이학박사 학위논문

Study on Loss Surface of Deep Neural Networks and Several Applications of Deep Learning

심층 신경망의 손실표면 및 딥러닝의 여러 적용에 관한 연구

2022년 8월

서울대학교 대학원

수 리 과 학 부

박 예 찬

Study on Loss Surface of Deep Neural Networks and Several Applications of Deep Learning

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Yeachan Park

Dissertation Director : Professor Myungjoo Kang

Department of Mathematical Sciences
Seoul National University

August 2022

Study on Loss Surface of Deep Neural Networks and Several Applications of Deep Learning

심층 신경망의 손실표면 및 딥러닝의 여러 적용에 관한 연구

지도교수 강 명 주

이 논문을 이학박사 학위논문으로 제출함

2022년 4월

서울대학교 대학원

수 리 과 학 부

박 예 찬

박 예 찬의 이학박사 학위논문을 인준함

2022년 6월

위 원 장 국 웅 (인)

부 위 원 장 강 명 주 (인)

위 원 Ernest K. Ryu (인)

위 원 곽 지 훈 (인)

위 원 이 병 준 (인)

Abstract

Study on Loss Surface of Deep Neural Networks and Several Applications of Deep Learning

Yeachan Park

Department of Mathematical Sciences

The Graduate School

Seoul National University

In this thesis, we study the loss surface of deep neural networks. Does the loss function of deep neural network have no bad local minimum like the convex function? Although it is well known for piece-wise linear activations, not much is known for the general smooth activations. We explore that a bad local minimum also exists for general smooth activations. In addition, we characterize the types of such local minima. This provides a partial explanation for the understanding of the loss surface of deep neural networks. Additionally, we present several applications of deep neural networks in learning theory, private machine learning, and computer vision.

Key words: Deep learning, neural network, local minimum

Student Number: 2015-22567

Contents

Chapter 1

Introduction

"There is Nothing More Practical Than A Good Theory."

- Kurt Lewin,

Modern machine learning with neural networks as shown remarkable results in many real-world applications. However, little is known about the theoretical foundation of how the neural network works. In particular, most of modern machine learning models rely on gradient descent-based optimization algorithm which minimize the difference between the output of neural network and the target function. In this thesis, we present and discuss the mathematical foundation of deep neural network. Generally, universal approximation theorems [17, 45, 57, 60] state that deep neural network can approximate any continuous functions. Thanks to the universal approximation property, deep neural networks can approximate any continuous function, and gradient-based optimization algorithms such as stochastic gradient descent (SGD), can realize this. For convex loss function, we can guarantee that SGD can converge to the global minimum of the loss [79]. However, since the

output and the loss of the deep neural network is highly non-convex, such convergence may not be guaranteed anymore. In practice, since the loss function is very non-convex, such loss surface may have local minima which hinder the convergence to the global minimum. Therefore, several methods enabling stable convergence to the global minimum without falling into the local minimum are studied.

However, in this context, an important fundamental question arises. Does the bad local minimum really exist? Here, a bad minimum means minimum which is not global. A bad minimum is also called sub-optimal, spurious, harmful, etc. Not much is known about the existence of a local minimum in deep neural networks. We investigate the existence of a bad local minimum in deep neural network with smooth activation functions. We present the results in Chapter 2 and summarize the key ingredients below.

- For partially linear activation, we show that a local minimum exists in 2-layer neural network.
- For general smooth activation and small sample size ($N = 1, 2$), we show that a bad local minimum exists in the 2-layer narrow network.
- For general smooth activation with some assumptions, we show that a bad non-attracting local minimum exists in the 2-layer neural network.
- For general smooth activation with some additional assumptions, we show that a bad non-attracting local minimum exists in the 3-layer neural network.

In the following chapters, we present several applications of deep neural networks. In Chapter 3, we present a novel self-knowledge distillation technique using dropout. In Chapter 4, we present a novel membership inference attack methods agains object detection models.

In Chapter 5, we present a novel single image de-raining models to effectively remove rain streaks from the rainy images.

Chapter 2

Existence of local minimum in neural network

2.1 Introduction

Modern machine learning with neural networks as shown remarkable results in many real-world applications. However, little is known about the theoretical foundation of how the neural network works. In particular, most of modern machine learning models rely on gradient descent-based optimization algorithm which minimize the difference between the output of neural network and the target function. In this context, understanding of the loss surface of neural networks is of fundamental importance.

The question of the existence of a local minimum is also very important, because it provides whether the gradient descent-based algorithms can stably reach the global minimum without falling into the local minimum. For convex loss function, it is widely known that the loss surface has a unique global minimum. For general

neural network, it is not easy to investigate the loss surface because of its strong non-convexity.

Several works suggest that there exists no bad local minimum in deep linear network. Kawaguchi [53] and Lu & Kawaguchi [71] show that there are only global minima and saddle point in deep linear network with squared error. Zhou & Liang [134] provides a analytic formulation of critical points in deep linear network. Laurent & von Brecht show that every local minimum of deep linear network is global under any differentiable convex loss function.

On the other hand, existence of a bad local minimum has reported in deep nonlinear network. Yun *et al.* [124] and He *et al.* [39] show that a bad local minimum exists in the neural network with piece-wise linear activations. For smooth activation functions, Petzka *et al.* [84] and Ding *et al.* [22] show that a bad local minimum exists in the deep neural network with sigmoid activation functions. However, existence of a bad local minimum for general smooth activations remains unclear.

In this context, we find that a bad local minimum exists in the 2-layer neural network with several smooth activation functions that satisfy some conditions. In addition, we show that the found local minimum is of non-attracting type.

We organize this chapter as follows. We first show that we can construct a local minimum in deep neural network with partially linear activation by borrowing parameters from the linear model. Next, we show that the borrowing from the linear model technique is not applicable to activation without linearity. That is, the borrowed parameters may not be a local minimum for the general smooth activation. To study the existence of the local minimum in the general smooth activation, we explore the case with small the sample size N. For $N = 1$, we show that no local minimum exits in the shallow $1 - 1 - 1$ network. Moreover, for $N = 2$, we

also show that no local minimum exists in the shallow $1-1-1$ network. Then for $N = 7$ and L^2 loss, we find that there exists a strict local minimum in the shallow $1-1-1$ network for generic X. We extend this local minimum to the $1-d_1-1$ network using local minimum embedding. Because of the property of the local minimum embedding, we can conclude that this constructed local minimum is non-attracting. Moreover, for $N \geq 29$, we find that shallow 3-layer $1-1-1-1$ network has a strict a local minimum for generic X. Similar to 2-layer network, we extend this local minimum to the wide $1-d_1-d_2-1$ network using local minimum embedding.

2.2 Local Minima and Deep Neural Network

2.2.1 Notation and Model

We begin by defining the notation. Let L be a the number of layers. Let (X, Y) be the training dataset with $X \in \mathbb{R}^{d_X \times N}$, and $Y \in \mathbb{R}^{d_Y \times N}$, where N is the number of samples. d_X and d_Y denote the dimension of the inputs and outputs, respectively. $d_1, d_2, ...d_{L-1}$ denote the width of the i-th layer.

2.2.2 Local Minima and Deep Linear Network

First, Goodfellow *et al.* [32] remark that Baldi & Hornik [3] show that every local minimum is a global minimum for shallow linear networks.

Proposition 2.1 (Baldi & Hornik [3])**.** *Consider shallow linear network with* $L = 2$*:*

$$F(W, X) = W_2 W_1 X.$$

Assume XX^T and XY^T are invertible. Assume $\Sigma := YX^T(XX^T)^{-1}XY^T$ has d_Y distinct eigenvalues and $d_1 < d_X = d_Y$. Every local minimum is a global minimum for the L^2 loss function $\mathcal{L}(W)$.

Kawaguchi [53] extends this result by showing that every local minimum is a global minimum in the deep linear network.

Proposition 2.2 (Kawaguchi [53])**.** *Consider shallow linear network with layer L:*

$$F(W, X) = W_L W_{L-1} ... W_2 W_1 X.$$

Assume XX^T and XY^T are of full rank with $d_Y \leq d_X$ and $\Sigma := YX^T(XX^T)^{-1}XY^T$ has d_Y distinct eigenvalues. Then every local minimum is a global minimum for the L^2 loss function $\mathcal{L}(W)$.

Lu & Kawaguchi [71] advance the result by relaxing the assumption.

Proposition 2.3 (Lu & Kawaguchi [71])**.** *Consider shallow linear network with layer L:*

$$F(W, X) = W_L W_{L-1} ... W_2 W_1 X.$$

Assume X and Y are full rank. Then every local minimum is a global minimum the L^2 loss function $\mathcal{L}(W)$.

Laurent & Brecht [59] show that every local minimum is global even for any convex loss function.

Proposition 2.4 (Laurent & Brecht [59])**.** *Consider shallow linear network with layer L:*

$$F(W, X) = W_L W_{L-1} ... W_2 W_1 X.$$

and the loss function

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} \ell(F(W, x_i), y_i).$$

Suppose ℓ *is convex and differentiable and* $\max\{d_X, d_Y\} \leq \min_{i=1}^{N-1} d_i$. *Then every local minimum is a global minimum.*

2.2.3 Local Minima and Deep Neural Network with piece-wise linear activations

So far, it is known that a bad (not global) local minimum does not exist in the linear network. On the other hand, the opposite result is known in deep neural network with non-linear activations. We introduce several results on existence of bad local minima in neural networks with piece-wise linear activations.

Yun *et al.* [124] shows that bad local minima exist in the deep nueral network with two-piece linear activation.

Proposition 2.5 (Yun *et al.* [124])**.** *Consider the 1-hidden layer neural network with* L^2 *loss:*

$$\hat{y} = W_2 h(W_1 X + b_1 \mathbf{1}_N^T) + b_2 \mathbf{1}_N^T, \ell(W, b) = \frac{1}{2}\|\hat{y} - Y\|_F^2 \tag{2.1}$$

where h *is a nonlinear activation function:*

$$h_{s_+,s_-}(x) = \max(s_+ x, 0) + \min(s_- x, 0) \tag{2.2}$$

where $s_+, s_- \geq 0$ *and* $s_+ \neq s_-$. *Suppose that the following conditions hold:*

(1) $d_Y = 1$ *and linear models* RX *cannot perfectly fit* Y.

(2) All data points x_i*'s are distinct*

(3) The hidden layer has at least width 2: $d_1 \geq 2$.

Then there exists infinitely many spurious local minima and whose losses are same as linear model.

This indicates that a small non-linearity including ReLU, can create a bad local minimum. Since two-piece linear activation functions, including ReLU, are used very widely, this indicates that most of the deep neural networks can have bad local minima.

He *et al.* [39] generalize the results of Yun *et al.* [124] by improving two-piece linear to piece-wise linear activation function, 2-layer neural network to general N-layer deep neural network, and L^2 loss to convex loss function.

Proposition 2.6 (He *et al.* [39])**.** *Consider a deep neural network with piecewise linear activation* $\sigma(x)$ *and* L *layers:*

$$F^{(j)} = \sigma(W_j F^{(j-1)} + b_j \mathbf{1}_N^T),\ j = 1, 2, ..., L-1$$

$$F^{(L)} = W_L F^{(L-1)} + b_L \mathbf{1}_N^T.$$

Suppose that the following assumptions hold:

(1) The training data cannot be fit by a linear model.

(2) All data points are distinct.

(3) $d_i > d_Y$ *for* $i = 1, ..., N-1$.

(4) For piece-wise linear activation functions, there exists some turning point that sum of the slops on the two slides does not equal to 0.

Then there exists infinitely many bad local minima under any differentiable convex loss.

2.2.4 Local Minima and Deep Neural Network with smooth activations

Although numerous researches reveal the existence of bad local minima in deep neural networks with piece-wise linear activations, but not much is yet known for general smooth activation functions. Since neural networks with various smooth activations other than ReLU are currently widely used, the study of the existence of bad local minimum for smooth activations is of great importance. Some results are known for sigmoid activations where sigmoid is defined as:

$$s(x) = \frac{1}{1 + e^{-x}}.$$

We classify the local minima into two types.

Definition 2.7 (Sprinkhuizen-Kuyper & Boers [94])**.** *Let $L : \mathbb{R}^n \to \mathbb{R}$ be a differentiable loss function. Let R be a connected component of local minima of $L(w)$ such that $\forall w \in R$, w is a local minimum and with value $L(w) = c$.*

- *R is called an attracting region of local minima, if there is a neighborhood U of R such that every non-increasing continuous path $w(t)$ in U, which starts from $w(0) \in R$, end in R.*
- *R is called an non-attracting region of local minima, if every neighborhood U of R contains a non-increasing continuous path $w(t)$ in U, which starts from $w(0) \in R$, end in a point $w(1)$ with $L(w(1)) < c$.*

Petzka *et al.* [84] shows that a non-attracting region of bad local minimum exist in the deep neural network with sigmoid activations.

Proposition 2.8 (Petzka *et al.* [84]). *Consider a deep neural network with sigmoid activation $\sigma(x)$ and L layers:*

$$F^{(j)} = \sigma(W_j F^{(j-1)} + b_j \mathbf{1}_N^T),\ j = 1, 2, ..., L-1 \tag{2.3}$$

$$F^{(L)} = W_L F^{(L-1)} + b_L \mathbf{1}_N^T. \tag{2.4}$$

Then, there exists a dataset (X, Y) such that L^2 loss function has a non-attracting region of local minima.

Proposition 2.9 (Ding *et al.* [22]). *Consider 2-layer network with $d_0 = d_2 = 1$, $N \geq 7$ and sigmoid activation. Then for generic $X \in \mathbb{R}^N$, there exists a positive measure of $Y \in \mathbb{R}^N$ such that the L^2 loss function of the network has a bad local minimum.*

2.2.5 Local Valley and Deep Neural Network

Definition 2.10 (Spurious Valley). *For $c \in \mathbb{R}$, define the sub-level set of L as $\Omega_L(c) = \{\theta \in \Theta : L(\theta) \leq c\}$. We define a spurious valley as a path-connected component of a sub-level set $\Omega_L(c)$ which does not contain a global minimum of the loss $L(\theta)$*

Lemma 2.11. *For any initial parameter $\theta_0 \in \Theta$, suppose there exists a continuous path $\theta_t \in \Theta,\ t \in [0, 1]$ such that*

- $\theta_1 \in \operatorname{argmin} L(\theta)$
- *The function $t \mapsto L(\theta_t)$ is non-increasing.*

Then this is implies that there is no spurious valley.

Proposition 2.12 (Venturi *et al.* [101]). *For any continuous function σ and r.v. $\mathbf{X}$ with finite upper intrinsic dimension $dim^*(\sigma, \mathbf{X})$, For one-hidden-layer NN $\Phi(x;\theta) = U\sigma(Wx)$, $\theta = (U, W) \in (\mathbb{R}^{m\times p}, \mathbb{R}^{p\times n})$, the empirical loss function*

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(\Phi(x;\theta), y_i). \tag{2.5}$$

admits no spurious valleys in the over-parametrized regime $p \geq dim^*(\sigma, \mathbf{X})$. Consider N data point $\{(x_i, y_i)\}_{i=1}^N \in \mathbb{R}^n \times \mathbb{R}^m$, then we already have $dim^*(\sigma, \mathbf{X}) \leq dim(L^2_{\mathbf{X}}) \leq N$.

Proposition 2.13 (Nguyen [81]). *Consider a deep neural network with smooth linear activation $\sigma(x)$ and L layers:*

$$F^{(j)} = \sigma(W_j F^{(j-1)} + b_j \mathbf{1}_N^T),\ j = 1, 2, ..., L-1 \tag{2.6}$$

$$F^{(L)} = W_L F^{(L-1)} + b_L \mathbf{1}_N^T. \tag{2.7}$$

Suppose $\sigma(x)$ is strictly increasing and $\sigma(\mathbb{R}) = \mathbb{R}$. Assume $d_X \geq N$,, $d_1 > d_2 > ... > d_L$, and $rank(X) = N$, then every sub-level set is connected.

