DesnowNet survey and CycleSnowGAN

Duowen(Justin) Chen
Columbia University
duowenchen1998@gmail.com

Abstract

Data-driven methods for snow removal have been proposed and research for recent years. To better understand the specific problem of snow particles compared with other weather particles, we review the first data-driven method designed for snow particle removal called DesnowNet [7]. However, such method rely on predefined snow-particle representation which doesn't necessarily correct. We propose an improvement taking advantage of the idea from CycleGAN [12]. We implicitly model the rule that snow-particles are being added to images and let Cycle-consistency helping us reversely apply such rule to get a snow-removed image. Meantime, our method won't be constraint by the supervised pairing input(Snow100K) which is using randomized behavior to approximate snow particles. This will allow us to use real-world snow data and real-world clean data to do training.

1. Introduction

In Computer Vision, weather particles effect like hazing, raining or snowing can impede all kinds of applications for example the object-centric labeling as shown in [7]. Among those atmospheric phenomenons, snowing is the hardest to solve and affects accuracy the most because of its transparent nature, uneven density and irregular shapes. Such identities of snow particles make hand-craft snow removal tasks intractable. The usage deep-learning method in the domain of weather particle removal has been proposed, such as [1] and [3]. However, such methods failed to accommodate the particular features of snow particles. As from the claim of [7], this is the first data-driven method proposed to specifically solve snow removal problems. The method deals with the complicated translucent and opaque snow particles. They use separate underlying descriptors to estimate and restore details lost to opaque snow particles coverage and model snow translucency using a snow mask and independent chromatic aberration map caused by color distortion. However, such methods including their subsequent improvement all reply on the snow-particle representation defined by a linear equation. And at the same time ignoring effects like veiling [2]. We choose to leverage the ability of neural network and use its ability to model arbitrary functions. We model the rule of adding snow particles using a DCNN-type network and using the idea from CycleGAN that by ensuring cycle-consistency in both ways [12], we should learn the rule to add or remove snow particles at the same time. In this way, we don't need to explicitly model snow-particles but achieve better results. Meantime, our method won't require human-generated fake snow images. Therefore, making sure the model won't inaccurately deal with real-world data. In conclusion section, we admit that our method quality will be highly dependent on training data which will be the problem we seek to solve in future works.

The content of the paper will be organized in the following way. We will first review some important building blocks for DesnowNet in related work section. We will also review CycleGAN [12] which our idea of improvement will be based on. In Method section, we will describe the full detail of the [7]. We will briefly show the improvement ideas for [5] in Method section as well. And finally, we will present our idea of improvement in method section. We will show result of our implementation of DesnowNet comparing with our implementation of CycleSnowGAN in result section in the end.

2. Related Works

2.1. Inception-V4

The inception module was first proposed in Inception-v1 / GoogLeNet. Inputs go through 1×1, 3×3 and 5×5 convolution layers and max pooling layers at the same time and being concatenated together as output. Such a structure ease the chosing of filter sizes and preserves different spatial features. Batch normalization was the improvement from Inception-v2. ReLU activation function was used to deal with the vanishing gradient and saturation problem caused by DCNN. The 5x5 convolution layer was also replaced with two 3x3 layers which preserves reception field size but shrinks parameter sizes. The 1x7 following 7x1 convolution

layer was introduced in Inception-v3 and stay unchanged in inception-v4 as it preserves accuracy but reduces parameter sizes to deal with the overfitting problem. Inception-v4 combines ideas from its predecessors and introduces more inception blocks to learn spatial features at multi-scale receptive fields.

2.2. Atrous Pooling

Repeated convolution blocks and pooling layers s reduces significantly the spatial resolution of the resulting feature maps. To maintain obtain arbitrarily large field-of-view without learning extra-parameters(de-convolution), Atrous spatial pyramid pooling(ASPP) is proposed. It introduces filters with value at different spatial locations and filled with 0 at the wholes between those spatial locations. Such a convolution can easily be implemented using a dilation convolution fashion. With different sizes(levels) of field-of-view being extracted, a pooling of either max pooling or summing up the features can be used to extract spatial features considering different levels.

