang Li, Ming Ding, Jie Tang, Jingren Zhou, and Hongxia Yang. M6-ufc: Unifying multi-modal controls for conditional image synthesis. *arXiv preprint arXiv:2105.14211*, 2021. 2

[73] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33:7559–7570, 2020. 1, 7

[74] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. *Advances in neural information processing systems*, 27, 2014. 2, 7

[75] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020. 11

```python
import flax.linen as nn
import jax.numpy as jnp

class TokenGenerator(nn.Module):
  n_token: int  # Number of token (S)
  n_class: int  # Number of class (C)
  n_factor: int # Number of factors (F)
  d_embed: int  # Embed dimension (P)
  d_token: int  # Token dimension (D)

  @nn.compact
  def __call__(self, cls_ids: jnp.ndarray):
    MLP_p = nn.Embed(self.n_token, [self.d_embed, self.n_factor])
    MLP_c = nn.Embed(self.n_class, [self.d_embed, self.n_factor])
    MLP_t = nn.Dense(self.d_token)
    MLP_f = nn.Embed(1, self.n_factor)

    pos_ids = jnp.arange(self.n_token)
    factor_ids = jnp.arange(1)[None, None, ...]
    pos_embed = MLP_p(pos_ids[None, ...])  # 1 x S x P x F
    cls_embed = MLP_c(cls_ids[..., None])  # B x 1 x P x F
    fac_embed = MLP_f([None, None, ...])  # 1 x 1 x 1 x F
    embed = (fac_embed * (pos_embed + cls_embed)).sum(-1)
    return MLP_t(nn.LayerNorm(embed))
```

Figure 13. An example code for the token generator in Flax-ish [22] format.

A. Pseudo-code for Token Generator

In Fig. 13 we provide an example code that implements the prompt token generator in Flax [22] format.

B. Comprehensive Experiment Description

B.1. Visual Task Adaptation Benchmark (VTAB)

B.1.1 Dataset Meta Information of Visual Task Adaptation Benchmark

In Tab. 5 we provide a dataset meta information, including the number of class and the number of images in each data split, of VTAB.

B.1.2 Hyperparameters

We provide hyperparameters used in our experiments in Tab. 6. Note that most hyperparameters are shared across datasets, except the number of training epochs. We use Adam optimizer [29] with a cosine learning rate decay [38]. When learning models from scratch, we find that learning rate warm-up is essential. To this end, we use a warm-up for the first two epochs for AR models, and 80 train epochs for NAR transformers.

B.1.3 Experimental Results

We provide complete results in Tab. 7 for autoregressive transformers, non-autoregressive transformers as well as GAN-based generative model transfer learning methods including MineGAN [64] and cGANTransfer [53]. For AR and NAR transformers, we report FIDs for prompt tuning, learning from scratch, as well as different transfer learning techniques including adapter [25] and fine-tuning [31].

B.1.4 Visualization of Generated Images

We visualize images generated by the models trained on each of VTAB tasks from Fig. 14 to Fig. 29.

Dataset	# class	train	val	test	all
Caltech-101	102	2754	306	6084	9144
CIFAR-100	100	45000	5000	10000	60000
SUN397	397	76128	10875	21750	108753
SVHN	10	65931	7326	26032	99289
Flowers102	102	1020	1020	6149	8189
Pet	37	2944	736	3669	7349
DTD	47	1880	1880	1880	5640
EuroSAT	10	16200	5400	5400	27000
Resisc45	45	18900	6300	6300	31500
Patch Camelyon	2	262144	32768	32768	327680
Diabetic Retinopathy	5	35126	10906	42670	88702
Kitti	4	6347	423	711	7481
Smallnorb (azimuth)	18	24300	12150	12150	48600
Smallnorb (elevation)	9	24300	12150	12150	48600
Dsprites (x position)	16	589824	73728	73728	737280
Dsprites (orientation)	16	589824	73728	73728	737280
Clevr (object distance)	6	63000	7000	15000	85000
Clevr (count)	8	63000	7000	15000	85000
DMLab	6	65550	22628	22735	110913
Mean	49.5	102851.2	15332.8	20416.0	138600.0

Table 5. Dataset meta information (*e.g.*, number of images, number of class) for tasks in VTAB.

	AR scratch	AR + Prompt	AR + Adapter	AR + Fine-tune	NAR scratch	NAR + Prompt	NAR + Adapter	NAR + Fine-tune
Learning rate	0.0005	0.001	0.001	0.0005	0.0001	0.001	0.001	0.001 / 0.0001
Batch size	128	256	256	128	128	256	256	128
Weight decay	0.045	0	0	0.045	0.045	0	0	0.045
Warmup epochs	2	0	0	0	80	0	0	0

Table 6. Hyperparameter used for experiments. For NAR + Fine-tune, we use the learning rate of 0.001 for new model parameters (*e.g.*, prompt) while using 0.0001 for pretrained ones (*e.g.*, transformer). The same hyperparameter is used across all datasets and scenarios.

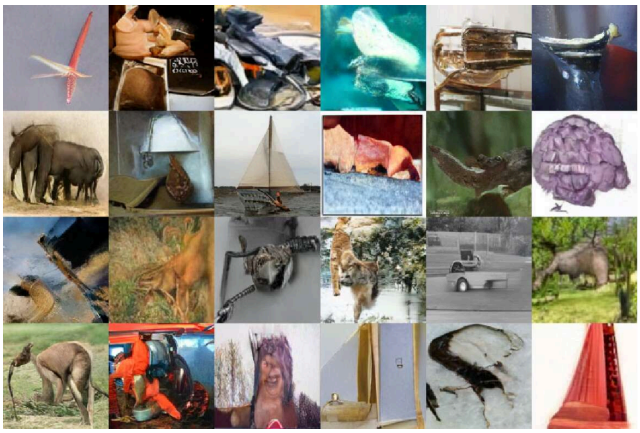
	Models	Caltech101	CIFAR100	SUN397	SVHN	Flower	Pet	DTD	EuroSAT	Resisc45	PC	DR	Kitti
	MineGAN	102.4	82.6	77.5	144.7	132.1	130.1	87.4	111.5	81.0	170.3	192.2	117.9
	cGANTransfer	89.6	31.4	31.1	64.7	61.6	48.6	70.3	45.6	50.3	119.9	149.8	48.9
NAR	Scratch	72.7	24.2	9.2	44.4	57.2	70.3	66.1	39.5	32.0	48.3	25.6	33.8
NAR	Scratch (3200 ep.)	14.5	22.5	7.3	43.5	14.9	8.5	29.2	26.4	24.2	51.1	26.0	26.1
NAR	P (S=1)	13.4	26.9	7.2	83.0	13.8	11.8	25.7	45.9	28.7	107.9	84.2	32.2
NAR	P (S=16)	12.7	25.5	7.3	80.8	13.2	11.0	26.0	35.8	25.1	71.0	34.2	30.0
NAR	P (S=128)	12.9	25.0	7.7	62.3	13.4	10.9	25.9	38.4	24.8	67.4	30.8	29.9
NAR	P (S=128, F=16)	11.8	25.0	7.5	63.4	13.3	11.5	26.0	35.8	24.3	61.4	29.2	27.0
NAR	P† (S=16)	12.4	25.3	7.3	72.5	12.7	11.2	25.4	36.9	23.7	71.7	34.3	31.2
NAR	P† (S=128)	12.2	25.2	7.5	60.4	12.3	11.0	25.7	35.4	24.3	71.7	28.2	29.6
NAR	Adapter	11.3	20.3	6.7	43.7	11.0	6.9	25.1	28.2	19.9	46.4	24.9	24.0
NAR	Fine-tune	11.3	18.2	6.5	43.9	10.2	6.3	24.2	23.1	18.2	48.0	24.4	22.8
AR	Scratch	76.1	27.1	13.5	31.2	56.1	52.5	92.7	19.4	29.5	32.9	37.0	31.6
AR	Scratch (3200 ep.)	30.5	25.8	14.4	27.9	24.3	28.1	45.1	15.5	11.5	32.3	37.7	33.2
AR	P (S=1)	45.4	25.7	18.8	80.4	28.9	42.2	37.1	37.3	35.1	74.9	93.1	66.8
AR	P (S=16)	41.4	22.5	16.4	55.5	19.6	36.6	33.4	32.6	28.8	49.8	60.7	41.3
AR	P (S=256)	39.6	19.8	15.0	44.0	17.3	34.9	32.5	29.6	26.7	44.0	45.4	37.1
AR	P (S=256, F=16)	27.2	17.6	12.8	42.8	14.1	27.2	30.0	26.4	22.2	44.3	45.4	34.6
AR	P† (S=16)	30.9	19.4	13.7	53.7	15.4	30.8	30.8	30.2	25.7	49.0	60.4	39.7
AR	P† (S=256)	24.6	17.5	12.3	43.1	13.7	25.1	29.8	26.7	20.9	43.6	46.1	35.1
AR	Adapter	27.0	16.7	12.6	29.9	11.8	19.1	30.8	22.4	22.0	39.4	37.3	29.0
AR	Fine-tune	17.6	13.2	9.1	27.7	17.7	10.7	35.4	15.1	11.6	30.9	34.5	29.6

