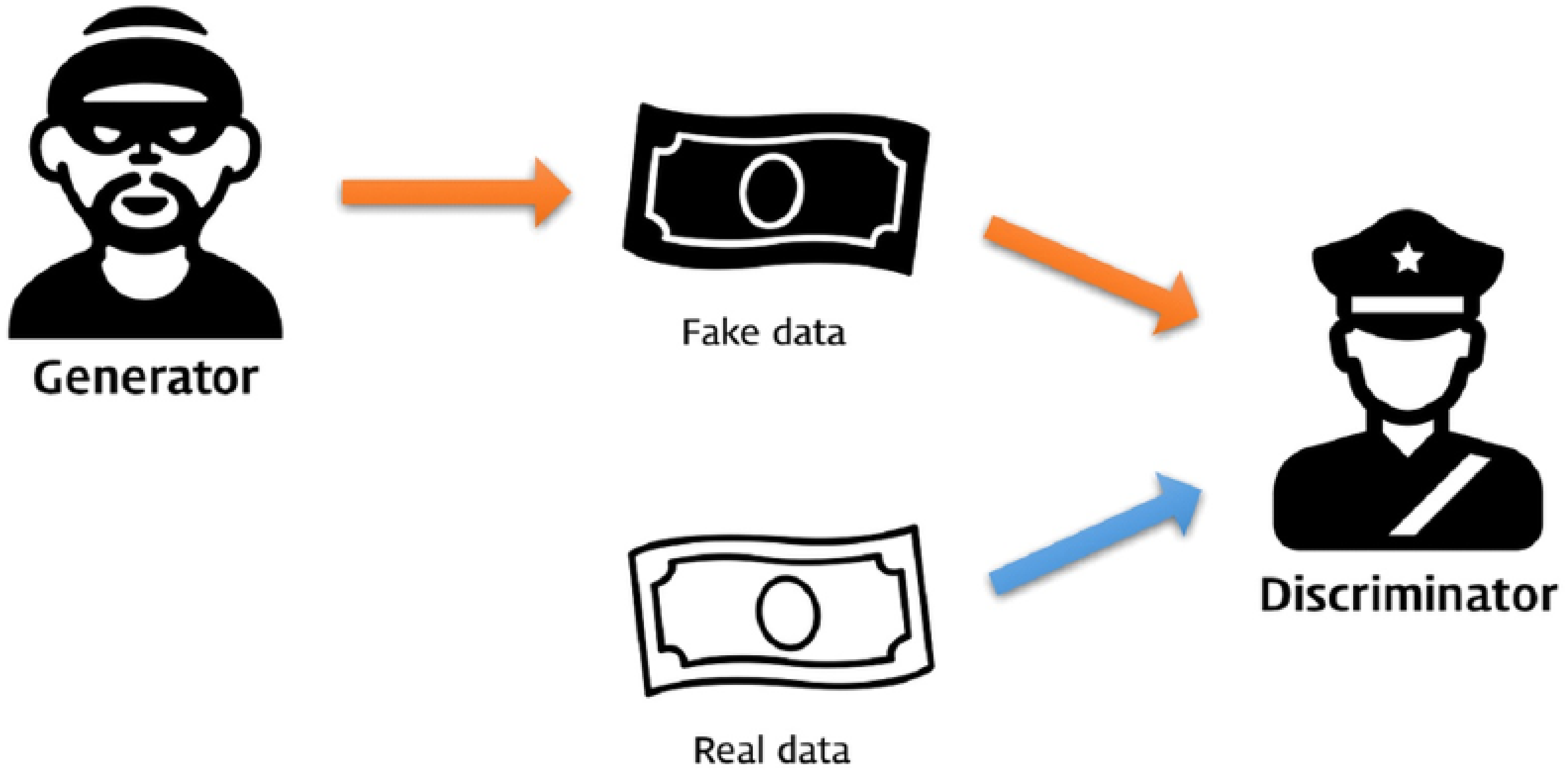
A U-Net Based Discriminator for Generative Adversarial Networks

CVPR 2020

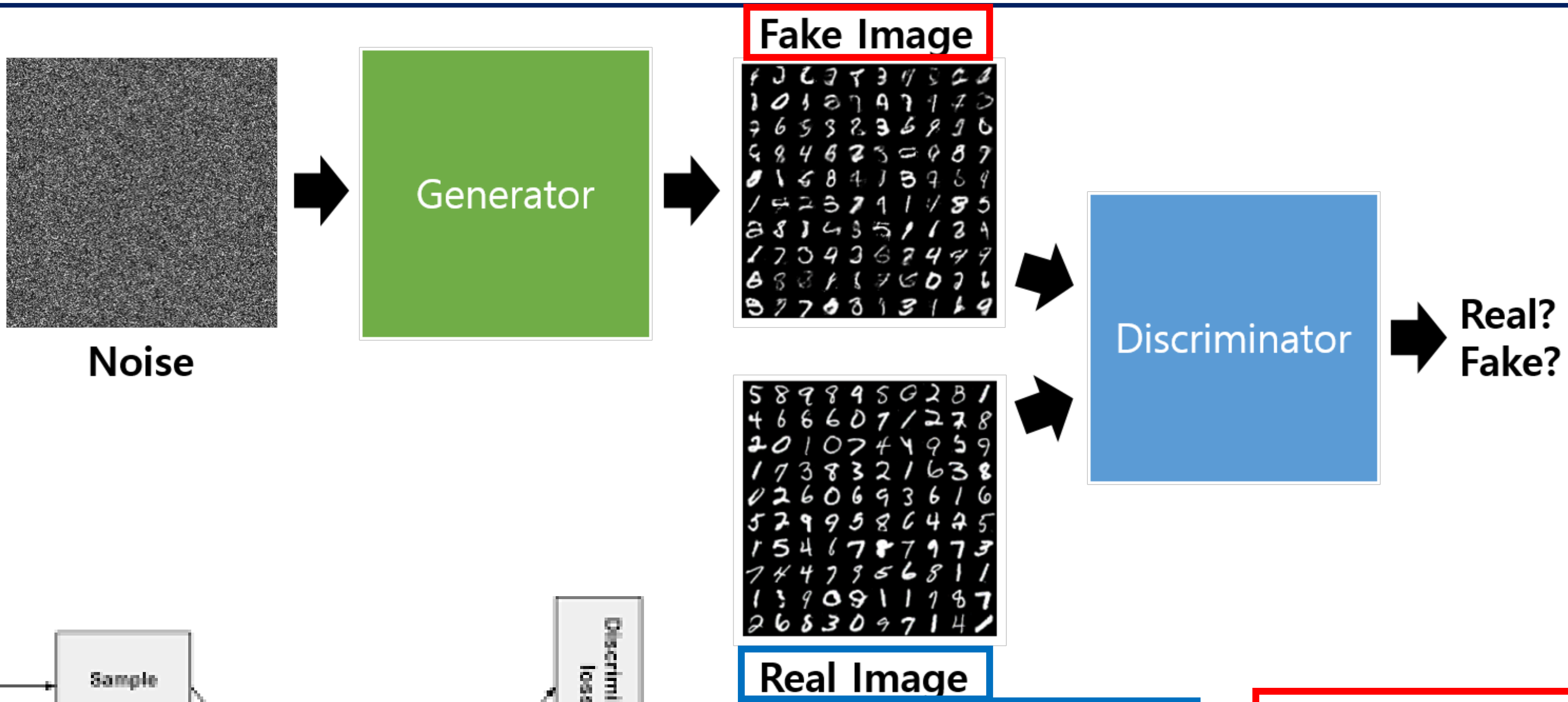
Haneol Lee

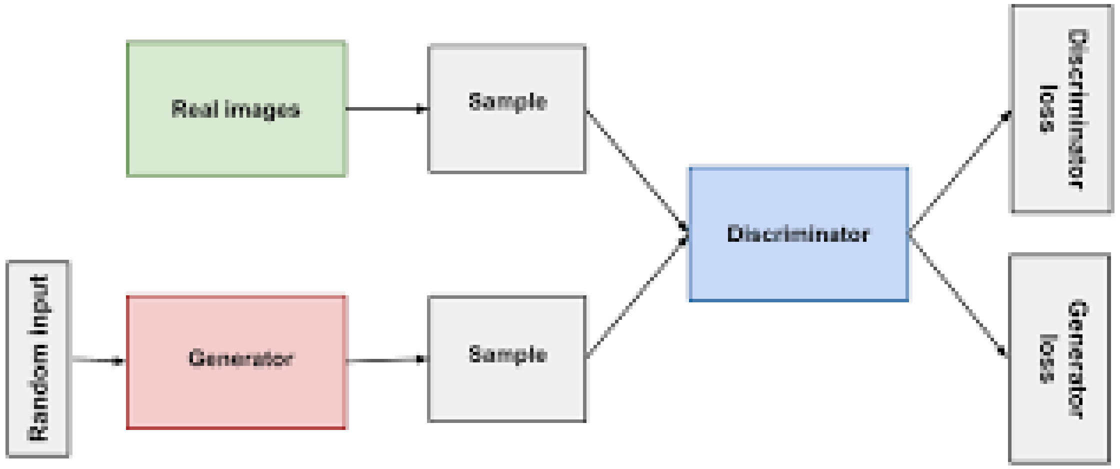
2th May, 2022

Introduction: GAN (Generative Adversarial Networks)



Introduction: Generator Loss VS Discriminator Loss





$$\mathcal{L}_D = \boxed{-\mathbb{E}_x[\log D(x)]} - \boxed{\mathbb{E}_z[\log(1 - D(G(z)))]},$$

$$\mathcal{L}_G = \boxed{-\mathbb{E}_z[\log D(G(z))]^1}$$

Introduction: BigGAN

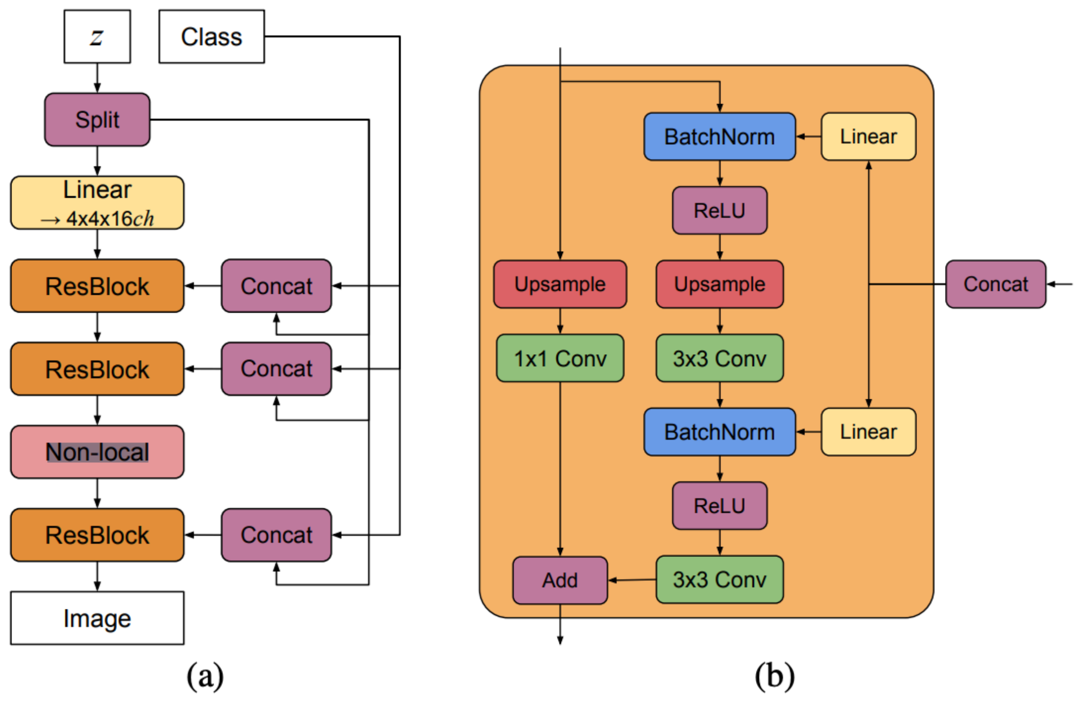
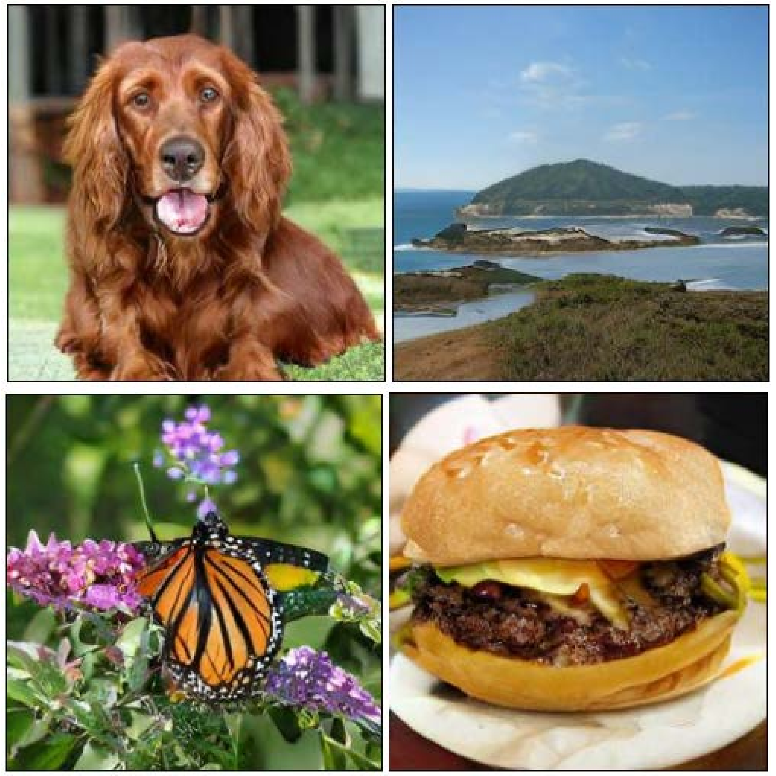


Figure 15: (a) A typical architectural layout for BigGAN's **G**; details are in the following tables. (b) A Residual Block (*ResBlock up*) in BigGAN's **G**. (c) A Residual Block (*ResBlock down*) in BigGAN's **D**.

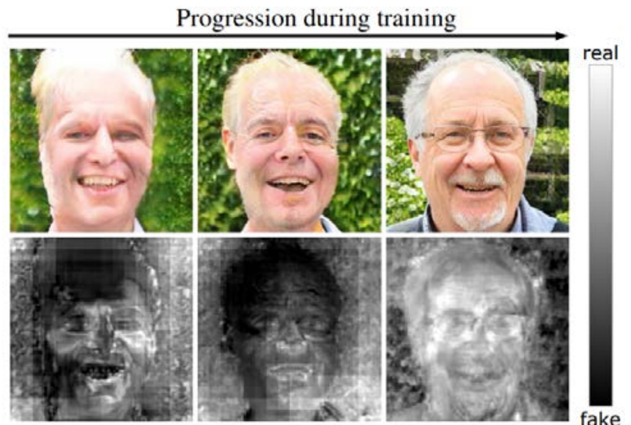


Class-conditional samples generated by our model.