2.3. CycleGAN

The idea of CycleGAN is basically: 1. We train two different generators to transform from style 1 to style 2 and vice versa. 2. We train the generator such that the generated sample of style 2 are indistinguishable from real images by a discriminator. 3. We make sure the generators are cycle-consistent such that mapping from style 1 to 2 and back again will reconstruct the original image. In method section, we will show our idea of adopting this network to help with snow removal.

3. Method

In this section, we first show detailed review of DesnowNet. We then show later improvement by the same group using GAN [5]. Finally, we show our idea of improvement which taking advantage of CycleGAN that implicitly learn the rule of adding and removing snow particles.

3.1. DesnowNet

3.1.1 Pre-defined snow image representation

Suppose we have an image $\boldsymbol{x} \in [0,1]^{p\times q\times 3}$ with snow particles where p and q are the size of the snow image. We call the snow-free image $\boldsymbol{y} \in [0,1]^{p\times q\times 3}$. If we define a snow mask $\boldsymbol{z} \in [0,1]^{p\times q\times 1}$ and a aberration map $\boldsymbol{a} \in \mathbb{R}^{p\times q\times 3}$, then the relationship between x and y can be presented as:

$$\boldsymbol{x} = \boldsymbol{a} \odot \boldsymbol{z} + \boldsymbol{y} \odot (1 - \boldsymbol{z}) \tag{1}$$

where $\odot$ denotes element-wise multiplication, $\boldsymbol{z}$ shows translucency of snow particles and $\boldsymbol{a}$ shows the color aberration correction should be applied.

The paper split the recovery of $\boldsymbol{y}$ into two parts. First is to recover the aberration and translucency caused by snow particles, which is called translucency recovery module(TR). The recovered image for this part is denoted as $\boldsymbol{y'}$. Second is to generate the residual parts of the snow-free image that get completely blocked by snow particles, which is called residual generation module(RG). The recovered image for this part is denoted as $\boldsymbol{r} \in \mathbb{R}^{p\times q\times 3}$. And the final recovered snow-free image is formed as:

$$\hat{\boldsymbol{y}} = \boldsymbol{y'} + \boldsymbol{r} \tag{2}$$

3.1.2 Network structure and variable definition

To accurately model such features, the paper proposed a network as described in 1. We now list out important for-

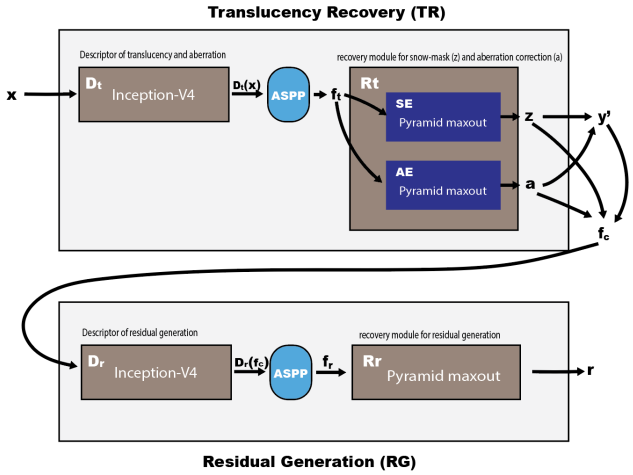


Figure 1. Here, D_t and D_r are the descriptors to extract features from input. D_t takes the image as input and D_r takes the image being applied snowmask and aberration correction as input. R_t and R_r are the recovery block. R_t consists of a SE block using pyramid pooling to reconstruct snow mask $\hat{\boldsymbol{z}}$ and AE block using the same structure to reconstruct aberration correction $\boldsymbol{a}$. And finally R_r block is used to recover the residual $\boldsymbol{r}$ obscured by snow particles.

