

## \* Abstract

CycleGAN은 최근 이미지 합성이 큰 발전이 있었지만, mobileNet-V3와 같은 다른 CNN의 비해 계산량이 크다.

따라서 논문에서는 보편적인 CycleGAN의 대신, inference time과 model size를 줄일 수 있는 방식을 제시한다.

따로 CNN compression을 진행하는 것은 GAN의 훈련이 어렵고, generator의 구조 때문에 성능의 저하를 야기한다.

1. GAN training의 안정화를 위해, 원래 model의 중간 representation들을 compress model로 transfer

Unpair, pair unifying

2. CNN을 그대로 쓰기보다 NAS로 architecture를 찾았다.

또한 효율적인 search를 위해, weight sharing으로 model training과 architecture search를 병합한다.

⇒ Compression의 효율화

## \* Introduction.

GAN은 human interactive는 복잡하며 많이 쓰이는지, edge device는 hardware로 훈련하기 때문에 bottleneck이 생긴다.

CycleGAN 등의 model들은 high computation이다. MobileNet의 몇십배.

⇒ GAN Compression 제안

Compression은 2가지 어려움이 있다.

1. Unstable to train GAN (특히, unpairing)

2. generator의 경우 기본 CNN과 달리 기존 CNN 구조를 reuse하기 힘들다.

따라서 먼저 중간 representation을 우선 teacher의 해당되는 student layer로 transfer한다.

그리고 teacher와 맞지 않은 student의 unpair면, pseudo pair를 만들어서, 훈련시에는 것이 더 좋다.

그 다음, fewer cost로 찾기위해 NAS를 사용한다.

가능한 모든 channel을 포함한 once-for-all network의 architecture search와 training을 병합한다.

One-for-all network는 weight sharing을 통해 많은 subnetwork를 만들 수 있고, 따로 재학습없이 evaluate 할 수 있다.

⇒ 모든 method의 적용 가능

## \* Related work

### - Model Acceleration

Hardware의 효율적인 deep learning을 위해 많은 노력이 있었다.

Network의 필요치 않은 부분을 자르기 위해, network connection의 pruning을 할 수 있다.

하지만 속도를 가속화하기 위해서는, 특수한 hardware가 필요하다.

혹속 연구로 network weight 자체의 대한 pruning도 존재한다.

AutoML for Model Compression (AMC)은 강화학습으로 각 layer의 pruning 비율을 결정한다.

→ evolutionary search et co-evolutionary searching 사용한 논문도 있다.

이런 method들은 특정 model에 맞춰져 있다.

또 다른 model-agnostic한 conditional GAN이 활용할 수 있는 method를 제시한다.

→ cGAN이 대안 시장 지침이 있고 다양한 앱들을 보인다.

### - knowledge distillation

큰 teacher network에서 작은 student network로 transfer를 위한 knowledge distillation

student network는 teacher network를 모방하도록 훈련됩니다.

최근 uncGAN의 distillation을 적용한 경우도 있다.

alec2021 cGAN이 집중했다.

### - Neural architecture search

NAS는 NN의 설계를 성능적으로 찾습니다.

설세 네트을 초기화하기 위해, 여러 subnetwork가 가능성을 공유할 수 있는 one-shot NAS가 있다.

GAN을 위한 NAS를 연구했다.

## \* Method

cGAN을 compress 하는 것은 훈련의 unstable 과 현재 존재하는 CNN Compression과는 architecture의

것이다. 또한, 적용하기 힘들다는 문제점이 있다.

→ 새로운 방법 개시.

### - Training Objective

#### 1. Unifying unpaired and paired learning

cGAN은 source domain  $X$ 와 target domain  $T$ 의 mapping  $G$ 를 훈련 시킨다.

Training data는 unpair와 pair 두 가지가 있다.

많은 Objective function이 paired/unpaired 모두에 대해 훈련 할 수 있도록 설계되는데,

이로 인해 general-purpose compression framework를 구현하기가 힘들다.

(1) teacher structure. 어떤 train 방식이 전제일지 model compression의 도도록

paired/unpair를 unify 한다.

Origin teacher generator인  $G'$ 가 주어지면, unpair를 pair로 바꿀 수 있다.

즉, unpair된  $G'$ 의 output을 compression generator  $G$ 의 GT로 쓸 수 있다.

