Multi-scale capsule generative adversarial network for snow removal

Fei Yang | Jialu Zhang | Qian Zhang

1International Doctoral Innovation Centre, University of Nottingham Ningbo China, Ningbo, China
2School of Computer Science, University of Nottingham Ningbo China, Ningbo, China

Correspondence
Qian Zhang, School of Computer Science, University of Nottingham Ningbo China, Ningbo, 315100, China. Email: qian.zhang@nottingham.edu.cn

Abstract
Snowflakes captured on photos may severely decrease the visual quality and cause difficulties for vision analysis systems. Most noise removal frameworks are designed for de-raining or de-hazing, regarding rain or haze as translucent masks on clean images. However, snowflakes are different from them in terms of sizes, shapes, transparency, and floating trajectories, which decreases the performance of de-raining or de-hazing models in processing snowy images. In this work, we propose an effective multi-scale generative adversarial network framework for single-image snow removal, which is built with a multi-scale structure to identify various scales of snowflakes and a capsule-based structure to fuse the features extracted from the multi-scale encoding branches, so that different scaled features could be summarised and learnt by a joint framework. The overall framework is supervised by a weighted joint loss with an iterative training procedure to keep the training stability for the multi-branch-based structure. The experimental results demonstrate that our model outperforms the state-of-the-art comparisons.

1 | INTRODUCTION

Vision tasks in surveillance systems are vulnerable to weather conditions, such as rain, hail, and snow, whose atmospheric particles may impede the normal interpretation, resulting in a disaster possibly. An experiment supported by Cascade-DilatedNet [1] illustrates this in Figure 1. Due to the snowstorm, the Cascade-DilatedNet fails to label the contents correctly or segment the objects clearly.

To counter this effect, various rain and snow removal techniques have been proposed to obtain clear images [2–8]. In the early stages of this research, prior-based approaches dominated. The atmospheric particles are detected and removed by well-designed handcrafted features, such as edge orientations [7,9,10], shapes [11] and streak patterns [3]. However, these methods not only heavily depend on the researcher’s experience, but also greatly limit the generality capabilities of the models. With the development of deep learning, learning-based techniques [8,12–14] have become the focus of research due to their greater efficiency and better generalisation capabilities.

Some researchers regard snowflakes as one special kind of noise and apply common de-noising models for de-snowing. However, snowflakes have more complex characteristics, such as various morphological structures, irregular trajectories, diversified distribution and non-uniform transparencies, which make de-snowing more difficult than other noise removal tasks. It is necessary to develop specific de-snowing models taking into account the characteristics of the snowflakes.

One snowy image contains various sizes of snowflakes, while handcrafted feature-based models are weak to learn them accurately and show quite poor generalisation abilities. As for deep-learning based frameworks [15], where features are obtained by learning, the performance is improved by the accurate learning of the snow feature. However, the existing approaches still have difficulties in handling snowflakes of various sizes and shapes. Thus, we propose a multi-scale structure so that snowflakes of different sizes can be processed at different scales at the same time for feature learning. Figure 2 shows the entire framework. The overall structure follows the idea of the image-conditioned generative adversarial network (image-cGAN), in which the de-snowing is achieved by generating completely snow-free images from snowy images, which can better restore the details of blocked parts. Furthermore, with the help of a capsule, a powerful structure was proposed by...
Hinton et al. [16] which enables the feature learning process to be carried out under the comparison with the global image content, and our framework can detect and remove snowflakes of diverse shapes more accurately and efficiently.

Learning-based models, especially convolutional neural network based ones, have shown a great power on particle removal tasks. But the performance of these models are heavily dependent on the quality of prepared training datasets. For the snow removal task, Liu et al. constructed a snow dataset named Snow100K [15] by adding synthesised snow masks onto clean images using Photoshop [17]. But they did not examine the image content carefully when synthesising the snowy images, resulting in some inappropriate training samples. It is improper to add snowflakes to some snow-meaningless pictures, such as indoor or underwater images. Having these inappropriate samples in training may lead to introducing unreasonable artefacts to the outputs when dealing with the real-world snowy images. Thus, we synthesise another large-scale snowy-image dataset, SnowySet, where meaningless samples are removed. Concurrently, a real-world snowy image dataset is collected for evaluation as well. All the details of SnowySet are presented in Section 4.2.

Our main contributions are summarised as follows:

- We propose a multi-scale image conditional generative adversarial network (GAN) framework for single-image de-snowing, within which three-scaled branches are designed to...
effectively remove various snowflakes. The experimental results demonstrate that our framework outperforms the state-of-the-art methods.

- We are the first to apply capsule units to combine multi-scale branches in the encoder for image generation, which is proved to be effective for learning snow features.
- To improve the stability and efficiency of the training process, we propose a selective training method, where a pre-check mechanism is applied to the discriminative loss to avoid unstable loss signals.
- We construct a dataset, SnowySet, by adding snowflakes to snow-free images. The snowflakes vary greatly in shapes, sizes, transparencies and densities. To make the dataset more reliable and meaningful, indoor, underwater and other images that are inappropriate for being snowy have been removed.

The rest of this article is organised as follows: In Section 2, the related work on de-raining, de-snowing and other noise removal techniques are introduced; The details of our model and training methods are described in Section 3; The datasets and experiments are presented in Section 4, while the results of an evaluation of these are discussed in this section as well. Finally, the conclusions are summarised in Section 5.

2 | RELATED WORK

Besides the snow removal, we introduce the rain removal and some other related literature as well in this section.

2.1 | Rain removal

There are two types of rain removal, video-based and image-based. The early research of rain removal was done in videos benefiting by the temporal information between the adjacent frames [18-20]. The image-based de-raining is harder as some image content is covered by the rain completely. In recent years, researchers have moved the attention to the image-based de-snowing.