mulas used to represent each variable used in the network. Denoting $D_t(\boldsymbol{x})$ and $D_r(\boldsymbol{x})$ as the output of inception-v4 networks. The ASPP block is constructed by

$$\begin{aligned} \boldsymbol{f}_t &= \|_{n=0}^{\gamma} A_{2^n}(D_t(\boldsymbol{x})) \\ \boldsymbol{f}_r &= \|_{n=0}^{\gamma} A_{2^n}(D_r(\boldsymbol{f_c})) \end{aligned} \tag{3}$$

Here A_{2^n} is the Atrous convolution layer. $\boldsymbol{f_c}$ will be defined just a moment. $\boldsymbol{f}_t$ and $\boldsymbol{f}_r$ are the feature output of ASPP for TR and RG respectively. $\|$ operator stands for the concatenation operation. Here, γ is the parameter controlling the level or field-of-view of the ASPP. We choose $\gamma = 4$ to align with the [7]. The SE and AE block uses a *pyramid*

maxout structure to select different spatial features which is defined as

$$\begin{aligned} \boldsymbol{a} &= AE(\boldsymbol{f}_t, \beta) \\ &= max(conv_1(\boldsymbol{f}_t), conv_3(\boldsymbol{f}_t) \dots conv_{2\beta-1}(\boldsymbol{f}_t)) \\ \hat{\boldsymbol{z}} &= SE(\boldsymbol{f}_t, \beta) \\ &= max(conv_1(\boldsymbol{f}_t), conv_3(\boldsymbol{f}_t) \dots conv_{2\beta-1}(\boldsymbol{f}_t)) \end{aligned} \tag{4}$$

Here, subscript of $conv$ denotes the kernel size of the operation. We choose $\beta = 4$ to align with the [7]. And $conv_5$ and $conv_7$ are implemented using the vector version as in inception-v4. And $\boldsymbol{y'}$ is constructed as

$$\boldsymbol{y'} = \boldsymbol{m} \odot \boldsymbol{x} + \neg \boldsymbol{m} \odot \frac{(\boldsymbol{x} - \boldsymbol{a} \odot \hat{\boldsymbol{z}})}{\boldsymbol{1} - \neg \boldsymbol{m} \odot \hat{\boldsymbol{z}}} \tag{5}$$

This is just a reformation of (1) and mask $\boldsymbol{m}$ is introduced to prevent dividing by 0 where $m_i = 1$ when $\hat{z}_i = 1$ and $\neg \boldsymbol{m}$ is the logical negation of mask $\boldsymbol{m}$. $\boldsymbol{1}$ is of same size of $\hat{\boldsymbol{z}}$ with entries to be 1. And finally, for residual generation, we first define $\boldsymbol{f_c}$ used in (3) to be $\boldsymbol{y'}\hat{\boldsymbol{z}}\boldsymbol{a}$ which is the recovered image after being applied snow-mask and aberration correction. The operations then are defined as:

$$\begin{aligned} \boldsymbol{r} &= R_r(\|_{n=0}^{\gamma} A_{2^n}(D_r(\boldsymbol{f_c}))) \\ &= R_r(\boldsymbol{f_r}) \\ &= \sum_{i=0}^{\beta} conv_{2i-1}(\boldsymbol{f_r}) \end{aligned} \tag{6}$$

Here, instead of using the *pyramid maxout*, summation is used here to aggregates the multi-scale features to model the variation in snow particles.

3.1.3 Loss Function

We now define the loss function. As suggested from [6] that pixel-wised MSE loss can't simulation of various viewing distances pertinent to human vision. Therefore, a simple pyramid maxing structure is used in the loss function:

$$\mathcal{N}(\boldsymbol{m}, \hat{\boldsymbol{m}}) = \sum_{i=0}^{\tau} \|P_{2^i}(\boldsymbol{m}) - P_{2^i}(\hat{\boldsymbol{m}})\|_2^2 \tag{7}$$

