TOHOKU UNIVERSITY

Graduate School of Information Sciences

Neural Image Restoration for Images with Diverse Distortion Factors
(ニューラルネットワークによる多様な劣化要因を持つ画像の画像復元)

A dissertation submitted for the degree of Doctor of Philosophy
(Information Science)

Graduate School of Information Sciences

by

Xing Liu

July 5, 2019

Neural Image Restoration for Images with Diverse Distortion Factors

Xing Liu

Abstract

Owing to the recent advancement of deep learning, applications of deep Convolutional Neural Networks (CNNs) have been developed for various industry purposes (*e.g.*, image classification, segmentation, detection, *etc.*). However, evidence that CNNs are vulnerable to image distortions have been witnessed in many recent studies in computer vision. This results in problems in the applications. For example, a self-driving car or drone which uses CNNs-based detection systems, could crash to objects in bad weather or when it moves fast. These are due to the distortions (*e.g.*, raindrops, motion-blur) on the images taken by the on-vehicle camera(s).

In this thesis, we study the problems of restoring clear images from their distorted versions on different factors, aiming at solving the aforementioned problem in real-world applications. we focus on designing effective deep neural networks for solving these problems. In Chapter 2, we propose a novel style of residual connections dubbed “dual residual connection”, which exploits the potential of paired operations, e.g., up- and down-sampling or convolution with large- and small-size kernels. We design a modular block implementing this connection style; it is equipped with two containers to which arbitrary paired operations are inserted. Adopting the “unraveled” view of the residual networks proposed by Veit et al., we point out that a stack of the proposed modular blocks allows the first operation in a block interact with the second operation in any subsequent blocks. Specifying the two operations in each of the stacked blocks, we build a complete network for each individual task of image restoration. We have experimentally evaluated the proposed approach on five image restoration tasks using nine datasets. The results show that the proposed networks with properly chosen paired operations outperform previous methods on almost all of the tasks and datasets.

Recent studies have demonstrated that training a CNN for multiple purposes, *e.g.*, a CNN is

trained on image caption retrieval, visual question answering, and visual grounding for learning vision-language representations, can lead performance improvement on these tasks by the synergetic effects. In Chapter 3, we apply this idea to image restoration tasks, we show that a single network having a single input and multiple output branches can solve multiple image restoration tasks. This is made possible by improving the attention mechanism and an internal structure of the basic blocks of dual residual connection. Experimental results show that the proposed approach achieves a new state-of-the-art performance on motion blur removal, haze removal (both in PSNR/SSIM) and JPEG artifact removal (in SSIM). To the author's knowledge, this is the first report of successful multi-task learning on image restoration tasks, which are diverse in the sense that they appear to be dissimilar on the surface level.

Restoring the clear image from its degraded versions corresponds to retrieving the visually original information of the clear image. This naturally becomes a promising solution to the problem of low classification accuracy on distorted images. Specifically, a CNN trained for image restoration is used as a "data-cleaner" for the CNNs trained for classification. In Chapter 4, we experimentally analyzed this approach on a material recognition task with additive Gaussian noise. The results show that the application of image restoration CNN improves the classification accuracy on noisy images up to humans' level. Based on this founding, we throw an attractive discussion bridging CNNs and human vision for future works.

In Chapter 5, we conduct a deeper discussion between CNNs and humans. It is observed that artificial systems are now rival to humans in several pattern recognition tasks. However, this is only the case with the tasks for which correct answers exist independent of human perception. There is another type of tasks for which what to predict is human perception itself, in which there are often individual differences. Then, there are no longer single "correct" answers to predict, which makes evaluation of artificial systems difficult. In this chapter, focusing on pairwise ranking tasks sensitive to individual differences, we propose an evaluation method. Given a ranking result for multiple item pairs that is generated by an artificial system, the proposed method quantifies the probability that the same ranking result will be generated by humans, and judges if it is distinguishable from human-generated results. Taking as an example a task of ranking image pairs according to material attributes of objects, we demonstrate how the proposed method works.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Images are important information carriers in our daily life. A picture (frame) taken by a on-vehicle camera on a street contains cars, pedestrians, buildings of different size; blue regions for the sky, white line segments for traffic lanes, etc. A picture of a glass mug taken towards a window has images of the mug and the view outside the window projected through the mug. Human's vision system is very effective at making use of images for getting information including those of high level. For example, humans can recognize different objects (*e.g.*, cars, pedestrians, buildings) shown in the street view picture. They can also estimate the distances (depth) between the camera and the objects, the direction a car is moving into and even how crowded is the street by the single picture. For the picture of the glass mug, identifying its material, shape as well as estimating the weight, fragility or transparency of the mug are not difficult for humans. Replicating human's vision system by some forms for applications is the ultimate target for engineers. This has been challenged in various studies. A group of these studies, which aim at replicating human's vision system using computational models, and applying them for solving engineering problems, is named *Computer Vision*.

Machine learning is an application of artificial intelligence (AI) that makes systems to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves. Recently, deep Convolutional Neural Networks (CNNs) which is one type of machine learning algorithms, has been progressively developed for computer vision. One of its most successful instances of the applications to computer vision is image classification, *i.e.*, classifying an image to one of a few given classes. In 2016, a series of CNNs constructed with residual connections [48] achieved humans' level accuracy on classifying 10,000 large scale images to 1,000 classes [108]. In the following year, CNNs [55] consisting of densely connected layers further improved the performance on the same task. Beside this, CNNs have been applied to many other tasks such as object detection and segmentation [78, 43], depth estimation [53] and image generation [41], implying their great potential for solving various problems like humans do.

However, a big gap between human vision system and CNNs developed for computer vision has been found by recent studies as well as in this thesis. That is, CNNs are not robust to image distortion (*e.g.*, noise, blur, geometric transformation, etc.) compared to human's. For example, we train CNNs using clean images and test them using images with additive Gaussian noise. It is observed that the CNNs can classify clean images as accurately as humans do, while the classification accuracy dropped with the noise level more steeply for the CNNs than for humans. Detailed information for the experiment is provided in Chapter 4. A similarly observation is reported in Geirhos *et al.*'s studies [36, 37] which are conducted in a scenario different to ours.

Vulnerability of CNNs to image distortions results in problems in many applications of computer vision. For instance, driving a car in a heavy rain/fog day can cause bad performance of the CNN-based systems relying on on-vehicle cameras. A CNN-based traffic violation detection system can be un-functional during nights due to increased noise level and lack of light. This kind of problems are categorized to the problem of domain-shift. Specifically, the data on which a CNN was trained is different from those on which the CNN is used for inference. Problems related to domain-shift has been studied by various works from different perspectives in the literature. For example, Cohen *et al.*[22] studied dealing with the domain-shift problems caused by geometric transformation. They generalized standard convolutional layers to a new version for improving the model's robustness to translations and rotations, based on the theoretical foundations of symmetry groups. Cohen *et al.*[21] proposed a building block of CNNs to deal with the geometric transformation caused by projecting a spherical image to a plane. On the other hand, studies such as [90, 89, 28, 120] focus on analyzing the domain-shift caused by image intensity changes. M.Dezfooli *et al.*[90] show that CNNs' prediction of an image can be altered by changing the intensities of a small number of pixels. Su *et al.*[120] showed that for some images, one can perturb CNNs' prediction by changing only one pixel of the input image. The recent study by Geirhos *et al.*[37] demonstrates a promising explanation to such observations. They pointed out that CNNs trained in the standard way are biased towards using textures to classify images. They also proposed that encouraging CNNs to learn and use shape-based features over texture-based features to classify images can improve their robustness to a wide range of image distortions to some extent. Sun *et al.*[124] demonstrates that an intensity distortion can shift the histogram of the values of a layer's activation maps, thus results in bad performance of classification. They proposed to use floor / exponential functions to quantify activation values for each layer, aiming at cancelling the shift of histogram. However, the previous methods are either not applicable to multiple types of domain-shift (specifically, image distortions), or not powerful enough to achieve a good improvement of performance on these factors. In this thesis, we pursue more powerful approaches for removing various types of image distortions, or equivalently, restoring the clean images from their distorted versions.

1.2 Theoretical Foundations for Supervised Learning

Supervised learning is a machine learning task of inferring a function from a labeled training dataset. In a supervised learning problem [5], a learning system receives a number of samples (*i.e.*, pairs of an input and its label), and computes a hypothesis function to approximate the true (target) mapping from inputs to labels of the learning problem, by fitting the function to the samples. The true mapping is also referred as the optimal function for the learning system to infer, and the training samples are usually considered as a mount of partial information about the true mapping. Solving a supervised learning problem with a neural network, the task is then posed as adjusting the parameters' states of the network in response to the training dataset [5]. The method by which the parameters are adjusted constitutes a learning algorithm. In another words, a learning algorithm describes what the architecture of a learning system is, and how to adjust its parameters' states. The training data of a learning task is considered as a set of pairs (x, y) where $x \in \mathcal{X}$ is an input (e.g. image) to the network, and $y \in \mathcal{Y}$ is the associated true label. Formally, we define a training data z by

$$z = ((x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)) = (z_1, z_2, \ldots, z_m) \in \mathcal{Z}^m. \tag{1.1}$$

Assume that each pair $z_i = (x_i, y_i)$ where $i \in (0, m]$, $i \in \mathbb{Z}^+$, is generated (*i.e.*, drawn i.i.d.) according to a fix distribution P on $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, that reflects the probability of an input x_i jointly appearing with the specific label y_i, which is also called a *pattern*, the probability of a training data z is defined as the product of the probabilities of all the m pairs

$$P^m(z) = \prod_{i=1}^{m} P(x_i, y_i). \tag{1.2}$$

Next, we define the space of functions a neural network can approximate. In order to do that, we first define a task-specific space S for data, such that $S \subseteq \mathcal{Z}$, representing all the possible inputs and labels related to the task. For example, considering the task of image based binary classification, the task-specific space S consists of all the combinations of the related images and $\{0, 1\}$. At the same time, when considering a regression task such as removing Gaussian noise from a natural image, the task-specific space S becomes combinations of all the Gaussian-noisy versions ($= X \subset \mathcal{Y}$) of natural images and the union ($= Y \subset \mathcal{Y}$) of natural images and their noisy versions, where $X \subset Y$. Additionally, we define a measure D that computes the difference between two objects (*e.g.*, images) by

$$D(v_1, v_2) = \sum_{n=1}^{N} \alpha_n d_n(v_1, v_2), \tag{1.3}$$

where $\alpha_n \in \mathbb{R}$, d_n means a function (*e.g.* Euclidean distance function $||v_1 - v_2||_2$) computing distance between two vectors. Based on the two definitions made above, and considering a scenario of regression task (for image restoration), we define the space of hypothesis functions on S as $H_{|S}$, and the function of error by

$$er(h) = P\left\{(x,y) \in S : D(h(x),y) > \gamma\right\}, \tag{1.4}$$

where $\gamma \in \mathbb{R}$ is a threshold that controls the strictness of judging a system's prediction as "bad", and the error $er(h)$ can be interpreted as the probability for drawing a "bad" (x, y) conditioned on the hypothesis function $h(\cdot)$. The purpose for training a neural network is to compute a hypothesis function making the error the minimum, that is in fact searching for the optimal function $h^\star$ over the hypotheses function space $H_{|S}$. Formally, the minimum error made by the $h^\star$ is written as

$$opt(H) = inf\left\{er(h)\right\}, h{\in}H. \tag{1.5}$$

However, such an minimum (and the values in its close neighborhood) is (are) usually impossible to achieve by a neural network with some learning algorithm, in many real-world problems. Pursuing a practically applicable way for measuring how good a neural network is trained, we employ

$$er(h) < opt(H) + \epsilon \tag{1.6}$$

for the measurement. $\epsilon \in (0, 1)$ is called *accuracy parameter* that represents how much is the error a hypotheses function h made larger than the $opt(H)$ on S. In such a way that we can consider the hypotheses function h computed by a neural network is ϵ-good according to 1.6. Recall that it is mentioned at the beginning that a learning system computes a hypotheses function by fitting it to the training dataset. A training dataset z is generated according to a probabilistic model (1.2), thus it is not guaranteed that the hypothesis function can always be ϵ-good. To handle this problem, equation (1.6) is reformulated within a probabilistic model by

$$P^m\left\{er(h) < opt(H) + \epsilon\right\} \geq 1 - \delta, \tag{1.7}$$

where $\delta \in (0, 1)$ is called *confidence parameter*, and m represent the size of the training data z. The interpretation of equation (1.7) is that, we at least hope to ensure that the hypothesis function h is ϵ-good with a high probability which is specifically at least 1 - δ. Finally, we formally write *learning algorithm* [5] that we have mentioned few times above. A learning algorithm L for $H_{|S}$ is a function

$$L : \bigcup_{m=1}^{\infty} Z^m \to H, \tag{1.8}$$

that maps the set of training datasets of all possible size m (except empty training dataset, *i.e.*, $m = 0$) to $H_{|S}$, with the the property that, for any $\epsilon, \sigma \in (0, 1)$ the following holds:

$$P^m \left\{ er(L(z)) < opt(H) + \epsilon \right\} \geq 1 - \delta. \tag{1.9}$$

It is obvious that the dataset's size m should vary inversely to ϵ, σ and γ (recall (1.4)), reflecting that a larger number of training samples is required for a better performance. An intuitive interpretation to equation (1.9) is that, given a fixed learning algorithm, one could train a neural network to have better performance with a higher probability by adding more training samples. On the other hand, given a set of training samples of fixed size, one could expect better performance with higher a probability by applying a *smarter* learning algorithm. In practice, most related studies as well as my research apply to the later. In this thesis, we focus on making a *smarter* learning algorithm by pursuing effective designs of neural network architecture on image restoration tasks. We propose a basic building block (DuRB) for effectively leveraging the potential of paired operations, and design entire networks for different image restoration tasks with the DuRB. The experimental results show that my networks outperform the state-of-the-art methods in five image restoration tasks and nine benchmark datasets.

1.3 Convolutional Neural Networks

In this section, we introduce i) basic operations used in my research, and ii) the main background on which my proposed network was developed.

1.3.1 Two-Dimensional (2D) Convolution

A 2D-convolution is a single computation of taking the weighted sum of the elements covered by a convolutional filter, which is a matrix holding the weights. A convolutional computation

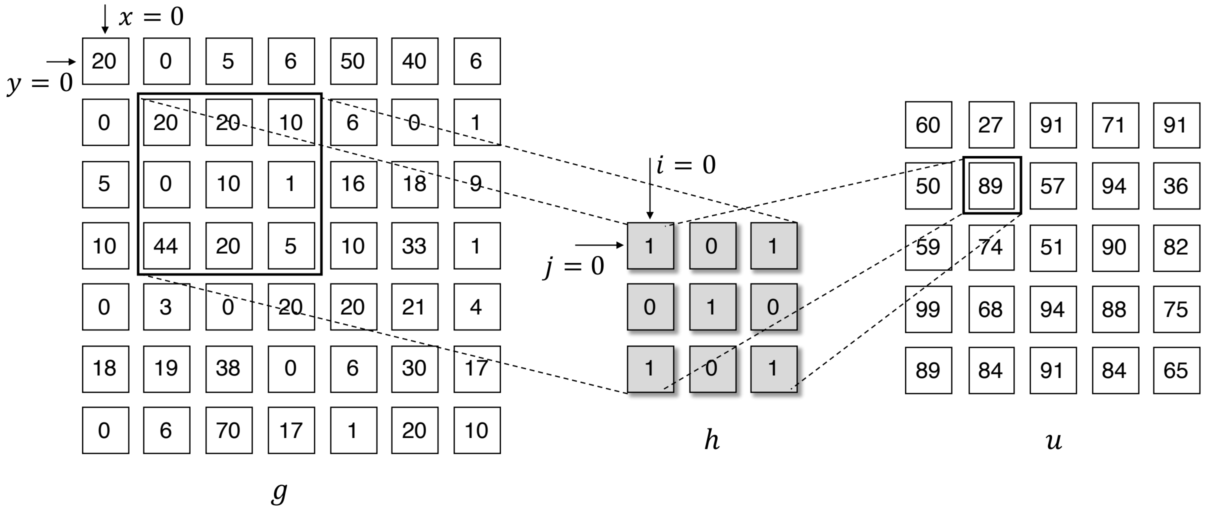


Figure 1-1: A visual example of 2D-convolution. g, h and u means an input array, an 2D-convolutional filter of size = 3×3 and the output, respectively.

outputs a scalar value. Conducting 2D-convolution on a two dimensional array (*e.g.*, a gray-scale image) is equivalent to sliding the convolutional filter over the array with a specified stride. A 2D-convolution with a $I \times J$ filter is formally defined by

$$u_{x,y} = \sum_{i=0}^{I-1}\sum_{j=0}^{J-1} g(x+i, y+j)h(i,j), \tag{1.10}$$

where x and y denote the coordinate of a pixel of the input array. $g(\cdot)$ and $h(\cdot)$ mean the values of the input array and the filter at a coordinate. The convolutional operation is illustrated in Fig .1-1.

In modern CNNs, information is usually stored as tensors having more than two dimensions (specifically, three or four dimensions). The 2D-convolution is thus generalized to be competitive with this setting. Specifically, each convolutional filter becomes a tensor of size $K \times I \times J$, where K is the number of $I \times J$ filters, the same to the channel number of the input. In addition, a bias term b is usually added to a 2D convolutional result in modern CNNs. The formulation 1.10 is thus revised to be

$$u_{x,y} = \sum_{k=0}^{K-1}\sum_{i=0}^{I-1}\sum_{j=0}^{J-1} g(x+i, y+j, k)h(i,j,k) + b. \tag{1.11}$$

In a convolutional layer of a CNN, 2D-convolutional operators are stacked independently of each other, performing the computation defined by 1.11. The number of the operators (*i.e.* the dimension of the convolutional layer) is called *channel number*, usually denoted by C. It is noteworthy that i) the channel number of a layer is equivalent to the dimension of the tensor outputted by the layer, and ii) each channel has an independent bias term b.

1.3.2 Batch Normalization

Batch Normalization [56] is a layer-wise normalization method against to a problem, called *internal covariate shift*, which refers to the fact that the distribution of a layer's input changes, as the parameters of the previous layers change during training a CNN. When the input's distribution shifts around for each iteration or epoch, the parameters of the layer and its later layers do not have a stable ground to stand on to be updated for learning features, which thus results in a complicated training of CNNs and requirement of lower learning rates and careful parameter initialization. Batch Normalization alleviates this problem by controlling the mean and the standard deviation of a layer's input. It first standardizing the layer's input using the mean and standard deviation computed in a mini-batch, then it re-scales and shifts the input by a linear transformation with two parameters (γ and β) learned in the training. This process makes the distribution of a layer's input have a stable mean and standard deviation, against to updating the parameters in the early layers. Formally, Batch Normalization is defined as following. In training, for a layer (*e.g.* 2D-convolutional layer) with a K-dimensional input $x = (x^{(1)}, x^{(2)} \dots x^{(k)})$, Batch Normalization is a function

$$BN_{\gamma,\beta} : x \rightarrow y, \tag{1.12}$$

where $x, y \in \mathbb{R}^{K \times P \times Q}$, P and Q are the height and width of a feature map, and $\gamma, \beta \in \mathbb{R}^K$, such that each element $x^{(k)}$ is first standardized by

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}[x^{(k)}]}{\sigma^{(k)}}, \tag{1.13}$$

then transformed by

$$y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)}. \tag{1.14}$$

$\gamma^{(k)}$ and $\beta^{(k)}$ are two trainable parameters. $\mathbb{E}[x^{(k)}]$ and $\sigma^{(k)}$ mean the expectation and the standard deviation of the activation values in the k^{th} dimension in a mini-batch, which are computed by

$$\mathbb{E}[x^{(k)}] = \frac{1}{MPQ} \sum_{m=1}^{M} \sum_{p=1}^{P} \sum_{q=1}^{Q} x_{m,p,q}^{(k)}, \tag{1.15}$$

and

$$\sigma^{(k)} = \sqrt{\epsilon + \frac{1}{MPQ} \sum_{m=1}^{M} \sum_{p=1}^{P} \sum_{q=1}^{Q} (x_{m,p,q}^{(k)} - \mathbb{E}[x^{(k)}])^2}, \tag{1.16}$$

where M means the number of training images within a mini-batch (*i.e.* batch-size), and $\epsilon \in \mathbb{R}$ is a constant added in the computation for numerical stability.

In the inference time, the $\mathbb{E}[x^{(k)}]$ and $\sigma^{(k)}$ are replaced with the expectation of each, denoted by $\mathbb{E}_{inf}^{(k)}$ and $\sigma_{inf}^{(k)}$, computed over training mini-batches. For the sake of clarity of notions, we denote the input of a layer of a test sample by $x_{te} \in \mathbb{R}^K$, and the value at its k^{th} dimension by $x_{te}^{(k)}$, against to that of a training sample $x^{(k)}$. The computations of $\mathbb{E}_{inf}^{(k)}$ and $\sigma_{inf}^{(k)}$ are originally defined as

$$\mathbb{E}_{inf}^{(k)} = \mathbb{E}_{batch}[\mathbb{E}[x^{(k)}]], \tag{1.17}$$

and

$$\sigma_{inf}^{(k)} = \sqrt{\frac{M}{M-1} \mathbb{E}_{batch}[\sigma^{(k)2}]}, \tag{1.18}$$

where $\mathbb{E}_{batch}[\cdot]$ means the operation of computing expectation over training mini-batches, and the $\frac{M}{M-1}$ is used for the unbiased variance estimate according to the Bessel's correction. It is noteworthy that popular deep learning libraries such as PyTorch [99] or TensorFlow [1] have their own versions of implementation for Batch Normalization, which might be slightly different from the original version introduced above. However, we don't include the related details in this thesis.

1.3.3 Instance Normalization

Instance Normalization [128] is a modified version of Batch Normalization. The main difference is that it computes the $\mathbb{E}[x^{(k)}]$ and $\sigma^{(k)}$ within each feature map of a single sample, and the computation is independent of other samples in a mini-batch. Formally, the $\mathbb{E}[x^{(k)}]$ and $\sigma^{(k)}$ are defined by

$$\mathbb{E}[x^{(k)}] = \frac{1}{PQ}\sum_{p=1}^{P}\sum_{q=1}^{Q} x_{p,q}^{(k)}, \tag{1.19}$$

and

$$\sigma = \sqrt{\epsilon + \frac{1}{PQ}\sum_{p=1}^{P}\sum_{q=1}^{Q}(x_{p,q}^{(k)} - \mathbb{E}[x^{(k)}])^2}. \tag{1.20}$$

It is noteworthy that Instance Normalization uses the same computation for training and inference.

1.3.4 Activation Functions

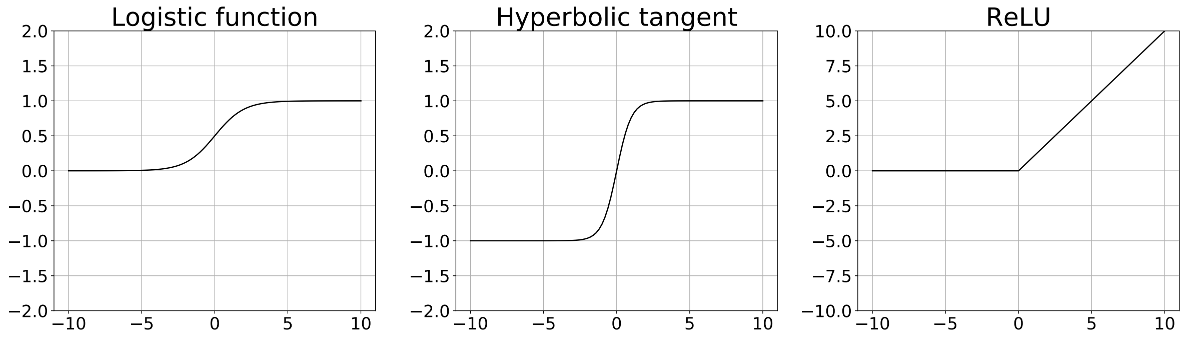


Figure 1-2: Left and middle: Sigmoid functions. Right: Rectified Linear Units (ReLU)

Sigmoid function It refers to the family of functions having “S” curves. Two widely used examples are given here: logistic function and hyperbolic tangent (tanh) function. Logistic function takes values of z in the range of $(-\infty, +\infty)$ of $\mathbb{R}$, and maps them into $(0, 1)$. $tanh$ function takes values of z from the same range as logistic function, and maps them into $(-1, 1)$.

Formally, their are defined by the following equations;

$$\text{Logistic function: } f(\mu) = \frac{1}{1+e^{-z}}, \tag{1.21}$$

and

$$\text{Tanh function: } tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}, \tag{1.22}$$

respectively. Both logistic function and tanh function have two properties, which are i) the output of the function will be saturated when the absolute value of the input z becomes large; and ii) the output value of the function grows smoothly as the input value becomes large. Visual examples of the two functions are given in Figure 1-2, on the left and middle.

Rectified Linear Unit (ReLU) [94] It has become one of the most popular activation functions used for training deep neural network algorithms in the recent years. A ReLU is defined by

$$\text{ReLU}(z) = max(0, z). \tag{1.23}$$

It takes takes values of z in the range of $(-\infty, +\infty)$ of $\mathbb{R}$, and maps them into $[0, \infty)$. It's shape is illustrated on the right hand of Figure 1-2.

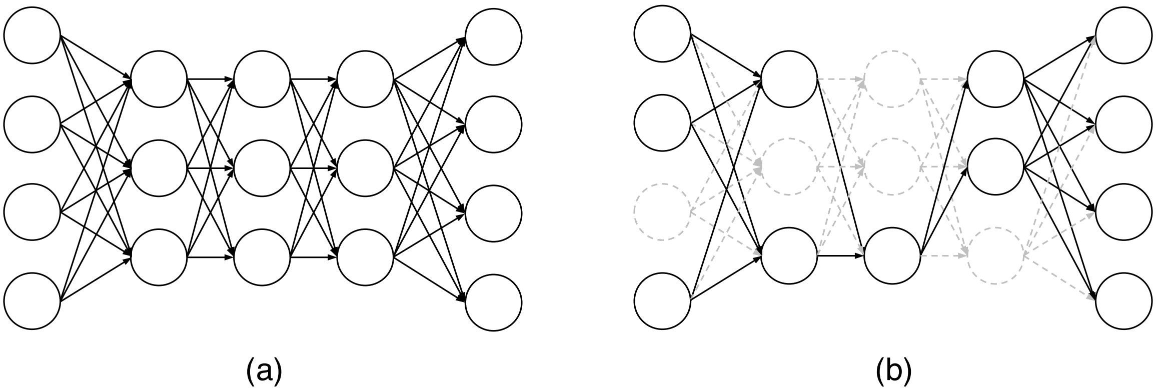


Figure 1-3: (a) is a multi-layers perceptron. (b) illustrates the effects of Dropout on the multi-layer perceptron.

1.3.5 Dropout

Dropout [119] is a method used for resolving *overfitting* problems for deep neural networks. The term “dropout” refers to dropping out units in training a neural network. By dropping a unit out (temporarily removing it) from the network, we mean all its incoming and outgoing connections are removed simultaneously as shown in Fig. 1-3. The sub-figure (a) illustrates a standard multi-layers perceptron which consists of layers of fully-connected (fc) layers. The sub-figure (b) illustrates the effects of Droput on the multi-layers perception. The choice of which units to drop is random. In the simplest case, each unit is retained with a fixed probability p independent of other units, where p is a hyper-parameter defined by the user.

1.3.6 Softmax

Softmax$(x) : \mathbb{R}^T \rightarrow \mathbb{R}^T$, maps a vector of T real numbers into a probability distribution, such that each element in its output is in the interval $(0, 1)$, and the summation over all elements is 1. Formally, it is defined by

$$\text{Softmax}(x)_i = \frac{e^{x_i}}{\sum_{t=1}^{T} e^{x_t}}, \text{ for } i = 1, \ldots, T. \tag{1.24}$$

This function is often used with the last layer of a CNN when the CNN is trained for classification. Assuming such a scenario, the Softmax function takes as input the output vector of the last layer, and outputs a posterior probability for each class, representing the confidence of the CNNs’ decision to the class.

1.3.7 Performance Metrics

Mean Square Error (MSE) It is also known as $L2$ loss, or Quadratic loss, when it is employed for training a learning system for regression. In the scenario of image-to-image transfer tasks, it is formally defined by

$$\text{MSE} \equiv \frac{1}{CWH} \sum_{c=1}^{C} \sum_{x=1}^{W} \sum_{y=1}^{H} (\hat{y}_{c,x,y} - y_{c,x,y})^2, \tag{1.25}$$

where C, W and H are the channel number, width and height of the output image $\hat{y}$ and the ground truth image y.

Mean Absolute Error (MAE) It is known as $L1$ loss, formally defined by

$$\text{MAE} \equiv \frac{1}{CWH} \sum_{c=1}^{C} \sum_{x=1}^{W} \sum_{y=1}^{H} |\hat{y}_{c,x,y} - y_{c,x,y}|. \tag{1.26}$$

It is noteworthy that when C, W and H are all equal to 1, the aforementioned metrics can be used on single real values (*i.e.*, $\hat{y}$ and $y \in \mathbb{R}$).

Structural Similarity (SSIM) Index It is a perceptual metric that measures the similarity between two images. Assuming the scenario of comparing two gray-scale images, one has the perfect quality (*i.e.*, the ground truth image y) while another has a worse quality by some kind of distortion (*e.g.*, the output image $\hat{y}$ by a CNN). The SSIM index models the amount of difference between $\hat{y}$ and y from three perspectives: luminance (l), contrast (c) and structure (s). In practice, the three factors are computed on local patches of the two images. The luminance (l) and contrast (c) of a patch (p) is represented by its mean intensity (μ_p) and the standard deviation (σ_p), computed by

$$\mu_p = \frac{1}{N} \sum_{n=1}^{N} h_n \tag{1.27}$$

and

$$\sigma_p = \sqrt{\frac{1}{N-1} \sum_{n=1}^{N} (h_n - \mu_p)^2}, \tag{1.28}$$

where N is the number of pixels in the patch p, and h_n means the intensity of the n^{th} pixel. The similarity on structure (s) is represented by their covariance ($\sigma_{\hat{p},p}$), which is computed by

$$\sigma_{\hat{p}p} = \frac{1}{N-1} \sum_{n=1}^{N} (\hat{h}_n - \mu_{\hat{p}})(h_n - \mu_p), \tag{1.29}$$

where the notions w/o the superscription "ˆ" , are those computed on the output image (by a CNN) / the ground truth image. Measuring the similarity between two images, it is desirable to make the metric have the following properties:

- Symmetry: SSIM($\hat{p}$, p) $=$ SSIM(p, $\hat{p}$)

- Boundedness: SSIM($\hat{p}$, p) ≤ 1

- Unique maximum: SSIM($\hat{p}$, p) $= 1 \iff y = \hat{y}$

It is made possible by formulating the luminance (l), contrast (c) and structure (s) as

$$l(\hat{p}, p) = \frac{2\mu_{\hat{p}}\mu_p + C_1}{\mu_{\hat{p}}^2 + \mu_p^2 + C1}, \tag{1.30}$$

$$c(\hat{p}, p) = \frac{2\sigma_{\hat{p}}\sigma_p + C2}{\sigma_{\hat{p}}^2 + \sigma_p^2 + C2}, \tag{1.31}$$

and

$$s(\hat{p}, p) = \frac{\sigma_{\hat{p}p} + C_3}{\sigma_{\hat{p}}\sigma_p + C_3}. \tag{1.32}$$

Thus,

$$\begin{aligned} \text{SSIM}(\hat{p}, p) &= l(\hat{p}, p)^{\alpha} \times c(\hat{p}, p)^{\beta} \times s(\hat{p}, p)^{\gamma} \\ &= \frac{(2\mu_{\hat{p}}\mu_p + C_1)(2\sigma_{\hat{p}p} + C_2)}{(\mu_{\hat{p}}^2 + \mu_p^2 + C_1)(\sigma_{\hat{p}}^2 + \sigma_p^2 + C_2)}, \end{aligned} \tag{1.33}$$

when $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$. It is noteworthy that changing the values assigned to α, β and γ adjusts the effect of luminance, contrast and structure on the measured result, and C_1, C_2 and C_3 are constant added in the computation to avoid instability when a denominator close to 0.

Cross-entropy Error Cross-entropy measures the difference between two probability distributions p and q over the same underlying set of events. It is formally defined by

$$\text{Cross-entropy}(p, q) = -\sum_{t=1}^{T} p(x_t)q(x_t). \tag{1.34}$$

This measure is often used as a loss function for training a CNN for classification. Specifically, the ground truth of an image to be classified is converted to a one-hot vector of T elements (classes), and assigned to q. p is the posterior probability predicted by the CNN with a Softmax function. The (1.34) computes the classification loss of the CNN on the input image.

1.3.8 Deep Residual Learning

It is known that a deeper network can achieve better performance than a shallow one. The evidence is with these studies [48, 118, 125, 157, 46]. However, training a very deep neural network is not an easy task. Two notorious problems, *i.e.*, gradients vanishing and exploding, are posed with this issue (discussed in [7, 38]). In the literature, these problems been addressed by various studies such as [38, 111, 47]. Another problem is, with an increasing depth for a network, the performance gets saturated and then degrades rapidly [48]. In addition, it is reported that this problem is not caused by overfitting [48]. Such a problem results in limit on using deeper networks for better performance, thus becomes a considerable obstacle in developing powerful CNNs. Residual Learning is proposed to address this problem. Formally it is defined by

$$y = \mathcal{F}(x; \mathcal{W}) + x, \tag{1.35}$$

where $\mathcal{F}(\cdot; \mathcal{W})$ is some function implemented by a stack of layers (a block) with trainable parameters $\mathcal{W}$, x and y are the input and output to/from the block. Assuming that there exists an optimal projection $\mathcal{H}$ that maps an input x to the optimal output y^*, such that $y^* = \mathcal{H}(x)$. The $\mathcal{F}(\cdot; \mathcal{W})$ is expected to estimate the difference $\mathcal{H}(x) - x$ (the so-called *residual*) based on x, and by this intention, it is defined by

$$\mathcal{F}(x; \mathcal{W}) \equiv \mathcal{H}(x) - x. \tag{1.36}$$

It is thus obvious that adding x to $\mathcal{F}(x; \mathcal{W})$ approaches the optimal output y^* ($= \mathcal{H}(x)$), when the $\mathcal{F}$ is properly designed and trained. Two basic implementations [48] of (1.35) are shown in Fig .1-4. The “1×1” and “3×3” mean the kernel size for a convolutional layer (conv.). A Batch Normalization layer is actually used right after a convolutional layer for all the convolutional layers, and it is not illustrated in the figure for the sake of simplicity.

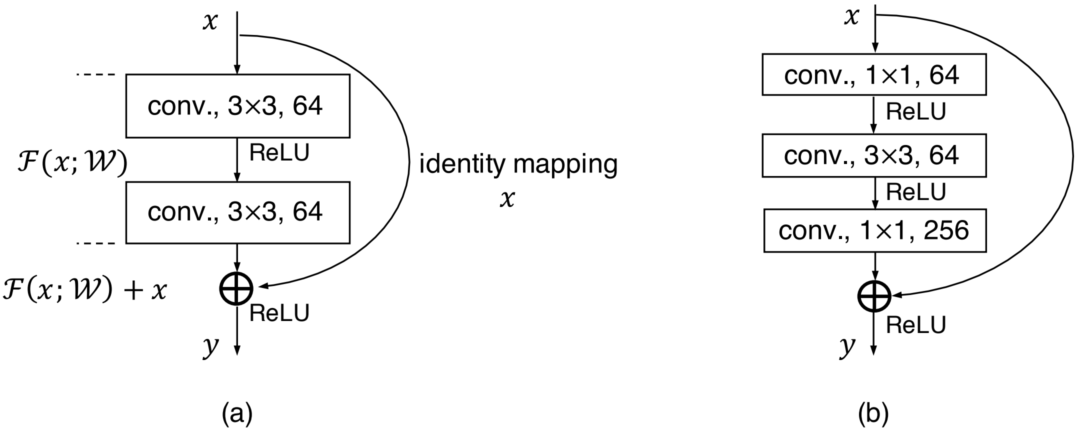


Figure 1-4: (a) The plain setting, 64 channels for all the layers. (b) The "bottle neck" setting, the input (x) and output (y) have 256 channels while the first two layers have 64 channels.

