Condition-Aware Neural Network for Controlled Image Generation

Han Cai[1*], Muyang Li[1], Qinsheng Zhang[2], Ming-Yu Liu[2], Song Han[1,2]
[1]MIT, [2]NVIDIA

https://github.com/mit-han-lab/efficientvit

2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) | 979-8-3503-5300-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/CVPR52733.2024.00687

Abstract

*We present **Condition-Aware Neural Network (CAN)**, a new method for adding control to image generative models. In parallel to prior conditional control methods, CAN controls the image generation process by dynamically manipulating the weight of the neural network. This is achieved by introducing a condition-aware weight generation module that generates conditional weight for convolution/linear layers based on the input condition. We test CAN on class-conditional image generation on ImageNet and text-to-image generation on COCO. CAN consistently delivers significant improvements for diffusion transformer models, including DiT and UViT. In particular, CAN combined with EfficientViT (CaT) achieves 2.78 FID on ImageNet 512×512, surpassing DiT-XL/2 while requiring 52× fewer MACs per sampling step.*

1. Introduction

Large-scale image [1–4] and video generative models [5, 6] have demonstrated astounding capacity in synthesizing photorealistic images and videos. To convert these models into productive tools for humans, a critical step is adding control. Instead of letting the model randomly generate data samples, we want the generative model to follow our instructions (e.g., class label, text, pose) [7].

Extensive studies have been conducted to achieve this goal. For example, in GANs [8, 9], a widespread solution is to use adaptive normalization [10, 11] that dynamically scales and shifts the intermediate feature maps according to the input condition. In addition, another widely used technique is to use cross-attention [1] or self-attention [12] to fuse the condition feature with the image feature. Though differing in the used operations, these methods share the same underlying mechanism, i.e., adding control by feature space manipulation. Meanwhile, the neural network weight (convolution/linear layers) remains the same for different conditions.

*Work done during an internship at NVIDIA.

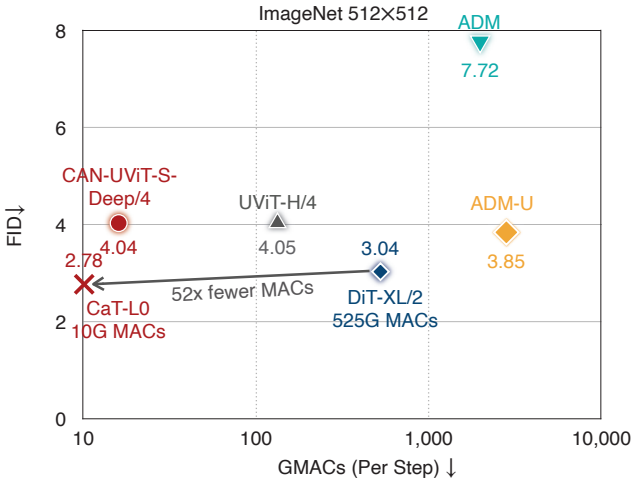

Figure 1. **Comparing CAN Models and Prior Image Generative Models on ImageNet 512×512.** With the new conditional control method, we significantly improve the performance of controlled image generative models. Combining CAN and EfficientViT [13], our CaT model provides 52× MACs reduction per sampling step than DiT-XL/2 [14] without performance loss.

This work aims to answer the following questions: *i) Can we control image generative models by manipulating their weight? ii) Can controlled image generative models benefit from this new conditional control method?*

To this end, we introduce **Condition-Aware Neural Network (CAN)**, a new conditional control method based on weight space manipulation. Differentiating from a regular neural network, CAN introduces an additional weight generation module (Figure 2). The input to this module is the condition embedding, which consists of the user instruction (e.g., class label) and the timestep for diffusion models [15]. The module's output is the conditional weight used to adapt the static weight of the convolution/linear layer. We conduct extensive ablation study experiments investigating the practical use of CAN on diffusion transformers. Our study reveals two critical insights for CAN. First, rather than making all layers condition-aware, we find carefully choosing a subset of modules to be condition-aware (Figure 3) is beneficial