Before applying deep learning for snow removal, researchers tried to develop handcrafted features to identify snowflakes. Kang et al. [10] proposed a single image de-raining framework firstly. The rainy images were decomposed into the several frequency components to separate the rain streaks against the image content. Chen et al. [9] removed the rain by extracting a hybrid feature set from the rainy images. In spite of the decomposition methods, the sparse coding approach attracted researchers’ attention. Sun et al. [21] developed an incremental dictionary learning strategy and Luo et al. [7] proposed a discriminative approach. Son et al. [11] designed a shrinkage-based algorithm to learn the sparse representation. However, the drawbacks of the sparse coding methods are obvious as well. They assumed that the rain noise is ‘streaks’-like and has the similar gradient orientation over the whole image [15]. Because of the harsh assumptions, these methods cannot identify various-direction rain streaks and would easily introduce strip-like artefacts. Except the above approaches, there are other methods which are not based on deep learning proposed in different algorithms. Kim et al. [22] made an adaptive non-local filter to separate the rain streaks. Chen et al. [23] proposed a generalized low-rank model to learn the rain feature. The Gaussian mixture models were implemented by Li et al. [3] as a layer-prior de-raining approach. The performance of the conventional generative models is limited and fell short of what deep learning methods achieved.

The rain removal methods based on deep learning have greatly improved the performance and have attracted widespread attention. Yang et al. [12] proposed a multi-branch framework to detect and remove rain streaks in rainy images. Fu et al. [8] constructed the detail layers to separate the rain streaks from the background. To remove the rain streaks with different densities, researchers tried to build complex structures with sub-networks and recurrent modules. Li et al. [14,24] built a recurrent structure to remove rain streaks in multiple stages. Zhang et al. [25] proposed a density aware multi-scale framework to identify the rain streaks with extra density information. More recently, researchers found that image enhancement is helpful to remove the artefacts that are introduced by the de-raining models. Thus, the enhancing layers or blocks are commonly implemented in recent works [25,26].

2.2 | Snow removal

Compared with de-raining, de-snowing does not attract so much attention. Some researchers applied the de-raining models on snow removal directly [4,5,27,28], while ignoring the unique characteristics of snowflakes such as various shapes, opaque and etc. Nevertheless, these special features make snowflakes more complicated than rain drops. In an earlier stage, researchers believed that snow inherits certain rain characteristics and considered it as a special form of rain. Hence, referencing to de-raining models, de-snowing models are designed to remove snow particles with the similar handcrafted features as well. Bossu et al. [20] separated the foreground and background by using Gaussian Mixture Model and constructed the snow features in foreground with the Histogram of Orientations of Snow Streaks. Rajderkar et al. [29] proposed an image decomposition approach based on Morphological component analysis, where the image would be decomposed into low and high frequency (LF/HF) parts by bilateral filters firstly and then applying dictionary learning and sparse coding methods to identify snow components later. Xu et al. [30] modelled snow particles by colour assumptions and removed snow with a guidance image. As this approach may result in losing detailed information of local regions, they improved their framework with a refined guidance image in [31]. However, these prior-based methods can only model limited features and are weak in generalisation. The hyper-parameters of guided filter in [31] may suit low transparency snowflakes, but failed for opaque ones.

Due to the limitation of handcrafted feature-based models, researchers developed learning-based models to learn effective features from existing noisy data distribution. The image translation frameworks have been applied for de-snowing
Now, more and more researchers start to notice the differences between snowflakes and rain streaks, and realize that these differences may significantly affect the removing effect. The frameworks specifically designed for snow removal begin to attract more attention [15,32].

2.3 | Other noise removal/inpainting works

The research of haze/cloud removal has a long history in image processing. He et al. [33] proposed the famous dark channel prior (DCP) method to remove haze from the single hazy image. Li et al. [34] proposed two multi-temporal dictionary learning algorithms to recover the image content from large-scale clouds. Cai et al. [35] proposed the DehazeNet, which firstly introduced a fundamental CNN structure for de-hazing. Later, researchers found that the learning of the image feature benefits from the multi-scale structure, and Ren et al. [36] built a multi-scale network to learn the estimation of the transmission map $t$. More recently, Li et al. [37] developed a non-negative matrix factorization and error correction method (S-NMF-EC) for cloud removal.

Indeed, image noise removal amounts to image inpainting, which aims to restore the missing or damaged regions in an image. Snowflakes and other kinds of noise are regarded as the damaged factors to the image. Bertalmio et al. [38] discussed the problem of image inpainting. Xu et al. [39] introduced an exemplar-based inpainting algorithm through investigating the sparsity of natural image patches and achieved quite good results. Guillenmot et al. [40] presented a brief overview about the research of image inpainting.

Though a great number of works have been proposed for noise removal, we focus on the snow removal in this work. Since, snowflakes are much different with other kinds of noise in terms of the sizes, shapes, densities and etc., we design a specific de-snowing framework targeting the characteristics of snow removal.

3 | METHODOLOGY

3.1 | Overall framework

We build the multi-scale model based on the image-cGAN, which has been successfully applied on image translation [41,42]. The overall framework is shown in Figure 2. The generator (G) is constructed with multiple branches to tackle different sizes of snowflakes, since, the snowflakes vary greatly in sizes, shapes, densities, orientations and trajectories. The discriminating part consists of three separate discriminators (Ds) with each targeting to distinguish one scale of the generated image. Each branch of G takes a certain size of snow image as its input, which is down-sampled from the original snow image. After layers of convolutional filters for each branch, the encoded features are jointly connected to the decoder to synthesise the corresponding sizes of clean images.

Our idea of the multi-scale structure came from the feature pyramid network [43], which enhances feature learning ability by setting pyramid-like multi-scale layers of filters. Thus, we designed the multi-scale structure for both G and D to improve their feature learning ability for various sizes of snowflakes. We are not the first to apply multi-scale receptive fields to noise removal. [25,32] introduced the similar idea by implementing different sizes of convolutional filters to expand the receptive fields for de-raining, where the branch with larger convolutional filters is designed for the feature learning of bigger raindrops. Compared with their approaches, our method obtains better results and the computational burden is reduced, since, we down-scale the input images for the small-scale branches.

In addition, inspired by the research of capsules [42,44,45], we implement capsule units in both G and D to improve the feature representation ability of the overall framework. According to previous research, capsule units are able to learn the part-to-whole relationship [45], and therefore enhance a model’s ability to identify objects on a global view. In our model, the capsule units help to learn the ‘hard samples’ of the snowflakes by checking them against the image background with global information. This implementation benefits both G and D and improve the generating and discriminating abilities.