Link: https://arxiv.org/pdf/1809.11096.pdf

Abstract

- Borrowing U-Net segmentation network for the discriminator to target the issue of the capacity to synthesize globally and locally coherent images with object shapes and textures indistinguishable from real images.
- U-Net based discriminator architecture allows to provide detailed per-pixel feedback to the generator while maintaining the global coherence of synthesized images.
- Propose a per-pixel consistency regularization technique based on the CutMix data augmentation encouraging discriminator to focus more on semantic and structural changes between real and fake images.
- Finally, the U-Net based discriminator enables the generator to enhance the quality of generated samples, and to synthesize images with varying structure, appearance and levels of detail.
- Compared to the BigGAN baseline, we achieve an average improvement of 2.7 FID across FFHQ, CelebA, and the newly introduced COCO-Animals dataset.



Introduction

Challenging Limitation issues:

1. **Global semantic coherence**

   ➔ Detailed and spatially coherent response to the generator

2. **Long-range structure**

   ➔ U-Net based architecture

3. **Exactness of detail**

   ➔Better qualitative performance
   of details

Introduction

Challenging Limitation issues:

1. **Global semantic coherence**

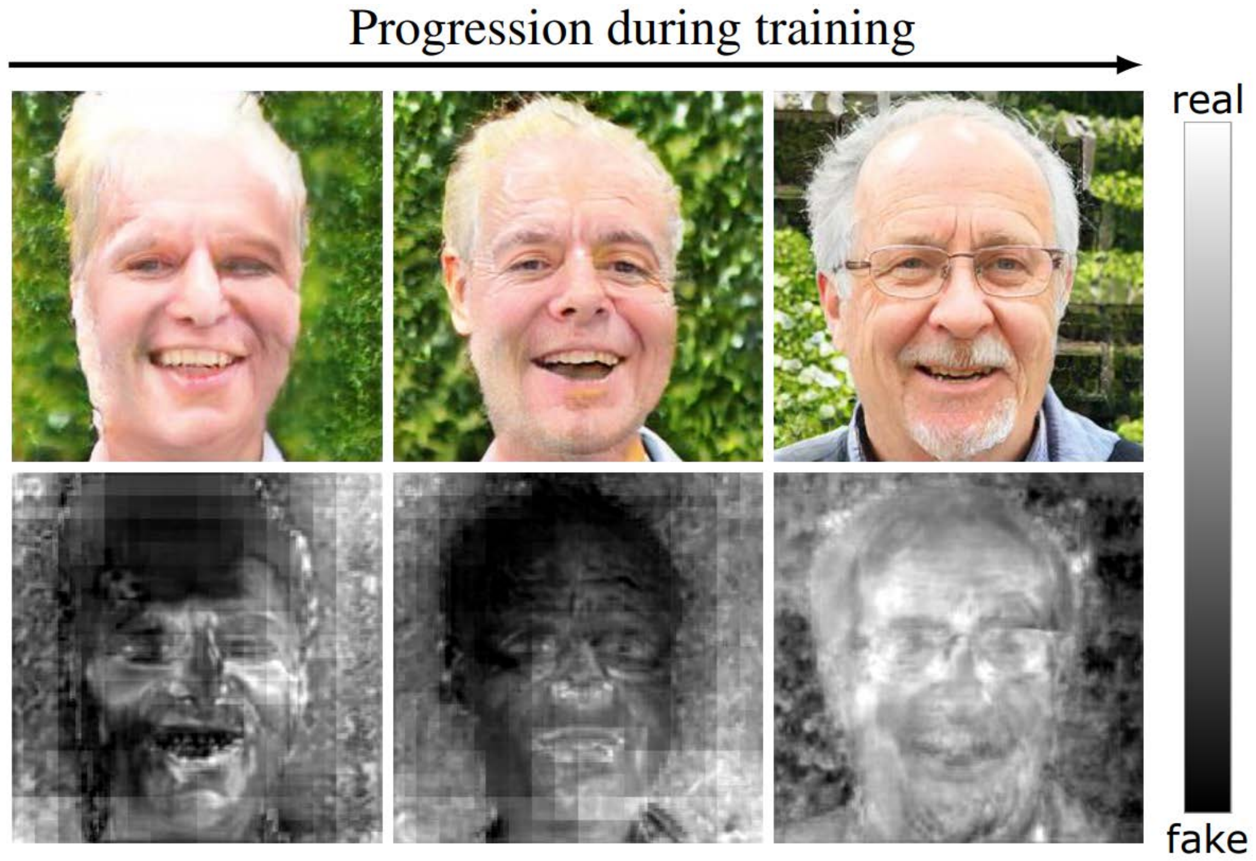
   ➔ Detailed and spatially coherent response to the generator

2. **Long-range structure**

   ➔ U-Net based architecture

3. **Exactness of detail**

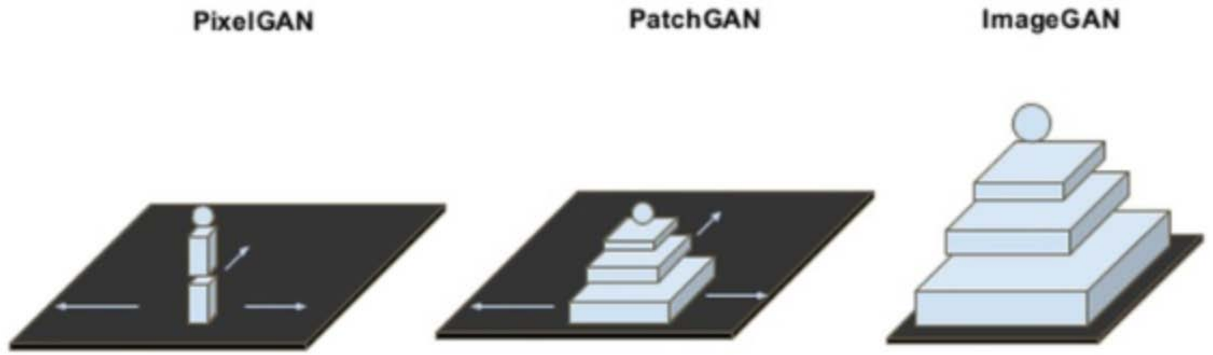
   ➔Better qualitative performance of details

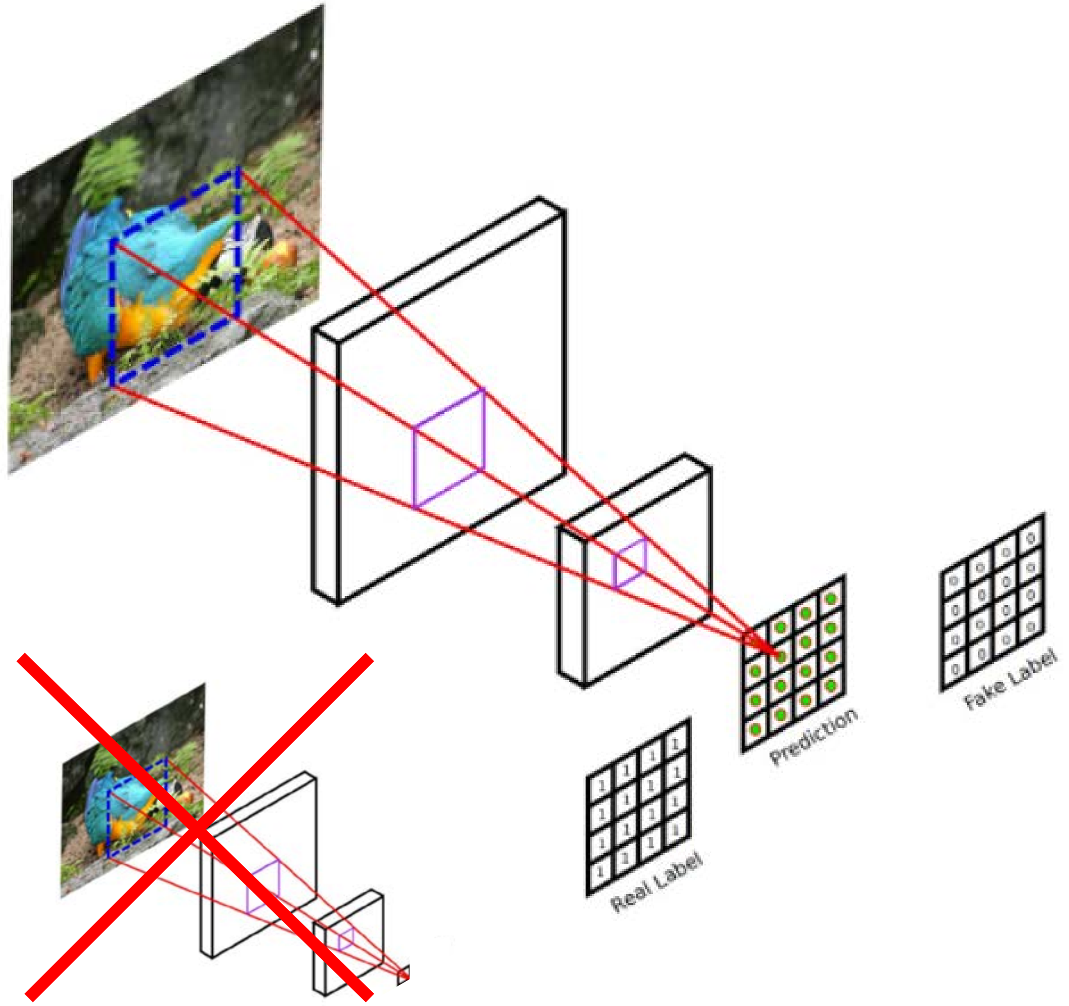


Introduction: PatchGAN

1. **Lighter parameters**

2. **Proper Receptive Field Size**

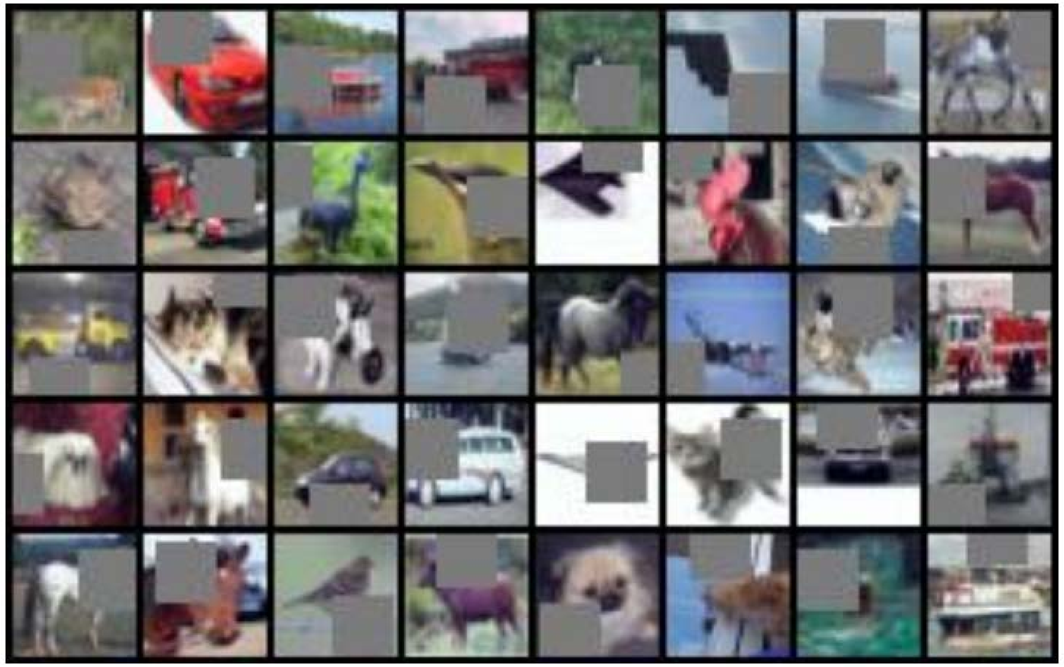




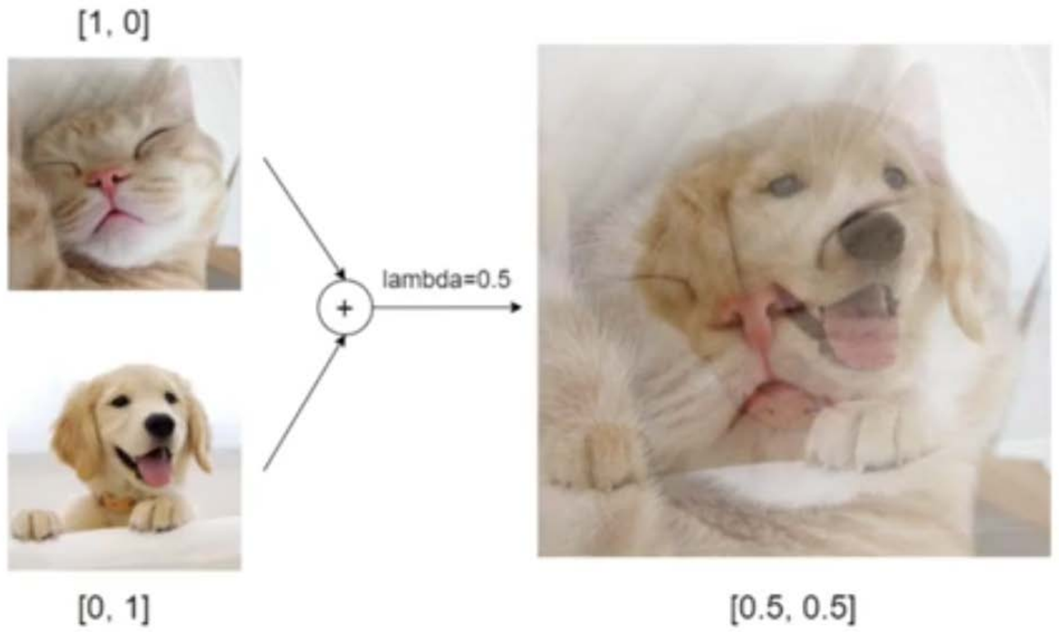
Link: https://arxiv.org/abs/1611.07004v3

Introduction: CutMix

CutMix = **CutOut** + **MixUp**



+

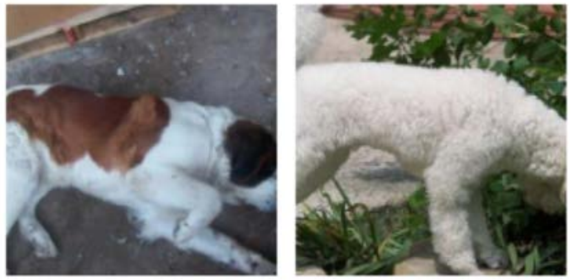


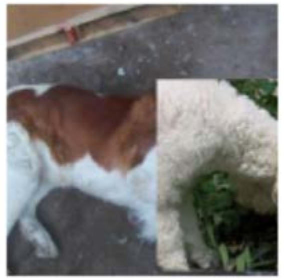
Mix-up works by blending 2 images with alpha % from image_1 and (1-alpha) % from image_2