2575-7075/24/$31.00 ©2024 IEEE
DOI 10.1109/CVPR52733.2024.00687

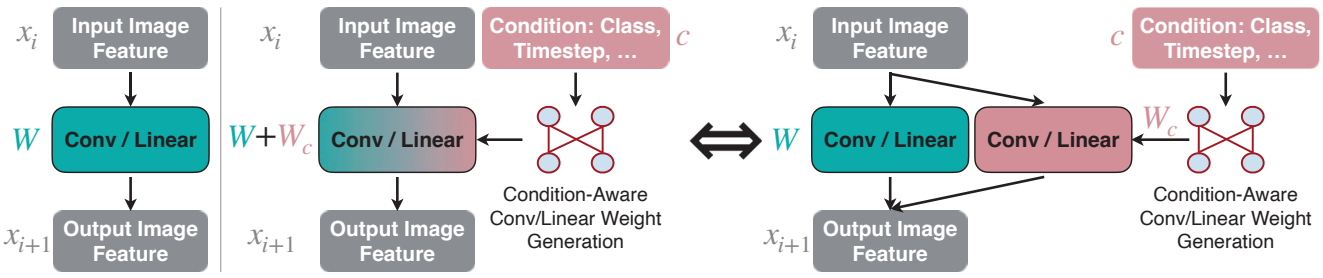

Figure 2. **Illustration of Condition-Aware Neural Network.** *Left:* A regular neural network with static convolution/linear layers. *Right:* A condition-aware neural network and its equivalent form.

for both efficiency and performance (Table 1). Second, we find directly generating the conditional weight is much more effective than adaptively merging a set of base static layers [16] for conditional control (Figure 4).

We evaluate CAN on two representative diffusion transformer models, including DiT [14], and UViT [12]. CAN achieves significant performance boosts for all these diffusion transformer models while incurring negligible computational cost increase (Figure 7). We also find that CAN alone provides effective conditional control for image generative models, delivering lower FID and higher CLIP scores than prior conditional control methods (Table 3). Apart from applying CAN to existing diffusion transformer models, we further build a new family of diffusion transformer models called **CaT** by marrying CAN and EfficientViT [13] (Figure 6). We summarize our contributions as follows:

- We introduce a new mechanism for controlling image generative models. To the best of our knowledge, our work is the first to demonstrate the effectiveness of weight manipulation for conditional control.
- We propose Condition-Aware Neural Network, a new conditional control method for controlled image generation. We also provide design insights to make CAN usable in practice.
- Our CAN consistently improves performances on image generative models, outperforming prior conditional control methods by a significant margin. In addition, CAN can also benefit the deployment of image generative models. Achieving a better FID on ImageNet 512×512, our CAN model requires 52× fewer MACs than DiT-XL/2 per sampling step (Figure 1), paving the way for diffusion model applications on edge devices.

2. Method

2.1. Condition-Aware Neural Network

The image generation process can be viewed as a mapping from the source domain (noise or noisy image) to the target domain (real image). For controlled image generation, the target data distribution is different given different conditions (e.g., cat images' data distribution vs. castle images' data distribution). In addition, the input data distribution is also different for diffusion models [15] at different timesteps. Despite these differences, prior models use the same static convolution/linear layers for all cases, limiting the overall performance due to negative transfer between different sub-tasks [17]. To alleviate this issue, one possible solution is to have an expert model [18] for each sub-task. However, this approach is infeasible for practical use because of the enormous cost. Our condition-aware neural network (CAN) tackles this issue by enabling the neural network to adjust its weight dynamically according to the given condition, instead of explicitly having the expert models.

Figure 2 demonstrates the general idea of CAN. The key difference from a regular neural network is that CAN has an extra conditional weight generation module. This module takes the condition embedding c as the input and outputs the conditional weight W_c. In addition to the conditional weight W_c, each layer has the static weight W. During training and inference, W_c and W are fused into a single kernel call by summing the weight values. This is equivalent to applying W_c and W independently on the input image feature and then adding their outputs.