3.2 | Multi-scale branches

The conventional generative adversarial network with a single network-based G is weak at tackling the de-snowing problem, because of the monotonicity of convolutional kernels and the variety of snowflakes. Our solution enlarges the receptive fields by applying multiple scales of filters, as shown in Figure 2. Taking the model of [25] for reference, we designed a multi-scale structure with three scaled branches, with scales of $512 \times 512$, $256 \times 256$ and $128 \times 128$. The snowy images are resized into the three scales, with each being processed by one branch of G. Each branch includes a PrimaryCaps layer to transform the layer maps into the capsule structure. All the PrimaryCaps capsules of the three branches are densely connected with an FC_Caps layer, which is responsible for merging features from three branches. The capsule structure is described in Section 3.3.1. The FC_Caps layer is connected with three decoding branches, aiming to synthesise three scales of images, $512 \times 512$, $256 \times 256$ and $128 \times 128$. Each branch consists of a DePrimaryCaps layer and several deconvolutional layers. Each deconvolutional layer enlarges the feature maps by a 2-stride filter. The $512 \times 512$ image is the de-snowing result, while the outputs from the branches of the $256 \times 256$ and $128 \times 128$ images are used to help the overall training of the framework.

To distinguish the three scales of synthesised images, we placed three separate discriminators, $512 \times 512$ ($D_{512}$), $256 \times 256$ ($D_{256}$), and $128 \times 128$ ($D_{128}$), each of which is constructed by several convolutional layers and a two-discriminating part (named two-branch D in [42]). The responsibility of each scale discriminator is to distinguish the synthesised de-snowing image from the snow-free images (ground truth) at the corresponding scale. The discriminating
loss is back propagated to G to improve G’s generating ability on every scale. In particular, we down-sample the generated 512 × 512 images into 256 × 256 and 128 × 128 and send them into D_{256} and D_{128} for multi-scale discriminations. In this way, the decoding branch of 512 × 512 could have useful supervision signals from all scales of Ds.

### 3.3 Capsule-based generative adversarial network

Generally, three kinds of capsule layers comprise of the capsule implementation of the framework: the PrimaryCaps layer [45], the FC_Caps layer [42,45] and the DePrimaryCaps layer [42]. The PrimaryCaps and DePrimaryCaps layers are responsible for the transformation between the conventional convolution based layer maps and the capsule layers. The FC_Caps layer learns and stores the features in capsule units. The connections between two capsule layers are weighted by dynamic routing [45], which finds the optimal way to flow the capsule information from the previous layer to the next one. In our multi-branch model, the FC_Caps layer within G is responsible for merging the features learnt by the three scale encoding branches and passing the features to different-scale decoders.

#### 3.3.1 Capsule implemented generator

Shown as in the Figure 2, the capsule based multi-scale branch generators are constructed with conventional convolutional/deconvolutional layers and capsule-based layers. On each scale branch of G, the input image is processed by layers of 2-stride convolutional filters with each layer reducing the feature map size by half. The PrimaryCaps layer applies $M \times N$ conventional convolutional filters to output $M$ feature maps with the capsule dimension of $N$ (the length of each capsule vector is $N$). Three PrimaryCaps layers are placed after the three scale convolutional encoders to transform the scalar neurons into the capsules with the same dimension of $N$. Then, all capsules from these three PrimaryCaps layers are able to be concatenated and fully connected to the FC_Caps layers, a set of capsules, via dynamic routing [45]. The dynamic routing calculates the similarity of each pair of capsules to determine whether their connection should be strengthened or weakened. The DePrimaryCaps layer is a capsule layer with the same size as the PrimaryCaps layer, fully connected to the previous FC_Caps layer. The layer maps are transformed by conventional convolutional layers after the DePrimaryCaps layer [42]. Three DePrimaryCaps layers are placed to transform the FC_Caps layer into three decoding branches. After layers of 2-stride deconvolutional filters, de-snowed images are generated. In addition to the capsule layers, skip connections are placed on every scale. In particular, we down-sample the generated 512 × 512 images into 256 × 256 and 128 × 128 and send them into D_{256} and D_{128} for multi-scale discriminations. In this way, the decoding branch of 512 × 512 could have useful supervision signals from all scales of Ds.

#### 3.3.2 Capsule implemented discriminator

Three scales of discriminators with the same structure are placed to distinguish the three scales of synthesised images. The images are processed by several convolutional layers and then are discriminated by two parts, the capsule part [42] and the conventional patchGAN part [41]. The capsule part consists of a PrimaryCaps layer and an FC_Caps layer followed by two additional capsules as the output. The patchGAN part has three conventional convolutional layers, with the last one as a single channel discriminating map (Every ‘pixel’ of it represents the distinguishing result). The two parts focus on discriminating the images in the global and the local view respectively. A balancing weight is placed to summarise the loss from the two discriminating parts, formulated as

\[
\mathcal{L}_{\text{disc}} = \mathcal{L}_{\text{caps}} + \lambda_1 \mathcal{L}_{\text{patchG}},
\]

where $\lambda_1$ is a balancing factor for losses of the two branches in the discriminator, $\mathcal{L}_{\text{caps}}$ and $\mathcal{L}_{\text{patchG}}$ are represented as

\[
\mathcal{L}_{\text{caps}} = E_{x \sim P_{\text{snowy}}(x), y \sim P_{\text{clean}}(y)} \left[ \mathcal{L}_M(D_{\text{caps}}(x, y), [1, 0]) + \mathcal{L}_M(D_{\text{caps}}(x, G(x)), [0, 1]) \right],
\]

and

\[
\mathcal{L}_{\text{patchG}} = E_{x \sim P_{\text{snowy}}(x), y \sim P_{\text{clean}}(y)} \left[ \log(1 - D_{\text{patchG}}(x, y)) + \log(D_{\text{patchG}}(x, G(x))) \right]
\]

where $P_{\text{snowy}}(x)$ and $P_{\text{clean}}(y)$ represent the distribution of snowy and clean images in the training set, respectively; $G(\cdot)$,

---

**Table 1** The training algorithm of the multi-scale capsule-cGAN

| Input | Hyper-parameter setting |
|-------|------------------------|
|       | Training samples snowy images $x$ and clean images $y$ |

For $i = 1; i < \text{training\_iterations} \text{ do}$

| Forward samples through Gs and Ds; |
| Optimise $D_{128}$ by calculating gradients on Equation (6); |
| Optimise $D_{256}$ by calculating gradients on Equation (5); |
| Optimise $D_{312}$ by calculating gradients on Equation (4); |
| Optimise $G_{128}$ by calculating gradients on Equation (16); |
| Optimise $G_{256}$ by calculating gradients on Equation (15); |
| Optimise $G_{312}$ by calculating gradients on Equation (14). |

---
where Down(·) represents the down-sampling process into size $*$. 

