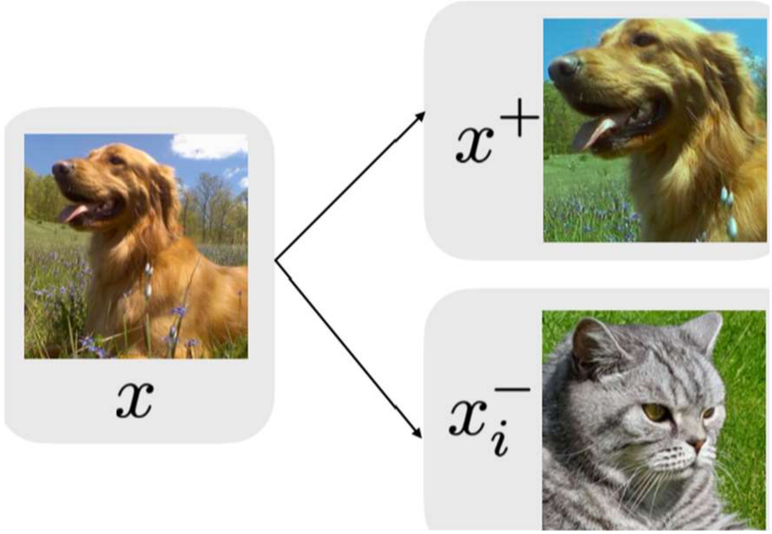
Contrastive Learning for Low-Level Tasks in Computer Vision

Contrastive learning is one of the most powerful approaches for representation learning.The goal of contrastive learning is to bring anchors closer to positive samples while pushing away negative samples in the latent embedding space



Constrative learning is widely used in the high level field, such as Moco, simCLR, etc., for classification tasks, but its application in the low level field is still limited.

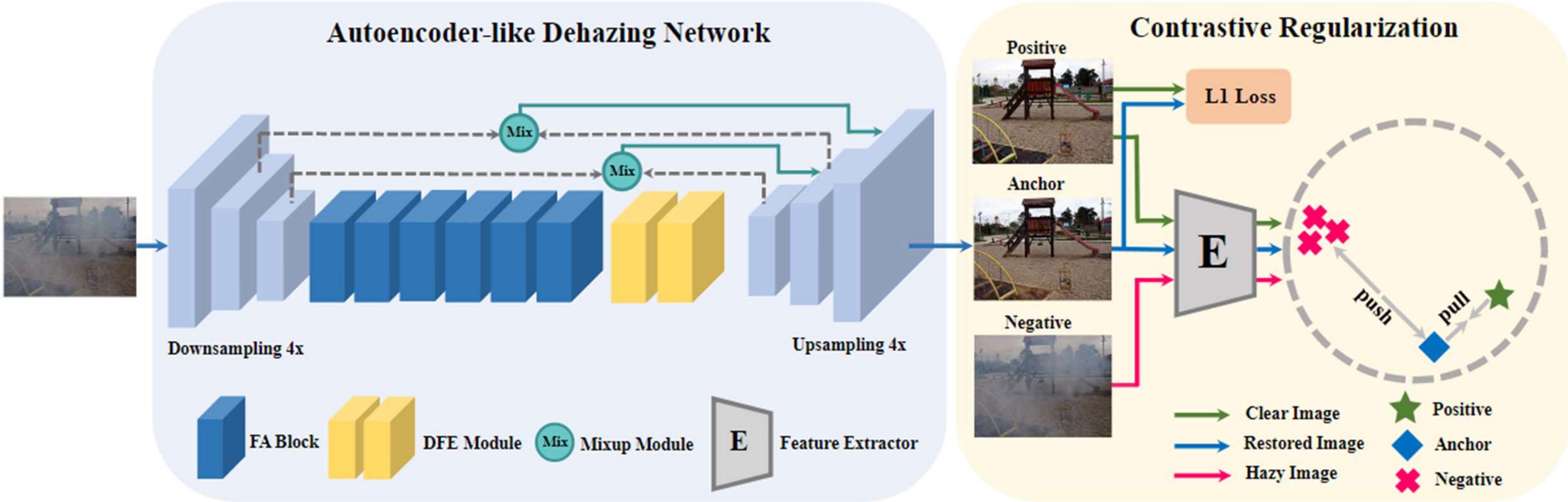
In low-level task, we mainly consider the following three aspects:
1.constructing suitable positive and negative samples to construct positive and negative pairs.
2.constructing appropriate models to extract features from the latent feature space.
3.designing a reasonable contrastive loss to pull the anchors into the positive samples and away from the negative samples in thepotential space.