1.4 Outline of the Thesis

Chapter 2 We propose a novel style of residual connections dubbed "dual residual connection", which exploits the potential of paired operations, e.g., up- and down-sampling or convolution with large- and small-size kernels. We design a modular block implementing this connection style; it is equipped with two containers to which arbitrary paired operations are inserted. Adopting the "unraveled" view of the residual networks proposed by Veit et al. [130], we point out that a stack of the proposed modular blocks allows the first operation in a block interact with the second operation in *any* subsequent blocks. Specifying the two operations in each of the stacked blocks, we build a complete network for each individual task of image restoration. We experimentally evaluate the proposed approach on five image restoration tasks using nine datasets. The results show that the proposed networks with properly chosen paired operations outperform previous methods on almost all of the tasks and datasets.

Chapter 3 In addition, we further propose a universal network that has a single input and multiple output branches, to solve multiple image restoration tasks in the same model. This is made possible by improving the attention mechanism and an internal structure of the basic blocks used in the dual residual networks proposed in chapter 2. Experimental results show that the newly proposed approach achieves a new state-of-the-art performance on motion blur removal, haze removal (both in PSNR/SSIM) and JPEG artifact removal (in SSIM). To our knowledge, this is the first report of successful multi-task learning on multiple orthogonal image restoration tasks.

Chapter 4 Finally, we recall the issue we mentioned at the beginning, *i.e.*, there is a gap between human vision system and CNNs developed for computer vision in terms of robustness to image distortions. We investigate whether the proposed image restoration strategy can close the gap. The experimental results show that a simplified version of the proposed approach improves the CNNs' classification accuracy on Gaussian noise images to humans' level.

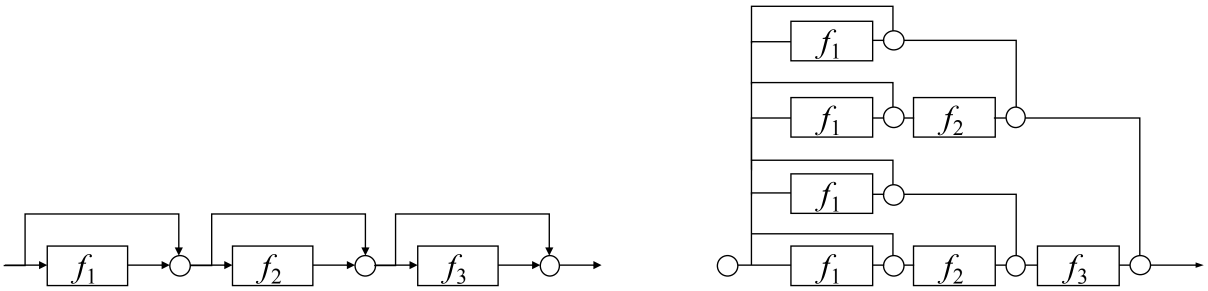
Chapter 5 Towards a deeper discussion between humans and CNNs, we further tackled the problem of evaluating CNNs' performance under humans' individual differences on a pair-wise ranking task. This is a difficult problem due to the reason that humans' judgments for a same question can split. We proposed a novel method that evaluates an artificial systems by judging if it is distinguishable from humans for ranking of N item pairs.

Chapter 2

Dual Residual Networks Leveraging the Potential of Paired Operations for Image Restoration

2.1 Introduction

The task of restoring the original image from its degraded version, or image restoration, has been studied for a long time in the fields of image processing and computer vision. As in many other tasks of computer vision, the employment of deep convolutional networks have made significant progress. In this study, aiming at further improvements, we pursue better architectural design of networks, particularly the design that can be shared across different tasks of image restoration. In this study, we pay attention to the effectiveness of paired operations on various image processing tasks. In [42], it is shown that a CNN iteratively performing a pair of up-sampling and down-sampling contributes to performance improvement for image-superresolution. In [122], the authors employ evolutionary computation to search for a better design of convolutional autoencoders for several tasks of image restoration, showing that network structures repeatedly performing a pair of convolutions with a large- and small-size kernels (*e.g.*, a sequence of conv. layers with kernel size 3, 1, 3, 1, 5, 3, and 1) perform well for image denoising. In this chapter, we will show further examples for other image restoration tasks.



(a) Three residual blocks

Unraveled view of (a)

Figure 2-1: A sequential connection of three residual blocks (left), and the unraveled view of it (right).

Assuming the effectiveness of such repetitive paired operations, we wish to implement them in deep networks to exploit their potential. We are specifically interested in how to integrate them with the structure of residual networks. The basic structure of residual networks is shown in Fig. 2-1(a), which have become an indispensable component for the design of modern deep neural networks. There have been several explanations for the effectiveness of the residual networks. A widely accepted one is the "unraveled" view proposed by Veit *et al.*[130]: a sequential connection of n residual blocks is regarded as an ensemble of many sub-networks corresponding to its implicit 2^n paths. A network of three residual blocks with modules f_1, f_2, and f_3, shown in Fig. 2-1(a), has $(2^3 =)8$ implicit paths from the input to output, i.e., $f_1 \rightarrow f_2 \rightarrow f_3$, $f_1 \rightarrow f_2$, $f_1 \rightarrow f_3$, $f_2 \rightarrow f_3$, f_1, f_2, f_3, and 1. Veit *et al.*also showed that each block works as a computational unit that can be attached/detached to/from the main network with minimum performance loss. Considering such a property of residual networks, how should we use residual connections for paired operations? Denoting the paired operations by f and g, the most basic construction will be to treat (f_i, g_i) as a unit module, as shown in Fig. 2-2(b). In this connection style, f_i and g_i are always paired for any i in the possible paths. In this study, we consider another connection style shown in Fig. 2-2(d), dubbed "dual residual connection". This style enables to pair[1] f_i and g_j for any i and j such that $i \leq j$. In the example of Fig.2-2(d), all the combinations of the two operations, (f_1, g_1), (f_2, g_2), (f_3, g_3), (f_1, g_2), (f_1, g_3), and (f_2, g_3), emerge in the possible paths. We conjecture that this increased

[1] Direct connection(s) of f_i to f_j is(are) impossible.

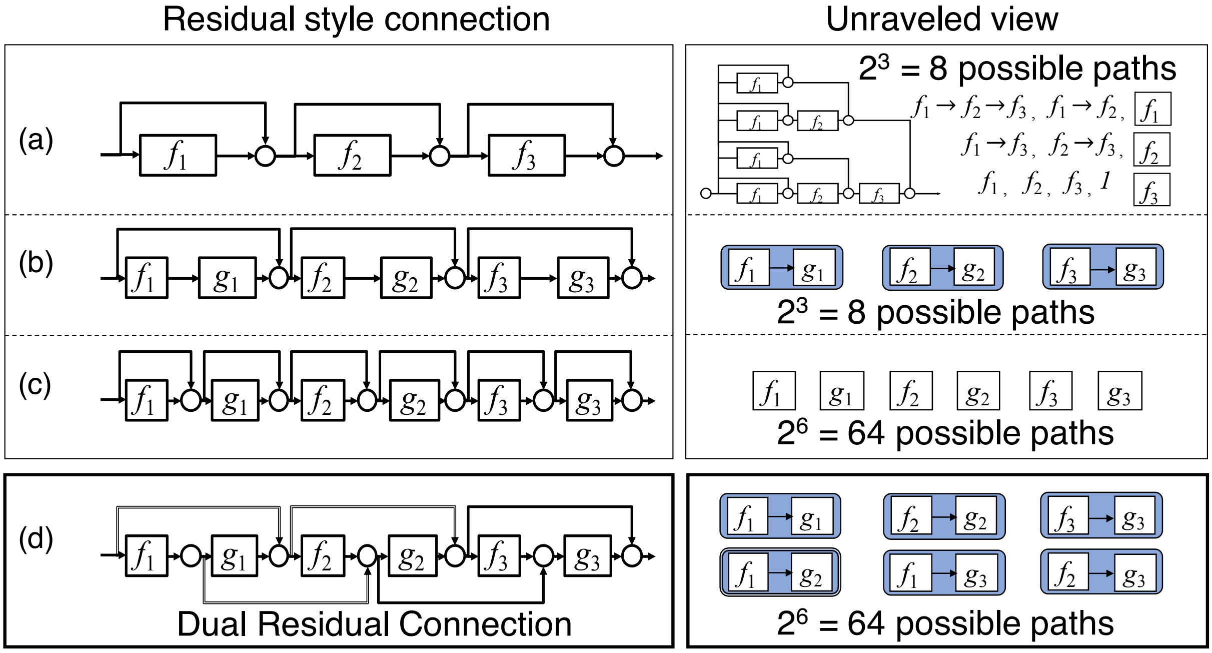

Residual style connection
Unraveled view
(a)
(b)
(c)
(d)
f_1 f_2 f_3
f_1 g_1 f_2 g_2 f_3 g_3
2^3 = 8 possible paths
$f_1 \rightarrow f_2 \rightarrow f_3$, $f_1 \rightarrow f_2$, $f_1 \rightarrow f_3$, $f_2 \rightarrow f_3$, f_1, f_2, f_3, 1
2^3 = 8 possible paths
2^6 = 64 possible paths
Dual Residual Connection
2^6 = 64 possible paths

Figure 2-2: Different construction of residual networks with a single or double basic modules. The proposed "dual residual connection" is (d).

number of potential interactions between $\{f_i\}$ and $\{g_j\}$ will contribute to improve performance for image restoration tasks. Note that it is guaranteed that $f_.$ and $g_.$ are always paired in the possible paths. This is not the case with other connection styles such as the one depicted in Fig. 2-2(c). Note that i) compared to (b), the proposed (d) has more possible paths and paired operations (depicted by blue squares with a f and a g in each); ii) compared to (c), the (d) has paired operations while the (c) doesn't. We call the building block for implementing the proposed dual residual connections *Dual Residual Block* (DuRB); see Fig. 2-3. We examine its effectiveness on five image restoration tasks shown in Fig. 2-3 using nine datasets. DuRB is a generic structure that has two containers for the paired operations, and the users choose two operations for them. For each task, we specify the paired operations of DuRBs as well as the entire network. My experimental results show that the proposed networks outperform the state-of-the-art methods in these tasks, which supports the effectiveness of my approach.

In this chapter, we will first briefly go over the recent studies on the five image restoration tasks, then we introduce the proposed approach and show the experimental results. Detailed information about the experimental settings as well as more visual results will be provided in

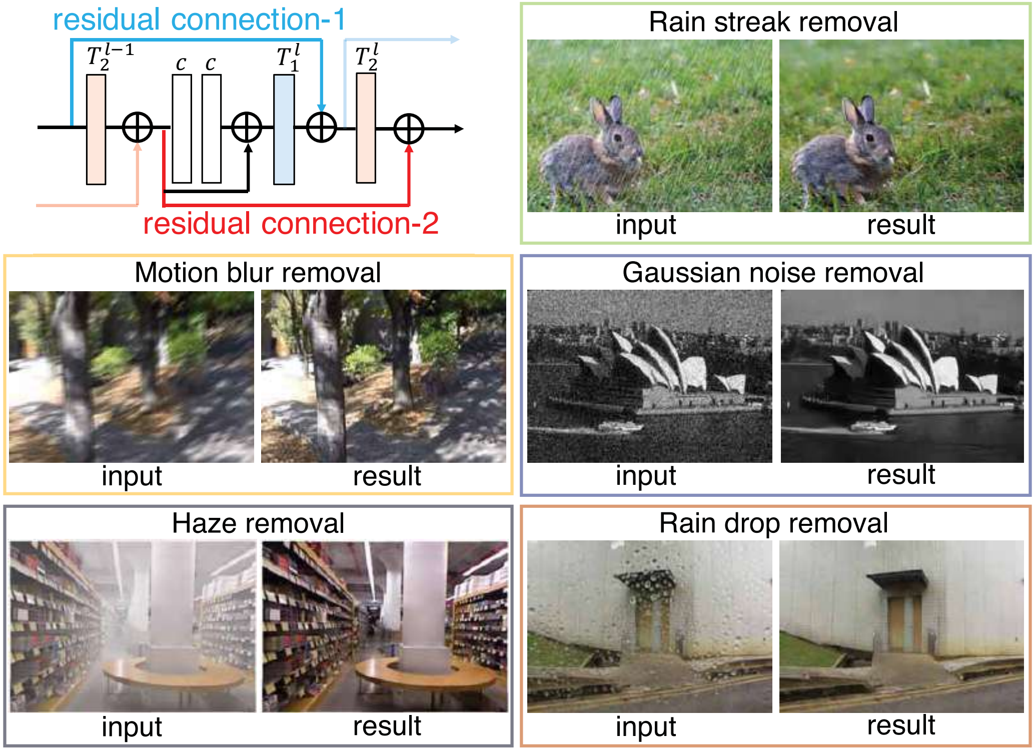


Figure 2-3: Upper-left: the structure of a unit block having the proposed dual residual connections; T_1^l and T_2^l are the containers for two paired operations; c denotes a convolutional layer. Other panels: five image restoration tasks considered in this paper.

Appendix.

2.2 Pioneering Work

Gaussian noise removal Application of neural networks to noise removal has a long history [58, 134, 2, 153, 154]. Mao *et al.*[84] proposed REDNet, which consists of multiple convolutional and de-convolutional layers with symmetric skip connections over them. Tai *et al.*[126] proposed MemNet with local memory blocks and global dense connections, showing that it performs better than REDNet. However, Suganuma *et al.*[122] showed that standard convolutional autoencoders with repetitive pairs of convolutional layers with large- and small-size kernels outperform them by a good margin, which are found by architectural search based on evolutionary computation.

Motion blur removal This task has a long history of research. Early works [29, 140, 139, 6] attempt to simultaneously estimate both blur kernels and sharp images. Recently, CNN-based methods [123, 39, 93, 66, 132] achieve good performance for this task. Nah *et al.*[93] proposed a coarse-to-fine approach along with a modified residual block [48]. Kupyn *et al.*[66] proposed an approach based on Generative Adversarial Network (GAN) [41]. New datasets were created in [93] and [66].

Haze removal Many studies assume the following model of haze: $I(x) = J(x)t(x)+A(x)(1-t(x))$, where I denotes a hazy scene image, J is the true scene radiance (the clear image), t is a transmission map, A is global atmospheric light. The task is then to estimate A, t, and thus $J(x)$ from the input $I(x)$ [44, 9, 87, 151, 144]. Recently, Zhang *et al.*[151] proposed a method that uses CNNs to jointly estimate t and A, which outperforms previous approaches by a large margin. Ren *et al.*[102] and Li *et al.*[74] proposed method to directly estimate $J(x)$ without explicitly estimating t and A. Yang *et al.*[144] proposed a method that integrates CNNs to classical prior-based method.

Raindrop detection and removal Various approaches [67, 149, 60, 105, 142] have been proposed to tackle this problem in the literature. Kurihata *et al.*[67] proposed to detect raindrops with raindrop-templates learned using PCA. Ramensh [60] proposed a method based on K-Means clustering and median filtering to estimate clear images. Recently, Qian *et al.*[100] proposed a hybrid network consisting of a convolutional-LSTM for localizing raindrops and a CNN for generating clear images, which is trained in a GAN framework.

Rain-streak removal Fu *et al.*[34] use “guided image filtering” [45] to extract high-frequency components of an image, and use it to train a CNN for rain-streak removal. Zhang *et al.*[152] proposed to jointly estimate rain density and de-raining result to alleviate the non-uniform rain density problem. Li *et al.*[75] regards a heavy rainy image as a clear image added by an accumulation of multiple rain-streak layers and proposed a RNN-based method to restore the clear image. Li *et al.*[72] proposed an non-locally enhanced version of DenseBlock [55] for this task, their network outperforms previous approaches by a good margin.

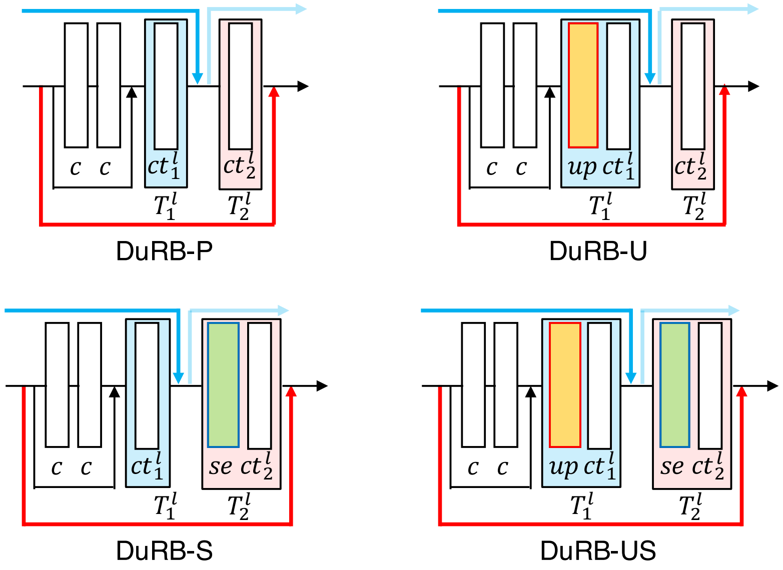


Figure 2-4: Four different implementations of the DuRB; c is a convolutional layer with 3×3 kernels; ct^l_1 and ct^l_2 are convolutional layers, each with kernels of a specified size and dilation rate; up is up-sampling (we implemented it using PixelShuffle [117]); se is SE-ResNet Module [52] that is in fact a channel-wise attention mechanism.

2.3 Dual Residual Blocks

The basic architecture of the proposed Dual Residual Block is shown in the upper-left corner of Fig. 2-3, in which we use c to denote a convolutional layer (with 3×3 kernels) and T^l_1 and T^l_2 to denote the containers for the paired first and second operations, respectively, in the l^{th} DuRB in a network. The two convolutional layers with a residual connection (set before T^l_1) are used to smooth information changing for the DuRB. This is rather an implementation trick. Normalization layers (such as batch normalization [56] or instance normalization [128]) and ReLU [94] layers can be incorporated when it is necessary. We design DuRBs for each individual task, or equivalently choose the two operations to be inserted into the containers T^l_1 and T^l_2. We use four different designs of DuRBs, DuRB-P, DuRB-U, DuRB-S, and DuRB-US, which are shown in Fig. 2-4. The specified operations for [T^l_1, T^l_2] are [conv., conv.] for DuRB-P, [up-sampling+conv., down-sampling (by conv. with stride=2)] for DuRB-U, [conv., channel-wise attention[2]+conv.] for DuRB-S, and [up-sampling+conv., channel-wise attention+down-

[2] It is implemented using the SE-ResNet Module [52].

Table 2.1: Performance of the three connection types of Fig. 1(b)-(c). '-'s indicate infeasible applications.

	(b)	(c)	(d)
Gaussian noise	24.92 / 0.6632	24.85 / 0.6568	**25.05 / 0.6755**
Real noise	36.76 / 0.9620	36.81 / 0.9627	**36.84 / 0.9635**
Motion blur	29.46 / 0.9035	-/-	**29.90 / 0.9100**
Haze	31.20 / 0.9803	-/-	**32.60 / 0.9827**
Raindrop	24.70 / 0.8104	25.12 / 0.8151	**25.32 / 0.8173**
Rain-streak	32.85 / 0.9214	33.13 / 0.9222	**33.21 / 0.9251**

sampling] for DuRB-US, respectively. We will use DuRB-P for noise removal and raindrop removal, DuRB-U for motion blur removal, DuRB-S for rain-streak and raindrop removal, and DuRB-US for haze removal.

Before proceeding to further discussions, we present here experimental results that show the superiority of the proposed dual residual connection to other connection styles shown in Fig. 2-2(b) and (c). In the experiments, three networks build on the three base structures (b), (c), and (d) of Fig. 2-2 were evaluated on the five tasks. For Gaussian&real-world noise removal, motion blur removal, haze removal, raindrop and rain-streak removal, we use DuRB-P, DuRB-U, DuRB-US, DuRB-S&DuRB-P and DuRB-S to construct the base structures. Number of blocks and all the operations in the three structures as well as other experimental configurations are fixed in each comparison. The datasets for the six comparisons are BSD-grayscale, Real-World Noisy Image Dataset, GoPro Dataset, Dehaze Dataset, RainDrop Dataset and DID-MDN Data. Table 2.1 shows their performance. Note that '-' in the table indicate that the connection cannot be applied to DuRB-U and DuRB-US due to the difference in size between the output of f and the input to g. It can be seen that the proposed structure (d) performs the best for all the tests.

2.4 Five Image Restoration Tasks

In this section, we describe how the proposed DuRBs can be applied to multiple image restoration tasks, noise removal, motion blur removal, haze removal, raindrop removal and rain-streak removal.

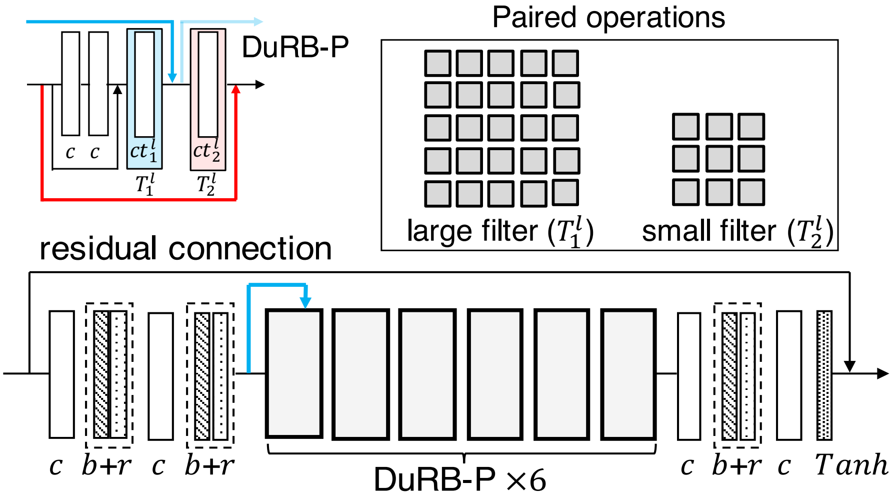


Figure 2-5: DuRN-P: dual residual network with DuRB-P's [conv. w/ a large kernel and conv. w/ a small kernel] for Gaussian noise removal. $b + r$ is a batch normalization layer followed by a ReLU layer; and $Tanh$ denotes hyperbolic tangent function.

2.4.1 Noise Removal

Network Design We design the entire network as shown in Fig. 2-5. It consists of an input block, the stack of six DuRBs, and an output block, additionally with an outermost residual connection from the input to output. The layers c, $b+r$ and $Tanh$ in the input and output blocks are convolutional layer (with 3×3 kernels, stride $=$ 1), batch normalization layer followed by a ReLU layer, and hyperbolic tangent function layer, respectively. We employ DuRB-P (i.e., the design in which each of the two operations is single convolution; see Fig. 2-4) for DuRBs in the network. Inspired by the networks discovered by neural architectural search for noise removal in [122], we choose for T_1 and T_2 convolution with large- and small-size receptive fields. We also choose the kernel size and dilation rate for each DuRB so that the receptive field of convolution in each DuRB grows its size with l. More details are given in the appendix. We set the number of channels to 32 for all the layers. We call the entire network DuRN-P. For this task, we employed l_2 loss for training the DuRN-P.

Results: Additive Gaussian Noise Removal The proposed network is tested on the task of

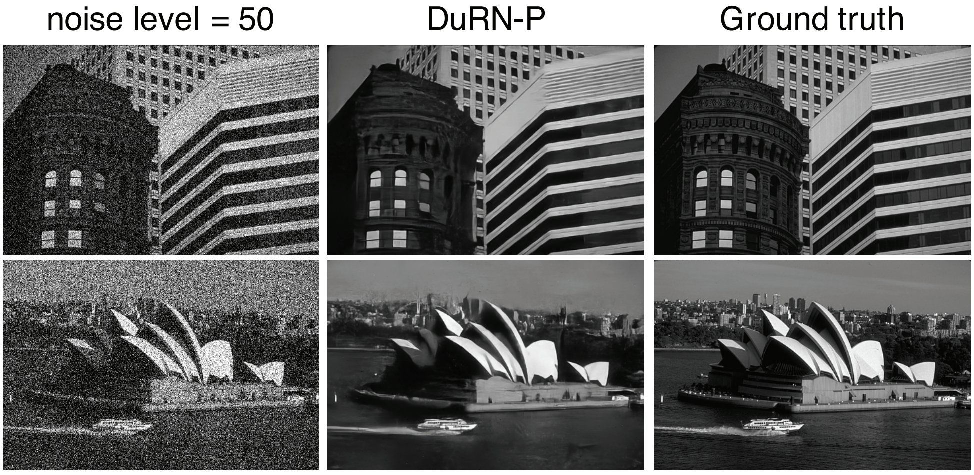


Figure 2-6: Some examples of the results by the proposed DuRN-P for additive Gaussian noise removal. Sharp images can be restored from heavy noises ($\sigma = 50$).

Table 2.2: Results for additive Gaussian noise removal on BSD200-grayscale and noise levels (30, 50, 70). The numbers are PSNR/SSIM.

	30	50	70
REDNet[84]	27.95 / 0.8019	25.75 / 0.7167	24.37 / 0.6551
MemNet[126]	28.04 / 0.8053	25.86 / 0.7202	24.53 / 0.6608
E-CAE[122]	28.23 / 0.8047	26.17 / 0.7255	24.83 / 0.6636
DuRN-P (ours)	**28.50 / 0.8156**	**26.36 / 0.7350**	**25.05 / 0.6755**

removing additive Gaussian noise of three levels (30, 50, 70) from a gray-scale noisy image. Following the same experimental protocols used by previous studies, we trained and tested the proposed DuRN-P using the training and test subsets (300 and 200 grayscale images) of the BSD-grayscale dataset [86]. More details of the experiments are provided in the Appendix. We show the quantitative results in Table 2.2 and visual results in Fig. 2-6. It is observed from Table 2.2 that the proposed network outperforms the previous methods for all three noise levels.

Results: Real-World Noise Removal We also tested the DuRN-P on the Real-World Noisy Image Dataset [136], which consists of 40 pairs of an instance image (a photograph taken by a CMOS camera) and the mean image (mean of multiple shots of the same scene taken by the CMOS camera). All the batch normalization layers are removed from the DuRN-P for this experiment, as the real-world noise captured in this dataset do not vary greatly. The details of

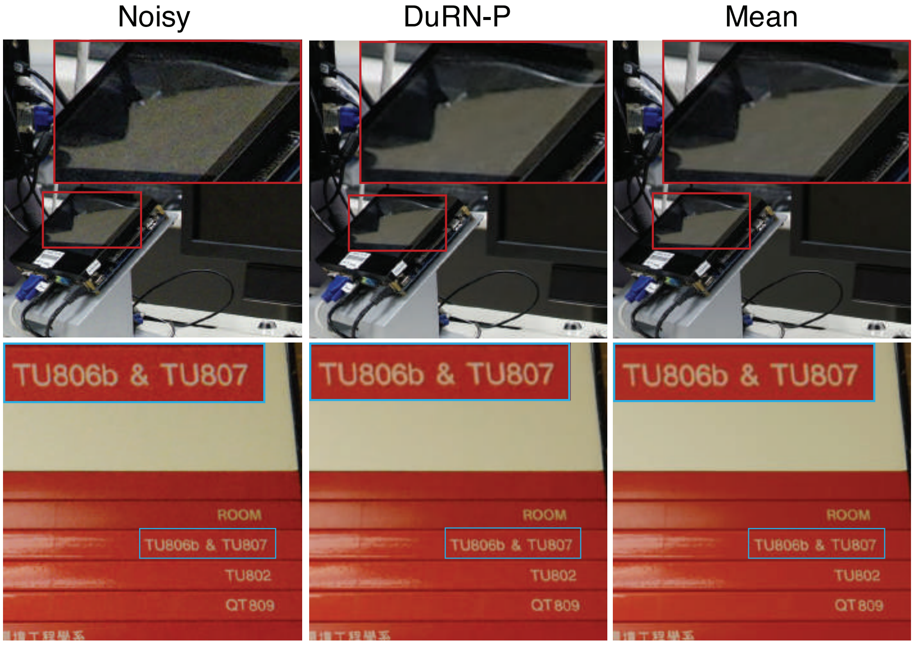


Figure 2-7: Examples of noise removal by the proposed DuRN-P for images from Real-World Noisy Image Dataset. The results are sometimes even better than the mean image (used as the ground truth); see the artifact around the letters in the bottom.

Table 2.3: Results on the Real-World Noisy Image Dataset [136]. The results were measured by PSNR/SSIM. The last row shows the number of parameters for each CNN.

	REDNet[84]	MemNet[126]	E-CAE[122]	**DuRN (ours)**
PSNR/SSIM	35.56 / 0.9475	- / -	35.45 / 0.9492	**36.83 / 0.9635**
# of param.	4.1×10^6	2.9×10^6	1.1×10^6	8.2×10^5

the experiments are given in the Appendix. The quantitative results of three previous methods and our method are shown in Table 2.3. We used the authors' code to evaluate the three previous methods. (As the MemNet failed to produce a competitive result, we left the cell empty for it in the table.) It is seen that the proposed method achieves the best result despite the smaller number of parameters. Examples of output images are shown in Fig. 2-7. It is observed that the proposed DuRN-P has cleaned noises well. It is noteworthy that the DuRN-P sometimes provides better images than the "ground truth" mean image; see the bottom example in Fig. 2-7.

2.4.2 Motion Blur Removal

The task is to restore a sharp image from its motion blurred version without knowing the latent blur kernels (i.e., the “blind-deblurring” problem).

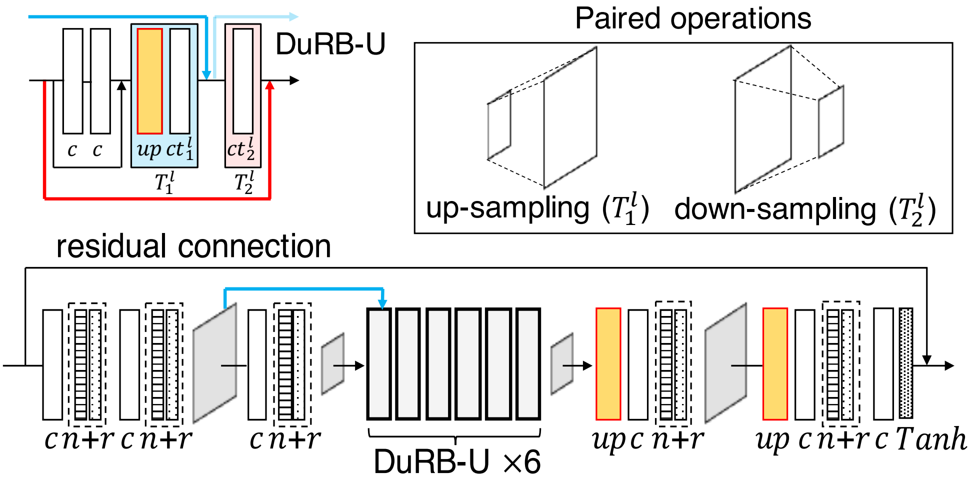


Figure 2-8: DuRN-U: Dual Residual Network with DuRB-U’s (up- and down-sampling) for motion blur removal. $n + r$ denotes an instance normalization layer followed by a ReLU layer.

Network Design Previous works such as [132] reported that the employment of up- and down-sampling operations is effective for this task. Following this finding, we employ up-sampling and down-sampling for the paired operations. we call this as DuRB-U; see Fig. 2-8. we use PixelShuffle [117] for implementing up-sampling. For the entire network design, following many previous works [132, 66, 151, 160], we choose a symmetric encoder-decoder network; see Fig. 2-8. The network consists of the initial block, which down-scales the input image by 4:1 down-sampling with two convolution operations (c) with stride $= 2$, and instance normalization + ReLU ($n+r$), and six repetitions of DuRB-U’s, and the final block which up-scales the output of the last DuRB-U by applications of 1:2 up-sampling (up) to the original size. We call this network DuRN-U. For this task, we employed a weighted sum of SSIM and l_1 loss for training the DuRN-U. The details abopout experimental settings are given in the Appendix.

Results: GoPro Dataset We tested the proposed DuRN-U on the GoPro-test dataset [93] and compared its results with the state-of-the-art DeblurGAN [66]. The GoPro dataset consists of 2,013 and 1,111 non-overlapped training (GoPro-train) and test (GoPro-test) pairs of blurred

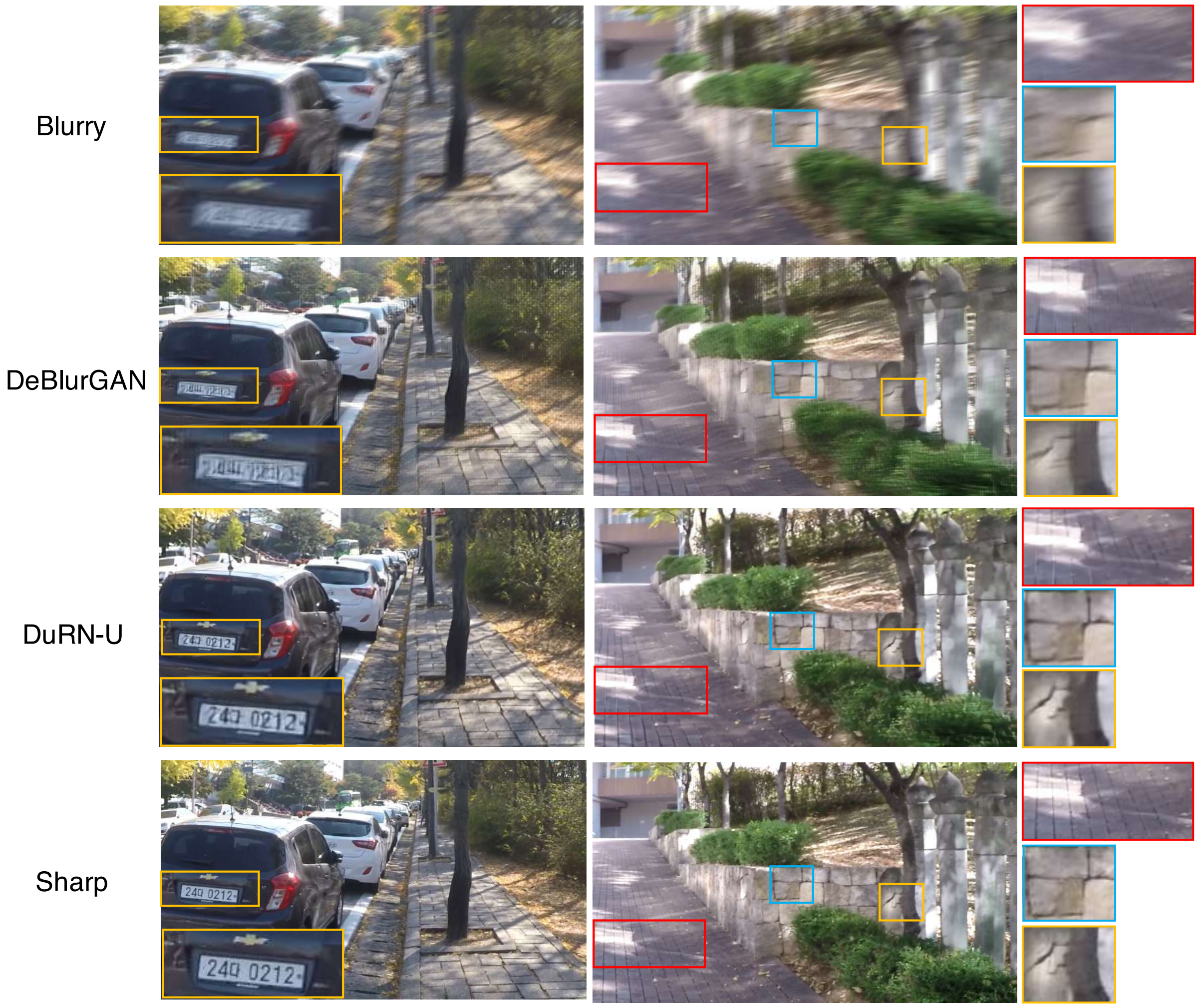


Figure 2-9: Examples of motion blur removal on GoPro-test dataset.

and sharp images. We show quantitative results in the Table 2.4. DeblurGAN yields outstanding SSIM number, whereas the proposed DuRN-U is the best in terms of PSNR. Examples of deblurred images are shown in Fig. 2-9. It is observed that the details such as cracks on a stone-fence or numbers written on the car plate are restored well enough to be recognized.

