

Real-Time Portrait Stylization on the Edge

Yanyu Li^{*1}, Xuan Shen^{*1}, Geng Yuan^{*1}, Jiexiong Guan²,
Wei Niu², Hao Tang³, Bin Ren², Yanzhi Wang¹

¹Northeastern University

²College of William & Mary

³CVL, ETH Zurich

{li.yanyu, shen.xu, yuan.geng, yanz.wang}@northeastern.edu,
{jguan, wniu}@email.wm.edu, bren@cs.wm.edu, hao.tang@vision.ee.ethz.ch

Abstract

In this work we demonstrate real-time portrait stylization, specifically, translating self-portrait into cartoon or anime style on mobile devices. We propose a latency-driven differentiable architecture search method, maintaining realistic generative quality. With our framework, we obtain 10× computation reduction on the generative model and achieve real-time video stylization on off-the-shelf smartphone using mobile GPUs.

1 Introduction

Thanks to hardware advancement, varieties of AI applications have been enabled on portable smart devices, such as foreground segmentation, face recognition, etc. In this work, we investigate portrait stylization which is a popular feature in social media Apps, transferring self portraits into a desired style, such as cartoon [Andersson and Arvidsson, 2020], anime [Kim *et al.*, 2020; Li *et al.*, 2021], etc.

Portrait stylization can be categorized as classic image-to-image translation, which is often achieved by conditional Generative Adversarial Networks (GANs) [Isola *et al.*, 2017; Zhu *et al.*, 2017]. In GAN training, a generator learns to generate a fake instance and fool the discriminator, while the discriminator takes true and generated images as input and learns to distinguish them. Consequently, in the portrait stylization task, the generator is utilized to map a portrait photo to the desired style domain during inference. Compared to naive paired supervised training, GANs exhibits stunning generative quality, e.g., sharp and realistic details, diverse features.

However, it still remains challenging to enable real-time face stylization to process videos on a mobile device, and the reason comes two-fold. Firstly, typically following an encoder-decoder design, image translation models suffer from high computation complexity, especially on high-resolution images. Secondly, GAN training is difficult and unstable, suffering from loss divergence and mode collapse. As a result, existing compression techniques are difficult to integrate into GAN training and preserve generative quality.

In this work, we propose a compiler-aware differentiable architecture search framework. We measure the latency of the

^{*}These authors contributed equally to this work.

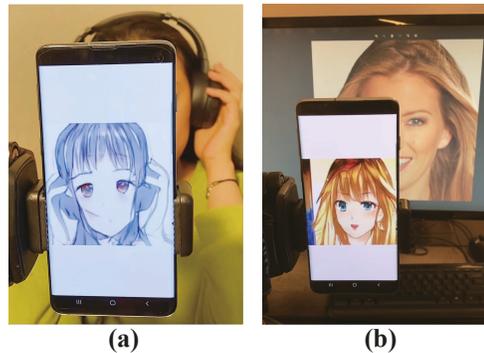


Figure 1: Demonstration of on-mobile real-time stylization for (a) real person and (b) static image. The input video stream is captured by the mobile phone’s camera in a real-time fashion. The full demo video is attached in the link¹.

building blocks with sufficient configurations (channels, feature sizes), and train a neural network to predict the latency. We show that a simple MLP speed model can make accurate predictions. In order to search compact architectures, we integrate learnable parameters in the GAN generator and regularize them by speed constraints to reduce the model width and depth. With the speed model that maps architecture parameters to latency, the speed penalty is differentiable so that we can perform search with end-to-end training. Plus, different from prior work, we do not select preserved/eliminated model weights by magnitude. Instead, we apply straight through estimator on architecture parameters to sparsify them into $\{0, 1\}$. The benefit is two-fold. The remained weights are represented by 1s so that we can easily predict the latency of a certain state. In addition, the gradients of pruned weights are completely zeroed out so that their functionalities are preserved. As a result, pruned weights are always ready to be reverted back to contribute to accuracy during exploration. This is especially important in GAN search because of the unstable training process.

Overall, our contributions include:

- We develop a latency-driven differentiable architecture search for GANs. Our sophisticated pipeline addresses the difficulty in GAN searching, achieves unprecedented compression rate while preserving generative quality.

- To the best of our knowledge, we are the first to achieve real-time portrait stylization on mobile phones. Mobile demos are attached in the link¹.