南京航空航天大學
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

Contrastive Learning for Compact Single Image Dehazing

Haiyan Wu [1,*] Yanyun Qu [2,*] Shaohui Lin [1†] Jian Zhou [3],
Ruizhi Qiao [3], Zhizhong Zhang [1], Yuan Xie [1†] Lizhuang Ma [1],
[1]School of Computer Science and Technology, East China Normal University, Shanghai, China
[2]School of Information Science and Engineering, Xiamen University, Fujian, China
[3]Tencent Youtu Lab, Shanghai, China

CVPR 2021



Autoencoder-like Dehazing Network
Mix
Mix
Downsampling 4x
Upsampling 4x
FA Block
DFE Module
Mix
Mixup Module
E
Feature Extractor
Contrastive Regularization
Positive
Anchor
Negative
L1 Loss
E
push
pull
Clear Image
Restored Image
Hazy Image
Positive
Anchor
Negative

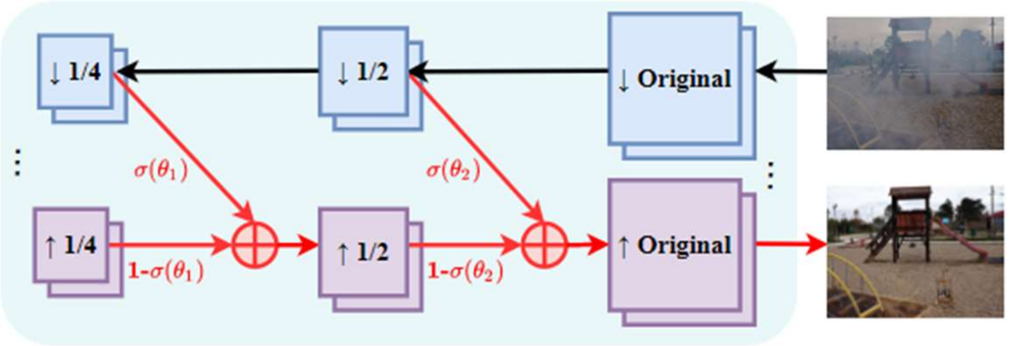


Figure 4. Adaptive mixup. The first and second rows are down-sampling and upsampling operations, respectively.

$$f_{\uparrow 2} = \mathrm{Mix}(f_{\downarrow 1}, f_{\uparrow 1}) = \sigma(\theta_1) * f_{\downarrow 1} + (1 - \sigma(\theta_1)) * f_{\uparrow 1}$$
$$f_{\uparrow} = \mathrm{Mix}(f_{\downarrow 2}, f_{\uparrow 2}) = \sigma(\theta_2) * f_{\downarrow 2} + (1 - \sigma(\theta_2)) * f_{\uparrow 2}$$