2.2. Practical Design

Which Modules to be Condition-Aware? Theoretically, we can make all layers in the neural network condition-aware. However, in practice, this might not be a good design. First, from the performance perspective, having too many condition-aware layers might make the model optimization challenging. Second, from the efficiency perspective, while the computational overhead of generating the conditional weight for all layers is negligible[1], it will incur a significant parameter overhead. For example, let's denote the dimension of the condition embedding as d (e.g., 384, 512, 1024, etc) and the model's static parameter size as #params. Using a single linear layer to map from the condition embedding to the conditional weight requires #params $\times d$ parameters,

[1]It is because the sequence length (or spatial size) of the condition embedding is much smaller than the image feature.

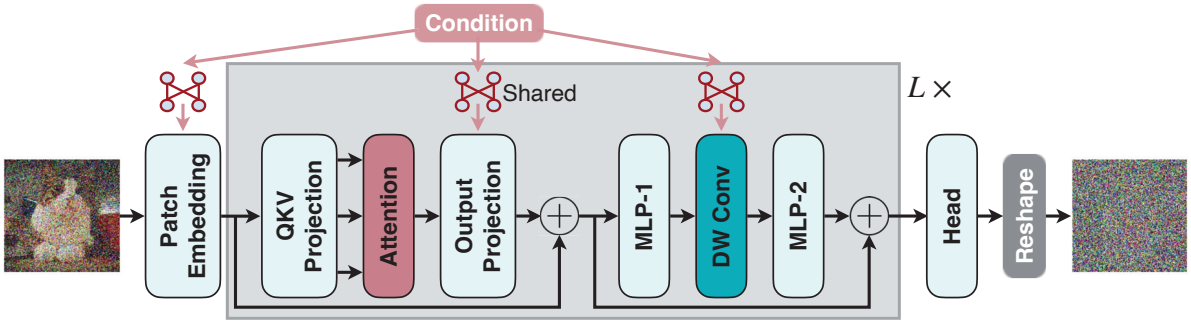

Figure 3. **Overview of Applying CAN to Diffusion Transformer.** The patch embedding layer, the output projection layers in self-attention, and the depthwise convolution (DW Conv) layers are condition-aware. The other layers are static. All output projection layers share the same conditional weight while still having their own static weights.

which is impractical for real-world use. In this work, we carefully choose a subset of modules to apply CAN to solve this issue.

An overview of applying CAN to diffusion transformer [12, 14] is provided in Figure 3. Depthwise convolution [19] has a much smaller parameter size than regular convolution, making it a low-cost candidate to be condition-aware. Therefore, we add a depthwise convolution in the middle of FFN following the previous design [13]. We conduct ablation study experiments on ImageNet 256×256 using UViT-S/2 [12] to determine the set of modules to be condition-aware. *All the models, including the baseline model, have the same architecture. The only distinction is the set of condition-aware modules is different.*

We summarize the results in Table 1. We have the following observations in our ablation study experiments:

- Making a module condition-aware does not always improve the performance. For example, using a static head gives a lower FID and a higher CLIP score than using a condition-aware head (row #2 vs. row #4 in Table 1).
- Making depthwise convolution layers, the patch embedding layer, and the output projection layers condition-aware brings a significant performance boost. It improves the FID from 28.32 to 8.82 and the CLIP score from 30.09 to 31.74.

Based on these results, we chose this design for CAN. Details are illustrated in Figure 3. For the depthwise convolution layers and the patch embedding layer, we use a separate conditional weight generation module for each layer, as their parameter size is small. In contrast, we use a shared conditional weight generation module for the output projection layers, as their parameter size is large. Since different output projection layers have different static weights, we still have different weights for different output projection layers.

CAN vs. Adaptive Kernel Selection. Instead of directly generating the conditional weight, another possible approach is maintaining a set of base convolution kernels and dynamically generating scaling parameters to combine these base kernels [2, 16]. This approach's parameter overhead is smaller than CAN. However, this adaptive kernel selection strategy cannot match CAN's performance (Figure 4). It suggests that dynamic parameterization alone is not the key to better performances; better condition-aware adaptation capacity is critical.