2.3 Existence of local minimum for partially linear activations

Consider the problem :

$$X \in \mathbb{R}^{d_X \times N}, Y \in \mathbb{R}^{d_Y \times N}, \tag{2.8}$$

and define $\tilde{X}_i = \begin{bmatrix} X_i \\ 1 \end{bmatrix}$.

Consider the 2-layer neural network for $[W_1, b_1, W_2, b_2] \in [\mathbb{R}^{d_1 \times d_X}, \mathbb{R}^{d_1}, \mathbb{R}^{d_Y \times d_1}, \mathbb{R}^{d_Y}]$:

$$\hat{Y}_i = W_2\sigma(W_1X + b_1) + b_2, \tag{2.9}$$

$$F_1 = \sigma(W_1X + b_1), F_2 = W_2\sigma(W_1X + b_1) + b_2. \tag{2.10}$$

Let $W = [W_1, b_1, W_2, b_2]$ be weights of 2-layer network. Then define the loss function of 2-layer network $\mathcal{R}(W)$ as

$$\mathcal{R}(W) = L([W_1, b_1, W_2, b_2]) = \sum_i \ell(Y_i, F_2([W_1, b_1, W_2, b_2])).$$

Assume σ is smooth, one-to-one and strictly increasing function.

Define $\ell(Y, \cdot)$ be a convex loss function, and $\bar{\ell} = \ell \circ \sigma$ Let $\bar{W}$ be a local minimizer of

$$\mathcal{R}_{linear}(W) = \sum_i \bar{\ell}(\sigma^{-1}(Y_i), W \begin{bmatrix} X_i \\ 1 \end{bmatrix}) = \sum_i \ell(Y_i, \sigma(W \begin{bmatrix} X_i \\ 1 \end{bmatrix})), \tag{2.11}$$

and define $F_{linear}(W)$ and $\bar{Y}$ as

$$F_{linear}(W) = W \begin{bmatrix} X_i \\ 1 \end{bmatrix}, \ \bar{Y} = \sigma(\bar{W} \begin{bmatrix} X_i \\ 1 \end{bmatrix}).$$

First, we show that if the activation function has some linearity, then we can construct a local minimum by borrowing local minima from the linear model.

Proposition 2.14. *Suppose σ is partially linear with $\sigma(x) = cx + d$ on a open interval (α, β). Then $\mathcal{R}(W)$ has a local minimum.*

(Proof) Consider the linear minimization problem

$$\sum_i \ell(Y_i, W \begin{bmatrix} X_i \\ 1 \end{bmatrix}). \tag{2.12}$$

Let $\bar{W}$ be a linear minimizer of the above problem.

Let $\bar{Y} \in \mathbb{R}^{d_Y}$ be the output of linear model and M, m be the maximum and minimum value of $\{\bar{Y}_i\}_{i=1}^N$

$$\bar{Y_i} = \bar{W} \begin{bmatrix} X_i \\ 1 \end{bmatrix}$$

$$M := \max_i \max(\bar{Y_i})$$

$$m := \min_i \min(\bar{Y_i}).$$

Then we can find $f(x) = px + q,\ p \neq 0,\ p, q \in \mathbb{R}$ such that,

$$f(M), f(m) \in (\alpha, \beta).$$

Let define the weight $\hat{W} = [\hat{W}_1, \hat{b}_1, \hat{W}_2, \hat{b}_2]$ of 2-layer network as :

$$\hat{W}_1 = \begin{bmatrix} f(\bar{W}_{[1:d_X]}) \\ \mathbf{0} \end{bmatrix}, \hat{b}_1 = \begin{bmatrix} f([\bar{W}]_{[d_X+1]}) \\ \mathbf{0} \end{bmatrix} \tag{2.13}$$

$$\hat{W}_2 = \begin{bmatrix} (cp)^{-1} I_{d_Y} & 0 \end{bmatrix}, \hat{b}_2 = (-p^{-1}q - (cp)^{-1}d)\mathbf{1}. \tag{2.14}$$

Since $\hat{W}_1 X_i + \hat{b}_1 = f(\bar{Y}_i) \in [\alpha, \beta]$, we have $\sigma(\hat{W}_1 X_i + \hat{b}_1) = cf(\bar{Y}_i) + d$. Then we

claim $\hat{W}$ is the local minimum. To show this, we introduce the small disturbance $\delta_W = (\delta_{w_1}, \delta_{b_1}, \delta_{w_2}, \delta_{b_2})$. Then, since $(\hat{W}_1 + \delta_{W_1})X_i + \hat{b}_1 + \delta_{b_1} \in (\alpha, \beta)$,

$$\sigma((\hat{W}_1 + \delta_{W_1})X_i + \hat{b}_1 + \delta_{b_1}) = \sigma(\begin{bmatrix} p\bar{Y}_i + q\mathbf{1}_{d_Y} \\ \mathbf{0} \end{bmatrix} + \delta_{W_1}X_i + \delta_{b_1}) = c(\begin{bmatrix} p\bar{Y}_i + q\mathbf{1}_{d_Y} \\ \mathbf{0} \end{bmatrix} + \delta_{W_1}X_i + \delta_{b_1})$$

$$F_2 = (\hat{W}_2 + \delta_{W_2})(\begin{bmatrix} cp\bar{Y}_i + cq\mathbf{1}_{d_Y} \\ \mathbf{0} \end{bmatrix} + c\delta_{W_1}X_i + c\delta_{b_1} + d) + \delta_{b_2}$$

$$= \bar{Y}_i + [p^{-1}\delta_{W_1}X_i + p^{-1}\delta_{b_1}]_{[:d_Y]} + \delta_{W_2}([cp\bar{Y}_i + cq\mathbf{1}_{d_Y}]_{[:d_Y]}c\delta_{b_1} + d) + c\delta_{W_2}\delta_{W_1}X_i + \delta_{b_2}$$

$$= \bar{Y}_i + \delta \begin{bmatrix} X_i \\ 1 \end{bmatrix} \tag{2.15}$$

where $\delta = [\delta_1, \delta_2]$ and

$$\delta_1 = p^{-1}\delta_{W_1[:,:d_Y]} + c\delta_{W_2}\delta_{W_1} \tag{2.16}$$

$$\delta_2 = p^{-1}\delta_{b_1[:,:d_Y]} + \delta_{W_2}([cp\bar{Y}_i + cq\mathbf{1}_{d_Y}]_{[:d_Y]}c\delta_{b_1} + d) + \delta_{b_2}. \tag{2.17}$$

Because, $\mathcal{R}(\hat{W}) = \mathcal{R}_{linear}(\bar{W}) < \mathcal{R}_{linear}(\bar{W} + \delta) = \mathcal{R}(\hat{W} + \delta_W)$, we can conclude that $\hat{W}$ is the local minimum. □

Proposition 2.15. *Let denote the Affine transformation of σ as $\sigma^a(x) = c\sigma(x)+d$. Then consider the linear minimization model with σ^a*

$$\mathcal{R}^a_{linear}(W) = \sum_i \bar{\ell}^a((\sigma^a)^{-1}(Y_i), W \begin{bmatrix} X_i \\ 1 \end{bmatrix}) = \sum_i \ell(Y_i, \sigma^a(W \begin{bmatrix} X_i \\ 1 \end{bmatrix})) \tag{2.18}$$

where $\bar{\ell}^a = \ell \circ \sigma^a$. Let $\bar{W}^a$ be the local minimizer of $R^a_{linear}(W)$.

Then define $\hat{W}^a = [\hat{W}_1^a, \hat{b}_1^a, \hat{W}_2^a, \hat{b}_2^a]$ *as*

$$\hat{W}_1^a = \begin{bmatrix} \bar{W}^a_{[1:d_X]} \\ \mathbf{0} \end{bmatrix}, \hat{b}_1^a = \begin{bmatrix} [\bar{W}]^a_{[d_X+1]} \\ \mathbf{0} \end{bmatrix} \tag{2.19}$$

$$\hat{W}_2^a = \begin{bmatrix} cI_{d_Y} & 0 \end{bmatrix}, \hat{b}_2^a = c^{-1}d\mathbf{1}. \tag{2.20}$$

Then, $\mathcal{R}(\hat{W}^a) = \mathcal{R}^a_{linear}(\bar{W}^a)$

(proof) Let $\bar{Y}^a = \sigma^a(\bar{W}^a X)$. $(\sigma^a)^{-1}(x) = \sigma^{-1}(c^{-1}(x-d))$

$$\sigma(\hat{W}_1^a X + \hat{b}_1^a) = c^{-1}\sigma^a(\hat{W}_1^a X + \hat{b}_1^a) - c^{-1}d = c^{-1}\bar{Y}^a - c^{-1}d \tag{2.21}$$

$$\hat{W}_2^a \sigma(\hat{W}_1^a X + b_1^a) + b_2^a = \bar{Y}^a. \tag{2.22}$$

Hence,

$$\mathcal{R}(\hat{W}^a) = \sum_i \ell(Y_i, F_2([\hat{W}_1^a, \hat{b}_1^a, \hat{W}_2^a, \hat{b}_2^a])) = \sum_i \ell(Y_i, \bar{Y}^a) \tag{2.23}$$

$$= \sum_i \ell(Y_i, \sigma^a(\bar{W}^a \begin{bmatrix} X_i \\ 1 \end{bmatrix})) = \mathcal{R}^a_{linear}(\bar{W}^a).\square \tag{2.24}$$

Remark 2.16. *For general smooth activation* $\sigma(x)$*, the borrowing technique does not apply. For linear activation,* $\mathcal{R}^a_{linear}(\bar{W}^a)$ *is constant because the linear model output is invariant to Affine transformation. In other words, the borrowed local minimum* $\hat{W}$ *has constant loss via Affine transformation path.*
However, for general smooth activation, $\mathcal{R}^a_{linear}(\bar{W}^a)$ *is continuously changing, hence the local minimum property of the borrowed weight is broken.*

2.4 Absence of local minimum in the shallow network for small N

Proposition 2.17. *Consider a 2-layer network with $N = 1$, and $d_X = d_1 = d_Y$. Then $\mathcal{R}(W)$ has no bad local minimum.*

(proof) The neural network output is represented as:

$$F_2(W) = w_2\sigma(w_1 x + b_1) + b_2.$$

First, For $x_1 \in \mathbb{R}, y_1 \in \mathbb{R}$, take $\hat{W} = (w_1, b_1, w_2, b_2) = (0, \sigma^{-1}(y_1), 1, 0)$, then since $F_2(\hat{W}) = y_1$, the global minimum value is zero.

$$\mathcal{R}(\hat{W}) = 0.$$

For any $W = (w_1, b_1, w_2, b_2)$, and $\mathcal{R}(\hat{W}) > 0$, introduce the disturbance $\delta_W = (\delta_{w_1}, \delta_{b_1}, \delta_{w_2}, \delta_{b_2}) = (0, 0, 0, \delta sign(y_1 - F_2(W))$, for sufficiently small $\delta \in \mathbb{R}$. Then since

$$\ell(y_1, F_2(W + \delta_W)) < \ell(y_1, F_2(W)),$$

there exists strictly decreasing path to the global minimum $\hat{W}$. Therefore W cannot be a local minimum. □

Proposition 2.18. *Consider a 2-layer network with $N = 2$, and $d_X = d_1 = d_Y$. Suppose the convex loss $\ell(x)$ has strictly increasing derivative i.e., $\ell'(x)$ is injective. Then $\mathcal{R}(W)$ has no bad local minimum if $x_1 \neq x_2$.*

(proof) The neural network output is represented as :

$$F_1(W,x) = w_1 x + b_1,\ F_2(W,x) = w_2\sigma(w_1 x + b_1) + b_2.$$

For the training samples $(x_1, y_1), (x_2, y_2)$,

$$\begin{bmatrix} F_1(W,x_1) & 1 \\ F_1(W,x_2) & 1 \end{bmatrix} \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}.$$

Suppose $x_1 \neq x_2$. If we take (w_1, b_1) such that $F_1(W,x_1) \neq F_1(W,x_2)$, then since the 2×2 matrix is invertible, we can take $W = (w_1, b_1, w_2, b_2)$ such that

$$F_2(W,x_1) = y_1,\ F_2(W,x_2) = y_2.$$

Therefore the global minimum value is zero. Suppose $W = (w_1, b_1, w_2, b_2)$ is a bad local minimum with $\mathcal{R}(W) > 0$. For sufficiently small disturbance $\delta_W = (\delta_{w_1}, \delta_{b_1}, \delta_{w_2}, \delta_{b_2})$, we have $\mathcal{R}(W) < \mathcal{R}(W + \delta_W)$. Suppose $F_1(W,x_1) < y_1$, and $F_1(W,x_2) < y_2$, then for $\delta_W = (0,0,0,\delta)$ with $\delta > 0$, we have

$$\ell(y_i, w_2\sigma(w_1 x_i + b_1) + b_2 + \delta) < \ell(y_i, w_2\sigma(w_1 x_i + b_1) + b_2),\ i = 1,2.$$

Therefore we get $\mathcal{R}(W + \delta_W) < \mathcal{R}(W)$, which is a contradiction. This method similarly applies for $F_1(W,x_1) > y_1,\ F_1(W,x_2) > y_2$. Therefore we assume

$$F_1(W,x_1) \leq y_1,\ F_1(W,x_2) \geq y_2.$$

Without loss of generality, assume $F_1(W,x_1) \neq y_1$.

Then for the same disturbance $\delta_W = (0, 0, 0, \delta)$, we have

$$\ell(y_1, w_2\sigma(w_1x_1 + b_1) + b_2 + \delta) < \ell(y_1, w_2\sigma(w_1x_1 + b_1) + b_2)$$

$$\ell(y_2, w_2\sigma(w_2x_1 + b_1) + b_2 + \delta) > \ell(y_2, w_2\sigma(w_1x_2 + b_1) + b_2).$$

First, assume $F_1(W, x_1) < y_1,\ F_1(W, x_2) = y_2$, then

$$|\frac{\partial\ell(y_1, w_2\sigma(w_1x_1 + b_1) + b_2 + \delta)}{\partial\delta}|_{\delta=0}| > |\frac{\partial\ell(y_2, w_2\sigma(w_1x_2 + b_1) + b_2 + \delta)}{\partial\delta}|_{\delta=0}|$$

because $\ell'(x) > \ell(0)$ for $x \neq 0$. Since $\ell(y_1, w_2\sigma(w_1x_1 + b_1) + b_2 + \delta)$ term decreases faster, we have

$$\mathcal{R}(W + \delta_W) < \mathcal{R}(W).$$

So W is not a local minimum.