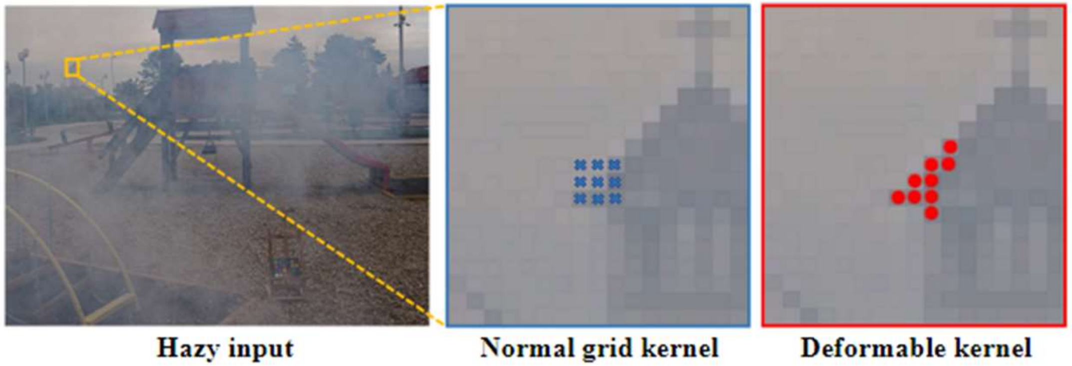
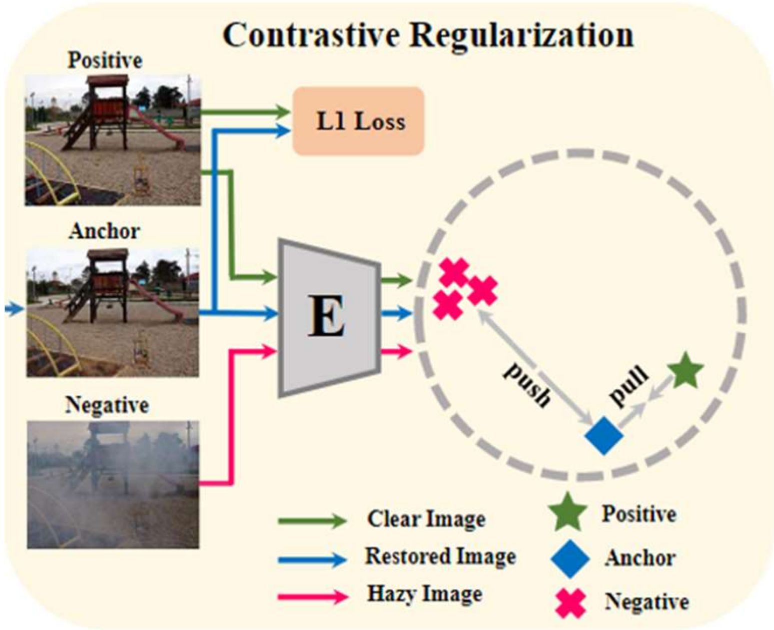


Figure 5. Dynamic feature enhancement module.

Method



E Feature Extractor — Potential features of layers 1, 5, 9, and 13 of the VGG-19 network selected.

Anchor — Image after dehazing through the dehazing network.

Positive — Clear images in the RESIDE dataset.

Negative — foggy images in the RESIDE dataset.

$$min\|J-\phi(I,w)\|_1+\beta\sum_{i=1}^{n}\omega_i\cdot\frac{D\big(G_i(J),G_i(\phi(I,w))\big)}{D\big(G_i(I),G_i(\phi(I,w))\big)}$$

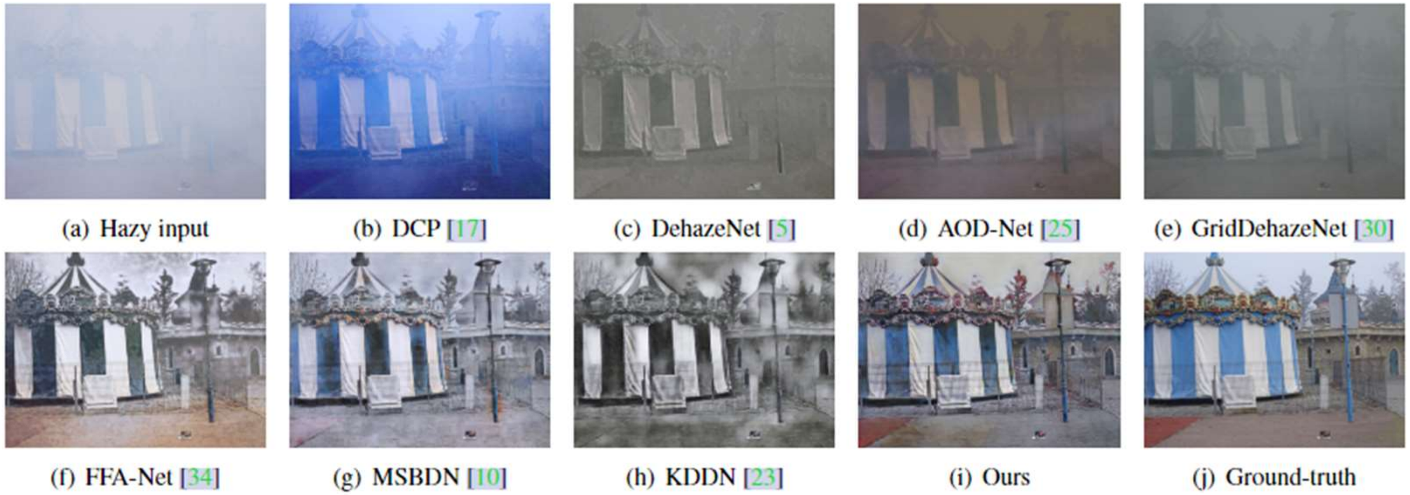


Figure 7. Visual comparison on the Dense-Haze dataset.