Here, P_{2^i} is just a maxpooling layer with kernel size 2^i and this captures the loss from various viewing distances. $\boldsymbol{m}$ or $\hat{\boldsymbol{m}}$ are images with same size. And the total loss is then defined as:

$$\mathcal{L}_{overall} = \mathcal{N}(\boldsymbol{y'}, \boldsymbol{y}) + \mathcal{N}(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda_{\hat{\boldsymbol{z}}} \mathcal{N}(\hat{\boldsymbol{z}}, \boldsymbol{z}) + \alpha \sum \boldsymbol{w}^2 \tag{8}$$

Here, $\lambda_{\hat{\boldsymbol{z}}} = 3$ and $\alpha = 5e^{-4}$. $\sum \boldsymbol{w}^2$ is the L2 regularization.

3.2. DesnowGAN

In this section, we will show more recent paper [5] that utilize the GAN structure but use similar way of representing snow particles. The snow image is still represented as (1). As the paper points out, the main problem with DesnowNet is its super large hyperparameter size introduced by inception-v4. Therefore, the fundamental building block is substituted by ResNeXt [10]. Meantime, they point out D_r block which in DesnowNet uses the full capacity of a inception-v4 network is also a waste of parameter. Therefore, a refinement block is used to substitude the original D_r block and the $\boldsymbol{r}$ prediction is now combined with $\boldsymbol{z}$ prediction and jointly get optimized. Also, instead of preserving the input image size in DesnowNet, a deconvolution layer is used to recover the original image size. And finally, a discriminator loss is applied combined with the Loss defined in (8). The discriminator loss is defined as :

$$\begin{aligned} \mathcal{L}_{\text{critic}} &= \mathbb{E}_{\hat{\boldsymbol{y}} \sim \mathbb{P}_G}[D(\hat{\boldsymbol{y}})] - \mathbb{E}_{\boldsymbol{y} \sim \mathbb{P}_r}[D(\boldsymbol{y})], \\ \mathcal{L}_{gp} &= \mathbb{E}_{\tilde{\boldsymbol{y}} \sim \mathbb{P}_{\tilde{\boldsymbol{y}}}} \left[\left(\|\nabla_{\tilde{\boldsymbol{y}}} D(\tilde{\boldsymbol{y}})\|_2 - 1 \right)^2 \right], \\ \mathcal{L}_{\text{Discriminator}} &= \mathcal{L}_{\text{critic}} + \lambda_{gp} \mathcal{L}_{gp}, \end{aligned} \tag{9}$$

Here, $\mathcal{L}_{\text{critic}}$ is the WGAN-GP GAN loss that the probability predicted by the discriminator say $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ being actual data should be close. $\mathcal{L}_{gp}$ this the term enforces 1-Lipschitz smoothness.

3.3. Improvement: CycleSnowGAN

We see snow particles are represented in a different way than what DesnowNet represents in JSTASR-DesnowNet [2]. However, such a way of modeling snow particles should be able to be learned as an implicit rule that is similar to a style-transfer. We adopt the idea from DesnowNet that spatially varying features should be extracted using ASPP. We use ResNet [4] or UNet [8] with skip connection as the building block for both generator and use a DCNN as the discriminator. This structure is typically used in CycleGAN and we added in an extra module of ASPP and convolution layer to extract spatial features. We prefer using UNet with skip connection as it captures information from different resolution and we add a ASPP to each different resolution as in [5] to capture varies snow particle sizes.

We also see in DesnowNet [7], they use approximated snow particles lying on top of real images to create effects like snowing. However, such effect seems unreal from human perspective and we can expect very different picture in snow days. We show some example below what the unreal images looks like in 2.