	Models	SNorb^A	SNorb^B	Dspr.^{A}	Dspr.^{B}	Clevr^A	Clevr^B	DMLab	Mean	$\leq 10K$	$\leq 100K$	$\geq 100K$	Natural	Special.	Struct.
	MineGAN	160.4	161.1	252.7	285.1	212.1	225.6	152.4	151.5	114.0	145.6	236.0	108.1	138.7	195.9
	cGANTransfer	93.3	90.5	133.7	165.4	109.4	115.0	98.8	85.1	63.8	80.0	139.7	56.8	91.4	106.9
NAR	Scratch	31.4	32.9	87.5	89.0	12.5	13.3	20.6	42.7	60.0	26.0	75.0	49.2	36.4	40.1
NAR	Scratch (3200 ep.)	29.4	30.5	90.1	88.3	13.7	13.5	19.6	30.5	18.6	23.3	76.5	20.1	31.9	38.9
NAR	P (S=1)	58.6	58.7	119.5	121.3	58.5	57.9	64.4	53.7	19.4	52.2	116.2	26.0	66.7	71.4
NAR	P (S=16)	46.1	42.8	98.7	98.8	27.3	28.2	43.4	39.9	18.6	36.1	89.5	25.2	41.5	51.9
NAR	P (S=128)	33.6	35.2	100.9	92.8	21.9	23.6	33.5	36.4	18.6	30.6	87.0	22.6	40.3	46.4
NAR	P (S=128, F=16)	36.0	36.1	98.7	99.3	25.6	24.1	32.0	36.2	17.9	30.8	86.5	22.6	37.7	47.4
NAR	P† (S=16)	44.1	44.7	96.5	99.0	26.0	27.1	38.9	39.0	18.6	34.6	89.1	23.8	41.7	50.9
NAR	P† (S=128)	34.6	38.4	92.2	95.4	24.7	27.5	32.9	36.3	18.2	30.8	86.4	22.0	39.9	46.9
NAR	Adapter	29.2	28.7	85.7	86.9	14.6	15.0	20.0	28.9	15.7	22.9	73.0	17.9	29.9	38.0
NAR	Fine-tune	67.2	51.3	86.5	88.0	20.6	19.7	23.4	32.3	15.0	28.8	74.1	17.2	28.4	47.4
AR	Scratch	23.1	23.4	76.5	76.6	12.3	12.2	27.8	39.6	61.8	23.3	62.0	49.9	29.7	35.4
AR	Scratch (3200 ep.)	23.4	23.3	76.5	75.1	12.1	11.4	25.5	30.2	32.2	20.8	61.3	28.0	24.3	35.1
AR	P (S=1)	62.2	62.0	215.9	214.0	90.6	91.6	69.0	73.2	44.1	60.5	168.3	39.8	60.1	109.0
AR	P (S=16)	52.9	52.6	102.3	99.8	51.0	49.8	53.6	47.4	34.5	43.3	83.9	32.2	42.9	62.9
AR	P (S=256)	42.4	42.3	83.7	83.7	29.5	28.9	45.2	39.0	32.3	33.5	70.5	29.0	36.4	49.1
AR	P (S=256, F=16)	43.4	42.7	83.8	81.6	30.2	29.0	45.9	36.9	26.6	32.6	69.9	24.5	34.6	48.9
AR	P† (S=16)	51.3	52.2	100.3	97.3	49.6	49.0	54.0	44.9	29.5	41.7	82.2	27.8	41.3	61.7
AR	P† (S=256)	43.5	43.5	86.9	84.3	30.4	29.8	45.7	37.0	25.7	32.7	71.6	23.7	34.3	49.9
AR	Adapter	36.0	36.3	77.8	77.9	15.5	14.9	29.6	30.8	23.5	24.8	65.1	21.1	30.3	39.6
AR	Fine-tune	23.2	23.2	76.8	77.2	11.8	11.5	25.6	26.4	22.2	18.8	61.6	18.8	23.0	34.9

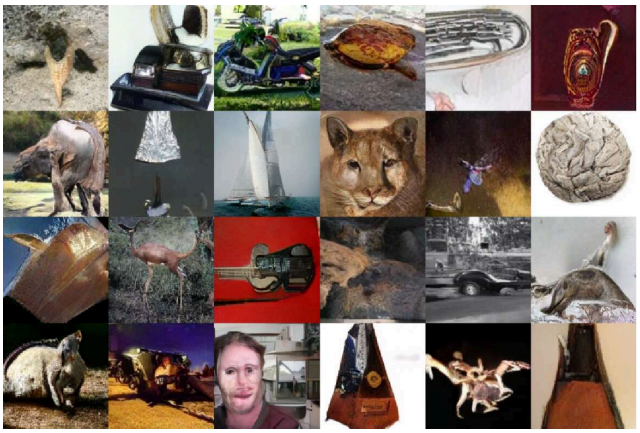
Table 7. FIDs on VTAB tasks tested with various models. We use the "all" set as a reference set for computing FIDs. Unless otherwise stated, all NAR models are trained for 200 epochs and AR models are trained for 400 epochs with the same hyperparameter settings specified in Tab. 6. "P" refers to the prompt tuning with the sequence length S and the number of factors F. "DTD": Describable Textures Dataset, "PC": Patch Camelyon, "DR": Diabetic Retinopathy, "SNorb^A": SmallNorb (azimuth), "SNorb^B": SmallNorb (elevation), "Dspr^A": Dsprites (x position), "Dspr^B": Dsprites (orientation), "Clevr^A": Clevr (object distance), "Clevr^B": Clevr (count).

(a) MineGAN

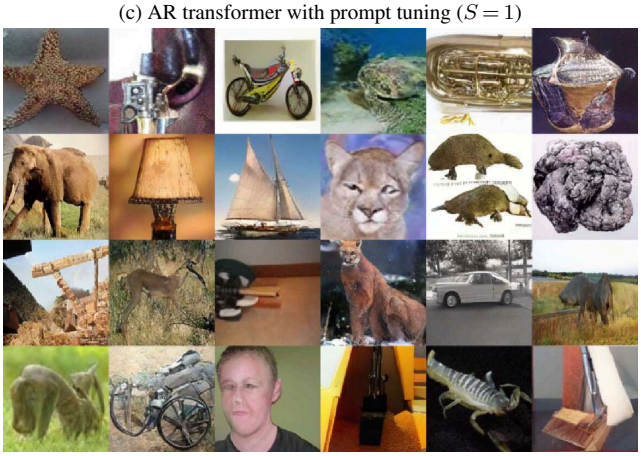
(b) cGANTransfer



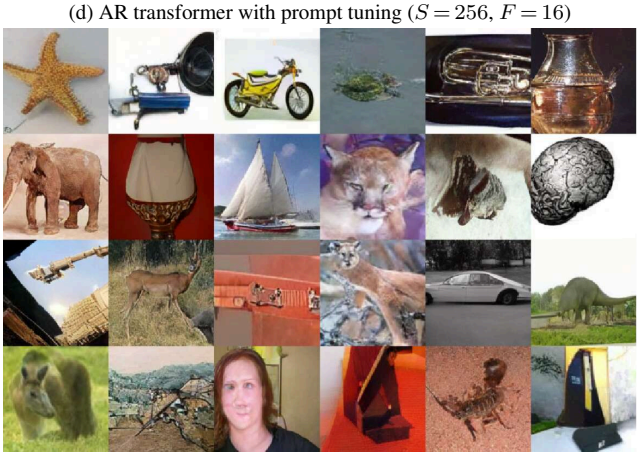
(c) AR transformer with prompt tuning ($S = 1$)



(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)