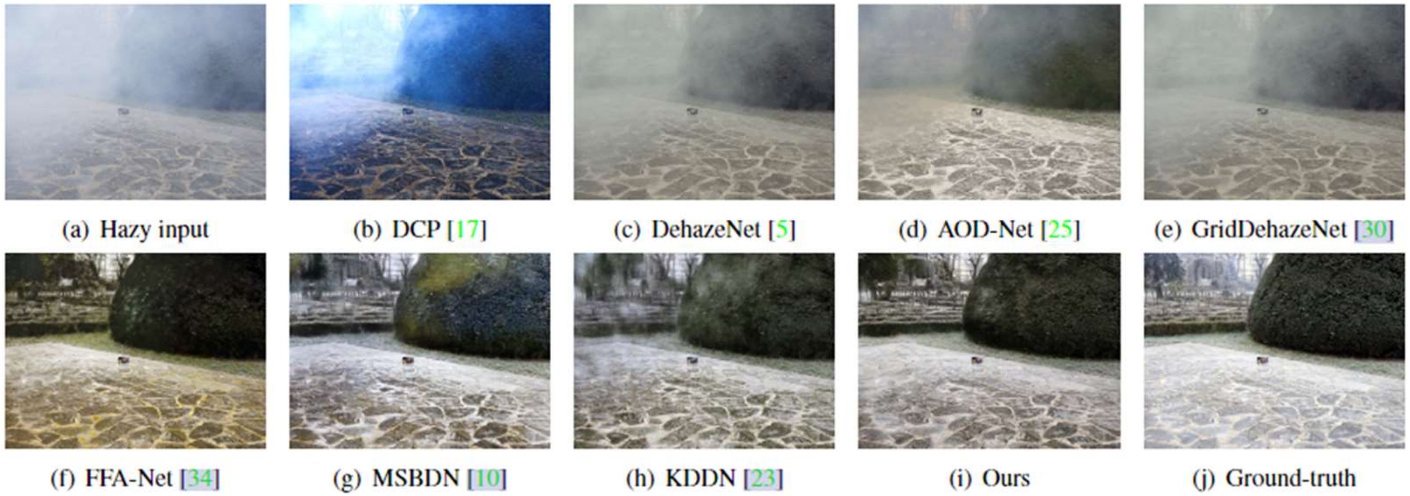


Figure 8. Visual comparison on NH-HAZE datasets.

Table 1. Quantitative comparisons with SOTA methods on the synthetic and real-world dehazing datasets.

Method	SOTS [27]		Dense-Haze [1]		NH-HAZE [2]		# Param
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	
(TPAMI'10) DCP [17]	15.09	0.7649	10.06	0.3856	10.57	0.5196	-
(TIP'16) DehazeNet [5]	20.64	0.7995	13.84	0.4252	16.62	0.5238	0.01M
(ICCV'17) AOD-Net [25]	19.82	0.8178	13.14	0.4144	15.40	0.5693	0.002M
(ICCV'19) GridDehazeNet [30]	32.16	0.9836	13.31	0.3681	13.80	0.5370	0.96M
(AAAI'20) FFA-Net [34]	36.39	0.9886	14.39	0.4524	19.87	0.6915	4.68M
(CVPR'20) MSBDN [10]	33.79	0.9840	15.37	**0.4858**	19.23	0.7056	31.35M
(CVPR'20) KDDN [23]	34.72	0.9845	14.28	0.4074	17.39	0.5897	5.99M
(ECCV'20) FDU [11]	32.68	0.9760	-	-	-	-	-
Ours	**37.17**	**0.9901**	**15.80**	0.4660	**19.88**	**0.7173**	2.61M

Table 4. Comparisons of different positive and negative sample rates on CR. The baseline is AECR-Net with the rate of 1:1.

Rate	# Positive	# Negative	PSNR	SSIM
1:1	1	1	37.17	0.9901
$1{:}r$	1	10	**37.41**	**0.9906**
$r{:}1$	10	1	35.61	0.9862
$r{:}r$	10	10	35.65	0.9861

Table 2. Ablation study on AECR-Net. * denotes only positive samples are used for training. SC means skip connection.