Next, assume $F_1(W, x_1) < y_1,\ F_1(W, x_2) > y_2$. In this case,

If

$$|\frac{\partial\ell(y_1, w_2\sigma(w_1x_1 + b_1) + b_2 + \delta)}{\partial\delta}|_{\delta=0}| > |\frac{\partial\ell(y_2, w_2\sigma(w_1x_2 + b_1) + b_2 + \delta)}{\partial\delta}|_{\delta=0}|,$$

then because $\ell(y_1, w_2\sigma(w_1x_1 + b_1) + b_2 + \delta)$ term decreases faster,

$$\mathcal{R}(W + \delta_W) < \mathcal{R}(W).$$

Otherwise if

$$|\frac{\partial\ell(y_1, w_2\sigma(w_1x_1 + b_1) + b_2 + \delta)}{\partial\delta}|_{\delta=0}| < |\frac{\partial\ell(y_2, w_2\sigma(w_1x_2 + b_1) + b_2 + \delta)}{\partial\delta}|_{\delta=0}|,$$

similarly,

$$\mathcal{R}(W - \delta_W) < \mathcal{R}(W).$$

Therefore we have to have

$$|\frac{\partial \ell(y_1, w_2\sigma(w_1x_1 + b_1) + b_2 + \delta)}{\partial \delta}|_{\delta=0}| = |\frac{\partial \ell(y_2, w_2\sigma(w_1x_2 + b_1) + b_2 + \delta)}{\partial \delta}|_{\delta=0}|,$$

and because $\ell'(x)$ is injective, we have

$$y_1 - (w_2\sigma(w_1x_1 + b_1) + b_2) = (w_2\sigma(w_2x_1 + b_1) + b_2) - y_2.$$

Similarly, by introducting the disturbance
$\delta_W = (\delta_{w_1}, \delta_{b_1}, \delta_{w_2}, \delta_{b_2}) = (\delta, 0, 0, 0), (0, \delta, 0, 0)$, we have

$$x_1\sigma'(w_1x_1 + b_1) = x_2\sigma'(w_1x_2 + b_1) \tag{2.25}$$

$$\sigma'(w_1x_1 + b_1) = \sigma'(w_1x_2 + b_1). \tag{2.26}$$

Because $\sigma'(x) > 0$, it would be a contradiction since $x_1 \neq x_2$. □

2.5 Existence of local minimum in the shallow network

Assumption 2.19. *Assume $\sigma(x)$ is analytic and*

$$\{1, \sigma(x), \sigma'(x), x\sigma(x), \sigma''(x), x\sigma''(x), x^2\sigma''(x)\}$$

is linearly independent. Actually $\sigma(x)$ is not a solution to any second order linear ODE with polynomial coefficient of the following form:

$$(Ax^2 + Bx + C)y'' + (Dx + E)y' + Fy + G = 0 \tag{2.27}$$

if A, B, C, D, E, F, and G are not zero at the same time.

We discover that widely used activation functions actually satisfy the assumption.

Lemma 2.20. *Tanh, Sigmoid, SiLU, SoftPlus, and GELU activation functions satisfy Assumption 2.19.*

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.28}$$

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.29}$$

$$SiLU(x) = \frac{x}{1 + e^{-x}} \tag{2.30}$$

$$SoftPlus(x) = \log(1 + e^x) \tag{2.31}$$

$$GELU(x) = \frac{x}{2}(1 + erf(\frac{x}{\sqrt{2}})). \tag{2.32}$$

(proof) Suppose $\sigma(x)$ is sigmoid. Suppose $\sigma(x)$ is a solution of some second order linear ODE of form (2.27). Then since

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)),\ \sigma''(x) = \sigma(x)(1 - \sigma(x))(1 - 2\sigma(x)),$$

we have

$$
\begin{aligned}
&(Ax^2 + Bx + C)(\sigma(x)(1 - \sigma(x))(1 - 2\sigma(x))) + (Dx + E)\sigma(x)(1 - \sigma(x)) + F\sigma(x) + G \\
=&(2Ax^2 + 2Bx + 2C)(\sigma(x))^3 + (-3Ax^2 + (-3B - D)x + (-3C - E))(\sigma(x))^2 + \\
&(Ax^2 + (B + D)x + (C + E + F))(\sigma(x)) + G = 0.
\end{aligned}
$$

Since $\sigma(x)$ can be viewed as the root of a cubic equation with polynomial coefficients, we can consider the field extension of the quotient field of polynomial ring. Let Q denote the quotient field of polynomial ring and $Q(\sigma(x))$ be a extension field of Q with $\sigma(x)$. Let K be a Galois extension of Q including element $\sigma(x)$. Since $\sigma(x)$ is the root of cubic equation, the degree of field extension is finite.

$$[Q(\sigma(x)) : Q] \leq [K : Q] \leq |S_3| = 6.$$

However, since $\sigma(x) = \frac{1}{1+e^{-x}}$ is transcendental function, it cannot be expressed in terms of a finite sequence of the algebraic operations, hence

$$[Q(\sigma(x)) : Q] = \infty.$$

This is a contradiction, therefore the sigmoid function satisfies Assumption 2.19. Similarly, because $tanh(x) = 2sigmoid(2x) - 1$, the tanh function satisfies Assumption 2.19.

For $\sigma(x) = SiLU(x)$, let $\sigma(x) = xs(x)$, where $s(x)$ denotes the sigmoid function.

Then we have

$$\sigma'(x) = s(x)(1 + x(1 - s(x)))$$
$$\sigma''(x) = s(x)(1 - s(x))(2 + x - (2x + 1)s(x) + x(s(x))^2).$$

By substituting into (2.27), we get the quartic equation with polynomial coefficient, of which $s(x)$ is a solution. Similarly, let K be a Galois extension of Q including element $s(x)$, the degree of field extension is

$$[Q(s(x)) : Q] \leq [K : Q] \leq |S_4| = 24.$$

Since $s(x)$ is transcendental, this is a contraction.

For $\sigma(x) = SoftPlus(x)$, we have $\sigma'(x) = s(x)$ where $s(x)$ denotes the sigmoid function. Then we have

$$\sigma'(x) = s(x)$$
$$\sigma''(x) = s'(x) = s(x)(1 - s(x)).$$

By substituting into (2.27), we have

$$\sigma(x) = P_2(s(x))$$

for some quadratic polynomial $P_2(X)$. By substituting into (2.27) again, we have

$$\sigma(x) = P_4(s(x))$$

for some quartic polynomial $P_4(x)$. Similar to $SiLU$ case, we have a contradiction. Therefore we conclude that for sigmoid activation function $s(x)$, if $\sigma(x)$ is a form of polynomial of $s(x)$, k-th derivative, k-th indefinite integral or their linear combination, *i.e*,

$$\sigma(x) \in span\{P_i(s(x)), s^{(k)}(x), \int^{(k)} s(s)dx^{(k)}\}$$

then $\sigma(x)$ satisfies Assumption 2.19.

For $\sigma(x) = GELU(x)$, direct substitution into (2.27) induces

$$(Ax^2+Bx+C)(\frac{\sqrt{2}}{\sqrt{\pi}}e^{-\frac{x^2}{2}} - x^2\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}})$$
$$+ (Dx+E)(\frac{1}{2}\text{erf}(\frac{x}{\sqrt{2}}) + x\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} + \frac{1}{2}) + \frac{F}{2}(x\text{erf}(\frac{x}{\sqrt{2}}) + x) + G = 0 \qquad (2.33)$$
$$\frac{1}{2}\text{erf}(\frac{x}{\sqrt{2}})(Dx+E+Fx) + \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}(-Ax^4 - Bx^3 + (2A-C+D)x^2 + (2B+E)x + 2C)$$
$$+ (\frac{D+F}{2})x + \frac{E}{2} + G = 0. \qquad (2.34)$$

Since $\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$ is not elementary function by Liouville's theorem, we have

$$(Dx+E+Fx) = 0.$$

Otherwise, $\text{erf}(x)$ would be elementary function. Therefore we have

$$D = -F,\ E = 0.$$

Similarly, since $e^{-\frac{x^2}{2}}$ is transcendental over Q, we have

$$A = 0,\ B = 0,\ (2A-C+D) = 0,\ (2B+E) = 0,\ C = 0, D+F = 0,\ E = 0,\ G = 0.$$

Therefore every coefficient is zero. Hence we have a contradiction. □

Proposition 2.21. *Consider a 2-layer network with $N \geq 7$, and $d_X = d_1 = d_Y$. Suppose ℓ is L^2 loss function, and $\sigma(x)$ satisfies Assumption 2.19. Then $\mathcal{R}(W)$ has a strict local minimum $\hat{W}$ for a generic dataset $X \in \mathbb{R}^{1\times N}$ with $\mathcal{R}(\hat{W}) > 0$.*

(proof) By assumption, $\{1, \sigma(x), \sigma'(x), x\sigma(x), \sigma''(x), x\sigma''(x), x^2\sigma''(x)\}$ is linearly independent. Let $A(X) = \{\mathbf{1}_N, [\sigma(x_i)], [\sigma'(x_i)], [x_i\sigma(x_i)], [\sigma''(x_i)], [x_i\sigma''(x_i)], [x_i^2\sigma''(x_i)]\}$. Consider the mapping

$$(x_1, x_2, ..., x_7) \mapsto det(A([x_i]_{i=1}^7)).$$

Because of linear independence, this map is not zero map. Since $\sigma(x)$ is analytic, this map is also analytic. Therefore, zero set of the map has measure zero.
For generic $X = (x_1, x_2, ...x_N)$, we have seven linearly independent N-dimensional vectors

$$\{\mathbf{1}_N, [\sigma(x_i)], [\sigma'(x_i)], [x_i\sigma(x_i)], [\sigma''(x_i)], [x_i\sigma''(x_i)], [x_i^2\sigma''(x_i)]\}_{i=1}^N.$$

Then we can find $N-5$ linearly independent N-dimensional vectors $\{u_k\}$ such that

$$\langle u_k, \mathbf{1}_N \rangle = 0 \tag{2.35}$$

$$\langle u_k, [\sigma(x_i)] \rangle = 0 \tag{2.36}$$

$$\langle u_k, [\sigma'(x_i)] \rangle = 0 \tag{2.37}$$

$$\langle u_k, [x_i\sigma(x_i)] \rangle = 0 \tag{2.38}$$

$$\langle u_k, [x_i\sigma''(x_i)] \rangle = 0 \tag{2.39}$$

$$\langle u_k, [\sigma''(x_i)] \rangle > 0 \tag{2.40}$$

$$\langle u_k, [x_i^2\sigma''(x_i)] \rangle > 0. \tag{2.41}$$

Let $\mathbf{n}(x_i) = w_1 x + b_1$ be a pre-activation output at the first layer.

Select data points $Y = [y_i]_{i=1}^N$ as

$$y_i = F_2(x_i) - w_2 \sum_{k=1}^{N-5} c_k [u_k]_i \tag{2.42}$$

for some positive $c_k \in \mathbb{R}$. Note that the degree of freedom of selecting Y is $N-5$. Then pick $\hat{W} = (w_1, b_1, w_2, b_2) = (1, 0, w_2, 0)$, where w_2 is fixed later. Define $\Delta y \in \mathbb{R}$ such that

$$[\Delta y]_i = F_2(x_i) - y_i = w_2 \sum_{k=1}^{N-5} c_k [u_k]_i.$$

Then we get

$$\langle \Delta y, \mathbf{1}_N \rangle = \langle \Delta y, [\sigma(x_i)] \rangle = \langle \Delta y, [\sigma'(x_i)] \rangle = \langle \Delta y, [x_i \sigma(x_i)] \rangle = \langle \Delta y, [\sigma''(x_i)] \rangle = 0 \tag{2.43}$$

$$\langle \Delta y, [x_i \sigma''(x_i)] \rangle > 0 \tag{2.44}$$

$$\langle \Delta y, [x_i^2 \sigma''(x_i)] \rangle > 0. \tag{2.45}$$

Then for the loss $\mathcal{R} = \ell(y_i, F_2(x_i)) = \|F_2(x_i) - y_i\|^2 = \langle F_2(X) - Y, F_2(X) - Y \rangle$, derivatives are

$$\frac{\partial \mathcal{R}}{\partial w_1} = 2\langle \Delta Y, X\sigma'(X) \rangle = 0 \tag{2.46}$$

$$\frac{\partial \mathcal{R}}{\partial b_1} = 2\langle \Delta Y, \sigma'(X) \rangle = 0 \tag{2.47}$$

$$\frac{\partial \mathcal{R}}{\partial w_2} = 2\langle \Delta Y, \sigma(X) \rangle = 0 \tag{2.48}$$

$$\frac{\partial \mathcal{R}}{\partial b_2} = 2\langle \Delta Y, \mathbf{1} \rangle = 0. \tag{2.49}$$

therefore $\hat{W}$ is a stationary point.

To show $\hat{W}$ is a local minimum, we introduce a small disturbance $\delta_W = (\delta_{w_1}, \delta_{b_1}, \delta_{w_2}, \delta_{b_2})$. The difference between losses is:

$$\mathcal{R}(\hat{W} + \delta_W) - \mathcal{R}(\hat{W}) = \langle F_2(\hat{W} + \delta_W) - Y, F_2(\hat{W} + \delta_W) - Y \rangle - \langle F_2(\hat{W}) - Y, F_2(\hat{W}) - Y \rangle$$

$$= \langle F_2(\hat{W} + \delta_W) - F_2(\hat{W}), F_2(\hat{W} + \delta_W) - F_2(\hat{W}) \rangle + 2\langle \Delta Y, F_2(\hat{W} + \delta_W) - F_2(\hat{W}) \rangle \tag{2.50}$$

$$= \|F_2(\hat{W} + \delta_W) - F_2(\hat{W})\|_2^2 + 2\langle \Delta Y, F_2(\hat{W} + \delta_W) - F_2(\hat{W}) \rangle. \tag{2.51}$$

Note that

$$F_2(\hat{W}+\delta_W)(x_i) - F_2(\hat{W})(x_i) \tag{2.52}$$

$$= ((w_2+\delta_{w_2})\sigma((w_1+\delta_{w_1})x + b_1 + \delta_{b_1}) + b_2 + \delta_{b_2}) - (w_2\sigma(w_1x+b_1)+b_2) \tag{2.53}$$

$$= (w_2+\delta_{w_2})(\sigma((1+\delta_{w_1})x_i + \delta_{b_1}) - \sigma(x_i)) + \delta_{w_2}\sigma(x_i) + \delta_{b_2}. \tag{2.54}$$

We consider the following two cases.

Case 1: $(\delta_{w_1}, \delta_{b_1}) \neq (0,0)$.

