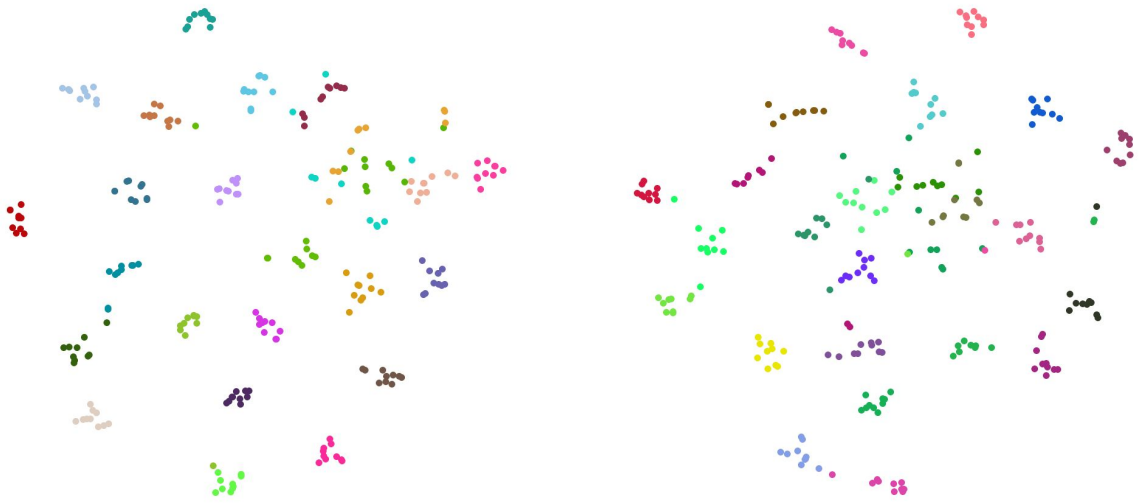


(a) $S = 1$ (NMI=0.848) (b) $S = 128$ (NMI=0.800)

Figure 10. 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.

A. Pseudo-code for Token Generator

In Fig. 13, we provide an example code that implements the prompt token generator in Flax [25] format.

B. Analysis and Discussion

In addition to the analysis in Sec. 4.3, we present extra study to better understand prompt tuning for generative transfer learning. In Appendix B.1 we study to understand prompt representations. In Appendix B.4, we make a comparison to other transfer learning methods in the context of image synthesis.

B.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 [68] of prompts in Fig. 10. 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. 10a 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. 10b, 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. We note that these are even better results than the number obtained using an embedding from ImageNet pretrained ResNet-50 [24] (NMI=0.734).

B.2. Adaptation-Diversity Trade-Off

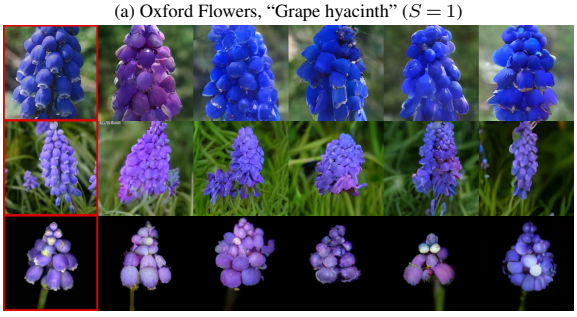
We study prompts with various lengths, but on a *single* image. We show generated images of models with different lengths in Fig. 11. With short prompts, the model produces diverse but less detailed images. On the other hand, a long prompt model generates images of a higher quality, more faithful to 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 Appendix B.1 where short prompts learn more discriminative information than long prompts.

In Fig. 12, we visualize images generated by models of Appendix B.1. Compared to images in Fig. 12a whose model is trained with $S = 1$, we clearly see in Fig. 12c that the model trained with a long prompt generates images that are more consistent with training instances.

Figure 11. A single training image in red box and those generated by models using prompts of various lengths from 1 to 128.

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

(b) SUN397, "Bedroom" ($S = 1$)



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

(d) SUN397, "Bedroom" ($S = 128$)

Figure 12. 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.

B.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. In addition to Sec. 4.3, we provide an extra study on different prompt token generators.

Tab. 7 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.

B.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. In addition, we 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 to transfer transformer-based sequence models.

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

Table 5. Qualitative comparison (*e.g.*, number of trainable parameters, train and generation time) among various learning strategies based on NAR transformers.

Method	10 epoch			200 epoch			800 (prompt) or 1600 epoch		
	S (<10k)	M (<100k)	L (>100k)	S (<10k)	M (<100k)	L (>100k)	S (<10k)	M (<100k)	L (>100k)
Prompt ($S=128$)	27.6	33.7	84.2	18.5	30.6	88.9	17.7	30.7	88.9
Prompt + Adapter	20.1	24.8	**72.7**	15.7	**22.8**	**73.0**	15.1	23.3	**74.0**
Prompt + Fine-tune	**19.5**	**23.2**	**72.7**	**15.0**	28.8	74.2	**14.2**	33.9	74.2
Scratch		–		60.0	26.0	74.9	22.7	**23.2**	76.5

Table 6. FID vs the number of train epochs for various learning methods for non-autoregressive transformer-based sequence models. Knowledge transfer is essential for faster convergence when training data is small. For adapter or fine-tuning, we also introduce learnable prompt of length $S=1$ to inject class-condition information.

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 7. Ablation on prompt token generators for (left) NAR and (right) AR transformers on VTAB. We report FIDs averaged by different categorizations of tasks.

For the completeness, we conduct an extended study for various learning methods of generative vision transformers.

To that end, we adopt adapter tuning and fine-tuning with a learnable prompt of length $S=1$ to inject condition information, in addition to the prompt tuning and learning from scratch. Adapter tuning [28] 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 [7] for NAR transformer).

For experiments, we vary the number of training epochs as training efficiency is one of the key differentiating factors across various learning strategies. We train models for 10, 200, 800 or 1600 epochs, while limiting the maximum number of training steps to $500K$ to train a model within a reasonable time window.

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. 5. 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. While the number of trainable parameters for prompt tuning is significantly less (*e.g.*, <0.5%) than that of fine-tuning or learning from scratch, the improvement in the training speed is less significant. This is because the prompt is attached at the first transformer layer and the model still needs to perform entire forward and backward passes to compute gradient. Nevertheless, the memory footprint for training and the parameter storage are the two major benefits of prompt tuning over the fine-tuning and learning from scratch. 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, As pointed out by [22], adapter tuning and prompt tuning are closely related, and we believe that our proposed idea of visual prompt tuning for generative model transfer would be further improved by exploiting this connection between adapter and prompt tuning methods.