ImageNet 256×256, UViT-S/2		
Models	FID ↓	CLIP Score ↑
1. Baseline (Static Conv/Linear)	28.32	30.09
Making Modules Condition-Aware:		
2. DW Conv	11.18	31.54
3. + Patch Embedding	10.23	31.61
4. or + Head (✗)	12.29	31.40
5. + Output Projection	8.82	31.74
6. or + QKV Projection (✗)	9.71	31.66
7. or + MLP (✗)	10.06	31.62

Table 1. **Ablation Study on Making Which Modules Condition-Aware.**

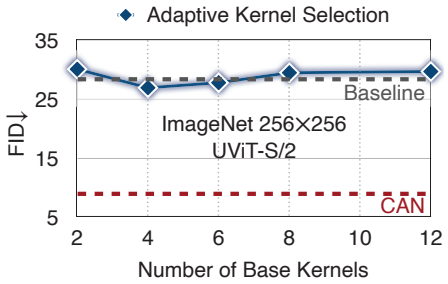

Figure 4. **CAN is More Effective than Adaptive Kernel Selection.**

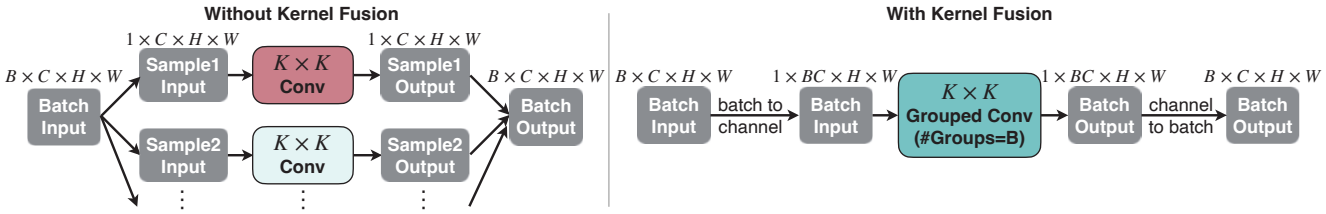

Figure 5. **Practical Implementation of CAN.** *Left:* The condition-aware layers have different weights for different samples. A naive implementation requires running the kernel call independently for each sample, which incurs a large overhead for training and batch inference. *Right:* An efficient implementation for CAN. We fuse all kernel calls into a grouped convolution. We insert a batch-to-channel transformation before the kernel call and add a channel-to-batch conversion after the kernel call to preserve the functionality.

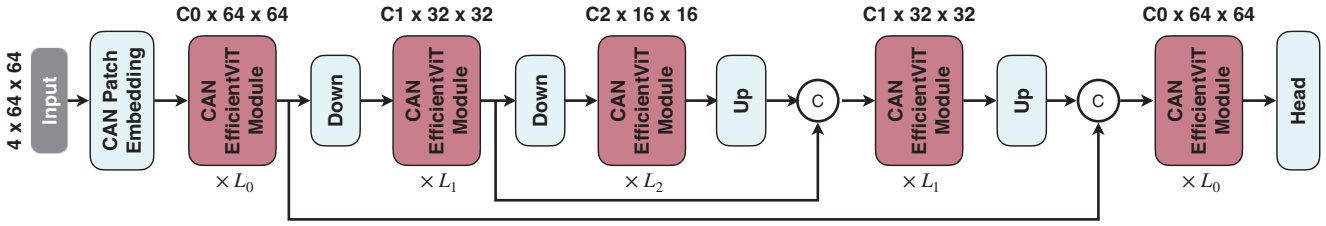

Figure 6. **Macro Architecture of CaT.** Benefiting from EfficientViT's linear computational complexity [13], we can keep the high-resolution stages without efficiency concerns.

Implementation. Since the condition-aware layers have different weights given different samples, we cannot do the batch training and inference. Instead, we must run the kernel calls independently for each sample, as shown in Figure 5 (left). This will significantly slow down the training process on GPU. To address this issue, we employ an efficient implementation for CAN (Figure 5 right). The core insight is to fuse all convolution kernel calls [20] into a grouped convolution where #Groups is the batch size B. We do the batch-to-channel conversion before running the grouped convolution to preserve the functionality. After the operation, we add the channel-to-batch transformation to convert the feature map to the original format.