In this case, Therefore, we need to show $\langle \Delta Y, F_2(\hat{W}+\delta_W) - F_2(\hat{W})\rangle > 0$. Let $\delta_{1,i} = \delta_{w_1}x_i + \delta_{b_1}$. By Taylor theorem,

$$\sigma(x_i+\delta_{1,i}) - \sigma(x_i) = \sigma'(x_i)\delta_{1,i} + \frac{1}{2}\sigma''(x_i)\delta_{1,i}^2 + o(|\delta_{1,i}|^2) \tag{2.55}$$

$$= \sigma'(x_i)\delta_{w_1}x_i + \sigma'(x_i)\delta_{b_1} + \frac{1}{2}\sigma''(x_i)(\delta_{w_1}^2x_i^2 + 2\delta_{w_1}x_i\delta_{b_1} + \delta_{b_1}^2) + o(|\delta_{1,i}|^2). \tag{2.56}$$

By using Equation (2.43), we have

$$\langle \Delta Y, F_2(\hat{W}+\delta_W) - F_2(\hat{W})\rangle =$$
$$\frac{1}{2}(w_2+\delta_{w_2})\delta_{w_1}^2\langle \Delta Y, \sigma''(X)X^2\rangle + \frac{1}{2}(w_2+\delta_{w_2})\delta_{b_1}^2\langle \Delta Y, \sigma''(X)\rangle + (w_2+\delta_{w_2})\langle \Delta Y, o(\|\delta_1\|^2)\rangle.$$

Pick δ_{w_2}, such that $|\delta_{w_2}| < \frac{1}{2}|w_2|$, then

$$(w_2 + \delta_{w_2})\delta_{w_1}^2 \langle \Delta Y, \sigma''(X)X^2 \rangle > \frac{1}{2} w_2 \delta_{w_1}^2 \langle \Delta Y, \sigma''(X)X^2 \rangle := M_1 > 0 \tag{2.57}$$

$$(w_2 + \delta_{w_2})\delta_{b_1}^2 \langle \Delta Y, \sigma''(X) \rangle > \frac{1}{2} w_2 \delta_{b_1}^2 \langle \Delta Y, \sigma''(X) \rangle := M_2 > 0 \tag{2.58}$$

$$(w_2 + \delta_{w_2}) \langle \Delta Y, o(\|\delta_1\|^2) \rangle \leq \frac{3}{2}|w_2| \langle \Delta Y, o(\|\delta_1\|^2) \rangle \leq \frac{3}{2}|w_2| \|\Delta Y\|_2 \|o(\|\delta_1\|^2)\|_2. \tag{2.59}$$

For sufficiently small δ_1,

$$\|o(\|\delta_1\|^2)\|_2 < \frac{1}{12|w_2|\|\Delta Y\|_2} \min\left(\frac{M_1}{\|X\|_2^2}, \frac{M_2}{N}\right).$$

Then we have

$$\langle \Delta Y, F_2(\hat{W} + \delta_W) - F_2(\hat{W}) \rangle > \frac{1}{2}(M_1 \delta_{w_1}^2 + M_2 \delta_{b_1}^2 - 3|w_2| \|\Delta Y\|_2 |o(\|\delta_1\|^2) \tag{2.60}$$

$$> \frac{1}{4}(M_1 \delta_{w_1}^2 + M_2 \delta_{b_1}^2) > 0. \tag{2.61}$$

Case 2: $(\delta_{w_1}, \delta_{b_1}) \neq (0, 0)$

In this case, we have

$$F_2(\hat{W} + \delta_W) - F_2(\hat{W}) = \delta_{w_2} \sigma(X) + \delta_{b_2} \mathbf{1}_N. \tag{2.62}$$

Since $\sigma(X)$ and $\mathbf{1}_N$ are linearly independent, ${}_2(\hat{W} + \delta_W) - F_2(\hat{W}) \neq \mathbf{0}_N$. Therefore,

$$\langle \Delta Y, F_2(\hat{W} + \delta_W) - F_2(\hat{W}) \rangle = 0 \tag{2.63}$$

$$\|F_2(\hat{W} + \delta_W) - F_2(\hat{W})\|_2 > 0. \tag{2.64}$$

In both cases, we conclude that

$$\mathcal{R}(\hat{W} + \delta_W) - \mathcal{R}(\hat{W}) > 0.$$

Hence $\mathcal{R}(W)$ has the strict local minimum at $W = \hat{W}$. $\square$

Lemma 2.22. *If $\sigma(x)$ satisfies Assumption 2.19, then*

$$\{1, \sigma(s), \sigma'(s), s\sigma(s), \sigma''(s), s\sigma''(s), s^2\sigma''(s)\} \tag{2.65}$$

is also linearly independent where $s = \sigma(x)$.

Next, we find the local minimum for shallow 3-layer network under the following assumption on $\sigma(x)$.

Assumption 2.23. *Assume $\sigma(x)$ is analytic and let*

$$\begin{aligned} B = \{&1, \sigma(\sigma(x)), \sigma'(\sigma(x))\{1, \sigma(x), \sigma'(x)\{1, x\}, \sigma''(x)\{1, x, x^2\}\}, \\ &\sigma''(\sigma(x))\{1, \sigma(x), \sigma(x)^2, \sigma'(x)\{1, x\}, \sigma''(x)\{1, x, x^2\}, \sigma'(x)^2\{1, x, x^2\}, \\ &\sigma(x)\sigma'(x)\{1, x\}, \sigma(x)\sigma''(x)\{1, x, x^2\}, \sigma'(x)'\sigma''(x)\{1, x, x^2, x^3\}\}\}. \end{aligned}$$

Let $B_1 = \{\sigma''(\sigma(x))\sigma'(x)^2\}$, $\bar{B}_1 = \{\sigma''(\sigma(x))\sigma(x)\sigma''(x)\}$, $B_2\} = \{\sigma''(\sigma(x))\sigma'(x)^2x^2\}$, $\bar{B}_2 = \{\sigma''(\sigma(x))\sigma(x)\sigma''(x)x^2\}$, and $\tilde{B} = B - B_1 - B_2 - \bar{B}_1 - \bar{B}_2$.

Then assume

$$span\{B_1\} \cap span\{\tilde{B}\} = \{0\} \tag{2.66}$$

$$span\{B_2\} \cap span\{\tilde{B}\} = \{0\} \tag{2.67}$$

$$span\{B_1\} \cap span\{B_2\} = \{0\}. \tag{2.68}$$

Remark 2.24. *If B is linearly independent, then $\sigma(x)$ satisfies Assumption 2.23.*

Lemma 2.25. *Tanh, Sigmoid activation functions satisfy Assumption 2.23.*

(proof) To prove the lemma, we need the following lemma.

Lemma 2.26. *Let Q be the quotient field of polynomial ring. Assume $\sigma(x)$ is transcendental function. Then $\sigma \circ \sigma(x)$ is transcendental over $Q(\sigma(x))$.*

(proof) Since $\sigma(x)$ is transcendental, we the following have an (field) isomorphism

$$Q(\sigma(x)) \cong \mathbb{R}(x, y), \tag{2.69}$$

for indeterminate x and y. Therefore, we have

$$Q(\sigma \circ \sigma(x)) \cong \mathbb{R}(x, \sigma(y)). \tag{2.70}$$

Since $\sigma(y)$ is transcendental, we conclude that $Q(\sigma \circ \sigma(x))$ is transcendental extension of $Q(\sigma(x))$. □

First, let $\sigma(x)$ be Sigmoid. Since $\sigma(x)$ contains an exponential part, by Lemma 2.26, we can say that $\sigma(\sigma(x))$ is transcendental over $Q(\sigma(x))$. Since $\sigma'(x) = \sigma(x)(1-\sigma(x))$, $\sigma'(x)$ and $\sigma''(x)$ are the second and third order polynomials in terms with

$\sigma(x)$. Now we claim that

$$span\{B\} = span\{\sigma''(\sigma(x))\{...\}\} \oplus span\{\text{Rest of } B\}\}.$$

Suppose not, then there exists a intersection between two spaces.
Since $\sigma''(\sigma(x)) = P_3(\sigma(x)), \sigma''(\sigma(x)) = P_2(\sigma(x))$, for some third and second order polynomials P_3, P_2, we can say $(\sigma(x))$ is a solution of a cubic polynomial in $Q(\sigma(x))$. This is a contradiction since $(\sigma(x))$ is transcendental over $Q(\sigma(x))$.
Therefore, we only need to show

$$span\{B_1\} \nsubseteq span\{\tilde{B}'\}$$

where $\tilde{B}' = \sigma''(\sigma(x))\{...\} - B_1 - B_2 - \bar{B}_1 - \bar{B}_2$.

Suppose not. Since $B_1 = \{\sigma''(\sigma(x))\sigma'(x)^2\}$ is represented as constant equation in terms of x, and quartic polynomial in terms of $\sigma(x)$, we conclude that

$$p(\sigma(x)) \in span\{B_1\}$$

where $p(x)$ is a cubic polynomial with constant coefficient. Since there is no such term in $span\{\tilde{B}'\}$, we conclude that there exists a quartic polynomial in terms of $\sigma(x)$ with polynomial degree $q(\sigma(x)) = 0$. However, since $\sigma(x)$ is transcendental over Q, it is a contradiction.
For case of $\sigma(x) = Tanh$, since $\sigma'(x) = -\sigma(x)^2$, we can apply the similar argument with *Sigmoid*. □

Proposition 2.27. *Consider a 3-layer network with $N \geq 29$, and $d_X = d_1 = d_2 =$*

d_Y. Suppose ℓ is L^2 loss function, and $\sigma(x)$ satisfies Assumption 2.23. Then $\mathcal{R}(W)$ has a strict local minimum $\hat{W}$ for a generic dataset $X \in \mathbb{R}^{1\times N}$ with $\mathcal{R}(\hat{W}) > 0$.

(Proof) Consider 3-layer network.

$$F_3(W) = w_3\sigma(w_2\sigma(w_1x + b_1) + b_2) + b_3. \tag{2.71}$$

First, we decompose $\mathcal{B}$ as

$$\mathcal{B} = \mathcal{B}_1 \oplus \mathcal{B}_2 \oplus \mathcal{B}^{\perp}. \tag{2.72}$$

where $\mathcal{B}_1 = span\{B_1\}, \mathcal{B}_2 = span\{B_2\}$ and $\mathcal{B}^{\perp}$ is the space in $\mathcal{B}$ which is orthogonal to $\mathcal{B}_1$ and $\mathcal{B}_2$. By the assumption, we can say

$$\tilde{\mathcal{B}} \subseteq \mathcal{B}^{\perp} \tag{2.73}$$

where $\tilde{\mathcal{B}} = span\{\tilde{B}\}$.
Similar to Proposition 2.21, Equations (2.72)-(2.73) hold for generic $X = (x_1, x_2, ..., x_N)$. Then we can find $(N - 29)$ independent N-dimensional vectors $\{u_k\}$ such that

$$\langle u_k, [b_{j,\mathcal{B}^{\perp}}(x_i)]\rangle = 0 \tag{2.74}$$

$$\langle u_k, [b_{j,\mathcal{B}_1}(x_i)]\rangle > 0 \tag{2.75}$$

$$\langle u_k, [b_{j,\mathcal{B}_2}(x_i)]\rangle > 0 \tag{2.76}$$

where $\{b_{j,\mathcal{B}^{\perp}(x)}\}_j$, $\{b_{j,\mathcal{B}_1}(x)\}$, and $\{b_{j,\mathcal{B}_2}(x)\}$ are basis of $\mathcal{B}^{\perp}$, $\mathcal{B}_1$, and $\mathcal{B}_2$, respectively. Therefore we conclude

$$\langle u_k, [b_{j,\tilde{\mathcal{B}}}(x_i)]\rangle = 0 \tag{2.77}$$

where $b_{j,\tilde{\mathcal{B}}}$ are basis of $\tilde{\mathcal{B}} = span(\tilde{B})$.

Then select data point $Y = [y_i]_{i=1}^N$ as

$$y_i = F_3(x_i) - w_3 \sum_{k=1}^{N-29} c_k [u_k]_i. \tag{2.78}$$

for some $c_k \in \mathbb{R}$. Now pick $\hat{W} = (w_1, b_1, w_2, b_2, w_3, b_3) = (1, 0, 1, 0, w_3, 0)$. To show $\hat{W}$ is a local minimum, we introduce a small disturbance $\delta_W = (\delta_{w_1}, \delta_{b_1}, \delta_{w_2}, \delta_{b_2}, \delta_{w_3}, \delta_{b_3})$. Then we have

$$\begin{aligned} &F_3(\hat{W} + \delta_W)(x) - F_3(\hat{W})(x) \\ &= ((w_3 + \delta_{w_3})\sigma((1 + \delta_{w_2})\sigma((1 + \delta_{w_1})x + \delta_{b_1}) + \delta_{b_2}) + \delta_{b_3}) - (w_3\sigma(\sigma(x))). \end{aligned} \tag{2.79}$$

We consider the following two cases.

Case 1: $(\delta_{w_1}, \delta_{b_1}) = (0, 0)$

In this case, this is very similar case with $L = 2$ network.

$$\begin{aligned} &F_3(\hat{W} + \delta_W)(x) - F_3(\hat{W})(x) \\ &= ((w_3 + \delta_{w_3})\sigma((1 + \delta_{w_2})\sigma(x) + \delta_{b_2}) + \delta_{b_3}) - (w_3\sigma(\sigma(x))). \end{aligned} \tag{2.80}$$

Only difference is x is replaced with $\sigma(x)$. By Lemma 2.22, the assumption of Proposition 2.21 is satisfied. Therefore, by using Proposition 2.21, we can conclude

$$\mathcal{R}(\hat{W} + \delta_W) - \mathcal{R}(\hat{W}) > 0.$$

Case 2: $(\delta_{w_1}, \delta_{b_1}) \neq (0, 0)$

In this case, we calculate $F_3(\hat{W}+\delta_W)(x)$. First, by Taylor theorem, we have

$$\sigma((1+\delta_{w_1})x+\delta_{b_1}) = \sigma(x)+\sigma'(x)(\delta_{w_1}x+\delta_{b_1})+\frac{1}{2}\sigma''(x)(\delta_{w_1}x+\delta_{b_1})^2+o(|\delta_1^2|) \tag{2.81}$$

where $\delta_1 = \delta_{w_1}x+\delta_{b_1}$ Then, by Taylor theorem again, we have

$$\sigma((1+\delta_{w_2})\sigma((1+\delta_{w_1})x+\delta_{b_1})+\delta_{b_2}) \tag{2.82}$$

$$=\sigma(\sigma(x)+\delta_{w_2}\sigma(x)+(1+\delta_{w_2})(\sigma'(x)(\delta_{w_1}x+\delta_{b_1})+\frac{1}{2}\sigma''(x)(\delta_{w_1}x+\delta_{b_1})^2+o(|\delta_1^2|))+\delta_{b_2})$$

$$=\sigma(\sigma(x))+\sigma'(\sigma(x))\delta_2+\frac{1}{2}\sigma''(\sigma(x))\delta_2^2+o(|\delta_2|^2) \tag{2.83}$$

where $\delta_2 = \delta_{w_2}\sigma(x)+(1+\delta_{w_2})(\sigma'(x)(\delta_{w_1}x+\delta_{b_1})+\frac{1}{2}\sigma''(x)(\delta_{w_1}x+\delta_{b_1})^2+o(|\delta_1^2|))+\delta_{b_2}$.
Therefore $F_3(\hat{W}+\delta_W)(x)$ is

$$F_3(\hat{W}+\delta_W)(x) = (w_3+\delta_{w_3})\sigma((1+\delta_{w_2})\sigma((1+\delta_{w_1})x+\delta_{b_1})+\delta_{b_2})+\delta_{b_3} \tag{2.84}$$

$$=(w_3+\delta_{w_3})(\sigma(\sigma(x))+\sigma'(\sigma(x))\delta_2+\frac{1}{2}\sigma''(\sigma(x))\delta_2^2+o(|\delta_2|^2))+\delta_{b_3}. \tag{2.85}$$

Therefore we have

$$F_3(\hat{W}+\delta_W)(x)-F_3(\hat{W})(x) \tag{2.86}$$

$$\delta_{w_3}\sigma(\sigma(x))+(w_3+\delta_{w_3})(\sigma'(\sigma(x))\delta_2+\frac{1}{2}\sigma''(\sigma(x))\delta_2^2+o(|\delta_2|^2)+\delta_{b_3} \tag{2.87}$$

By Equations (2.74)-(2.76),

$$
\begin{aligned}
&\langle \Delta Y, F_3(\hat{W}+\delta_W) - F_3(\hat{W})\rangle = \\
&\frac{1}{2}(w_3+\delta_{w_3})(1+\delta_{w_2})^2\delta_{w_1}^2\langle \Delta Y, \sigma''(\sigma(X))\sigma'(X)^2X^2\rangle \\
+&\frac{1}{2}(w_3+\delta_{w_3})(1+\delta_{w_2})\delta_{w_2}\delta_{w_1}^2\langle \Delta Y, \sigma''(\sigma(X))\sigma(X)\sigma''(X)X^2\rangle \\
+&\frac{1}{2}(w_3+\delta_{w_3})(1+\delta_{w_2})^2\delta_{b_1}^2\langle \Delta Y, \sigma''(\sigma(X))\sigma'(X)^2\rangle \\
+&\frac{1}{2}(w_3+\delta_{w_3})(1+\delta_{w_2})\delta_{w_2}\delta_{b_1}^2\langle \Delta Y, \sigma''(\sigma(X))\sigma(X)\sigma''(X)\rangle + o(|\delta|^2) > 0
\end{aligned} \tag{2.88}
$$

for sufficiently small δ.

So far, we construct a local minimum point for $1-1-1$ and $1-1-1-1$ neural network. In the next section, we stretch the existence of local minima to the larger network using the method called the local minimum embedding.