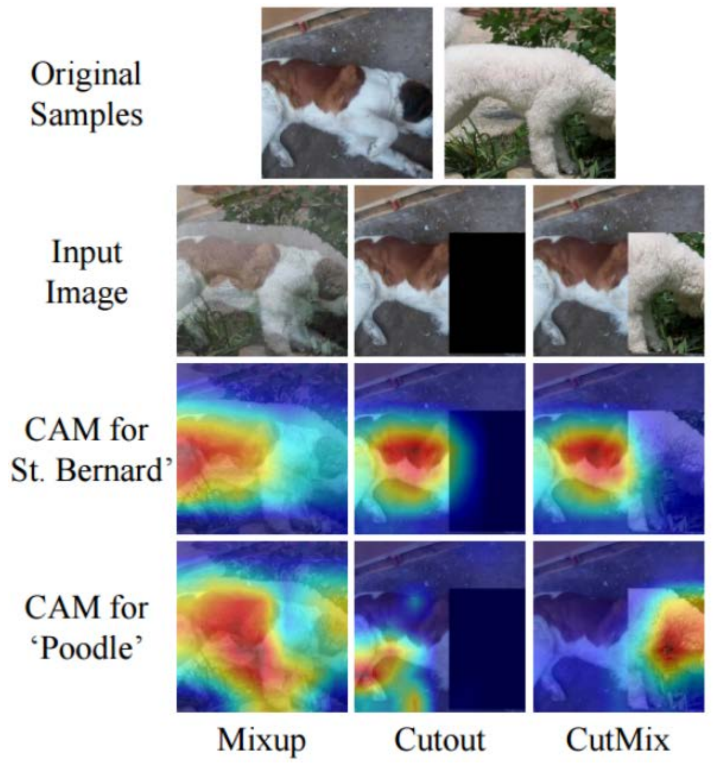
Link1(MixUp): https://arxiv.org/pdf/1710.09412.pdf
Link2(CutOut): https://arxiv.org/pdf/1708.04552.pdf

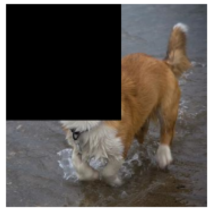
Introduction: CutMix

1. Regularization method to **increase generalization** and **localization performance** by **allowing models** to view and **learn less distinctive parts** and the **overall region** of the image, rather than focusing on where they can identify differences in objects


original image


cutmix



	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.6 (+2.3)**
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3 (+1.0)**
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	**76.7 (+1.1)**

Link: https://arxiv.org/pdf/1905.04899.pdf

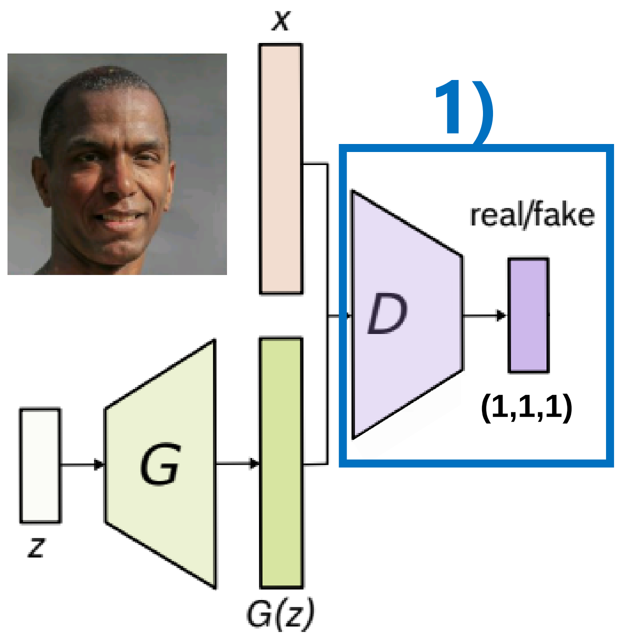
Methods

Redesign the role of the discriminator as 1) classifier, 2) segmenter

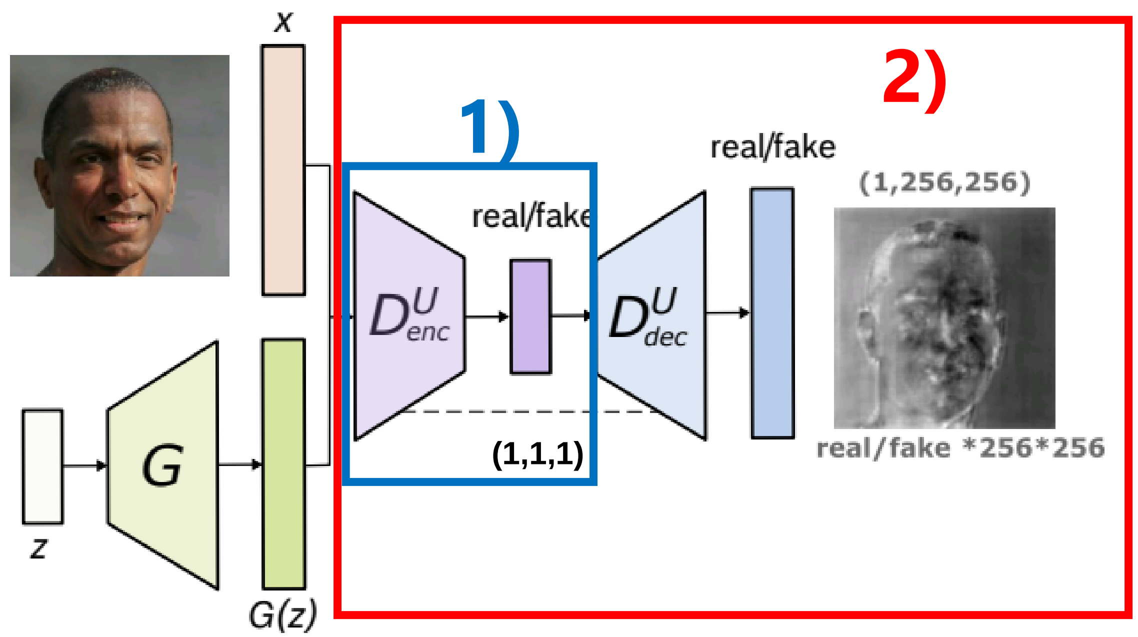
(1) Baseline GAN (Vanilla or BigGAN) VS. (2) U-Net GAN

- **Architecture: Same Generator, Different Discriminator**

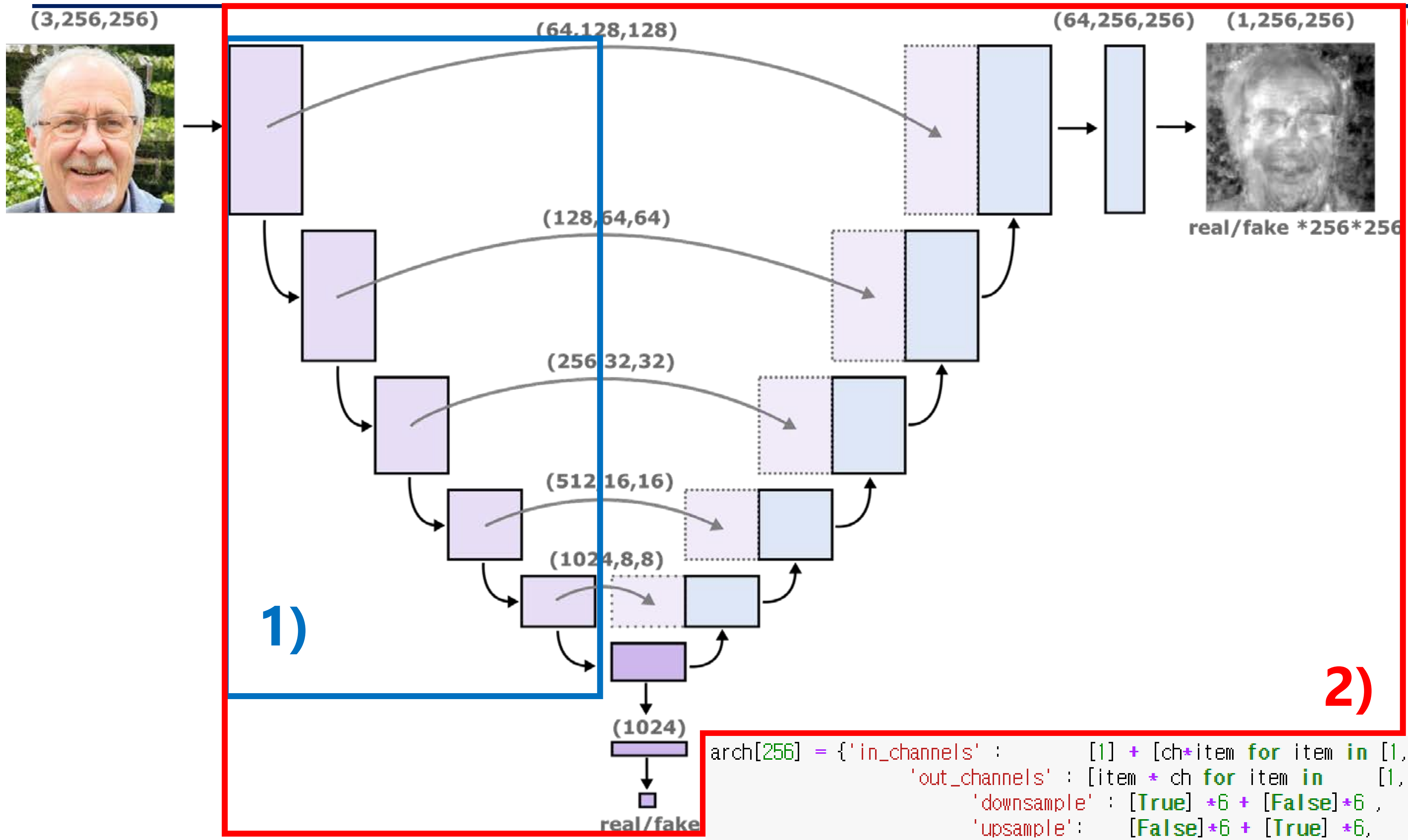
(1)

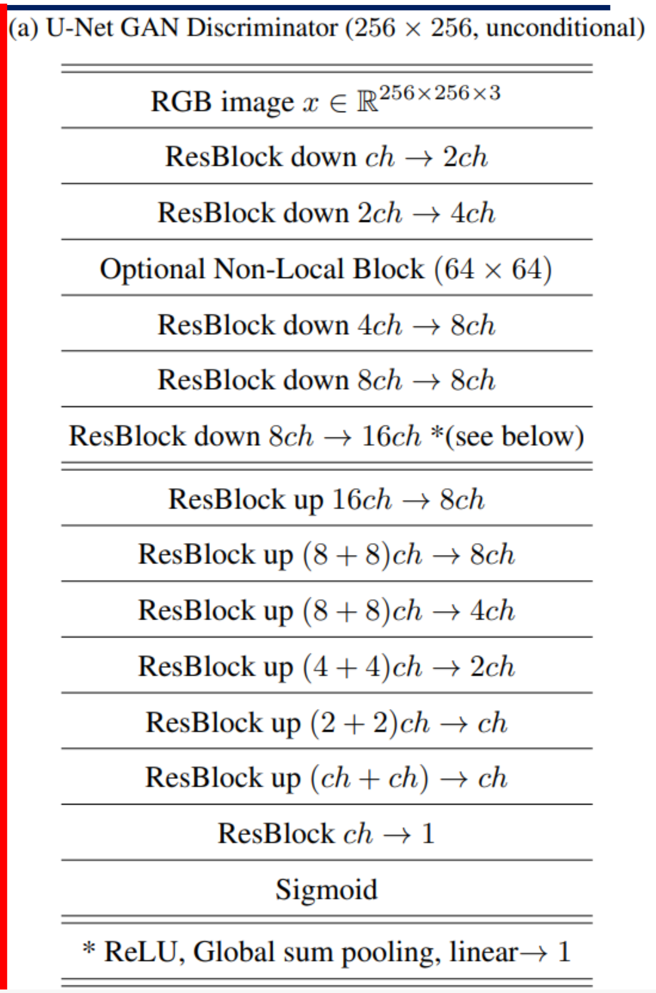


(2)



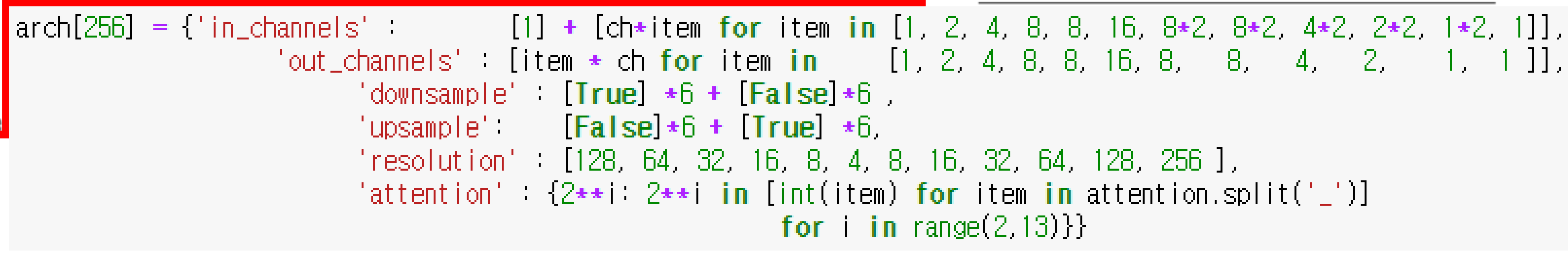
Methods: U-Net Based Discriminator Architecture





(a) U-Net GAN Discriminator (256 × 256, unconditional)

RGB image $x \in \mathbb{R}^{256\times256\times3}$
ResBlock down $ch \rightarrow 2ch$
ResBlock down $2ch \rightarrow 4ch$
Optional Non-Local Block (64×64)
ResBlock down $4ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$ *(see below)
ResBlock up $16ch \rightarrow 8ch$
ResBlock up $(8+8)ch \rightarrow 8ch$
ResBlock up $(8+8)ch \rightarrow 4ch$
ResBlock up $(4+4)ch \rightarrow 2ch$
ResBlock up $(2+2)ch \rightarrow ch$
ResBlock up $(ch+ch) \rightarrow ch$
ResBlock $ch \rightarrow 1$
Sigmoid
* ReLU, Global sum pooling, linear→ 1



```
arch[256] = {'in_channels' :         [1] + [ch*item for item in [1, 2, 4, 8, 8, 16, 8*2, 8*2, 4*2, 2*2, 1*2, 1]],
              'out_channels' : [item * ch for item in     [1, 2, 4, 8, 8, 16, 8,   8,   4,   2,   1,   1 ]],
               'downsample' : [True] *6 + [False]*6 ,
               'upsample':    [False]*6 + [True] *6,
               'resolution' : [128, 64, 32, 16, 8, 4, 8, 16, 32, 64, 128, 256 ],
               'attention' : {2**i: 2**i in [int(item) for item in attention.split('_')]
                               for i in range(2,13)}}
```