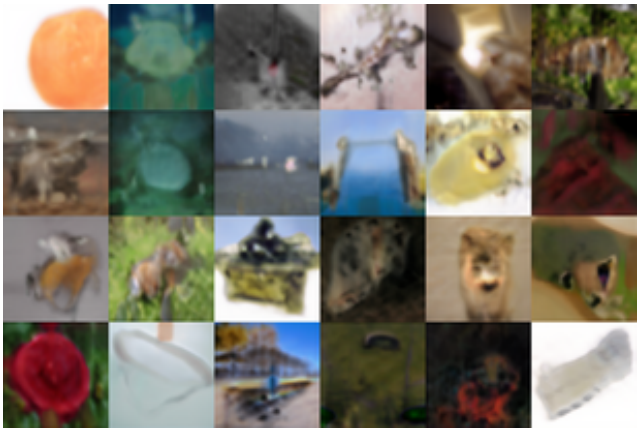
(e) NAR transformer with prompt tuning ($S = 1$)



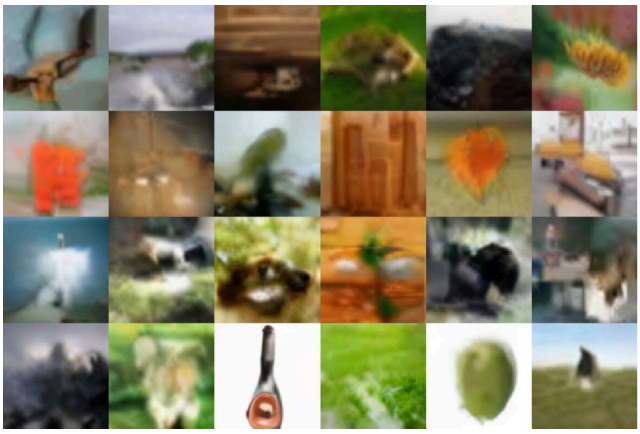
(f) NAR transformer with prompt tuning ($S = 128$)

Figure 14. Visualization of generated images with different models on Caltech101 of VTAB.

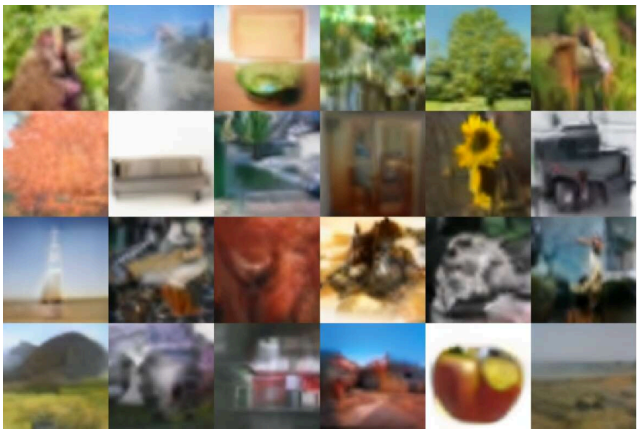
(a) MineGAN



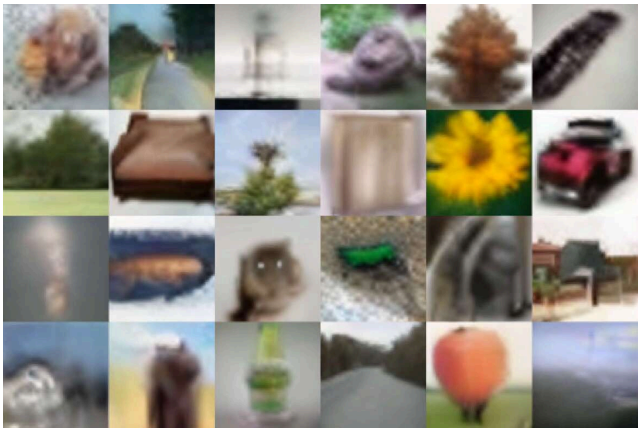
(b) cGANTransfer



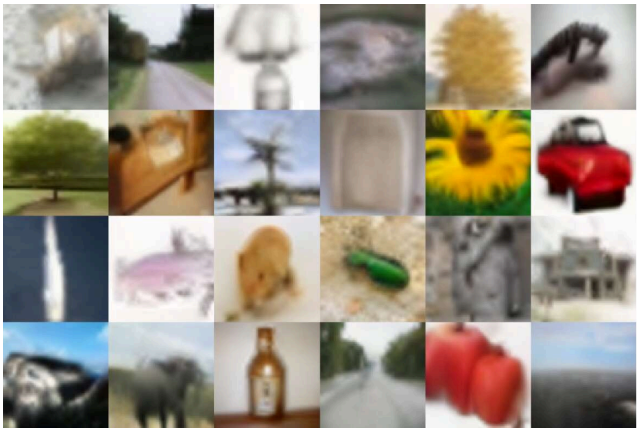
(c) AR transformer with prompt tuning ($S = 1$)



(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)



(e) NAR transformer with prompt tuning ($S = 1$)



(f) NAR transformer with prompt tuning ($S = 128$)

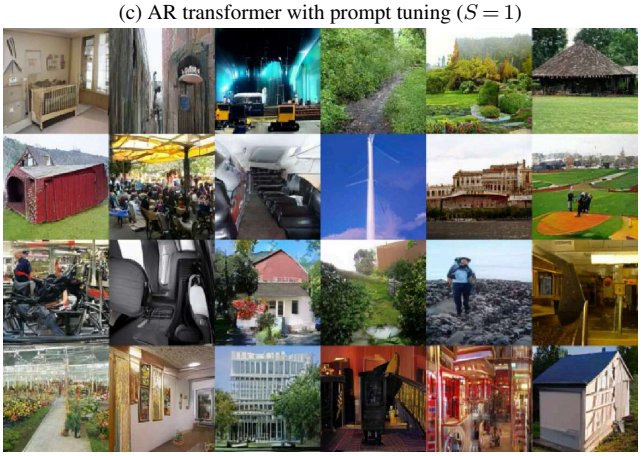
Figure 15. Visualization of generated images with different models on CIFAR100 of VTAB.

(a) MineGAN

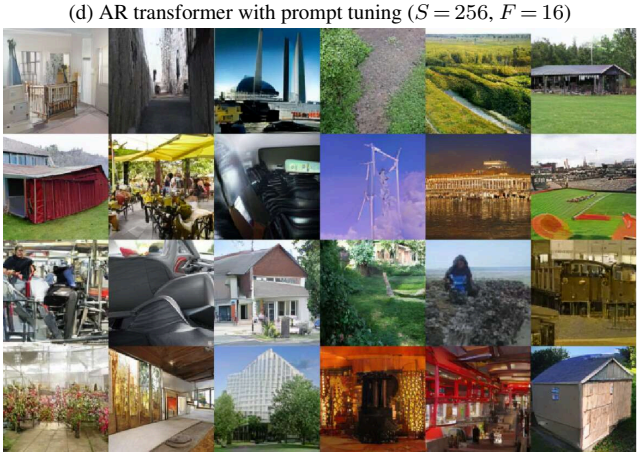
(b) cGANTransfer

(c) AR transformer with prompt tuning ($S = 1$)

(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)



(e) NAR transformer with prompt tuning ($S = 1$)

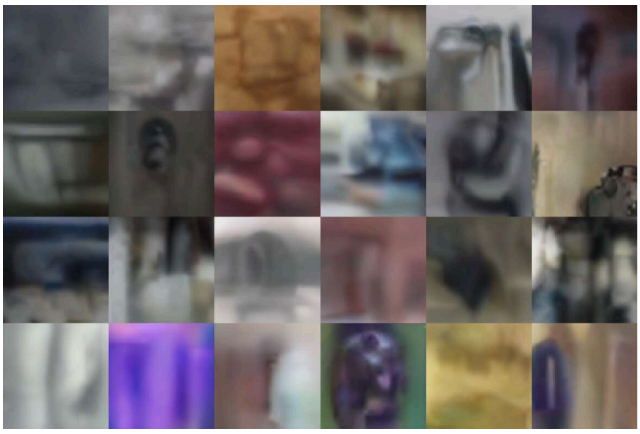


(f) NAR transformer with prompt tuning ($S = 128$)

Figure 16. Visualization of generated images with different models on SUN397 of VTAB.