2.6 Local Minimum Embedding

Consider a neural network and a neuron $\mathbf{n}(l,r)$ with index r in layer l. Let $[u_{r,i}]_i$ be incoming weights into $\mathbf{n}(l,r)$ and $[v_{s,r}]_i$ be outgoing weights of $\mathbf{n}(l,r)$. Consider the larger network by adding a new neuron $\mathbf{n}(l,-1)$ with new weights $[u_{-1,i}]_i$ and $[v_{s,-1}]_i$. Then define the local minimum embedding function γ_λ^r mapping the parameters $([u_{r,i}]_i, [v_{s,r}]_i, \bar{w})$ of the smaller network to the parameters $([u_{-1,i}]_i, [v_{s,-1}]_i, [u_{r,i}]_i, [v_{s,r}]_i, \bar{w})$ of the larger network, where $\bar{w}$ denotes the collection of all remaining parameters

$$
\gamma_\lambda^r([u_{r,i}]_i, [v_{s,r}]_i, \bar{w}) := ([u_{r,i}]_i, \lambda[v_{s,r}]_i, [u_{r,i}]_i, (1-\lambda)[v_{s,r}]_i, \bar{w}). \tag{2.89}
$$

Lemma 2.28 (Hessian [84])**.** *Let $\mathcal{L}$ denote the loss function of the larger network*

and ℓ be the loss function of smaller network. Let $\lambda = \frac{\beta}{\alpha+\beta}$. Then the Hessian of the loss L with respect to the basis $\mathcal{B} = [u_{-1,r}+u_{r,i}, v_{s,-1}+v_{s,r}, \bar{w}, \alpha u_{-1,i}-\beta u_{r,i}, v_{s,-1}-v_{s,r}]$ is given by:

$$H = \begin{bmatrix} \frac{\partial^2 \ell}{\partial u_{r,i}\partial u_{r,j}} & 2\frac{\partial^2 \ell}{\partial u_{r,i}\partial v_{s,r}} & \frac{\partial^2 \ell}{\partial \bar{w}\partial u_{r,i}} & 0 & 0 \\ 2\frac{\partial^2 \ell}{\partial u_{r,i}\partial v_{s,r}} & 4\frac{\partial^2 \ell}{\partial v_{s,r}\partial v_{t,v}} & 2\frac{\partial^2 \ell}{\partial \bar{w}\partial v_{s,r}} & (\alpha-\beta)[D_i^{r,s}] & 0 \\ \frac{\partial^2 \ell}{\partial \bar{w}\partial u_{r,i}} & 2\frac{\partial^2 \ell}{\partial \bar{w}\partial v_{s,r}} & \frac{\partial^2 \ell}{\partial \bar{w}\partial \bar{w}'} & 0 & 0 \\ 0 & (\alpha-\beta)[D_i^{r,s}] & 0 & \alpha\beta[B_{i,j}^r] & (\alpha+\beta)[D_i^{r,s}] \\ 0 & 0 & 0 & (\alpha+\beta)[D_i^{r,s}] & 0 \end{bmatrix} \tag{2.90}$$

where

$$B_{i,j} = \sum_\alpha \sum_k \frac{\partial \ell_\alpha}{\partial \mathbf{n}(l+1,k;x_\alpha)} \cdot v_{k,r} \cdot \sigma''(\mathbf{n}(l,r;x_\alpha))\mathbf{act}(l-1,i;x_\alpha)\mathbf{act}(l-1,j;x_\alpha) \tag{2.91}$$

and

$$D_i^{r,s} := \sum_\alpha \frac{\partial \ell_\alpha}{\partial \mathbf{n}(l+1,s;x_\alpha)} \sigma'(\mathbf{n}(l,r;x_\alpha))\mathbf{act}(l-1,i;x_\alpha). \tag{2.92}$$

Lemma 2.29 (Conditions on $D_i^{r,s} = 0$, [84])**.** *Suppose for the outgoing weights $v_{r,s}$ of $\mathbf{n}(l,r;x)$, we have $\sum_s v_{s,r} \neq 0$. Then $D_i^{r,s} = 0$ if one of the following holds.*

- *The layer l is the last hidden layer.*

- *For all t, t', α, we have*

$$\frac{\partial \ell_\alpha}{\partial \mathbf{n}(l+1,t;x_\alpha)} = \frac{\partial \ell_\alpha}{\partial \mathbf{n}(l+1,t';x_\alpha)}. \tag{2.93}$$

- *For each α, t,*

$$\frac{\partial \ell_\alpha}{\partial \mathbf{n}(l+1,t;x_\alpha)} = 0. \tag{2.94}$$

Remark 2.30. *Lemma 2.29 holds for $1-1-1$ neural network. In other words, $D_i^{r,s} = 0$ for the case $d_X = d_1 = d_Y = 1$.*

Let $H^{small} = \begin{bmatrix} \frac{\partial^2 \ell}{\partial u_{r,i}\partial u_{r,j}} & 2\frac{\partial^2 \ell}{\partial u_{r,i}\partial v_{s,r}} & \frac{\partial^2 \ell}{\partial \bar{w}\partial u_{r,i}} \\ 2\frac{\partial^2 \ell}{\partial u_{r,i}\partial v_{s,r}} & 4\frac{\partial^2 \ell}{\partial v_{s,r}\partial v_{t,v}} & 2\frac{\partial^2 \ell}{\partial \bar{w}\partial v_{s,r}} \\ \frac{\partial^2 \ell}{\partial \bar{w}\partial u_{r,i}} & 2\frac{\partial^2 \ell}{\partial \bar{w}\partial v_{s,r}} & \frac{\partial^2 \ell}{\partial \bar{w}\partial \bar{w}'} \end{bmatrix}$ be the smaller matrix of Hessian H.

Theorem 2.31. *Suppose $d_X = d_Y = 1, d_1 \geq 2$, $N \geq 7$, ℓ is L^2 function and the activation function $\sigma(x)$ satisfies Assumption 2.19. Then there exists the network $1 - d_1 - 1$, which has a bad local minimum.*

(proof) Consider the small 2-layer network with $d_X = d_1 = d_Y = 1$. Then by Proposition 2.21, there exists a local minimum $\hat{W}$ with $\mathcal{R}(\hat{W}) > 0$. In this situation, we have

$$B_{1,1} = \sum_\alpha \frac{\partial \ell_\alpha(x_\alpha, y_\alpha)}{\partial f(x_\alpha)} \cdot v_{s,r} \cdot \sigma''(\mathbf{n}(1,r;x_\alpha)) x_{\alpha,1}^2 \tag{2.95}$$

$$= \sum_i [\Delta Y]_i w_2 \sigma''(x_i) x_i^2 \tag{2.96}$$

$$= w_2 \sum_i [\Delta Y]_i \sigma''(x_i) x_i^2 \tag{2.97}$$

$$= w_2 \langle \Delta Y, \sigma''(X) X^2 \rangle. \tag{2.98}$$

Now, we consider the local minimum embedding on the hidden layer with λ. We add a new neuron using embedding function. Denote $W^{(1)}$ be parameters of the larger network at the first step. By Lemma 2.28, we have the Hessian matrix H

and $D_i^{r,s} = 0$ by Lemma 2.29. Without loss of generality, suppose $w_2 > 0$. Pick $\lambda \in (0,1)$. Since $\hat{W}$ is the strict local minimum, H^{small} is strictly positive definite, and $\alpha\beta[B_{1,1}]^r$ is positive. On the last axis $[v_{s,-1} - v_{s,r}]$ of $\mathcal{B}$, the loss of the larger network is constant on the last axis by direct calculation. Therefore, we conclude $W^{(1)}$ is a local minimum of the larger network.

Consider the path λ from $(0,1)$ to $(-\infty, 0) \cup (1, \infty)$. Note that the loss is constant along the path. Additionally, $\alpha\beta[B_{1,1}]^r$ becomes positive to negative along the path, hence we conclude that the point become saddle finally. Because there exists a decreasing path from $W^{(1)}$, $W^{(1)}$ is not global minimum, *i.e.* the bad local minimum.

We add a new neuron every step. At step t, we have

$$B_{1,1}^{(t)} = \sum_{\alpha} \frac{\partial \ell_\alpha(x_\alpha, y_\alpha)}{\partial f(x_\alpha)} \cdot (1-\lambda)^{t-1} v_{s,r} \cdot \sigma''(\mathbf{n}(1, r; x_\alpha)) x_{\alpha,1} x_{\alpha,1} \tag{2.99}$$

$$= w_2 (1-\lambda)^{t-1} \langle \Delta Y, \sigma''(X) X^2 \rangle > 0. \tag{2.100}$$

Therefore by similar argument, we conclude that W^t is a local minimum.

Finally, we construct sufficiently wide neural network (1-$(t+1)$-1) which has a bad local minimum for each t. □

Theorem 2.32. *Suppose $d_X = d_Y = 1, d_1 \geq 2, d_2 \geq 2$, $N \geq 29$, ℓ is L^2 function and the activation function $\sigma(x)$ satisfies Assumption 2.23. Then there exists the network $1 - d_1 - d_2 - 1$, which has a bad local minimum. Moreover, constructed bad local minimum is non-attracting.*

Chapter 3

Self-Knowledge Distillation via Dropout

3.1 Introduction

Deep neural networks (DNN) have achieved a state-of-the-art performance in many domains, including image classification, object detection, and segmentation [92, 42, 41]. In designing models that are deeper and more complex for a higher performance, model compression is essential in delivering a deep learning model for practical application. To develope lightweight models, many previous attempts have been made, including an efficient architecture [46, 97], model quantization [31], pruning [38], and knowledge distillation [44].

Although knowledge distillation is a popular DNN compression method, conventional offline knowledge distillation methods have several limitations. To distill the knowledge from a teacher network to a student network, two main steps are required. First, we train a large teacher network, followed by a student network

using distillation. Fully training the teacher model with massive datasets requires considerable effort. Second, it is difficult to search for an appropriate teacher model that correspond to the target student model. In addition, The common belief in traditional knowledge distillation is to expect a larger or more accurate teacher network to be a good teacher. However, a teacher network with a deeper structure and higher accuracy does not guarantee an improved performance of the student network [13, 122].

Self-knowledge distillation is a solution to these limitations. In a self-knowledge distillation, a teacher network becomes a student network itself. Knowledge is efficiently distilled in a single training process in a single model without the guidance of other external models. Several self-distillation methods have been proposed [115, 125, 131, **?**]. However, these methods also have the following drawback: 1) Some methods require subnetworks with additional parameters. 2) Some methods request additional ground-truth label information, which means that the model depends on the class distribution of the training datasets.

Inspired by these observations, we propose a simple self-knowledge distillation using a dropout (SD-Dropout). Our method generates ensemble models with identical architecture, but different weights through a dropout sampling. After all feature extraction layers, we sample the global feature vector to obtain two different features with different perspectives. These two feature vectors are then passed to the last fully connected layer, and two different posterior distributions are generated. We then match these posterior distributions using Kullback-Leibler divergences (KL-divergences). Dark knowledge between these two internal models can improve their performance through knowledge distillation [44]. The proposed method does not require any additional parameters nor does it require additional label infor-

mation. Furthermore, the SD-Dropout method is a model-agnostic and a method-agnostic, meaning it can be easily implemented with various backbone models and other self-distillation methods.

For more effective distillation, we consider the way to use KL-divergence. In most of the other methods, the gradient of the reference distribution in the KL-divergence is not propagated through model parameters. However, we theoretically demonstrate that the gradient of the reference distribution is greater than that of the other distribution, and we empirically verify the effectiveness of using the gradient of both distributions in the KL-divergence.

We conduct extensive experiments to verify the effectiveness and generalization of our method on various image classification tasks (i.e., CIFAR-100 [55], CUB-200-2011 [102], and Stanford Dog [54]). In addition, when acquiring two different sampled feature vectors, only a dropout layer is used after obtaining the feature vectors; thus, the proposed method can be easily applied to any network architecture structure and any knowledge distillation methods. Experimental results demonstrate that our simple and effective regularization method improves the performance of various model architectures (i.e., ResNet [42], and DenseNet [49]) and is in good agreement with other knowledge distillation methods [115, 131, 125, 133]. Furthermore, our experiments show that our method improves the calibration performance and adversarial robustness.

Our contributions are summarized as follows:

- We present a simple self-knowledge distillation methodology using dropout techniques.
- Our self-knowledge distillation method can collaborate easily with other knowl-

edge distillation methods.

- We describe experimental observations regarding the forward and reverse KL-divergence commonly used in knowledge distillation.
- Extensive experiments demonstrate the effectiveness of our methodology.

3.2 Related work

3.2.1 Knowledge Distillation

Knowledge distillation [44] is one of the most popular compression methods for transferring knowledge from a large and complex network (known as a teacher network) into a small and simple network (known as a student network). Most of these methods assume that they have a pre-trained teacher network. Thus, these methods are referred to as offline knowledge distillation.

Offline methods are simple and easy to apply. Offline methods focus on improving student networks by matching the features or distributions. An attempt at achieving knowledge distillation by matching the probability distribution was proposed by [83]. In addition, some researchers used training samples generated by adversarial attack methods close to the decision boundary of the teacher network [43], and [77] proposed a teacher assistant network to bridge the gap between the teacher and student networks.

The offline knowledge distillation methods are simple and effective. However, there are several limitations to offline distillation [77]. These limitations are caused by the gap between small students and large teacher networks. Thus, an online knowledge distillation method, which simultaneously trains the teacher network

and student network, is proposed. In addition, [133] introduced a method in which a teacher network and a student network distill the knowledge from each other using KL-divergence. Moreover, [6] proposed the use of multiple auxiliary peers and a group leader with an attention-based mechanism. An adversarial mechanism was used to discriminate the feature map distributions from each network [14].

3.2.2 Self-Knowledge Distillation

Self-knowledge distillation is a method in which a teacher and student network are by themselves the same. Self-knowledge distillation methods can be considered a special case of online knowledge distillation methods. Several knowledge distillation methods differ in their methodologies for generating KL-divergences [115, 125, 131].

Specifically, [115] proposed a method for matching predictions from different distorted data of the same training data. In addition, [131] distilled from the knowledge between its deeper and shallower layers. A matching of the posterior distributions of a model between intra-class data was introduced by [125].

3.2.3 Semi-supervised and Self-supervised Learning

Several semi-supervised and self-supervised learning methods are investigated. Concurrently, several semi-supervised and self-supervised methods [10, 11, 30, 34, 40, 58, 99, 127] have a similar idea with our work. There are several works on distilling via model ensemble for uncertainty estimation [26, 65, 73, 74].

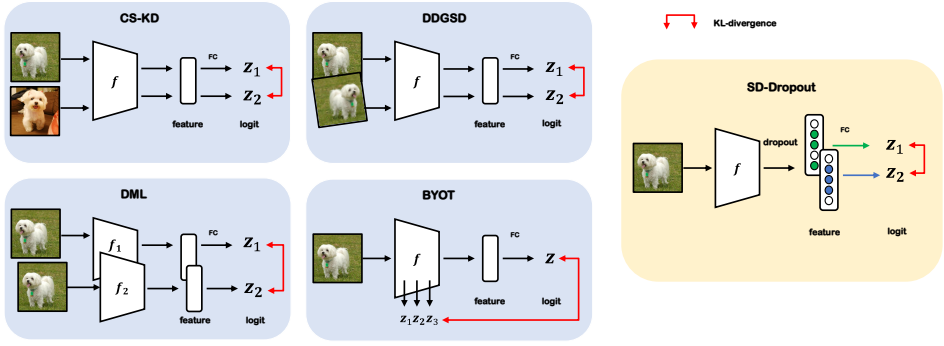


Figure 3.1: Self-Knowledge Distillation Methods

3.3 Self Distillation via Dropout

Throughout this study, we focus on supervised classification tasks. We denote $\mathbf{x} \in \mathcal{X}$ as the input data and $y \in \mathcal{Y} = \{1, 2, ..., N\}$ as its ground-truth label class. Let $f(\mathbf{x})$ be a global feature vector of the input data $\mathbf{x}$, and let $h(\cdot)$ be the last fully connected layer in a network. Now, we define $\mathbf{z} = \mathcal{M}_\theta(\mathbf{x}) = h(f(\mathbf{x}))$ as the logit of the output layer, where $\mathcal{M}_\theta$ is the neural network parametrized by θ. In classification tasks, neural networks typically use a *softmax* classifier to produce class posterior probability. Thus, we can consider that the posterior probability of class i is as follows:

$$p(y = i|\mathbf{x}; \theta, T) = \frac{exp(z^i/T)}{\sum_j^N exp(z^j/T)}, \tag{3.1}$$

where z^i as the logit of class i and $T > 0$ is the temperature, which is usually set to 1. In knowledge distillation, the temperature T is set to greater than 1.