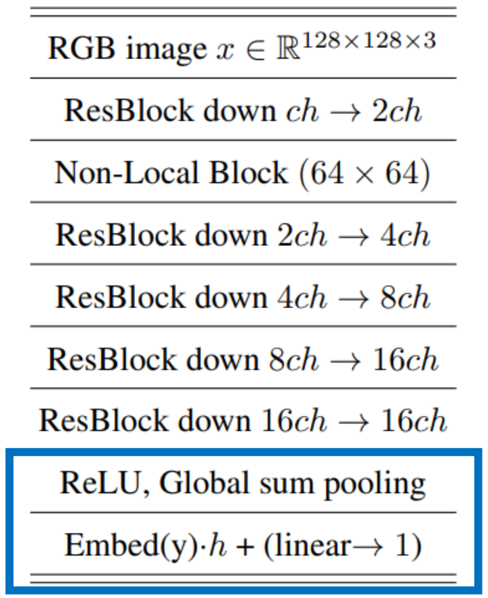
Methods

(1) Baseline GAN (Vanilla or BigGAN) VS. (2) U-Net GAN

- **Architecture: Different Discriminator (Baseline D ➔ U-Net based D)**

(1)

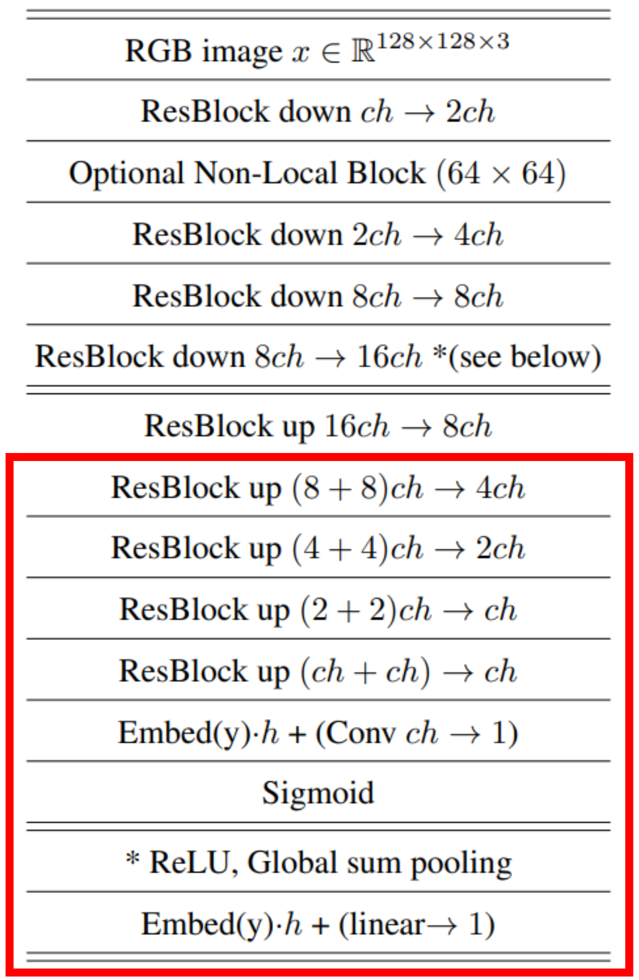
(b) BigGAN Discriminator (128 × 128, class-conditional)



RGB image $x \in \mathbb{R}^{128\times128\times3}$
ResBlock down $ch \rightarrow 2ch$
Non-Local Block (64×64)
ResBlock down $2ch \rightarrow 4ch$
ResBlock down $4ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$
ResBlock down $16ch \rightarrow 16ch$
ReLU, Global sum pooling
Embed(y)·h + (linear$\rightarrow$ 1)

(2)

(b) U-Net GAN Discriminator(128 × 128, class-conditional)

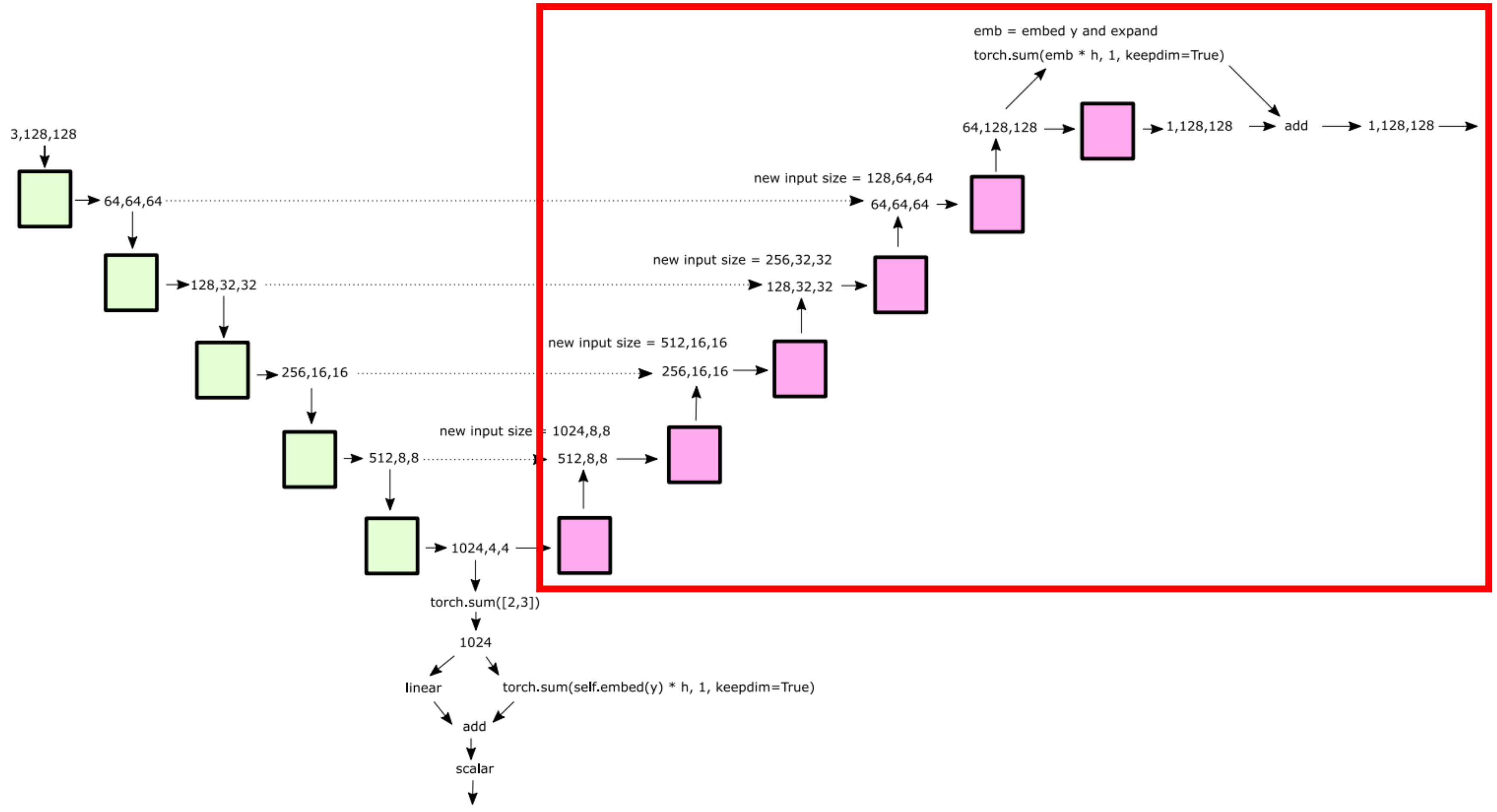


RGB image $x \in \mathbb{R}^{128\times128\times3}$
ResBlock down $ch \rightarrow 2ch$
Optional Non-Local Block (64×64)
ResBlock down $2ch \rightarrow 4ch$
ResBlock down $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$ *(see below)
ResBlock up $16ch \rightarrow 8ch$
ResBlock up $(8+8)ch \rightarrow 4ch$
ResBlock up $(4+4)ch \rightarrow 2ch$
ResBlock up $(2+2)ch \rightarrow ch$
ResBlock up $(ch+ch) \rightarrow ch$
Embed(y)·h + (Conv $ch \rightarrow 1$)
Sigmoid
* ReLU, Global sum pooling
Embed(y)·h + (linear$\rightarrow$ 1)

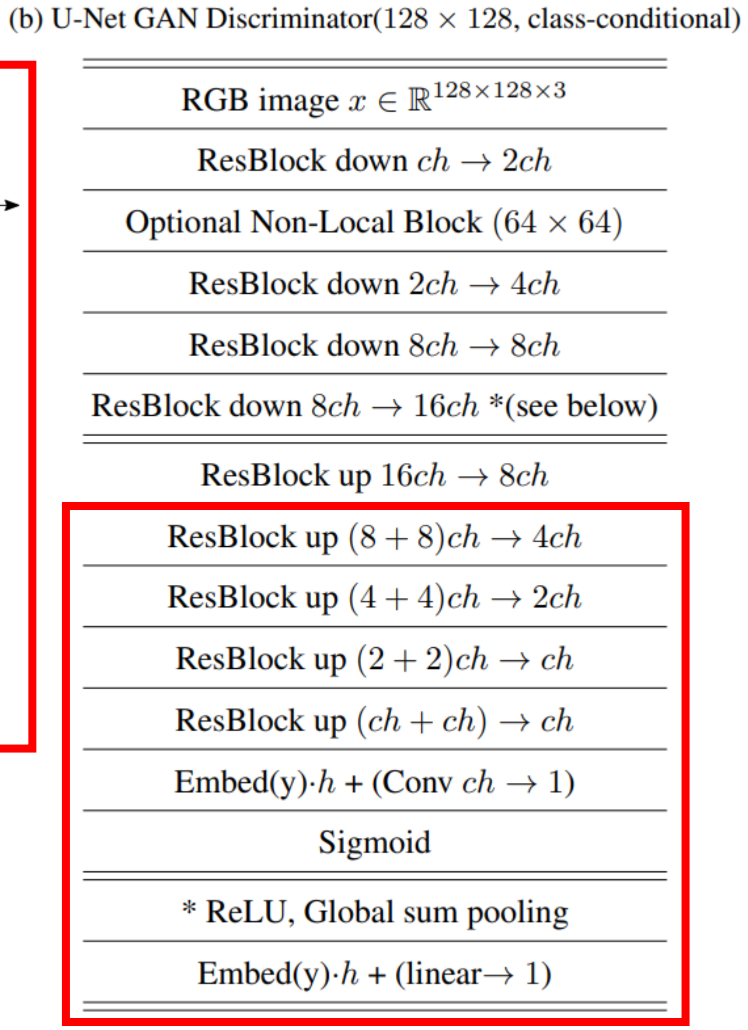
Methods

(1) Baseline GAN (Vanilla or BigGAN) VS. (2) U-Net GAN

- **Architecture: Different Discriminator (Baseline D ➔ U-Net based D)**



3,128,128
64,64,64
128,32,32
256,16,16
512,8,8
1024,4,4
torch.sum([2,3])
1024
linear
torch.sum(self.embed(y) * h, 1, keepdim=True)
add
scalar
new input size = 1024,8,8
512,8,8
new input size = 512,16,16
256,16,16
new input size = 256,32,32
128,32,32
new input size = 128,64,64
64,64,64
64,128,128
emb = embed y and expand
torch.sum(emb * h, 1, keepdim=True)
1,128,128
add
1,128,128


(b) U-Net GAN Discriminator(128 × 128, class-conditional)

RGB image $x \in \mathbb{R}^{128\times128\times3}$
ResBlock down $ch \to 2ch$
Optional Non-Local Block (64×64)
ResBlock down $2ch \to 4ch$
ResBlock down $8ch \to 8ch$
ResBlock down $8ch \to 16ch$ *(see below)
ResBlock up $16ch \to 8ch$
ResBlock up $(8+8)ch \to 4ch$
ResBlock up $(4+4)ch \to 2ch$
ResBlock up $(2+2)ch \to ch$
ResBlock up $(ch+ch) \to ch$
Embed(y)$\cdot h$ + (Conv $ch \to 1$)
Sigmoid
* ReLU, Global sum pooling
Embed(y)$\cdot h$ + (linear$\to$ 1)

Methods

(1) Baseline GAN (Vanilla or BigGAN) VS. (2) U-Net GAN

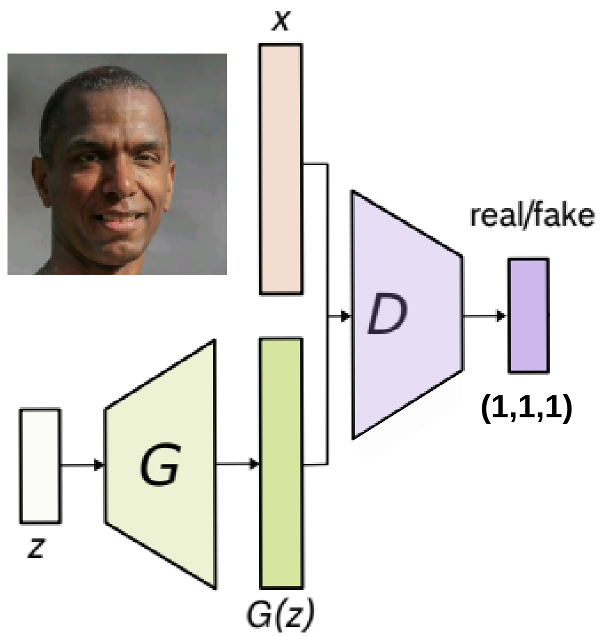
- **Architecture: Same Generator, Different Discriminator**