Model	CR	PSNR	SSIM
base	-	33.85	0.9820
base+mixup	-	34.04	0.9838
base+DFE	-	35.50	0.9853
base+DFE+SC	-	35.59	0.9858
base+DFE+mixup	-	36.20	0.9869
base+DFE+mixup+CR*	√(w/o negative)	36.46	0.9889
Ours	√	**37.17**	**0.9901**

Contrastive Learning for Unpaired Image-to-Image Translation

Taesung Park[1]   Alexei A. Efros[1]   Richard Zhang[2]   Jun-Yan Zhu[2]

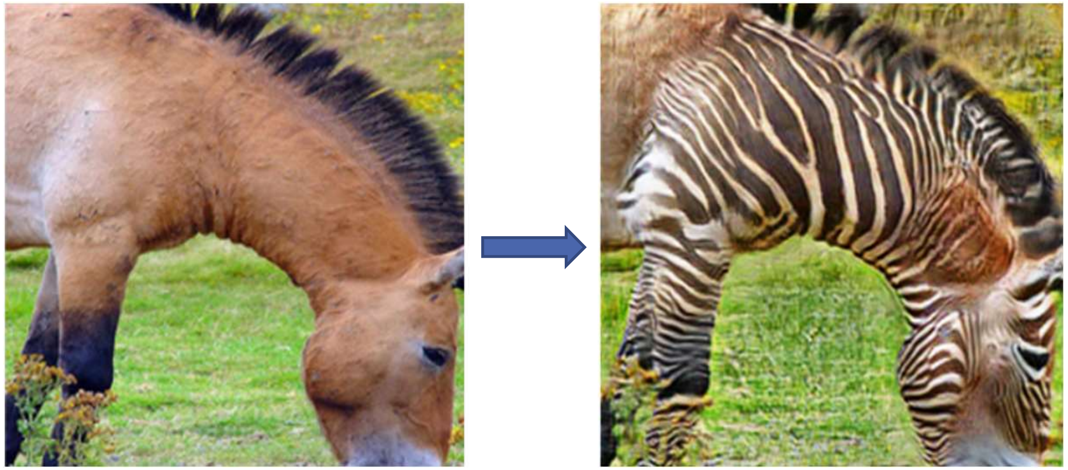
University of California, Berkeley[1]   Adobe Research[2]