Tab. 6 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 to new data

```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 [25] format.

distributions quickly, though it takes more time per step for model training. Complete FID results are in Tab. 10 of Appendix.

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.

C. Comprehensive Experiment Description

C.1. Visual Task Adaptation Benchmark (VTAB)

C.1.1 Dataset Meta Information of Visual Task Adaptation Benchmark

In Tab. 8 we provide a dataset meta information, including the number of class and the number of images in each data split, of VTAB.

C.1.2 Hyperparameters

We provide hyperparameters used in our experiments in Tab. 9. Note that most hyperparameters are shared across datasets, except the number of training epochs. We use Adam optimizer [33] with a cosine learning rate decay [43]. 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.

C.1.3 Experimental Results

We provide complete results in Tab. 10 for autoregressive transformers, non-autoregressive transformers as well as GAN-based generative model transfer learning methods including MineGAN [71] and cGANTransfer [60]. For AR and NAR transformers, we report FIDs for prompt tuning, learning from scratch, as well as different transfer learning techniques including adapter [28] and fine-tuning [35].

C.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 8. 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 9. 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.

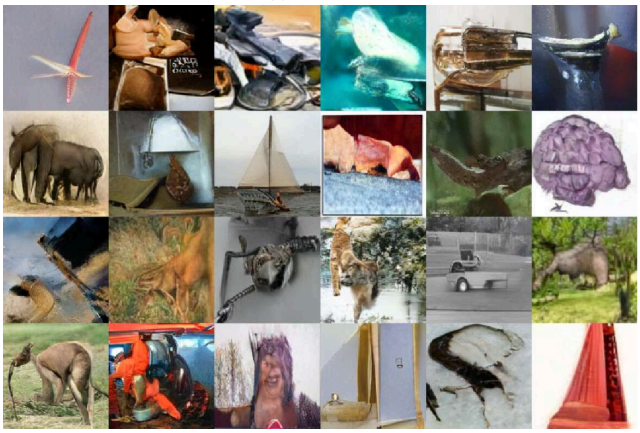
	Models	Caltech101	CIFAR100	SUN397	SVHN	Flower	Pet	DTD	EuroSAT	Resisc45	PC	DR	Kitti
	MineGAN	102.4	82.6	77.5	144.7	132.1	130.1	87.4	111.5	81.0	170.3	192.2	117.9
	cGANTransfer	89.6	31.4	31.1	64.7	61.6	48.6	70.3	45.6	50.3	119.9	149.8	48.9
NAR	Scratch	72.7	24.2	9.2	44.4	57.2	70.3	66.1	39.5	32.0	48.3	25.6	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=256)	13.0	24.4	7.6	56.4	13.7	11.0	26.4	35.2	25.4	61.8	30.7	31.5
NAR	P (S=512)	13.1	26.6	7.7	62.9	13.9	10.1	26.8	34.5	25.2	57.3	28.9	31.8
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	P (S=1) + 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	P (S=1) + 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	P (S=1) + 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	P (S=1) + 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=256)	36.8	35.2	100.9	92.8	23.4	24.7	30.0	35.8	19.1	30.0	85.2	21.8	46.9	38.3