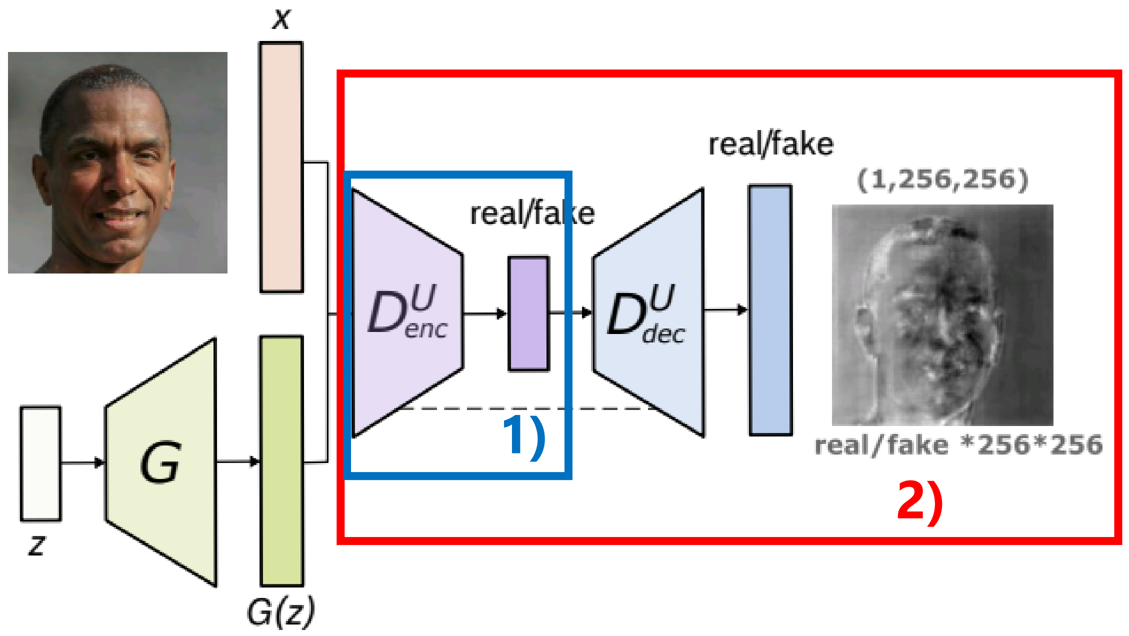
(1) $$\mathcal{L}_G = -\mathbb{E}_z[\log D(G(z))]^1$$

(2) $$\mathcal{L}_G = -\mathbb{E}_z\Big[\boxed{\log D^U_{enc}(G(z))} + \boxed{\sum_{i,j} \log[D^U_{dec}(G(z))]_{i,j}}\Big]$$

$$\mathcal{L}_D = -\mathbb{E}_x[\log D(x)] - \mathbb{E}_z[\log(1 - D(G(z)))],$$

$$\mathcal{L}_{D^U} = \mathcal{L}_{D^U_{enc}} + \mathcal{L}_{D^U_{dec}},$$





Methods

(1) Baseline GAN (Vanilla or BigGAN) VS. (2) U-Net GAN

- Architecture: Different Discriminator (Baseline D ➔ U-Net based D)

(1)

$$\mathcal{L}_G = -\mathbb{E}_z[\log D(G(z))]^1$$

$$\mathcal{L}_D = -\mathbb{E}_x[\log D(x)] - \mathbb{E}_z[\log(1 - D(G(z)))],$$

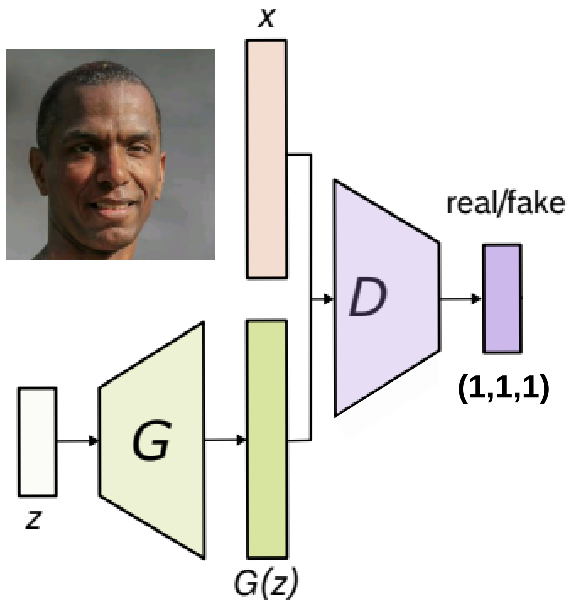
(2)

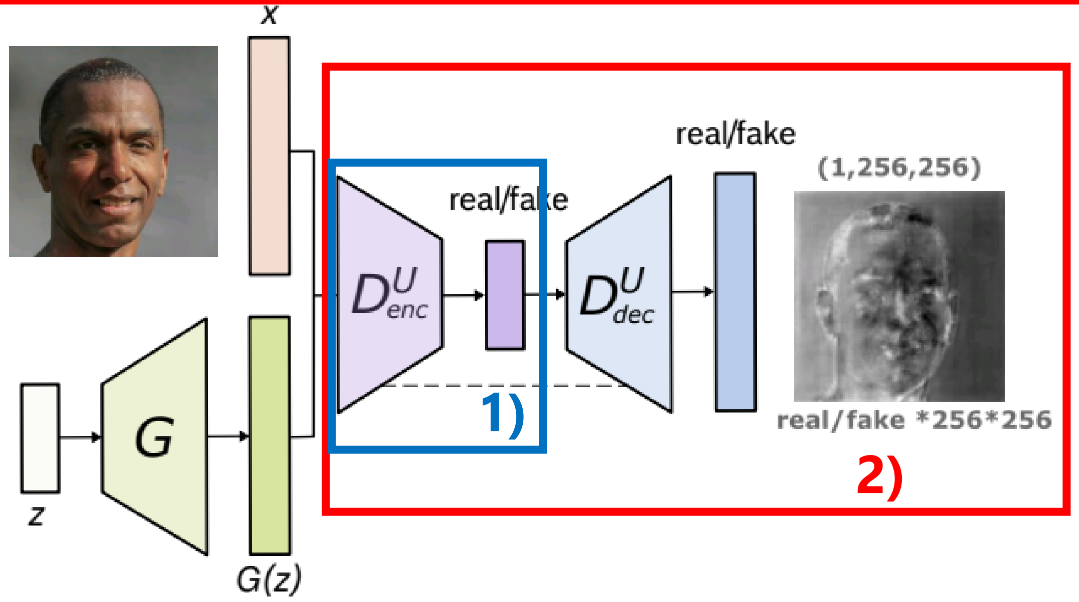
$$\mathcal{L}_G = -\mathbb{E}_z\Big[\log D^U_{enc}(G(z)) + \sum_{i,j} \log[D^U_{dec}(G(z))]_{i,j}\Big]$$

$$\mathcal{L}_{D^U} = \boxed{\mathcal{L}_{D^U_{enc}}} + \boxed{\mathcal{L}_{D^U_{dec}},}$$

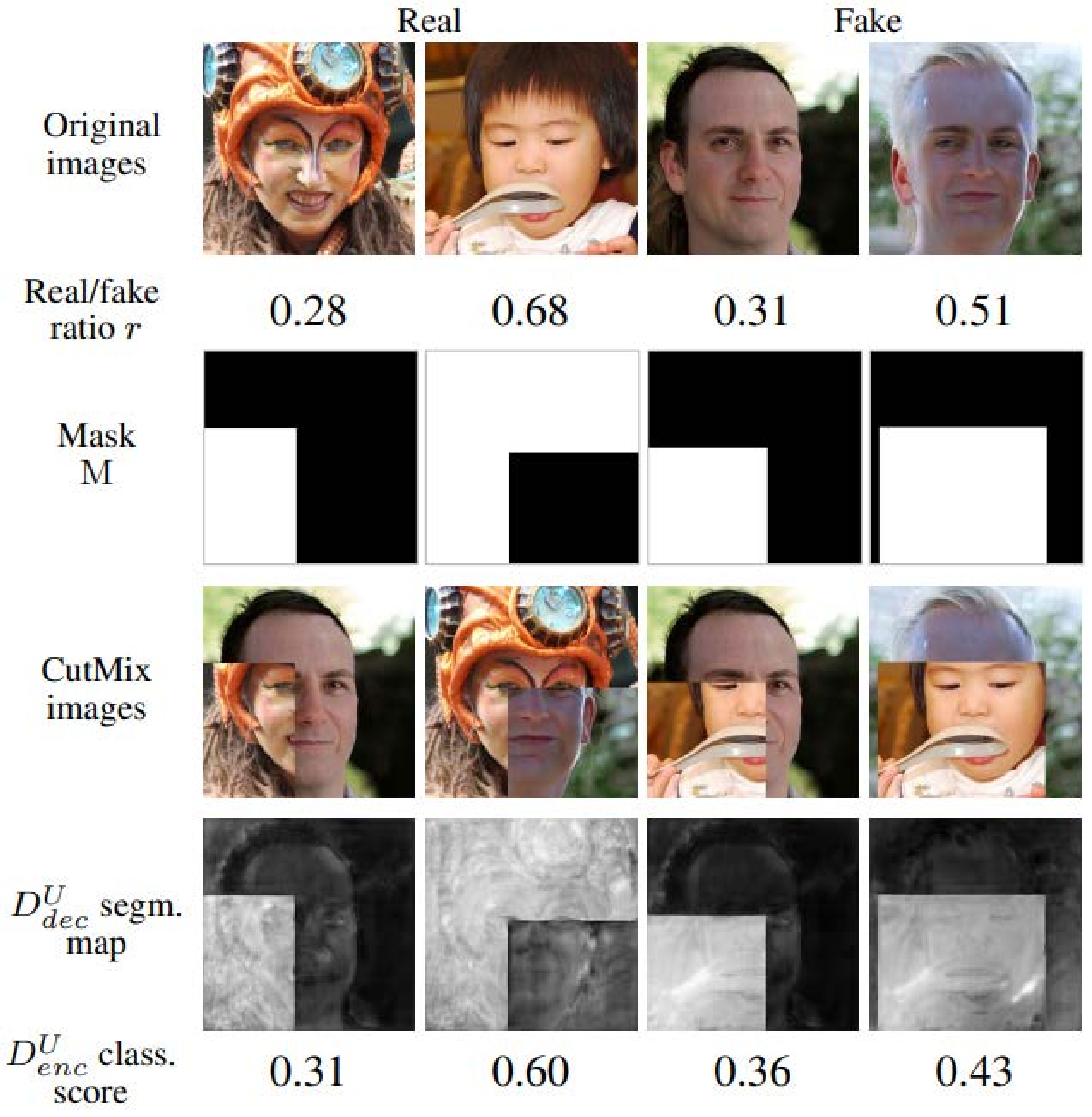
$$\boxed{\mathcal{L}_{D^U_{enc}} = -\mathbb{E}_x[\log D^U_{enc}(x)] - \mathbb{E}_z[\log(1 - D^U_{enc}(G(z)))],}$$

$$\boxed{\mathcal{L}_{D^U_{dec}} = -\mathbb{E}_x\Big[\sum_{i,j} \log[D^U_{dec}(x)]_{i,j}\Big] - \mathbb{E}_z\Big[\sum_{i,j} \log(1 - [D^U_{dec}(G(z))]_{i,j})\Big]}$$





Methods: Consistency Regularization



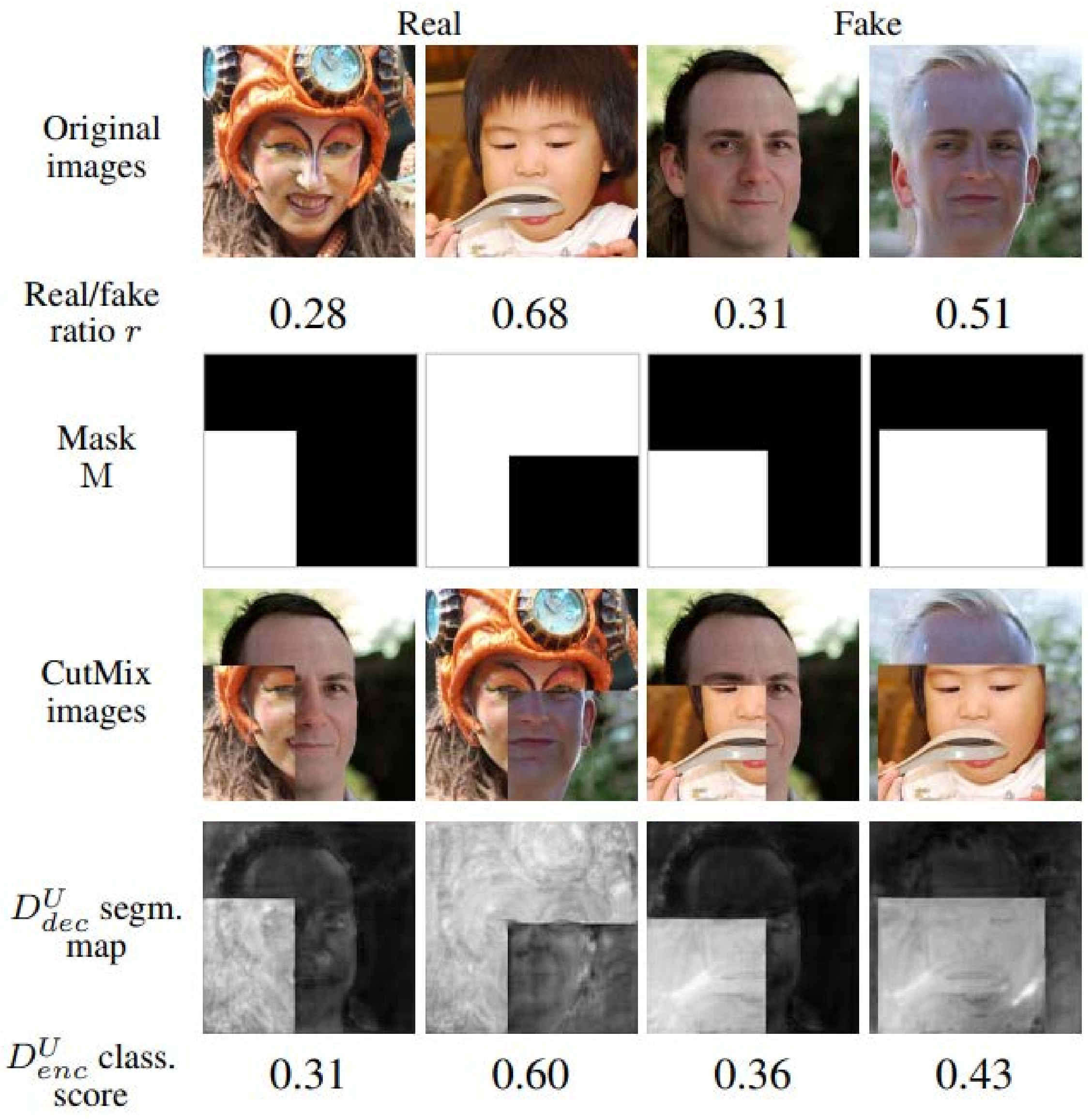
$$\mathrm{M} \in \{0,1\}^{W\times H}$$
$$\text{fake}(\mathrm{M}_{i,j} = 0)$$
$$\text{real}(\mathrm{M}_{i,j} = 1)$$

$$\tilde{x} = \mathrm{mix}(x, G(z), \mathrm{M}),$$
$$\mathrm{mix}(x, G(z), \mathrm{M}) = \mathrm{M} \odot x + (1-\mathrm{M}) \odot G(z), \tag{6}$$

$$D^U_{dec}(\mathrm{mix}(x, G(z), \mathrm{M})) \approx \mathrm{mix}(D^U_{dec}(x), D^U_{dec}(G(z)), \mathrm{M}),$$

$$\mathcal{L}^{cons}_{D^U_{dec}} = \Big\| D^U_{dec}\Big(\mathrm{mix}(x, G(z), \mathrm{M})\Big) - \mathrm{mix}\Big(D^U_{dec}(x), D^U_{dec}(G(z)), \mathrm{M}\Big)\Big\|^2, \tag{7}$$