(a) MineGAN

(b) cGANTransfer



(c) AR transformer with prompt tuning ($S = 1$)

(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)

(e) NAR transformer with prompt tuning ($S = 1$)

(f) NAR transformer with prompt tuning ($S = 128$)

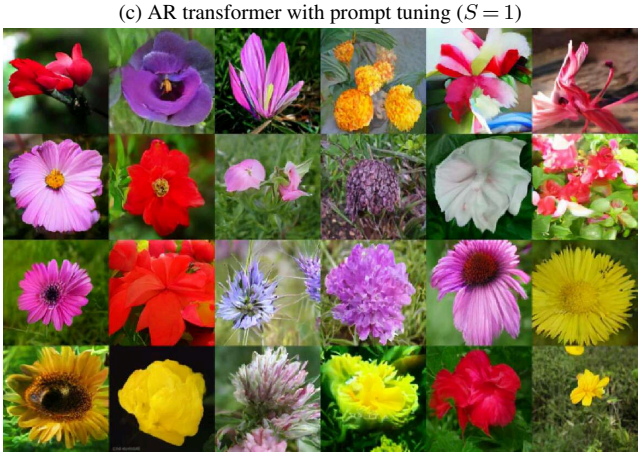
Figure 17. Visualization of generated images with different models on SVHN of VTAB.

(a) MineGAN

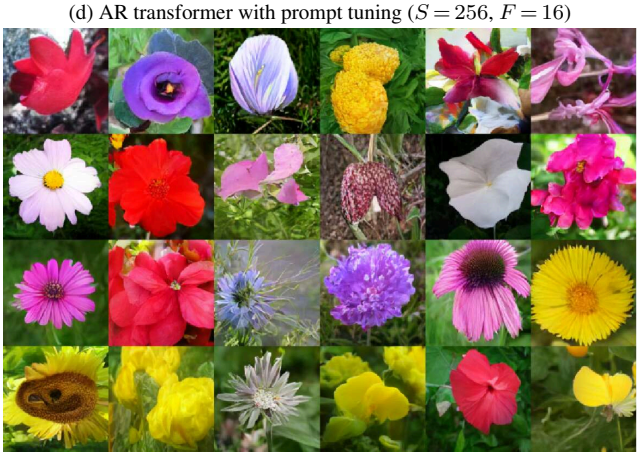
(b) cGANTransfer

(c) AR transformer with prompt tuning ($S = 1$)

(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)



(e) NAR transformer with prompt tuning ($S = 1$)



(f) NAR transformer with prompt tuning ($S = 128$)

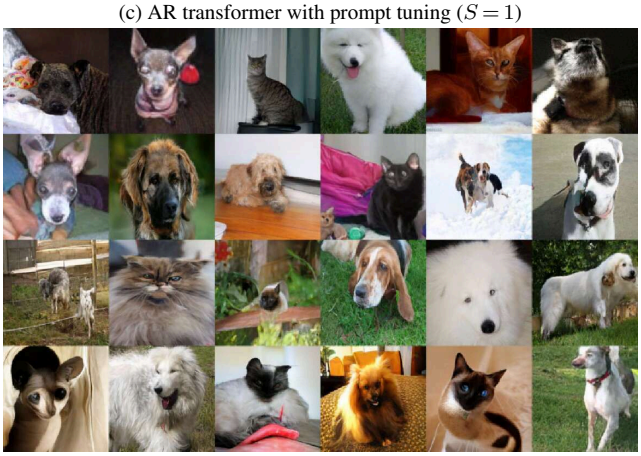
Figure 18. Visualization of generated images with different models on Oxford Flowers102 of VTAB.

(a) MineGAN

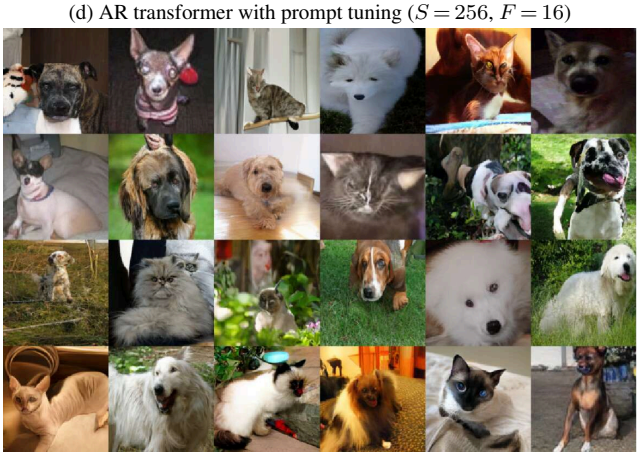
(b) cGANTransfer

(c) AR transformer with prompt tuning ($S = 1$)

(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)



(e) NAR transformer with prompt tuning ($S = 1$)



(f) NAR transformer with prompt tuning ($S = 128$)

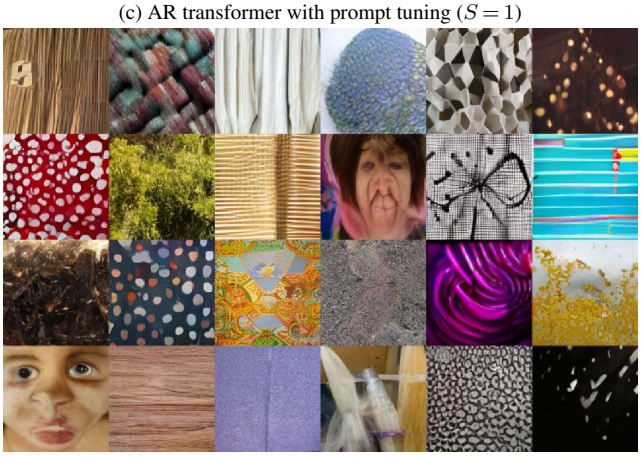
Figure 19. Visualization of generated images with different models on Oxford iiit Pet of VTAB.

(a) MineGAN

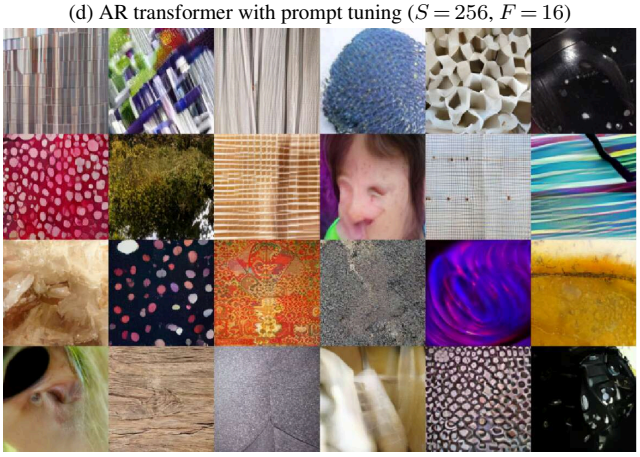
(b) cGANTransfer

(c) AR transformer with prompt tuning ($S = 1$)

(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)

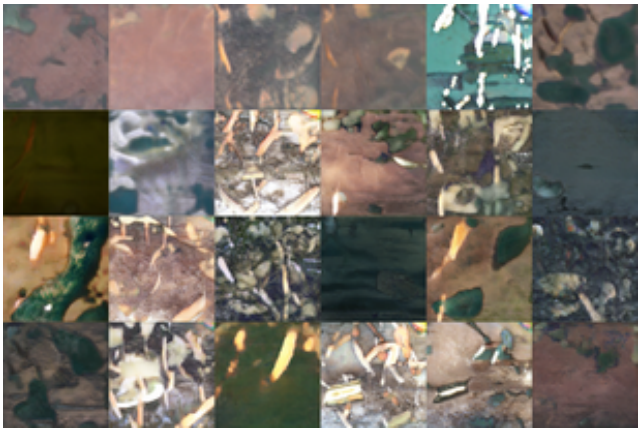


(e) NAR transformer with prompt tuning ($S = 1$)

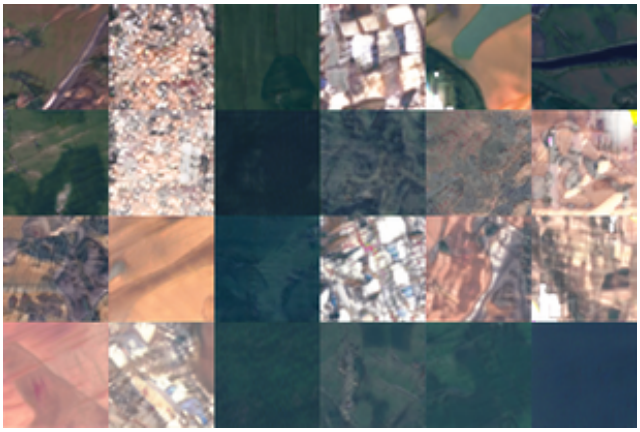


(f) NAR transformer with prompt tuning ($S = 128$)

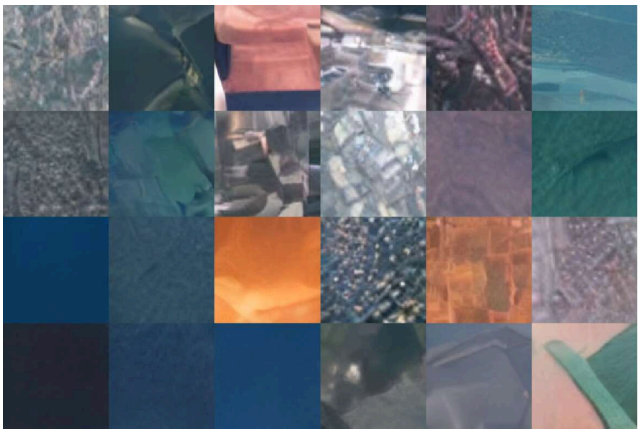
Figure 20. Visualization of generated images with different models on DTD of VTAB.