2 Background

Image-to-Image Style Transfer. [Isola *et al.*, 2017; Liu *et al.*, 2017; Huang *et al.*, 2018; Lee *et al.*, 2018; Liu *et al.*, 2019; Park *et al.*, 2020; Kim *et al.*, 2020; Li *et al.*, 2021; Chong and Forsyth, 2021; Tang *et al.*, 2019] aim to translate a image from the source domain to match certain styles in the target domain, while preserving semantics of the origin image. Early works train the generative model with paired data [Isola *et al.*, 2017], however they cannot be applied to more common unpaired datasets. Later work [Zhu *et al.*, 2017] proposed a cycle consistency loss to train on unpaired data domains, inspiring lots of successor research on image stylization [Park *et al.*, 2020; Kim *et al.*, 2020; Li *et al.*, 2021; Chong and Forsyth, 2021; Tang *et al.*, 2021].

As for the domain of cartoon or anime, UGATIT [Kim *et al.*, 2020] released the selfie2anime benchmark and develop an adaptive mixture of instance and layer normalization. AniGAN [Li *et al.*, 2021] released a larger scale face2anime dataset and studied style-guided anime translation. GNR [Chong and Forsyth, 2021] refine the content and style to produce controllable and diverse synthesis.

Compressing GANs. Because of wide applications, the compression of GAN has drawn research attention [Fu *et al.*, 2020; Wang *et al.*, 2020; Li *et al.*, 2020; Chen *et al.*, 2021; Jin *et al.*, 2021]. Recent work [Li *et al.*, 2020] proposed to incorporate neural architecture search (NAS) and feature level knowledge distillation (KD). [Jin *et al.*, 2021] integrated an Inception-like residual block and performed self-distillation.

3 Compiler-Aware Architecture Search

3.1 GAN Basics

We follow CycleGAN [Zhu *et al.*, 2017] and UGATIT [Kim *et al.*, 2020] paradigm to develop our architecture search for stylization. According to CycleGAN, we learn mapping functions between two unpaired domain $X = \{x_i\}_{i=1}^N$ and $Y = \{y_j\}_{j=1}^M$. There are two generators in inverse direction $G: X \rightarrow Y$ and $F: Y \rightarrow X$, as well as two discriminators D_X and D_Y . Note that we refer G as the generator mapping portraits to stylized images, which is the only required model during inference.

Adversarial Loss. We match the distribution of translated instances with the target domain as follows.

$$L_{gan}^X = \mathbb{E}_{y \sim Y} [D_Y^2(y)] + \mathbb{E}_{x \sim X} [(1 - D_Y(G(x)))^2]. \quad (1)$$

Cycle Consistency Loss. To minimize reconstruction error,

$$L_{cyc} = \mathbb{E}_{x \sim X} [|F(G(x)) - x|_1] + \mathbb{E}_{y \sim Y} [|G(F(y)) - y|_1]. \quad (2)$$

Note that we also incorporate the identity loss L_{id} and CAM loss L_{CAM} as proposed in UGATIT, for simplicity we

skip detailed formulations here and please refer to [Kim *et al.*, 2020].

Our overall GAN objective is:

$$L = \lambda_1 L_{gan}^X + \lambda_1 L_{gan}^Y + \lambda_2 L_{cyc} + \lambda_3 L_{id} + \lambda_4 L_{CAM}, \quad (3)$$

where $\lambda_1=1, \lambda_2=10, \lambda_3=10, \lambda_4=1000$ are the hyperparameters to control each loss.

3.2 Layerwise Width Search

Width search is performed for each CONV layer. We choose the supernet from [Zhu *et al.*, 2017; Kim *et al.*, 2020], which is a commonly employed generator. In order to create a learnable binary mask, we insert a depth-wise 1×1 CONV layer following each CONV layer to be pruned, as shown below,

$$\mathbf{a}_l^n = \mathbf{v}_l^n \odot (\mathbf{w}_l^n \odot \mathbf{a}_{l-1}^n), \quad (4)$$

where \odot denotes the convolution operation. $\mathbf{w}_l^n \in R^{o \times i \times k \times k}$ is the weight parameters in the l^{th} CONV layer of the n^{th} block, with o output channels, i input channels, and kernels of size $k \times k$. $\mathbf{a}_l^n \in R^{B \times o \times s \times s'}$ represents the output features of l^{th} layer (with the trainable mask), with o channels and $s \times s'$ feature size. B denotes the batch size. $\mathbf{v}_l^n \in R^{o \times 1 \times 1 \times 1}$ is the corresponding weights of the depth-wise CONV layer (i.e., the mask layer).