Methods: Consistency Regularization



$$\mathrm{M} \in \{0,1\}^{W\times H}$$
$$\text{fake}\,(\mathrm{M}_{i,j} = 0)$$
$$\text{real}\,(\mathrm{M}_{i,j} = 1)$$

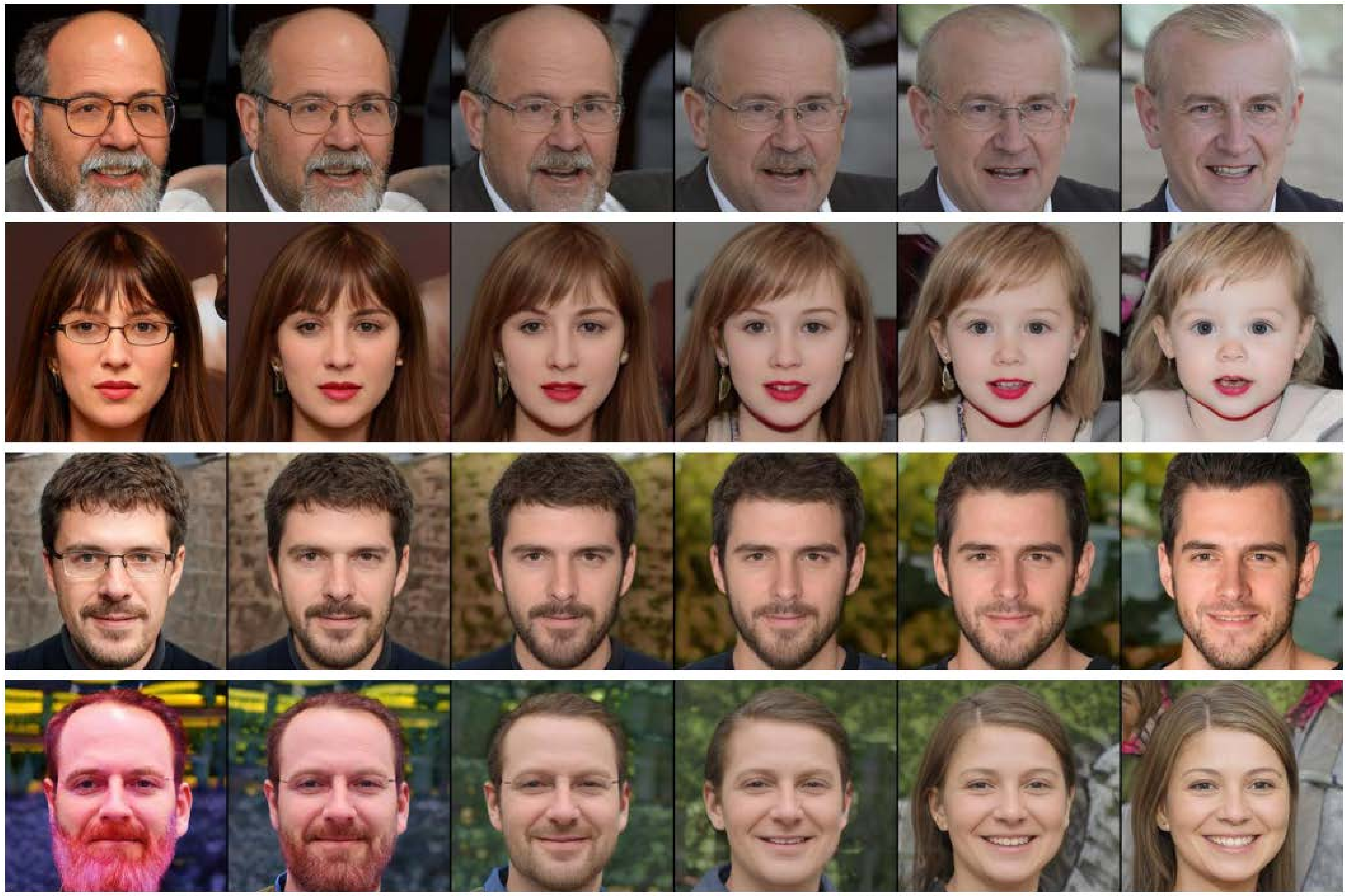
$$r = \frac{|\mathrm{M}|}{W * H}$$

$$\begin{aligned} \tilde{x} &= \text{mix}(x, G(z), \mathrm{M}), \\ \text{mix}(x, G(z), \mathrm{M}) &= \mathrm{M} \odot x + (1 - \mathrm{M}) \odot G(z), \end{aligned} \tag{6}$$

$$D^U_{dec}\big(\text{mix}(x, G(z), \mathrm{M})\big) \approx \text{mix}\big(D^U_{dec}(x), D^U_{dec}(G(z)), \mathrm{M}\big),$$

$$\boxed{\begin{aligned} \mathcal{L}^{cons}_{D^U_{dec}} = \Big\| D^U_{dec}\Big(\text{mix}(x, G(z), \mathrm{M})\Big) \\ - \text{mix}\Big(D^U_{dec}(x), D^U_{dec}(G(z)), \mathrm{M}\Big)\Big\|^2, \end{aligned} \quad (7)}$$

Experiments



Experiments

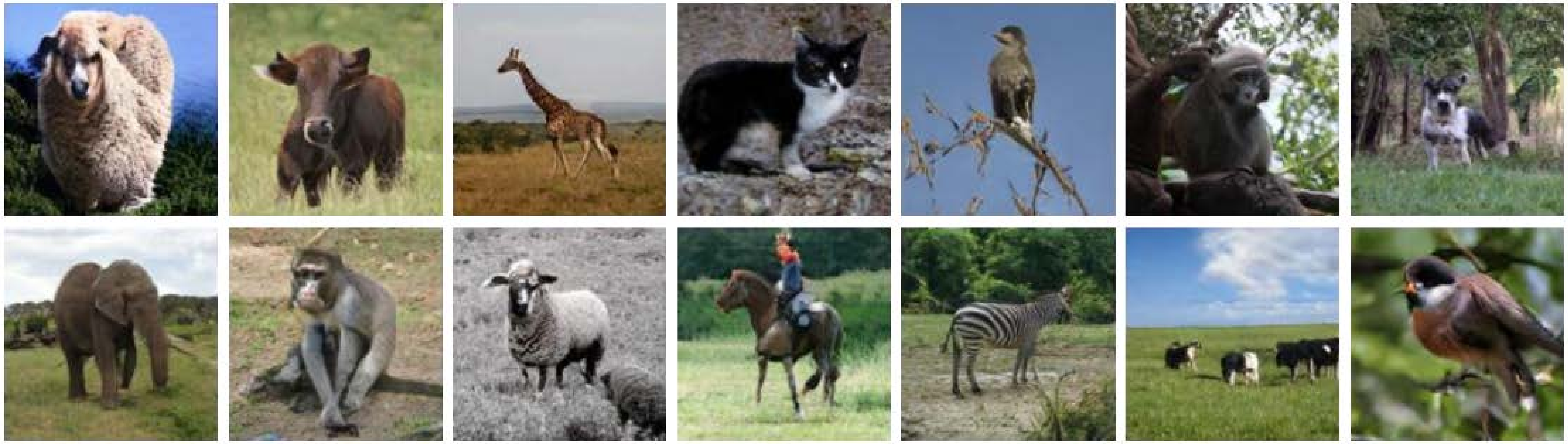


Figure 5: Images generated with U-Net GAN trained on COCO-Animals with resolution 128×128.

Experiments

Method	FFHQ Best FID↓	FFHQ Best IS↑	FFHQ Median FID↓	FFHQ Median IS↑	COCO-Animals Best FID↓	COCO-Animals Best IS↑	COCO-Animals Median FID↓	COCO-Animals Median IS↑
BigGAN [5]	11.48	3.97	12.42	4.02	16.37	11.77	16.55	11.78
U-Net GAN	**7.48**	**4.46**	**7.63**	**4.47**	**13.73**	**12.29**	**13.87**	**12.31**

Table 1: Evaluation results on FFHQ and COCO-Animals. We report the best and median FID score across 5 runs and its corresponding IS, see Section 4.2 for discussion.

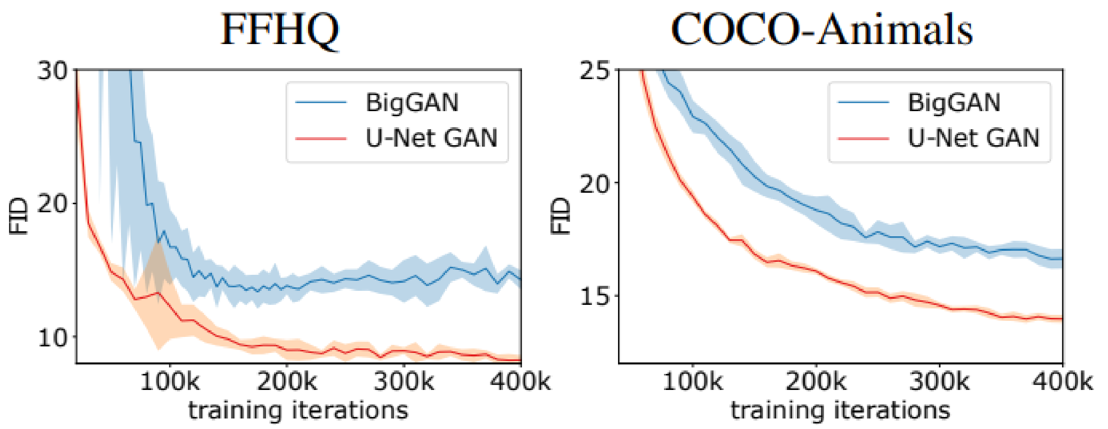


Figure 6: FID curves over iterations of the BigGAN model (blue) and the proposed U-Net GAN (red). Depicted are the FID mean and standard deviation across 5 runs per setting.

Method	Dataset	FID Best	FID Median	FID Mean	FID Std
BigGAN	COCO-Animals	16.37	16.55	16.62	0.24
U-Net GAN		**13.73**	**13.87**	**13.88**	**0.11**
BigGAN	FFHQ	11.48	12.42	12.35	0.67
U-Net GAN		**7.48**	**7.63**	**7.73**	**0.56**
BigGAN	CelebA	3.70	3.89	3.94	0.16
U-Net GAN		**2.03**	**2.07**	**2.08**	**0.04**

Table S2: Best, median, mean and std of FID values across 5 runs.

Experiments

Method	COCO-Animals	FFHQ
BigGAN [5]	16.55	12.42
U-Net based discriminator	15.86	10.86
+ CutMix augmentation	14.95	10.30
+ Consistency regularization	**13.87**	**7.63**

Table 2: Ablation study of the U-Net GAN model on FFHQ and COCO-Animals. Shown are the median FID scores. The proposed components lead to better performance, on average improving the median FID by 3.7 points over BigGAN [5]. See Section 4.2 for discussion.

Method	FID ↓	IS ↑
PG-GAN [19]	7.30	–
COCO-GAN [27]	5.74	–
BigGAN [5]	4.54	3.23
U-Net GAN	**2.95**	**3.43**

Table 3: Comparison with the state-of-the-art models on CelebA (128 × 128). See Section 4.2 for discussion.

		PyTorch		TensorFlow	
Dataset	Method	FID ↓	IS ↑	FID ↓	IS ↑
FFHQ	BigGAN [5]	11.48	3.97	14.92	3.96
(256 × 256)	U-Net GAN	**7.48**	**4.46**	**8.88**	**4.50**
COCO-Animals	BigGAN [5]	16.37	11.77	16.42	11.34
(128 × 128)	U-Net GAN	**13.73**	**12.29**	**13.96**	**11.77**
	PG-GAN [19]	–	–	7.30	–
CelebA	COCO-GAN [27]	–	–	5.74	–
(128 × 128)	BigGAN [5]	3.70	3.08	4.54	3.23
	U-Net GAN	**2.03**	**3.33**	**2.95**	**3.43**

Table S1: Evaluation results on FFHQ, COCO-Animals and CelebA with PyTorch and TensorFlow FID/IS scores. The difference lies in the choice of framework in which the inception network is implemented, which is used to extract the inception metrics. See Section A for discussion.

Experiments

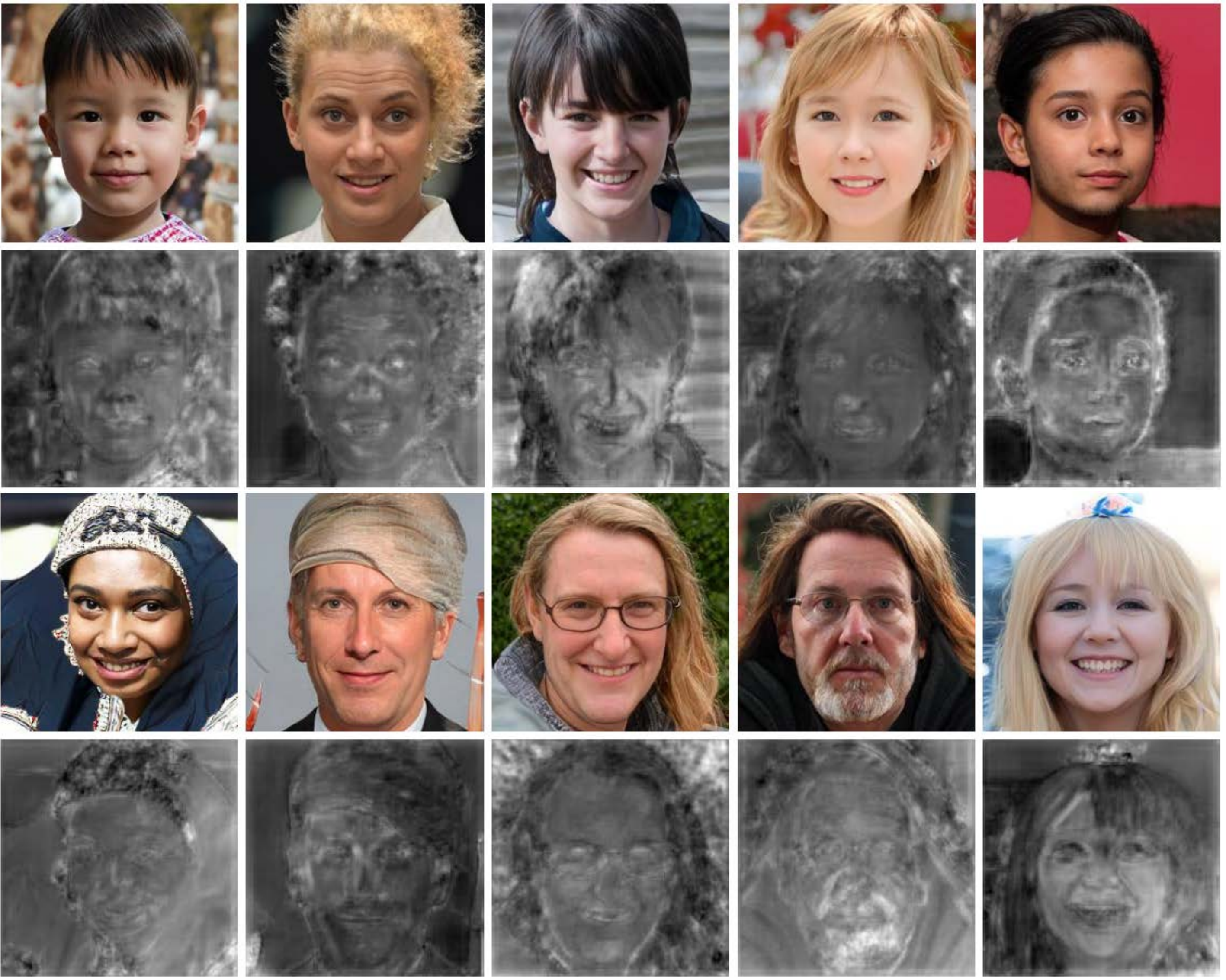


Figure S2: Samples generated by U-Net GAN and the corresponding real-fake predictions of the U-Net decoder. Brighter colors correspond to the discriminator confidence of pixel being real (and darker of being fake).