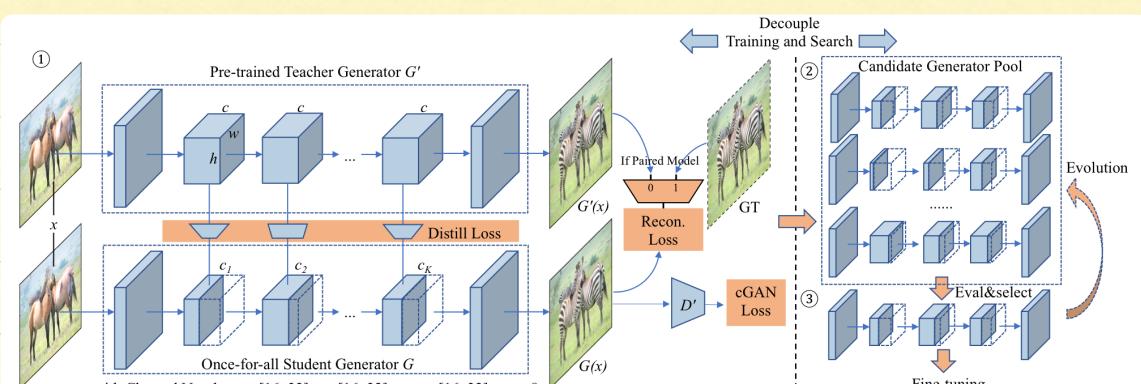


Fig. 3: GAN Compression framework: ① Given a pre-trained teacher generator  $G'$ , we distill a smaller once-for-all student generator  $G$  that contains all possible channel numbers through weight sharing. We choose different channel numbers  $\{c_k\}_{k=1}^K$  for the student generator  $G$  at each training step. ② We then extract many sub-generators from the once-for-all generator and evaluate their performance. No retraining is needed, which is the advantage of the once-for-all generator. ③ Finally, we choose the best sub-generator given the compression ratio target or performance target (FID or mIoU) using either brute-force or evolutionary search method. Optionally, we perform additional fine-tuning to obtain the final model.

$$L_{\text{recon}} = \begin{cases} E_{x,y} \|G(x) - y\|, & \text{if paired cGAN} \\ E_x \|G(x) - G'(x)\|, & \text{if unpaired cGAN} \end{cases}$$

위의 pseudo pairs는 unpair train보다 성능이 안정적이다.

## 2. Inheriting the teacher discriminator

$G$ 는 compression하는 것을 목표로 하지만,  $D$ 는  $y$ 를  $G$ 의 출력한 뿐만 아니라 학습하기 때문이다.

훈련된 GAN이 다른 정보가 있다.

따라서 동일한  $D$ 의 pre-trained weight를 사용하여 fine-tuning 시킨다.

실제적으로, pre-trained  $D$ 가 훈련도 지속시간이不稳定한 random initialize된  $D$ 보다 좋다.

$$L_{\text{cGAN}} = E_{x,y} [\log D(x, y)] + E_x [\log (1 - D(x, G(x)))]$$

## 3. Intermediate feature distillation.

model compression의 대체적으로 쓰이는 방식은 output layer의 logit의 블록을 맞추는 knowledge distillation이 많이 쓰인다.

하지만 cGAN의 output의 확률 분포가(또는 deterministic image) 대체로

teacher pixel과 dark knowledge를 distill하기 쉽지 않다.

혹시 paired의 경우 G가 비슷해서  $G(x)$ 가 다른 점이 없기 때문에 잘 동작하지 않는다.

따라서 teacher의 같은 layer를 matching 시킨다.

$$L_{\text{distill}} = \sum_{t=1}^T \|G_t(x) - f_t(G'_t(x))\|_2$$

$G_t$ 는  $t$  layer의 feature activation이고,  $f_t$ 는 teacher의 student model channel  $t$ 를 맞추는  $1 \times 1$  conv이다.

$G_t$ 와  $f_t$ 를 optimizes,  $L_{\text{distill}}$ 을 minimize 한다.

$$\text{Full Objective: } L = L_{\text{GAN}} + \lambda_{\text{recon}} L_{\text{recon}} + \lambda_{\text{distill}} L_{\text{distill}}$$

### - Efficient Generator Design space

knowledge distillation의 architecture의 선택은 중요하다.

GAN의 channel을 흐리는 것은 compact한 student를 생성하기에 도움이 있다.

↳ 성능이 높고 학습이 빠름

↳ GAN의 architecture가 보통 recognize를 위한 architecture와 차별화된다.