Theoretically, with this efficient implementation, there will be negligible training overhead compared to running a static model. In practice, as NVIDIA GPU supports regular convolution much better than grouped convolution, we still observe 30%-40% training overhead. This issue can be addressed by writing customized CUDA kernels. We leave it to future work.

3. Experiments

3.1. Setups

Datasets. Due to resource constraints, we conduct class-conditional image generation experiments using the ImageNet dataset and use COCO for text-to-image generation experiments. For large-scale text-to-image experiments [21], we leave them to future work.

Evaluation Metric. Following the common practice, we use FID [22] as the evaluation metric for image quality. In addition, we use the CLIP score [23] as the metric for controllability. We use the public CLIP ViT-B/32 [24] for measuring the CLIP score, following [21]. The text prompts are constructed following CLIP's zero-shot image classification setting [24].

Implementation Details. We apply CAN to recent diffusion transformer models, including DiT [14] and UViT [12]. We follow the training setting suggested in the official paper or GitHub repository. By default, classifier-free guidance [25] is used for all models unless explicitly stated. The baseline models' architectures are the same as the CAN models', having depthwise convolution in FFN layers. We implement our models using Pytorch and train them using A6000 GPUs. Automatic mixed-precision is used during training. In addition to applying CAN to existing models, we also build a new family of diffusion transformers called **CaT** by marrying CAN and EfficientViT [13]. The macro architecture of CaT is illustrated in Figure 6.

3.2. Ablation Study

We train all models for 80 epochs with batch size 1024 (around 100K iterations) for ablation study experiments unless stated explicitly. All models use DPM-Solver [26] with 50 steps for sampling images.

Effectiveness of CAN. Figure 7 summarizes the results of CAN on various UViT and DiT variants. CAN signifi-

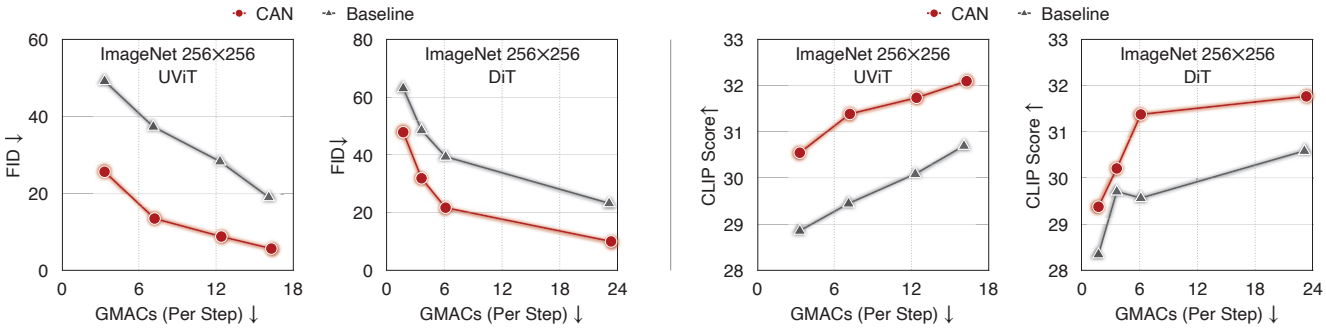

Figure 7. **CAN Results on Different UViT and DiT Variants.** CAN consistently delivers lower FID and higher CLIP score for UViT and DiT variants.

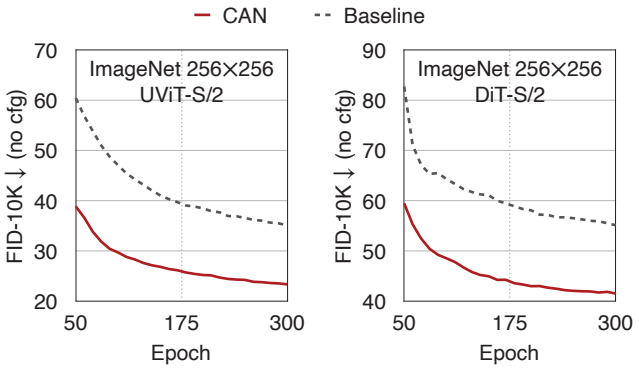

Figure 8. **Training Curve.** CAN's improvements are not due to faster convergence. We observe consistent FID improvements when trained longer.

cantly improves the image quality and controllability over the baseline for all variants. Additionally, these improvements come with negligible computational cost overhead. Therefore, CAN also enhances efficiency by delivering the same FID and CLIP score with lower-cost models.