Larger elements of \mathbf{m}_l^n mean that the corresponding channels should be preserved while smaller elements indicate that the corresponding channels should be pruned. Formally, we use a threshold t to convert \mathbf{m}_l^n into a binary mask as below,

$$\mathbf{b}_l^n = \begin{cases} 1, & \mathbf{v}_l^n > t. \\ 0, & \mathbf{v}_l^n \leq t. \end{cases} \quad (\text{element-wise}), \quad (5)$$

where $\mathbf{b}_l^n \in \{0, 1\}^{o \times 1 \times 1 \times 1}$ is the binarized \mathbf{v}_l^n . Typically, we initialize \mathbf{v}_l^n with 1, and the adjustable t is set to 0.5. In order to make the mask differentiable to enable backpropagation, we utilize Straight Through Estimator (STE) [Bengio *et al.*, 2013; Chang *et al.*, 2020] as shown below,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}_l^n} = \frac{\partial \mathcal{L}}{\partial \mathbf{v}_l^n}. \quad (6)$$

Our trainable binary mask has the following advantages: (i) The mask can be trained along with the network parameters via gradient descent, thus saving search cost compared to NAS methods [Zoph and Le, 2017; Zhong *et al.*, 2018]. (ii) Different from previous methods [Han *et al.*, 2015; Yu *et al.*, 2017; He *et al.*, 2017; Guan *et al.*, 2020], which determine the pruning according to the parameter magnitudes, we decouple the parameter magnitudes of CONV or BN layer from pruning, and utilize independent mask layers, thus the remained parameters are not harmed. (iii) The discrete values can directly provide the width information for each CONV layer, which is compatible with speed prediction.

3.3 Length Search by Block

Note that although per-layer width search may also converge to zero width, which eliminates the entire block, we find that there are usually a few channels left in each block preventing us to remove the entire block. Plus, the pruning indicator for each layer cannot represent the latency reduction of entirely

¹The link for the demo video: <https://youtu.be/2SFAVwaymvQ>



Figure 2: Stylization results on face2anime dataset. Top: source image. Middle: styles generated by baseline UGATIT model with 57.1 GMACs. Bottom: our searched model under 5.56 GMACs.



Figure 3: Stylization results on photo2cartoon dataset released (partly) by MiniVision. Top: source image. Note that the source domain data (Asian women) comes from generated fake data by StyleGAN2. Bottom: our searched model under 1.34 GMACs.

pruned blocks. Thus it is necessary to perform length search separately from width search.

We construct two paths in each customized residual block, one is the masked convolution block and the other is skip connection. In the aggregation layer of the dual paths, we integrate binarized variables β_1 and β_2 , then the forward computation can be represented as follows,

$$\mathbf{a}^n = \beta_1 \cdot \mathbf{a}^{n-1} + \beta_2 \cdot \mathbf{a}_L^n. \quad (7)$$

The aggregation layer contains two trainable parameters α_1 and α_2 , shares similar STE recipe with width parameter. In the forward pass, it selects the skip path or the masked convolution path based on the relative relationship of α_1 and α_2 ,

$$\beta_1 = 0 \text{ and } \beta_2 = 1, \text{ if } \alpha_1 \leq \alpha_2. \quad (8)$$

$$\beta_1 = 1 \text{ and } \beta_2 = 0, \text{ if } \alpha_1 > \alpha_2. \quad (9)$$

3.4 Speed Prediction with Speed Model

We take inference speed on mobile GPUs to constrain the optimization. A DNN-based speed model is adopted to predict the inference speed of the block based on its architecture configurations. Then the final predicted latency is accumulated by the aggregated blocks so that we can compute the latency loss $L_{latency}$ and integrate it into the searching pipeline.

$$L_{search} = L + L_{latency}(\mathbf{v}, \alpha). \quad (10)$$

The trained speed model is accurate in predicting the speed of different layer widths in the block (with 5% error at most).

Face2anime	FID ↓	MACs ($\times 10^9$)
CycleGAN [Zhu <i>et al.</i> , 2017]	50.09	56.8
UGATIT [Kim <i>et al.</i> , 2020]	42.84	57.1
MUNIT [Huang <i>et al.</i> , 2018]	43.75	77.3
FUNIT [Liu <i>et al.</i> , 2019]	56.81	-
DRIT [Lee <i>et al.</i> , 2018]	70.59	-
AniGAN [Li <i>et al.</i> , 2021]	38.45	-
Ours	57.32	5.56

Table 1: Quantitative evaluation of our searched stylization model on Face2anime dataset. Our model perform inference on video at 18 FPS on SAMSUNG Galaxy S10 smartphone using mobile GPU.

4 Experiments and Demonstration

We conduct experiments on Face2anime dataset published by [Li *et al.*, 2021]. Face2anime consists of 17,796 images. We set input size to 256×256 . Further, we also demonstrate realistic quality on cartoon stylization released by MiniVision, along with Asian women face training data generated by [Karras *et al.*, 2020].

Experiment Setups. As for the supernet, we search from [Kim *et al.*, 2020]. Learning rate is set to 2×10^{-4} for both generator and discriminator, with Adam optimizer. Learning rate is fixed for the first 30k iterations and then linearly decayed to zero in another 30k iterations.