(a) MineGAN

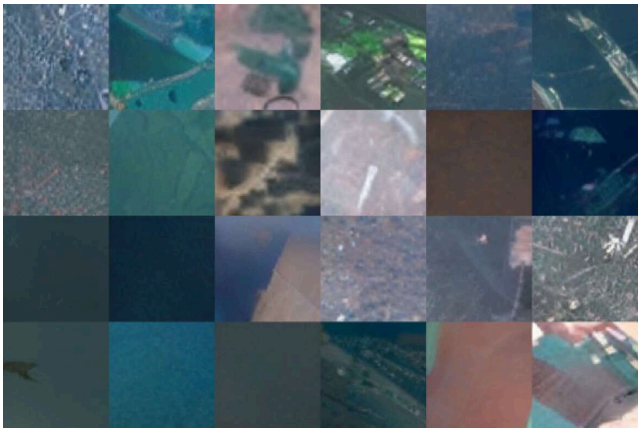


(b) cGANTransfer

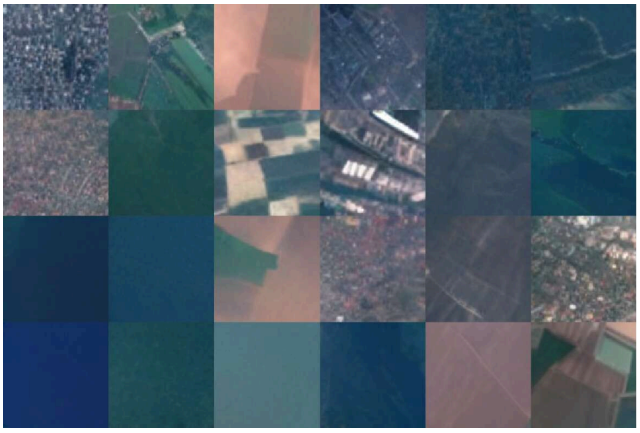


(c) AR transformer with prompt tuning ($S = 1$)

(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)

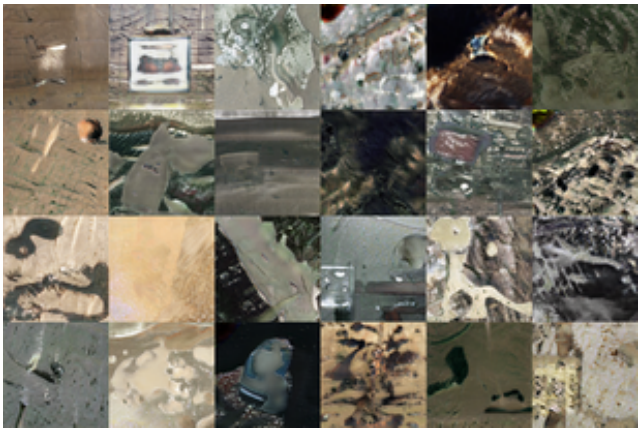


(e) NAR transformer with prompt tuning ($S = 1$)



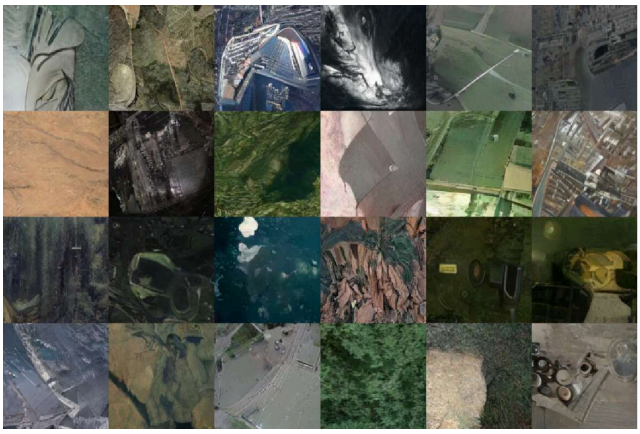
(f) NAR transformer with prompt tuning ($S = 128$)

Figure 21. Visualization of generated images with different models on EuroSAT of VTAB.



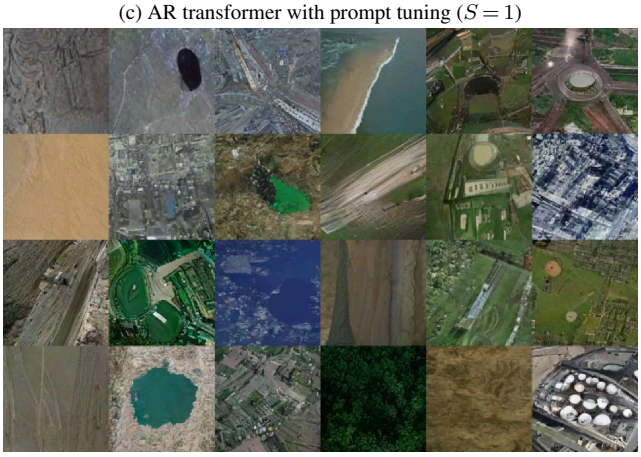
(a) MineGAN

(b) cGANTransfer

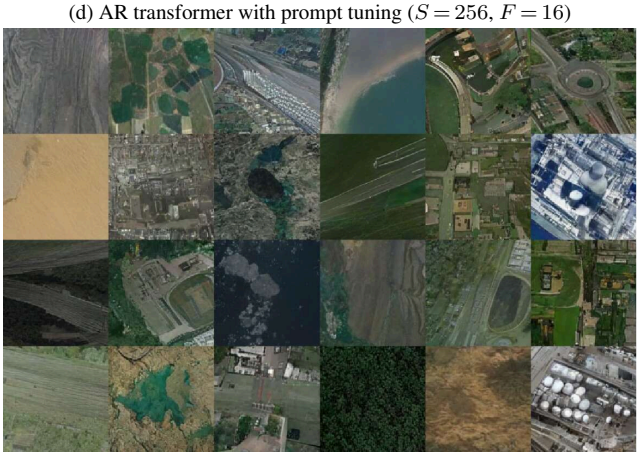


(c) AR transformer with prompt tuning ($S = 1$)

(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)

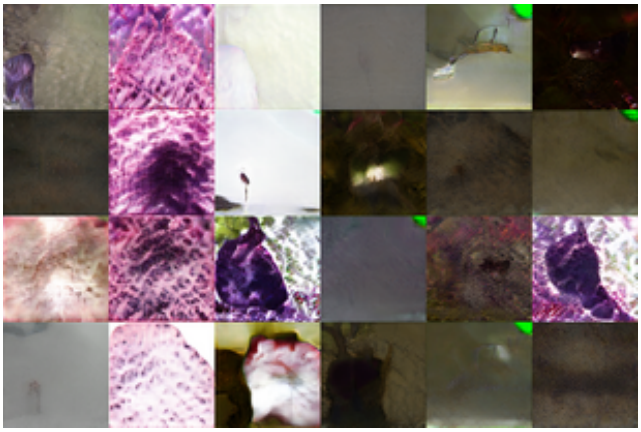


(e) NAR transformer with prompt tuning ($S = 1$)

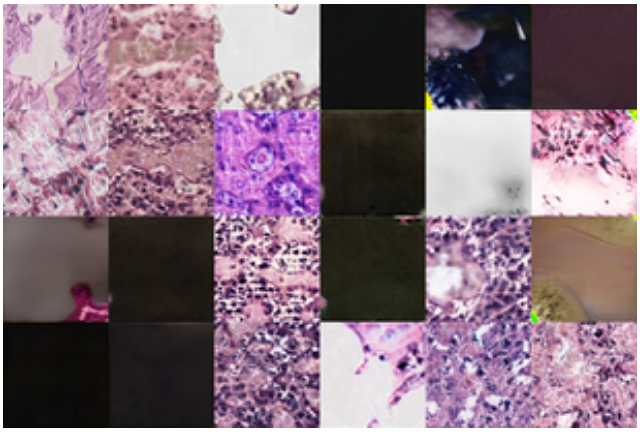


(f) NAR transformer with prompt tuning ($S = 128$)

Figure 22. Visualization of generated images with different models on Resisc45 of VTAB.