### 3.4 Overall optimization functions

In the training phase, the generator is optimised by a joint loss of pixel loss and Structure SIMilarity Index (SSIM) loss [46], along with the supervision signal from multiple discriminators. The pixel loss is to reduce the overall difference between the synthesised image and the ground truth by calculating the $L_1$ norm distance. The SSIM loss is to minimise the structural error of the generated outputs from the ground truth. Since, SSIM is closer to the visibility of human eyes than other evaluation metrics [46], it is commonly used as a good evaluation method in experiments in comparing the similarity of two images. Based on this, SSIM would be an effective loss function intuitively if we want to enhance the visual quality of the synthesised images. The idea of SSIM loss has also been recently introduced, [14,47]. Different losses contribute to the final loss differently with a series of loss balancing weights.

The supervision signal for the $G_s(·)$ is from the combination of multiple losses, shown in Equation (11).

$$L_{gen} = L_{genGAN} + \lambda_2 L_{pixel} + \lambda_3 L_{SSIM}$$  \hspace{1cm} (11)

Here, $L_{pixel}$ is the pixel loss, which is the $L_1$ distance; $L_{SSIM}$ is the SSIM loss calculated as $1 - SSIM$, shown as

$$L_{pixel} = L_1(G(x), y),$$  \hspace{1cm} (12)

and

$$L_{SSIM} = 1 - SSIM(G(x), y),$$  \hspace{1cm} (13)

where $L_1$ is the $L_1$ distance and SSIM(·) means the calculation of the SSIM score for the two images.

The three branches within the $G$ are optimised in terms of three overall loss functions, shown in Equations (14), (15) and (16).

$$L_{gen_{512}} = L_{genGAN_{512}} + \lambda_2 L_1(G_{512}(x), y) + \lambda_3 (1 - SSIM(G_{512}(x), y)).$$  \hspace{1cm} (14)

$$L_{gen_{256}} = L_{genGAN_{256}} + \lambda_2 L_1(G_{256}(x), y) + \lambda_3 (1 - SSIM(G_{256}(x), y)).$$  \hspace{1cm} (15)
\[ L_{\text{gen}} = L_{\text{genGAN}} + \lambda_2 L_1(G_{128}(x), y) + \lambda_3(1 - \text{SSIM}(G_{128}(x), y)). \]  

(16)

G and Ds are trained iteratively, shown in detail in Table 1. The parameters of the FC_Caps layer within G are updated along with \( G_{312} \). When one branch is being optimised, the parameters of the other branches remain unchanged. Equations (14), (15) and (16) are calculated in the form of Equation (17).

3.5 | Selective training

A typical way to train a GAN is to iteratively update the gradients for the weights/parameters of G and D. It is obvious that both G and D are not trained well during the early iterations. Indeed, the discriminator loss fluctuates over the whole training procedure. When G generates ‘better images’ after several optimization steps, the loss of D increases. While, at this moment, D needs some ‘time’ (more training iterations) to optimise itself and learns to distinguish the generated ‘better image’ and the ground truth.

The discriminating signal passed backward to G is not always effective, especially when D is not trained well with a larger discriminating loss. This gives G the wrong signal, which leads G to learn improper feature representations and produce low-quality images. After trials of training, we found that the loss fluctuation of D has a big probability of causing the training to fail sometimes by encouraging G to produce a black or a meaningless image. To reduce the side effect of these loss fluctuations, one solution is to use extra training signals (\( L_1 \) loss) to guide the learning of G, which is applied in the most recent research [41,42]. Another solution is to set a bigger learning rate for D to speed up the optimization of D, thus shortening the ineffective learning time of G. Both methods can help the training, but do not solve the problem completely. In addition, the second method has an obvious drawback: it causes an instability of D and brings obstacles to D in arriving at the optimum. In the present paper, except for the extra training signals (\( L_1 \) loss and SSIM loss described in Section 3.4), we propose a selective training method to have a pre-check on the loss of D, named selective training. The training signal from D can be applied to G only when D is checked with a low D loss.

The judgement criterion is to check whether the discriminative loss, \( L_{\text{disc}} \), is below a reasonable value. Set an empirical value \( \xi_{\text{dis}} \) as the threshold, and \( L_{\text{genGAN}} \) is back-propagated when \( L_{\text{disc}} \) is lower than \( \xi_{\text{dis}} \). Otherwise, the gradients calculated from \( L_{\text{genGAN}} \) will not be applied. Therefore, Equations (14), (15) and (16) are rewritten in the form of Equation (17).

\[
L_{\text{gen}} = \begin{cases} 
L_{\text{genGAN}} + \lambda_2 L_\text{pixel}, & \text{if } L_{\text{disc}} < \xi_{\text{dis}} \\
\lambda_2 L_\text{pixel} + \lambda_3 L_\text{SSIM}, & \text{if } L_{\text{disc}} \geq \xi_{\text{dis}} 
\end{cases}
\]  

(17)

where the * means the corresponding scale (512, 256 or 128).

4 | EXPERIMENT

4.1 | Experiment setting

The proposed model (MC-DS) is compared with auto-encoder (AE) [48], Pix2pix [41], cycleGAN [49], DID-MDN [25], DesnowNet [15], LSTM-GAN [50] and ComposGAN [51] on two datasets, Snow100K [15] and the constructed SnowySet. Due to the lack of the source code of DesnowNet, LSTM-GAN and ComposGAN, we re-implement their frameworks in tensorflow with their recommended hyper-parameters. The experiments are conducted on both synthetic and real-world snowy images. Peak Signal-to-Noise Ratio (PSNR) and SSIM [46] are used for quantitative evaluation. The normalized mean square error (rmse) and the absolute error (\( L_1 \) norm distance) are also listed. The visual de-snowing samples of both the synthetic and the real-world datasets are represented. To check the effectiveness of the multi-scale structure, an ablation study is set by examining a framework with a single-scale generator constructed.