Figure 8 compares the training curves of CAN and baseline on UViT-S/2 and DiT-S/2. We can see that the absolute improvement remains significant when trained longer for both models. It shows that the improvements are not due to faster convergence. Instead, adding CAN improves the performance upper bound of the models.

Analysis. For diffusion models, the condition embedding contains both the class label and timestep. To dissect which one is more important for the conditional weight generation process, we conduct the ablation study experiments using UViT-S/2, and summarize the results in Table 2. We find that:

- The class label information is more important than the timestep information in the weight generation process. Adding class label alone provides 5.15 lower FID and 0.33 higher CLIP score than adding timestep alone.

ImageNet 256×256, UViT-S/2		
Models	FID ↓	CLIP Score ↑
Baseline	28.32	30.09
CAN (Timestep Only)	15.16	31.26
CAN (Class Label Only)	10.01	31.59
CAN (All)	8.82	31.74

Table 2. **Ablation Study on the Effect of Each Condition for CAN.**

ImageNet 256×256, DiT-S/2		
Models	FID ↓	CLIP Score ↑
Adaptive Normalization	39.44	29.57
CAN Only	26.44	30.54
CAN + Adaptive Normalization	21.76	30.86
ImageNet 256×256, UViT-S/2		
Models	FID ↓	CLIP Score ↑
Attention (Condition as Tokens)	28.32	30.09
CAN Only	8.79	31.75
CAN + Attention (Condition as Tokens)	8.82	31.74

Table 3. **Comparison with Prior Conditional Control Methods.** CAN can work alone without adding other conditional control methods.

- Including the class label and the timestep in the condition embedding delivers the best results. Therefore, we stick to this design in the following experiments.

Comparison with Prior Conditional Control Methods. In prior experiments, we kept the original conditional control methods of DiT (adaptive normalization) and UViT (attention with condition as tokens) unchanged while adding CAN. To see if CAN can work alone and the comparison between

ImageNet 512×512					
Models	FID-50K (no cfg) ↓	FID-50K ↓	#MACs (Per Step) ↓	#Steps ↓	#Params ↓
ADM [27]	23.24	7.72	1983G	250	559M
ADM-U [27]	**9.96**	3.85	2813G	250	730M
UViT-L/4 [12]	-	4.67	77G	50	287M
UViT-H/4 [12]	-	4.05	133G	50	501M
DiT-XL/2 [14]	12.03	3.04	525G	250	675M
CAN (UViT-S-Deep/4)	23.40	4.04	16G	50	185M
CaT-L0	14.25	2.78	10G	20	377M
CaT-L1	10.64	**2.48**	12G	20	486M
ImageNet 256×256					
Models	FID-50K (no cfg) ↓	FID-50K ↓	#MACs (Per Step) ↓	#Steps ↓	#Params ↓
LDM-4 [1]	10.56	3.60	-	250	400M
UViT-L/2 [12]	-	3.40	77G	50	287M
UViT-H/2 [12]	-	2.29	133G	50	501M
DiT-XL/2 [14]	9.62	2.27	119G	250	675M
CAN (UViT-S/2)	16.20	3.52	12G	50	147M
CAN (UViT-S-Deep/2)	11.89	2.78	16G	50	182M
CaT-B0	**8.81**	**2.09**	12G	30	475M

Table 4. **Class-Conditional Image Generation Results on ImageNet.**

CAN and previous conditional control methods, we conduct experiments and provide the results in Table 3. We have the following findings:

- CAN alone can work as an effective conditional control method. For example, CAN alone achieves a 13.00 better FID and 0.97 higher CLIP score than adaptive normalization on DiT-S/2. In addition, CAN alone achieves a 19.53 lower FID and 1.66 higher CLIP score than attention (condition as tokens) on UViT-S/2.
- CAN can be combined with other conditional control methods to achieve better results. For instance, combining CAN with adaptive normalization provides the best results for DiT-S/2.
- For UViT models, combining CAN with attention (condition as tokens) slightly hurts the performance. Therefore, we switch to using CAN alone on UViT models in the following experiments.