NAR	P (S=512)	41.6	35.2	100.9	92.8	21.0	23.4	27.8	35.9	19.1	30.4	83.7	23.0	46.8	36.5
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	P (S=1) + 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	P (S=1) + 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	P (S=1) + 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	P (S=1) + 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

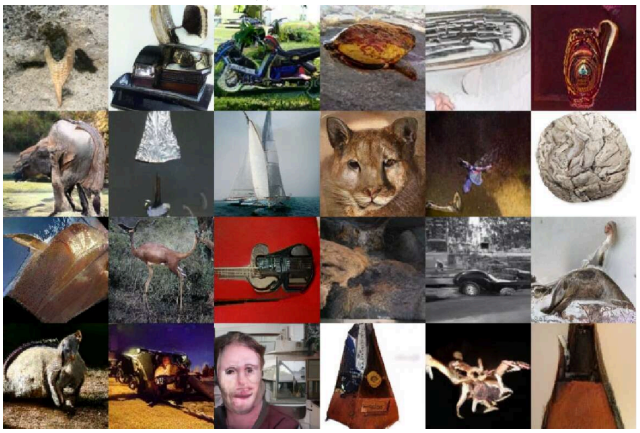
Table 10. FIDs on VTAB tasks tested with various models. We use the "all" set as a reference set for computing FIDs. Unless otherwise stated, all NAR models are trained for 200 epochs and AR models are trained for 400 epochs with the same hyperparameter settings specified in Tab. 9. "P" refers to the prompt tuning based on the proposed design of prompt generators with sequence length S and the number of factors F, while "P†" refers to the prompt tuning with a naive prompt design without any factorization. "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

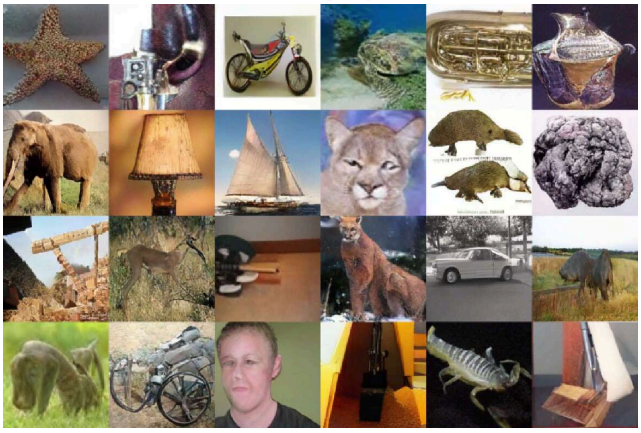
(b) cGANTransfer



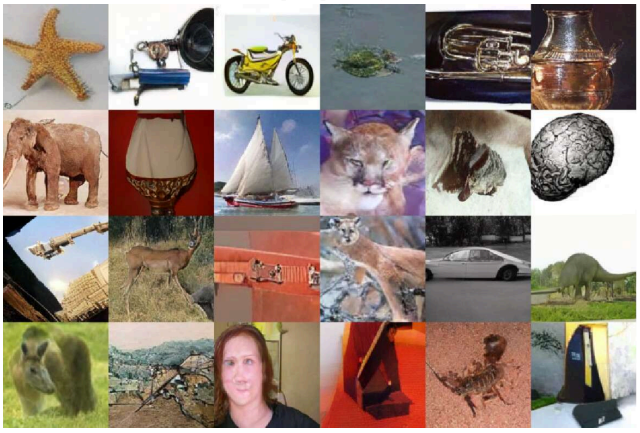
(c) AR transformer with prompt tuning ($S = 1$)



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



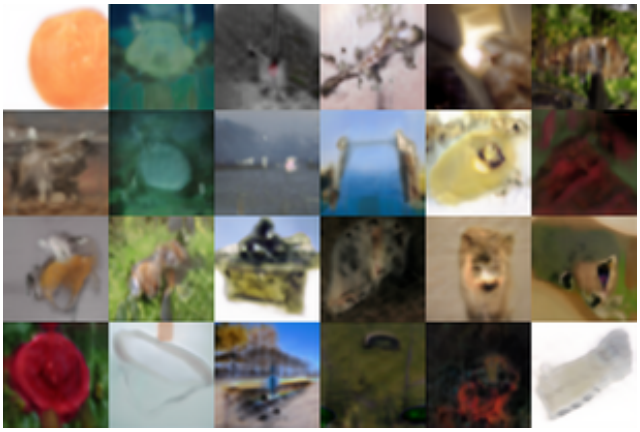