4.2 | Dataset

4.2.1 | Snow100K

Snow100K [15] contains 100,000 synthesised snowy images, 50,000 for training and 50,000 for testing. The snow masks used for synthesising the snowy images are also provided. According to the density of snowflakes, the testing set is separated into three subsets, Snow100K-S, Snow100K-M and Snow100K-L, representing Small, Medium and Large, respectively. Besides the synthesised images, [15] also provides 1329 realistic snowy images downloaded via the Flickr api.

4.2.2 | SnowySet

To increase the diversity of snowflakes, we propose another dataset, SnowySet, which is synthesised with snowflakes in great variations of the shapes, sizes, density, transparency and floating trajectories. The clean images are selected from BSDS500 [52], UDIc.x2 and Snow100K [15] by removing the images that are meaningless to the snowy weather (e.g. indoor, underwater, close-view or water sports images), 4236 for training and 1040 for testing. The snow masks synthesised in the Photoshop [17] are added onto the clean images to synthesise the snowy images, resulting in 42,360 training samples and 10,400 testing samples. Similar to Snow100K and inspired by [53], we prepare the dataset with three density levels, SnowySet-L (Large), SnowySet-M (Medium) and SnowySet-S (Small), to simulate different densities of snowflakes in reality. Hundred real-world snowy images are collected from the Internet for testing with different image contents and various snowflakes. Comparing with the realistic snowy images provided by Snow100K, the ones in SnowySet are with snowflakes...
of more variations. The datasets and the synthesising instructions will be publicly available.

### 4.3 Implementation details

The images are resized into $584 \times 584$ before being randomly cropped into $512 \times 512$ patches for training. The testing images are resized into $512 \times 512$ before being forwarded through the well-trained generator. The kernel sizes and strides in all convolutional/deconvolutional layers are fixed into $5 \times 5$ and 2. The height and width of feature maps are reduced into half after each convolution layer. The number of the convolution layers for each branch varies according to the inputting scale so that $16 \times 16$ feature maps are obtained before being connected to the PrimaryCaps layer. The filter number (layer channels) of each convolutional layer doubles that of the previous layer, with the first layer channel of 32. The layers of the decoder branches of G (deconvolutional layer) have the same settings corresponding to the encoder branches. According to [42,45], we design the capsule structure with the dimension of eight for each convolutional layer. The num ber of the convolution layers for all capsules and set capsule numbers in PrimaryCaps, FC_Caps, DePrimaryCaps layers of G to 16, 48, and 16, respectively.

The first several convolutional layers of Ds are similar to the encoder branches of G, outputting the feature maps in the size of $32 \times 32$ before being sent into the two-sub-branch discriminating part. The PrimaryCaps and the FC_Caps layers of Ds contain 32 and 16 capsules appended by another two capsules as the output. The patchGAN branch of Ds consists three convolutional layers with the last one outputting a $9 \times 9$ single channel feature map.

The training is conducted with a batch size of eight for 100 epochs where the learning rate is decayed by 0.1 per 20 epochs from a initialisation of 0.0002. The loss balancing weights are $\lambda_1 = 0.1$, $\lambda_2 = 10$, $\lambda_3 = 5$, $\lambda_4 = 0.1$. The experiments are executed in tensorflow on NVIDIA GPU Tesla V100 (32 GB).

### 4.4 Quantitative results

The quantitative results of the two synthetic datasets are shown in Tables 2 and 3. The subsets are evaluated separately. The overall results are calculated on the whole datasets, which are equivalent to the average of the subsets. The boldface values in the tables indicate the best results.

Overall, the proposed MC-DS performs the best and ComposGAN [51] ranks the second. DID-MDN [25] and DesnowNet [15] produce comparable results, which are still better than LSTM-GAN [50]. The comparisons are observed more obviously on the two heavy-snow subsets, Snow100K-L and SnowySet-L. The performance of AE [48] and cycleGAN [49] cannot meet the state-of-the-art, due to the simpleness of AE structure and the lack of paired information for cycleGAN [49]. Pix2pix [41] gives quite comparable results on Medium and Small snow subsets, which shows that an U-net generator with a patchGAN discriminator [41] is capable to handle the lightweight work of de-snowing.

In Table 2, DID-MDN [25] produces quite good SSIM scores on Snow100K-M and Snow100K-S, benefiting from its refinement layers to improve the image quality. But it suffers the degradation for heavy-snow images on Snow100K-L with worse scores than DesnowNet [15] and ComposGAN [51].