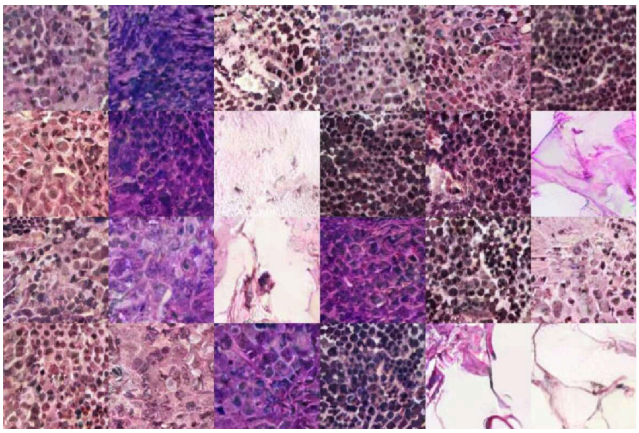


(a) MineGAN

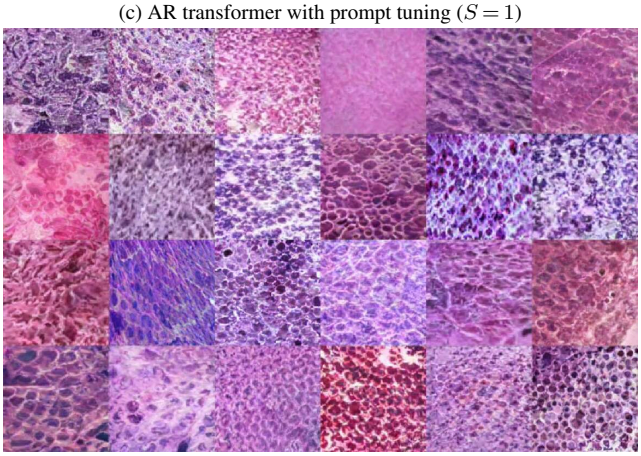


(b) cGANTransfer

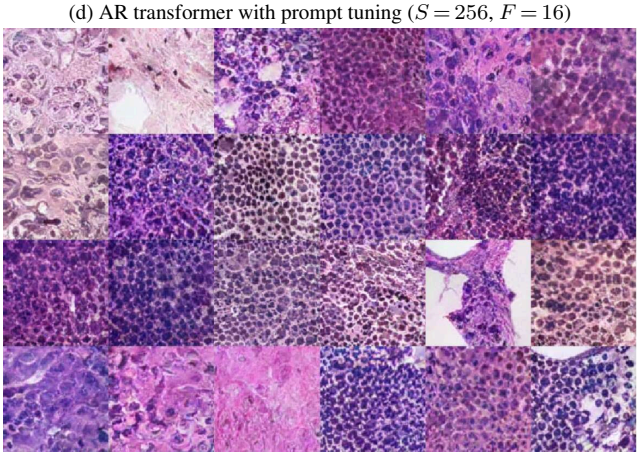
(c) AR transformer with prompt tuning ($S = 1$)



(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)

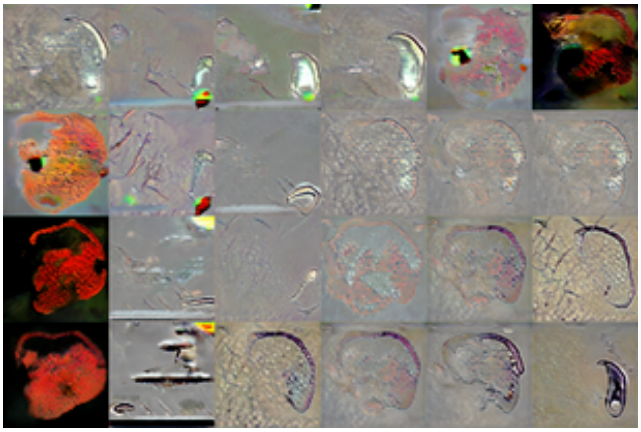


(e) NAR transformer with prompt tuning ($S = 1$)



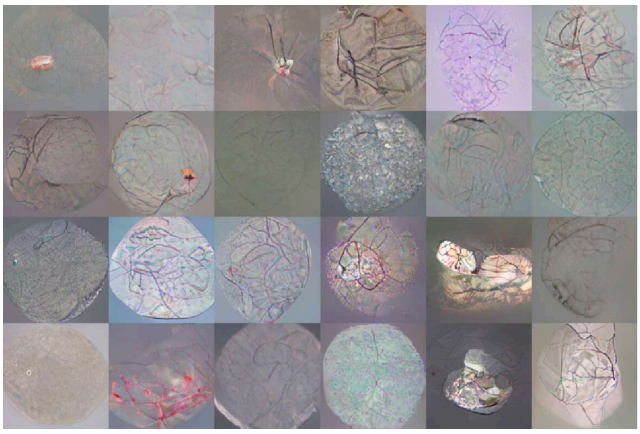
(f) NAR transformer with prompt tuning ($S = 128$)

Figure 23. Visualization of generated images with different models on Patch Camelyon of VTAB.

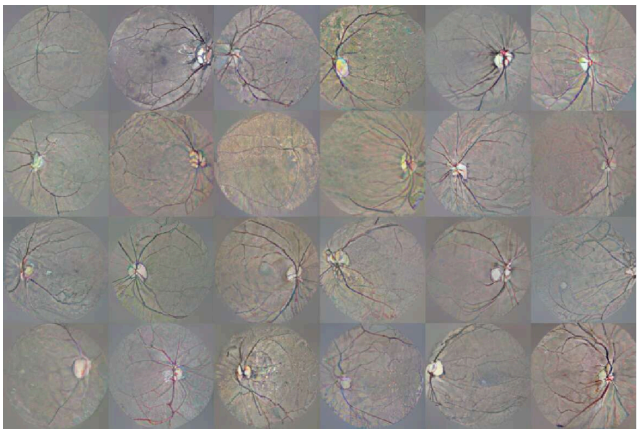


(a) MineGAN

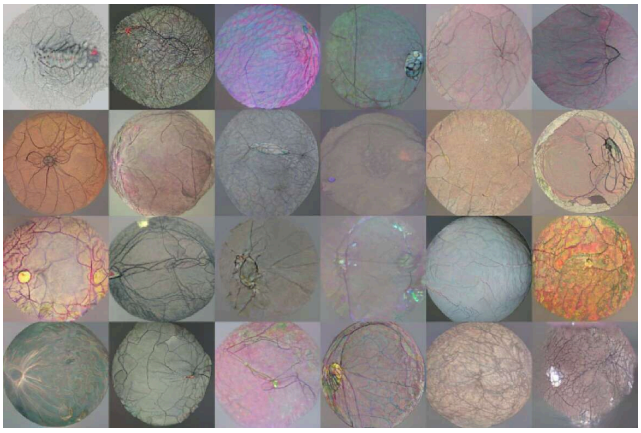
(b) cGANTransfer



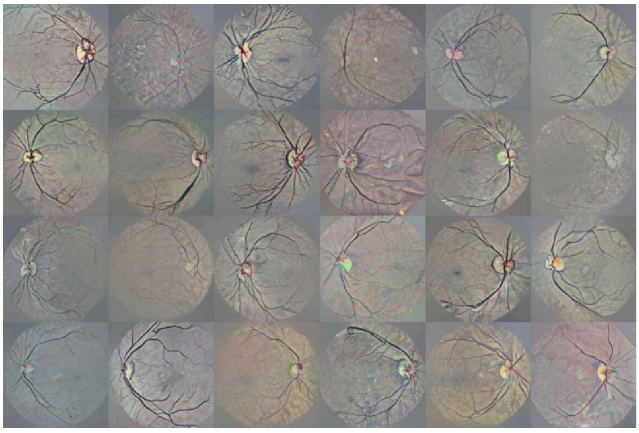
(c) AR transformer with prompt tuning ($S = 1$)



(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)



(e) NAR transformer with prompt tuning ($S = 1$)

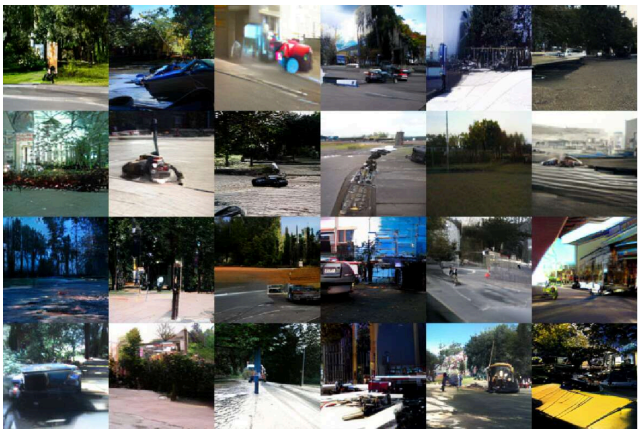


(f) NAR transformer with prompt tuning ($S = 128$)

Figure 24. Visualization of generated images with different models on Diabetic Retinopathy of VTAB.