(e) NAR transformer with prompt tuning ($S = 1$)



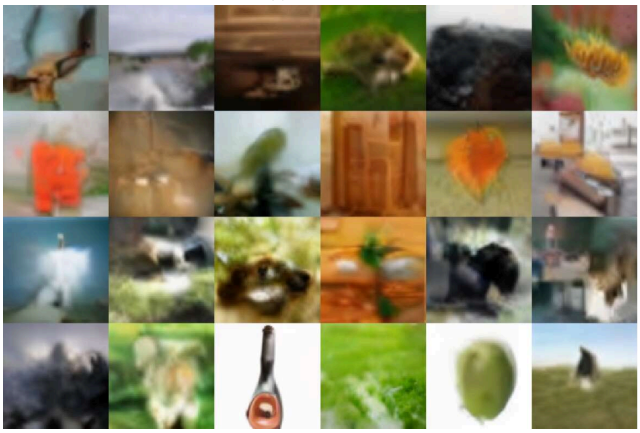
(f) NAR transformer with prompt tuning ($S = 128$)

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

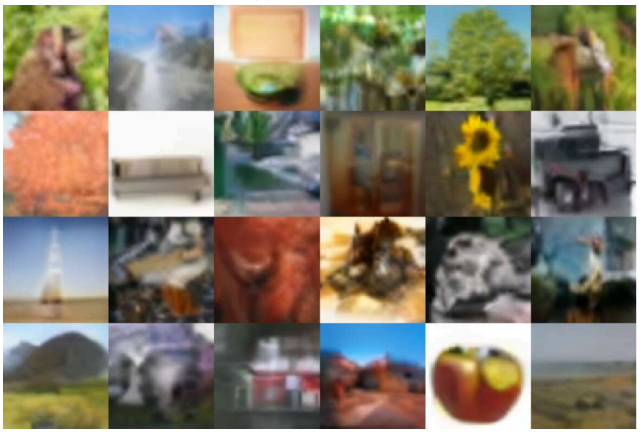
(a) MineGAN



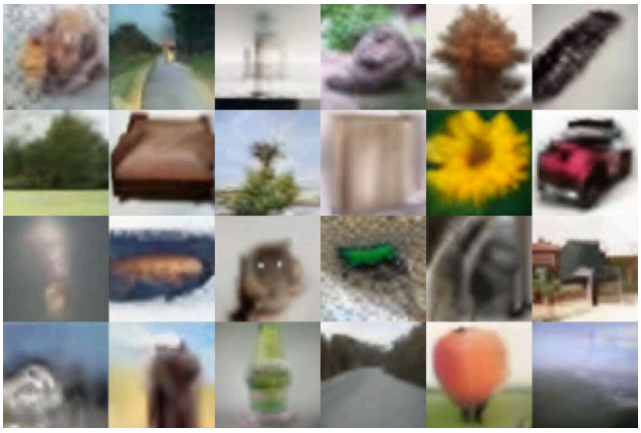
(b) cGANTransfer



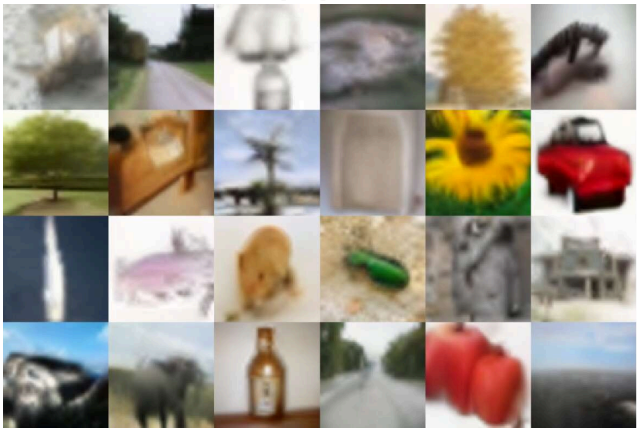
(c) AR transformer with prompt tuning ($S = 1$)



(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.

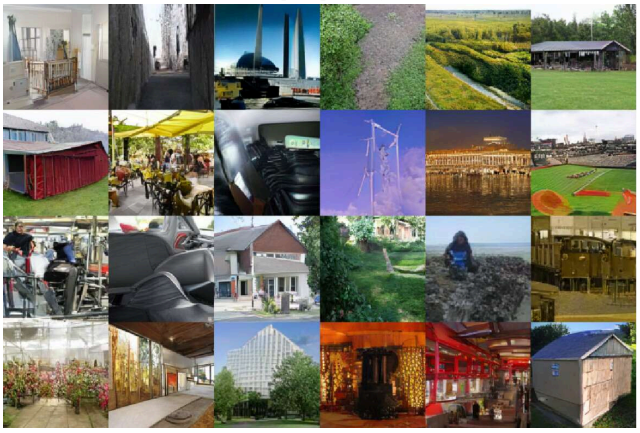
(a) MineGAN

(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 ($S = 1$)

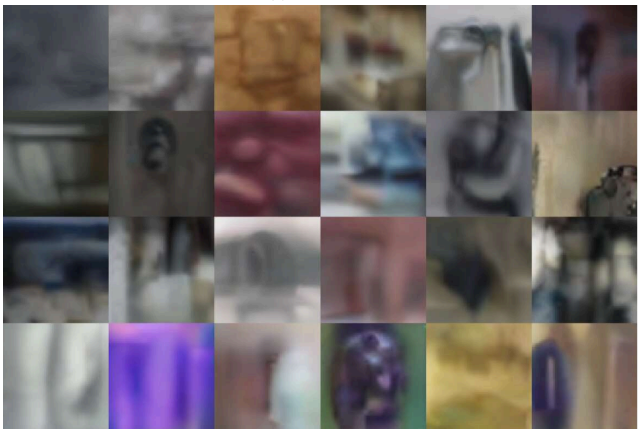


(f) NAR transformer with prompt tuning ($S = 128$)

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 ($S = 128$)

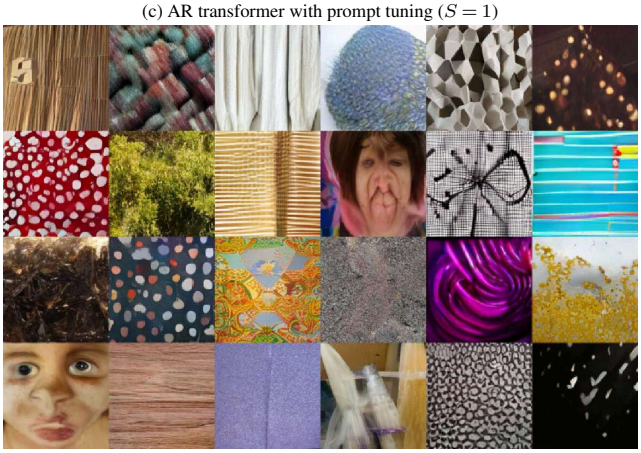
Figure 19. Visualization of generated images with different models on Oxford iiit Pet of VTAB.

(a) MineGAN

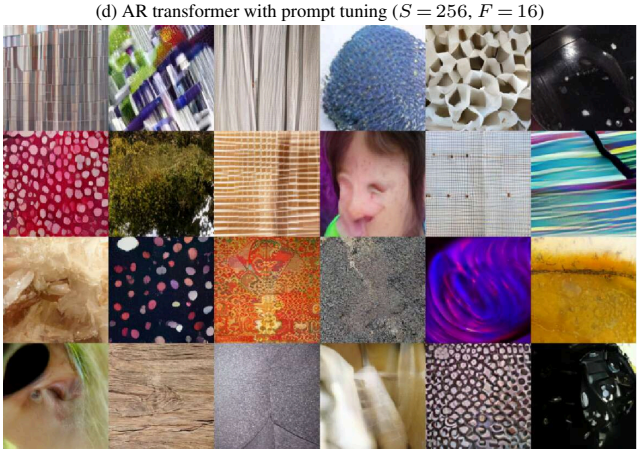
(b) cGANTransfer

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

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

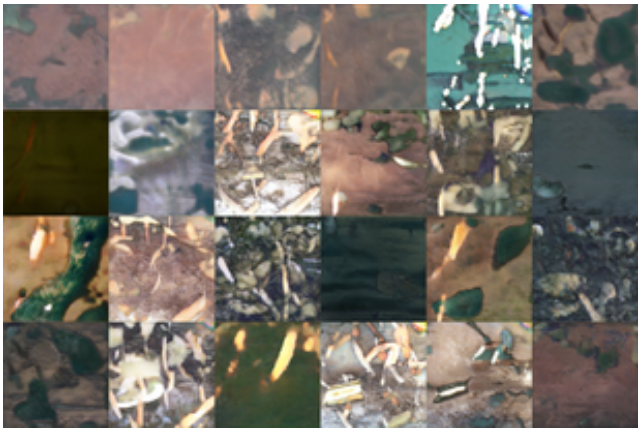


(e) NAR transformer with prompt tuning ($S = 1$)

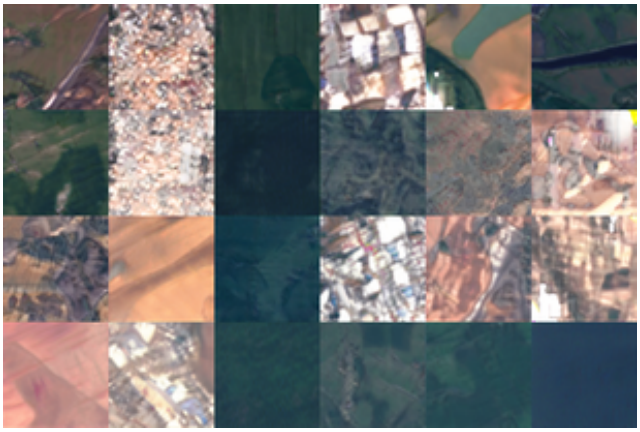


(f) NAR transformer with prompt tuning ($S = 128$)