### Table 2 Evaluation comparison on Snow100K [15]

| Dataset       | Models          | PSNR   | SSIM   | nrmsse | L1     |
|---------------|-----------------|--------|--------|--------|--------|
| Snow-100K-L   | AE [48]         | 21.50  | 0.7528 | 0.1938 | 0.1773 |
|               | Pix2pix [41]    | 25.83  | 0.8490 | 0.1170 | 0.1534 |
|               | cycleGAN [49]   | 19.21  | 0.6575 | 0.3596 | 0.2876 |
|               | DID-MDN [25]    | 26.34  | 0.8969 | 0.1097 | 0.0897 |
|               | DesnowNet [15]  | 27.17  | 0.8983 | 0.1101 | 0.0398 |
|               | LSTM-GAN [50]   | 26.12  | 0.8616 | 0.1214 | 0.1102 |
|               | ComposGAN [51]  | 29.54  | 0.9021 | 0.1001 | 0.0301 |
|               | MC-DS(ours)     | 28.14  | 0.9032 | 0.0995 | 0.0304 |
| Snow-100K-M   | AE [48]         | 23.38  | 0.8422 | 0.1865 | 0.1298 |
|               | Pix2pix [41]    | 29.29  | 0.9261 | 0.0798 | 0.0243 |
|               | cycleGAN [49]   | 20.30  | 0.7377 | 0.2215 | 0.2654 |
|               | DID-MDN [25]    | 28.79  | 0.9423 | 0.0841 | 0.0285 |
|               | DesnowNet [15]  | 30.87  | 0.9409 | 0.0775 | 0.0210 |
|               | LSTM-GAN [50]   | 30.84  | 0.9384 | 0.0721 | 0.0212 |
|               | ComposGAN [51]  | 31.21  | 0.9431 | 0.0622 | 0.0218 |
|               | MC-DS(ours)     | 31.59  | 0.9444 | 0.0619 | 0.0208 |
| Snow-100K-S   | AE [48]         | 23.60  | 0.8444 | 0.2015 | 0.1472 |
|               | Pix2pix [41]    | 29.90  | 0.9232 | 0.1201 | 0.0785 |
|               | cycleGAN [49]   | 21.40  | 0.7825 | 0.2012 | 0.2043 |
|               | DID-MDN [25]    | 30.15  | 0.9521 | 0.0765 | 0.0265 |
|               | DesnowNet [15]  | 32.33  | 0.95   | 0.0599 | 0.0210 |
|               | LSTM-GAN [50]   | 30.09  | 0.9411 | 0.0914 | 0.0255 |
|               | ComposGAN [51]  | 30.43  | 0.9612 | 0.0641 | 0.0207 |
|               | MC-DS(ours)     | 32.49  | 0.9674 | 0.0557 | 0.0174 |
| Over-all      | AE [48]         | 22.83  | 0.8131 | 0.1939 | 0.1514 |
|               | Pix2pix [41]    | 28.34  | 0.8994 | 0.1056 | 0.0854 |
|               | cycleGAN [49]   | 20.30  | 0.7259 | 0.2607 | 0.2524 |
|               | DID-MDN [25]    | 28.42  | 0.9304 | 0.0901 | 0.0482 |
|               | DesnowNet [15]  | 30.11  | 0.9296 | 0.0825 | 0.0272 |
|               | LSTM-GAN [50]   | 29.01  | 0.9137 | 0.0949 | 0.0523 |
|               | ComposGAN [51]  | 30.39  | 0.9355 | 0.0754 | 0.0242 |
|               | MC-DS(ours)     | 30.74  | 0.9383 | 0.0724 | 0.0229 |

Abbreviations: AE, auto-encoder; GAN, generative adversarial network; L1, absolute error; MDN, normalized mean square error; PSNR, Peak Signal-to-Noise Ratio; SSIM, Structure SIMilarity Index.
ComposGAN [51] gives the best PSNR on Snow100K-L with 29.54, but the SSIM is still lower than ours.

In Table 3, DesnowNet [15] and DID-MDN [25] give comparable results with ComposGAN [51], but still perform worse than ours. SnowySet contains snowflakes of more variations, which cause more difficulties to de-snowing models. DID-MDN [25] with density estimation contains the ability to learn the feature of multi-density snowflakes. But our MC-DS with multi-scale branches shows better ability of learning the variations of snowflakes. On SnowySet-L, LSTM-GAN [50] gives quite low SSIM of 0.8641, showing that the LSTM structure may not help a lot to learn heavy-snow features. The patchGAN-based discriminator encourages Pix2pix [41] to learn more local details and texture features, resulting in a reasonable L1 score on the SnowySet-L.

4.5 Visual comparison results

We check the visual inspection of the de-snowing results in Figures 3–6. We select samples with diverse image contents from both the synthetic and the real-world datasets.

4.5.1 Synthetic images

Figures 3 and 5 show the de-snowing results on synthetic images from the testing datasets of SnowySet and Snow100K [15], respectively. Figure 4 shows the enlarged details of images in Figure 3.

CycleGAN [49] and Pix2pix [41] fail to remove the snowflakes and CycleGAN even changes the global colour into ‘Green’ effects. The fourth columns of the Figures 3 and 5 present better de-snowing results, but some local details are also eliminated. The images are over smoothed by DID-MDN [25], which is quite obvious in the first image of Figure 4. For a comparison, ComposGAN [51] and MC-DS restore more details of the ‘grass’. The same phenomenon can be seen in the fourth image of Figure 3, and also in the first and the second images of Figure 5. Compared with DesnowNet [15] and LSTM-GAN [50], the proposed MC-DS removes the snowflakes more completely, which can be examined by checking the ‘sky’ regions of the third sample in Figure 4. ComposGAN’s result is close to ours, but some snowflakes are still left. The ‘sky’ regions from the first sample in Figure 5 presents the similar conclusion. There are snow marks that are not removed on the ‘bear face’ and ‘human face’ from the second and forth images of Figures 3 and 4.

We can see that the proposed MC-DS gives the best de-snowing results close to the ground truth by successfully removing the snowflakes on various kinds of background and recovering most details of the image content.

4.5.2 Real-world images

Figure 6 shows some de-snowing results on real-world snowy images. With the similar phenomenon to that on synthetic images, CycleGAN [49] produces blurry images with ‘Green’ effects. Pix2pix [41] leaves some snowflakes undetected apparently. Compared with DID-MDN [25], DesnowNet [15], LSTM-GAN [50] and ComposGAN [51] from the first and
second images of Figure 6, the proposed MC-DS produces better images by removing more snowflakes and generating a smoother ‘sky’. The down-left corner of the third sample shows ‘two people’ getting on the ‘black car’, which are synthesised blurry by DID-MDN [25], DesnowNet [15] and LSTM-GAN [51]. ComposGAN’s result is much worse. On the forth sample, DID-MDN [25] considers the ‘yellow tie’ on the human in black clothes as one snowflake and removes it totally from the outputting result. For the comparison, our MC-DS recovers it much better. From these samples we can see the superiority of MC-DS in distinguishing the image contents against various snowflakes.

4.6 | Ablation study

To check the effectiveness of the multi-scale structure, we set an ablation study with a single-scale structure by removing the branches of scale_128 and scale_256, marked as Single_scale. The proposed MC-DS is marked as Multi_scale here. Since, the heavy-snow images contain snowflakes of more variations, we conduct the ablation study on Snow100K-L and SnowySet-L. The results are shown in Table 4.

Without the multi-scale structure, the performance of MC-DS-single_scale decreases with all the evaluation metrics on the two subsets. The single-scale model faces

FIGURE 3 The de-snowing results of different frameworks on testing images from SnowySet. Our model produces the best clear ‘sky’ and recovers more details, such as the ‘plant’ and the ‘face’. Larger patches are shown in Figure 4. Best view on screen. GAN, generative adversarial network.