Results: Object Detection from Deblurred Images In [66], the authors evaluated their deblurring method (DeBlurGAN) by applying an object detector to the deblurred images obtained by their method. Following the same procedure and data (Car Dataset), we evaluate the proposed DuRN-U that is trained on the GoPro-train dataset. The Car Dataset contains 1,151 pairs of blurred and sharp images of cars. We employ YOLO v3 [101] trained using the Pascal VOC [27] for the object detector. The detection results obtained for the sharp image by the same

Table 2.4: Results of motion blur removal for the GoPro-test dataset.

	GoPro-test
Sun *et al.*[123]	24.6 / 0.84
Nah *et al.*[93]	28.3 / 0.92
Xu *et al.*[140]	25.1 / 0.89
DeBlurGAN[66]	27.2 / **0.95**
DuRN-U (ours)	**29.9** / 0.91

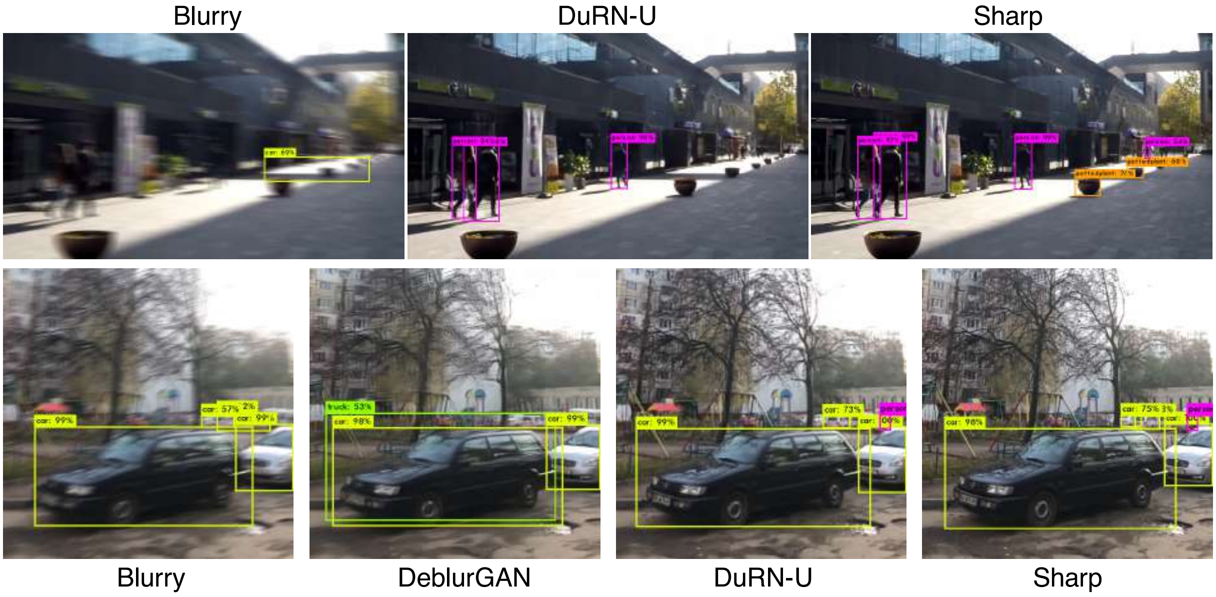

Blurry
DuRN-U
Sharp
Blurry
DeblurGAN
DuRN-U
Sharp

Figure 2-10: Examples of object detection from original blurred images and their deblurred versions.

Table 2.5: Accuracy of object detection from deblurred images obtained by DeBlurGAN [66] and the proposed DuRN-U on Car Dataset.

	Blurred	DeBlurGAN[66]	**DuRN-U (ours)**
mAP (%)	16.54	26.17	**31.15**

YOLO v3 detector are utilized as the ground truths used for evaluation. Table 2.5 shows quantitative results (measured by mAP), from which it is seen that the proposed DuRN-U outperforms the state-of-the-art DeBlurGAN. Figure 2-10 shows examples of detection results on the GoPro-test dataset and Car Dataset. It is observed that DuRN-U can recover details to a certain extent that improves accuracy of detection.

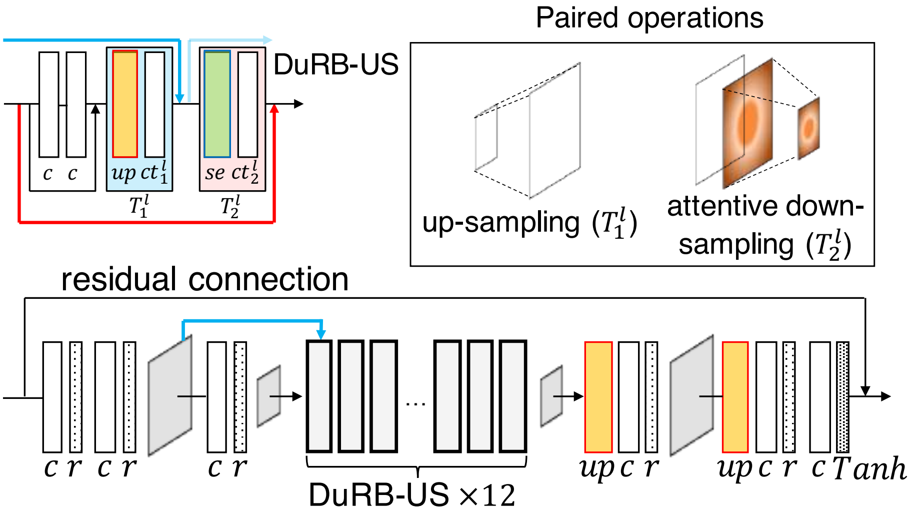


Figure 2-11: DuRN-US: dual residual network with DuRB-US's (up- and down-sampling and channel-wise attention (SE-ResNet Module)) for haze removal.

2.4.3 Haze Removal

Network Design In contrast with previous studies where a CNN is used to explicitly estimate a transmission map that models the effects of haze, we pursue a different strategy, which is to implicitly estimate a transmission map using an attention mechanism. The model estimates the dehazed image from an input image in an end-to-end fashion. We design DuRB's for this task by employing up-sampling (up) implemented using PixelShuffle [117] with a convolutional layer (ct_1^l) in T_1^l and channel-wise attention (se) implemented using SE-ResNet module [52] with a conv. layer (ct_2^l) in T_2^l. More details are given in Appendix. The entire network (named DuRN-US) has an encoder-decoder structure similar to the DuRN-U designed for motion blur removal, as shown in Fig. 2-11. 12 DuRB-US's are stacked in the middle of the network; the number of channels is 64 for all the layers. In Appendix, we demonstrate how the network estimates a transmission map inside its attention mechanisms. For this task, we employed a weighted sum of SSIM and l_1 loss for training the DuRN-US.

Results In order to evaluate the proposed DuRN-US, we trained and tested it on two datasets, the Dehaze Dataset and the RESIDE dataset. The training and test (Dehaze-TestA) subsets in the Dehaze Dataset consist of 4,000 and 400 non-overlapped samples of indoor scenes, respectively. RESIDE contains a training subset of 13,990 samples of indoor scenes and a few test subsets.

Table 2.6: Results for haze removal on Dehaze-TestA dataset and RESIDE-SOTS dataset. The measure SSIM and SSIM/PSNR is employed for the first and second dataset.

Dehaze-TestA	
He *et al.*[44]	0.8642
Zhu *et al.*[161]	0.8567
Berman *et al.*[8]	0.7959
Li *et al.*[69]	0.8842
Zhang *et al.*[151]	0.9560
DuRN-US (ours)	**0.9827**

RESIDE-SOTS	
Berman *et al.*[8]	17.27 / 0.75
Ren *et al.*[102]	17.57 / 0.81
Cai *et al.*[14]	21.14 / 0.85
Li *et al.*[69]	19.06 / 0.85
Ren *et al.*[103]	22.30 / 0.88
DuRN-US (ours)	**32.12 / 0.98**

Following [103], we used a subset SOTS (Synthetic Objective Testing Set) that contains 500 indoor scene samples for evaluation. It should be noted that the state-of-the-art method on the Dehaze Dataset, DCPDN [151], is trained using i) hazy images, ii) ground truth images, iii) ground truth global atmosphere light , iv) ground truth transmission maps; additionally, its weights are initialized by those of DenseNet [55] pre-trained on the ImageNet[108]. The proposed DuRN-US is trained only using i) and ii). Table 2.6 show results on Dehaze-TestA and RESIDE-SOTS datasets, respectively. Figure 2-12 shows examples of the results obtained by the proposed network and others for the same input images. In sub-figure (A), we show results for two synthesized images produced by the DCPDN (the second best approach in terms of SSIM and PSNR) and our DuRN-US. It is observed that DuRN-US yields better results for these two images. In sub-figure (B), we show results for two real-world hazy images produced by two state-of-the-art methods, GFN[103] and DCPDN[151], and by ours. It can be observed that our network yields the most realistic dehazed images. It is noteworthy that our DuRN-US can properly deal with strong ambient light (sunshine coming behind the girl). See the example in the left-bottom of Fig. 2-12.

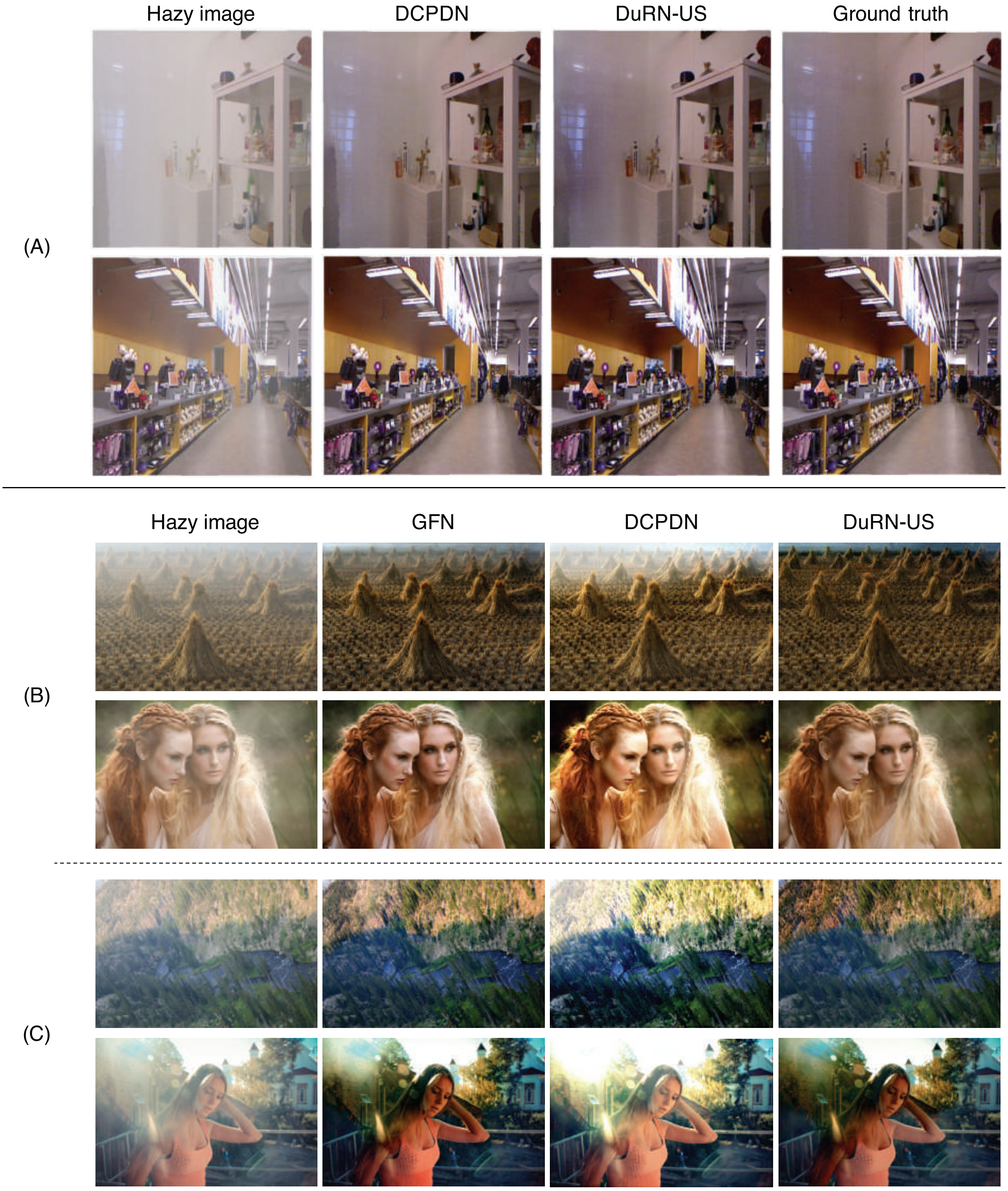


Figure 2-12: Examples of de-hazing results obtained by DuRN-US and others on (A) synthesized images, (B) real images and (C) light hazy images.

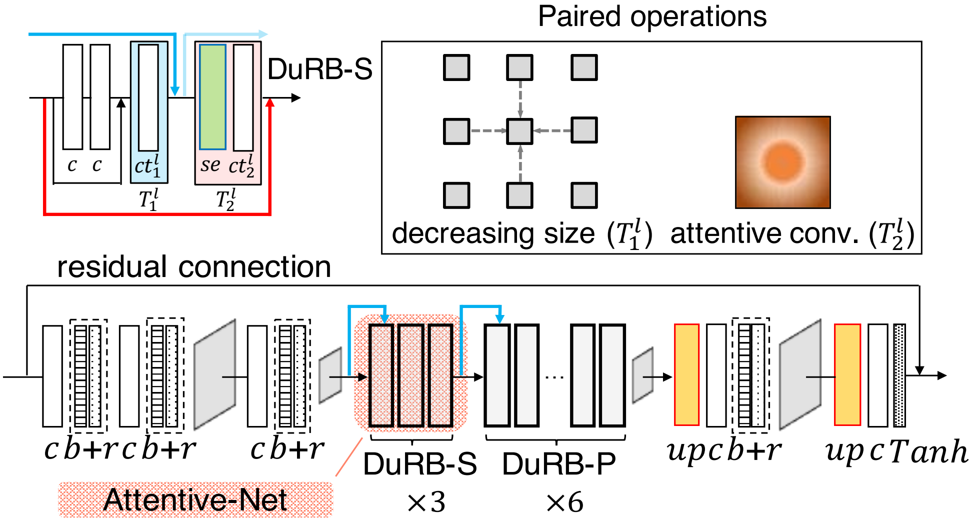


Figure 2-13: DuRN-S-P: Hybrid dual residual network with DuRB-S's and DuRB-P's for raindrop removal.

2.4.4 Raindrop removal

Network Design The task can naturally be divided into two stages, that of identifying the regions of raindrops and that of recovering the pixels of the identified regions. The second stage is similar to image inpainting and may not be difficult, as there are a lot of successful methods for image inpainting. Then, the major issue is with the first stage. Following this two-stage approach, the state-of-the-art method [100] uses an attentive-recurrent network to produce an attention map that conveys information about raindrops; then, the attention map along with the input image are fed to a convolutional encoder-decoder network to estimate the ground truth image. It also employs adversarial training with a discriminator to make the generated images realistic. We show the proposed DuRBs are powerful enough to perform these two-stage computations in a standard feedforward network, if the DuRBs are properly designed in proper positions in the entire network. To be specific, we choose the encoder-decoder structure for the entire network, and in its bottleneck part, we set three DuRB-S's followed by six DuRB-P's. For ct_1^l in the three DuRB-S's, we use convolution with a 3×3 kernel with decreasing dilation rates, 12, 8, and 6, in the forward direction, aiming to localize raindrops in a coarse-to-fine manner in the three DuRB-S's in a row. For the six DuRB-P's, we employ the same

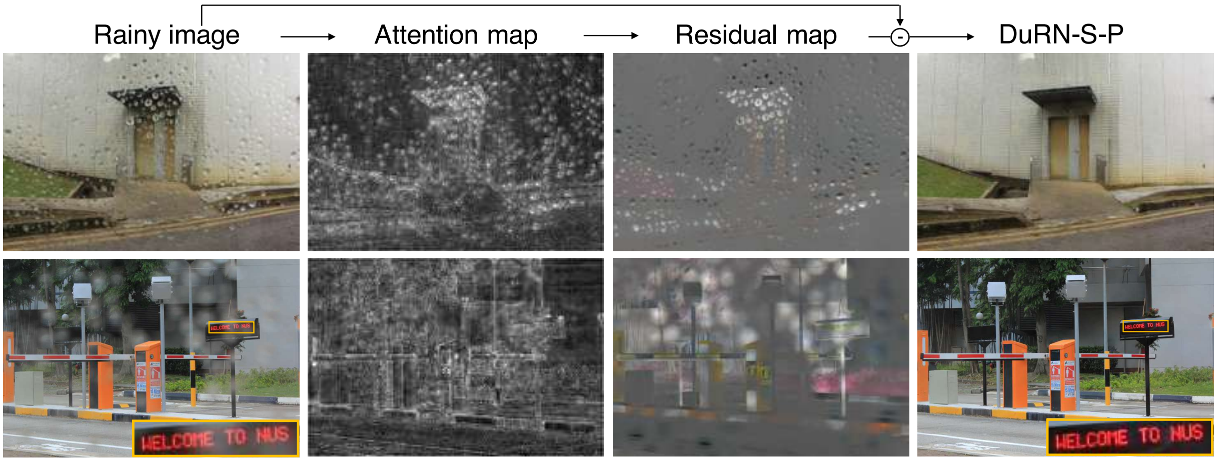


Figure 2-14: Examples of raindrop removal along with internal activation maps of DuRN-S-P. The "Attention map" and "Residual map" are the outputs of the Attentive-Net and the last $Tanh$ layer shown in Fig. 2-13; they are normalized for better visibility.

Table 2.7: Quantitative result comparison on RainDrop Dataset [100].

	Qian *et al.*[100]	**DuRN-S-P (ours)**
TestSetA	**31.51** / 0.9213	31.24 / **0.9259**
TestSetB	24.92 / 0.8090	**25.32 / 0.8173**

strategy as in noise removal etc., which is to apply a series of convolution with an increasing receptive field size in the forward direction. We call the entire network DuRN-S-P. For this task, we employed a weighted sum of SSIM and l_1 loss for training the DuRN-S-P.

Results We trained and evaluated the DuRN-S-P on the RainDrop Dataset. It contains 861 training samples and 58/249 test samples called TestSetA/TestSetB. TestSetA is a subset of TestSetB, and is considered to have better alignment than TestSetB by the dataset's author. Table 2.7 shows the results. It is seen that the proposed method outperforms the state-of-the-art method for three out of four combinations of two test sets and two evaluation metrics. It is noteworthy that the proposed method does not use a recurrent network or adversarial training. Figure 2-14 shows some examples of the results obtained by the proposed method. It is seen that the results of the proposed method are visually good, and comparable to the method of [100] (more examples for visual comparison is provided in the appendix). The "Attention map" and "Residual map" of Fig. 2-14 are the over-channel summation of the output of Attentive-Net and the output of the last $Tanh$ layer, respectively; see Fig. 2-13.

2.4.5 Rain-streak Removal

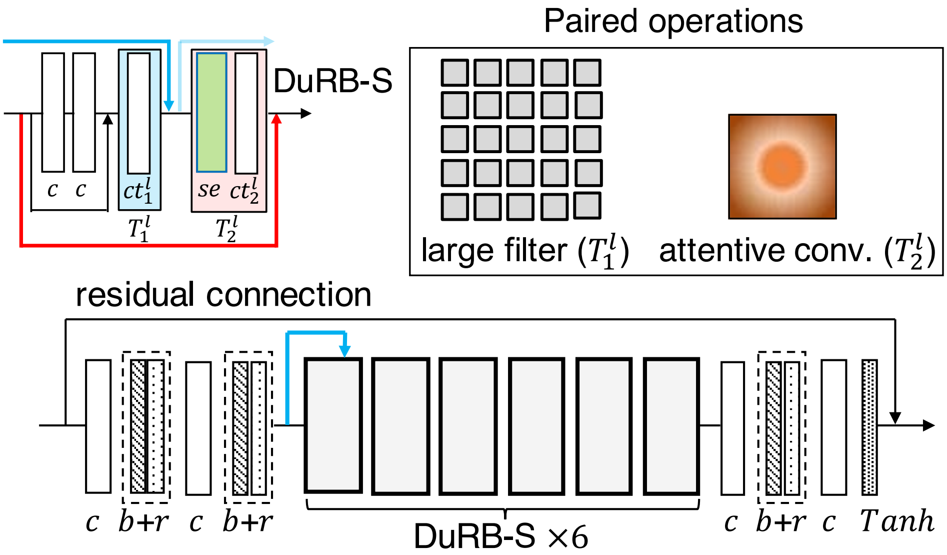


Figure 2-15: DuRN-S: dual residual network with DuRB-S's (large filter convolution and channel-wise attention (SE-ResNet Module) with small filter convolution for the pair) for rain-streak removal.

Network Design It is shown in [72] that the mechanism that selectively weighs feature maps using global information works effectively for this task. Borrowing this idea, we employ a channel-wise attention mechanism to perform similar feature weighting. The overall design of the network for this task is similar to the DuRN-P designed for Gaussian noise removal. A difference is that we use DuRB-S instead of DuRB-P to use the attention mechanism. The details are given in Appendix. For this task, we employed a weighted sum of SSIM and l_1 loss for training the network.

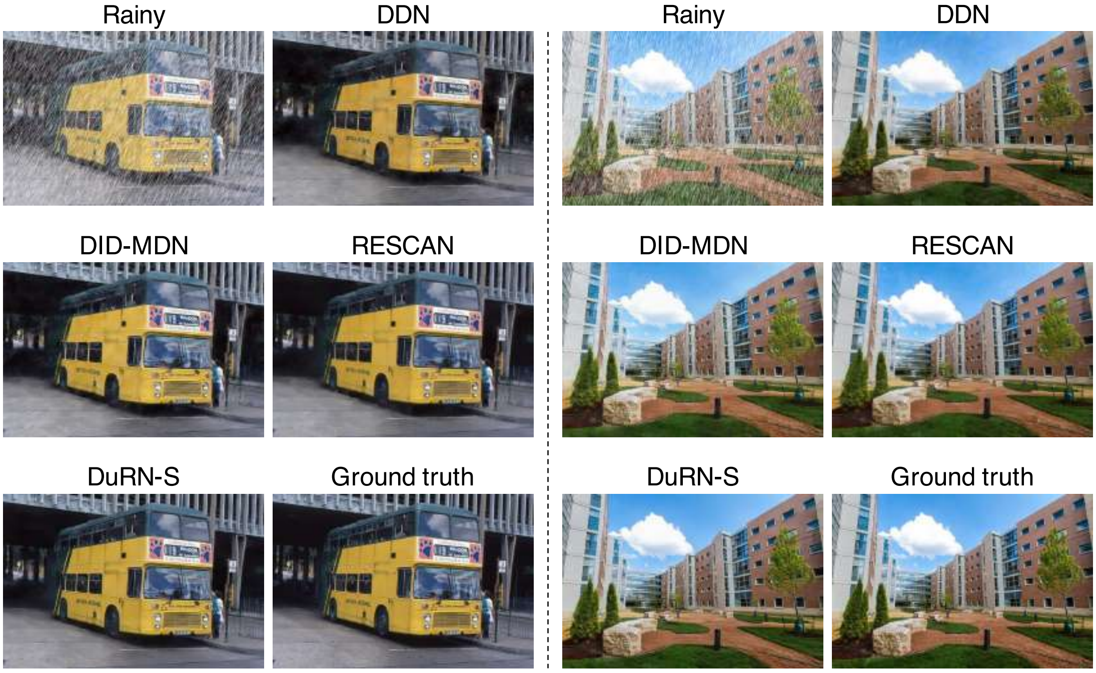


Figure 2-16: Examples of rain-streak removal obtained by four methods including the proposed one (DuRN-S).

Table 2.8: Results on two rain-streak removal datasets.

	DDN Data	DID-MDN Data
DDN[34]	28.24 / 0.8654	23.53 / 0.7057
JORDER [145]	28.72 / 0.8740	30.35 / 0.8763
DID-MDN [152]	26.17 / 0.8409	28.30 / 0.8707
RESCAN [75]	-/-	32.48 / 0.9096
NLEDN [72]	29.79 / 0.8976	33.16 / 0.9192
DuRN-S (ours)	**30.61 / 0.9136**	**33.21 / 0.9251**

Results We tested the proposed network (DuRN-S) on two benchmark datasets, the DDN-Data, which consists of 9,100 training pairs and 4,900 test pairs of rainy and clear images, and the DID-MDN Data, which consists of 12,000 training pairs and 1,200 test pairs. Table 2.8 shows the results. Those for the previous methods except RESCAN [75] are imported from [72]. It is seen that the proposed network achieves the best performance. Examples of the output images are provided in Fig. 2-16.

2.5 Summary and Discussions

We have proposed a style of residual connection, dubbed “dual residual connection”, aiming to exploit the potential of paired operations for image restoration tasks. We have shown the design of a modular block (DuRB) that implements this connection style, which has two containers for the paired operations such that the user can insert any arbitrary operations to them. We have also shown choices of the two operations in the block as well as the entire networks (DuRN) containing a stack of the blocks for five different image restoration tasks. The experimental results obtained using nine datasets show that the proposed approach consistently works better than previous methods.

Chapter 3

A Universal Network Applicable to Multiple Image Restoration Tasks

3.1 Introduction

There are many factors causing image degradation/distortion, for each of which there are a (large) number of studies in the past, such as motion/defocus blur [113, 97, 140, 139, 20], several types of noises (e.g., Gaussian, real-world noise, etc.) [23, 17, 138, 137], JPEG compression noise [146, 16, 12, 49], rain streak [76, 59, 18], raindrop [67, 149, 60], haze [44, 87, 8] etc. Previous studies have treated each of these degradation types individually and developed "dedicated" methods for each factor. This is also the case with recent studies [66, 100, 122, 75, 147, 151] utilizing deep learning, including the one introduced in Chapter 2. There are different networks for different degradation types. It should be noted that a few recent studies attempt to deal with combined degradation [121, 150], having proposed single networks that can deal with images having mixed degradation types with unknown mixing ratio. However, their restoration accuracy (for images with a single degradation type) tends to be much lower than the dedicated methods; moreover, they can only deal with small image patches (i.e., 64×64 pixels) for now. Thus, it is fair to say that they are still in an experimental stage. In this section, we consider another approach and choose to use a single network to deal with multiple degradation factors. Figure 3-1 shows the standard approach and the proposed approach; we use a network having

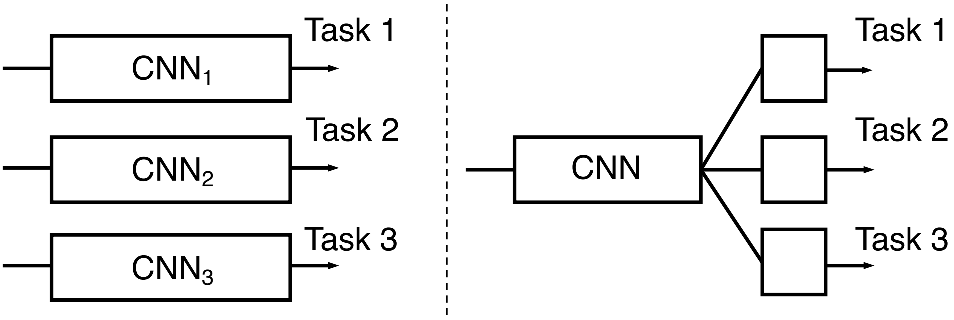


Figure 3-1: Left: Approach employed in recent studies, i.e., designing/training a different network for each image restoration task dealing with a single degradation factor. Right: The proposed approach; a single network having a single input and multiple output branches is trained on multiple image restoration tasks.

a single input and multiple output branches, each of which is dedicated to a different image restoration task. Although this is also a standard formulation of multi-task learning (MTL), it remains unclear so far if MTL works for image restoration tasks. Moreover, my motivation is not merely to improve the current state-of-the-art by employing MTL. This study is motivated more by a desire to know what is (and should be) learned by the CNNs on image restoration tasks and what is their optimal design for the tasks. In earlier studies of image restoration, it was of a primary interest to model and represent the statistics or prior of natural images, which is then utilized to restore the original image from an input by, for instance, using it in the framework of the Bayesian inference [29, 88, 106]. On the other hand, in the recent approaches that use CNNs in an end-to-end fashion, there is no explicit modeling of image prior. We conjecture that in order to restore original images accurately, a network must learn natural image prior in some form for any degradation factors. If this is true, optimal networks for individual degradation factors may share representation of natural image prior. The proposed method is built upon this conjecture. To enable to deal with multiple degradation factors with a single network, we propose a new design of networks build upon the *Dual Residual Networks (DuRNs)* introduced in the last chapter [82]. Although we have shown that the base network architecture is effective for various degradation factors, the internal components of the networks need to be changed for different degradation factors. The improvements to the DuRNs are two folds. One is an improved attention mechanism. We have employed the squeeze-and-excitation (SE) mechanism in two of the proposed DuRBs (DuRB-S and DuRB-US). Since the SE mechanism was originally proposed for object classification task, the SE mechanism has been successfully applied to various tasks such as super-resolution (SR) [157], single-view depth estimation [53], etc. Its concept is to use global average pooling of activations in each individual channel of a layer to generate channel-wise attention weights on its layer activations. We extend it to additionally use amplitude of spatial derivatives of activation (i.e., $|I_x^{(c)}|$ and $|I_y^{(c)}|$) to compute attention weights. The other extension is a new design of basic block, in which the two operations employed in DuRB-U and -US are fused; we call it as DuRB-M. We show through experiments that these two extensions make it possible for a single network to learn multiple image restoration tasks, updating the current state-of-the-art for some of them.

3.2 Pioneering Works

Single Net for Multiple Degradation Factors A widely considered formulation is to train and use *different networks* with the same architectural design on multiple restoration tasks, such as Kligvasser *et al.*[63] for noise removal, rain-streak removal and Super-Resolution (SR); and Zhang *et al.*[158] for noise, mosaic, JPEG-compression noise removal and SR; and the DuRN proposed in Chapter 2 of this thesis. The approach proposed in this chapter differs from the above in that we attempt to train the *same network* on multiple different restoration tasks, which is also called multi-task learning.

Another more general formulation of image restoration is to restore the clean image from a degraded input with unknown combination of several degradation factors. There are several studies on this formulation [150, 121, 155, 3, 141, 40]. However, their performance tend to be (often quite) inferior to the state-of-the-art method designed for each individual restoration task. It is noteworthy that some existing studies such as [155, 141, 40] address a small number (specifically, 2) of similar/related factors, theirs' performance is comparatively close to the state-of-the-art performance. However, the main differences between the existing studies and the proposed approach is, the existing studies deal with only tasks that are inherently similar (e.g., deblurring and super-resolution) or closely related with each other (e.g., denoising vs.

deblurring/decompression), and thus the number of tasks is limited to two. On the other hand, the approach proposed in this chapter considers up to four tasks, which are diverse in the sense that they appear to be dissimilar on the surface level. In addition, the proposed approach outperforms the state-of-the-art methods for some factors. From a methodological point of view, it is fair to consider this study to be the first work that shows multi-task learning (MTL) of multiple diverse tasks of image restoration is feasible.

Multi-task Learning It is well known that multi-task learning [15] of deep networks is effective for many computer vision tasks; [15, 83, 61, 143, 43] to name a few. To enable MTL to work, there should arguably be some relation among the tasks jointly learned; in other words, there should be overlaps among the representations to be learned for those tasks. Combinations of tasks in the successful MTL examples include scene recognition and object recognition [43]; depth estimation and scene parsing [135]; facial expression recognition and landmark detection [50]; vision and language [95] etc.. However, it remains unknown if the same holds true for image restoration tasks. Although there should be some similarity among them, different types of degradations seem to be somewhat orthogonal with each other. In fact, the aforementioned studies on restoration from combined degradations attempt to deal with different degradation factors by adaptively selecting different networks [150] or different operations [121] depending on the degradation factors in input images.

Attention Mechanism Attention mechanisms have been developed and employed to solve various computer vision problems [131, 73, 4, 52, 121] . Hu *et al.*proposed a squeeze-and-excitation (SE) block, which produces and applies channel-wise weights on an input tensor [52]. This block has been successfully applied to various tasks such as classification [52], super resolution [157] and single-view depth estimation [53]. A number of later studies [77, 54, 133, 51, 35] aim at improving the SE-block. Woo *et al.*propose to use channel-wise and spatial attention weights [133]. Hu *et al.*study how to efficiently combine a SE-block and a ResNet module [54]. Gao *et al.*[35] propose to use correlation of activations of each pair of channels to generate attention. Hu *et al.*improve SE-block by replacing global average pooling with a pooling operation with trainable parameters [51].

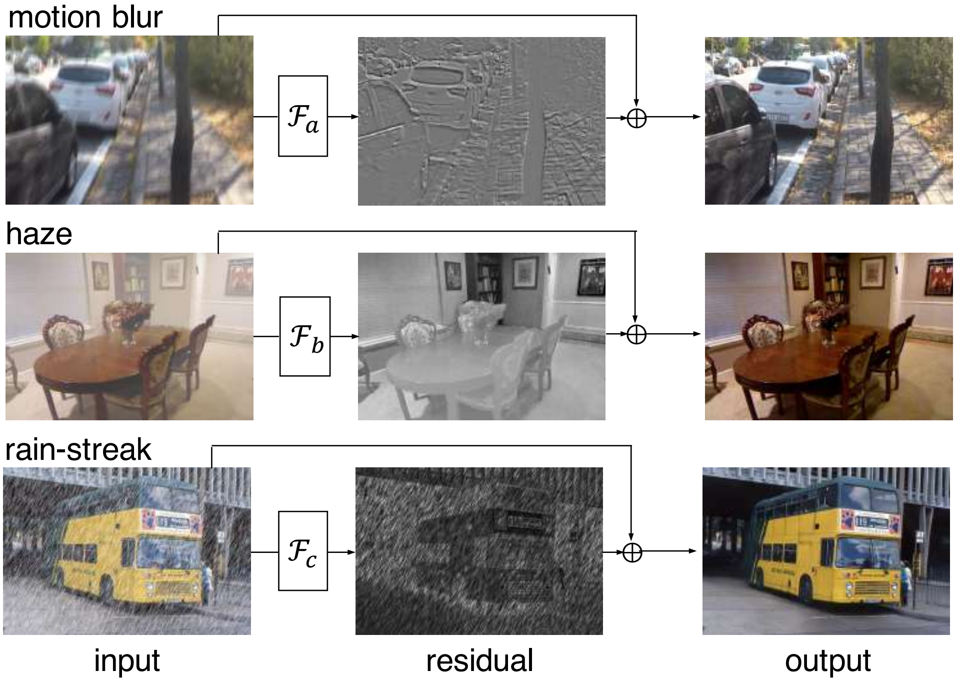


Figure 3-2: $\mathcal{F}_a$, $\mathcal{F}_b$ and $\mathcal{F}_c$ denotes a CNN trained for motion blur removal, haze removal and rain-streak removal. The images in the second column are normalized for better visibility.