3.3. Comparison with State-of-the-Art Models

We compare our final models with other diffusion models on ImageNet and COCO. The results are summarized in Table 4 and Table 6. For CaT models, we use UniPC [28] for sampling images to reduce the number of sampling steps.

Class-Conditional Generation on ImageNet 256×256. As shown in Table 4 (bottom), with the classifier-free guidance (cfg), our CaT-B0 achieves 2.09 FID on ImageNet 256×256, outperforming DiT-XL/2 and UViT-H/2. More importantly, our CaT-B0 is much more compute-efficient than these models: 9.9× fewer MACs than DiT-XL/2 and 11.1× fewer MACs than UViT-H/2. Without the classifier-free guidance, our CaT-B0 also achieves the lowest FID among all compared models (8.81 vs. 9.62 vs. 10.56).

Class-Conditional Generation on ImageNet 512×512. On the more challenging 512×512 image generation task, we observe the merits of CAN become more significant. For example, our CAN (UViT-S-Deep/4) can match the performance of UViT-H (4.04 vs. 4.05) while only requiring 12% of UViT-H's computational cost per diffusion step. Additionally, our CaT-L0 delivers 2.78 FID on ImageNet 512×512, outperforming DiT-XL/2 (3.04 FID) that requires 52× higher computational cost per diffusion step. In addition, by slightly scaling up the model, our CaT-L1 further improves the FID from 2.78 to 2.48.

In addition to computational cost comparisons, Table 5 compares CaT-L0 and DiT-XL/2 on NVIDIA Jetson AGX Orin. The latency is measured with TensorRT, fp16. Delivering a better FID on ImageNet 512×512, CaT-L0 combined with a training-free fast sampling method (UniPC) runs 229× faster than DiT-XL/2 on Orin. It is possible to further push the efficiency frontier by combining CaT with training-based few-step methods [29, 30], showing the potential of enabling real-time diffusion model applications on

ImageNet 512×512			
Models	#Steps ↓	Orin Latency ↓	FID ↓
DiT-XL/2 [14]	250	45.9s	3.04
CaT-L0	20	0.2s	2.78

Table 5. **NVIDIA Jetson AGX Orin Latency vs. FID.** Latency is profiled with TensorRT and fp16.

COCO 256×256			
Models	FID-30K ↓	#MACs ↓	#Params ↓
Friro	8.97	-	512M
UViT-S/2	5.95	15G	45M
UViT-S-Deep/2	5.48	19G	58M
CaT-S0	5.49	3G	169M
CaT-S1	**5.22**	7G	307M

Table 6. **Text-to-Image Generation Results on COCO 256×256.**

edge devices.

Apart from quantitative results, Figure 9 provides samples from the randomly generated images by CAN models, which demonstrate the capability of our models in generating high-quality images.

Text-to-Image Generation on COCO 256×256. For text-to-image generation experiments on COCO, we follow the same setting used in UViT [12]. Specifically, models are trained from scratch on the COCO 2014 training set. Following UViT [12], we randomly sample 30K text prompts from the COCO 2014 validation set to generate images and then compute FID. We use the same CLIP encoder as in UViT for encoding the text prompts. The results are summarized in Table 6. Our CaT-S0 achieves a similar FID as UViT-S-Deep/2 while having much less computational cost (19G MACs → 3G MACs). It justifies the generalization ability of our models.

4. Related Work

Controlled Image Generation. Controlled image generation requires the models to incorporate the condition information into the computation process to generate related images. Various techniques have been developed in the community for controlled image generation. One typical example is adaptive normalization [11] that regresses scale and shift parameters from the condition information and applies the feature-wise affine transformation to influence the output. Apart from adaptive normalization, another typical approach is to treat condition information as tokens and use either cross-attention [1] or self-attention [12] to fuse the condition information. ControlNet [7] is another representative technique that uses feature-wise addition to add extra control to pre-trained text-to-image diffusion models. In parallel to these techniques, this work explores another mechanism for adding conditional control to image generative models, i.e., making the weight of neural network layers (conv/linear) condition-aware.