FIGURE 4 The detail comparison of different frameworks on testing images from SnowySet. Best view on screen. GAN, generative adversarial network.
difficulties in processing diverse sizes of snowflakes from different image contents. However, the Single_scale still obtains comparable results with DID-MDN [25] and DesnowNet [15] of Tables 2 and 3, which benefits by the contributions of the SSIM loss and the capsule-based structure.

5 | CONCLUSION

In this work, we build a multi-scale image-cGAN to remove snowflakes from snowy images. Compared with existing de-snowing models, the proposed MC-DS outperforms the state-of-the-art. The experiments demonstrate the
TABLE 4 Ablation study on multi-scale structure

| Dataset           | Models   | PSNR     | SSIM     | nrmse | L1    |
|------------------|----------|----------|----------|-------|-------|
| Snow100K-L       | Multi_scale | 27.54    | 0.9012   | 0.0995 | 0.0304 |
|                  | Single_scale | 26.31    | 0.8910   | 0.1129 | 0.1009 |
| SnowySet-L       | Multi_scale | 27.40    | 0.8974   | 0.0932 | 0.0613 |
|                  | Single_scale | 25.31    | 0.8803   | 0.1143 | 0.0899 |

Abbreviations: L, absolute error; MDN; nrmse, normalized mean square error; PSNR, Peak Signal-to-Noise Ratio; SSIM, Structure SIMilarity Index.

effectiveness of the multi-scale structure in removing various sizes of snowflakes. We implement the capsule layers to fuse the features of different branches and learn the part-to-whole relationship between local regions and the global image content.

Although MC-DS performs well in the experiments, there are more to explore and discuss. Firstly, better methods to connect the capsule block with convolutional/deconvolutional layers can be explored by designing new structures of PrimaryCaps and DePrimaryCaps layers [42,45]. In this work, it is achieved by simply replacing the neurons in different dimensions. Secondly, more effective routing algorithms might be developed for the feature fusion of the three branches if better capsule structures are constructed. DynamicRouting [45] or EMRouting [16] are used for the routing connections between adjacent capsule layers, which might not perform the best for feature fusion. Thirdly, more advanced discriminators could be developed by sharing the features learnt from low-level layers as we place three separate discriminators to distinguish different scales of images in this work. But it should be noted that the training procedure will become more complicated for feature-sharing discriminators than separate discriminators. Fourthly, advanced structures, such as ResNet or Densenet, could be designed with capsule structure implementation.

ACKNOWLEDGEMENTS

The author acknowledges the financial support from the International Doctoral Innovation Centre, Ningbo Education Bureau, Ningbo Science and Technology Bureau, and the University of Nottingham. This work was also supported by the UK Engineering and Physical Sciences Research Council (grant number EP/L015463/1). We are grateful for access to the University of Nottingham Ningbo China High Performance Computing Facility. This work is part of the project of Model and Data Driven Hyper-Heuristics for Combinatorial Optimization and Their Applications in Port Operation Integrated Scheduling, supported by the National Natural Science Foundation of China (Project code: 72071116). It is also a part of the Key Technological Enhancement and Applications for Ningbo Port Terminal Operating System project, supported by the Science and Technology Innovation 2025 Major Project, Ningbo Science and Technology Bureau (Project code: 2019B10026).

REFERENCES

1. Zhou, B., et al.: Springer. ‘Semantic understanding of scenes through the ade20k dataset’. Int. J. Comput. Vis. 127(3), 302–321 (2019)
2. Zhang, H., Sindagi, V., Patel, V.M.: Image de-raining using a conditional generative adversarial network. IEEE Trans. Circ. Syst. Video Technol. 30(11), 3943–3956 (2019)
3. Li, Y., et al.: Rain streak removal using layer priors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2736–2744. IEEE, Las Vegas (2016)
4. Zheng, X., et al.: Single-image-based rain and snow removal using multi-guided filter. In: International Conference on Neural Information Processing, pp. 258–265. Springer, Berlin, Heidelberg (2013)
5. Ding, X., et al.: Single image rain and snow removal via guided 10 smoothing filter. Multimed. Tool. Appl. 75(5), 2607–2712 (2016)
6. Huang, D.A., et al.: Context-aware single image rain removal. In: 2012 IEEE International Conference on Multimedia and Expo, pp. 164–169. IEEE, Melbourne (2012)
7. Luo, Y., Xu, Y., Ji, H.: Removing rain from a single image via discriminative sparse coding. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 3397–3405. IEEE, Santiago (2015)
8. Fu, X., et al.: Removing rain from single images via a deep detail network. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1715–1723. IEEE, Puerto Rico (2017)
9. Chen, D.Y., Chen, C.C., Kang, L.W.: Visual depth guided colour image rain streaks removal using sparse coding. IEEE Trans. Circ. Syst. Video Technol. 24(6), 1430–1445 (2014)
10. Kang, L.W., Lin, C.W., Fu, Y.H.: Automatic single-image-based rain streaks removal via image decomposition. IEEE Trans. Image Process. 21(4), 1742–1755 (2012)
11. Son, C.H., Zhang, X.P.: Rain removal via shrinkage of sparse codes and learnt rain dictionary. In: IEEE International Conference on Multimedia and Expo Workshops (ICMEW), pp. 1–6. IEEE, Seattle, WA, USA (2016)
12. Yang, W., et al.: Deep joint rain detection and removal from a single image. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1357–1366. IEEE, Puerto Rico (2017)
13. Fu, X., et al.: Clearing the skies: a deep network architecture for single-image rain removal. IEEE Trans. Image Process. 26(6), 2944–2956 (2017)
14. Ren, D., et al.: Progressive image deraining networks: a better and simpler baseline. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3937–3946. IEEE, Long Beach (2019)
15. Liu, Y.F., et al.: Desnownet: context-aware deep network for snow removal. IEEE Trans. Image Process. 27(6), 3064–3073 (2018)
16. Hinton, G.E., Saloud, S., Fross, N.: Matrix capsules with em routing. In: International Conference on Learning Representations (ICLR). Vancouver (2018)
17. Adobe photshop. https://www.adobe.com/products/photoshop.html
18. Garg, K., Nayar, S.K.: Detection and removal of rain from videos. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 1–I. IEEE, Washington (2004)
19. Zhang, X., et al.: Rain removal in video by combining temporal and chromatic properties. In: IEEE International Conference on Multimedia and Expo (ICME), pp. 461–464. IEEE, Toronto (2006)
20. Bossu, J., Hautière, N., Tarel, J.P.: Rain or snow detection in image sequences through use of a histogram of orientation of streaks. Int. J. Comput. Vis. 93(3), 348–367 (2011)
21. Sun, S.H., Fan, S.P., Wang, Y.C.: Exploiting image structural similarity for single image rain removal. In: IEEE International Conference on Image Processing (ICIP), pp. 4482–4486. IEEE, Paris (2014)
22. Kim, H.J., et al.: Single-image deraining using an adaptive nonlocal means filter. In: IEEE International Conference on Image Processing (ICIP), pp. 914–917. IEEE, Pairs (2014)