3.3 Problem Formulation

Traditionally, the problem of image restoration is formulated as an inverse problem, where the forward process of image formation is modeled and utilized. For example, an image suffering from non-uniform motion blur is modeled [66] as

$$B = K(M) \otimes I + n, \tag{3.1}$$

where B is the blurred image; I is the sharp image; $K(M)$ is the blur kernel determined by motion field M; and n is noise. The image of a scene with haze degradation is modeled [151] as

$$H = It + A(1 - t), \tag{3.2}$$

where H is the hazy image of a scene; I is its clean image; t is a transmission map; and A is global atmospheric light. An image having rain-streak degradation is modeled [75] as

$$O = I + \sum_{i=1}^{N} S_i, \tag{3.3}$$

where O, I and S_i's denotes an image with rain-streak, the clean image, and different rain-streak effects. The case of $N > 1$ can consider an accumulation of multiple effects. Using the models, the problem is formulated as minimization of an objective function measuring the difference between the input degraded image and its model given as above. It is minimized with respect to the unknown clean image along with some other unknowns (*e.g.*, blur kernels etc..). It is also a common practice to incorporate natural image prior as a regularizer in the the minimization. Such image prior is shared among image restoration for different types of degradation. On the other hand, the recent trend is to use CNNs to directly predict the clean image from its degraded version. This approach does not use any image prior explicitly. A successful method of applying a CNN to the tasks [66, 34, 82] is to make it predict a compensating image such that adding it to the input image will yield a clean image, i.e., the difference from the input image to the true clean image. This method can be applied to any image degradation type. Figure 3-2 illustrates a few examples. This is formally written by

$$I = \mathcal{F}_\rho(x_\rho) + x_\rho, \tag{3.4}$$

where I is the clean image to predict; $\mathcal{F}_\rho$ is a CNN designed and trained for degradation type ρ; and x_ρ is an input image having degradation type ρ. My goal is to develop a universal network that can deal with various degradation types. Although the ultimate goal will be a monolithic CNN that has a single input and output, it is very hard to design and train such a CNN that can achieve high performance. Taking the above formulation of predicting the compensating component into account, we instead consider networks having multi-heads $g_1, \ldots$ for different degradation types, as shown in Fig. 3-1:

$$I = g_\rho \circ f(x_\rho) + x_\rho, \tag{3.5}$$

where g_ρ is a head, which we call a decoder, for degradation type ρ and f is an encoder shared by all the degradation types.

3.4 Universal Networks for Image Restoration

In this section, we describe the design of networks that are applicable to different types of degradation. We make two improvements to the dual residual networks introduced in the last chapter, intending to enhance its representation capacity. One is an improvement of the attention mechanism and the other is a new design of the structure of base blocks. These two improvements will be explained in turn, followed by the description of the overall network architecture. For decoder g_ρ, we use a stack of PixelShuffle [117] modules with convolutional operations.

3.4.1 Improved Attention Mechanism

An attention mechanism is employed in the dual residual networks. It is the channel-wise attention that was originally developed for object recognition in the study of squeeze-and-excitation (SE) networks [52], and has been widely used for many other tasks. A SE-block computes and applies attention weights on the channels of the input feature map. To determine the weight on each channel, it computes the averages of activation values of channels; then, they are converted by two fully-connected layers with ReLU and sigmoid activaton functions to generate channel-wise weights. The aggregation of activation values is equivalent to global average pooling. We enhance this attention mechanism by incorporating a different aggregation method of channel activation. The idea is to use different statistics of channel activation values in addition to their averages. For this, we choose to use (absolute) spatial derivatives of channel activation values. More specifically, denoting an activation value at spatial position (i, j) of channel c by $y_{c,i,j}$, we calculate

$$v_c = \frac{1}{N}\sum_{i,j}|y_{c,i+1,j} - y_{c,i,j}|^\beta + |y_{c,i,j+1} - y_{c,i,j}|^\beta, \tag{3.6}$$

where (i, j) denotes a spatial position of channel c, and β is a scalar to enhance derivative values. This is also known as the total variation [107], which has been used as a regularization term for various image processing tasks; a notable example is the classical image denoising, where the

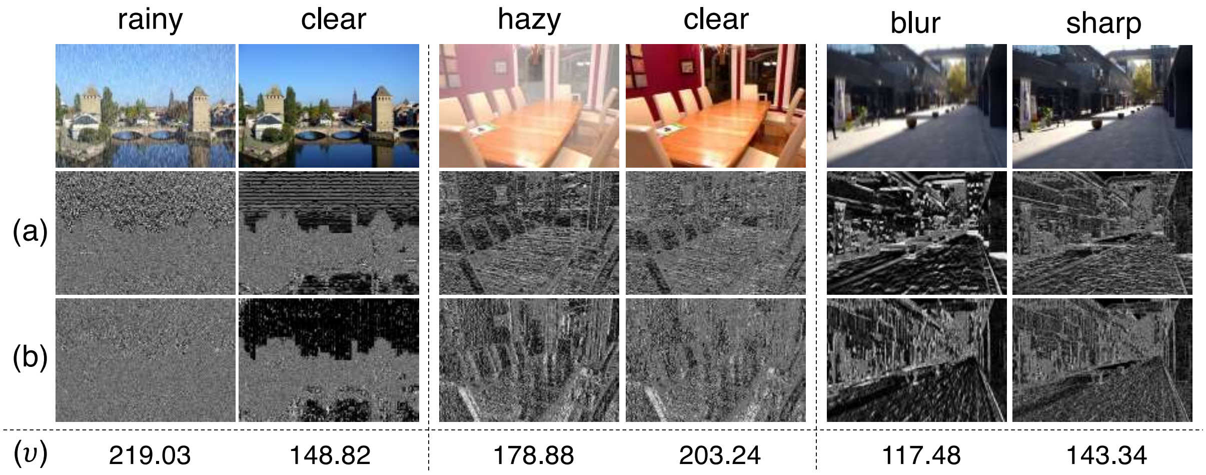


Figure 3-3: Absolute spatial derivatives of images in (a) vertical and (b) horizontal directions. (The values in the three color channels are summed together.) The values in the bottom show v_c of (3.6).

total variation helps to obtain a smoother solution while preserving edges. Figure 3-3 shows how the absolute spatial derivatives behaves for different inputs using input images (instead of intermediate layer features) as examples. It is observed that they provide different responses between clean and degraded images of the same scenes. Figure 3-4 illustrates the proposed attention mechanism. We compute the global average and the total variation of activation values of each channel and input it to the same pipeline as the SE-block to generate attention weights over the channels. This mechanism is built into a ResNet module, as shown in Fig. 3-5. The effectiveness of this design will be shown though experiments including ablation tests.

3.4.2 Improved Design of a Dual Residual Block

The design of the dual residual networks aims at making maximum use of paired operations that are believed to fit for image restoration tasks. The choice of the paired operations is arbitrary and four choices are suggested depending on the type of degradations. We pay attention on the two of them, in both of which the first operation is up-convolution. Specifically, one is the pair of up-convolution (i.e., up-sampling followed by convolution) and simple convolution. The block employing the pair is named DuRB-U and applied to motion blur removal. (See the

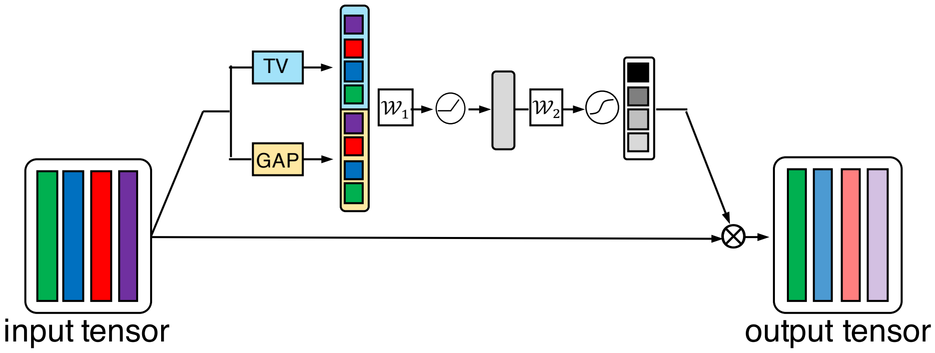


Figure 3-4: The proposed attention mechanism improving the SE block. It generates channel-wise attention weights by global average pooling (the same as the standard SE block) and total variation (TV) of each channel activation values.

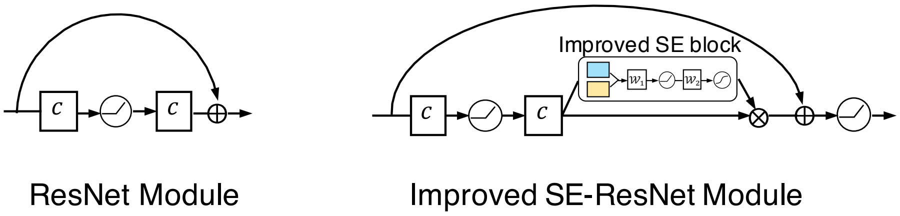


Figure 3-5: The improved SE-ResNet Module, which incorporates the improved attention mechanism into a ResNet module.

upper panel of Fig. 3-6.) The other is the pair of up-sampling followed by convolution and a SE-block. The block is named DuRB-US and applied to haze removal. Towards development of universal networks that can deal with motion blur, haze, and even more, we propose a new design of the block structure, which we call DuRB-M. The idea is to integrate the above two designs (i.e., DuRB-U and -US). To be specific, while keeping the same up-convolution for the first operation, we employ parallel computation of the second operations of DuRB-U and -US, i.e., convolution and a SE-block, for the second operation of the new block design; see the lower panel of Fig. 3-6. The output maps of the two operations are merged by concatenation in the channel dimension, followed by 3×3 convolution to adjust the number of channels. We also replace the ResNet module in the original DuRB structure with the aforementioned improved SE-ResNet module with the enhanced attention mechanism, as shown in Fig. 3-5.

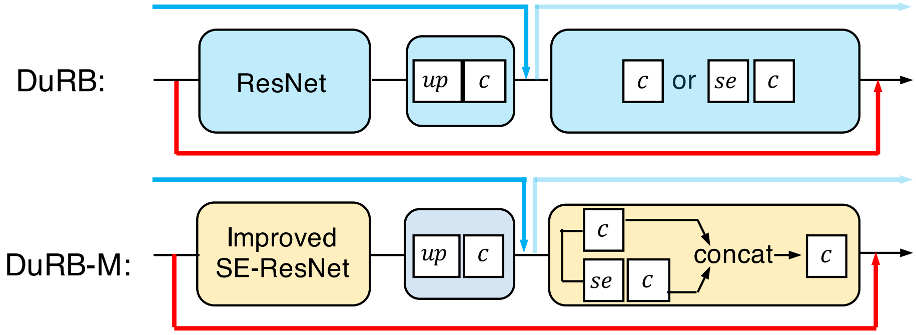


Figure 3-6: The proposed basic block (DuRB-M) used for building our network.

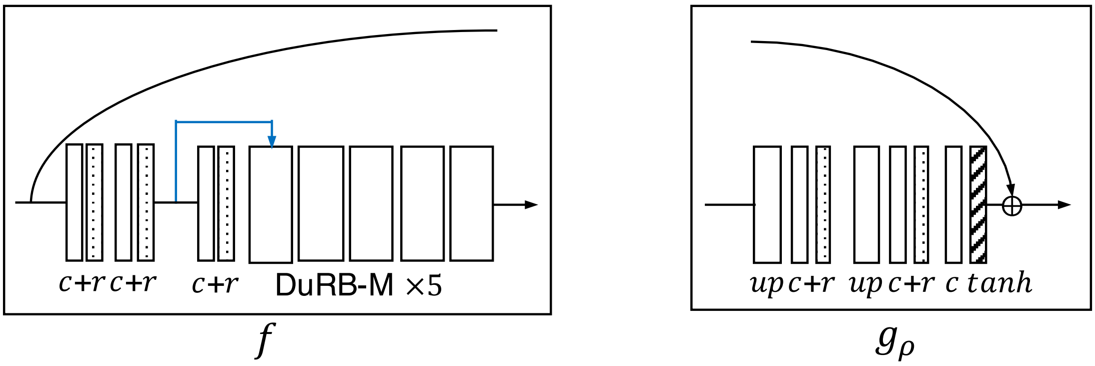


Figure 3-7: Architecture of the encoder f and the task-specific decoder g_ρ for the task ρ; c, r and $tanh$ denotes a convolutional layer, a ReLU layer and a hyperbolic tangent function, respectively; up means conv. + PixelShuffle [117]. The encoder f has 68 weight layers (each DuRB-M has 13 weight layers) and a decoder g_ρ has 5 weight layers.

3.4.3 Overall Design of the Universal Network

As mentioned earlier, the proposed network consists of a shared encoder and multiple decoders. Figure 3-7 shows the overall design. To train it on T tasks, we use T decorders $g_1, \ldots, g_T$. Each decoder g_ρ ($\rho = 1, \ldots, T$) starts with two sets of up-sampling plus convolution (implemented by PixelShuffle [117]) and ReLU in this order, followed by convolution with a hyperbolic tangent activation function. All the convolution layer of the decoder employs 3×3 kernels. The number of its channels is 96 for the first two conv. layers and 48 for the last one. We use the same design for all the decoders for different tasks. As they have learnable weights in the convolution layers, they work differently after training. The encoder f starts with three convolution layers with ReLU activation, followed by a stack of the proposed DuRB-M's. The 2nd and 3rd convolution layers use stride $= 2$, and thus the input image is down-scaled to 1/4 of the original size when inputted to the first DuRB-M. Note that there is a skip connection from the output from the second ReLU to the first DuRB-M. Other details of the encoder are given in the Appendix.

3.5 Experimental Results

We conduct experiments to evaluate the proposed method. We choose three tasks, i.e., motion blur removal, haze removal, and rain-streak removal, for the main experiments (i.e., detailed architectural design search, ablation study, etc.); we additionally use JPEG compression noise removal for performance evaluation.

3.5.1 Experimental Configuration

Datasets

In the experiments, we choose the dataset(s) for each task that is the most widely used in recent studies. We use the GoPro dataset [93] for motion blur removal. It consists of 2,013 and 1,111 non-overlapped training (GoPro-train) and test (GoPro-test) pairs of blurred and sharp images, respectively. Each blurred image in this dataset is synthesized by averaging a number of continual frames in a 720p-video taken by a go-pro camera. We use the RESIDE dataset [71] for haze removal, which consists of 13,990 samples of indoor scenes and a few test subsets. Following [82] and [103], we use a subset SOTS (Synthetic Objective Testing Set) that contains 500 indoor scene samples for evaluation. For both training and test subsets of RESIDE, synthetic hazy effects are made using (3.2). We use the DID-MDN dataset [152] for rain-streak removal. It consists of 12,000 training pairs and 1,200 test pairs of clear image and synthetic rainy image. Rain-streak effects for an image in this dataset are made using Photoshop. As for JPEG compression noise removal, we use the training subset (800 images) of the DIV2K dataset [127] and LIVE1 dataset (29 images) for training and testing the proposed networks, following the study of the state-of-the-art method [158]. An additional setting we made for our experiment is that we re-sized the original DIV2K images (larger than 2000×1000 for most of them) to their half for training out network for computational efficiency.

Training on Multiple Tasks

We jointly train the proposed network on multiple tasks in the following way. We split the training into a series of cycles, in each of which the network is trained on a combination of all the tasks. To be specific, each cycle contains one or more randomly chosen minibatches per a single task. Considering that the loss decreases at a different speed for different tasks, we choose the number of minibatches in one cycle, specifically, one for haze removal, one for rain-streak removal, and three for motion blur removal. The minibatches are randomly chosen from the training split of each dataset and packed in a random order in a row for the cycle. We then iterate this cycle until convergence. Each input image in a batch is obtained by randomly cropping a 256×256 region from an original training image. The learning rate is set to 0.0001 at beginning, and is divided by 10 when the training loss stops decreasing. For loss functions, we use a weighted sum of SSIM and l_1 loss, specifically, $1.1 \times$ SSIM $+ 0.75 \times l_1$ for training all the CNNs. We use Adam [62] optimizer with $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 1 \times 10^{-8}$. We use PyTorch [99] to conduct all the experiments.

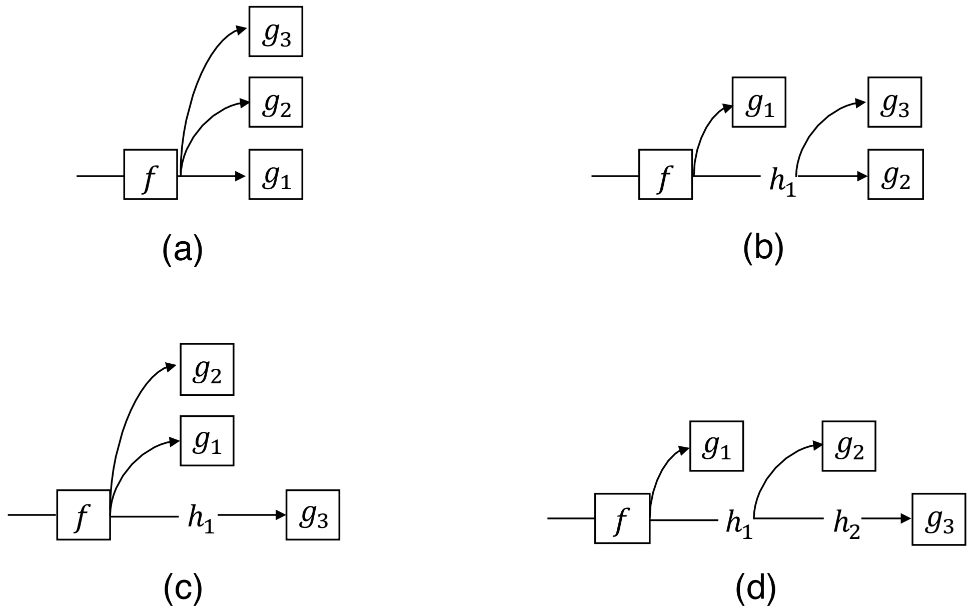


Figure 3-8: Experimental designs of an overall network consisting of the encoder f and task-specific decorders g_1, g_2, and g_3; h is a DuRB-M block.

3.5.2 Extended Design of the Entire Network

We have found in our preliminary experiments that while the architecture of a shared encoder followed by multiple task-specific decorders (illustrated in Fig. 3-8 (a)) shows strong performance, more general architectures achieves even better performance. By general architectures, we mean those having extended DuRB-M blocks on top of the shared encoder, to which some of the task-specific decorders are connected, as shown in Fig. 3-8 (b)-(d). To explore what structure shows good performance, we consider the four architectural designs shown in Fig. 3-8. When we have three target tasks, there are thirteen designs of assigning them to the four architectures, which are listed in Table 3.1. In the table, we use B, H and R to denote motion blur removal, haze removal, and rain-streak removal, respectively, and use “$\rightarrow$” to denote one DuRB-M block. In the table we report the performance of each of the thirteen designs in terms of accuracy (PSNR/SSIM) averaged over the three tasks. In this experiment, we trained each of our thirteen networks for 1×10^5 iterations under the same experimental setting. It is seen that R$\rightarrow$B$\rightarrow$H performs the best. This indicates differences among the three tasks cannot be fully absorbed by the single encoder, and implies that there is a hierarchy that is probably associated

Table 3.1: Comparison of performance of thirteen different designs of the network for three tasks (B: motion-blur removal, H: haze removal, and R: rain-streak removal). The values (PSRN/SSIM are averaged accuracy over the three tasks.

Alignment	PSNR / SSIM
B→H→R	30.61 / 0.9244
B→R→H	30.42 / 0.9227
H→B→R	30.57 / 0.9244
H→R→B	30.34 / 0.9219
R→B→H	**30.79 / 0.9246**
R→H→B	30.29 / 0.9201

Alignment	PSNR / SSIM
BH→R	30.24 / 0.9217
B→HR	30.36 / 0.9216
HR→B	30.02 / 0.9189
H→RB	30.41 / 0.9236
RB→H	30.47 / 0.9219
R→BH	30.38 / 0.9228
RBH	30.32 / 0.9206

with their difference in difficulty in this order.

3.5.3 Comparison with the State-of-the-art

We compare the proposed method with the state-of-the-art methods for different tasks. Table 3.2 shows the results. We choose the best four published methods (ranked by PSNR) for each task. "Ours(3)" indicate our method trained on the three tasks. We report here the accuracy values obtained for the best architecture found in the experiment explained above. It is observed that the proposed method outperforms others for motion blur removal and haze removal and achieves comparable performance to the previous methods for other tasks.

Table 3.2 also shows the results ("Ours(4)") obtained by simultaneously training our network on four tasks, i.e., the three tasks plus JPEG compression noise removal. It is seen that the addition of this task contributes to further improvements on haze removal and rain-streak removal. In the experiment, we search for a good design for the four tasks; to do this with a modest computational cost, we considered only combinations of inserting either a new decorder or a new decoder with a DuRB-M to the above three-task network. The best performer is the one with additional DuRB-M inserted in between B and H, i.e., R→B→J→H, where J is the JPEG compression noise removal. The results with "Ours(4)" in Table .3.2 is obtained by this design. A few examples of the output images for the four tasks are shown in Fig. 3-10 and Fig. 3-11.

Table 3.2: Comparison of state-of-the-art methods in terms of accuracy (PSNR/SSIM) for four different tasks. DuRN-M(3) and DuRN-M(4) are the proposed network trained on the three and four tasks, respectively. The best one is in bold and the second is with underline. The value with the superscript * means a result unable to replicate.

	motion blur removal		haze removal	
	Xu *et al.*[140]	24.6 / 0.84	Li *et al.*[70]	19.06 / 0.85
	Kupyn *et al.*[66]	27.2 / 0.95*	Cai *et al.*[14]	21.14 / 0.85
	Nah *et al.*[93]	28.3 / **0.92**	Ren *et al.*[103]	22.30 / 0.88
	Liu *et al.*[82]	29.9 / 0.91	Liu *et al.*[82]	32.12 / **0.98**
DuRN-M(3)	**30.18 / 0.92**		33.90 / **0.98**	
DuRN-M(4)	30.17 / **0.92**		**34.16 / 0.98**	

rain-streak removal		JPEG artifacts removal (q=10)		
Fu *et al.*[34]	30.92 / 0.89	Dong *et al.*[25]	28.98 / 0.82	
Li *et al.*[75]	32.48 / 0.91	Chen *et al.*[19]	29.15 / 0.81	
Li *et al.*[72]	33.16 / 0.92	Zhang *et al.*[153]	29.19 / 0.81	
Liu *et al.*[82]	**33.21 / 0.93**	Zhang *et al.*[158]	**29.63** / 0.82	
32.76 / 0.92		-/-		DuRN-M(3)
32.87 / 0.92		28.20 / **0.83**		DuRN-M(4)

3.5.4 Ablation Study

The proposed method consists of several components. To evaluate the contribution of each component, we conducted two ablation tests.

Improved Attention Mechanism and Dual Redisual Block

In the first test, we evaluate the contributions of the three components, i) the channel-wise total variation and ii) the channel-wise average pooling, both of which are used for attention computation, and iii) the improved design of dual residual block that employ fused operations. Table 3.3 shows the results on three different tasks when performing multi-task learning on the same three tasks. It is first seen that the use of all the three components yields the maximum accuracy for each task. It is also observed that each component has a certain amount of positive impact on the resulting accuracy, although it differs for different degradation types.

Impact of Multi-Task Learning

To evaluate the effectiveness of multi-task learning, we train the proposed network (the best design for the three tasks explained in Sec. 3.5.2) on each of the three tasks separately. In this experiment, we simply neglect other g's than the one for the target task. The results are shown in the bottom of Table 3.3. It is seen that multi-task learning improves performance on motion blur removal and haze removal by a good margin, while it decreases the performance on rain-streak removal.

Table 3.3: Results of an ablation test with different components and employment of multi-task learning.

TV	GAP	Fusion	motion blur removal	haze removal	rain-streak removal
✗	✗	✗	28.25 / 0.8724	26.58 / 0.9646	31.32 / 0.8976
✓	✗	✗	28.49 / 0.8809	29.15 / 0.9699	31.99 / 0.9003
✗	✓	✗	28.11 / 0.8703	29.66 / 0.9721	32.01 / 0.9015
✓	✓	✗	28.51 / 0.8811	29.06 / 0.9719	32.03 / 0.9014
✓	✓	✓	**28.91 / 0.8911**	**31.32 / 0.9778**	**32.15 / 0.9048**
motion blur			27.87 / 0.8653	-/-	-/-
haze			-/-	28.58 / 0.9685	-/-
rain-streak			-/-	-/-	32.60 / 0.9139

3.5.5 Visualization of Internal Features

To explore how different types of degradation are learned and represented inside our network, we visualize the internal activations of the best three-task model (i.e., R→B→H of Table 3.1). We input each sample of the test splits from the datasets of these tasks to the trained network. We then apply t-SNE [129] to the set of activations of selected intermediate layers to map them to two-dimensional space. Figure 3-9 shows the results. It is observed that the images having different degradation factors are quickly disentangled as they propagate through the layers and are clearly separated at the final output of the encoder. This demonstrates that the proposed network is able to learn different image restoration tasks with a single network. It also implies that the proposed network clearly distinguishes different types of degradations and represents them differently inside its layers. A further analysis is left for future studies.

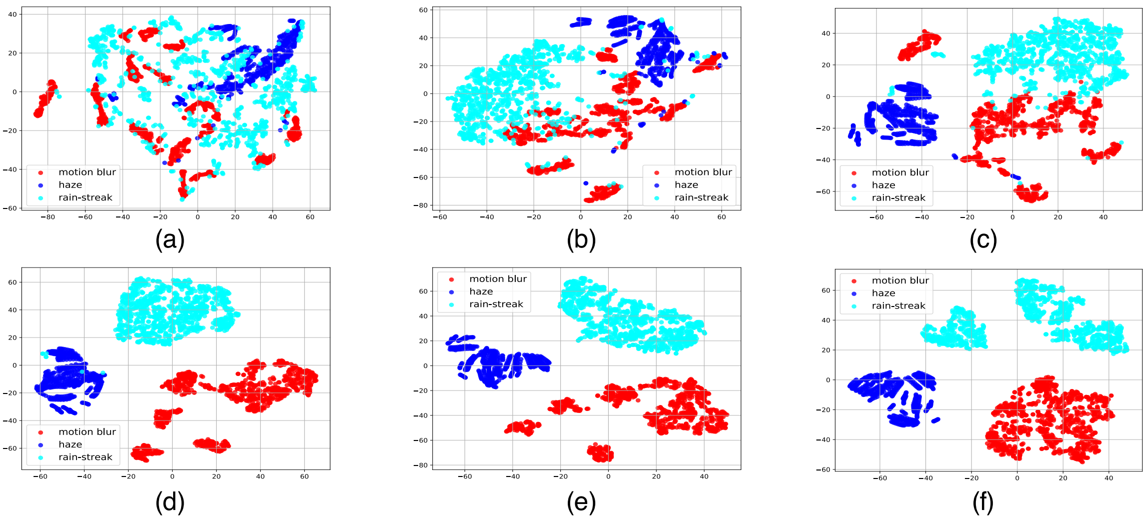


Figure 3-9: Visualization of activations of selected intermediate layers of our network trained on the three tasks. Each feature space is mapped to two-dimensional space by t-SNE [129]. The results of lower to higher layers are shown from left to right. (a) Output of the first ReLU layer. (b) The second ReLU layer. (c)-(e) Output of the first, third, and fifth DuRB-M blocks.

3.6 Summary and Conclusion

We have considered the design of a universal network applicable to multiple image restoration tasks. We have shown through experiments that a single network can learn to perform motion-blur removal, haze-removal, rain-streak removal and JPEG compression noise removal. We have proposed two extensions to a backbone design of networks called the dual residual network, which was recently proposed and shown to be effective for various image restoration tasks. Our extensions are the improved attention mechanism that considers the spatial gradients of feature maps to generate channel-wise attention weights and the basic block design named DuRB-M that performs two operations in parallel and fuses them.

Discussion related to Chapter 2: In Chapter 2, We have made a foundation, *e.g.*, a basic architecture (DuRB) of CNNs which is share-able cross multiple tasks, for developing new CNN-based algorithms for image restoration tasks. In Chapter 3, from a methodological point of view, we have shown for the first time that multi-task learning (MTL) of orthogonal tasks of image restoration is feasible. This is achieved to a meaningful level by improving the design of DuRB. It is thus not surprising that DuRN-M outperforms DuRNs on some restoration tasks. The present study also provides the first counterexample to the current research trend – (researchers) design/employ networks/formulations that are dedicated to individual degradation types. The DuRN-M (Chapter 3) should not be considered as a rival to DuRN (Chapter 2) in terms of restoration performance, but a continues development to discover new knowledge that is academically interesting.

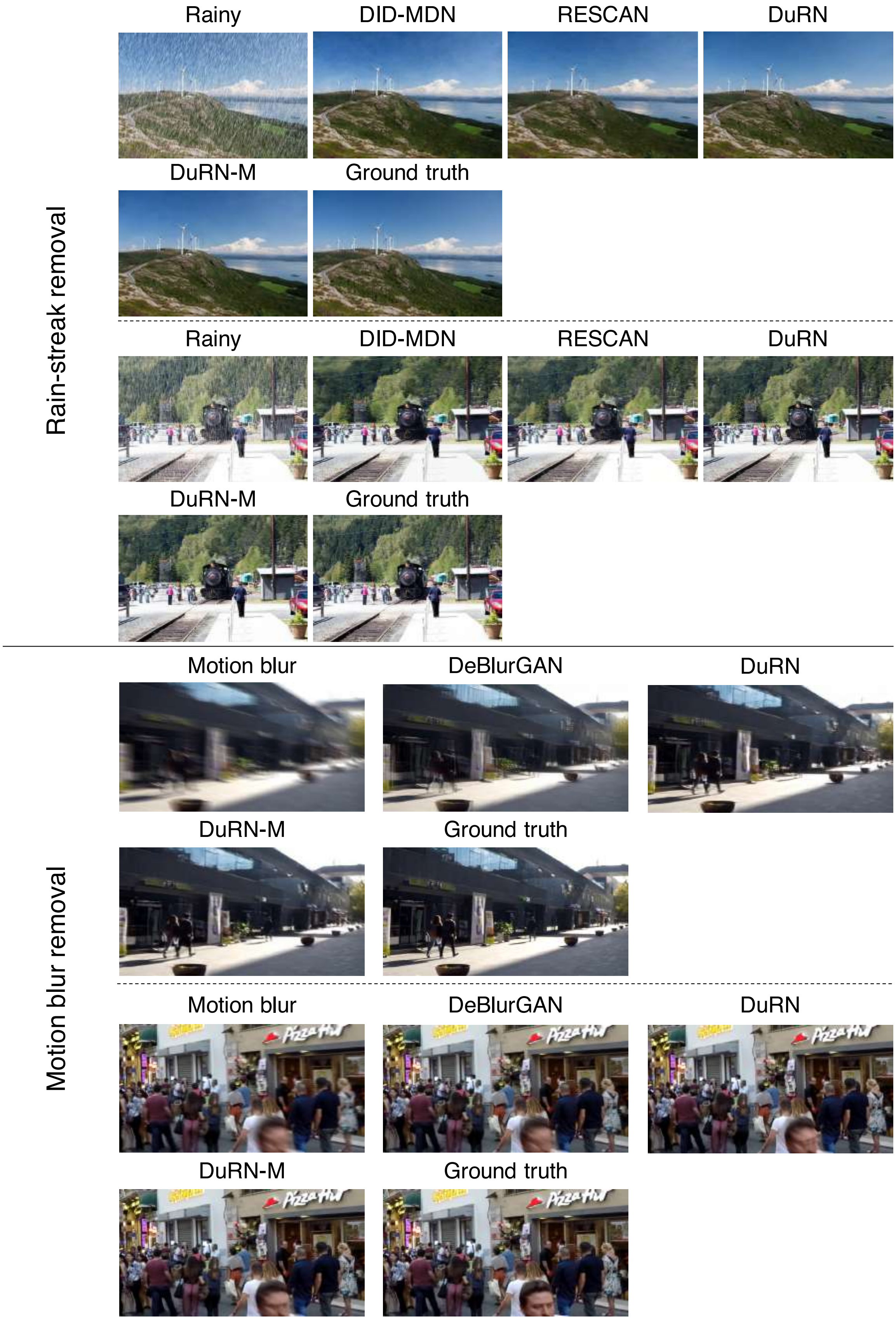


Figure 3-10: Examples of qualitative results for rain-streak removal and motion blur removal tasks.

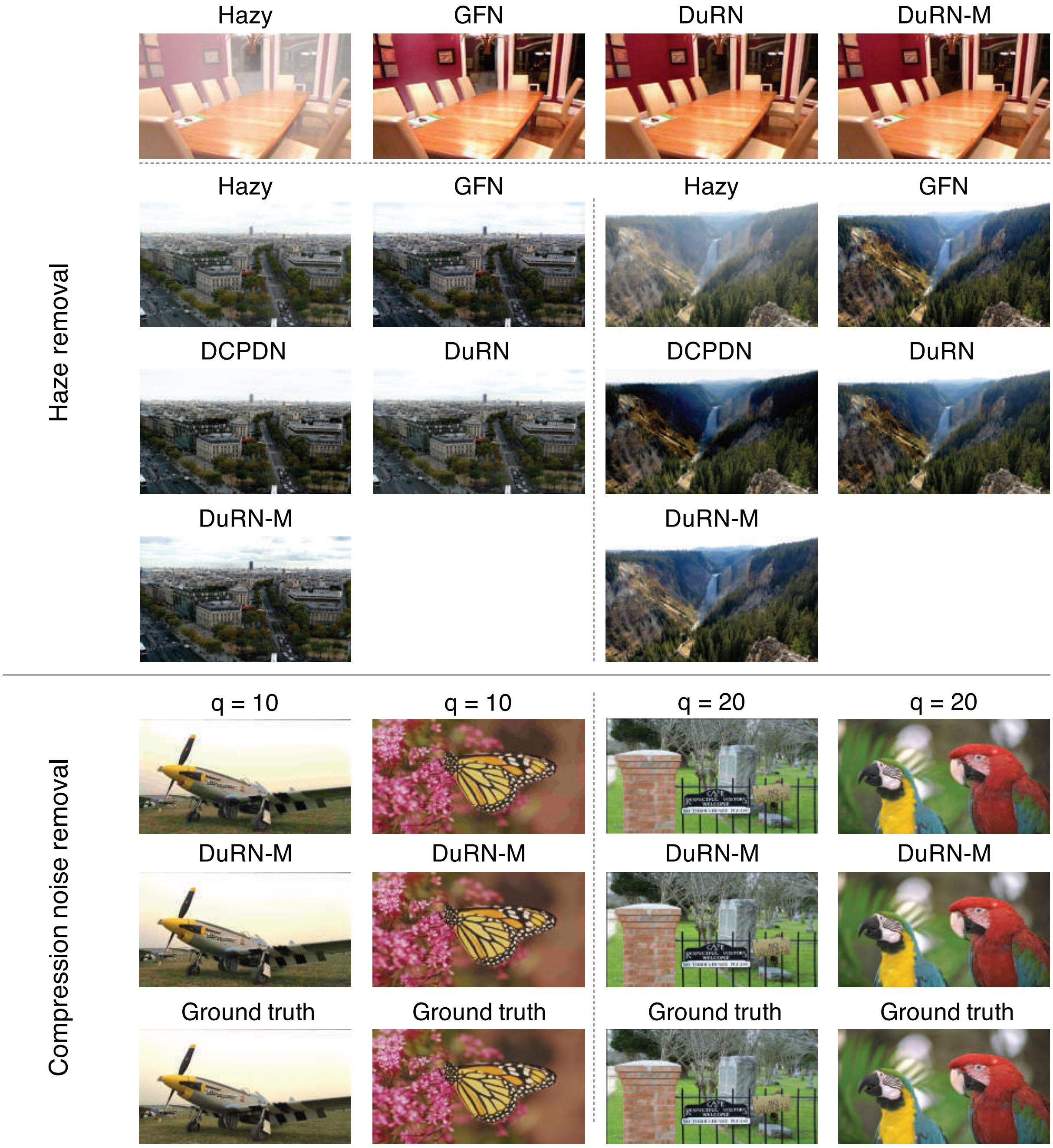


Figure 3-11: Examples of qualitative results for the haze removal and compression noise removal tasks.