3.3.1 Method Formulation

In this section, we introduce a new self-knowledge distillation method called SD-Dropout. We use the dropout layer after all feature extraction layers. We define

$$\mathcal{M}_\theta^{\mathbf{u}}(\mathbf{x}) = h(\mathbf{u} \odot f(\mathbf{x})) \qquad \text{where } u^j \sim \text{Bernoulli}(\beta) \tag{3.2}$$

where $\odot$ is the element-wise product, and β is the dropout rate. Now, $\mathcal{M}_\theta^u$ is the neural network using a dropout and it produces the posterior probability $p(y|\mathbf{x}; \mathbf{u}, \theta, T)$. For brevity, we denote $p_\theta^{\mathbf{u}}(y|\mathbf{x}) := p(y|\mathbf{x}; \mathbf{u}, \theta, T)$. Similarly, we can also extract an additional feature vector $\mathbf{v} \odot f(\mathbf{x})$, where $v^j \sim \text{Bernoulli}(\beta)$. Thus, we can define $p_\theta^{\mathbf{v}}(y|\mathbf{x})$.

We propose a new regularization loss to distill knowledge by reducing the KL-divergence between two logits $\mathcal{M}_\theta^{\mathbf{u}}(\mathbf{x})$ and $\mathcal{M}_\theta^{\mathbf{v}}(\mathbf{x})$. Our method is visualized in Figure 3.1. This method has computational advantages because, unlike conventional methods, it uses a single existing model, does not require additional modules, shares an encoder, and only requires post fully connected layer operations. Because the two features have no superior relationship with each other, we use this loss in a symmetric manner.

As a result, we use the forward and reverse KL-divergence of both instances. Further discussion on this matter is provided in Section 3.3.3. Formally, given an input data $\mathbf{x}$, label y, and randomly dropped operations $\mathbf{u}, \mathbf{v}$, the loss of the SD-Dropout method is defined as follows:

$$\begin{aligned} \mathcal{L}_{SDD}(\mathbf{x}; \mathbf{u}, \mathbf{v}, \theta, T) := {} & D_{KL}(p_\theta^{\mathbf{u}}(y|\mathbf{x}) || p_\theta^{\mathbf{v}}(y|\mathbf{x})) \\ & + D_{KL}(p_\theta^{\mathbf{v}}(y|\mathbf{x}) || p_\theta^{\mathbf{u}}(y|\mathbf{x})). \end{aligned} \tag{3.3}$$

Our method matches the predictions of different dropout features from a single network, whereas the conventional knowledge distillation method matches predictions from a teacher and a student network. Thus, the total loss $\mathcal{L}_{Total}$ is defined as follows:

$$\begin{aligned}\mathcal{L}_{Total}(\mathbf{x}, y; \mathbf{u}, \mathbf{v}, \theta, T) = \mathcal{L}_{CE}(\mathbf{x}, y;\ \theta) \\ + \lambda_{SDD} \cdot T^2 \cdot \mathcal{L}_{SDD}(\mathbf{x}; \mathbf{u}, \mathbf{v}, \theta, T)\end{aligned} \tag{3.4}$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss and λ_{SDD} is the weight hyperparameter of the SD-Dropout method.

3.3.2 Collaboration with other method

We visualize other self-knowledge distillation methods in diagram forms in Figure 3.1. Self-knowledge distillation methods use the KL-divergence of logits as a loss function. All knowledge distillation methods differ in their methodologies used to obtain their KL-Divergences. Data-Distortion Guided Self-Distillation (DDGSD) [115] is a method for distilling knowledge between different distorted data. DDGSD utilizes distorted instances of the same training data to minimize the KL-divergence between the two distorted data. The distortion data can be generated through a horizontal flip and random crop augmentation. In addition, Be Your Own Teacher (BYOT) [131] is a self-distillation method that distills knowledge between its deeper and shallow layers. First, the network is divided into several blocks. The knowledge of the deeper block is then transferred to the shallow portion. Thus, additional modules are required to extract intermediate posterior distributions. Class-wise self-knowledge distillation (CS-KD) [125] is a method of matching the posterior distributions of a model between intra-class instances. In the training procedure, an

instance is randomly sampled, which is the same class as the training instance, and KL-divergence is measured between two instances. Deep Mutual Learning (DML) [133] is an online knowledge distillation method in which a teacher network and a student network distill the knowledge from each other. The DML method trains two networks from scratch.

Our method can easily collaborate with various self-knowledge distillation methods because it has no additional module or training scheme constraints. In collaboration, the loss can be described as Eq. (3.5), where $\mathcal{L}_{KD}$ is an additional self-knowledge distillation loss for collaboration.

$$\begin{aligned}\mathcal{L}_{Total}(\mathbf{x}, y; \mathbf{u}, \mathbf{v}, \theta, T) = \mathcal{L}_{CE}(\mathbf{x}, y;\ \theta) \\ + \lambda_{SDD} \cdot T^2 \cdot \mathcal{L}_{SDD}(\mathbf{x}; \mathbf{u}, \mathbf{v}, \theta, T) \\ + \lambda_{KD} \cdot T^2 \cdot \mathcal{L}_{KD}(\mathbf{x}; \theta, T)\end{aligned} \tag{3.5}$$

where λ_{KD} is the weight hyperparameter of the other distillation methods. The discussion of the appropriate λ_{KD} value is detailed in Table 3.1.

3.3.3 Forward versus reverse KL-Divergence

Let $p_\theta(\mathbf{x})$ and $q_\theta(\mathbf{x})$ be the probability distributions. Let N denote the size of input vector $\mathbf{x}$. Then, two kinds of KL-divergences, forward and reverse KL-divergence, are defined as follows:

$$D_{KL}^{fw.}(p_\theta, q_\theta) = D_{KL}(p||q_\theta) + D_{KL}(q||p_\theta) \tag{3.6}$$

$$D_{KL}^{bw.}(p_\theta, q_\theta) = D_{KL}(p_\theta||q) + D_{KL}(q_\theta||p). \tag{3.7}$$

Note that the absence of θ in p indicates that p is considered constant with respect to θ, meaning that the gradient is not propagated.

In the field of knowledge distillation, it is widely accepted that the forward KL-divergence is based on a similarity with the Cross-Entropy loss. That is, the features of the teacher network are considered as the ground-truth, and the features of the student network are considered as logits that approximate the features of the teacher networks. However, we also adopt a reverse KL-divergence direction to further reduce the divergence between the two distributions. According to the proposition below, we can see that the derivative of reverse divergence is stronger than that of forward divergence. Therefore, by adding reverse divergence, we expect stronger self-knowledge distillation.

First, we observe the representation of the derivatives of forward and reverse KL-divergence.

Lemma 3.1. *The derivatives of forward and reverse divergence is represented as follows:*

$$\nabla_\theta D_{KL}^{fw.}(p_\theta, q_\theta) = \sum_{i=1}^{N}(1 - \frac{p(\mathbf{x})_i}{q(\mathbf{x})_i})\nabla_\theta q(\mathbf{x})_i + \sum_{i=1}^{N}(1 - \frac{q(\mathbf{x})_i}{p(\mathbf{x})_i})\nabla_\theta p(\mathbf{x})_i \tag{3.8}$$

$$\nabla_\theta D_{KL}^{bw.}(p_\theta, q_\theta) = \sum_{i=1}^{N}\log(\frac{p(\mathbf{x})_i}{q(\mathbf{x})_i})\nabla_\theta p(\mathbf{x})_i + \sum_{i=1}^{N}\log(\frac{q(\mathbf{x})_i}{p(\mathbf{x})_i})\nabla_\theta q(\mathbf{x})_i \tag{3.9}$$

(Proof) Let $Z_q = \sum_{i=1}^{N} q(\mathbf{x})_i$. Because $\sum_{i=1}^{N} q(\mathbf{x})_i = 1$,
we have $Z_q = 1$ and $q_\theta = \frac{q_\theta}{Z_q}$. Then, we can calculate the forward derivative as

$$\begin{aligned}
&\nabla_\theta D_{KL}(p, q_\theta) \\
&= \nabla_\theta(\sum_{i=1}^{N} p(\mathbf{x})_i(\log p(\mathbf{x})_i - \log q_\theta(\mathbf{x})_i + \log Z_q)) \\
&= \sum_{i=1}^{N} p(\mathbf{x})_i(-\frac{\nabla_\theta q_\theta(\mathbf{x})_i}{q_\theta(\mathbf{x})_i} + \frac{\nabla_\theta Z_q}{Z_q}) \\
&= \sum_{i=1}^{N} -\frac{p(\mathbf{x})_i}{q(\mathbf{x})_i}\nabla_\theta q_\theta(\mathbf{x})_i + \frac{\nabla_\theta Z_q}{Z_q}\sum_{i=1}^{N} p(\mathbf{x})_i
\end{aligned} \tag{3.10}$$

Because $Z_q = 1, \sum_{i=1}^{N} p(\mathbf{x})_i = 1$, and $\nabla_\theta Z_q = \sum_{i=1}^{N} \nabla_\theta q(\mathbf{x})_i$, we finally obtain

$$\nabla_\theta D_{KL}(p, q_\theta) = \sum_{i=1}^{N} \nabla_\theta q(\mathbf{x})_i (-\frac{p(\mathbf{x})_i}{q_\theta(\mathbf{x})_i} + 1). \tag{3.11}$$

Similarly, we can calculate the following reverse derivative:

$$\begin{aligned}
&\nabla_\theta D_{KL}(p_\theta, q) \\
&= \nabla_\theta \sum_{i=1}^{N} p_\theta(\mathbf{x})_i (\log p_\theta(\mathbf{x})_i - \log Z_p - \log q(\mathbf{x})_i) \\
&= \sum_{i=1}^{N} \nabla_\theta p_\theta(\mathbf{x})_i (\log p_\theta(\mathbf{x})_i - \log Z_p - \log q(\mathbf{x})_i) \\
&+ \sum_{i=1}^{N} p_\theta(\mathbf{x})_i (\frac{\nabla_\theta p_\theta(\mathbf{x})_i}{p_\theta(\mathbf{x})_i} - \frac{\nabla_\theta Z_p}{Z_p}) \\
&= \sum_{i=1}^{N} \nabla_\theta p_\theta(\mathbf{x})_i (\log \frac{p_\theta(\mathbf{x})_i}{q(\mathbf{x})_i} - \log Z_p) \\
&+ \sum_{i=1}^{N} \nabla_\theta p_\theta(\mathbf{x})_i - \frac{\nabla_\theta Z_p}{Z_p} \sum_{i=1}^{N} p_\theta(\mathbf{x})_i \\
&= \sum_{i=1}^{N} \nabla_\theta p_\theta(\mathbf{x})_i \log \frac{p_\theta(\mathbf{x})_i}{q(\mathbf{x})_i}. \quad \square
\end{aligned} \tag{3.12}$$

Before beginning the main proposition, we make the following assumptions.

Assumption 3.2. *If* $|p(\mathbf{x})_i| > |q(\mathbf{x})_i|$, *then* $|\nabla_\theta p(\mathbf{x})_i| > |\nabla_\theta q(\mathbf{x})_i|$. *If* $|p(\mathbf{x})_i| < |q(\mathbf{x})_i|$, *then* $|\nabla_\theta p(\mathbf{x})_i| < |\nabla_\theta q(\mathbf{x})_i|$. *i.e.,* $(|\frac{p(\mathbf{x})_i}{q(\mathbf{x})_i}| - 1)(|\frac{\nabla_\theta p(\mathbf{x})_i}{\nabla_\theta q(\mathbf{x})_i}| - 1) > 0$.

Assumption 3.3. *Let* $r = \log(|\frac{p(\mathbf{x})_i}{q(\mathbf{x})_i}|)$ *and* $\rho = |\frac{\nabla_\theta p(\mathbf{x})_i}{\nabla_\theta q(\mathbf{x})_i}| > 1$. *Then* $r \leq r_1$ *where* $r_1 = |\log(\rho) + \log(\log(\rho + (e - 1)))|$.

Assumption 3.2 implies that if $|p(\mathbf{x})|$ is greater than $q(\mathbf{x})|$, then the derivative $\nabla_\theta p(\mathbf{x})$ is also greater than $\nabla_\theta q(\mathbf{x})$. Assumption 3.3 implies that the ratio $|\frac{p(\mathbf{x})}{q(\mathbf{x})}|$

is not significantly different from the ratio $|\frac{\nabla_\theta p(\mathbf{x})}{\nabla_\theta q(\mathbf{x})}|$. We empirically validate that Assumptions 3.2 and 3.3 hold in probability during the experiment. Now, we posit the main proposition indicating that the reverse derivative is greater than the forward derivative under Assumptions 3.2 and 3.3. In other words, we can demand a stronger connectedness (or bond) between logits $p(\mathbf{x})$ and $q(\mathbf{x})$ in the training process by adding reverse derivatives.

Proposition 3.4. *Under Assumptions 3.2 and 3.3, let:*

$$\begin{aligned}(D_{i,\theta}) = &|\log(\frac{p(\mathbf{x})_i}{q(\mathbf{x})_i})\nabla_\theta q(\mathbf{x})_i| + |\log(\frac{q(\mathbf{x})_i}{p(\mathbf{x})_i})\nabla_\theta p(\mathbf{x})_i| \\ &-(\ |(1-\frac{p(\mathbf{x})_i}{q(\mathbf{x})_i})\nabla_\theta q(\mathbf{x})_i| + |(1-\frac{q(\mathbf{x})_i}{p(\mathbf{x})_i})\nabla_\theta p(\mathbf{x})_i|\ ).\end{aligned} \tag{3.13}$$

Then we have:

$$(D_{i,\theta}) > 0. \tag{3.14}$$

Moreover, (D_i) has a maximum value at $r = |\log(\rho)|$. Here, (D_i) implies the difference between the L_1 norm of the reverse derivatives and the L_1 norm of the forward derivatives.