따라서 GAN의 다른 다른 architecture를 위해 NAS 사용

#### 1. Convolution decomposition and layer sensitivity

GAN의 classification or segmentation은 vanilla CNN이다.

최근 사용되는 (depth + point) wise conv는 성능면과 계산량이나 학습이 좋고 이는 GAN에서 매우 좋다.

모든 layer를 decomposition하는 것은 성능이 문제가 있기 때문에 모든 것을 갖기 어렵다.

성능 저하 예방은 param 개수의 증가 없이 layer의 특성 (recognition의 경우와 다른)

Resblock은 upsampling layer를 더하기 영향을 많이 빼지 않기 때문에 Resblock에만 적용된다.

#### 2. Automated channel reduction with NAS

기존의 G는 손으로 만든 균일한 channel의 architecture는 optimal 고려가 어렵다

충분히 알아가 은은한 channel은 automated channel pruning을 사용하여 channel을 선택한다. (reduce computation)

↳ layer의 선택은 conv를 선택할 수 있고, MAC과 hardware 병렬 처리 사이의 균형

$\{C_1, C_2, \dots, C_k\}$ 는  $k$ 개의 선택할 수 있는 prime layer,  $F_t$ 가 computation constraint 일 때, NAS를 사용하여

$$\{C_1^*, C_2^*, \dots, C_k^*\} = \operatorname{argmin}_{C_1, C_2, \dots, C_k} L, \text{ s.t. } \text{MAC} < F_t$$

모든 가능한 channel 조합을 봄에, 그 중 최적화하고, 평가하며, 가장 좋은 G를 고르기 위해 훈련한다.

그러나,  $K$ 가 증가하면, 가능한 channel 구성은 극단적으로 증가하고, 각 구성은 hyper-param 설정에 많은 시간을 필요로 한다.

### - Decouple Training and Search

위와 같은 문제를 해결하기 위해 one-shot NAS와 같이 training과 architecture search를 decoupling 한다.

먼저 once-for-all을 소개한다.

다른 channel의 수를 가진 각 subnetwork는 동일하게 훈련되고, 독립적으로 동작한다.

Subnetwork는 one-for-all 과 weight를 공유한다.

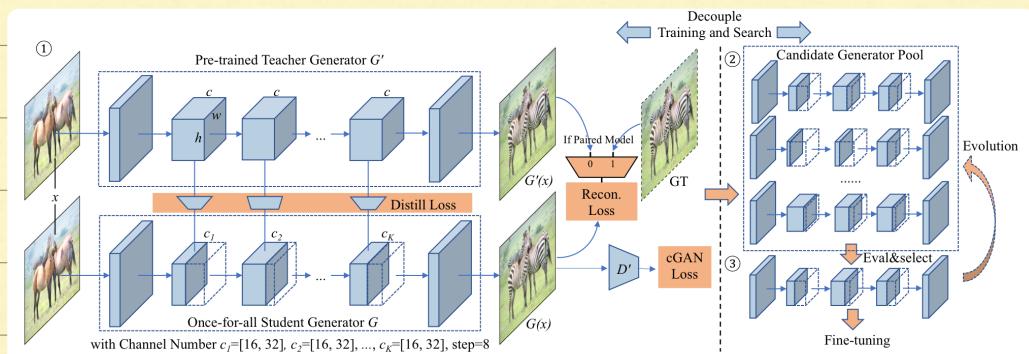


Fig. 3: GAN Compression framework: ① Given a pre-trained teacher generator  $G'$ , we distill a smaller once-for-all student generator  $G$  that contains all possible channel numbers through weight sharing. We choose different channel numbers  $\{c_k\}_{k=1}^K$  for the student generator  $G$  at each training step. ② We then extract many sub-generators from the once-for-all generator and evaluate their performance. No retraining is needed, which is the advantage of the once-for-all generator. ③ Finally, we choose the best sub-generator given the compression ratio target or performance target (FID or mIoU) using either brute-force or evolutionary search method. Optionally, we perform additional fine-tuning to obtain the final model.

Teacher는  $\{C_k^o\}_{k=1}^K$ 로 가정한다.

주어진  $\{C_k\}_{k=1}^K$ ,  $C_k \leq C_o$  및  $[15]$ 의 ODL, once-for-all에서 해당 tensor의 모든 weight를 측정한다.

훈련 단계에서, subnetwork를 random하게 sampling하고 output의 gradient를 계산하고, 그로 update 한다.