Dynamic Neural Network. Our work can be viewed as a new type of dynamic neural network. Apart from adding conditional control explored in this work, dynamically adapting the neural network can be applied to many deep learning applications. For example, CondConv [16] proposes to dynamically combine a set of base convolution kernels based on the input image feature to increase the model capacity. Similarly, the mixture-of-expert [18] technique uses a gating network to route the input to different experts dynamically. For efficient deployment, once-for-all network [31] and slimmable neural network [32] dynamically adjust the neural network architecture according to the given efficiency constraint to achieve better tradeoff between efficiency and accuracy.

Weight Generating Networks. Our conditional weight generation module can be viewed as a new kind of weight generation network specially designed for adding conditional control to generative models. There are some prior works exploiting weight generation networks in other scenarios. For example, [33] proposes to use a small network to generate weights for a larger network. These weights are the same for every example in the dataset for better parameter efficiency. Additionally, weight generation networks have been applied to neural architecture search to predict the weight of a neural network given its architecture [34] to reduce the training and search cost [35] of neural architecture search.

Efficient Deep Learning Computing. Our work is also connected to efficient deep learning computing [36, 37] that aims to improve the efficiency of deep learning models to make them friendly for deployment on hardware. State-of-the-art image generative models [1, 2, 4, 21] have enormous computation and memory costs, which makes it challenging to deploy them on resource-constrained edge devices while maintaining high quality. Our work can improve the efficiency of the controlled generative models by delivering the same performance with fewer diffusion steps and lower-cost models. For future work, we will explore combining our work and efficient deep learning computing techniques [31, 38] to futher boost efficiency.

5. Conclusion

In this work, we studied adding control to image generative models by manipulating their weight. We introduced a

Figure 9. **Samples of Generated Images by CAN Models.**

new conditional control method, called **Condition-Aware Neural Network (CAN)**, and provided efficient and practical designs for CAN to make it usable in practice. We conducted extensive experiments on class-conditional generation using ImageNet and text-to-image generation using COCO to evaluate CAN's effectiveness. CAN delivered consistent and significant improvements over prior conditional control methods. We also built a new family of diffusion transformer models by marrying CAN and EfficientViT. For future work, we will apply CAN to more challenging tasks like large-scale text-to-image generation, video generation, etc.

Acknowledgments

This work is supported by MIT-IBM Watson AI Lab, Amazon, MIT Science Hub, and National Science Foundation.

References

[1] 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. 1, 6, 7

[2] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10124–10134, 2023. 3, 7

[3] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

[4] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 1, 7

[5] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. 1

[6] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 1

[7] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 1, 7

[8] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 1

[9] 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. 1

[10] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017. 1

[11] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 1, 7

[12] Fan Bao, Chongxuan Li, Yue Cao, and Jun Zhu. All are worth words: a vit backbone for score-based diffusion models. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022. 1, 2, 3, 4, 6, 7

[13] Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17302–17313, 2023. 1, 2, 3, 4

[14] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 1, 2, 3, 4, 6, 7

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 2

[16] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in neural information processing systems*, 32, 2019. 2, 3, 7

[17] Sen Wu, Hongyang R Zhang, and Christopher Ré. Understanding and improving information transfer in multi-task learning. *arXiv preprint arXiv:2005.00944*, 2020. 2

[18] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 2, 7

[19] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 3

[20] 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. 4

[21] Zhan Shi, Xu Zhou, Xipeng Qiu, and Xiaodan Zhu. Improving image captioning with better use of captions. *arXiv preprint arXiv:2006.11807*, 2020. 4, 7

[22] 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. 4

[23] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021. 4

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 4

[25] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 4

[26] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022. 4

[27] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 6

[28] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024. 6

[29] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 6

[30] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. 6

[31] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 7

[32] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018. 7

[33] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. 7

[34] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*, 2017. 7

[35] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 7

[36] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 7

[37] Han Cai, Chuang Gan, Ji Lin, and Song Han. Network augmentation for tiny deep learning. *arXiv preprint arXiv:2110.08890*, 2021. 7

[38] Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17535–17545, 2023. 7