Chapter 4

CNNs versus Human Vision

4.1 Introduction

The problem of classification, *i.e.*, classifying an image to one of a few given classes, has been studied for decades in computer vision and pattern recognition. In recent years, deep learning has been developed for this problem, which has led to significant improvement of performance. A deep Convolutional Neural Network (CNN) constructed with residual connections [48] achieved human's level accuracy on classifying 10,000 large scale images to 1,000 classes [108] in 2016. In the following year, a CNN [55] consisting of densely connected layers further improved the performance on the same task. Beside this, CNNs have been applied to many other tasks such as object detection and segmentation [78, 43], depth estimation [53] and image generation [41], implying their great potential for solving various problems like humans do. On the other hand, understanding how human's perception work has been drawing great attention in human vision and psychology research fields. This is a challenging problem for the reason that human's perception is hard to materialize and thus difficult to model. A promising approach is employing a CNN as an agent that approximates the computation of human's vision system. Conducting manipulation and analyses on the CNN, we intend to investigate how human's perception work for various problems. In this chapter, we focus on the task of material classification. We are specifically interested in comparing perturbation tolerance of CNNs and humans on this task. We first demonstrate a fact that CNNs that classify standard images of

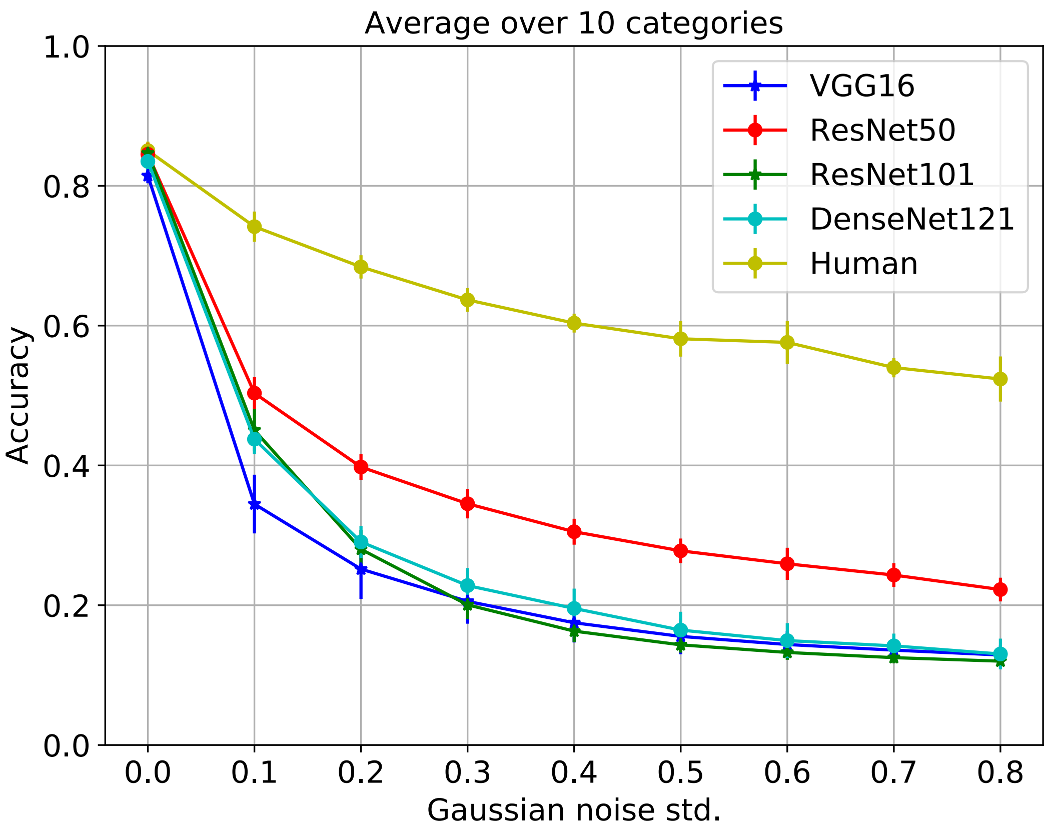


Figure 4-1: Performance comparison on FMD-test [114].

materials as accurately as humans do, has worse tolerance to Gaussion noise than humans. This implies a difference between human's perception and its computational model established by the CNNs. Studying approaches that improve CNNs' classification accuracy for noisy images, we intends to explain why human vision is robust to perturbations such as Gaussian noise. We consider two approaches in this chapter, i) employ an image restoration CNN (namely, *cleaner*), specifically, a CNN that is trained for Gaussian noise removal, before the CNN (namely, *classifier*) employed for classifying standard images. This strategy in fact establishes a sequential process, that first "clean" noisy images then classify them. This is made based on the hypotheses that humans might infer the clear version of a noisy image so that to classify it correctly. ii) Train a classifier using standard images with a number of noisy images. A hypotheses behind this is, humans might have experienced classifying images with similar noise in their daily life, which enable them to classify the noisy test images correctly. We investigate how likely are the two approaches the pattern of human's, by comparing classification accuracy and behavior (represented by confusion matrices) of theirs with humans'. It is seen by the results that both approaches improved classification accuracy up to humans' level.

4.2 Pioneering Studies

Material Recognition Early studies on reflectance properties (*e.g.*albedo, color) have been considered related to humans material recognition [115]. Bloj *et al.*[10] showed that human's color perception is effected by light reflection. Boyaci *et al.*[11] proposed that surface orientation perceived by human observers is partially discount in estimating surface albedo. Nishida *et al.*[96] demonstrated that luminance histogram is an important factor for surface reflectance perception for human's visual system. Robilotto and Zaidi *et al.*[104] examined whether humans veridically perceive reflectances of real objects under natural viewing conditions. Motoyoshi *et al.*[91] proposed that the skewness of the luminance histogram and the skewness of sub-band filter outputs of an image, are diagnostic of its surface properties (*i.e.*glossiness and albedo). Fleming *et al.*[32] found that humans use tacit knowledge about the statistics of real-world illumination to estimate surface reflectance. Fleming *et al.*[31] studied on how humans identify translucent materials. They suggested that humans' visual system do that by gathering image statistics from key regions of a translucent object. With the researches that have been conducted in the literature of psychophysics and human vision, Sharan *et al.*[115] studied material recognition on a combination of human vision and computer vision. They proposed i) a dataset, Flicker Material Database (FMD), that consists of 1,000 material images, and ii) a recognition system, which classifies material images with features found based on the studies of human material recognition. The FMD has been used in many studies related to material recognition [112, 156, 80, 81, 79] in computer vision, in the following years. Zhang *et al.*[156] proposed an approach for selectively integrating features learned by CNNs for improving material recognition accuracy. Liu *et al.*[81] demonstrated a poor perturbation tolerance of CNNs compared to human's on material recognition. Liu *et al.*[79] studied individual differences of humans on recognition of material attributes such as *glossiness*, *transparency* and *beauty*. On the other hand, many contributions have been made in recent years, for understanding materials and their properties. Sawayama *et al.*[109] showed increasing overall chromatic saturation, darkness and glossiness of an image enhances the perception of wetness, and developed an effective wetness enhancing transformation based on that. Sharan *et al.*[116] investigated how come humans can fast and accurately classify materials and suggested that it is a distinct, basic ability of human's visual system. Sawayama *et al.*[110] proposed that material and shape perception of human's mainly relies on intensity gradient magnitude and intensity gradient order.

Development of CNNs' architectures A pioneering study is conducted by LeCun *et al.*[68] in 1998, where a shallow network (LeNet-5) consisting of two convolutional layers and three fully-connected (fc) layers was proposed for classifying handwritten digits. This study established a standard architecture, *i.e.*a stack of convolutional layers followed by one or more fc layers, for later studies on CNNs' architecture. From 2012, various CNNs have been developed for solving classification problems. Krizhevsky *et al.*[65] proposed an eight layers CNN (AlexNet) for this purpose. Simonyan *et al.*[118] developed a set of deeper versions (VGG-Nets) of AlexNet for a better classification accuracy. He *et al.*[48] proposed a simple yet effective layer connection style, *residual connection*, that adds the input x of a computation $\mathcal{F}$ to its output, for classifying images. They designed a modular block (ResNet block) implementing this connection style, and built very deep CNNs (ResNets) by stacking the blocks repeatedly. Huang *et al.*[55] proposed to build CNNs in a style different to the ResNets. In their proposed connection style, each convolutional layer takes as input the feature maps of all preceding layers. They designed a modular block (DenseBlock) to implement this style and build CNNs by repeatedly stacking DenseBlocks. Hu *et al.*[52] proposed a channel-wise weighting mechanism, *Squeeze-and-Excitation* mechanism, for selectively enhancing important features and suppress useless features for classification. Detailed explanations for it are provided in Sec. 4.4.

Studies on CNNs and Humans CNNs have been developed successfully for many tasks. It is promising to think that such a system has a good potential approximating human vision. This starts a new trend of research that goes on the combination of human vision system and CNNs, bridging one to another. Geirhos *et al.*[36] analyzed the robustness of humans and CNNs to various types of image degradation on object classification. They showed that human visual system is more robust than CNNs for most degradation types. They also showed that CNNs' performance surpasses human's only when they are tested on the distortion types the CNNs were trained on, and the strong robustness a CNN earned for one distortion type doesn't generalize to other types. Zhou *et al.*[159] studied CNNs and Humans with adversarial samples, *i.e.*, images that are altered for incorrect classification for CNNs. They showed that humans can predict to what class an adversarial sample is incorrectly classified by a CNN. Geirhos *et al.*[37] proposed that normally trained CNNs are strongly biased to using textures of an image to identify its class. They pointed out that this is in contrast to human vision in terms of behavioural evidence, and proposed to force a CNN to learn shape-based features for a human's alike classification, by distorting local textures of training images.

4.3 Noise Tolerance of CNNs and Humans

CNNs and Datasets We employ four popular CNNs for our experiments, VGG16 [118], ResNet50 [48], ResNet101 [48] and DenseNet161 [55]. These CNNs are firstly trained on the ImageNet [108] dataset[1] for object recognition, then fine-tuned on two datasets consisting of material images. Adopting this two-step training procedure, we intends to enable the CNNs to re-use the features learned for object recognition for achieving human-level performance on material recognition [80]. The two datasets for material recognition are i) Extended Flicker Material Database (EFMD) [156] which consists of 10,000 images (1000/class for 10 material classes); and ii) Flicker Material Database (FMD) [116], which consists of 100 images/class for the same 10 classes (1,000 images in total). We randomly split the 100 images in a class into 50/50 for training/test for all the classes for FMD. We denote the training/test subset of FMD by FMD-train and FMD-test. All the CNNs are trained on ImageNet, then EFMD, and then FMD-train; and tested on FMD-test.

Data Augmentation We use data augmentation for training the CNNs on the two material datasets. We employed random cropping (of size 224×224) and random horizontal flip for EFMD and FMD, and random vertical flip, color jitter and Random re-sized cropping [125] for EFMD alone. By color jitter we mean the process that randomly change the brightness,

[1] We use the pre-trained model provided by PyTorch [99]

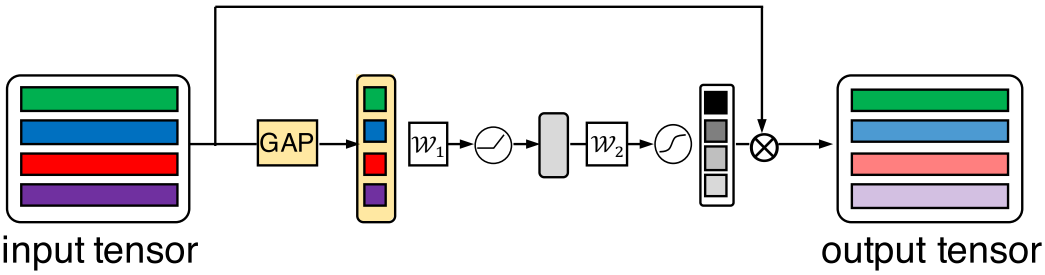


Figure 4-2: The Squeeze-and-Excitation (SE) block.

contrast and saturation of an image. By Random re-sized cropping, we mean the process on an image that first produce a crop of random size of 0.08 to 1.0 of the original size and a random aspect ratio of 3/4 to 4/3 of the original aspect ratio, then resize the crop to a fixed size which is 224×224.

Details for Training and Testing CNNs In training, we use a SGD optimizer with momentum $= 0.9$. The learning rates are set to 0.001 and 0.0001 for training the CNNs on EFMD and FMD-train. We don't change learning rate during the training. All the input images are first re-resized to 256×256, then processed by the aforementioned data augmentation. For loss function, we employ cross entropy loss. The networks are trained for 50 and 40 epochs on EFMD and FMD-train. In order to have a reliable result, we generated 10 splits of FMD-train and FMD-test with random sampling. We train and test each CNN on the 10 splits of FMD, and use the average result for our analysis.

Humans' Data We collect classification results of noise-free and noisy images for humans on FMD-test, using Amazon Mechanical Turk (AMT). Each image is viewed by five human subjects via the AMT system. The majority of their decisions is used as a human classification result. We collect human classification results for the test set of one split of FMD but all the 10 splits, unlike we do for the CNNs.

Results It is seen in Fig. 4-1 that the CNNs can classify materials as accurately as humans do on noise-free images, while the categorization accuracy dropped with the noise level more steeply for the CNNs than for humans, implying lower perturbation tolerance for the CNNs. In the following sections, we introduce two approaches for improving classification accuracy on noisy images to human's level for noisy images.

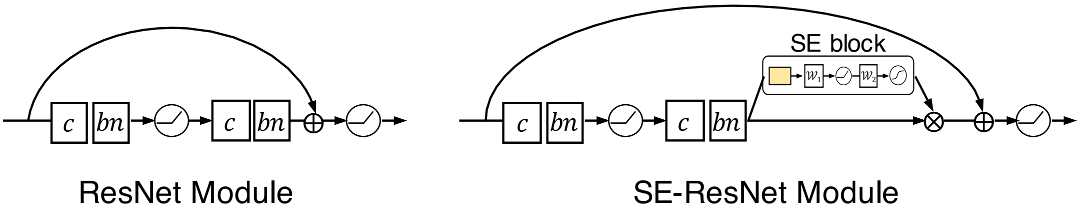


Figure 4-3: The SE-ResNet module.

4.4 Squeeze-and-Excitation Mechanism

Squeeze-and-excitation (SE) mechanism is originally proposed with deep neural networks for object recognition [52]. It is in fact a channel-wise weighting mechanism for selectively enhancing important features and suppress useless features on the internal tensors of a network. To determine the weight on each channel, it computes the averages of activation values of channels; then, they are converted by two fully-connected layers with ReLU and sigmoid activation functions to generate channel-wise weights. The aggregation of activation values is equivalent to global average pooling. We illustrate this mechanism in Fig. 4-2. Towards better leveraging the potential of a SE block, this mechanism is built into a ResNet Module [48], following previous studies such as [52, 82]. We illustrate it in Fig. 4-3 and name it as SE-ResNet module.

4.5 Squeeze-Excitation-Distillation (SED) Modular for Image Restoration

We propose a novel modular for restoring clear images from their noisy version. The modular consists of i) a SE-ResNet module that enhances important features of natural images and suppress features of noise, ii) a module consisting of two sets of [convolution (with 1×1 kernels)+batch normalization+ReLU] where the convolutional layer in the first and the second set has d and $d/2$ channels, and iii) a convolutional layer with 3×3 kernels and d channels followed by a batch normalization layer and a ReLU layer. It is noteworthy in ii), that setting less channels ($d/2$) for the second convolutional layer against to the first's (d) encourages this module to extract essential features and remove noise's features on its input tensor enhanced by i). Constructing a modular in such a way, we intend to enhance, then extract the essences of a natural image from its noisy version, and reconstruct the detailed information of the natural image using its essential features. The proposed modular is illustrated in Fig. 4-4.

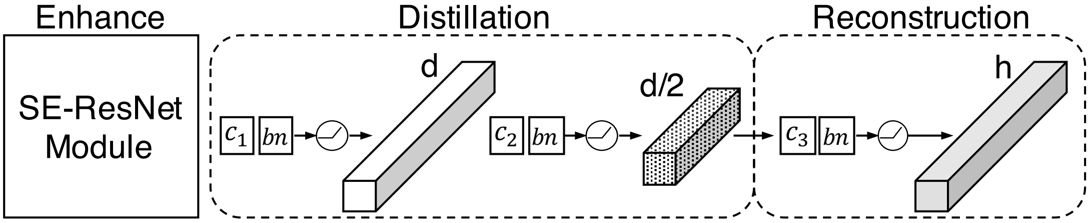


Figure 4-4: The proposed Squeeze-Excitation-Distillation (SED) modular.

4.6 Image Restoration for Improving Classification Accuracy

Image Restoration Network We construct a CNN (SEDNet) for Gaussian noise removal using the proposed SED modular. We show the architecture of the network in Fig .4-5. Batch normalization layers and ReLU layers used in the network are not illustrated in the figure for the sake of simplicity. It consists of two initial convolutional layers for down-scaling an input image into its 1/4 size, and a stack of four SED modules in the middle of the network, and a [*up*, SED, *up*] block at the last of the network. The *up* means a 1:2 up-scaling operation (by nearest-neighbor interpolation) and a convolutional operation with 3×3 kernels.

Results Visual examples of input noisy images, internal feature maps, restored images and clear images are shown in Fig 4-8. The images shown in the 2nd, 3rd and 4th columns are produced by summing up the values of internal tensors along their channels on the last SED modular of the SEDNet. The *enhance*, *distill* and *reconstruct* means that the images are computed on the output of the enhance, distillation and reconstruction module of the SED modular (see Fig. 4-4), respectively. It is seen that the SEDNet restores noisy images to a good extent.

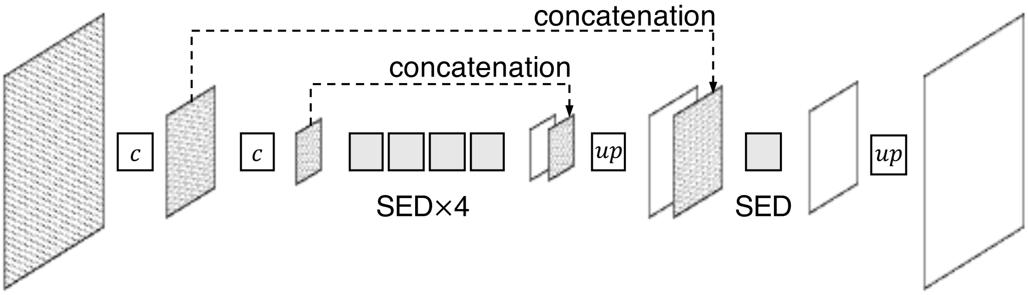


Figure 4-5: The proposed network (SEDNet) for Gaussian noise removal. A up contains a up-sampling operation by nearest and a convolutional layer.

4.7 Training CNNs with Noisy Images for Improving Their Perturbation Tolerance

It is a widely known approach that training a CNN with distorted images improves its robustness to the distortion. We adopt this approach to our training procedure introduced in Sec. 4.3. Given a training image of EFMD, we add an equal sized noise mask onto it. The noise mask consists of values randomly generated under a Gaussian distribution with $mean = 0$ and $std. = q$, q is randomly selected from $[0.1, \ldots, 0.8]$. We set the probability to 10% that adding a noise mask to an image for all the training images. The results are shown in Fig. 4-6.

4.8 Experimental Analysis

We plot classification accuracy on noise-free/noisy images for the standard CNNs, the human subjects and the two approaches (*i.e.*noise fine-tune and restoration) in Fig. 4-6. We use std.$= 0$ to represent noise-free images for all the four sub-figures. It is seen that the both approaches can improve accuracy of classifying noisy images to humans' level. In addition, we demonstrate the behaviours of CNNs' and humans' on the classification using confusion matrices. We show examples of the VGG16 in Fig. 4-7. It is seen that the both approaches managed to make the VGG16 behave like humans do.

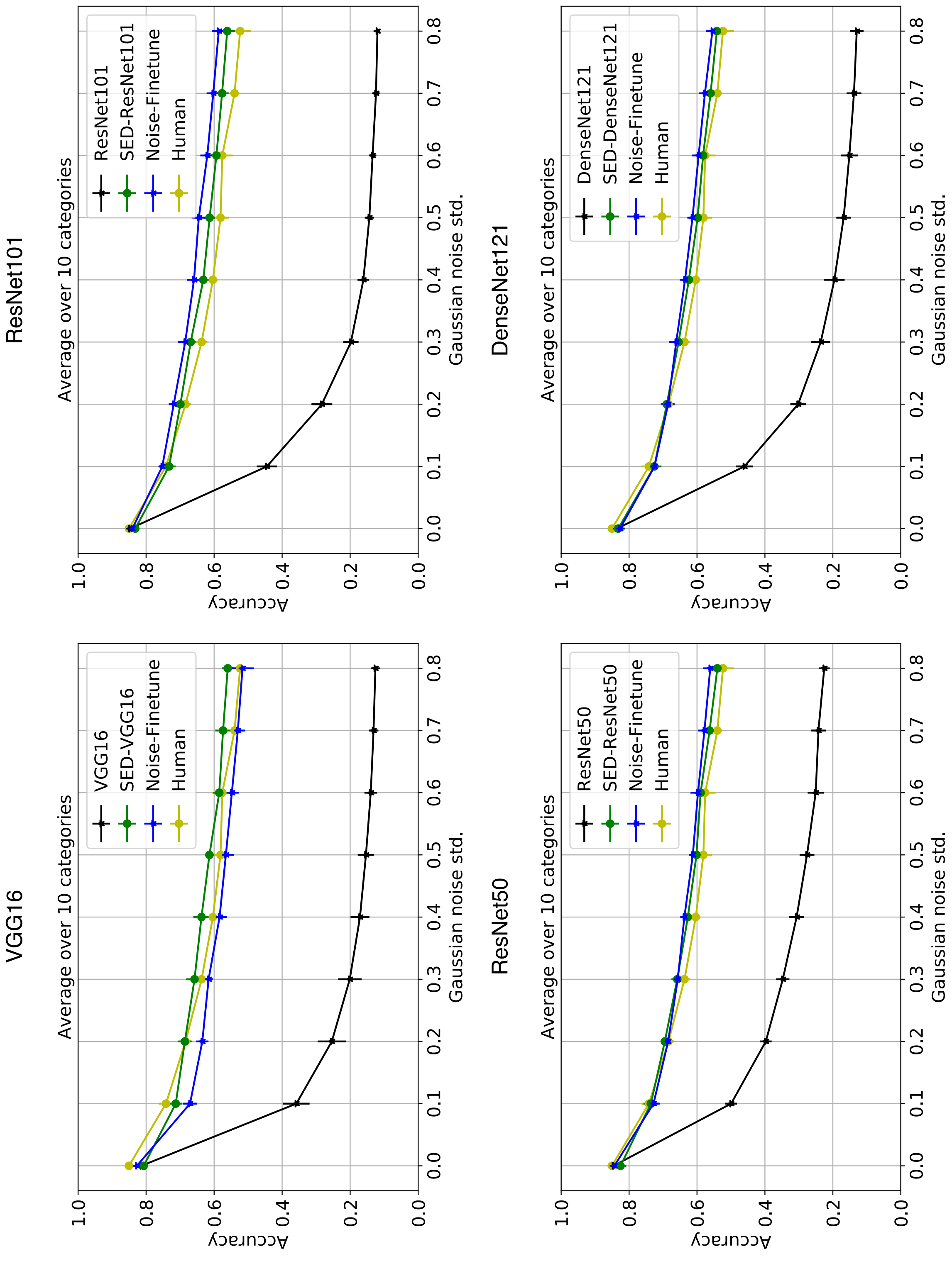


Figure 4-6: Performance comparison.

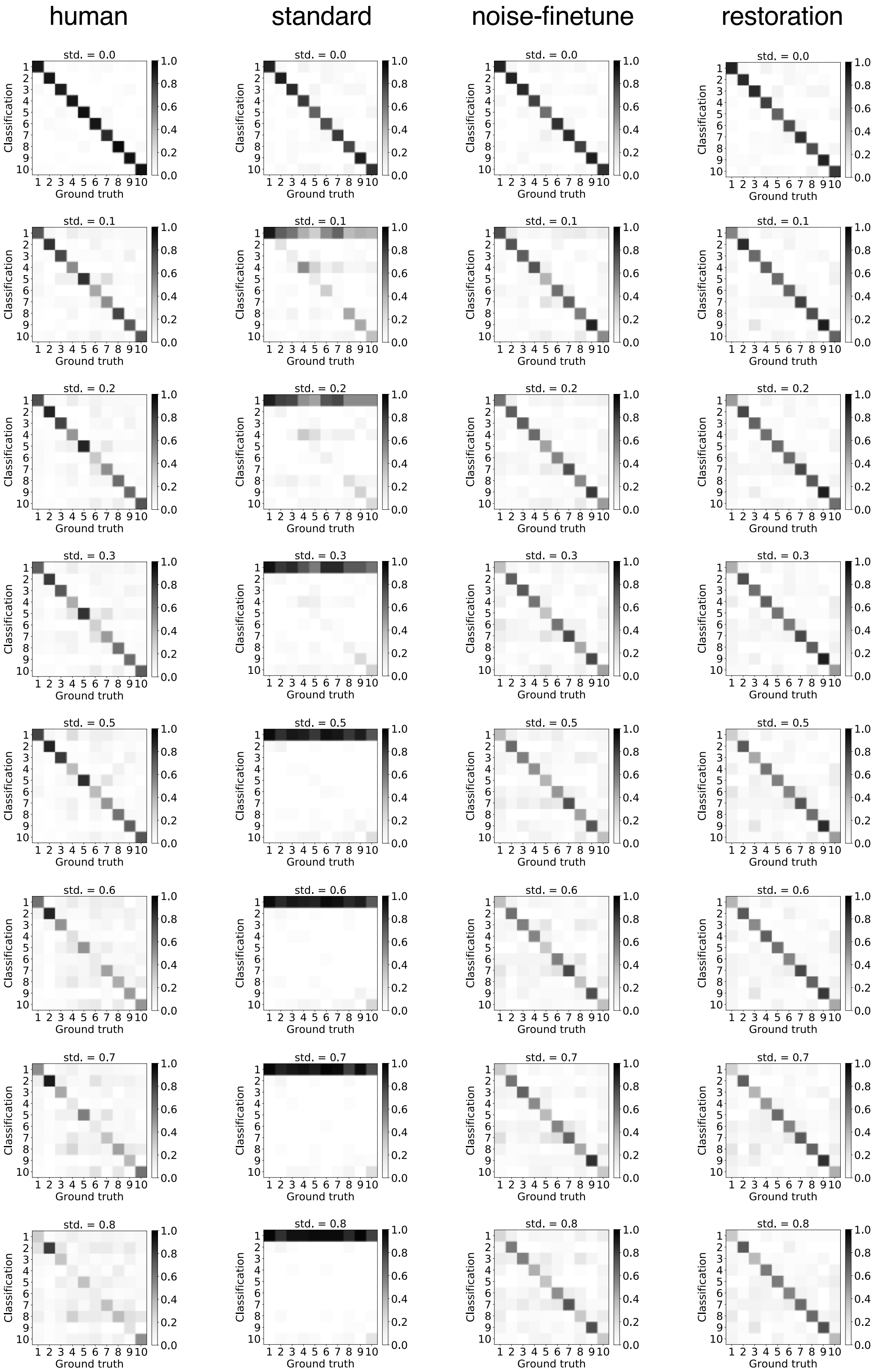


Figure 4-7: Performance comparison by confusion matrices.

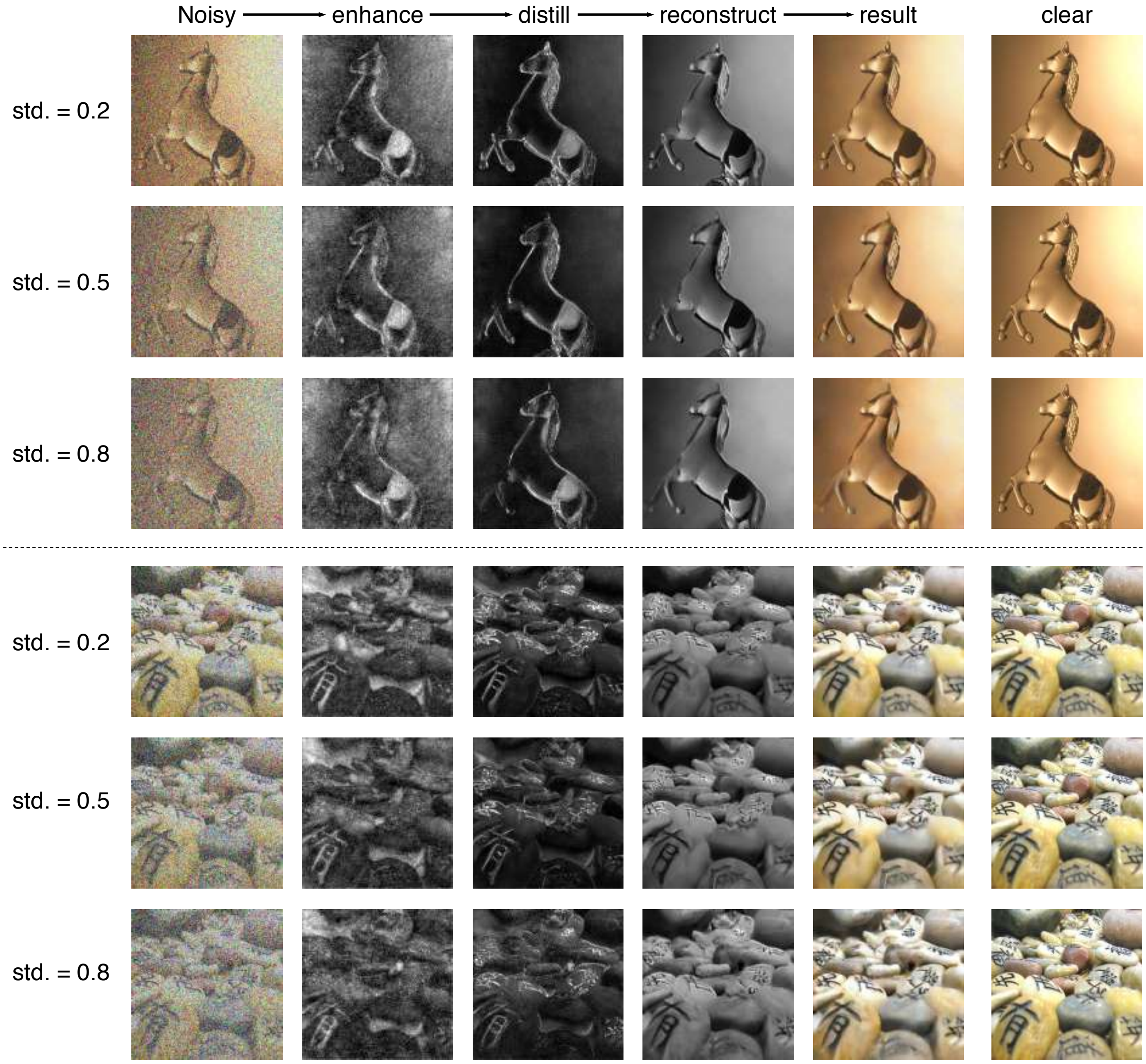


Figure 4-8: Examples of result.

Chapter 5

Artificial Intelligent System under Human Individual Differences

5.1 Introduction

Owing to the advancement of deep learning, artificial systems are now rival to humans in several pattern recognition tasks, such as visual recognition of object categories. However, this is only the case with the tasks for which correct answers exist independent of human perception. There is another type of tasks for which what to predict is human perception itself, in which there are often individual differences. Then, there are no longer single "correct" answers to predict, which makes evaluation of artificial systems difficult. In this section, focusing on pairwise ranking tasks sensitive to individual differences, we propose an evaluation method. Given a ranking result for multiple item pairs that is generated by an artificial system, our method quantifies the probability that the same ranking result will be generated by humans, and judges if it is distinguishable from human-generated results. We introduce a probabilistic model of human ranking behavior, and present an efficient computation method for the judgment. To estimate model parameters accurately from small-size samples, we present a method that uses confidence scores given by annotators for ranking each item pair. Taking as an example a task of ranking image pairs according to material attributes of objects, we demonstrate how the proposed method works.

Standard recognition tasks, in which some entities (e.g., labels) are to be predicted from an input (e.g., an image etc.), can be roughly categorized into two groups:

T_1: Tasks in which what to predict is given independent of human perception.

T_2: Tasks in which what to predict is human perception itself.

For the sake of explanation, we limit our attention to visual tasks in what follows, although the discussions apply to other modalities. Then, examples of T_1 are visual recognition of object categories and individual faces, and examples of T_2 are prediction of aesthetic quality ([92, 85, 64, 24]) and memorability ([57]) of images. In order to build a machine-learning-based system for task group T_2, it is first necessary to materialize what humans perceive from images. This can be performed by, for example, asking human subjects to give a score for an image or asking them to rank multiple images. Then, we consider training artificial systems (e.g., convolutional neural networks) so that they will predict the scores or ranking results as accurately as possible. As is well recognized, CNNs can now rival humans for several visual recognition tasks ([48, 125]), when they are properly trained in a supervised manner. Although it may not be widely recognized, this is only the case with task group T_1, in which labels to predict are given independent of human perception and thus there should exist correct answers to predict; therefore it is straightforward to define and measure the performance of CNNs. For task group T_2, however, it remains unclear whether CNNs can achieve the human level of performance, although the advancement of deep learning arguably has contributed to significant performance boost. This may be attributable to individual differences of annotators that often emerge in tasks of T_2. When there are individual differences, there is no unique correct answer to predict, which makes it difficult to evaluate the performance of artificial systems. Figure 5-1 shows an example of such cases, that is, pairwise ranking of images according to material attributes of objects. While, for some image pairs and attributes, human annotators will give unanimous ranking as depicted in Figure 5-1(a), for others, they will give diverged rankings. The latter can be divided into two cases, i.e., when the annotators confidently make diverged rankings, which mostly occurs for subjective cases, as in Figure 5-1(b), and when they are uncertain and give diverged rankings, as in Figure 5-1(c).

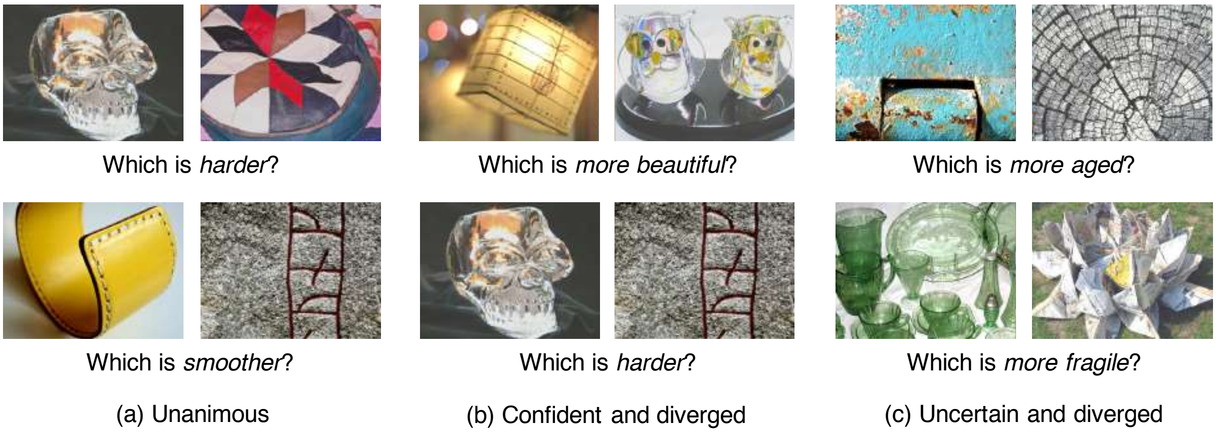


Figure 5-1: Examples of pairwise ranking of images according to material attributes of objects. Rankings given by different annotators are (a) unanimous, (b) diverged with confidence, or (c) uncertain and diverged.