As networks are more capable of interpolation than extrapolation, such generated images used in training may cause problem of extrapolation required by testing in real-world. Therefore, we propose to use CycleGAN as the

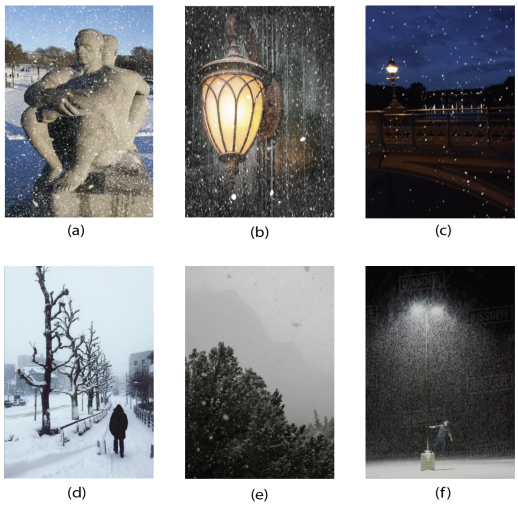


Figure 2. first row images are generated snow images and second row iamges are real world ones. We see comparing (a) and (d), no snow will happen in sunny day which means snow removal won't happen for a sunny sky in real-world. Comparing (b) and (c), we see large patch of losing image information is rare and real-world snow particle tend to be more gray because of the foggy or white weather. Comparing (c) and (f) we see, particles won't be visible in unlighted areas like what (c) has.

building block for our method. The reason is straight forward that CycleGAN won't require paired up data like what Pix2pix or DesnowGAN [5] requires. Therefore, real-world images can be used to make sure our trained network fits to the real world image distribution instead of the fake generated image space.

Our network takes the structure described 3. The loss we take is in same definition as CycleGAN, which includes the GAN loss for the two discriminator which is defined as:

$$\begin{aligned}\mathcal{L}_{\text{GAN}}\left(G, D_Y, X, Y\right) = &\mathbb{E}_{y \sim p_{\text{data}}(y)}\left[\log D_Y(y)\right] \\ &+ \mathbb{E}_{x \sim p_{\text{data}}(x)}\left[\log\left(1 - D_Y(G(x))\right)\right]\end{aligned} \tag{10}$$

where G is the generator's output and D_Y is the discriminator in domain Y. In our case, Y or X just stands for either clear data or snow data. $\mathcal{L}_{\text{GAN}}\left(F, D_X, Y, X\right)$ is defined in a similar way where F is the other generator. As for reconstruction loss, the pixel-wise difference loss is used.

$$\begin{aligned}\mathcal{L}_{\text{cyc}}\left(G, F\right) = &\mathbb{E}_{x \sim p_{\text{data}}(x)}\left[\|F(G(x)) - x\|_1\right] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}\left[\|G(F(y)) - y\|_1\right]\end{aligned} \tag{11}$$

And the total loss is defined below with λ being hyperparameter controlling reconstruction loss. We use 0.1 in our case:

$$\begin{aligned}\mathcal{L}_{Overall} = &\mathcal{L}_{\text{GAN}}\left(F, D_X, Y, X\right) + \mathcal{L}_{\text{GAN}}\left(G, D_Y, X, Y\right) \\ &+ \lambda \mathcal{L}_{\text{cyc}}\left(G, F\right)\end{aligned} \tag{12}$$

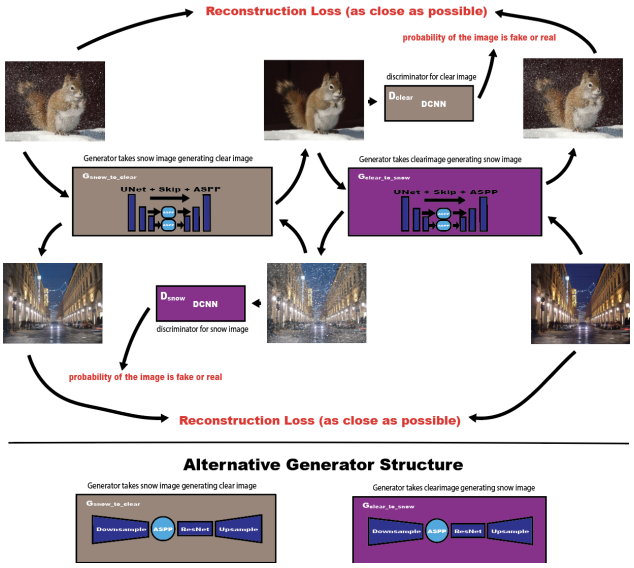


Figure 3. Notice there is only one pair of generators and the reconstruction loss ensures cycle consistency

However, we do seek to further improve the reconstruction loss as the pyramid loss which is used in [7] or as a loss network as proposed in [6].