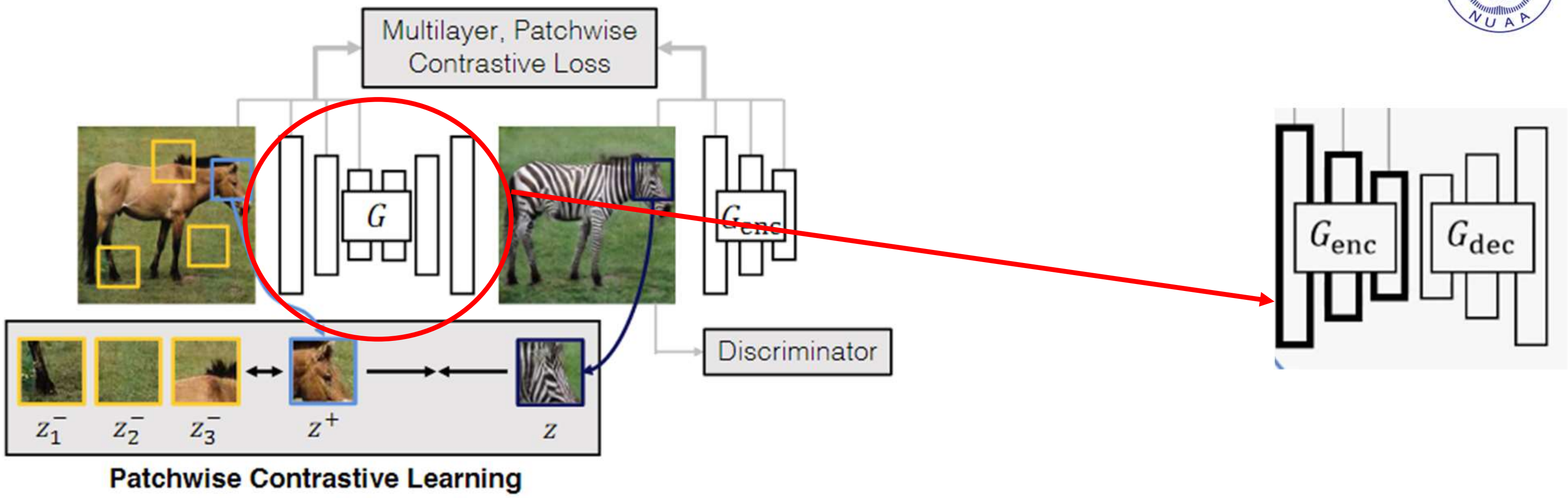
ECCV 2020

Target： While preserving the structure of the input image, incorporate the appearance of the target image.

Classic example: converting a horse to a zebra

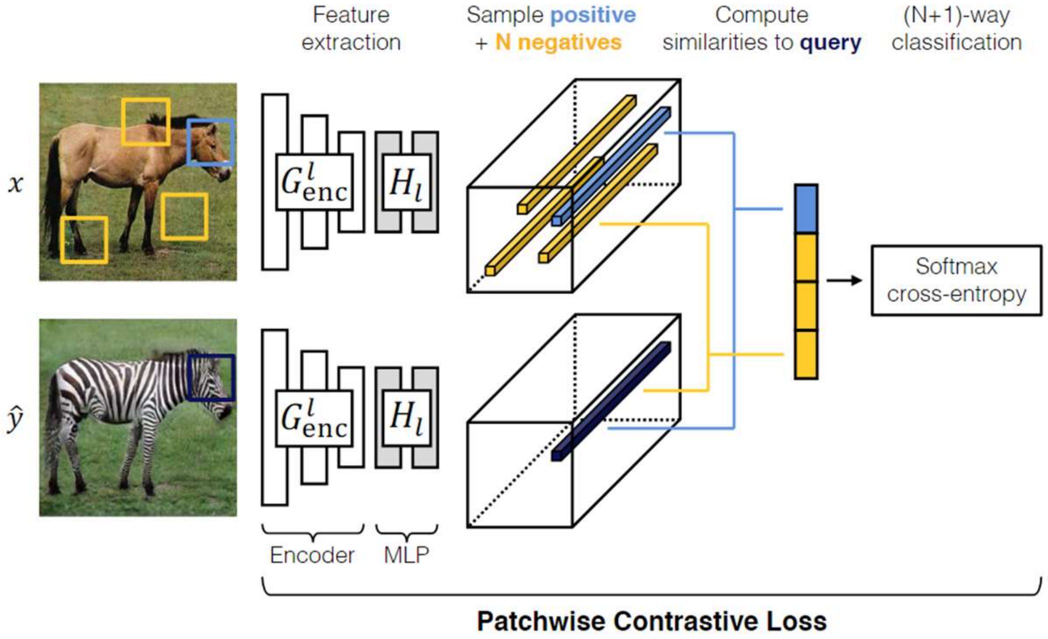
We wish for the output to take on the appearance of the target domain (a zebra), while retaining the structure, or content, of the specific input horse.

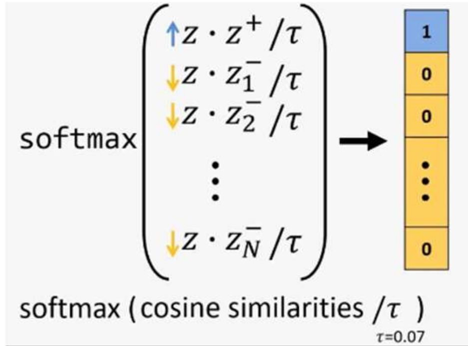




$$\hat{\boldsymbol{y}} = G(\boldsymbol{z}) = G_{\mathrm{dec}}(G_{\mathrm{enc}}(\boldsymbol{x}))$$

Cycling in two directions is usually included in Cycle GAN, but in the method of this paper, only one direction of transformation is used, avoiding the use of the opposite direction of transformation for assisting cycle consistency.