In order to build an artificial system that rivals humans for this type of tasks, we need to answer the following questions:

Q_1. How can we measure its performance or judge its equivalence to humans?

Q_2. How can we build such an artificial system?

In this paper, targeting at pairwise ranking tasks, we attempt to give answers to these questions. For Q_1, we propose a method that is based on a probabilistic model of ranking results given by human subjects. Suppose that an artificial system predicts ranking of N image pairs. The proposed method quantifies *how probable it is that the same ranking result is generated by human annotators* for the same N image pairs. It properly considers the above individual differences. Despite its simplicity, there are a few difficulties to overcome with this approach. One is the difficulty with obtaining an accurate probabilistic model of human ranking from small sample size data[1]. To resolve this, we propose to collect confidence scores from annotators for ranking each item pair, and utilize them to accurately estimate a probabilistic model of human ranking (Sec.5.3). Thus, our proposal includes an annotation scheme for achieving the goal. Another difficulty is with computational complexity. The presence of individual differences increases

[1] We have M human annotators rank each of N item pairs. Considering data collection cost, increasing both M and N is prohibitive. As N needs to be large, M has to be small.

the number of ranking patterns that humans can generate for N item pairs, making naive computation infeasible. We present a method that performs the necessary computation efficiently (Sec.5.2.3). Following the proposed data collection scheme, we have created a dataset named Material Attribute Ranking Dataset (MARD), on which we tested the proposed method. The dataset will be made publicly available. For Q_2, we argue that learning to predict distribution of rankings given by multiple annotators performs better than previous methods. In previous studies ([98, 26, 13]), individual differences are usually ignored; the task is converted to binary classification by taking the majority if there are individual differences. Our experimental results support this argument.

5.2 Distinguishing Artificial Systems from Humans

5.2.1 Outline of the Proposed Approach

We consider a pairwise ranking task using N pairs, where, for instance, two images in each pair $(i = 1, 2, \ldots, N)$ are to be ranked. We denote a ranking result of a human subject (or an artificial system) by a sequence $X \triangleq \{x_1, x_2, \ldots, x_N\}$, where x_i is a binary variable, such that $x_i = 1$ if a subject chooses the first image, and $x_i = 0$ if the subject chooses the second image. Considering the aforementioned individual differences, we introduce a probabilistic model for X. Let $p(X)$ be the probability mass function of human generated sequences X's. Given a sequence $\mathtt{X}$ generated by an artificial system, we wish to use $p(X)$ to estimate how probable a human can generate $\mathtt{X}$. If the probability is very low, it means that $\mathtt{X}$ is distinguishable from human sequences; then, we may judge the artificial system behaved differently from humans. If it is high, then we cannot distinguish $\mathtt{X}$ from human sequences, implying than the artificial system behaves similarly to humans, as far as the given task/dataset is concerned. Ideally, any X's with $p(X) \neq 0$ can be generated from human subjects, and thus it could be possible to use $p(\mathtt{X}) = 0$ or $\neq 0$ for the above judgment. However, this is not appropriate. As the exact $p(X)$ is not available, we have to use an approximate model built upon several assumptions. Moreover, $p(X)$ is estimated from the data collected from human subjects, which could contain noises. It tends to have a long-tail (i.e., many sequences with a very small probability). Thus, the model $p(X)$ may be unreliable particularly for X's with low $p(X)$'s. These sequences are also considered to be minor and eccentric sequences that the majority of humans will not generate. Therefore, we use a criterion that excludes such minor sequences with the lowest probabilities. Let S denote a subset of 2^N possible sequences for the N pairs. In particular, we consider a subset S with the minimum cardinality $|S|$ that satisfies the following inequality:

$$\sum_{X \in S} p(X) > 1 - \epsilon, \tag{5.1}$$

where ϵ is a small number. We denote it by S_ϵ. S_ϵ indicates a set of sequences that humans are likely to generate. The complementary set $S_0 \setminus S_\epsilon$ contains the above-mentioned minor sequences. Given a ranking result $\mathtt{X}$ created by an artificial system for the same N pairs, we check if $\mathtt{X}$ belongs to S_ϵ, i.e., whether $\mathtt{X} \in S_\epsilon$ or not. We declare $\mathtt{X}$ to be indistinguishable from human results if $\mathtt{X} \in S_\epsilon$, and to be distinguishable if $\mathtt{X} \notin S_\epsilon$.

5.2.2 Model of Human Ranking

We now describe the model $p(X)$. Assuming that ranking results of different pairs are independent of each other, we model the probability of a sequence X by

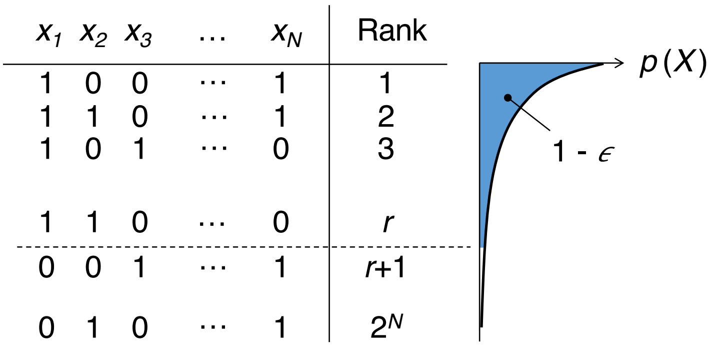
$$p(X) = p(x_1, x_2, \ldots, x_N) = \prod_{i=1}^{N} p(x_i). \tag{5.2}$$

We then model $p(x_i)$ using a Bernoulli distribution with a parameter θ_i, that is,

$$p(x_i = 1) = \theta_i \;\text{ and }\; p(x_i = 0) = 1 - \theta_i. \tag{5.3}$$

Determination of θ_i will be explained later. Human subjects can provide many different sequences; each sequence X will occur with probability $p(X)$.

It should be noted that the Bradley-Terry model, a popular model of pairwise ranking, is not fit for our problem. It considers a closed set of items (e.g., sport teams and scientific journals), and is mainly used for the purpose of obtaining a ranking of all the items in the set from observations of ranking of item pairs). On the other hand, in our case, we consider an open set



x_1	x_2	x_3	…	x_N	Rank
1	0	0	⋯	1	1
1	1	0	⋯	1	2
1	0	1	⋯	0	3
1	1	0	⋯	0	r
0	0	1	⋯	1	r+1
0	1	0	⋯	1	2^N

Figure 5-2: Each N bit sequence represents an instance of pairwise rankings for N item pairs. These sequences are sorted in the descending order of their probabilities. The hatched area of $p(X)$ on the right indicates the cumulative probability of $1 - \epsilon$. We check a machine-generated sequence is ranked above the lowest rank r. This test is efficiently performed by calculating its percentile value Q and see if $Q < 1 - \epsilon$. It is noteworthy that $p(X)$ has a long-tail.

of items. Our interest is not with the item set itself but with evaluation of an intelligent system performing the task.

5.2.3 Percentile of a Sequence in Probability-Ordered List

Percentile Value Q As mentioned above, we want to check whether $\mathtt{X} \in S_\epsilon$ or $\notin S_\epsilon$. To perform this, we calculate the *percentile* of $\mathtt{X}$ in the ordered list of all possible sequences X's in the order of $p(X)$. It is defined and computed as follows (see Fig.5-2): *i) Sort all (i.e., 2^N) possible binary sequences X's in descending order of $p(X)$ of (5.2) and ii) Compute the percentile of* $\mathtt{X}$ *(denoted by Q) using the cumulative sum of probabilities from the first rank to the position where* $\mathtt{X}$ *is ranked.* Using Q thus computed for the target $\mathtt{X}$, the condition $\mathtt{X} \in S_\epsilon$ is equivalent to $Q \leq 1 - \epsilon$ (and $\mathtt{X} \notin S_\epsilon$ is equivalent to $Q > 1 - \epsilon$). It is noteworthy that Q represents how close the sequence is to the most probable ranking result of humans, which corresponds to $Q = 0$.

Efficient Computation of Q When N is large, it is not feasible to naively perform the above procedure, as the number of possible sequences explodes. It is also noted that θ_i may differ for each i, and thus the standard statistics of binomial distribution cannot be computed for the whole sequence. Thus, we group the elements having the same θ_i to a subsequence, which are specified by an index set $I_g \triangleq \{i \,|\, \theta_i = \theta_g\}$ for a constant θ_g. Suppose that this grouping splits X into G subsequences $X_1, X_2, \ldots, X_G$. Using the independence of the elements, we have

$$p(X) = \prod_{g=1}^{G} p(X_g). \tag{5.4}$$

In this grouping, we may redefine the variable x_i by swapping the first and second images so that $\theta_i \geq 0.5$. This enables us to minimize the number G of groups without loss of generality to improve computational efficiency as will be discussed below.

Let n_g be the number of elements belonging to X_g (i.e., $n_g = |I_g|$), and θ_g be their Bernoulli parameter (i.e. $\theta_g = \theta_i$ for any $i \in I_g$). Then, the probability of X_g being a sequence $\mathtt{X}_g$ is computed by

$$P(X_g = \mathtt{X}_g) = \theta_g^{k_g}(1-\theta_g)^{(n_g-k_g)}, \tag{5.5}$$

where k_g is the number of 1's (i.e., $x_i = 1$) in $\mathtt{X}_g$. Note that the number of possible sequences having k_g 1's is $\binom{k_g}{n_g}$, and each of them has the same probability computed as above. For the entire sequence, the probability of X being a sequence $\mathtt{X}$ is given by

$$P(X = \mathtt{X}) = \prod_{g=1}^{G} P(X_g = \mathtt{X}_g). \tag{5.6}$$

Using (5.6) along with (5.5), we can compute the probability that a sequence $\mathtt{X}$ consists of subsequences $\mathtt{X}_g (g = 1, 2, \ldots, G)$, each of which has k_g 1's. The number of such sequences is calculated by $M(k_1, k_2, \ldots, k_g) \triangleq \prod\limits_{g=1}^{G} \binom{k_g}{n_g}$, and each sequence has the same probability.

Now, we consider sorting all 2^N sequences of X. We construct a sequence $\{k_1, k_2, \ldots, k_G\}$ by choosing each of its elements k_g (the number of 1's in the g^{th} group) from $[0, n_g]$. We obtain $M(k_1, k_2, \ldots, k_G)$ sequences having the same probability by employing this assignment scheme. We denote the j^{th} assignment by K_j ($j = 1, 2, \ldots, J$), where J is the number of possible assignments, which is given by $\prod_{g=1}^{G}(n_g + 1)$. (Note that $\sum_{j=1}^{J} M(K_j) = 2^N$.)

As it is not necessary to sort sequences having the same probability, we only need to sort J blocks of sequences instead of 2^N individual sequences. For each j^{th} block associated with an assignment K_j, we use (5.6) for computation of the probability of each sequence belonging to this block. Let P_j be this probability. Then, using P_j, we sort J blocks, which can be done much more efficiently than sorting 2^N sequences.

Finally, we compute Q for a sequence $\mathtt{X}$. In order to compute its rank, we count the number of 1's in $\mathtt{X}$ (more specifically, the number of 1's in each $\mathtt{X}_g$ of $\mathtt{X} = \{\mathtt{X}_1, \mathtt{X}_2, \ldots, \mathtt{X}_G\}$, and find the block $j_{\mathtt{X}}$ to which $\mathtt{X}$ belongs. Let $I_{\mathtt{X}}$ be the index set of blocks ranked higher than $j_{\mathtt{X}}$, i.e., $I_{\mathtt{X}} \triangleq \{j \,|\, P_j \geq P_{j_{\mathtt{X}}}\}$. Then, the cumulative probability down to block $j_{\mathtt{X}}$ (including it) can be computed by

$$Q = \sum_{j \in I_{\mathtt{X}}} M(K_j) P_j. \tag{5.7}$$

5.3 Estimation of the Bernoulli Parameter Using Confidence Scores

5.3.1 Small-sample Estimate of θ_i

We modeled human ranking by the Bernoulli distribution as in (5.3). We now consider how to estimate its parameter $\theta_i, \forall i$. Suppose n_i subjects participate in ranking the i^{th} image pair, $\forall i$. Let $n'_i (\leq n_i)$ be the number of subjects who chose the first image. Considering a pairwise ranking task with exclusive choice, the number of subjects who chose the second image is $n_i - n'_i$. Then, the maximum likelihood estimate (MLE) of θ_i is immediately given by

$$\theta_i = \frac{n'_i}{n_i}, \forall i. \tag{5.8}$$

Despite its simplicity, this method could have an issue when the subjects unanimously choose the same image of an image pair, i.e., either $n'_i = n_i$ or $n'_i = 0$. In this case, the above MLE gives $\theta_i = 1$ or $\theta_i = 0$, which leads to $p(x_i = 0) = 0$ or $p(x_i = 1) = 0$. However, this result is quite sensitive to (in)accuracy of θ_i and thus results may not be useful. If a CNN chooses the one with $p(x_i) = 0$ for only a single unanimous pair, then $p(X)$ immediately vanishes irrespective of ranking of other pairs (i.e., $Q = 100\%$), declaring that this CNN behaves completely differently from human. The estimate (5.8) could be inaccurate if n_i is not large enough. Although this issue will be mitigated by using a large n_i, it will first increase the cost of data collection; it will also increase the computational complexity of Q (because the number G of groups having an identical θ_i tends to increase).

5.3.2 A More Accurate Estimate Using a Confidence Score

Therefore, we instead consider collecting additional information from human subjects. For each image pair i, we ask them to additionally give a *confidence score* of their ranking. We use a score $s \in \{0, 1, 2\}$, which correspond to "not confident", "somewhat confident" and "very confident", respectively. As we ask n_i subjects for a single image pair i, we introduce an index $h(= 1, \ldots, n_i)$ to represent each subject and denote the ranking choice and score of h-th subject by x_{ih} and s_{ih}, respectively. Let $X^{(i)} = [x_{i1}, \ldots, x_{in_i}]$ and $S^{(i)} = [s_{i1}, \ldots, s_{in_i}]$. Assuming independence of individual annotations, we have

$$\begin{aligned} p(X^{(i)}, S^{(i)}|\theta_i) &= \prod_{h=1}^{n_i} p(x_{ih}, s_{ih}|\theta_i) \\ &= \prod_{h=1}^{n_i} p(s_{ih}|x_{ih}, \theta_i) p(x_{ih}|\theta_i). \end{aligned} \tag{5.9}$$

We use this model to perform maximum likelihood estimation for the unanimously ranked pairs. Suppose that all n_i subjects chose the first image. Then we have

$$\prod_{h=1}^{n_i} p(x_{ih} = 1|\theta_i) = {\theta_i}^{n_i}. \tag{5.10}$$

To model $p(s_{ih}|x_{ih}, \theta_i)$, we consider the probability of occurrence of each score $s = 0, 1, 2$; we denote it by $q_s^{(i)} \equiv p(s_{ih} = s)$. Then we have

$$\prod_{h=1}^{n_i} p(s_{ih}|x_{ih} = 1, \theta_i) = {q_0}^{n_0} {q_1}^{n_1} {q_2}^{n_2}, \tag{5.11}$$

where n_0, n_1 and n_2 denotes the number of subjects who chose confidence scores 0, 1, and 2, respectively; we omit the superscript in q_0, q_1, q_2 etc. for simplicity. Thus, from (5.9) we have

$$p(X^{(i)} = \mathbf{1}, S^{(i)}|\theta_i) = {\theta_i}^{n_i}{q_0}^{n_0}{q_1}^{n_1}{q_2}^{n_2}. \tag{5.12}$$

Given the observations $S^{(i)}$ and (n_0, n_1, n_2), we want to maximize the likelihood (5.12) with respect to θ_i as well as the introduced unknowns q_0, q_1, and q_2. As they are probabilities, there are a few constraints for q_0, q_1, and q_2 defined as

$$q_0 + q_1 + q_2 = 1, \tag{5.13a}$$

$$0 \leq q_0, q_1, q_2 \leq 1. \tag{5.13b}$$

We then assume a relation between the confidence scores and θ_i leading to the following equation:

$$\frac{1}{2}q_0 + \frac{3}{4}q_1 + q_2 = \theta_i. \tag{5.14}$$

This equation indicates that the three scores, *not confident*, *confident*, and *very confident*, are mapped to $\theta_i = 0.5$, 0.75, and 1.0, respectively. In other words, subjects who are not confident irrespective of the choice $x_i = 1$ will choose the other with 50% in the future; those who are very confident will always do the same choice in the future; those who have intermediate confidence will do the same with the intermediate probability 75% in the future.

Using the constraints (5.13) and (5.14), we can maximize the likelihood (5.12) with respect to the unknowns. To be specific, we eliminate, say, q_0 and q_1, from (5.12) using (5.13a) and (5.14) and maximize it for θ_i and q_2 with the inequality constraints (5.13b). Any numerical constrained maximization method can be used for this optimization. As a result, we have the MLE for θ_i. It should be noted that we use this estimation method only when the ranking results are unanimous; we use the standard estimate (5.8) otherwise.

Discussion on the Use of Confidence Score As is described above, we propose to use a confidence score to estimate each θ_i. An alternative use of confidence scores is to simply eliminate the ranking results with a confidence score less than 2 and go with only maximally confident ranking results. We do not adopt this approach due to the following reason. We think that there are two cases for how individual differences emerge (see also Fig.5-1): i) annotators make selections confidently, which are nevertheless split (subjectivity); and ii) annotators make selections without confidence, which makes their decisions fluctuated and then split (uncertainty). Elimination of samples with score 0 or 1 will remove data of type (ii). This will be fine when we are only interested in data of type (i), which seems to be the case with most existing studies. On the other hand, our study also considers data type (ii). Suppose, for instance, the case where people perceive very similar glossiness for two different objects, e.g., a plastic cup and a glass mug; uncertainty will be fairly informative for such cases. This is why we attempt to model the uncertainty instead of eliminating it. As we are interested in modeling individual differences, we don't eliminate samples with low inter-rater agreement, either.

5.4 Experimental Results

5.4.1 Material Attribute Ranking Dataset

To test our method, we have created a dataset of a pairwise ranking task. It has been shown ([33, 30]) that humans can visually perceive fairly accurately the "material attribute" of an object surface, such as hardness, coldness, lightness etc. Motivated by this, we chose a task of ranking a pair of images according to such material attributes. We consider thirteen material attributes, namely *aged, beautiful, clean, cold, fragile, glossy, hard, light, resilient, smooth, sticky, transparent*, and *wet*.

The dataset, which we name Material Attribute Ranking Dataset (MARD), consists of 1,000 training and 300 testing samples for each of the thirteen attributes. Each sample contains ranking results of an image pair of five Mechanical Turkers. For test samples for which the initial five Turkers make a unanimous selection, the dataset provides additional ranking results of ten more Turkers as well as three-level scores of their confidence in their ranking.

It should be noted that MARD differs from any of similar existing datasets of pairwise ranking, e.g., Emotion Dataset ([148]). The existing datasets simply discard individual differences by taking the majority of nonunanimous annotations, which is equivalent to regarding the most probable ranking result of humans as the only correct prediction. Although this greatly simplifies the problem, this makes it impossible to consider individual differences in any ways.

Details of Creation of MARD MARD uses images of the Flicker Material Database (FMD) ([116]), a benchmark dataset of classification of ten materials categories. We split 1,000 FMD images into two sets of non-overlapping 500 images (one set is used for training and the other set is used for testing). We then created 1,000 and 300 image pairs by randomly choosing images from each set, respectively. We asked five Turkers to rank each image pair in terms of each of the thirteen material attributes by showing the paired images in a row. (The same image pairs were used for all the attributes.) Specifically, we asked them to choose one of three options, the first image, the second image, and "unable to decide", for each pair. We discarded the image pairs with three or more "unable to decide" in the training set and those with one or more "unable to decide" in the test set. For the image pairs of the test set which were ranked unanimously for an attribute by the five Turkers, we further have ten more Turkers rank the same image pair and attribute. In this second task, we removed the "unable to decide" option, and also asked the Turkers to additionally provide their confidence on their ranking by choosing one of three level confidence, i.e., "Not confident", "Somewhat confident", and "Very confident". The confidence scores are used in order to estimate the parameter θ_i to mitigate the sensitivity issue with unanimous ranking, as was discussed in Section 5.3.

5.4.2 Evaluation of Differently Trained CNNs

Using the MARD, we conducted experiments to test our evaluation method. To see how it evaluates different prediction methods, we consider four methods. The first three are existing methods that convert the task into binary classification by regarding the majority of human ranking results as the correct label to predict. For the fourth method, we present a method that considers the individual differences as they are.

- RankCNN ([26]): A CNN is trained to predict binary labels by minimizing weighted squared distances computed for each training sample to its binary label.
- RankNet$_h$ ([98]): In their original work, a linear classifier is trained to predict binary labels by searching for maximum-margin hyper-planes in a feature space. In this study,

we instead employ a hinge-loss.

- RankNet$_p$ ([13]): A neural network (a CNN in this study) is trained to predict binary labels by minimizing the cross-entropy loss.

- RankDist (ours): A CNN is trained to predict the distribution of ranking of each item pair. To be specific, it is trained to predict the ML estimate (5.8) of the Bernoulli parameter θ_i. (Recall that training samples do not provide scores.) The cross-entropy loss is used.

We applied the above four methods to the training set of the MARD. For all the four methods, we use the same CNN, VGG-19 ([118]). It is first pretrained on the ImageNet and finetuned on the EFMD ([156]). Then it is trained using the MARD where its ten lower weight layers are fixed and the subsequent layers are updated. It is used in a Siamese fashion, providing two outputs, from which each loss is computed. The losses for all the thirteen adjectives are summed and minimized. Note that the four methods differ only in their employed losses.

Table 5.1 shows the Q values of the four methods for the thirteen attributes that are computed using the test set of the MARD. When choosing 90.0% for the threshold for Q, any ranking results with $Q \geq 90.0\%$ are declared to be distinguishable from human ranking. It is observed that the number of such attributes is 3, 2, 3, and 1, for RankCNN, RankNet$_h$, RankNet$_p$, and RankDist, respectively. RankDist tends to provide smaller Q's for many attributes, indicating that its ranking results are closer to the most probable human results than other three. The three methods show more or less similar behaviours.

In summary, the proposed evaluation method enables to show which method provides ranking that are (in)distinguishable from human ranking for which attribute. For instance, the ranking results for the attribute *hard* are the most dissimilar to human ranking, implying that there is room for improvements. Our method also makes it possible to visualize the different behaviours of the four methods; in particular, RankDist that considers individual differences performs differently from the three existing methods that neglect individual differences.

Table 5.1: Evaluation of four differently trained CNNs on the MARD dataset. Numbers which are larger than 90% are displayed in bold fonts, indicating that the ranking results are distinguishable from human-generated results.

(VGG19)	RankCNN	RankNet$_h$	RankNet$_p$	RankDist
aged	0.7	1.9	4.0	3.4
beautiful	39.8	38.9	32.3	38.3
clean	31.6	5.7	27.9	16.3
cold	1.8	8.4	10.7	0.2
fragile	**99.3**	**97.4**	**98.7**	89.1
glossy	72.4	88.0	**90.9**	32.9
hard	**100**	**100**	**100**	**93.8**
light	35.9	20.8	36.5	4.9
resilient	18.6	26.9	16.5	4.3
smooth	54.3	61.3	71.5	33.1
sticky	31.2	31.2	10.0	12.5
transparent	**100**	10.0	7.4	8.7
wet	25.3	27.2	7.9	0.3

5.5 Summary

In this study, we have discussed how to compare artificial systems with humans for the task of ranking a pair of items. We have proposed a method for judging if an artificial system is distinguishable from humans for ranking of N item pairs. More rigorously, we check if an N-pair ranking result given by an artificial system is distinguishable from those given by humans. It relies on a probabilistic model of human ranking that is based on the Bernoulli distribution. We have proposed to collect confidence scores of ranking each item pair from annotators and utilize them to estimate the Bernoulli parameter accurately for each item pair. We have also shown an efficient method for the judgment that calculates and uses the percentile value Q of the target N-pair ranking result. Taking annotation noises and inaccuracies with the models into account, Q is compared with a specified threshold (e.g., 90.0%); if it is smaller than the threshold, we declare the artificial system is indistinguishable from humans for rankings of the N-item pairs. The value Q may also be used as a measure of how close the ranking result of the artificial system is to the most probable ranking result of humans.

Chapter 6

Conclusions

In this thesis, we have studied image restoration on a number of distortion factors. Evidence that CNNs are not robust to image distortions, have been shown in many works including this thesis. This can result in failures in applications of CNNs. Various studies have tackled this problem from the perspective of domain-shift. By domain-shift, we mean the data on which a CNN was trained is different from those on which the CNN is used for inference. However, their approaches are either not applicable to many factors of image distortion, or not powerful enough to achieve a good improvement of performance on the factors. Towards developing powerful approaches for restoring clear images from their various distorted versions. We have proposed a style of residual connection, dubbed "dual residual connection", aiming to exploit the potential of paired operations for image restoration tasks. We have shown the design of a modular block (DuRB) that implements this connection style, which has two containers for the paired operations such that the user can insert any arbitrary operations to them. We have also shown choices of the two operations in the block as well as the entire networks (DuRN) containing a stack of the blocks for five different image restoration tasks. The experimental results obtained using nine datasets show that the proposed approach consistently works better than previous methods. This is introduced in Chapter 2.

In Chapter 3, we have further proposed a universal network that has a single input and multiple output branches, to solve multiple image restoration tasks in the same model. We have shown through experiments that the single network can learn to perform motion-blur removal, haze-removal, rain-streak removal and JPEG compression noise removal. We have proposed two extensions to the backbone design of the dual residual network, which was proposed in Chapter 2. The extensions are the improved attention mechanism that considers the spatial gradients of feature maps to generate channel-wise attention weights and the basic block design named DuRB-M that performs two operations in parallel and fuses them. Experimental results show that the universal netowrk achieves a new state-of-the-art performance on motion blur removal, haze removal (both in PSNR/SSIM) and JPEG artifact removal (in SSIM).

In Chapter 4, we first demonstrate that CNNs have worse tolerance to Gaussion noise than humans. This implies differences between human's perception and its computational model established by the CNNs. We investigated two approaches that improves classification accuracy on noisy images up to humans' level, intending to explain how come is human vision robust to noise. Although this study is still in an experimental stage, our experimental findings provide novel insights into human visual mechanism for material recognition.

In Chapter 5, towards a deeper discussion between humans and CNNs, we have discussed how to compare artificial systems with humans for the task of ranking a pair of items under human individual differences. We have proposed a method for judging if an artificial system is distinguishable from humans for ranking of N item pairs. More rigorously, we check if an N-pair ranking result given by an artificial system is distinguishable from those given by humans. It relies on a probabilistic model of human ranking that is based on the Bernoulli distribution. We have proposed to collect confidence scores of ranking each item pair from annotators and utilize them to estimate the Bernoulli parameter accurately for each item pair. We have also shown an efficient method for the judgment that calculates and uses the percentile value Q of the target N-pair ranking result. Taking annotation noises and inaccuracies with the models into account, Q is compared with a specified threshold (e.g., 90.0%); if it is smaller than the threshold, we declare the artificial system is indistinguishable from humans for rankings of the N-item pairs. The value Q may also be used as a measure of how close the ranking result of the artificial system is to the most probable ranking result of humans.

Appendix A

Appendix for Dual Residual Networks (Chapter 2)

This document provides additional explanations about the experimental setting for each of the five image restoration tasks.

A.1 Implementation Details and Additional Results for the Five Tasks

A.1.1 Details of Training Method

We use the Adam optimizer with $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 1.0 \times 10^{-8}$ for training all the proposed DuRNs. For loss functions, we use a weighted sum of SSIM and l_1 loss, specifically, $1.1 \times$ SSIM $+ 0.75 \times l_1$, for all the tasks. There are two exceptions. One is Gaussian noise removal on the BSD500-grayscale dataset [86], where we use l_2 loss. The other is raindrop removal, where we use the same weighted loss for the first 4,000 epochs, and then switch it to a single l_1 loss for additional 100 epochs. The initial learning rate is set to 0.0001 for all the tasks. All the experiments are conducted using PyTorch [99]. Our code and trained models are made publicly available at *https://github.com/liu-vis/DualResidualNetworks*

A.1.2 Noise Removal

Specification of ct_1^l and ct_2^l We show the specification of ct_1^l and ct_2^l for each DuRB-P in Table A.1, in which $l (= 1, \dots, 6)$ denotes the block-id of a DuRB; the "recep." denotes the receptive field of convolution. It is observed that the paired convolution has a large- and small- receptive field for each DuRB-P (see each row in the table), and the size of the receptive fields of ct_1^l and ct_2^l increases with l with an exception at $l = 5$, which is to avoid too large a receptive field. By this design we intend to make each block look at the input image at an increasing scale with layers in the forward direction.

Experimental Setting for Gaussian Noise Removal In training, we set batch size $= 100$. Each input image in a batch is obtained by randomly cropping a 64×64 region from an original training noisy image. We exactly followed the procedure of [122] to generate noisy images for training our network.

Experimental Setting for Real-World Noise Removal In training, we randomly select 30 out of 40 pairs of a high resolution noisy image and a mean image (used as ground truth) for constructing the training dataset. We set input patch size $= 128 \times 128$, and use 30 patches (each of which is randomly cropped from a different training image) to create one batch. To test the CNNs including ours and the baselines, we use the remaining 10 image pairs; specifically, we randomly crop ten 512×512 patches from each of them, yielding 100 patches that are used for the test.

Table A.1: The specification of ct_1^l and ct_2^l for DuRB-P's for noise removal. The "recep." denotes the receptive field of convolution, i.e., delation rate $\times$ (kernel size - 1) $+ 1$.

DuRB-P

layer	kernel	dilation	recep.	layer	kernel	dilation	recep.
$ct_1^{l=1}$	5	1	5×5	$ct_2^{l=1}$	3	1	3×3
$ct_1^{l=2}$	7	1	7×7	$ct_2^{l=2}$	5	1	5×5
$ct_1^{l=3}$	7	2	13×13	$ct_2^{l=3}$	5	1	5×5
$ct_1^{l=4}$	11	2	21×21	$ct_2^{l=4}$	7	1	7×7
$ct_1^{l=5}$	11	1	11×11	$ct_2^{l=5}$	5	1	5×5
$ct_1^{l=6}$	11	3	31×31	$ct_2^{l=6}$	7	1	7×7

A.1.3 Motion Blur Removal

Table A.2: The specification of ct_1^l for DuRB-U's for motion blur removal.

DuRB-U

layer	kernel	dilation	recep.
$ct_1^{l=1}$	3	3	7
$ct_1^{l=2}$	7	1	7
$ct_1^{l=3}$	3	3	7
$ct_1^{l=4}$	7	1	7
$ct_1^{l=5}$	3	2	5
$ct_1^{l=6}$	5	1	5

Specification of ct_1^l and ct_2^l The specification of ct_1^l is shown in Table A.2. For ct_2^l, we use an identical configuration, kernel size $= 3 \times 3$, dilation rate $= 1$ and stride $= 2$, for all DuRB-U's. We intend to simply perform down-sampling with ct_2^l.

Experimental Setting on GoPro Dataset In training, we set batch size $=$ 10. Each input image in a batch is obtained by randomly cropping a 256×256 patch from the re-sized version (640×360) of an original training image of size 1280×720. In testing, we use the re-sized version (640×360) of the original test images of size 1280×720 as in training.

Experimental Setting on Car Dataset The Car dataset was used only for evaluation. We down-scale the blur images from their original size 720×720 to 360×360 and input them to the DuRN-U trained using GoPro-train dataset for de-blurring. The result is then up-scaled to 700×700 and fed into YOLOv3.

Additional Examples More examples of motion blur removal on GoPro-test dataset are shown in Fig. A-1.

A.1.4 Haze Removal

Specification of ct_1^l and ct_2^l The specification of ct_1^l is shown in Table A.3. For ct_2^l, we use an identical configuration, i.e., kernel size $= 3 \times 3$, dilation rate $= 1$ and stride $= 2$, for all the DuRB-US's. We intend to simply perform down-sampling with ct_2^l.

Table A.3: The specification of ct_1^l for DuRB-US's for haze removal.

DuRB-US

layer	kernel	dilation	recep.	layer	kernel	dilation	recep.
$ct_1^{l=1}$	5	1	5	$ct_1^{l=7}$	11	1	11
$ct_1^{l=2}$	5	1	5	$ct_1^{l=8}$	11	1	11
$ct_1^{l=3}$	7	1	7	$ct_1^{l=9}$	11	1	11
$ct_1^{l=4}$	7	1	7	$ct_1^{l=10}$	11	1	11
$ct_1^{l=5}$	11	1	11	$ct_1^{l=11}$	11	1	11
$ct_1^{l=6}$	11	1	11	$ct_1^{l=12}$	11	1	11

Experimental Setting on Dehaze Dataset In training, we set batch size $= 20$. Each input image in a batch is obtained by randomly cropping a 256×256 region from an original training image of size 512×512.

Experimental Setting on RESIDE In training, we set batch size $= 48$. Each input image in a batch is obtained by randomly cropping a 256×256 region from an original image of size 620×460.

Visualization of Internal Layer Activation Figure A-5 shows activation maps of several chosen blocks (i.e., DuRB-US's) in the network for different input images. They are the sums in the channel dimension of activation maps of the input to the first DuRB-US ($l = 0$), and of the output from the third ($l = 3$), sixth ($l = 6$), and twelfth ($l = 12$) DuRB-US's. It is seen that the DuRN-US computes a map that looks similar to transmission map at around $l = 3$.

Additional Examples More examples of haze removal are shown on Figs. A-2, A-3 and A-4. In Fig. A-4, we show the results for images of hazy scenes that are captured using iPhone-6 plus by us.

A.1.5 Raindrop Removal

Specification of ct_1^l and ct_2^l The specification of ct_1^l for the three DuRB-S's and the six DuRB-P's is shown in Table A.4. For ct_2^l, we use an identical configuration, kernel size $= 3 \times 3$ and dilation rate $= 1$, for all the DuRB-S's, and use an identical configuration, kernel size $= 5 \times 5$ and dilation rate $= 1$, for all the DuRB-P's.

Table A.4: The specification of ct_1^l for DuRB-S's and DuRB-P's of the DuRN-S-P for raindrop removal.

DuRB-S

layer	kernel	dilation	recep.
$ct_1^{l=1}$	3	12	25
$ct_1^{l=2}$	3	8	17
$ct_1^{l=3}$	3	6	13

DuRB-P

layer	kernel	dilation	recep.
$ct_1^{l=1}$	3	2	5
$ct_1^{l=2}$	5	1	5
$ct_1^{l=3}$	3	3	7
$ct_1^{l=3}$	7	1	7
$ct_1^{l=3}$	3	4	9
$ct_1^{l=3}$	7	1	7

Experimental Setting on RainDrop Dataset In training, we set batch size $= 24$. Each input image in a batch is obtained by randomly cropping a 256×256 region from the original image of size 720×480. As mentioned before, we train the network $1.1 \times \text{SSIM} + 0.75 \times l_1$ using the loss for 4,000 epochs, and then switch the loss to l_1 alone, training the network for additional 100 epochs. We did this for faster converging.

Additional Examples More examples of raindrop removal are shown in Fig. A-6.