4. Result

We show our result comparing different implementations of networks. Notice that because of the limitation of our computing power, we shrinked the size of inception-v4 to fit into our GPU. We also put more results in the Appendix A if the reader is interested.

4.1. Snow removal

We show the average (1000 images) PSNR result of CycleSnowGAN comparing with the result from origianl [7] paper in Table 1. Notice in table 1 in [7], the number of large particle masks is significantly less than the other two. Since GAN is highly probabilistic sensitive, we do expect it works better on snow-mask with smaller particles. To address this issue, we can balance out the training data in future works.

We show some visual result which is more straight forward as a comparison in 4.

4.2. Snow generation

We further show the ability for CycleSnowGAN to learn the generation of snow particles implicitly below in 5.

5. Limitation and Discussion

We also argue that with a longer training and tunning time, our implementation of both DesnowNet and Cy-

Network	PSNR
DesnowNet (Paper)	30.1741 [7]
DesnowNet (JSTASR [2])	25.58 [2]
CycleSnowGAN (UNet + ASPP)	27.4979
CycleSnowGAN (ResNet + ASPP)	24.4876

Table 1. Notice, we didn't introduce extra tuning and being much more light-weighted. Notice we also didn't try out the structures mentioned in Future Work section. With carefully tuning and introducing pyramid pooling, ResNeXt and Loss Network, we think our result will be better. Notice DesnowNet's performance varies across different papers, we list two PSNR reported, one form DesnowNet paper, one from JSTASR paper

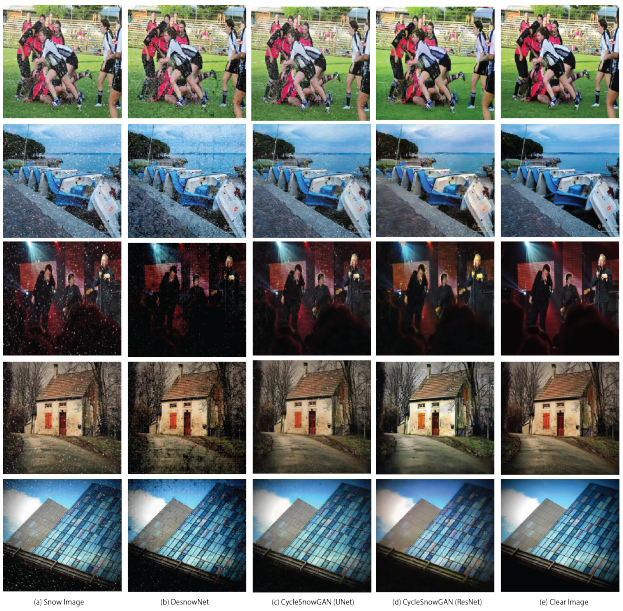


Figure 4. We see PSNR may not be a great comparison tool since slight color-style change may cause huge PSNR change, which may not directly evaluate the task of snow coverage or removal.

cleSnowGAN should perform much better than this result.

We admit that our result highly depends on training set quality. We see artifacts for large snow patches and this is because large particles has smaller dominance compared to smaller ones, especially if we do random cropping. Also, We haven't test our idea on training on real-life dataset because lacking of data. However, we do think this will work on realistic datasets.

6. Conclusion and Future Works

We reviewed DesnowNet [7] which currently has no model or benchmark being implemented. Due to some ambiguity in implementation detail in [7] and constraints in computational resources, the result of our training is not comparable with the original paper. However, due to these constraints, we refrain from questioning the result of DesnowNet.