처음 여러 channel의 가중치가 더 가중 update 되기 때문에 모든 weight 중에서 중요한 역할을 한다.

Once-for-all이 훈련된 다음엔, val에 대한 혹은 subnetwork 평균으로 바로 가장 좋은 subnetwork를 찾는다.

Once-for-all은 weight sharing으로 훈련되어 때문에 fine-tuning이 필요 없다.

성능 향상을 위한 architecture fine-tuning은 16.3에서 설명.

## \* Experiments

- Models, datasets, evaluation metrics

### 1. Models.

일반성을 입증하기 위해, CycleGAN, pix2pix, GauGAN으로 실험한다.

### 2. Dataset

Edges → shoes (pix2pix)

Cityspaces (pix2pix, GauGAN)

Horse ↔ Zebra (CycleGAN)

Map ↔ aerial photo (pix2pix)

### 3. Evaluation metrics

FID: inceptionV3를 사용하여, real과 generated feature vectors의 거리를 측정. 낮을수록 좋다.

## Semantic Segmentation Metrics

Cityscape dataset을 활용해 semantic segmentation Metrics를 사용한다.

semantic segmentation model을 gen image로 카운트 실행하고 성능을 비교한다.

L<sub>1</sub>mAPE와 DFN-D-105 model을 사용한다.

## - Result

### 1. Quantitative Result

Model	Dataset	Method	#Parameters		MACs	Metric	
						FID ( $\downarrow$ )	mAP ( $\uparrow$ )
CycleGAN	horse→zebra	Original	11.3M	—	56.8G	—	—
		Shu <i>et al.</i> [52]	—	—	13.4G (1.2 $\times$ )	96.15 (34.6 $\ominus$ )	—
		Ours (w/o fine-tuning)	0.34M (33.3 $\times$ )	2.67G (21.2 $\times$ )	64.95 (3.42 $\ominus$ )	—	—
		Ours	0.34M (33.3 $\times$ )	2.67G (21.2 $\times$ )	71.81 (10.3 $\ominus$ )	—	—
Pix2pix	edges→shoes	Original	11.3M	—	56.8G	—	—
		Ours (w/o fine-tuning)	0.70M (16.3 $\times$ )	4.81G (11.8 $\times$ )	31.30 (7.12 $\ominus$ )	—	—
	cityscapes	Ours	0.70M (16.3 $\times$ )	4.81G (11.8 $\times$ )	26.60 (2.42 $\ominus$ )	—	—
		Original	11.3M	—	56.8G	—	35.62 (6.35 $\ominus$ )
GauGAN	cityscapes	Ours (w/o fine-tuning)	0.71M (16.0 $\times$ )	5.66G (10 $\times$ )	—	29.27 (6.35 $\ominus$ )	—
		Ours	0.71M (16.0 $\times$ )	5.66G (10.0 $\times$ )	—	34.34 (1.28 $\ominus$ )	—
	map→aerial photo	Original	11.3M	—	56.8G	—	—
		Ours (w/o fine-tuning)	0.75M (15.1 $\times$ )	4.68G (11.4 $\times$ )	71.82 (24.1 $\ominus$ )	—	—
		Ours	0.75M (15.1 $\times$ )	4.68G (11.4 $\times$ )	48.02 (0.26 $\ominus$ )	—	—
		Original	93.0M	—	281G	—	58.89 (2.14 $\ominus$ )
		Ours (w/o fine-tuning)	20.4M (4.6 $\times$ )	31.7G (8.8 $\times$ )	—	56.75 (0.48 $\ominus$ )	—
		Ours	20.4M (4.6 $\times$ )	31.7G (8.8 $\times$ )	—	58.41 (0.48 $\ominus$ )	—

Table 1: Quantitative evaluation of GAN Compression: We use the mAP metric (the higher the better) for the Cityscapes dataset and FID (the lower the better) for other datasets. Our method can compress state-of-the-art conditional GANs by **9-21 $\times$**  in MACs and **5-33 $\times$**  in model size, with only minor performance degradation. For CycleGAN compression, our systematic approach outperforms previous CycleGAN-specific Co-Evolution method [52] by a large margin.

특히, co-Evolution 블록 낫다.

### 2. Performance vs. Computation Trade-off

압축 비율 이외에도, 모델 사이즈와 상관없이 성능을 지속적으로 높인다.

pix2pix의 trade-off에는 차이가 같다.