A.1.6 Rain-streak Removal

Specification of ct_1^l and ct_2^l We use the same configuration as noise removal. See Table. A.1. Note that we use DuRB-S for this task.

Experimental Setting on DDN Data To train the DuRN-S, we set batch size $= 40$. Each input image in a batch is obtained by randomly cropping a 64×64 region from an original training image.

Experimental Setting on DID-MDN Data In training, we set batch size $= 80$. Each input image in a batch is obtained by randomly cropping a 64×64 region from an original training image.

Additional Examples More examples of rain-streak removal on synthetic rainy images and on real-world rainy images are shown in Fig. A-7 and Fig. A-8, respectively.

Table A.5: Performance (PSNR/SSIM) of the four versions of DuRBs (i.e., -P, -U, -US, and -S) on different task.

	Real-noise	Motion blur	Haze	Raindrop	Rain-streak
DuRB-P	**36.83 / 0.9635**	29.40 / 0.8996	29.33 / 0.9761	24.69 / 0.8067	32.88 / 0.9214
DuRB-U	36.63 / 0.9600	29.90 / 0.9100	30.79 / 0.9800	24.30 / 0.8067	33.00 / **0.9265**
DuRB-US	36.61 / 0.9591	**29.96 / 0.9101**	**32.60 / 0.9827**	22.72 / 0.7254	32.84 / 0.9238
DuRB-S	36.82 / 0.9629	29.55 / 0.9023	31.81 / 0.9792	**25.13 / 0.8134**	**33.21** / 0.9251

A.1.7 Performance of DuRBs on Non-target Tasks

We have presented the four versions of DuRB, each of which is designed for a single task. To verify the effectiveness of the design choices, we examine the performance of each DuRB on its non-target tasks. Specifically, we evaluate the performance of every combination of the four versions of DuRB and the five tasks. For noise, motion blur, haze, raindrop and rain-streak removal, we train and test networks consisting of each version of DuRB on Real-World Noisy Image Dataset, GoPro Dataset, Dehaze Dataset, RainDrop Dataset and DID-MDN Data. The results are shown in Table A.5. It is seen that in general, each DuRB yields the best performance for the task to which it was designed. For motion blur removal, DuRB-US performs comparably well or even slightly better than DuRB-U, which is our primary design for the task. We think this is reasonable, as DuRB-US contains the same paired operation as DuRB-U (i.e., up- and down-sampling), contributing to the good performance. Their performance gap is almost negligible and thus DuRB-U is a better choice, considering its efficiency.

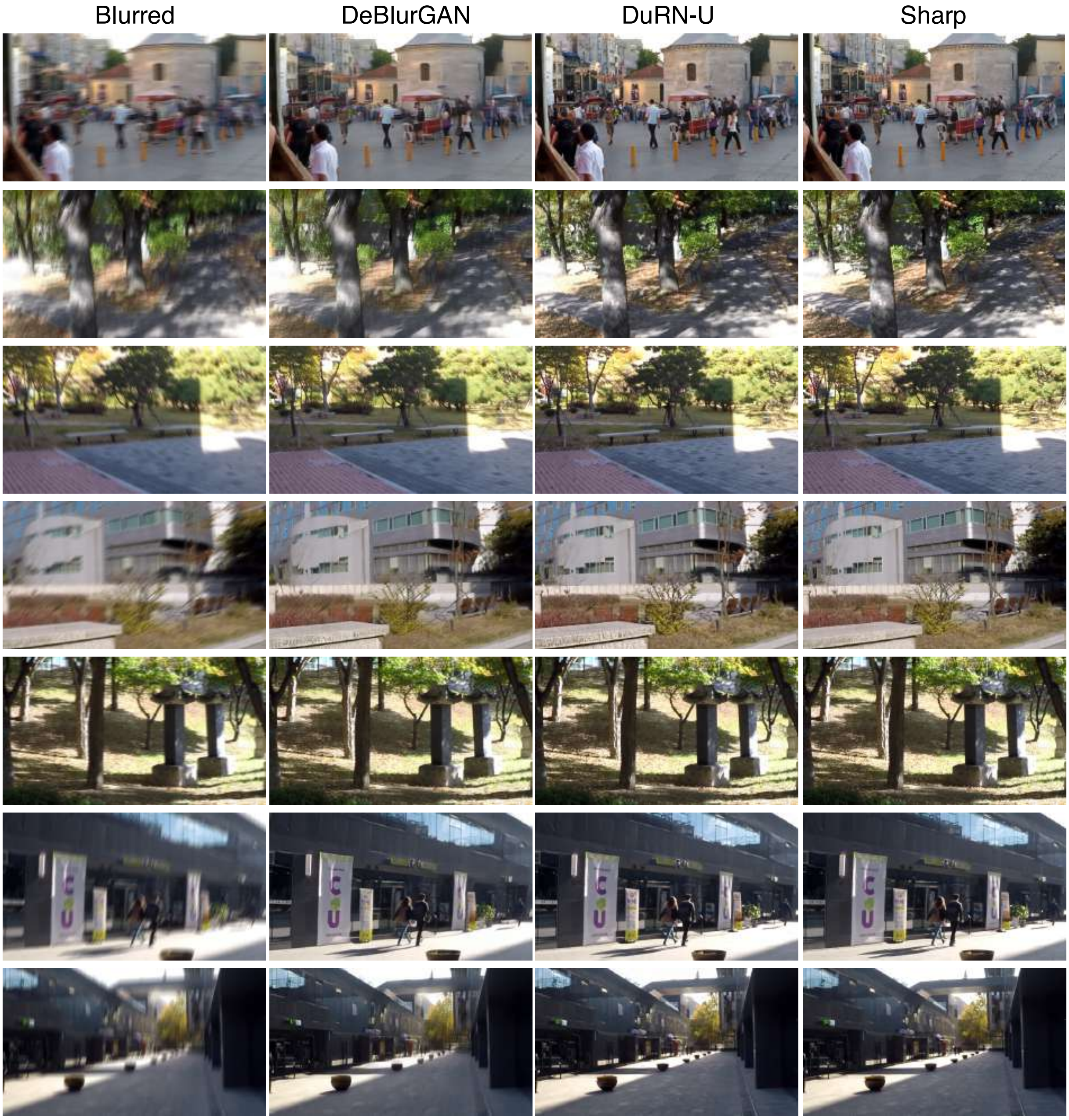


Figure A-1: Examples for motion blur removal on GoPro-test dataset.

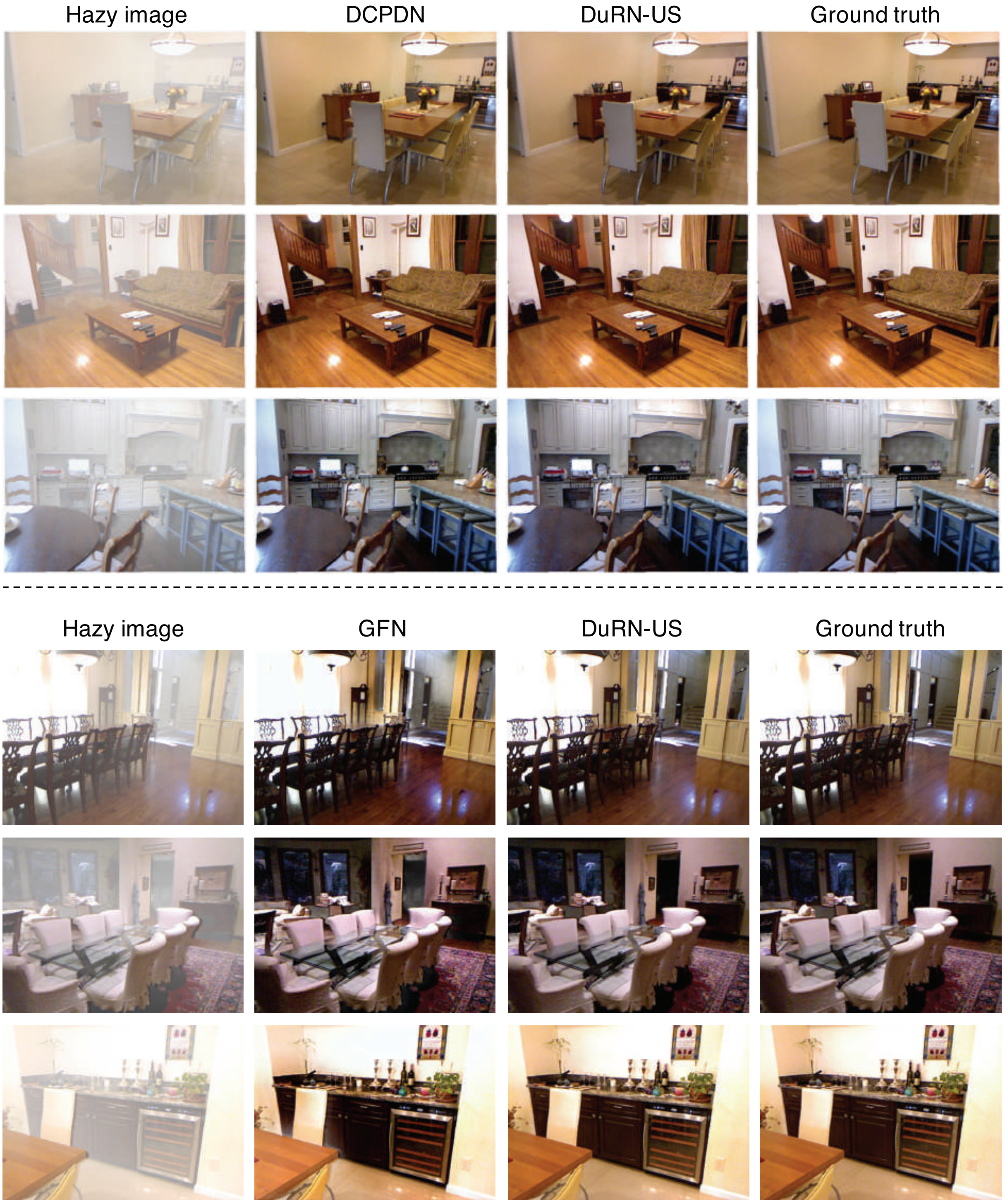


Figure A-2: Examples of haze removal on synthetic hazy images.

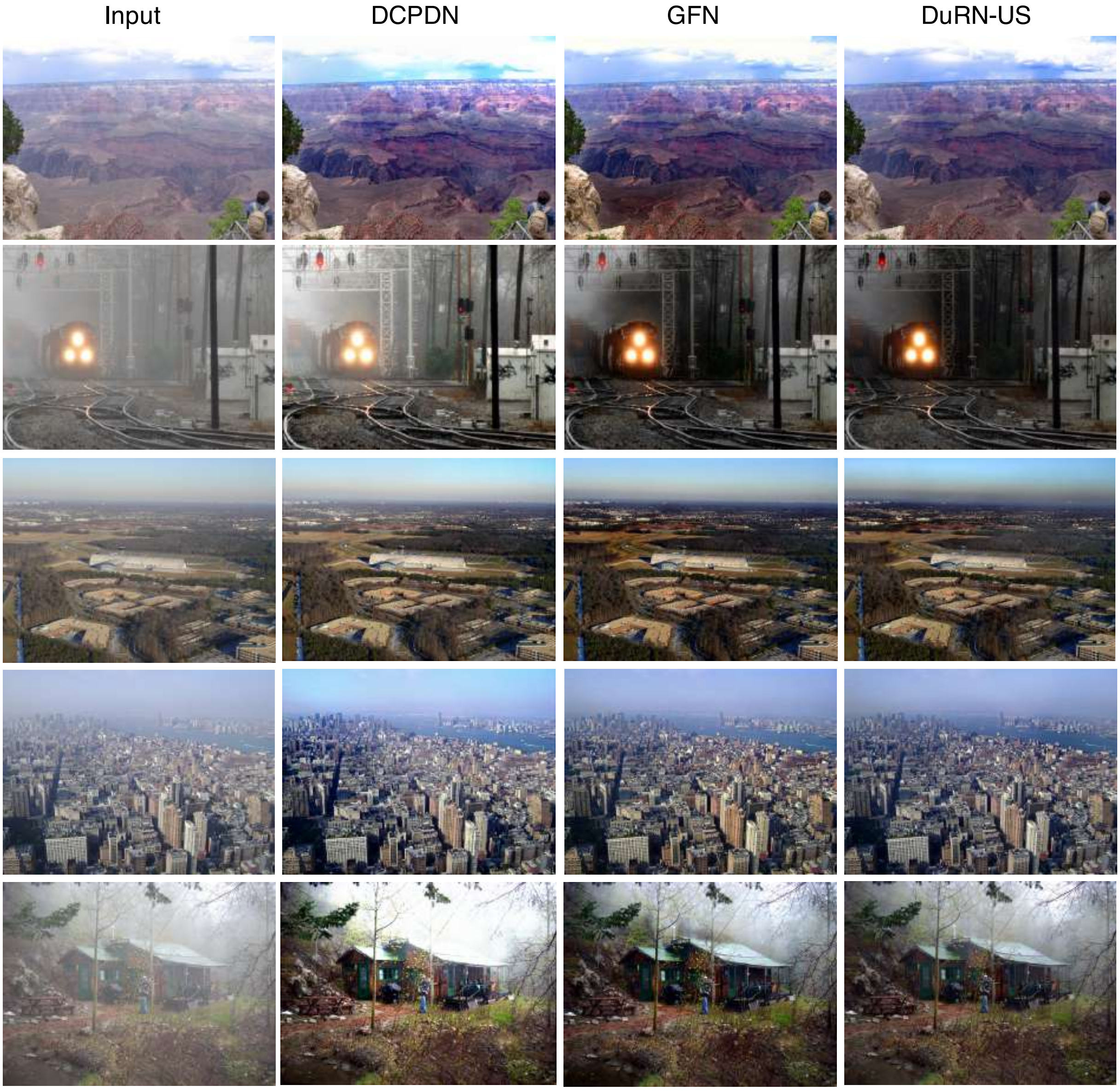


Figure A-3: Examples of haze removal on the hazy images used in previous works such as [103, 151, 44]

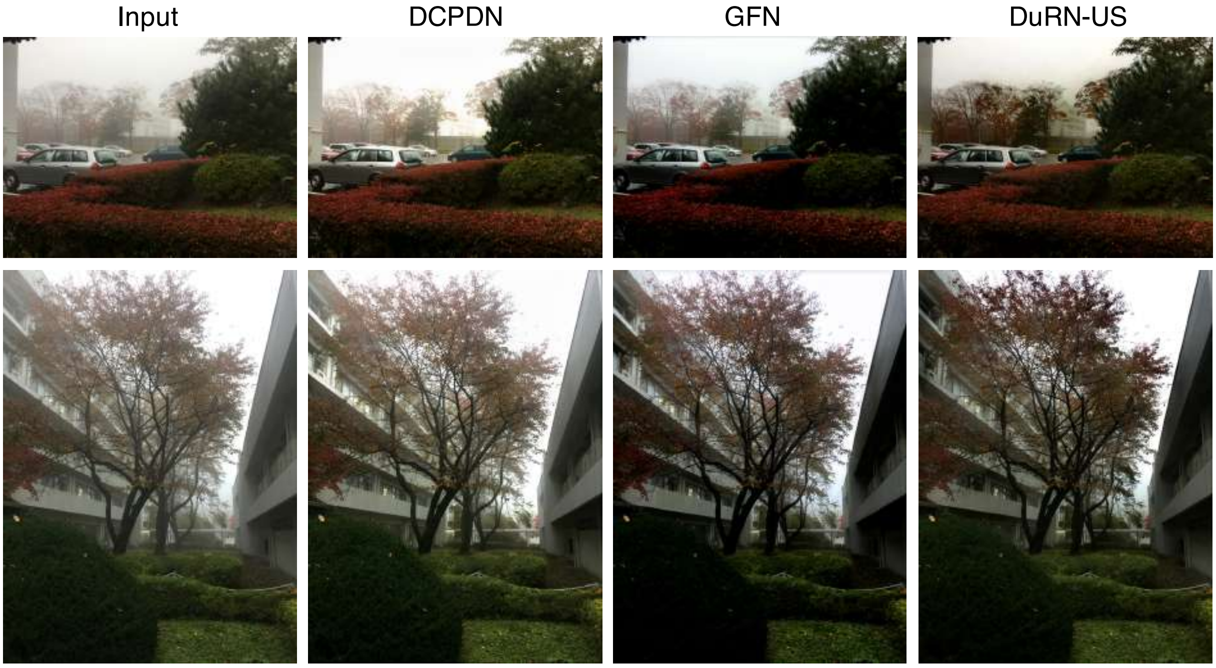


Figure A-4: Examples of haze removal on real-world hazy images.

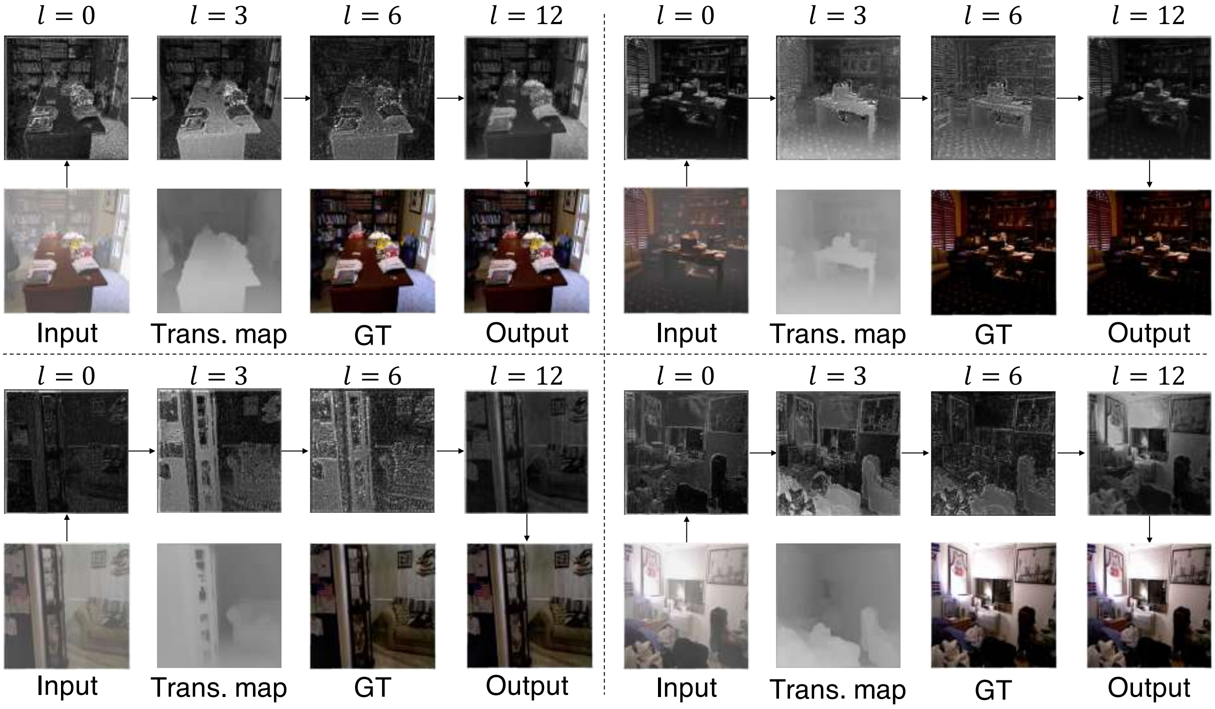


Figure A-5: Visualization of internal activation maps of the DuRN-US.

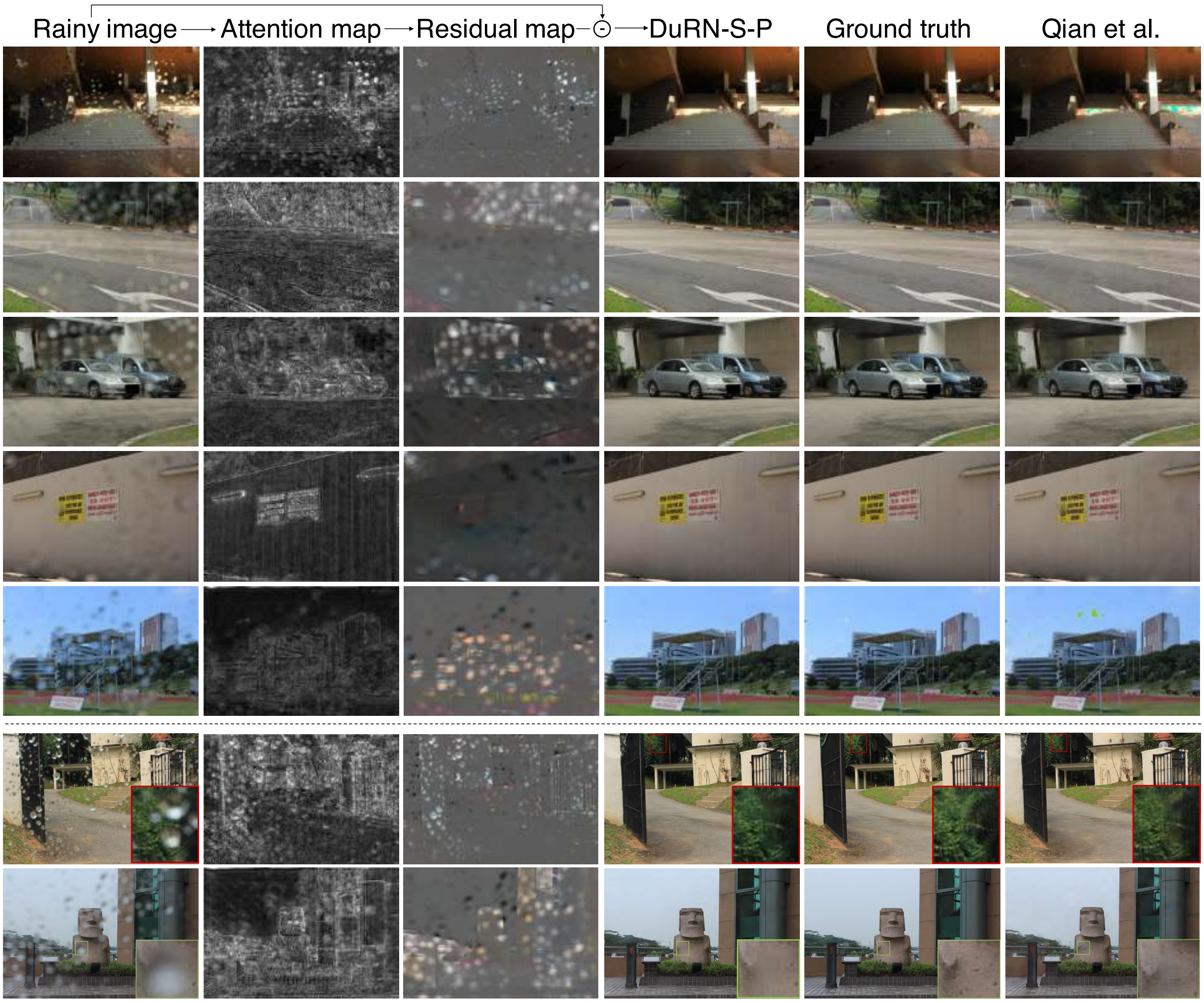


Figure A-6: Examples of raindrop removal along with interal activation maps of DuRN-S-P. The “Attention map” and “Residual map” are the outputs of the Attentive-Net and the last $Tanh$ layer shwon in Fig. 13 in the main-text; they are normalized for better visibility.

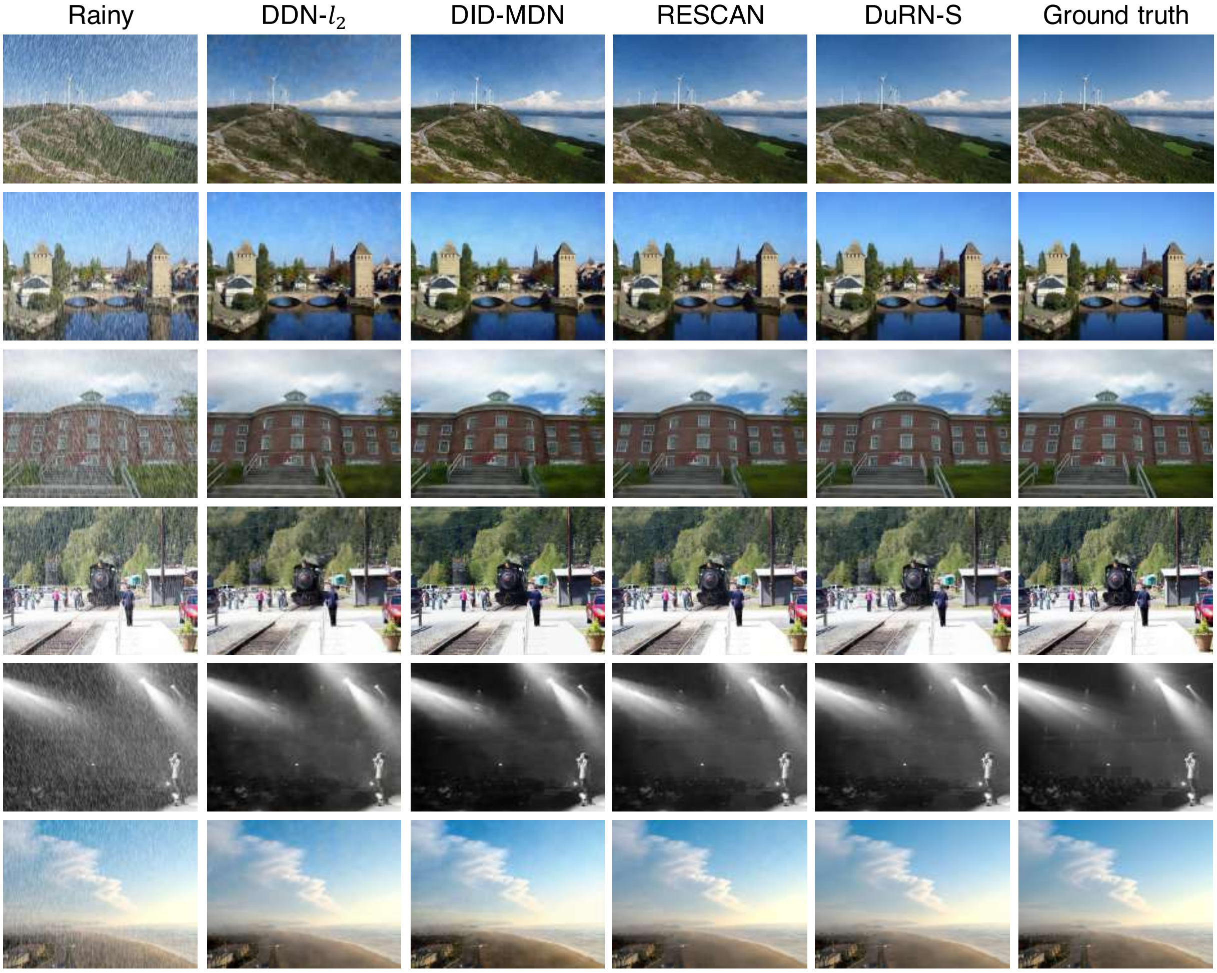


Figure A-7: Examples of deraining on synthetic rainy images.

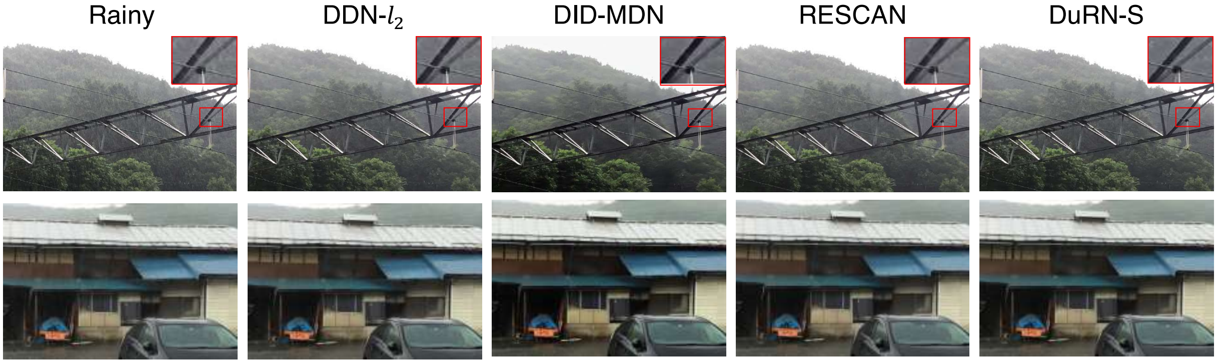


Figure A-8: Examples of deraining on real-world rainy images.

Appendix B

Appendix for the Universal Network Proposed for Jointly Solving Multiple Restoration Tasks (Chapter 3)

B.1 Details of the Encoder

B.1.1 Overall Design

The proposed network consists of the shared encoder and the task-specific decorders. The encoder is split into two parts, a stack of three convolutional layers and a stack of the proposed dual residual blocks (i.e., DuRB-M's). The initial three convolutional layers have 48, 96, and 96 channels, respectively. Each layer employs 3×3 kernels and a ReLU activation function at its output. The details of the DuRB-M are described below. The task-specific decorders are already explained in the main paper.

B.1.2 Detailed Internal Design of DuRB-M

Figure B-1 illustrates the design of our DuRB-M block. In our network, as mentioned above, multiple DuRB-M's are stacked on top of each other to form the dual residual structure, which comprise the second part of the encoder mentioned above. As explained in the main paper, the

Table B.1: The specification of ct_1^l and ct_2^l for DuRB-M for the proposed network. The "recep." denotes the receptive field of convolution, i.e., delation rate $\times$ (kernel size - 1) $+ 1$.

DuRB-M, ct_1^l

layer	kernel	dilation	recep.	stride
$ct_1^{l=1}$	3	2	5×5	1
$ct_1^{l=2}$	5	1	5×5	1
$ct_1^{l=3}$	3	2	5×5	1
$ct_1^{l=4}$	5	1	5×5	1
$ct_1^{l=5}$	7	1	7×7	1
$ct_1^{l=6}$	7	2	13×13	1
$ct_1^{l=7}$	11	1	11×11	1

DuRB-M, ct_2^l

layer	kernel	dilation	recep.	stride
$ct_2^{l=1}$	3	1	3×3	2
$ct_2^{l=2}$	3	1	3×3	2
$ct_2^{l=3}$	5	1	5×5	2
$ct_2^{l=4}$	5	1	5×5	2
$ct_2^{l=5}$	5	1	5×5	2
$ct_2^{l=6}$	5	1	5×5	2
$ct_2^{l=7}$	5	1	5×5	2

major differences from the four DuRBs (i.e., -P, -U, -S, -US) in [41] are the employment of the improved SE-ResNet module instead of a plain ResNet module (shown in the first rectangle of Fig. B-1) and the two parallel paths of different operations in the last part (the third rectangle of Fig. B-1).

Each DuRB-M in the stack has the same design except the two conv. layers c_1^l and c_2^l, which are shown in Fig. B-1. Following the network design in [41], we use different configurations for c_1^l and c_2^l for each of the stacked DuRB-M's according to its position $l(= 1, \ldots, 7)$. The parameters for c_1^l and c_2^l with different l's are shown in Table B.1. For all other components, we use the same configuration for each of the stacked DuRB-M's. We use 3×3 kernels for all other convolution layers; their stride is set to 1 except "c" right before the concatenation (Fig. B-1), where we perform 2:1 down-sampling that is paired with the up-sampling performed in "up". For the components "up" and "se", we use the same design as in [41]. The channel size is 96 throughout the stack of DuRB-M's. We don't employ any normalization layer in DuRB-M's.

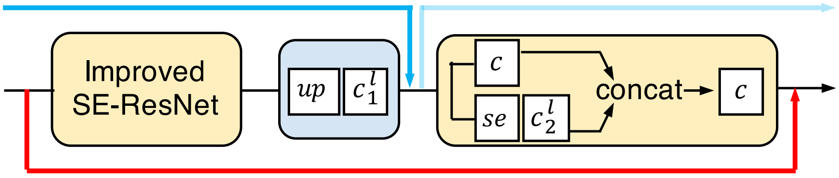
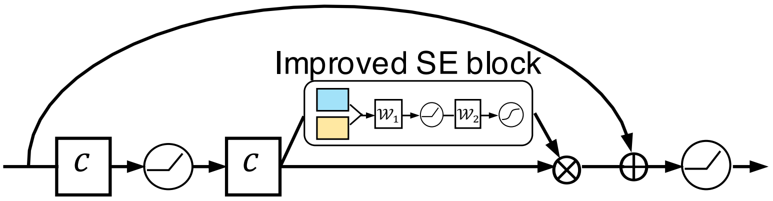
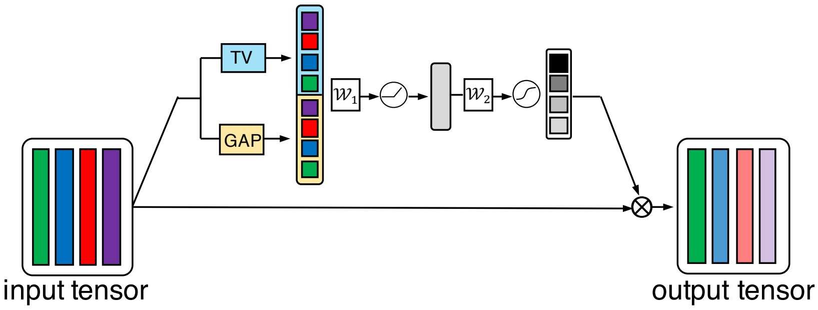


Figure B-1: The proposed building block: DuRB-M.

The improved SE-ResNet module (Fig. B-2(a)) has a bottle neck layer that can have an



(a) Improved SE-ResNet module



(b) Improved SE block

Figure B-2: (a) The improved SE-ResNet module. (b) The improved SE block inside the "improved SE-ResNet module".

arbitrary number of units (the vertical gray bar in the middle of "Improved SE Block" of Fig. B-2(b)). We set it to 64. We utilized a code[1] from a study of CNN visualization[2] for implementation of the spatial derivatives (or equivalently total variation, represented as "tv" in Fig.B-2(b)) of layer activation.

Additional Results

We show more examples of restored images for rain-streak removal, haze removal, motion blur removal, and JPEG compression noise removal, in Figs. B-3, B-4, B-5 and B-6, respectively.

[1] https://github.com/jacobgil/pytorch-explain-black-box

[2] R.C. Fong and A. Vedaldi. Interpretable Explanations of Black Boxes by Meaningful Perturbation. Proceedings of ICCV 2017.