Performance Evaluation. We quantitatively compare our searched stylization model with representative works in the quality of created images (FID) and computation costs (MACs). As shown in Table 1, our generative model achieves $10 \times$ computation reduction and preserves generative quality. Thanks to the significant computation reduction, we achieve high quality real-time stylization on mobile.

Results Visualization and On-mobile Demonstration. Figure 2 shows the visualization of the created images on face2anime test dataset and the comparison of our 5.56 GMACs model with the baseline UGATIT model with 57.1 GMACs. Figure 3 shows the created images using photo2cartoon test dataset. With simpler styles, our search method further reduce the model size to 1.34 GMACs. We also demonstrate our method on the mobile device, as shown in Figure 1. The full demo video is available in the link¹.

Acknowledgments

This research is partially supported by National Science Foundation CNS-1909172, CCF-1919117, and CCF-2047516 (CAREER).

References

- [Andersson and Arvidsson, 2020] Filip Andersson and Simon Arvidsson. Generative adversarial networks for photo to hayao miyazaki style cartoons. *arXiv preprint arXiv:2005.07702*, 2020.
- [Bengio *et al.*, 2013] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- [Chang *et al.*, 2020] Sung-En Chang, Yanyu Li, Mengshu Sun, Runbin Shi, Hayden K-H So, Xuehai Qian, Yanzhi Wang, and Xue Lin. Mix and match: A novel fpga-centric deep neural network quantization framework. *arXiv preprint arXiv:2012.04240*, 2020.
- [Chen *et al.*, 2021] Xuxi Chen, Zhenyu Zhang, Yongduo Sui, and Tianlong Chen. Gans can play lottery tickets too. *arXiv preprint arXiv:2106.00134*, 2021.
- [Chong and Forsyth, 2021] Min Jin Chong and David Forsyth. Gans n’roses: Stable, controllable, diverse image to image translation (works for videos too!). *arXiv preprint arXiv:2106.06561*, 2021.
- [Fu *et al.*, 2020] Yonggan Fu, Wuyang Chen, Haotao Wang, Haoran Li, Yingyan Lin, and Zhangyang Wang. Autogandistiller: Searching to compress generative adversarial networks. In *ICML*, 2020.
- [Guan *et al.*, 2020] Yushuo Guan, Ning Liu, Pengyu Zhao, Zhengping Che, Kaigui Bian, Yanzhi Wang, and Jian Tang. Dais: Automatic channel pruning via differentiable annealing indicator search. *arXiv preprint arXiv:2011.02166*, 2020.
- [Han *et al.*, 2015] Song Han, Jeff Pool, et al. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015.
- [He *et al.*, 2017] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017.
- [Huang *et al.*, 2018] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.
- [Isola *et al.*, 2017] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.
- [Jin *et al.*, 2021] Qing Jin, Jian Ren, Oliver J Woodford, Jiazhao Wang, Geng Yuan, Yanzhi Wang, and Sergey Tulyakov. Teachers do more than teach: Compressing image-to-image models. In *CVPR*, 2021.
- [Karras *et al.*, 2020] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020.
- [Kim *et al.*, 2020] Junho Kim, Minjae Kim, Hyeonwoo Kang, and Kwang Hee Lee. U-gat-it: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. In *ICLR*, 2020.
- [Lee *et al.*, 2018] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *ECCV*, 2018.
- [Li *et al.*, 2020] Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In *CVPR*, 2020.
- [Li *et al.*, 2021] Bing Li, Yuanlue Zhu, Yitong Wang, Chia-Wen Lin, Bernard Ghanem, and Linlin Shen. Anigan: Style-guided generative adversarial networks for unsupervised anime face generation. *IEEE TMM*, 2021.
- [Liu *et al.*, 2017] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *NeurIPS*, 2017.
- [Liu *et al.*, 2019] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *ICCV*, 2019.
- [Park *et al.*, 2020] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *ECCV*, 2020.
- [Tang *et al.*, 2019] Hao Tang, Dan Xu, Nicu Sebe, Yanzhi Wang, Jason J Corso, and Yan Yan. Multi-channel attention selection gan with cascaded semantic guidance for cross-view image translation. In *CVPR*, 2019.
- [Tang *et al.*, 2021] Hao Tang, Hong Liu, Dan Xu, Philip HS Torr, and Nicu Sebe. Attentiongan: Unpaired image-to-image translation using attention-guided generative adversarial networks. *IEEE TNNLS*, 2021.
- [Wang *et al.*, 2020] Haotao Wang, Shupeng Gui, Haichuan Yang, Ji Liu, and Zhangyang Wang. Gan slimming: All-in-one gan compression by a unified optimization framework. In *ECCV*, 2020.
- [Yu *et al.*, 2017] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *CVPR*, 2017.
- [Zhong *et al.*, 2018] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *CVPR*, 2018.
- [Zhu *et al.*, 2017] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.
- [Zoph and Le, 2017] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.