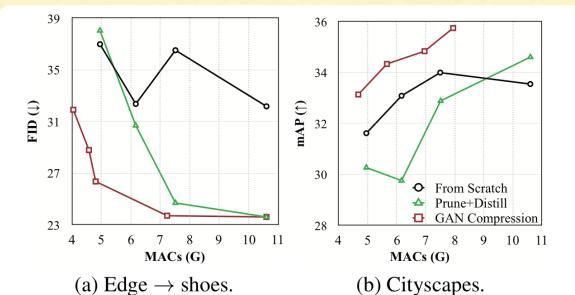


Figure 6: Uniform channel pruning + distillation (without NAS) outperforms training from scratch for larger models, but works poorly when the model is aggressively shrunk. GAN Compression consistently improves the performance vs. computation trade-off at various scales.

large model의 distill+prune은 충분히 빠르지만, NAS가 없으면, channel이 초기화 때,

sensitive한 layer가 너무 많이 초기화된다.

⇒ NAS 필요.

## 2. Qualitative results

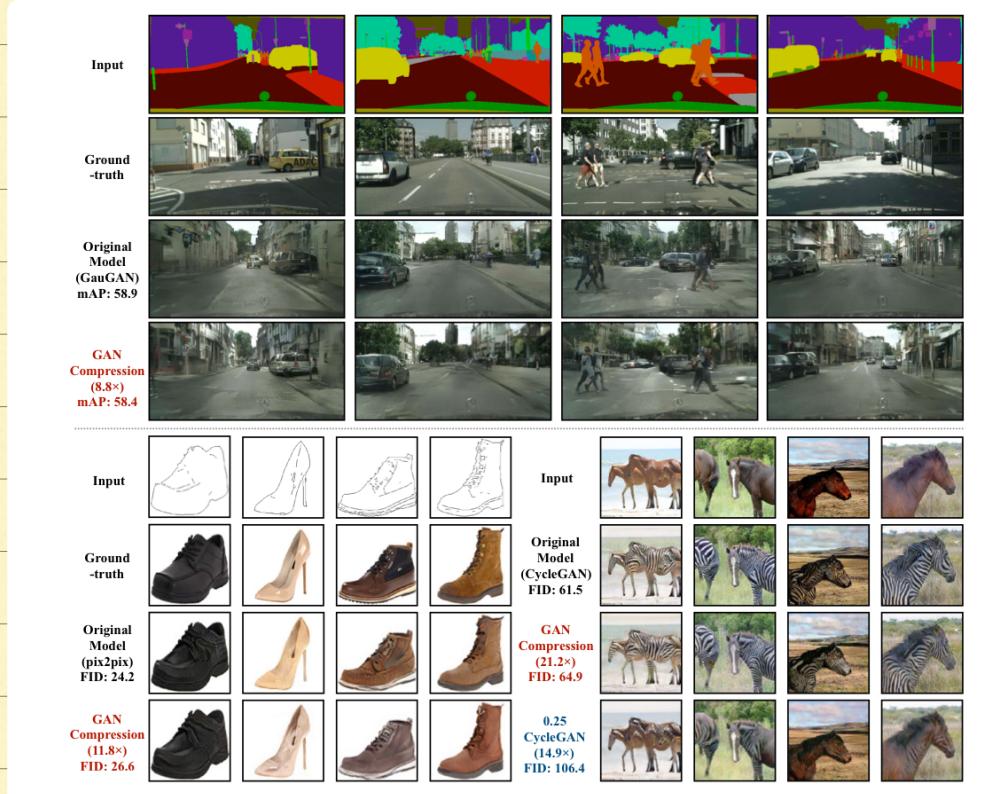


Figure 4: Qualitative compression results on Cityscapes, Edges→Shoes and Horse→Zebra. GAN Compression preserves the fidelity while significantly reducing the computation. In contrast, directly training a smaller model (e.g., 0.25 CycleGAN, which linearly scales each layer to 25% channels) yields poor performance.

## 3. Accelerate Inference on Hardware

real-world interactive application의 hardware inference acceleration은 계산량 줄이기로 즐是最好的.

여러 장치에서 compression model의 inference 속도 향상.