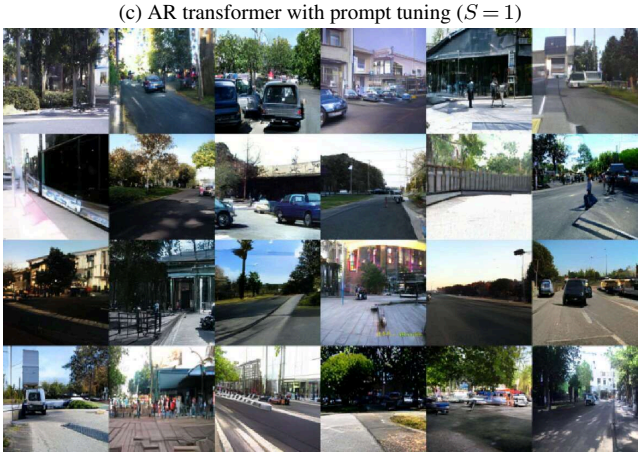
(a) MineGAN

(b) cGANTransfer



(c) AR transformer with prompt tuning ($S = 1$)

(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)

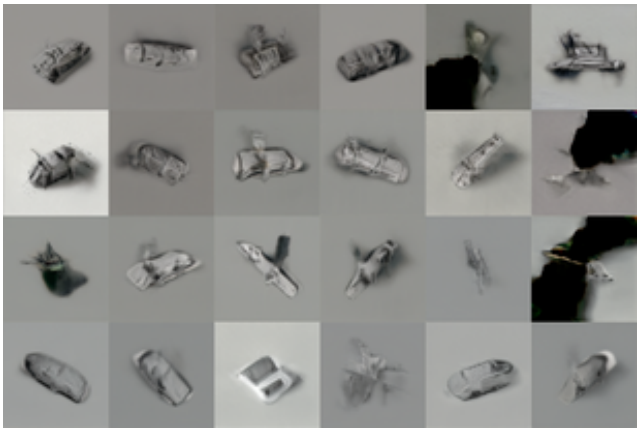


(e) NAR transformer with prompt tuning ($S = 1$)

(f) NAR transformer with prompt tuning ($S = 128$)

Figure 25. Visualization of generated images with different models on Kitti of VTAB.

(a) MineGAN



(b) cGANTransfer

(c) AR transformer with prompt tuning ($S = 1$)

(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)

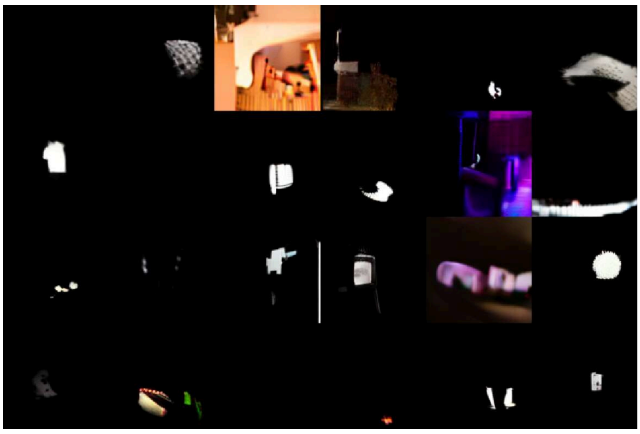
(e) NAR transformer with prompt tuning ($S = 1$)

(f) NAR transformer with prompt tuning ($S = 128$)

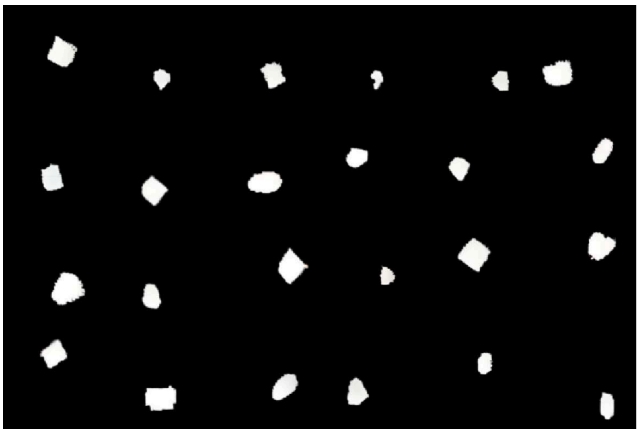
Figure 26. Visualization of generated images with different models on Smallnorb of VTAB.

(a) MineGAN

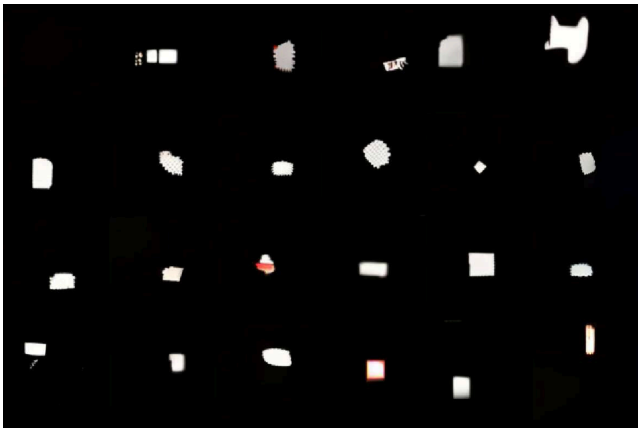
(b) cGANTransfer



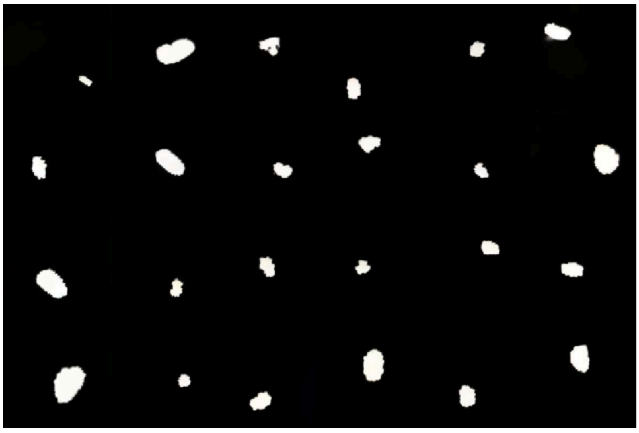
(c) AR transformer with prompt tuning ($S = 1$)



(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)

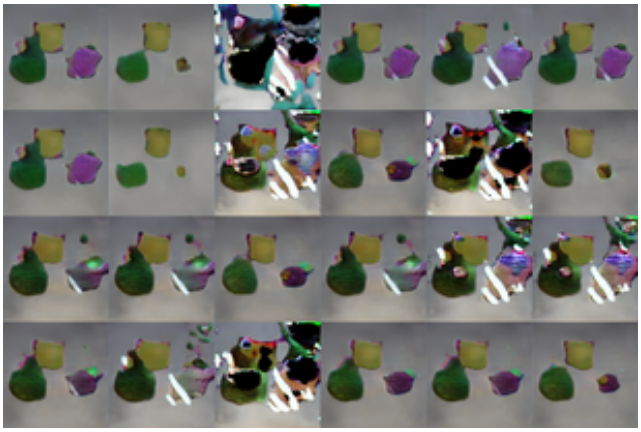


(e) NAR transformer with prompt tuning ($S = 1$)



(f) NAR transformer with prompt tuning ($S = 128$)

Figure 27. Visualization of generated images with different models on Dsprites of VTAB.