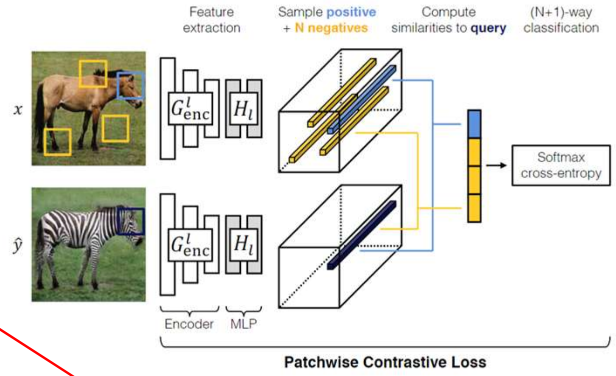


$$\ell(\boldsymbol{v}, \boldsymbol{v}^{+}, \boldsymbol{v}^{-}) = -\log\left[\frac{\exp(\boldsymbol{v}\cdot\boldsymbol{v}^{+}/\tau)}{\exp(\boldsymbol{v}\cdot\boldsymbol{v}^{+}/\tau) + \sum_{n=1}^{N}\exp(\boldsymbol{v}\cdot\boldsymbol{v}_n^{-}/\tau)}\right]$$

$$\ell(\boldsymbol{v}, \boldsymbol{v}^+, \boldsymbol{v}^-) = -\log\left[\frac{\exp(\boldsymbol{v}\cdot\boldsymbol{v}^+/\tau)}{\exp(\boldsymbol{v}\cdot\boldsymbol{v}^+/\tau) + \sum_{n=1}^{N}\exp(\boldsymbol{v}\cdot\boldsymbol{v}_n^-/\tau)}\right]$$

anchor positive negative

$$\mathcal{L}_{\mathrm{PatchNCE}}(G, H, X) = \mathbb{E}_{\boldsymbol{x}\sim X}\sum_{l=1}^{L}\sum_{s=1}^{S_l}\ell(\hat{\boldsymbol{z}}_l^s, \boldsymbol{z}_l^s, \boldsymbol{z}_l^{S\backslash s}).$$



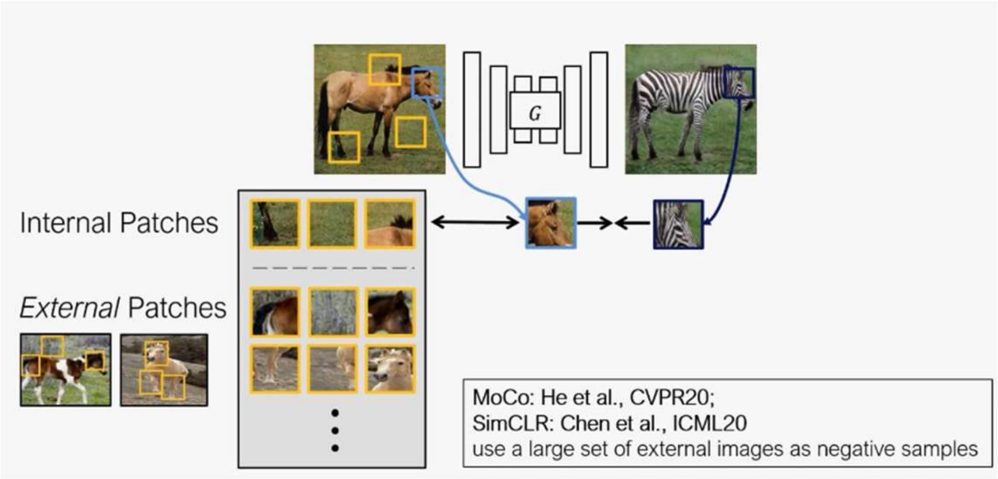
$$\hat{\boldsymbol{z}}_l^s \in \mathbb{R}^{C_l}$$

$$\boldsymbol{z}_l^s \in \mathbb{R}^{C_l}$$

$$\boldsymbol{z}_l^{S\backslash s} \in \mathbb{R}^{(S_l-1)\times C_l}$$

$$\{\hat{\boldsymbol{z}}_l\}_L = \{H_l(G_{\mathrm{enc}}^l(G(\boldsymbol{x})))\}_L$$