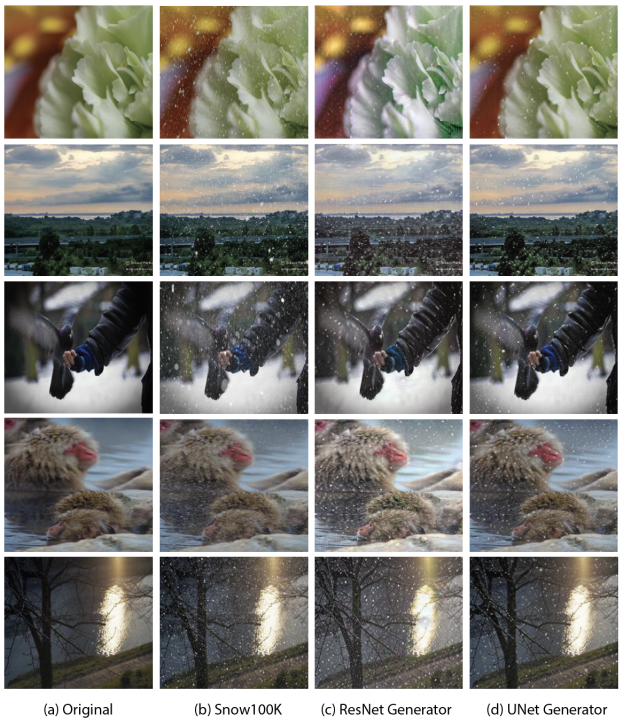


Figure 5. We see variation of snow particle sizes and translucency being generated correctly by the CycleSnowGAN.

We also proposed another CycleSnowGAN which is more light-weighted and provides implementation based on released CycleGAN pytorch implementation. We showed our result for snow removal is comparable with the results showed in DesnowNet paper and we also showed its capability to implicitly encode snow-particle generation rule and generate realistic snow images.

We seek to further improve this method for training on real-life snow images as we don't require paired data and snow-mask. We also seek to further improve CycleGAN encoder and decoder structure and loss function to work more suitably for snow removal. For example, ResNeXt [11] block, Pyramid Pooling Layers, Loss Network [6] or UC-TransNet [9] can be used.

References

[1] Bolun Cai, Xiangmin Xu, Kui Jia, Chunmei Qing, and Dacheng Tao. Dehazenet: An end-to-end system for single image haze removal. *IEEE Transactions on Image Processing*, 25(11):5187–5198, Nov 2016. 1

[2] Wei-Ting Chen, Hao-Yu Fang, Jian-Jiun Ding, Chen-Che Tsai, and Sy-Yen Kuo. Jstasr: Joint size and transparency-aware snow removal algorithm based on modified partial

convolution and veiling effect removal. In *European Conference on Computer Vision*, 2020. 1, 3, 5

[3] Xueyang Fu, Jiabin Huang, Xinghao Ding, Yinghao Liao, and John Paisley. Clearing the skies: A deep network architecture for single-image rain removal. *IEEE Transactions on Image Processing*, 26(6):2944–2956, Jun 2017. 1

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 3

[5] Da-Wei Jaw, Shih-Chia Huang, and Sy-Yen Kuo. Desnowgan: An efficient single image snow removal framework using cross-resolution lateral connection and gans. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(4):1342–1350, 2021. 1, 2, 3, 4

[6] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016. 3, 4, 5

[7] Yun-Fu Liu, Da-Wei Jaw, Shih-Chia Huang, and Jenq-Neng Hwang. Desnownet: Context-aware deep network for snow removal. *CoRR*, abs/1708.04512, 2017. 1, 2, 3, 4, 5

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 3

[9] Haonan Wang, Peng Cao, Jiaqi Wang, and Osmar R. Zaiane. Uctransnet: Rethinking the skip connections in u-net from a channel-wise perspective with transformer, 2022. 5