(Proof) For $i \in [N]$, without a loss of generality, we set $|p(\mathbf{x})_i| \geq |q(\mathbf{x})_i|$, *that is*, $r \geq 1$. By Assumption 3.2, we take $|\nabla_\theta p(\mathbf{x})_i| = \rho|\nabla_\theta q(\mathbf{x})_i|$ for $\rho > 1$. Then,

$$\begin{aligned}&|(\log(\frac{p(\mathbf{x})_i}{q(\mathbf{x})_i})\nabla_\theta q(\mathbf{x})_i| + |(\log(\frac{q(\mathbf{x})_i}{p(\mathbf{x})_i})\nabla_\theta p(\mathbf{x})_i| - \\ &|(1-\frac{p(\mathbf{x})_i}{q(\mathbf{x})_i})\nabla_\theta q(\mathbf{x})_i| + |(1-\frac{q(\mathbf{x})_i}{p(\mathbf{x})_i})\nabla_\theta p(\mathbf{x})_i| = \\ &[(1+\rho)r - \{(e^r-1) + \rho(1-e^{-r})\}]|\nabla_\theta q(\mathbf{x})_i|.\end{aligned} \tag{3.15}$$

Let

$$k(r) := (1+\rho)r - \{(e^r - 1) + \rho(1 - e^{-r})\}. \tag{3.16}$$

Then, we have $k'(r) = (1 - e^r) + \rho(1 - e^{-r}) = -e^{-r}(e^{2r} - (1+\rho)e^r + \rho) = -e^{-r}(e^r - 1)(e^r - \rho)$. Thus, $k'(r)$ has roots at $r = 0$ and $r = \log(\rho)$. Because $k(0) = k'(0) = 0$, $k(r)$ increases in $r \in (0, \log(\rho))$. Furthermore, $k(r)$ has a maximum value $k(\log(\rho)) = (1+\rho)\log(\rho) - 2\rho + 2$ at $r = \log(\rho)$.
Now, we show $k(\log(\rho)+\log(\log(\rho+(e-1)))) \geq 0$. Let $l(\rho) = k(\log(\rho)+\log(\log(\rho+(e-1))))$, and

$$\begin{aligned} l(\rho) &= (\rho+1)(\log(\rho) + \log(\log(\rho + (e-1)))) \\ &- \rho(\log(\rho + (e-1)) + 1) + \log(\rho + (e-1))^{-1} + 1. \end{aligned} \tag{3.17}$$

The derivative of $l(\rho)$ is :

$$\begin{aligned} l'(\rho) &= -((\rho + (e-1))\log(\rho + (e-1))^2)^{-1} \\ &- (\rho\log(\rho + (e-1)))^{-1} - \rho(\rho + (e-1))^{-1} \\ &+ (\rho+1)(\rho + (e-1))^{-1}\log(\rho + (e-1))^{-1} + 1 \\ &+ \rho^{-1} - \log(\rho + (e-1)) - 1 + \log(\rho) \\ &+ (\rho\log(\rho + (e-1)))^{-1}\log(\log(\rho + (e-1))). \end{aligned} \tag{3.18}$$

Then, we shall prove the following lemma:

Lemma 3.5. *$l'(\rho)$ has local minimum at $\rho = 1$ with $l(1) = 0$.*

(Proof) Because $l''(\rho) > 0$ for $\rho \geq 1$, $l'(\rho)$ is convex for $\rho \geq 1$. In addition, because $l'(1) = 1$, $l'(\rho)$ has a local minimum at $\rho = 1$ with $l(1) = 0$. Therefore,

$l'(\rho) \geq 0$ for $\rho \geq 1$. Since $l(0) = 0$, we can conclude that $l(\rho) \geq 0$ for $\rho \geq 1$. $\square$

3.4 Experiments

In this section, we present the effectiveness of the proposed network. We demonstrate our method on multiple datasets, various backbone models, the adversarial robustness, and expected calibration error.

3.4.1 Implementation Details

Dataset To validate the general performance of our method, we test various datasets including CIFAR-100 [55], CUB-200-2011 [102], and Stanford Dogs [54]. CIFAR-100 is composed of 100 classes with large contextual differences between classes. By contrast, CUB-200-2011 and Stanford Dogs are composed of 200 and 120 fine grained classes, and unlike CIFAR-100, there are smaller contextual differences between classes. The Performance in various dataset domains can be used to evaluate the overall performance of the model.

Hyperparameters For a fair comparison, we use the same hyperparameters in all experiments unless specifically mentioned. We use a stochastic gradient descent optimizer (learning rate=0.1, momentum=0.9, weight decay=1e-4) and train 200 epochs during all experiments. The learning rate is scheduled for decay 0.1 on 100 and 150 epochs. As a common setting for CIFAR-100, we set a batch size of 128 and use ResNet, which modifies the first convolution layer with a 3×3 kernel instead of a 7×7 kernel. We set input image sizes of 224×224 and batch sizes of 32 on CUB-200-2011 and Stanford Dogs. All results are the average values obtained by repeating the experiment three times.

Table 3.1: Accuracy (%) comparison of ResNet-18 on CIFAR-100 dataset over various hyper-parameters β and λ_{SDD}. Best result is indicated in bold.

λ_{SDD} \ β	0.1	0.3	0.5	0.7
0.1	76.31	76.47	75.58	76.91
0.5	75.83	76.31	76.88	76.43
1.0	75.72	76.75	**77.10**	76.82
2.0	76.86	76.79	77.07	76.91
5.0	76.92	76.79	76.57	69.47

Training Procedure The training procedure of SD-Dropout is summarized as PyTorch-like style pseudo code, as described in Algorithm 1. It should be noted that we do not use the .detach() method to calculate the KL-divergence terms to maintain both directions of KL-divergence.

3.4.2 Results

Classification results

While using the fixed backbone model (ResNet-18), we compare the accuracy of the methods and datasets. Table 3.2 shows the accuracy on CIFAR-100, CUB-200-2011, and Stanford Dogs in comparison with distillation and regularization methods, i.e., Cross-Entropy, CS-KD, DDGSD, BYOT, DML, and label smoothing (LS). Moreover, the [+ SD-Dropout] column is the result of a collaboration between existing methods. The dropout collaboration improves the performance for all datasets and self-distillation methodologies. Compared to previous self-distillation methods, although simply applicable, the SD-Dropout method performs best on the CUB-200-2011 dataset at 66.6%. Furthermore, the largest increase in the performance

Algorithm 1: Pseudo code of SD-Dropout in a PyTorch-like style.

```
# x: input image
# y: ground-truth label
# backbone : feature extractor composed of convolutions
# classifier : fully connected layer
# lambda: weight hyperparamter of knowledge distillation method
for (x, y) in Batches:
    # extract the feature
    feat = backbone(x)
    # the values of the logits
    output = classifier(feat)
    # calculate cross-entropy loss
    loss_ce = ce_loss(output, y)
    # sampling two features by dropout
    feat_dps = [dropout(feat) for _ in range(2)]
    # the logits of two dropout sampled feature
    output_dp1, output_dp2 =
        [classifier(feat_dps[i]) for i in range(2)]
    # Foward reverse KL-divergence between two logits
    loss_kd1 = kl_div_loss(output_dp1, output_dp2)
    loss_kd2 = kl_div_loss(output_dp2, output_dp1)
    loss_kd = loss_kd1 + loss_kd2
    # total loss
    loss_total = loss_ce + lambda*loss_kd
    # update
    loss_total.backward()
```

showed that the compatibility with BYOT is the best (0.5% on CIFAR-100, 8.1% on CUB-200-2011, and 2.5% on Stanford Dogs).

Backbone Network

Our self-distillation method is easily adaptable to various backbone models. We compare several backbone networks with and without SD-Dropout. We apply SD-Dropout to ResNet-18, ResNet-34, and DenseNet-121. Table 3.3 shows that the SD-Dropout method can improve the network performance regardless of the backbone networks. In particular, our SD-Dropout improves the accuracy of the baseline networks from 74.8% to 77.0% for ResNet-18, and from 75.7% to 77.2% for ResNet-34 on the CIFAR-100 dataset. For DenseNet-121, the accuracy increased by 1.1%.

ImageNet Classification and Object Detection

To verify our method on large-scale dataset, we evaluate the classification tasks on ILSVRC 2012 dataset [19].

Table 3.2: Accuracy (%) of ResNet-18 with self-knowledge distillation methods on various image classification tasks.

	CIFAR-100		CUB-200-2011		Standford Dogs	
	Base	+SD-Dropout	Base	+SD-Dropout	Base	+SD-Dropout
Cross-Entropy	74.8	77.0 (+2.2)	53.8	66.6 (+12.8)	63.8	69.9 (+6.1)
CS-KD	77.3	77.4 (+0.1)	64.9	65.4 (+0.6)	68.8	69.3 (+0.5)
DDGSD	76.8	77.1 (+0.3)	58.3	62.9 (+4.6)	66.9	68.1 (+1.3)
BYOT	77.2	77.7 (+0.5)	60.6	68.7 (+8.1)	68.7	71.2 (+2.5)
DML	78.9	78.8 (-0.1)	61.5	65.7 (+4.2)	70.5	72.0 (+1.6)
LS	76.8	76.9 (+0.1)	56.2	67.6 (+11.5)	65.2	70.1 (+4.9)

Table 3.3: Accuracy (%) comparison with different backbone networks on CIFAR-100.

	Base	+SD-Dropout
ResNet-18	74.8	77.0
ResNet-34	75.7	77.2
DenseNet-121	77.3	78.4

Table 3.4: Robustness comparison against the adversarial attack. Accuracy (%) of ResNet-18 on various datasets.

Dataset	Base	+SD-Dropout
CIFAR-100	37.9	47.1
CUB-200-2011	17.0	24.8
Stanford Dogs	19.2	22.6

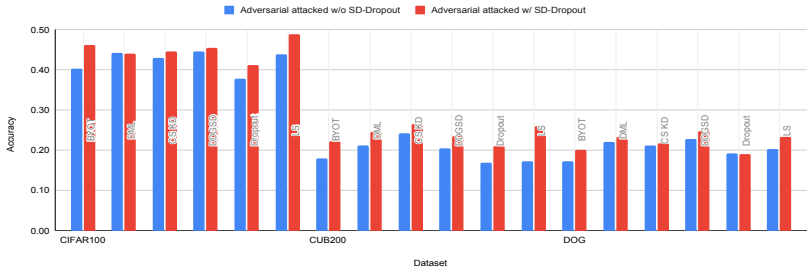


Figure 3.2: Adversarial robustness with collaborate cases. the blue bar denotes model accuracy when attacked by a model learned by the base self-KD method. The red bar denotes model accuracy when attacked by a model that trained with SD-dropout.

In addition, to verify our method on the objection task, we conduct the experiment on the Faster R-CNN [89] object detection model using the COCO dataset [67].

Out-of-Distribution Task

We verify our method on out-of-distribution tasks. We utilize ODIN detector [66] on LSUN [120], iSUN [114], DTD [15], and SVHN [80] datasets. The experimental results are shown in Table 3.8.

Table 3.5: ECE comparison results of SD-Dropout combined with various KD methodologies. Lower is better.

Method	Base	+SD-Dropout
CrossEntropy	0.120	0.075
DDGSD	0.067	0.034
CS-KD	0.068	0.046
BYOT	0.117	0.056
DML	0.058	0.039

Table 3.6: Results on object detection. mAP@0.5 denotes mean average precision with IOU threshold 0.5. mAP denotes COCO-style mAP. Best results are indicated in bold.

Method	0.5 × Schedule	1 × Schedule
	mAP / mAP@0.5	
Cross-Entropy	31.2 / 50.6	39.4 / 60.1
SD-Dropout	**32.5** / **52.8**	**39.8** / **60.7**

Robustness to adversarial attack

To evaluate the robustness to adversarial attacks [23], in Table 3.4, we compare the accuracy of the baseline models with accuracy of the models learned through the SD-Dropout method. An adversarial attack approach is the Fast gradient sign method [33] that exploits the gradient of network with respect to the input image to increase the loss, where the maximum perturbation size is ($\epsilon = 0.2$).

As shown in Figure 3.2, we demonstrate the adversarial robustness with collaborate cases. The blue bar denotes the accuracy when attacked by a model learned using the original self-knowledge distillation method. The red bar denotes the accuracy when attacked by a model trained using SD-Dropout. All results show an

Table 3.7: Results on object detection. mAP@0.5 denotes mean average precision with IOU threshold 0.5. mAP denotes COCO-style mAP. Best results are indicated in bold.

Method	0.5 × Schedule	1 × Schedule
	mAP / mAP@0.5	
Cross-Entropy	31.2 / 50.6	39.4 / 60.1
SD-Dropout	**32.5** / **52.8**	**39.8** / **60.7**

Table 3.8: Result on the out-of-distribution dataset. The CIFAR-100 dataset is used as an in-distribution dataset. ↑ means larger is better, and ↓ means lower is better. The trained network is ResNet-18. Best results are indicated in **bold**.

Dataset	FPR (at 95% TPR) ↓	Detection Error ↓	AUROC ↑	AUPR (in) ↑	AUPR (out) ↑
	Cross-Entropy/SD-Dropout (%)				
LSUN	74.7/**63.5**	25.0/**22.9**	82.0/**84.9**	84.5/**84.8**	78.5/**83.4**
iSUN	76.0/**63.8**	25.3/**23.0**	82.0/**84.8**	**85.9**/85.6	76.0/**81.9**
DTD	82.7/**77.5**	28.6/**27.4**	**77.3**/77.2	**86.3**/83.7	59.8/**64.0**
SVHN	82.6/**75.2**	**22.7**/27.4	**82.9**/79.1	**86.6**/78.3	75.5/**76.7**
Average	79.0/**70.0**	25.4/**25.1**	81.1/**81.5**	**85.8**/83.1	72.4/**76.5**

increase in the robustness of the adversarial attacks. We conjecture that the SD-Dropout method has an effect to similar to that of adversarial training.

Calibration Effect

The expected calibration error (ECE) [78] is a metric that shows the difference between the confidence of the model predictions and the actual accuracy. The ECE

Table 3.9: Accuracy (%) comparison between dropout and SD-Dropout of ResNet-18.

	Base	+Dropout	+SD-Dropout
CIFAR-100	74.8	75.4	77.0
CUB-200-2011	53.8	64.6	66.6
Stanford Dogs	64.1	69.5	69.8

can be calculated as

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} \, |acc(B_m) - conf(B_m)|, \tag{3.19}$$

where M, n, B_m, $acc(B_m)$, and $conf(B_m)$ denote the number of bins, the number of total samples, the number of samples in the m-th bin, the average accuracy of samples in the bin, and the average model confidence of samples in the bin. We set the number of bins to 10. The results shown in Table 3.5 and Figure 3.3 indicate that SD-Dropout suppresses overconfidence and improves confidence calibration.

Comparison to conventional Dropout

The dropout technique plays the most important role in the SD-Dropout method. To demonstrate the importance of dropout distillation with our method, we compare it to networks using conventional dropout methods. Table 3.9 compares the experimental results of the conventional dropout and our SD-Dropout methods on the CIFAR-100, CUB-200-2011, and Stanford Dogs datasets. It is observed that our SD-Dropout method outperforms the conventional dropout method on all datasets. For CIFAR-100, the SD-Dropout method increases by 2.2% over the baseline network ResNet-18, whereas the conventional dropout method increases by 0.6%. Fur-

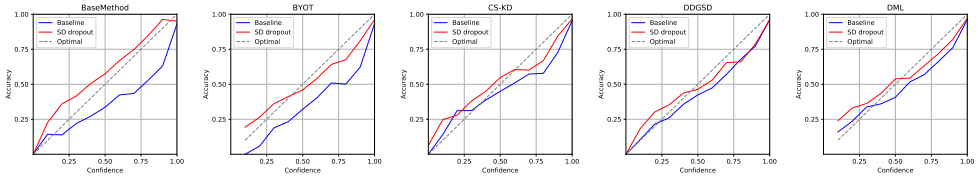


Figure 3.3: Reliability diagrams ([82]). The x-axis is confidence bin and the y-axis is average accuracy on bin. The blue and red line show average accuracy w/ and w/o SD-dropout method.

Table 3.10: Accuracy (%) comparison between the forward, reverse, and both directions (forward and reverse) of KL-divergence on ResNet-18. Best results are indicated in bold.

Dataset	Base	Forward	Reverse	Both Directions
CIFAR-100	74.8	76.6	76.3	**77.0**
CUB-200-2011	53.8	65.4	63.8	**66.6**
Stanford Dogs	64.1	69.6	69.7	**69.8**

thermore, the SD-Dropout method increases the accuracy by 12.8% and 5.7% for the CUB-200-2011 and Stanford Dogs datasets, whereas the conventional dropout method increases by 10.6% and 5.3%, respectively.