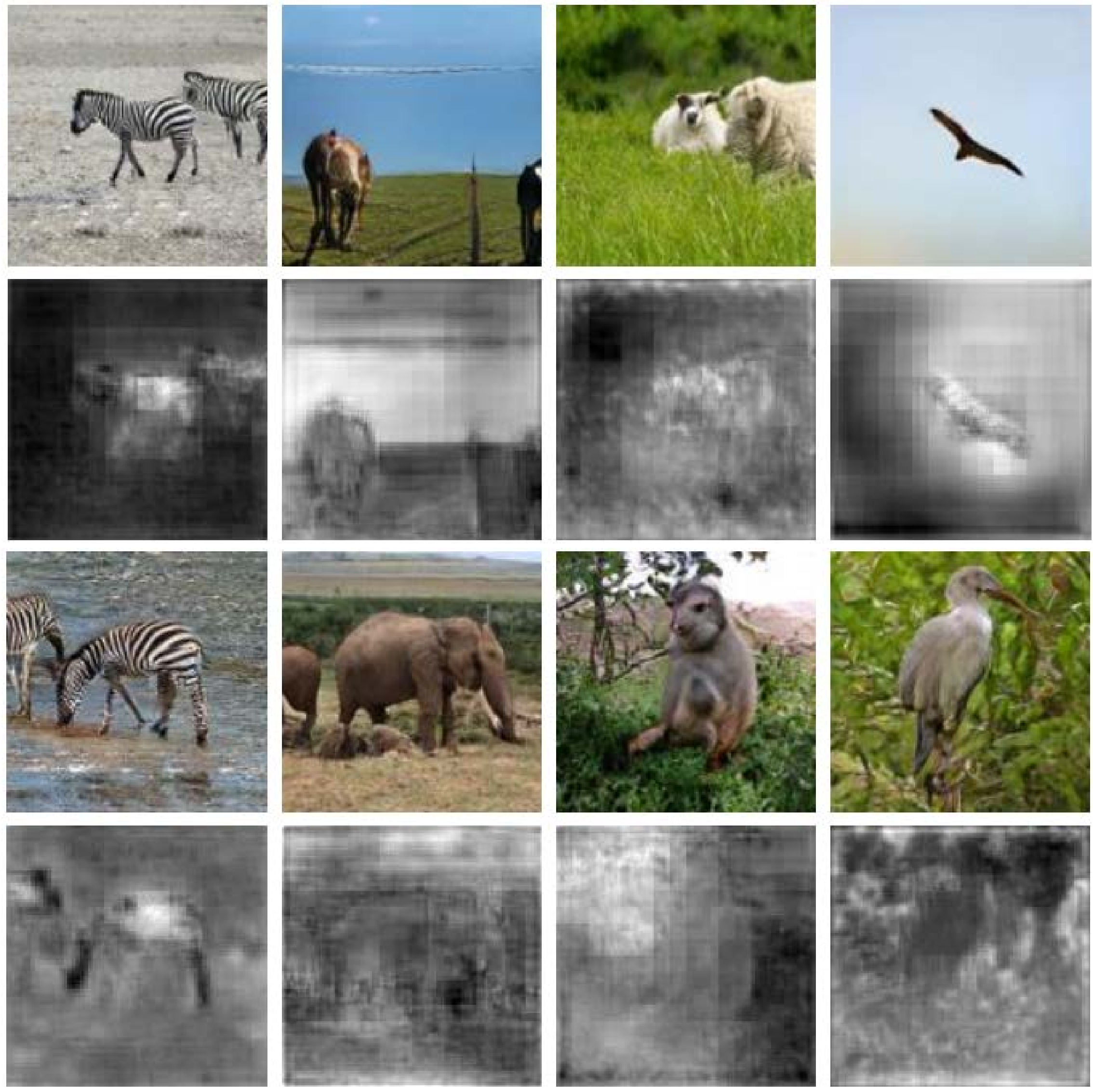


Figure S7: Generated samples on COCO-Animals and the corresponding U-Net decoder predictions.

Experiments

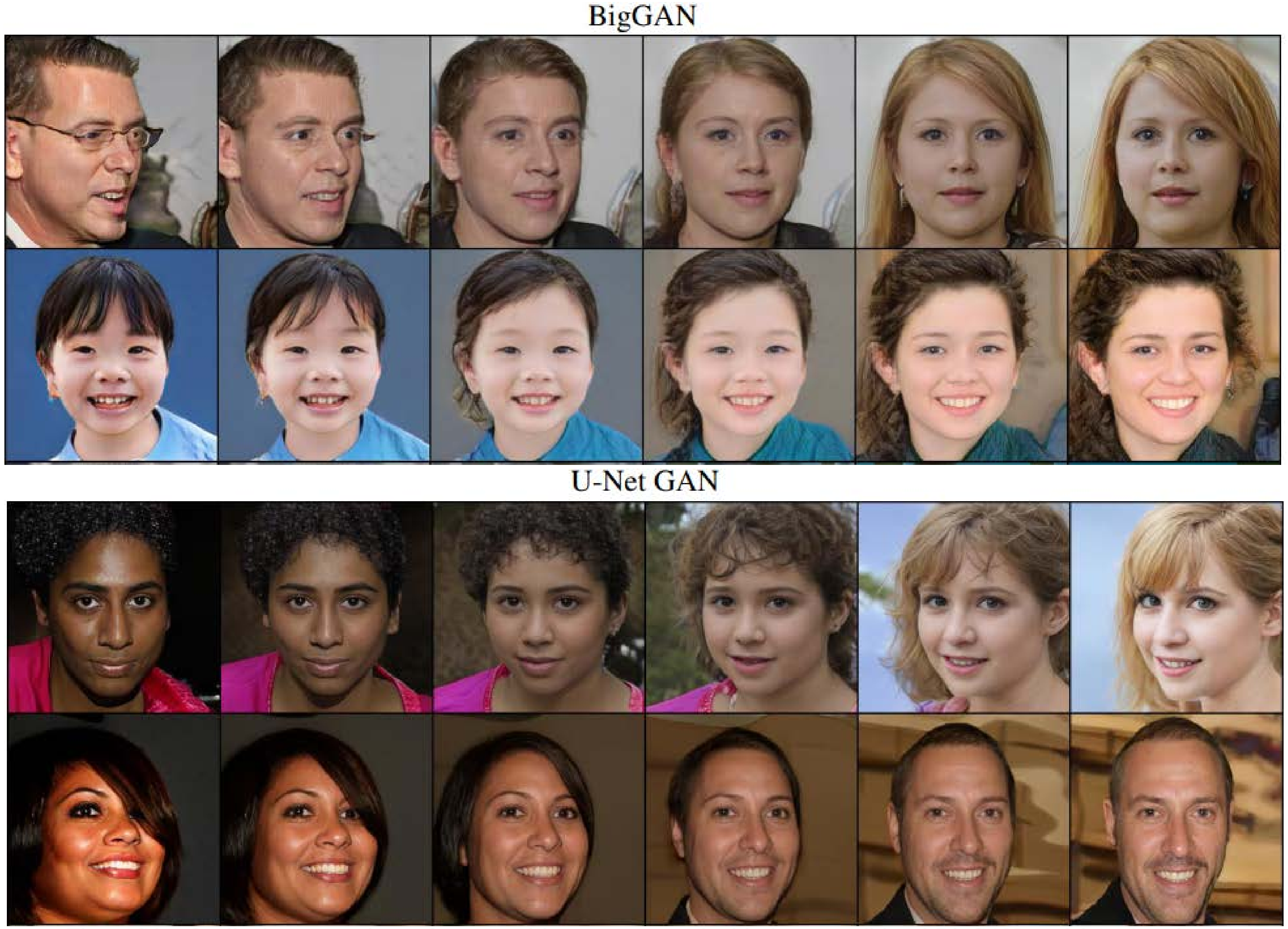


Figure S4: Qualitative comparison of uncurated images generated with the unconditional BigGAN model (top) and our U-Net GAN (bottom) on FFHQ with resolution 256 × 256. Note that the images generated by U-Net GAN exhibit finer details and maintain better local realism.

Experiments

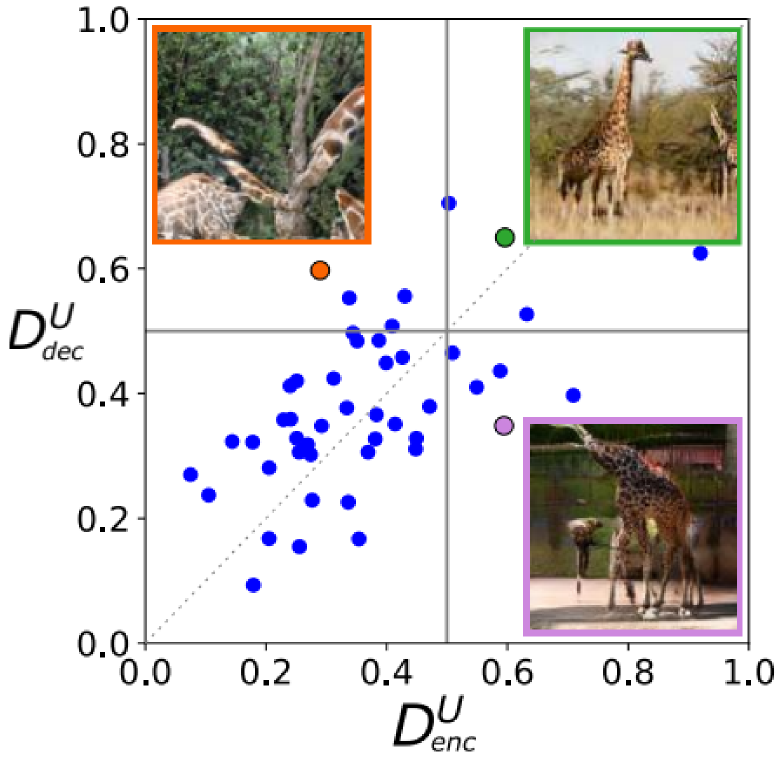


Figure 7: Visualization of the predictions of the encoder D^U_{enc} and decoder D^U_{dec} modules during training, within a batch of 50 generated samples. For visualization purposes, the D^U_{dec} score is averaged over all pixels in the output. Note that quite often decisions of D^U_{enc} and D^U_{dec} are not coherent with each other. As judged by the U-Net discriminator, samples in the upper left consist of locally plausible patterns, while not being globally coherent (example in orange), whereas samples in the lower right look globally coherent but have local inconsistencies (example in purple: giraffe with too many legs and vague background).

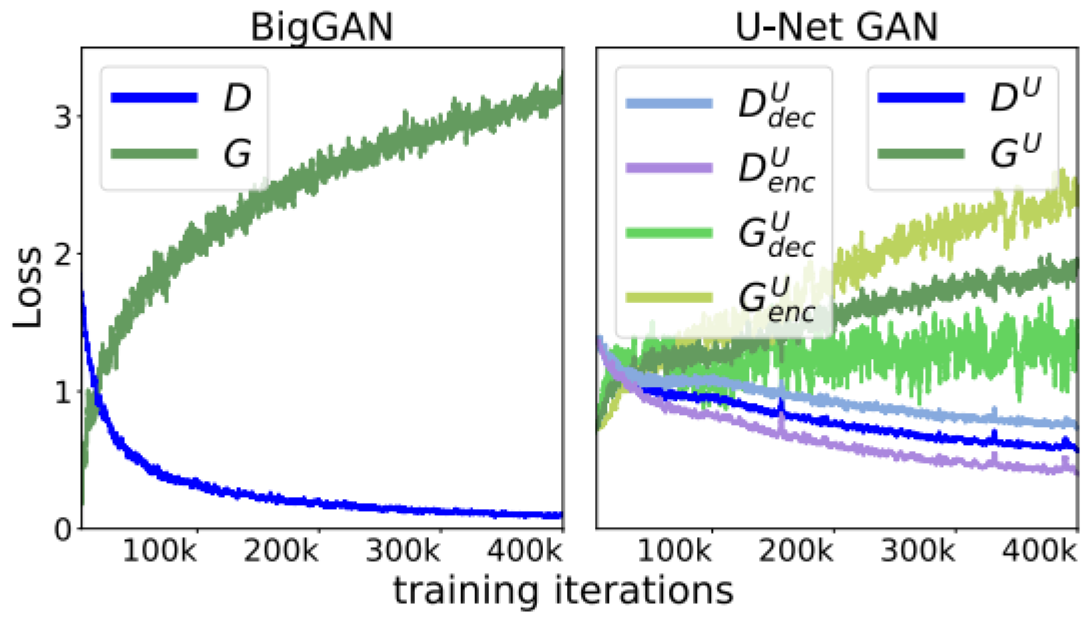


Figure 8: Comparison of the generator and discriminator loss behavior over training for U-Net GAN and BigGAN. The generator and discriminator loss of U-Net GAN is additionally split up into its encoder- and decoder components.