Model	CycleGAN	Pix2pix	GauGAN	
Metric	FID ( $\downarrow$ )	61.5 → 65.0	24.2 → 26.6	—
	mAP ( $\uparrow$ )	—	—	58.9 → 58.4
MAC Reduction	21.2x	11.8x	8.8x	
Memory Reduction	2.0x	1.7x	1.8x	
Xavier	CPU	1.65s (18.5x)	3.07s (9.9x)	21.2s (7.9x)
Speedup	GPU	0.026s (3.1x)	0.035s (2.4x)	0.10s (3.2x)
Nano	CPU	6.30s (14.0x)	8.57s (10.3x)	65.3s (8.6x)
Speedup	GPU	0.16s (4.0x)	0.26s (2.5x)	0.81s (3.3x)
1080Ti Speedup		0.005s (2.5x)	0.007s (1.8x)	0.034s (1.7x)
Xeon Silver 4114	CPU Speedup	0.11s (3.4x)	0.15s (2.6x)	0.74s (2.8x)

⇒ Jetson에서 Cityscape → 40FPS.

Table 2: Measured memory reduction and latency speedup on NVIDIA Jetson AGX Xavier, NVIDIA Jetson Nano, 1080 Ti GPU and Xeon CPU. CycleGAN, pix2pix, and GauGAN are trained on horse→zebra, edges→shoes and Cityscapes.

## - Ablation study

### 1. Advantage of unpaired-to-paired transform

Teacher로부터 pseudo pairs를 이용해 조사했다.

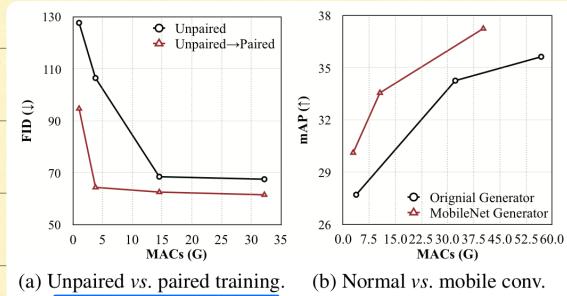


Figure 7: (a) Transforming unpaired training into paired training (using the pseudo pairs generated by the teacher model, without NAS) significantly improves the performance of efficient models. (b) Decomposing the convolutions in the original ResNet-based generator into a channel-wise and depth-wise convolutions (MobileNet generator) improves the performance vs. computation trade-off.

⇒ computation budget이 줄어들면서 unpaired는 성능 저하가 심하게 일어난다.

⇒ pseudo pair의 경우 유익함

⇒ unpair를 학습하는 model이 강력한학습 (source와 target의 아래쪽 판별도 가능할 만큼)

⇒ 거시적인 방법에서는 teacher에서 직접 학습하는 수 있고, discriminator로 실제 target에 대한 정보를 지속적으로 학습할 수 있다 (?)

### 2. Effectiveness of convolution decomposition

Convolution decomposition transform에 관한 CycleGAN의 인강도를 볼 예정이다.

ResNet-base의 CycleGAN으로 test를 진행한다.

↳ 3부분으로 나눔 : Downsample, Resblock, Upsample

↳ 모든 conv ⇒ separable conv (mobileNet V1)

Model	ngf	FID	MACs	#Parameters
Original	64	61.75	56.8G	11.38M
Only change downsample	64	68.72	55.5G	11.13M
Only change upsample	64	61.04	48.3G	11.05M
<b>Only change resBlocks</b>	<b>64</b>	<b>62.95</b>	<b>18.3G</b>	<b>1.98M</b>
Only change downsample	16	74.77	3.6G	0.70M
Only change upsample	16	95.54	3.3G	0.70M
<b>Only change resBlocks</b>	<b>16</b>	<b>79.49</b>	<b>1.4G</b>	<b>0.14M</b>

Table 3: We report the performance after applying convolution decomposition in each of the three parts (Downsample, ResBlocks, and Upsample) of the ResNet generator respectively on the horse→zebra dataset. ngf denotes the number of the generator's filters. Both computation and model size are proportional to  $ngf^2$ . We evaluate two settings  $ngf=64$  and  $ngf=16$ . We observe that modifying ResBlock blocks shows a significantly better performance vs. computation trade-off, compared to modifying other parts of the network.

⇒ Resblock만 수정할 수 있음

⇒ Channel of 16, 64 일 때 trade-off ↑

⇒ 또한 fig 1b 및 decomposition을 하면 trade-off ↑ 가능

## \* Conclusion

General한 model size, computation cost를 줄이는 방법을 제시했다.

paired upair를 합쳤다.

model의 unstable한 특성을 줄이기 위해 MAS와 distillation을 사용했다.

좋은 성능을 보였다.