$$\{\boldsymbol{z}_l\}_L = \{H_l(G_{\mathrm{enc}}^l(\boldsymbol{x}))\}_L$$

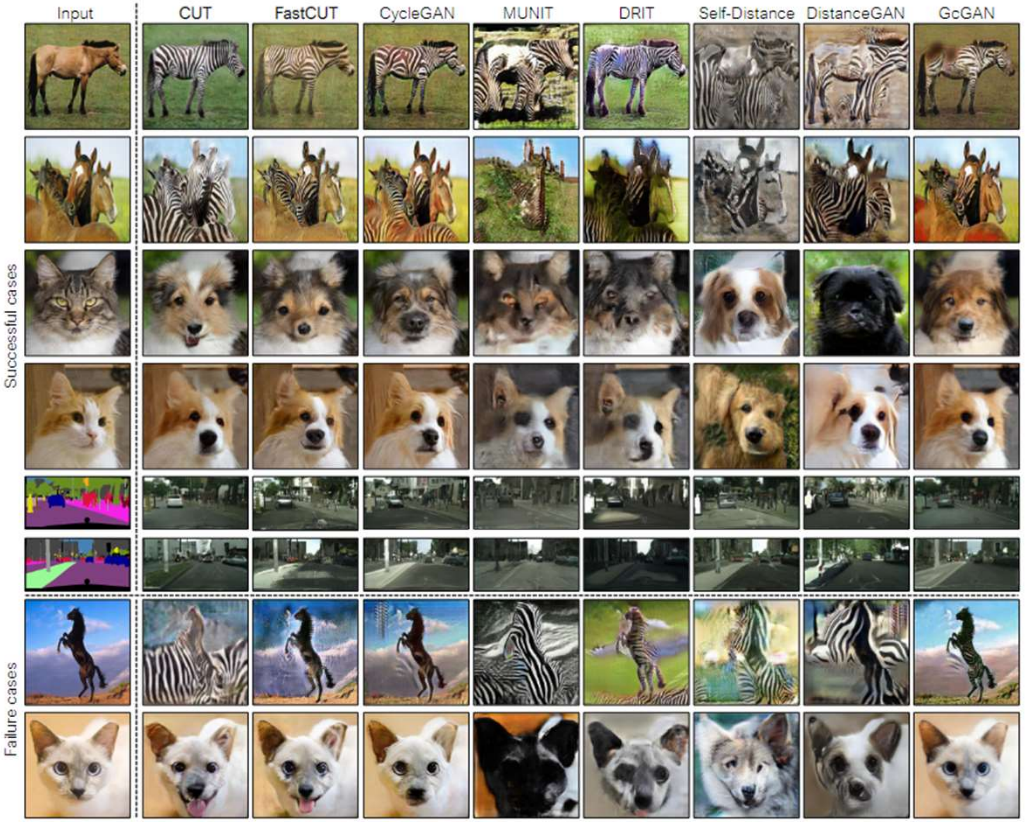


$$\mathcal{L}_{\text{external}}(G, H, X) = \mathbb{E}_{\boldsymbol{x}\sim X, \tilde{\boldsymbol{z}}\sim Z^-} \sum_{l=1}^{L} \sum_{s=1}^{S_l} \ell(\hat{\boldsymbol{z}}_l^s, \boldsymbol{z}_l^s, \tilde{\boldsymbol{z}}_l)$$

$$\mathcal{L}_{\text{GAN}}(G, D, X, Y) + \lambda_X \mathcal{L}_{\text{PatchNCE}}(G, H, X) + \lambda_Y \mathcal{L}_{\text{PatchNCE}}(G, H, Y)$$

CUT: $\lambda_X = \lambda_Y = 1$

FastCUT: $\lambda_X = 10, \lambda_Y = 0$



Method	Cityscapes				Cat→Dog	Horse→Zebra		
	mAP↑	pixAcc↑	classAcc↑	FID↓	FID↓	FID↓	sec/iter↓	Mem(GB)↓
CycleGAN [89]	20.4	55.9	25.4	76.3	85.9	77.2	0.40	4.81
MUNIT [44]	16.9	56.5	22.5	91.4	104.4	133.8	0.39	3.84
DRIT [41]	17.0	58.7	22.2	155.3	123.4	140.0	0.70	4.85
Distance [4]	8.4	42.2	12.6	81.8	155.3	72.0	**0.15**	2.72
SelfDistance [4]	15.3	56.9	20.6	78.8	144.4	80.8	0.16	2.72
GCGAN [18]	21.2	63.2	26.6	105.2	96.6	86.7	0.26	2.67
CUT	**24.7**	**68.8**	**30.7**	**56.4**	**76.2**	**45.5**	0.24	3.33
FastCUT	19.1	59.9	24.3	68.8	94.0	73.4	**0.15**	**2.25**

Thanks!