Conclusion

- **Proposing an alternative U-Net based architecture for the discriminator, which allows to provide both global and local feedback to the generator.**
- **U-Net based discriminator architecture allows to provide detailed per-pixel feedback to the generator while maintaining the global coherence of synthesized images.**
- **Introducing a consistency regularization technique for the U-Net discriminator based on the CutMix data augmentation**
- **Showing that all the proposed changes result in a stronger discriminator, enabling the generator to synthesize images with varying levels of detail, maintaining global and local realism.**
- **We demonstrate the improvement over the state-of-the-art BigGAN model [5] in terms of the FID score on three different datasets.**

pr
di
lo

Q&A

Question 1

- **논문의 3.2 Consistency regularization 챕터에서 Class-domain-perturbation이 무슨 말인가요?**
  - **Class가 정확하게 무엇을 지칭하나요?**
    - **해당 class-domain-perterbation에서 명시적으로 calss c={0,1}라고 적혀 있고, perturbation이 될 만한 부분은 real/fake class이라고 생각합니다.**
    - **논문의 해당 파트에 명시적으로 Class c = {0,1}로 적혀 있고, BigGAN의 condition 이 명확히 언급되지 않은 것으로 보아서도 e.g. real=1, fake=0 가 맞는 것 같습니다.**
  - **Class-domain-perturbation이 무슨 말인가요? 뒤에서 설명드리겠습니다.**

Question List

Question 2 (1/4)

- **Consistency regularization의 motivation이 무엇이고 구체적인 효과(성능이 왜 높아지는지)에 대해서 자세히 설명해주세요.**
    - **Real/Fake Domain의 bias 관점에서:**
    - **이미지를 [A] = [cutmix하고 D에 넣은 결과]와 [B] = [D에 넣은 결과를 cutmix] 한 것을 같도록 제약을 주는 것이 consistency regulation loss term인데요. CutMix output결과가 (real)M, (fake)1-M에 마스크 영역 의해 real/fake가 나뉘어지므로 real image pixel들은 real끼리 fake image pixel들은 fake끼리 뭉쳐 있는 상황이겠습니다. 만약 real pixels domain, / fake pixels domain별로 domain 차이에 의한 pixel data 분포에 bias있다고 가정해봅시다.**
    - **이중 Cutmix결과 이미지가 Enc(U-Net D ConvNet Encdoer) 이미지로 들어갈 때, 최종 Discriminator output(1,256,256)의 각 output pixel별로 대응되는 input의 Receptive field를 생각해보면, (앞에서 말한 real domain, fake domain 사이의 bias에 의해서) Receptive field에서 real pixel의 비율이 많은 곳은 pixel을 real이라고 판별할 가능성이 높고, fake pixel들의 비율이 많은 곳은 output을 fake라고 판별할 가능성이 높아질 수 있을 것 같습니다. (domain간의 bias 정보를 Discriminator가 활용할 수 있기 때문이에요.)**
- **Class-domain-perturbation이 무슨 말인가요?**
    - **Class-domain-perturbation이라는 것은 바로 Output의 특정 pixel에 대응하는 Receptive field에서 fake pixels, real pixels 수의 비율 차이에 따라 각 개별 pixel 정보에 알게 모르게 남아있던 일종의 domain 별 bias값을 Discriminator가 활용하는 문제를 지칭하는 것이 아닐까 생각합니다. 이런 문제는 Cutmix를 수행해주는 의도에 맞지 않으니 제약해주는 게 맞지 않을까 합니다.**

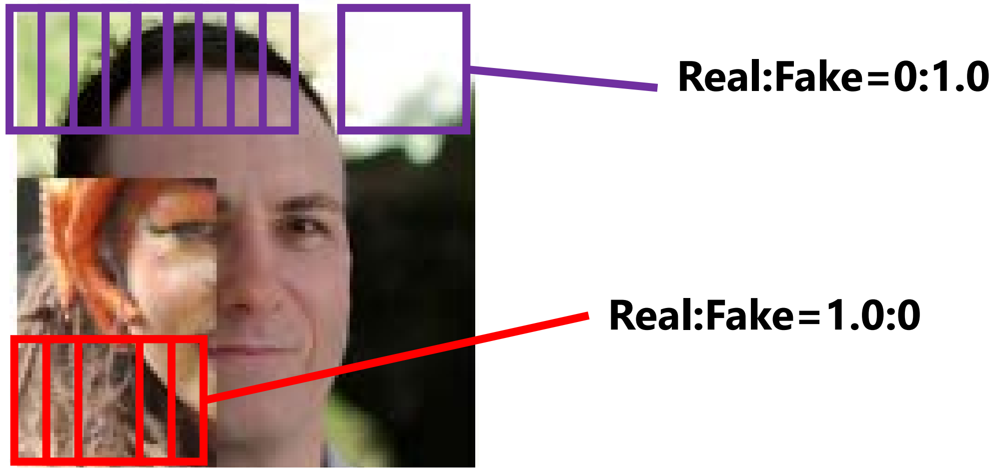
Question 2 (2/4)

- **Consistency regularization의 motivation이 무엇이고 구체적인 효과(성능이 왜 높아 지는지)에 대해서 자세히 설명해주세요.**
  - **반면에 정보를 학습할 때 주변영역에 대한 dependence를 학습하지 못하게 하는 효과가 조금이라도 더 생기고 동시에 pixel별 독립적인 판단을 하도록 제약을 걸어 주게 되는 것 같습니다.**
  - **왜냐하면 consistency 제약 Loss를 전체 loss term에 추가해주었을 때, 구체적으로 (cutmix를 하는 순서와 D에 통과시키는 순서가 달라지는 것)[A], [B]사이에 어떤 차이가 있느냐 자세히 살펴본다면, ([A]일 때와 [B]일 때 모두) receptive field에 real pixel만 있거나 fake pixel만 있을 때는 동일한 연산과정을 거쳐 동일한 값이 나오게 되므로, Discriminator입장에서 Q2 (3/4) 그림에서처럼 결과가 동일하게 됩니다.**
  - **그러나 output의 특정 pixel 지점에 대응하는 Receptive field에서 real/fake mask 주변의 경계선에서 real pixels과 fake pixels이 섞여 있다면, Q2 (4/4) 그림에서처럼, convNet window filter가 sliding함에 따라 real/fake 자체의 비율이 그 때 그 때 달라지므로, 모델 입장에서 real과 fake의 비율 정보를 discriminating 결과에 덜 참고하게 학습이 되는 제약 효과가 주어질 것 같습니다.**
  - **결과적으로 Discriminator가 real/fake 비율이나 그 비율에 따른 bias 정보를 컷닝(?)하기보다는 개별 pixel 정보들을 independent하게 좀 더 집중하게 하는 효과를 주지 않나 싶습니다.**
    따라서, consistency 제약을 주어 학습을 하면 Discriminator가 (pixel 단위로 독립적으로) independently pixel-wise하게 pixel 정보만을 활용해서 Discriminator가 집중해서 학습할 수 있도록 유도할 수 있고, 그에 따라 Generator도 전체적으로 합성 이미지가 semantic하게 결과를 판별하도록 학습 할 수 있을 것 같습니다.

Question 2 (3/4)

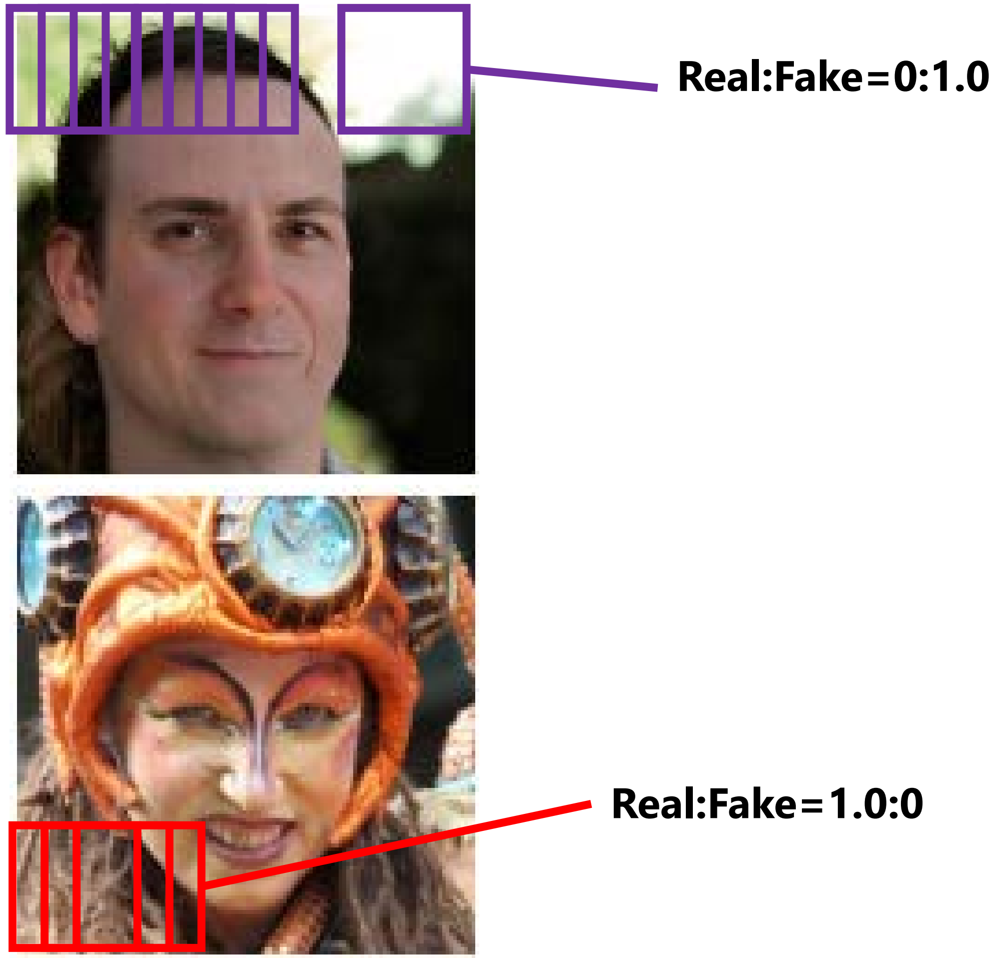
[A]

= Cumix를 먼저해서 D에 통과시키는 경우



[B]

= real/fake를 각각 D에 통과시키고 나서, 그 다음에 Cutmix하는 경우

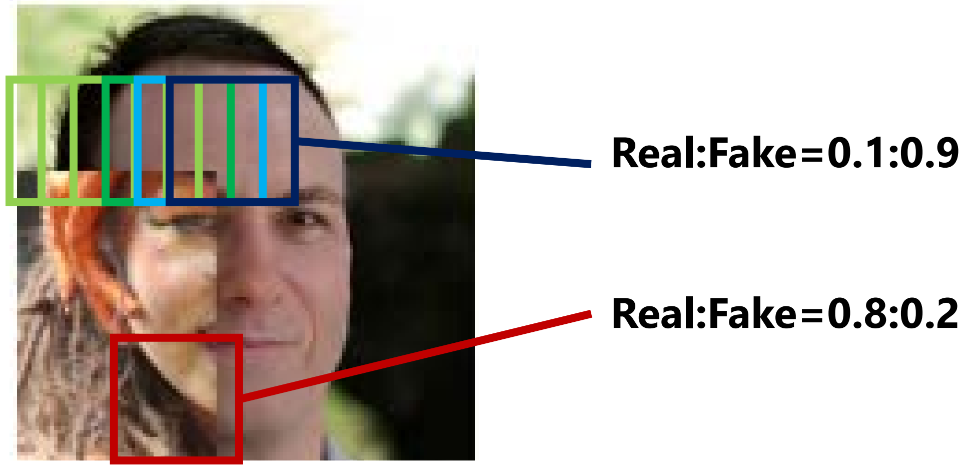


Receptive field안에 real pixels만, 혹은 fake pixes만 있는 경우, [A], [B]의 연산결과는 동일하다.

Question 2 (4/4)

[A]

= Cumix를 먼저해서 D에 통과시키는 경우

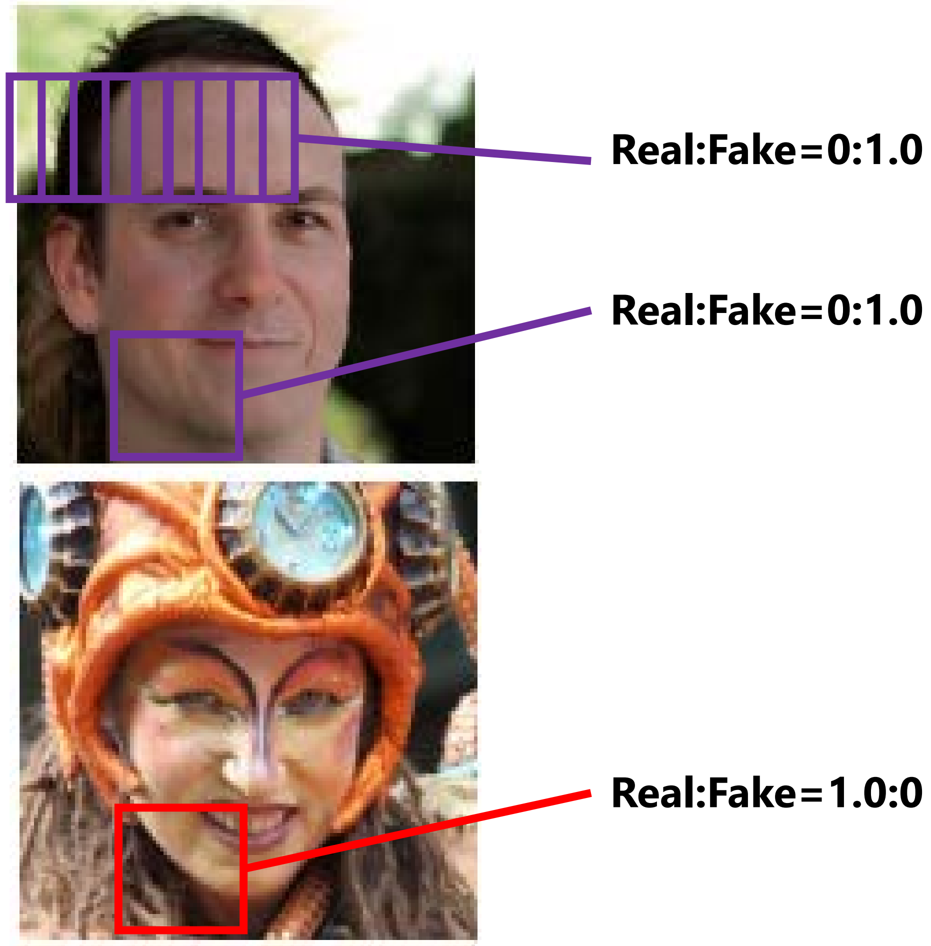


Receptive field안에 real pixels 일부와 fake pixes일부가 섞여 있는 경우, [A], [B]의 연산 결과는 다르다.

ConvLayer Filter가 input에 대해서 sliding 하게 Conv 연산을 수행할 때, 한 receptive field 안의 real, fake의 비율도그 때 그 때 달라진다. ➔ 즉, real/fake의 비율이 sliding 할 때마다 달라지게 되면서, Discriminator입장에서는 real과 fake의 비율의 정보를 활용하는 것이 loss를 줄이는 방향과 직결되지 않는다.

[B]

= real/fake를 각각 D에 통과시키고 나서, 그 다음에 Cutmix하는 경우



References

[1] “A U-Net Based Discriminator for Generative Adversarial Networks”, CVPR 2019

[2] “mixup: BEYOND EMPIRICAL RISK MINIMIZATION”

[3] “Improved Regularization of Convolutional Neural Networks with Cutout”

[4]“Large Scale GAN Training for High Fidelity Natural Image Synthesis” (BigGAN)