ORCID

Fei Yang https://orcid.org/0000-0002-1510-0675
23. Chen, Y.L., Hsu, C.T.: A generalized low-rank appearance model for spatio-temporally correlated rain streaks. In: IEEE International Conference on Computer Vision (ICCV), pp. 1968–1975. IEEE, Sydney (2013)
24. Li, R., Cheong, I.F., Tan, R.T.: Single image deraining using scale-aware multi-stage recurrent network. arXiv preprint arXiv:171206830, 1, 1712–06830 (2017)
25. Zhang, H., Patel, V.M.: Density-aware single image de-raining using a multi-stream dense network. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 695–704. IEEE, Salt Lake City (2018)
26. Qu, Y., et al.: Enhanced pix2pix dehazing network. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8160–8168. IEEE, Long Beach (2019)
27. Barnum, P.C., Narasimhan, S., Kanade, T.: Analysis of rain and snow in frequency space. Int. J. Comput. Vis. 86 (2–3), 256 (2010)
28. Wang, Y., et al.: A hierarchical approach for rain or snow removing in a single colour image. IEEE Trans. Image Process. 26(8), 3936–3950 (2017)
29. Rajderkar, D., Mohod, P.: Removing snow from an image via image decomposition. In: 2013 IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICECCN), pp. 576–579. IEEE, Tirunelveli (2013)
30. Xu, J., et al.: Removing rain and snow in a single image using guided filter. In: 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE), vol. 2, pp. 304–307. IEEE, Shanghai (2012)
31. Xu, J., et al.: An improved guidance image based method to remove rain and snow in a single image. Comput. Inf. Sci. 5(3), 49 (2012)
32. Li, P., et al.: Stacked dense networks for single-image snow removal. Neurocomputing, 367, 152–163 (2019)
33. He, K., Jian, S., Tang, X.: Single image haze removal using dark channel prior. IEEE transactions on pattern analysis and machine intelligence. 33(12), 2341–2353 (2010)
34. Li, X., et al.: Recovering quantitative remote sensing products contaminated by thick clouds and shadows using multimtemporal dictionary learning. IEEE Trans. Geosci. Rem. Sens. 52(11), 7086–7098 (2014)
35. Cai, B., et al.: Dehazenet: an end-to-end system for single image haze removal. IEEE Trans. Image Process. 25(11), 5187–5198 (2016)
36. Ren, W., et al.: Single image dehazing via multi-scale convolutional neural networks. In: European Conference on Computer Vision, pp. 154–169. Springer, Munich (2016)
37. Li, X., et al.: Cloud removal in remote sensing images using nonnegative matrix factorization and error correction. ISPRS J. Photogrammetry Remote Sens. 148, 103–113 (2019)
38. Bertalmío, M., et al.: Image inpainting. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 417–424. ACM Press, New York (2000)
39. Xu, Z., Sun, J.: Image inpainting by patch propagation using patch sparsity. IEEE Trans. Image Process. 19(5), 1153–1165 (2010)
40. Guillemon, C., LeMeur, O.: Image inpainting: overview and recent advances. IEEE Signal Process. Mag. 31(1), 127–144 (2013)
41. Isola, P., et al.: Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5967–5976. IEEE, Pittsburgh (2017)
42. Yang, F., et al.: Capsule based image synthesis for interior design effect rendering. In: Asian Conference on Computer Vision (ACCV), pp. 183–198. Springer, Perib (2018)
43. Lin, T.Y., et al.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117–2125. IEEE, Honolulu, Hawaii (2017)
44. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: International Conference on Artificial Neural Networks (ICANN), pp. 44–51. Springer, Berlin, Heidelberg (2011)
45. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in Neural Information Processing Systems (NIPS), pp. 3883–3891. MIT Press, Long Beach, CA (2017)
46. Zhou, W., et al.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. 13(4), 600–612 (2004)
47. Wang, T., et al.: Spatial attentive single-image deraining with a high quality real rain dataset. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12270–12279. IEEE, Long Beach (2019)
48. Xie, G.S., Zhang, X.Y., Liu, C.L.: Efficient feature coding based on auto-encoder network for image classification. In: Asian Conference on Computer Vision (ACCV), pp. 628–642. Springer, Singapore (2014)
49. Zhu, J.Y., et al.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE International Conference on Computer Vision (ICCV), pp. 2223–2232. IEEE, Venice (2017)
50. He, L., Zhang, J.: Snowflakes removal for single image based on model pruning and generative adversarial networks. In: 2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC), pp. 172–176. IEEE, Xiamen (2019)
51. Li, Z., et al.: Single image snow removal via composition generative adversarial networks. IEEE Access. 7, 25016–25025 (2019)
52. Arbelaez, P., et al.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 33(5), 898–916 (2011)
53. Yang, F., et al.: Quality classified image analysis with application to face detection and recognition. In: 24th International Conference on Pattern Recognition (ICPR), pp. 2863–2868. IEEE, Beijing (2018)

How to cite this article: Yang F, Zhang J, Zhang Q. Multi-scale capsule generative adversarial network for snow removal. IET Comput. Vis. 2021;1–13. https://doi.org/10.1049/cvi2.12038