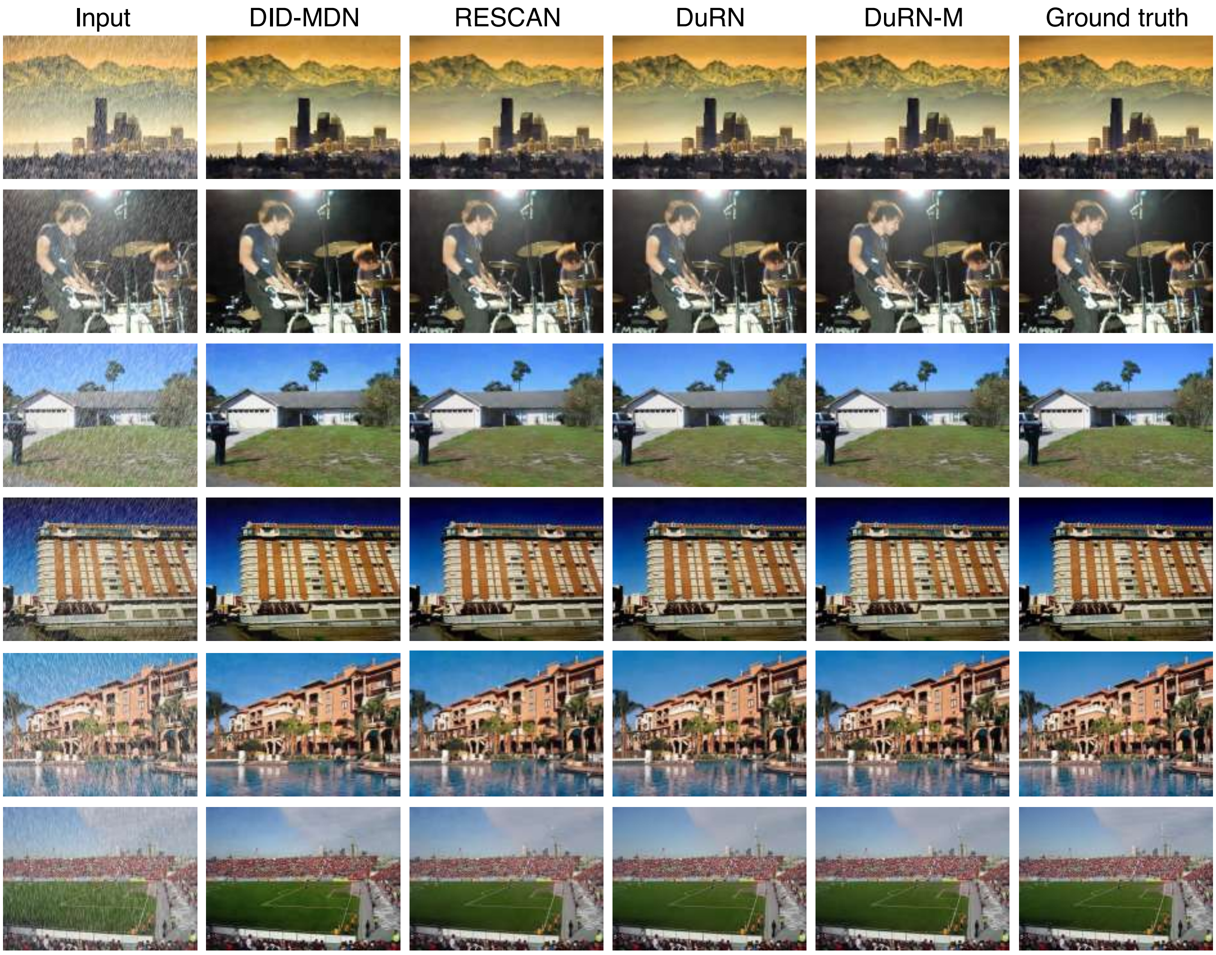


Figure B-3: Results of rain-streak removal.

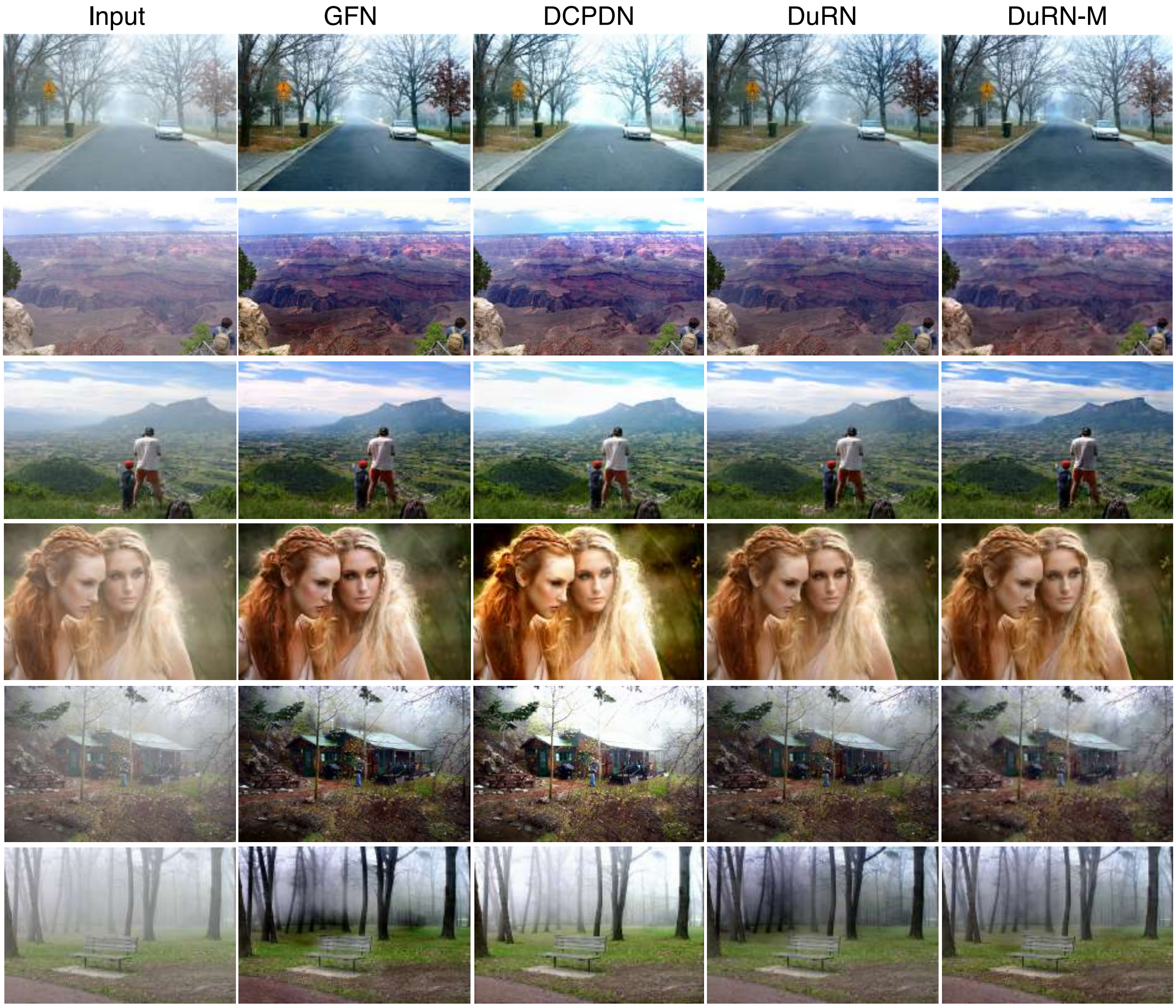


Figure B-4: Results of haze removal.

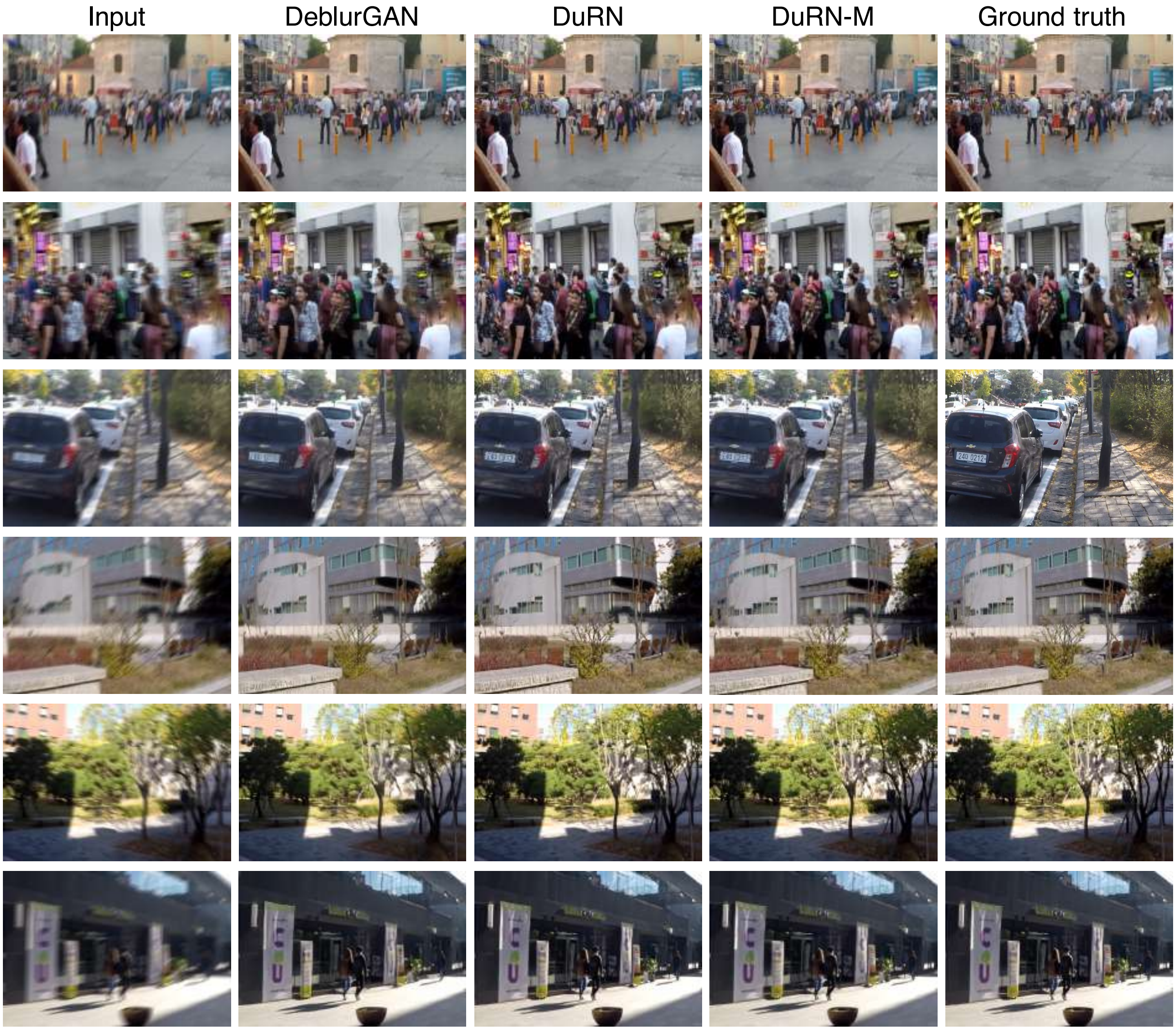


Figure B-5: Results of motion-blur removal.

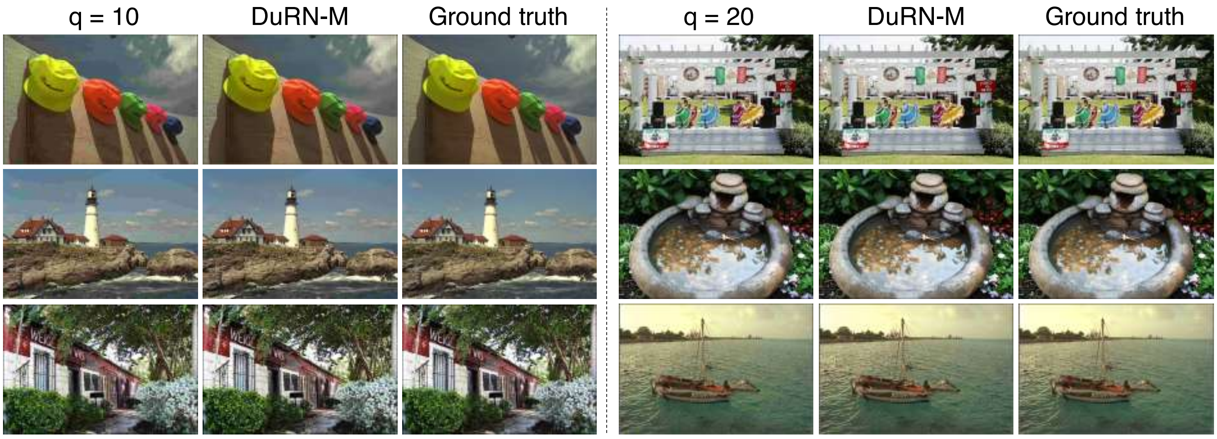


Figure B-6: Results of JPEG compression noise removal. q means compression quality.

Bibliography

[1] Martín Abadi and Other 39 authors. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Forest Agostinelli, Michael R Anderson, and Honglak Lee. Adaptive multi-column deep neural networks with application to robust image denoising. In *Proc. Conference on Neural Information Processing Systems*, 2013.

[3] Miika Aittala and Fredo Durand. Burst image deblurring using permutation invariant convolutional neural networks. In *Proc. European Conference on Computer Vision*, pages 748–764, 2018.

[4] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 6077–6086, 2018.

[5] Martin Anthony and Peter Bartlett. Neural network learning : theoretical foundations (book). 1999.

[6] Derin Babacan, Rafael Molina, Minh Do, and Aggelos Katsaggelos. Bayesian blind deconvolution with general sparse image priors. In *Proc. European Conference on Computer Vision*, pages 341–355, 2012.

[7] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[8] Dana Berman, Tali Treibitz, and Shai Avidan. Non-local image dehazing. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 1674–1682, 2016.

[9] Dana Berman, Tali Treibitz, and Shai Avidan. Air-light estimation using haze-lines. In *Proc. International Conference on Computational Photography*, pages 115–123, 2017.

[10] Marina Bloj, Daniel Kersten, and Anya Hurlbert. Perception of three-dimensional shape influences colour perception through mutual illumination. *Nature*, 402(6764):877–879, 1999.

[11] Huseyin Boyaci, Laurence Maloney, and Shari Hersh. The effect of perceived surface orientation on perceived surface albedo in binocularly viewed scenes. *Journal of Vision*, 3(8):541–553, 2003.

[12] Kristian Bredies and Martin Holler. A total variation-based jpeg decompression model. *SIAM Journal on Imaging Sciences*, 5(1):366–393, 2012.

[13] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proc. International Conference on Machine Learning*, pages 89–96, 2005.

[14] 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, 2016.

[15] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[16] Huibin Chang, Michael K Ng, and Tieyong Zeng. Reducing artifacts in jpeg decompression via a learned dictionary. *IEEE Transactions on Signal Processing*, 62(3):718–728, 2014.

[17] Fei Chen, Lei Zhang, and Huimin Yu. External patch prior guided internal clustering for image denoising. In *Proc. International Conference on Computer Vision*, pages 603–611, 2015.

[18] Yi-Lei Chen and Chiou-Ting Hsu. A generalized low-rank appearance model for spatio-temporally correlated rain streaks. In *Proc. International Conference on Computer Vision*, pages 1968–1975, 2013.

[19] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 2017.

[20] Hejin Cheong, Eunjung Chae, Eunsung Lee, Gwanghyun Jo, and Joonki Paik. Fast image restoration for spatially varying defocus blur of imaging sensor. *Sensors*, 15(1):880–898, 2015.

[21] Taco Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *Proc. International Conference on Learning Representations*, 2018.

[22] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proc. International Conference on Machine Learning*, pages 2990–2999, 2016.

[23] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007.

[24] Yubin Deng, Chen Change Loy, and Xiaoou Tang. Image aesthetic assessment: An experimental survey. *IEEE Signal Processing Magazine*, 34(4):80–106, 2016.

[25] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *Proc. International Conference on Computer Vision*, pages 576–584, 2015.

[26] Yuan Dong, Chong Huang, and Wei Liu. Rankcnn: When learning to rank encounters the pseudo preference feedback. *Computer Standards & Interfaces*, 36(3):554–562, 2014.

[27] Mark Everingham, Ali Eslami, Luc Van Gool, Christopher Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[28] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *Proc. Conference on Neural Information Processing Systems*, pages 1624–1632, 2016.

[29] Rob Fergus, Barun Singh, Aaron Hertzmann, Sam Roweis, and William Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics*, 25(3):787–794, 2006.

[30] Roland Fleming. Visual perception of materials and their properties. *Vision Research*, 94:62 – 75, 2014.

[31] Roland Fleming and Heinrich Bülthoff. Low-level image cues in the perception of translucent materials. *ACM Transactions on Applied Perception*, 2(3):346–382, 2005.

[32] Roland Fleming, Ron Dror, and Edward Adelson. Real-world illumination and the perception of surface reflectance properties. *Journal of Vision*, 3(3):347–368, 2003.

[33] Roland Fleming, Christiane Wiebel, and Karl Gegenfurtner. Perceptual qualities and material classes. *Journal of Vision*, 13(8):9–9, 2013.

[34] Xueyang Fu, Jiabin Huang, Delu Zeng, Yue Huang, Xinghao Ding, and John Paisley. Removing rain from single images via a deep detail network. *Proc. Conference on Computer Vision and Pattern Recognition*, pages 1715–1723, 2017.

[35] Zilin Gao, Jiangtao Xie, Qilong Wang, and Peihua Li. Global second-order pooling convolutional networks. In *arXiv, preprint arXiv:1811.12006*, 2018.

[36] Robert Geirhos, Carlos Medina Temme, Jonas Rauber, Heiko Schütt, Matthias Bethge, and Felix Wichmann. Generalisation in humans and deep neural networks. In *Proc. Conference on Neural Information Processing Systems*, pages 7549–7561, 2018.

[37] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *Proc. International Conference on Learning Representations*, 2019.

[38] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[39] Dong Gong, Jie Yang, Lingqiao Liu, Yanning Zhang, Ian Reid, Chunhua Shen, Anton van den Hengel, and Qinfeng Shi. From motion blur to motion flow: A deep learning solution for removing heterogeneous motion blur. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 3806–3815, 2017.

[40] Mario González, Javier Preciozzi, Pablo Musé, and Andrés Almansa. Joint denoising and decompression using cnn regularization. In *Proc. Conference on Computer Vision and Pattern Recognition Workshops*, pages 2598–2601, 2018.

[41] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. Conference on Neural Information Processing Systems*, pages 2672–2680, 2014.

[42] Muhammad Haris, Greg Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 1664–1673, 2018.

[43] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proc. International Conference on Computer Vision*, pages 2980–2988, 2017.

[44] Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2341–2353, 2011.

[45] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, 2013.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Proc. European Conference on Computer Vision*, pages 346–361, 2014.

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. International Conference on Computer Vision*, pages 1026–1034, 2015.

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[49] Noda Hideki and Niimi Michiharu. Local map estimation for quality improvement of compressed color images. *Pattern Recognition*, 44(4):788–793, 2011.

[50] Guosheng Hu, Li Liu, Yang Yuan, Zehao Yu, Yang Hua, Zhihong Zhang, Fumin Shen, Ling Shao, Timothy Hospedales, Neil Robertson, and Yongxin Yang. Deep multi-task learning to recognise subtle facial expressions of mental states. In *Proc. European Conference on Computer Vision*, pages 106–123, 2018.

[51] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. In *Proc. Conference on Neural Information Processing Systems*, pages 9423–9433, 2018.

[52] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[53] Junjie Hu, Mete Ozay, Yan Zhang, and Takayuki Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. *Proc. Winter Conference on Applications of Computer Vision*, pages 1043–1051, 2018.

[54] Yang Hu, Guihua Wen, Mingnan Luo, Dan Dai, and Jiajiong Ma. Competitive inner-imaging squeeze and excitation for residual network. In *arXiv, preprint arXiv:1807.08920*, 2018.

[55] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 2261–2269, 2017.

[56] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. International Conference on International Conference on Machine Learning*, pages 448–456, 2015.

[57] Phillip Isola, Jianxiong Xiao, Devi Parikh, Antonio Torralba, and Aude Oliva. What makes a photograph memorable? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1469–1482, 2014.

[58] Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. In *Proc. Conference on Neural Information Processing Systems*, pages 769–776, 2009.

[59] Li-Wei Kang, Chia-Wen Lin, and Yu-Hsiang Fu. Automatic single-image-based rain streaks removal via image decomposition. *IEEE Transactions on Image Processing*, 21(4):1742–1755, 2012.

[60] Ramesh Kanthan and Naganandini Sujatha. Rain drop detection and removal using k-means clustering. In *Proc. International Conference on Computational Intelligence and Computing Research*, pages 1–5, 2015.

[61] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018.

[62] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. International Conference on Learning Representations*, 2015.

[63] Idan Kligvasser, Tamar Rott Shaham, and Tomer Michaeli. xunit: Learning a spatial activation function for efficient image restoration. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 2433–2442, 2018.

[64] Shu Kong, Xiaohui Shen, Zhe Lin, Radomir Mech, and Charless Fowlkes. Photo aesthetics ranking network with attributes and content adaptation. In *Proc. European Conference on Computer Vision*, pages 662–679, 2016.

[65] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Conference on Neural Information Processing Systems*, pages 1097–1105, 2012.

[66] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiri Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 8183–8192, 2018.

[67] Hiroyuki Kurihata, Tatsuro S Takahashi, Ichiro Ide, Yoshito Mekada, Hiroshi Murase, Yukimasa Tamatsu, and Takayuki Miyahara. Rainy weather recognition from in-vehicle camera images for driver assistance. In *Proc. Intelligent Vehicles Symposium*, pages 205–210, 2005.

[68] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[69] Boyi Li, Xiulian Peng, Zhangyang Wang, Ji-Zheng Xu, and Dan Feng. Aod-net: All-in-one dehazing network. In *Proc. International Conference on Computer Vision*, pages 4780–4788, 2017.

[70] Boyi Li, Xiulian Peng, Zhangyang Wang, Jizheng Xu, and Dan Feng. Aod-net: All-in-one dehazing network. In *Proc. International Conference on Computer Vision*, pages 4780–4788, 2017.

[71] Boyi Li, Wenqi Ren, Dengpan Fu, Dacheng Tao, Dan Feng, Wenjun Zeng, and Zhangyang Wang. Reside: A benchmark for single image dehazing. In *arXiv, preprint arXiv:1712.04143*, 2017.

[72] Guanbin Li, Xiang He, Wei Zhang, Huiyou Chang, Le Dong, and Liang Lin. Non-locally enhanced encoder-decoder network for single image de-raining. In *Proc. ACM International Conference on Multimedia*, pages 1056–1064, 2018.

[73] Kunpeng Li, Ziyan Wu, Kuan-Chuan Peng, Jan Ernst, and Yun Fu. Tell me where to look: Guided attention inference network. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 9215–9223, 2018.

[74] Runde Li, Jinshan Pan, Zechao Li, and Jinhui Tang. Single image dehazing via conditional generative adversarial network. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 8202–8211, 2018.

[75] Xia Li, Jianlong Wu, Zhouchen Lin, Hong Liu, and Hongbin Zha. Recurrent squeeze-and-excitation context aggregation net for single image deraining. In *Proc. European Conference on Computer Vision*, pages 262–277, 2018.

[76] Yu Li, Robby Tan, Xiaojie Guo, jiangbo Lu, and Michael Brown. Rain streak removal using layer priors. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 2736–2744, 2016.

[77] Drew Linsley, Dan Scheibler, Sven Eberhardt, and Thomas Serre. Global-and-local attention networks for visual recognition. In *arXiv, preprint arXiv:1805.08819*, 2018.

[78] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander Berg. Ssd: Single shot multibox detector. In *Proc. European Conference on Computer Vision*, pages 21–37, 2016.

[79] Xing Liu and Takayuki Okatani. Evaluating artificial systems for pairwise ranking tasks sensitive to individual differences. In *arXiv preprint arXiv:1905.13560*, 2019.

[80] Xing Liu, Mete Ozay, Yan Zhang, and Takayuki Okatani. Learning deep representations of objects and materials for material recognition. In *Proc. Annual Meeting of the Vision Sciences Society*, 2016.

[81] Xing Liu, Masataka Sawayama, Ryusuke Hayashi, Mete Ozay, Takayuki Okatani, and Shin'ya Nishida. Perturbation tolerance of deep neural networks and humans in material recognition. In *Proc. Annual Meeting of the Vision Sciences Society*, 2018.

[82] Xing Liu, Masanori Suganuma, Zhun Sun, and Takayuki Okatani. Dual residual networks leveraging the potential of paired operations for image restoration. In *Proc. Conference on Computer Vision and Pattern Recognition*, 2019.

[83] Yang Liu, Zhaowen Wang, Hailin Jin, and Ian Wassell. Multi-task adversarial network for disentangled feature learning. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 3743–3751, 2018.

[84] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Proc. Conference on Neural Information Processing Systems*, pages 2802–2810, 2016.

[85] Luca Marchesotti, Florent Perronnin, Diane Larlus, and Gabriela Csurka. Assessing the aesthetic quality of photographs using generic image descriptors. In *Proc. International Conference on Computer Vision*, pages 1784–1791, 2011.

[86] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. International Conference on Computer Vision*, pages 416–425, 2001.

[87] Gaofeng Meng, Ying Wang, Jiangyong Duan, Shiming Xiang, and Chunhong Pan. Efficient image dehazing with boundary constraint and contextual regularization. In *Proc. International Conference on Computer Vision*, pages 617–624, 2013.

[88] James Miskin and David MacKay. Ensemble learning for blind image separation and deconvolution. In *Book of Advances in Independent Component Analysis*, pages 123–141. 2000.

[89] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 86–94, 2017.

[90] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[91] Isamu Motoyoshi, Shin'ya Nishida, Lavanya Sharan, and Edward Adelson. Image statistics and the perception of surface qualities. *Nature*, 447(7141):206–209, 2007.

[92] Naila Murray, Luca Marchesotti, and Florent Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 2408–2415, 2012.

[93] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 257–265, 2017.

[94] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. International Conference on International Conference on Machine Learning*, pages 807–814, 2015.

[95] Duy-Kien Nguyen and Takayuki Okatani. Multi-task learning of hierarchical vision-language representation. In *Proc. Conference on Computer Vision and Pattern Recognition*, 2019.

[96] Shin'ya Nishida and Mikio Shinya. Use of image-based information in judgments of surface-reflectance properties. *Journal of the Optical Society of America A*, 15(12):2951–2965, 1998.

[97] Jinshan Pan, Zhe Hu, Zhixun Su, and Ming-Hsuan Yang. Deblurring text images via l0-regularized intensity and gradient prior. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 2901–2908, 2014.

[98] Devi Parikh and Kristen Grauman. Relative attributes. In *Proc. International Conference on Computer Vision*, pages 503–510, 2011.

[99] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Proc. Conference on Neural Information Processing Systems*

Workshop: The Future of Gradient-based Machine Learning Software and Techniques, 2017.

[100] Rui Qian, Robby Tan, Wenhan Yang, Jiajun Su, and Jiaying Liu. Attentive generative adversarial network for raindrop removal from a single image. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 2482–2491, 2018.

[101] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. In *arXiv, preprint arXiv:1804.02767*, 2018.

[102] Wenqi Ren, Si Liu, Hua Zhang, Jinshan Pan, Xiaochun Cao, and Ming-Hsuan Yang. Single image dehazing via multi-scale convolutional neural networks. In *Proc. European Conference on Computer Vision*, pages 154–169, 2016.

[103] Wenqi Ren, Lin Ma, Jiawei Zhang, Jinshan Pan, Xiaochun Cao, Wei Liu, and Ming-Hsuan Yang. Gated fusion network for single image dehazing. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 3253–3261, 2018.

[104] Rocco Robilotto and Qasim Zaidi. Limits of lighness identification for real objects under natural viewing conditions. *Journal of Vision*, 4(9):779–797, 2004.

[105] Martin Roser and Andreas Geiger. Video-based raindrop detection for improved image registration. In *Proc. International Conference on Computer Vision Workshops*, pages 570–577, 2009.

[106] Stefan Roth and Michael Black. Fields of experts: a framework for learning image priors. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 860–867, 2005.

[107] Leonid Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Journal of Physics D*, 60(1-4):259–268, 1992.

[108] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[109] Masataka Sawayama, Edward Adelson, and Shin'ya Nishida. Visual wetness perception based on image color statistics. *Journal of Vision*, 17(5):1–24, 2017.

[110] Masataka Sawayama and Shin'ya Nishida. Material and shape perception based on two types of intensity gradient information. *PLoS Computational Biology*, 14(4), 2018.

[111] Andrew Saxe, James McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proc. International Conference on Learning Representations*, 2014.

[112] Gabriel Schwartz and Ko Nishino. Automatically discovering local visual material attributes. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 3565–3573, 2015.

[113] Qi Shan, Jiaya Jia, and Aseem Agarwala. High-quality motion deblurring from a single image. *ACM Transactions on Graphics*, 27(3):73, 2008.

[114] Lavanya Sharan, Ce Liu, Ruth Rosenholtz, and Edward Adelson. Recognizing materials using perceptually inspired features. *International Journal of Computer Vision*, 108(3):348–371, 2013.

[115] Lavanya Sharan, Ce Liu, Ruth Rosenholtz, and Edward Adelson. Recognizing materials using perceptually inspired features. *International Journal of Computer Vision*, 103(3):348–371, 2013.

[116] Lavanya Sharan, Ruth Rosenholtz, and Edward Adelson. Accuracy and speed of material categorization in real-world images. *Journal of Vision*, 14(9):12–12, 2014.

[117] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016.

[118] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. International Conference on Learning Representations*, 2015.

[119] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[120] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. In *arXiv, preprint arXiv:1710.08864*, 2017.

[121] Masanori Suganuma, Xing Liu, and Takayuki Okatani. Attention-based adaptive selection of operations for image restoration in the presence of unknown combined distortions. In *Proc. Conference on Computer Vision and Pattern Recognition*, 2019.

[122] Masanori Suganuma, Mete Ozay, and Takayuki Okatani. Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search. In *Proc. International Conference on Machine Learning*, pages 4778–4787, 2018.

[123] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 769–777, 2015.

[124] Zhun Sun, Mete Ozay, Yan Zhang, Xing Liu, and Takayuki Okatani. Feature quantization for defending against distortion of images. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 7957–7966, 2018.

[125] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper

with convolutions. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[126] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *Proc. International Conference on Computer Vision*, pages 4549–4557, 2017.

[127] Radu Timofte and Other 76 authors. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Proc. Conference on Computer Vision and Pattern Recognition Workshops*, pages 1110–1121, 2017.

[128] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. In *arXiv, preprint arXiv:1607.08022*, 2016.

[129] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[130] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Proc. Conference on Neural Information Processing Systems*, pages 550–558, 2016.

[131] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 6450–6458, 2017.

[132] Patrick Wieschollek, Michael Hirsch, Bernhard Schölkopf, and Hendrik Lensch. Learning blind motion deblurring. In *Proc. International Conference on Computer Vision*, pages 231–240, 2017.

[133] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proc. European Conference on Computer Vision*, pages 3–19, 2018.

[134] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Proc. Conference on Neural Information Processing Systems*, pages 350–358, 2012.

[135] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 675–684, 2018.

[136] Jun Xu, Hui Li, Zhetong Liang, David Zhang, and Lei Zhang. Real-world noisy image denoising: A new benchmark. In *arXiv, preprint arXiv:1804.02603*, 2018.

[137] Jun Xu, Lei Zhang, and David Zhang. A trilateral weighted sparse coding scheme for real-world image denoising. In *Proc. European Conference on Computer Vision*, pages 21–38, 2018.

[138] Jun Xu, Lei Zhang, Wangmeng Zuo, David Zhang, and Xiangchu Feng. Patch group based nonlocal self-similarity prior learning for image denoising. In *Proc. International Conference on Computer Vision*, pages 244–252, 2015.

[139] Li Xu and Jiaya Jia. Two-phase kernel estimation for robust motion deblurring. In *Proc. European Conference on Computer Vision*, pages 157–170, 2010.

[140] Li Xu, Shicheng Zheng, and Jiaya Jia. Unnatural L0 sparse representation for natural image deblurring. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 1107–1114, 2013.

[141] Xiangyu Xu, Deqing Sun, Jinshan Pan, Yujin Zhang, Hanspeter Pfister, and Ming-Hsuan Yang. Learning to super-resolve blurry face and text images. In *Proc. International Conference on Computer Vision*, pages 251–260, 2017.

[142] Atsushi Yamashita, Yuu Tanaka, and Toru Kaneko. Removal of adherent waterdrops from images acquired with stereo camera. In *Proc. International Conference on Intelligent Robots and Systems*, pages 400–405, 2005.

[143] Yan Yan, Chenliang Xu, Dawen Cai, and Jason Corso. Weakly supervised actor-action segmentation via robust multi-task ranking. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 1022–1031, 2017.

[144] Dong Yang and Jian Sun. Proximal dehaze-net: A prior learning-based deep network for single image dehazing. In *Proc. European Conference on Computer Vision*, pages 729–746, 2018.

[145] Wenhan Yang, Robby Tan, Jiashi Feng, Jiaying Liu, Zongming Guo, and Shuicheng Yan. Joint rain detection and removal from a single image. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 1685–1694, 2017.

[146] Yongyi Yang, Nikolas Galatsanos, and Aggelos Katsaggelos. Regularized reconstruction to reduce blocking artifacts of block discrete cosine transform compressed images. *IEEE Transactions on Circuits and Systems for Video Technology*, 3(6):421–432, 1993.

[147] Jaeyoung Yoo, Sang-ho Lee, and Nojun Kwak. Image restoration by estimating frequency distribution of local patches. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 6684–6692, 2018.

[148] Quanzeng You, Jiebo Luo, Hailin Jin, and Jianchao Yang. Building a large scale dataset for image emotion recognition: The fine print and the benchmark. In *Proc. International Conference on Association for the Advancement of Artificial Intelligence*, 2016.

[149] Shaodi You, Robby Tan, Rei Kawakami, Yasuhiro Mukaigawa, and Katsushi Ikeuchi. Adherent raindrop modeling, detectionand removal in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1721–1733, 2016.

[150] Ke Yu, Chao Dong, Liang Lin, and Change Loy Chen. Crafting a toolchain for image restoration by deep reinforcement learning. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 2443–2452, 2018.

[151] He Zhang and Vishal Patel. Densely connected pyramid dehazing network. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 3194–3203, 2018.

[152] He Zhang and Vishal Patel. Density-aware single image de-raining using a multi-stream dense network. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 695–704, 2018.

[153] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[154] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for CNN based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[155] Xinyi Zhang, Hang Dong, Zhe Hu, Wei-Sheng Lai, Fei Wang, and Ming-Hsuan Yang. Gated fusion network for joint image deblurring and super-resolution. In *arXiv preprint arXiv:1807.10806*, 2018.

[156] Yan Zhang, Mete Ozay, Xing Liu, and Takayuki Okatani. Integrating deep features for material recognition. *Proc. International Conference on Pattern Recognition*, pages 3697–3702, 2016.

[157] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proc. European Conference on Computer Vision*, pages 294–310, 2018.

[158] Yulun Zhang, Kunpeng Li, Kai Li, Bineng Zhong, and Yun Fu. Residual non-local attention networks for image restoration. In *Proc. International Conference on Learning Representations*, 2019.

[159] Zhenglong Zhou and Chaz Firestone. Humans can decipher adversarial images. *Nature Communications*, 10(1):877–879, 2019.

[160] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Proc. International Conference on Computer Vision*, pages 2242–2251, 2017.

[161] Qingsong Zhu, Jiaming Mai, and Ling Shao. A fast single image haze removal algorithm using color attenuation prior. *IEEE Transactions on Image Processing*, 24(11):3522–3533, 2015.

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisor Professor Takayuki Okatani. I would like to thank him for supporting my research with his great passion, strong responsibility and immense knowledge. He has been setting a perfect example of being a good researcher by his own behaviour. I have learned a lot from him. In addition, I would like to thank Professor Mete Ozay for teaching me lots of mathematics of machine learning, computer vision and etc.. I also would like to thank him for his time for weekly discussing my research progresses with me.

Secondly, I would like to thank Professor Kota Yamaguchi for building the powerful computing system in our laboratory. I also would like to thank Dr. Masaki Saito, Dr. Zhun Sun, Professor Masanori Suganuma, Nimar Blume, Davis Sorenson for maintaining the system.

Finally, I would like to thank Professor Shin'ya Nishida (Kyoto university), Dr. Masataka Sawayama (NTT Communication Science Laboratories) and Dr. Ryusuke Hayashi (Systems Neuroscience Group, AIST) for their kind help for my study in the SHITSUKAN project. I also would like to thank Professor Masanori Suganuma, Dr. Yan Zhang, Dr. Zhun Sun, Yuta Shirakawa, Hiroto Date, Taro Hatsutani, Rito Murase, Yusuke Hosoya, and other members in our laboratory, for accompanying me while I am studying in Japan for my Master's and Doctoral degrees.