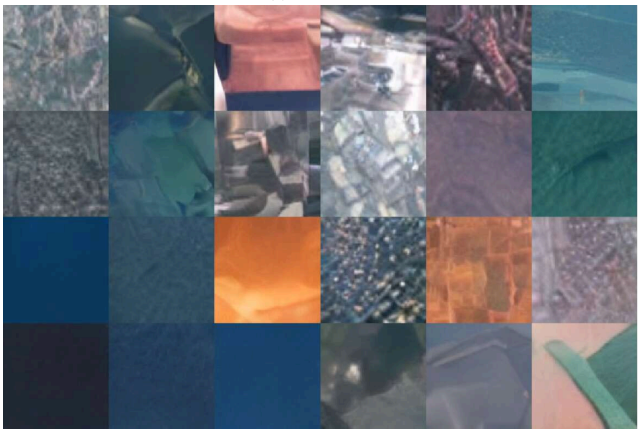
Figure 20. Visualization of generated images with different models on DTD of VTAB.



(a) MineGAN

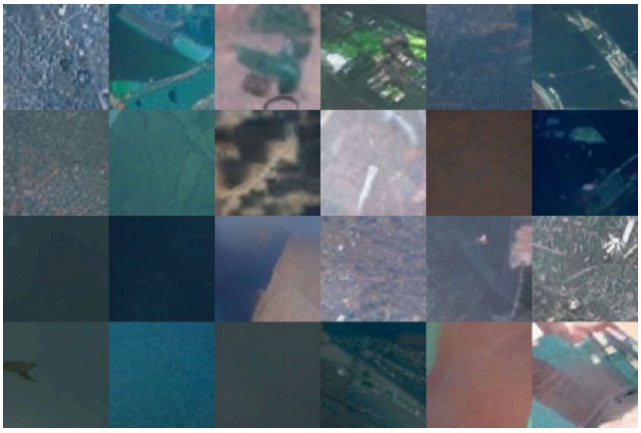


(b) cGANTransfer

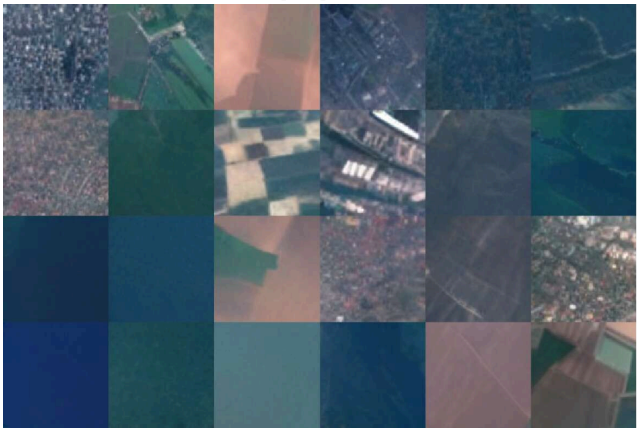


(c) AR transformer with prompt tuning ($S = 1$)

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

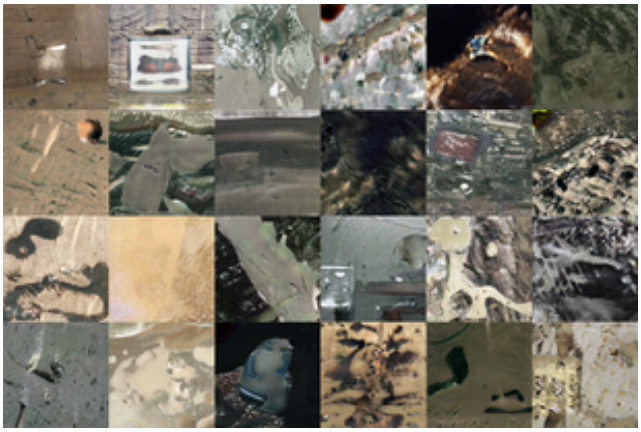


(e) NAR transformer with prompt tuning ($S = 1$)



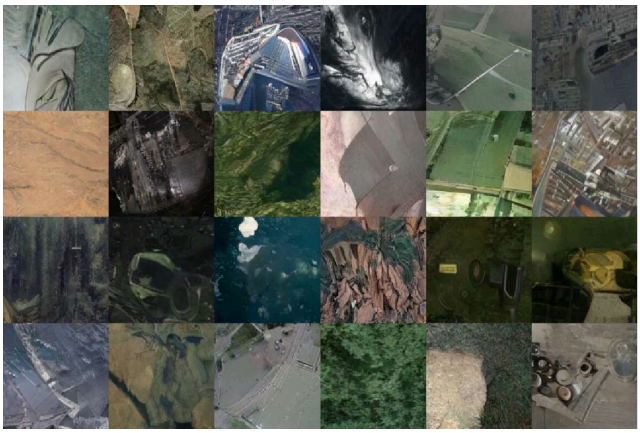
(f) NAR transformer with prompt tuning ($S = 128$)

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



(a) MineGAN

(b) cGANTransfer



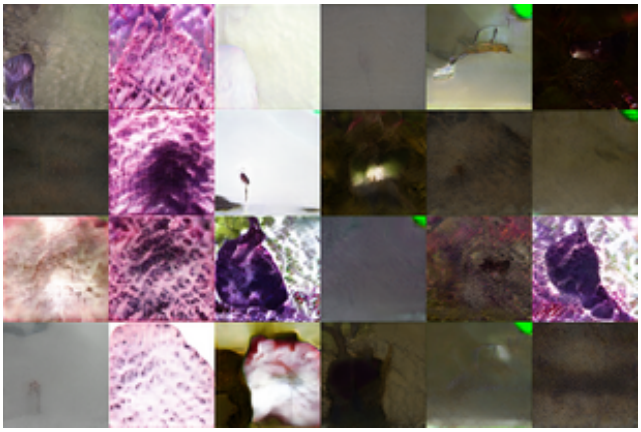
(c) AR transformer with prompt tuning ($S = 1$)

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

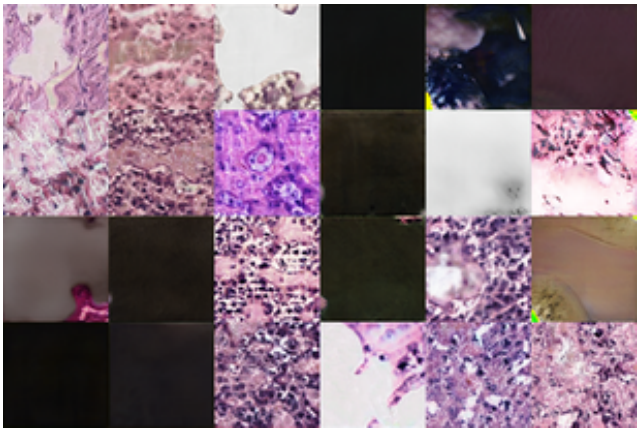
(e) NAR transformer with prompt tuning ($S = 1$)

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

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

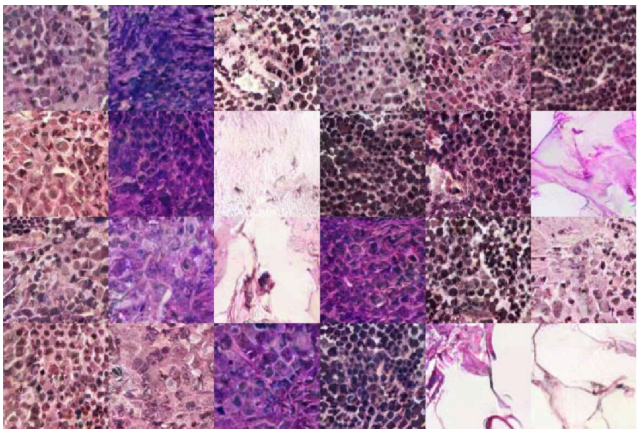


(a) MineGAN

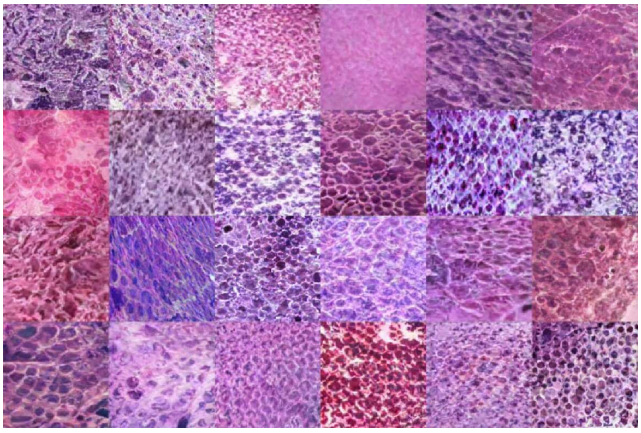


(b) cGANTransfer

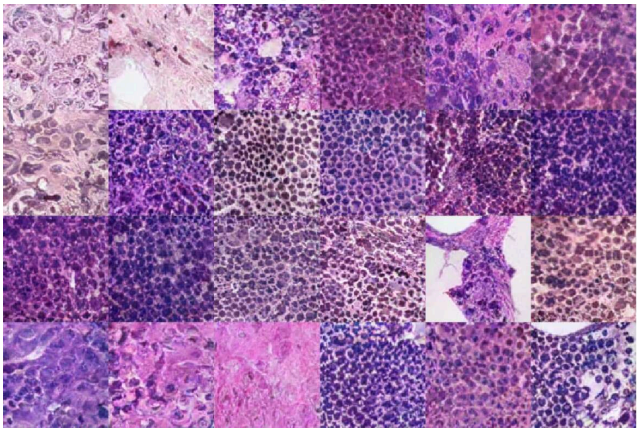
(c) AR transformer with prompt tuning ($S = 1$)



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