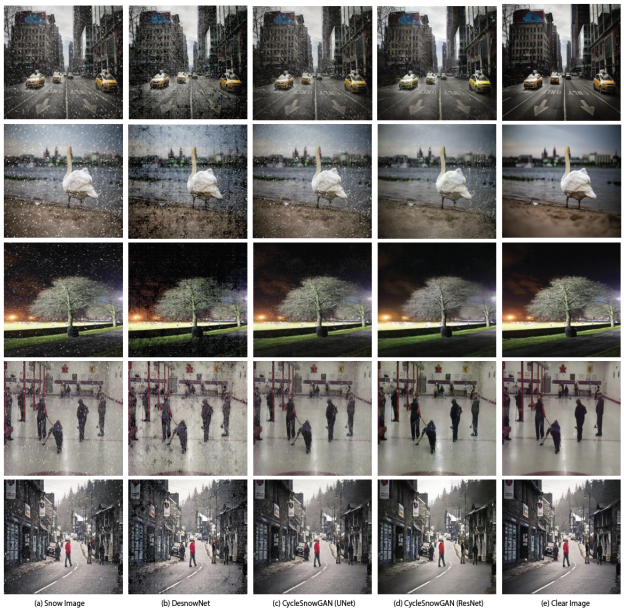
[10] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017. 3

[11] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017. 5

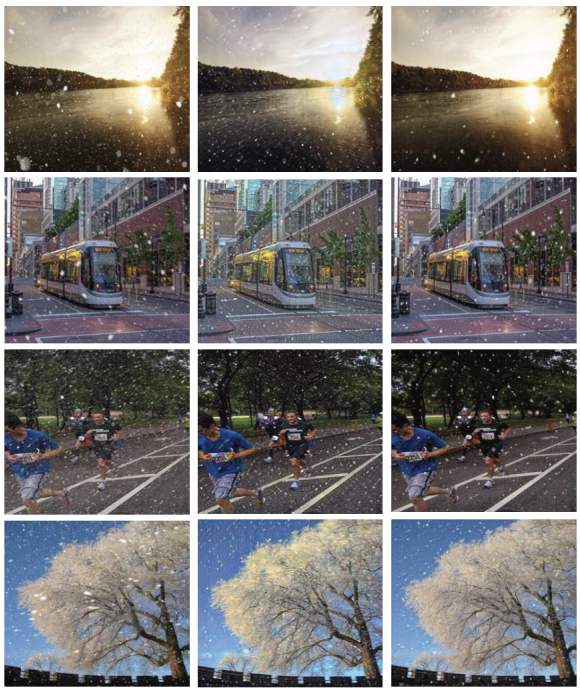
[12] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 1

A. Appendix: More results for CycleSnowGAN and DesnowNet

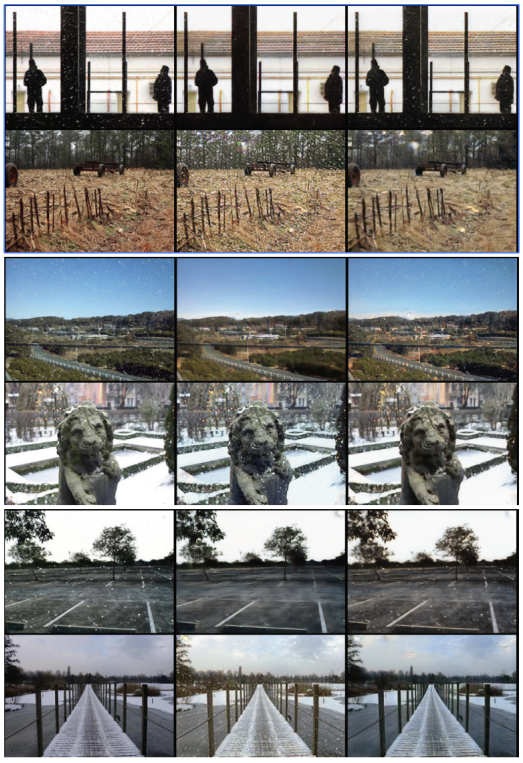
A.1. snow removal



A.2. snow generation



A.3. cycle style



row one:	snow image	generated clear image	reconstructed snow image
row two:	clear image	generated snow image	reconstructed clear image