(a) MineGAN

(b) cGANTransfer

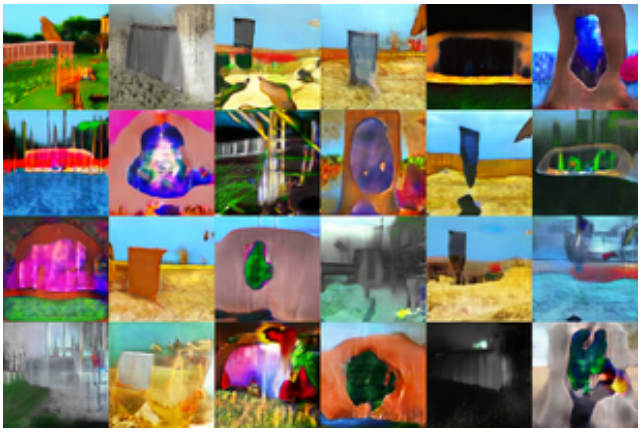
(c) AR transformer with prompt tuning ($S = 1$)

(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)

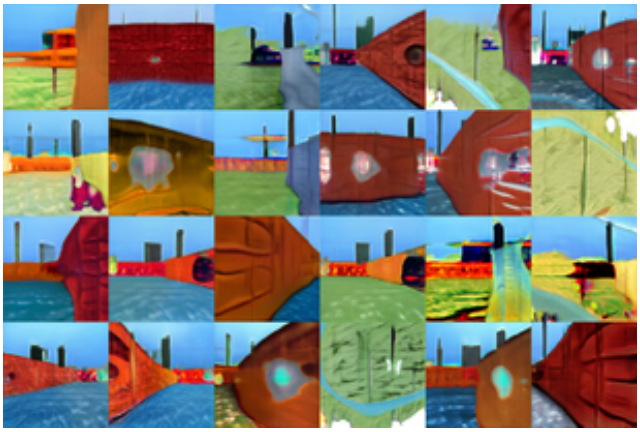
(e) NAR transformer with prompt tuning ($S = 1$)

(f) NAR transformer with prompt tuning ($S = 128$)

Figure 28. Visualization of generated images with different models on Clevr of VTAB.



(a) MineGAN



(b) cGANTransfer

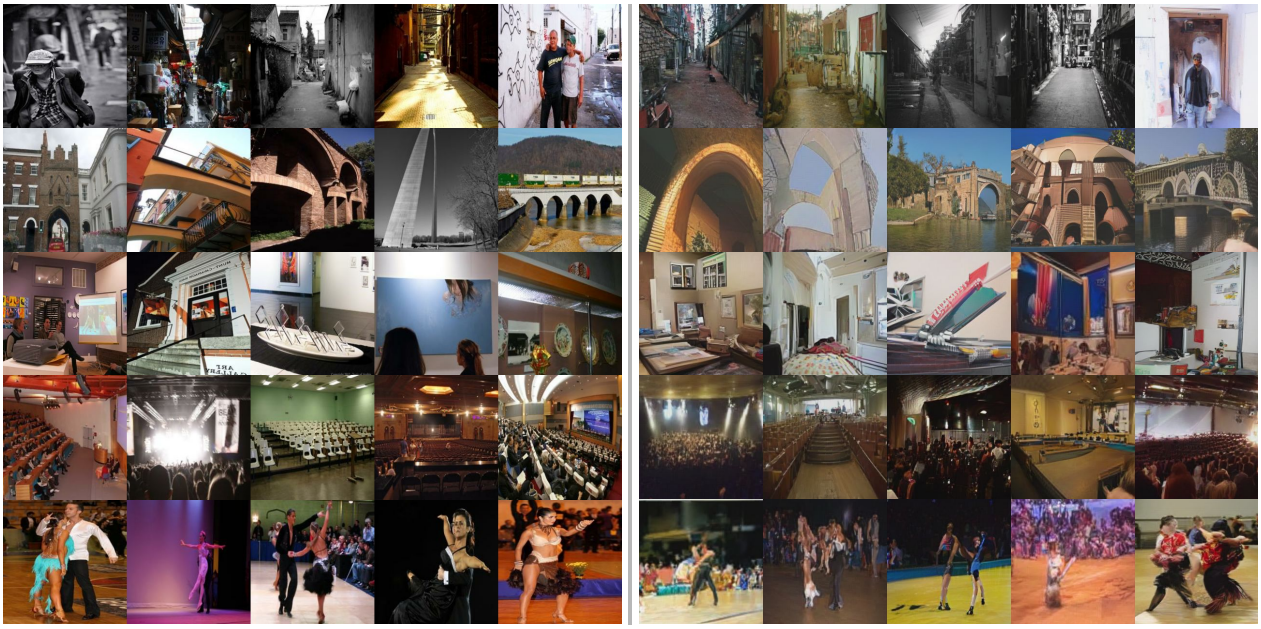
(c) AR transformer with prompt tuning ($S = 1$)

(d) AR transformer with prompt tuning ($S = 256$, $F = 16$)

(e) NAR transformer with prompt tuning ($S = 1$)

(f) NAR transformer with prompt tuning ($S = 128$)

Figure 29. Visualization of generated images with different models on DMLab of VTAB.



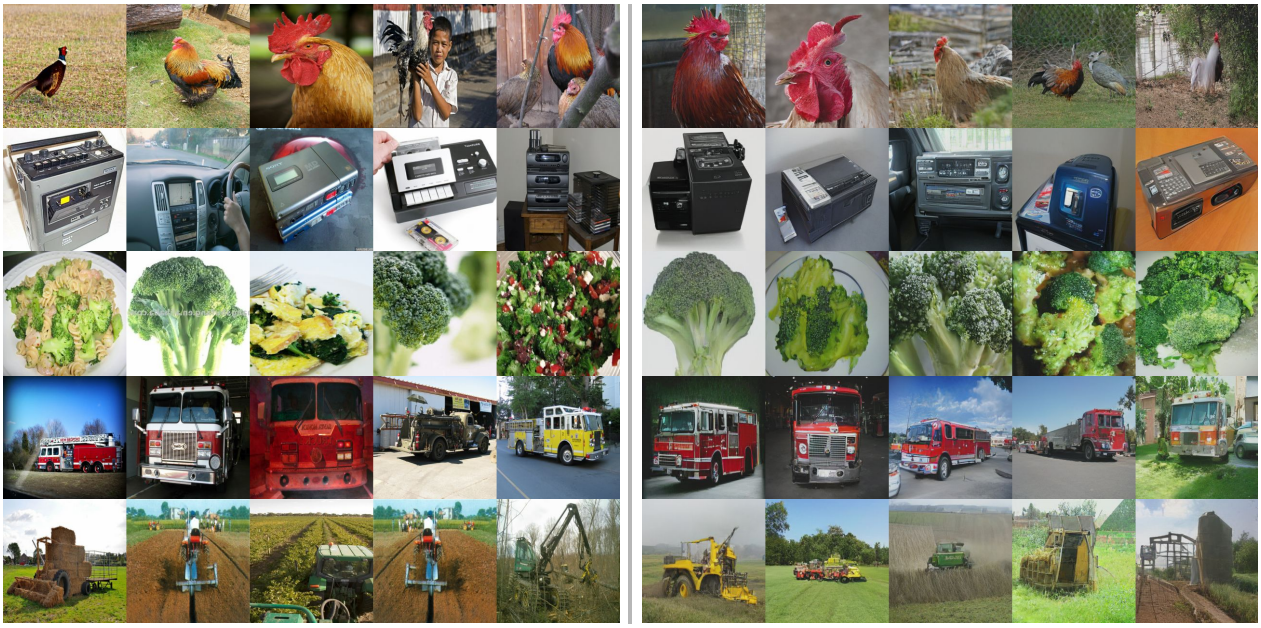
(a) Places, 5-shot, Left: real, Right: generation.

(b) Places, 500-shot, All generation, without cherry-picking.

Figure 30. Fewshot generation on places.

B.2. Few-shot Generative Transfer

B.2.1 Visualization of Generated Images



(a) ImageNet, 5-shot, Left: real, Right: generation.

(b) ImageNet, 500-shot, All generation, without cherry-picking.

Figure 31. Fewshot generation on ImageNet.

(a) Animal Face, 5-shot, Left: real, Right: generation.

(b) Animal Face, 100-shot, All generation, without cherry-picking.

Figure 32. Fewshot generation on Animal Face.