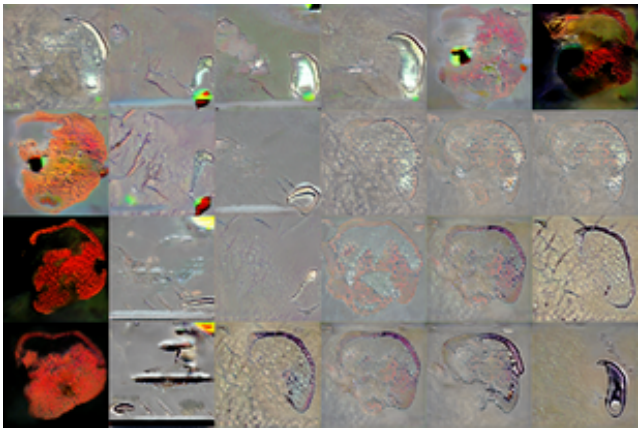


(e) NAR transformer with prompt tuning ($S = 1$)



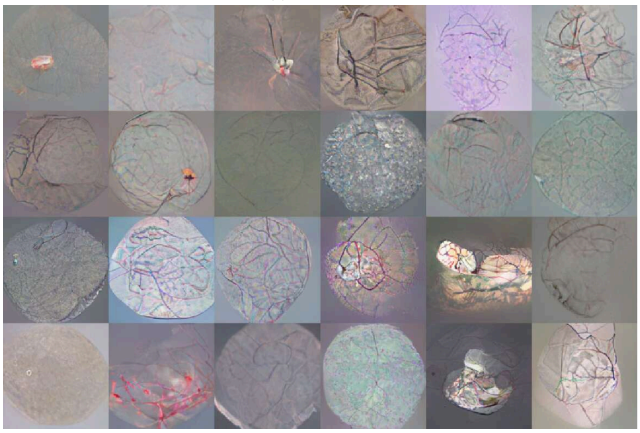
(f) NAR transformer with prompt tuning ($S = 128$)

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

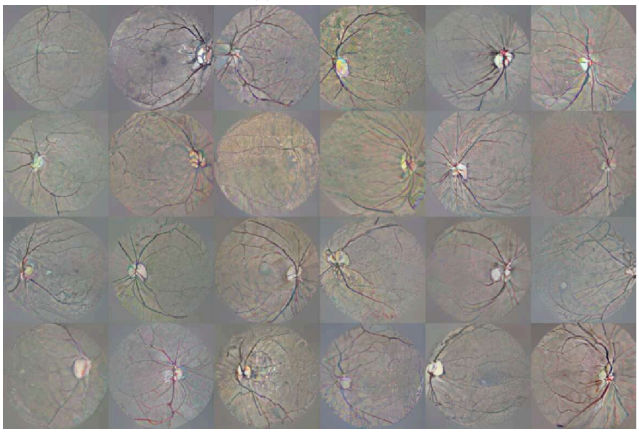


(a) MineGAN

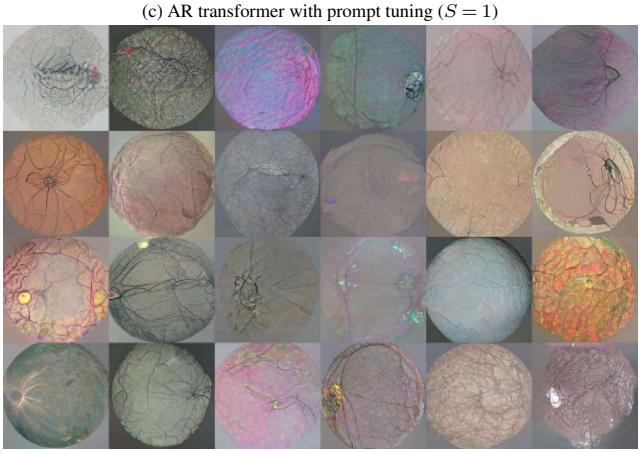
(b) cGANTransfer



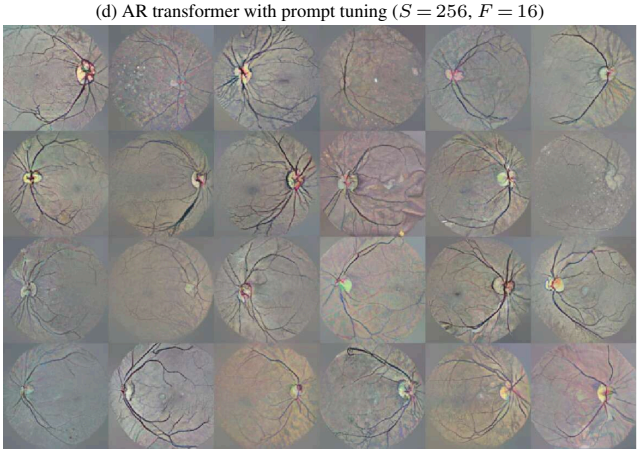
(c) AR transformer with prompt tuning ($S = 1$)



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



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

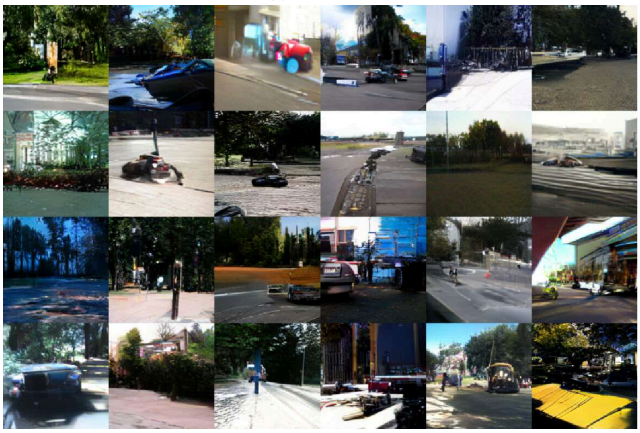


(f) NAR transformer with prompt tuning ($S = 128$)

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

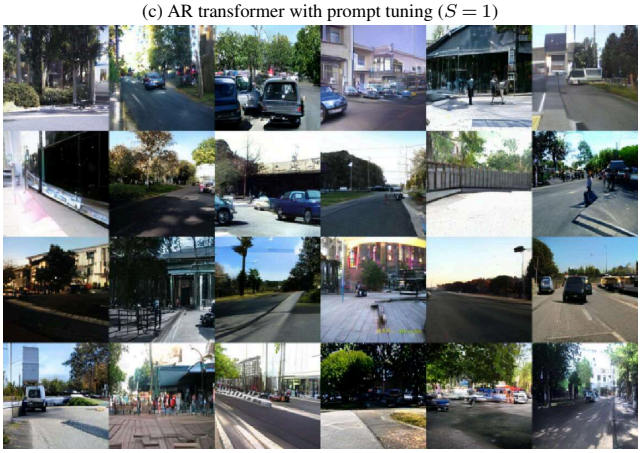
(a) MineGAN

(b) cGANTransfer



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

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

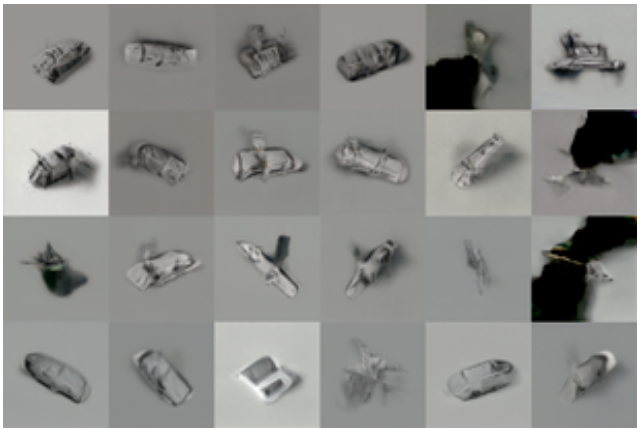


(e) NAR transformer with prompt tuning ($S = 1$)

(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 ($S = 256$, $F = 16$)

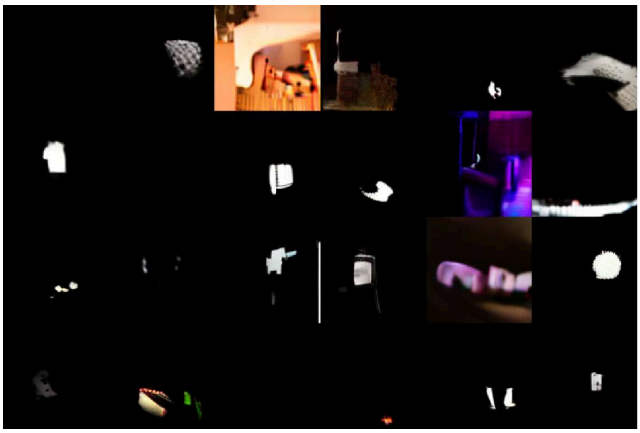
(e) NAR transformer with prompt tuning ($S = 1$)

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

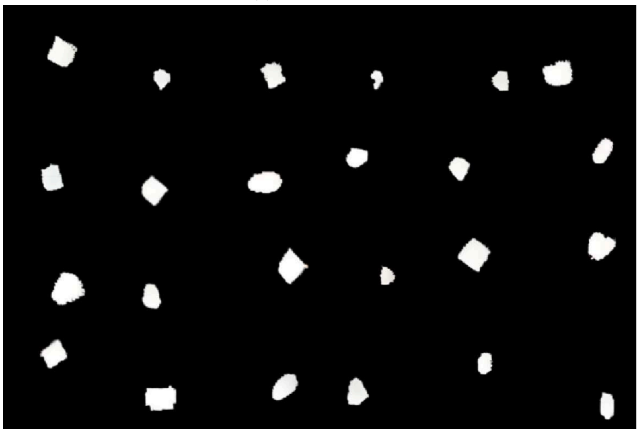
Figure 26. Visualization of generated images with different models on Smallnorb of VTAB.

(a) MineGAN

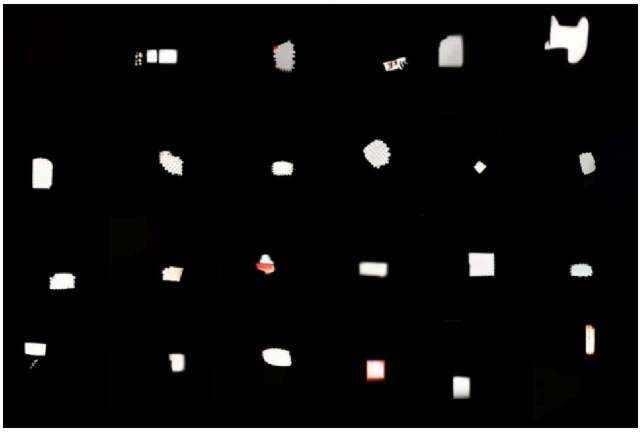
(b) cGANTransfer



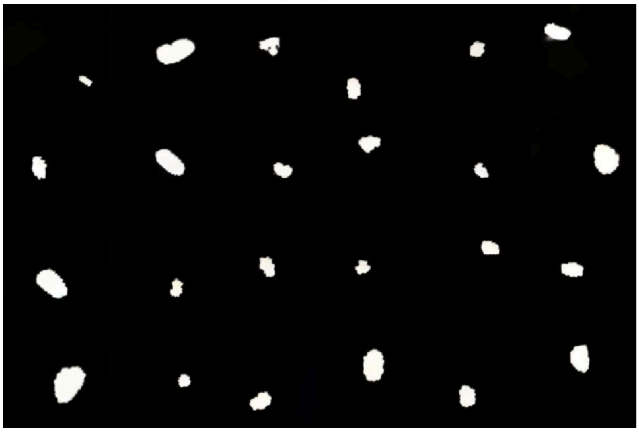
(c) AR transformer with prompt tuning ($S = 1$)



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

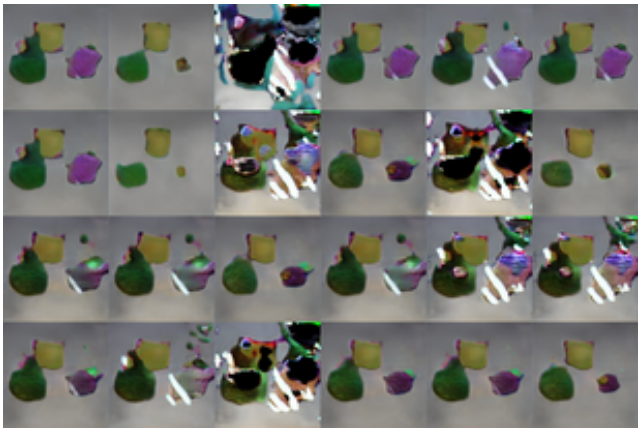


(e) NAR transformer with prompt tuning ($S = 1$)



(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

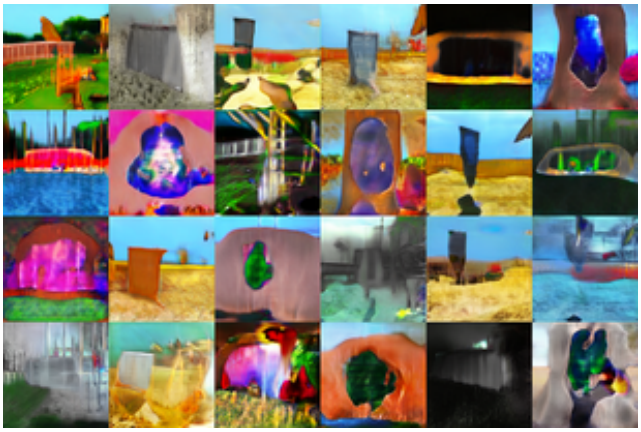
(c) AR transformer with prompt tuning ($S = 1$)

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

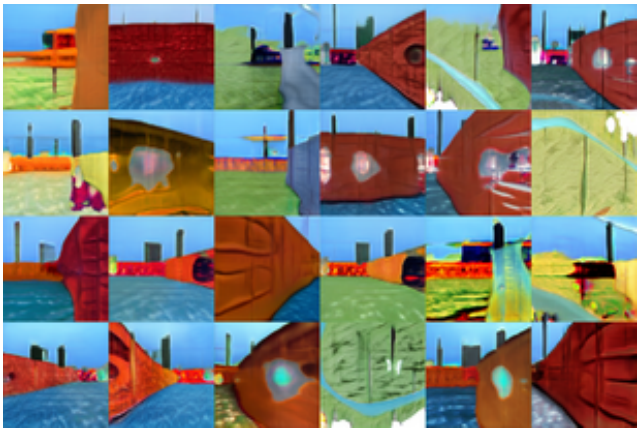
(e) NAR transformer with prompt tuning ($S = 1$)

(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

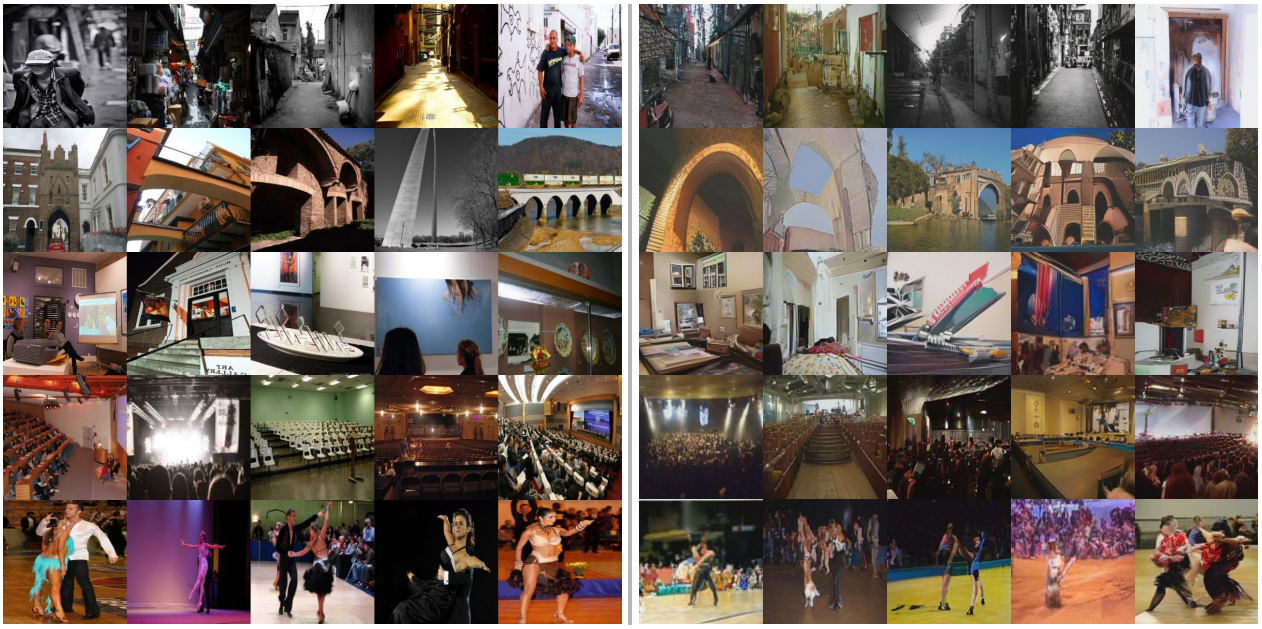
(c) AR transformer with prompt tuning ($S = 1$)

(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.



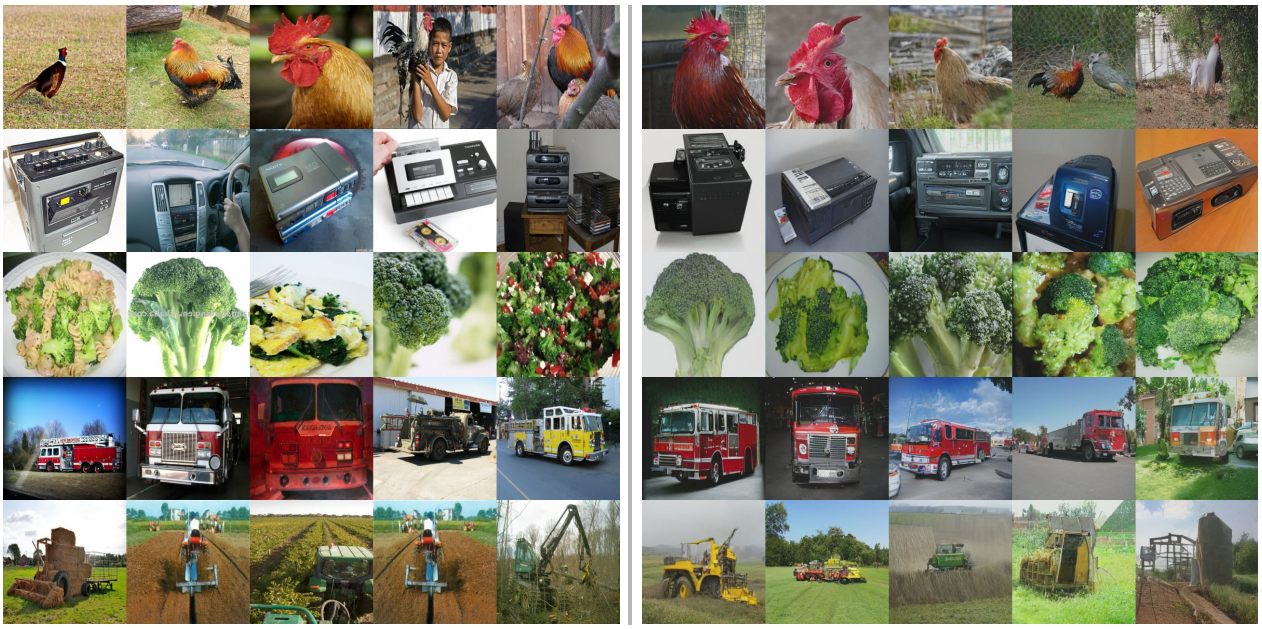
(a) Places, 5-shot, Left: real, Right: generation.

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

Figure 30. Fewshot generation on places.

C.2. Few-shot Generative Transfer

C.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.