Experiment on the direction of KL-divergence

We empirically verify through experiments that Assumptions 3.2 and 3.3 in Section 3.3.3 are convincing (see Table 3.11). We use ResNet-18 on the CIFAR-100 dataset in the experiment. The probability that Assumption 3.2 holds is greater than 0.5 in all epoch. L1 norm of r in Assumption 3.3 is smaller than r_1 in all epochs. In addition, as shown in Table 3.10, the model using both directions of KL-divergence achieves the higher performance.

Table 3.11: Verification of assumptions in Section 3.3.3

Epoch	P(Assumption 3.2)	r	r_1
0	0.638	0.0681	0.1368
100	0.653	0.1403	0.2799
200	0.594	0.1292	0.2662

3.5 Conclusion

We propose a new and simple self-knowledge distillation method. This method samples different models through a dropout and distills the knowledge of both. Because they were sampled using a dropout, the sampled model does not have superiority over the other models, and knowledge is shared among them using forward and reverse KL-divergence. We also experimentally and analytically show the characteristics of reverse KL-divergence. We demonstrate that the proposed method improves generalization, calibration performance, and adversarial robustness. From the perspective of the regularization domain, our method is superior to the conventional label smoothing method through multiple datasets. Thus, we expect our method to be used as a regularization method that can effectively improve the performance of a single network in various domains.

Chapter 4

Membership inference attacks against object detection models

4.1 Introduction

Over the past few years, deep neural networks have been widely adopted in various computer vision tasks such as image classification, object detection, and semantic segmentation. Many deep learning models in various fields have been developed using a wide variety of data. These data often contain privately sensitive information such as medical records, personal photos, personal profiles, and financial information. If designed without considering adversarial threats, the model can leak sensitive information of the dataset it has trained. In [91], it was demonstrated that even with black-box access, an adversary can conduct a membership inference attack that determines whether a data record is a part of the training set.

Early studies on membership inference attacks have focused on classification tasks [2]. Several other adversarial attacks against the object detection model have

been studied, the results of which indicate the potential leakage of the model[113, 111]. Through this study, we have begun extending the membership inference attack to object detection tasks.

Datasets used in an object detection model can also be subject to privacy leaks. Examples of such data include outdoor pedestrian data, photos with sensitive text, and video data for autonomous driving. The membership inference of detection models can be helpful to assess whether data are collected illegally for training purposes, and attack vulnerability can be viewed as a gateway to further attacks.

Compared to classifiers, there are difficulties in attack detection models: 1) In classification tasks, only the last logit of the same size is regarded, whereas in object detection tasks, all predictions based on the location of the objects are of concern. 2) Object detection tasks may have multiple objects in a single image, whereas a usual image classification task has a single object. To address these issues, we propose the canvas method for attacking an object detection model and tracing the differences in the views among the trained and test data.

In summary, this paper makes the following contributions:

- We first propose a new membership inference attack on object detection models with black-box access. We describe the proposed canvas method, which draws a predicted bounding box distribution on an empty canvas for convolutional neural network (CNN) classification networks. Using this method, we can achieve a higher performance than conventional machine learning methods on the PASCAL VOC dataset.

- We found experimentally that our attack method is robust to various types of object detection models. In addition, we showed that membership infer-

ence attacks are also successful on privately sensitive data with seemingly little difference between accuracy of the training and test datasets. We also conducted a transfer attack between different models and datasets.

- We suggest the use of defense methods applying a differentially private algorithm. Experiment results show that the differentially private (DP) algorithm can defend against a membership inference with a calculated amount of privacy loss.

4.2 Background and Related Work

4.2.1 Membership Inference Attack

The end of a membership inference attack is to determine whether the given data record is in the training dataset of the target model. A membership inference attack is based upon the assumption that the target model has a different view of the training data than that of test data that was not seen before. Although overfitting is considered to be a root cause of this membership disclosure, it cannot be the only cause [70]. The attack model may have black-box and white-box access to the target model. Under the white-box access scenario, the attack model has access to certain versions of input data or intermediate layers as well as trained parameters of the target model. White-box knowledge is powerful but not realistic because the target model may not provide detailed information. In a black-box setting, the attacker does not have direct access to the target model parameters. The attack model can only access the input data and the model output predictions. The attack model should identify the difference between the inferred predictions of the training and test samples of the target model. To achieve this aim, shadow models trained

using the same algorithm are built on shadow datasets sampled from a similar distribution as the target datasets but do not contain the target training data. The attack model queries the shadow model and learns to distinguish whether the shadow model output comes from the training set.

Shokri et al. shokri first presented the first membership inference attack against machine learning models. Ahmed et al. ml-leaks enhanced an attack by relaxing some of the assumptions. Hayes et al. hayes2019logan describes a membership inference attack against generative models. To mitigate the risk of a membership inference, Rahman et al. rahman2018membership and Nasr et al. nasr2018machine designed differentially private models and devised an adversarial regularization, respectively.

4.2.2 Object Detection

Object detection is a widely used computer vision task that deals with detecting an instance of a semantic objects in images or videos. There are mainly two types of methods for object detection using deep learning, namely, one-stage and two-stage detection.

One-Stage Detection One-stage detectors such as YOLO [87] or SSD [69] treat an object detection problem as an end-to-end simple regression problem. The one-stage model directly predicts the class scores and bounding box coordinates concurrently.

Two-Stage Detection A two-stage detection model such as Faster R-CNN [89] is divided into two stages. The model first generates region proposals by narrowing down the number of possible object locations by filtering out most of the back-

ground samples on a region proposal network (RPN). The model then passes the proposals through the CNN head to classify the labels and regress the bounding boxes.

4.2.3 Datasets

PASCAL VOC Dataset (2007,2012) [27] PASCAL VOC datasets have been widely adopted as benchmark datasets in basic object detection tasks. The PASCAL VOC datasets consist of VOC2007 and VOC2012. The datasets contain 20 object categories including people, bicycles, birds, bottles, dogs, etc.

INRIA Pedestrian Dataset [18] The INRIA Pedestrian dataset is popular for pedestrian detection, which consists of 614 images for training and 288 images for testing.

SynthText [36] The SynthText dataset is a synthetically generated text dataset in which several words are placed in imgaes of natural scenes. The dataset consists of approximately 800 thousand images and 8 million synthetic word instances in various languages.

4.3 Attack Methodology

In this section, we propose a membership inference attack for object detection models. An overview of the membership inference attack is illustrated in Figure 5.1. The setting of our membership inference attack is as followes:

Assumption We assume that the adversary has black-box access to the target model. The adversary can obtain final logit values but no other specific intermediate layer weight information of the target models. For the given target

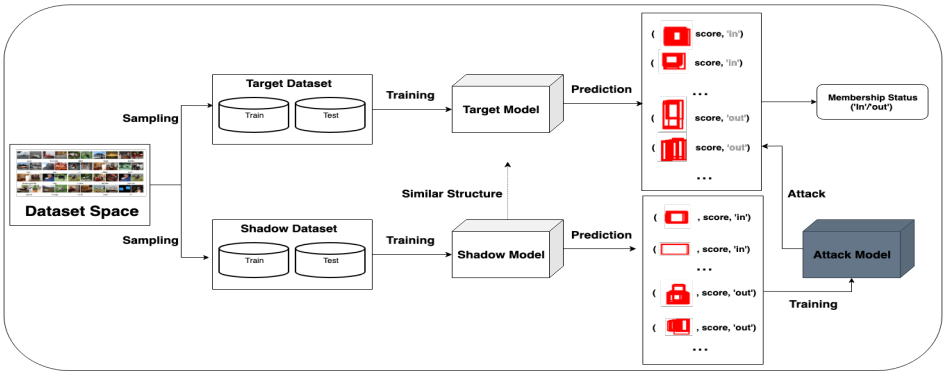


Figure 4.1: Overview of membership inference attack on object detection model. The target and shadow datasets are sampled from the same dataset space. The target model trains using its target dataset and the shadow model, which has a similar structure as the target model, trains using its shadow dataset. The predicted values of the target and shadow models are expressed as bounding boxes and their prediction scores along with their membership status labels ("in" for the training set "out" for the test set). Finally, the attack model which trains using the shadow model's prediction and membership status, attacks the target model by passing the target records, and estimates their membership status probabilities for each target example.

object detection model f_{target} and input image sample x_i the target model returns the proposed bounding boxes $bbox_j = ((x_j^0, y_j^0), (x_j^1, y_j^1))$ and prediction scores $s_j, (j = 1, 2, ..., N_b)$ where (x_j, y_j) and N_b denote the corner of the bounding box and the number of proposed boxes, respectively. In addition, the adversary can set a score threshold θ_{score} and non-maximum suppression (NMS) thresholds(θ_{nms} for one-stage, $\{ \theta_{nms}^{rpn}, \theta_{nms}^{head} \}$ for two-stage detectors) to customize the personal preference of the attacker. In addition, it is assumed that the target and shadow data do not overlap, i.e., $D_{shadow}^{train} \cap D_{target}^{train} = \varnothing$.

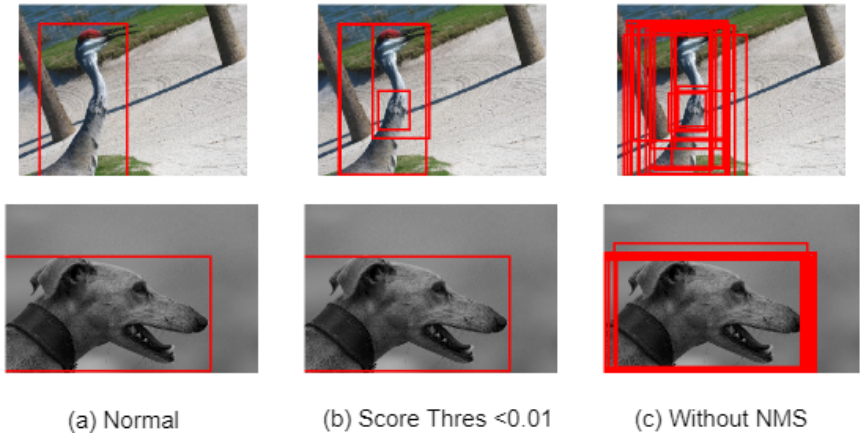


Figure 4.2: Predicted bounding boxes in training and test examples. The first row shows the training examples and their predicted boxes. Below are test examples and their predicted boxes.

4.3.1 Motivation

The basic idea of a membership inference attack is that the model has a different view on the trained data and unseen data. For a classification task, the model tends to achieve a high prediction score on the training samples over the test samples. Therefore, the attack model is able to classify the membership status using the last posterior logit value of a given sample. Similarly, as shown in Figure 4.2, the object detection model tends to achieve consistent box predictions on the training samples while showing an uncertainty regarding the test samples.

4.3.2 Gradient Tree Boosting

Gradient tree boosting is a widely used classification algorithm for numerous applications. Specifically, we use XG-BOOST [9], a popular algorithm applied feature classification, to distinguish whether a given example is in the training sample.

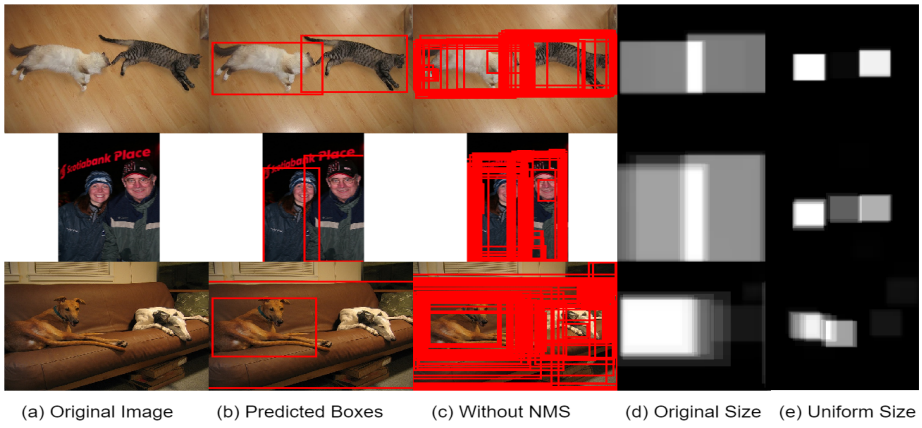


Figure 4.3: Examples of bounding box drawn canvas images using the proposed canvas method. The first row is the training data and the second and third-row images are the test data.

For the predicted bounding box coordinates and prediction scores $(bbox_j, s_j)$, we concatenate them in a long 1-D vector: $(x_1^0, y_1^0, x_1^1, y_1^1, s_1, ..., x_{N_b}^0, y_{N_b}^0, x_{N_b}^1, y_{N_b}^1, s_{N_b})$, and pad them with zero values to allow all vectors to have the same length. Using these vectors, we proceed with the membership classification using XG-BOOST.

4.3.3 Convolutional Neural Network Based Method

The next method applied to the attack model is CNN based approach. An object detection task differs from a classification because the model predicts 1) the box location information and 2) the bulk of the bounding boxes, most of which may be unhelpful. Therefore, we propose a new approach, called the canvas Method, to adequately process a predicted array for a CNN-based attack model.

Canvas Method In the object detection model, the model extracts numerous candidate boxes. Even for only a single object, the model predicts many predicted boxes. The NMS algorithm used in an object detection task is designed to filter

out messy boxes that are predicted for a single object and predict them as a single proposed box. Using NMS, the box with the highest score is first chosen and boxes that overlap above the threshold are filtered out. To see the clear distribution of predicted boxes before the NMS, the threshold of the NMS is set to be a high value during the prediction. Because the detection model shows a different positional variance in predicting the training and test samples, this location information is important in a CNN-based attack model. In addition, similar to a classification model, the model also shows a high prediction score in the trained samples, which is crucial to a membership inference.

To facilitate this information, we propose the use of the canvas method, which draws a predicted bounding box distribution on an empty canvas for a CNN classification network. The canvas is initially set to an image of 300×300 pixels in size, where every pixel has a value of zero and the boxes drawn on the canvas have the same center as the predicted boxes and the same intensity as the prediction scores. Regarding the size of the boxes drawn on the canvas, we applied two design approaches. The first one is drawing a box equal in size as the predicted box, and the other is to draw all boxes with an identical size on the canvas regardless of the original size of the predicted box. We call the first approach the original box size, and the second the uniform box size. We use the uniform box size to make objects of all sizes detected achieve the same effect on the canvas. We set the size of a uniform box at 10% of the canvas size. Figure 4.3 shows the examples of the canvas methods.

Augmentation Because a bounding box distribution in a canvas image should be robust to rotations and flipping, we adopt rotation and flipping when training the attack model. We do not apply other augmentation methods such as ran-

dom cropping or perspective transformation because these augmentations generate transformed bounding box distribution which might distract the target model's view on the training or test samples.

Score Rescaling The prediction score of the detection model's predicted bounding box refers to how confident the model is with the objectness of bounding boxes. Because the score values are calculated after the softmax layer, the values are between zero and one. With the canvas method, bounding boxes are drawn on the canvas at the same intensity as the prediction score, and the confidence of the model might not be fully represented. For example, if the model predicts two bounding boxes with scores of 0.9 and 0.9999 respectively, it indicates that the model is much more certain that the latter is an object. However, these values do not themselves represent a significant difference on the canvas. To emphasize the model's prediction scores of the model, we utilize a score rescaling function.

$$s_{rescale} = -\log(1 - s). \tag{4.1}$$

In a Taylor expansion, this function is represented as $-log(1-s) = s + \frac{s^2}{2} + \frac{s^3}{3} + ...$ Therefore in the case of an extremely small s, a rescale function is an approximate identity function, which means the rescaling has little effect on small scores. Using this function, the minuscule difference between the two scores (i.e. 0.0999 from 0.9 and 0.9999) is changed to 6.91 (from 2.30 and 9.21), which can be seen as significant.

4.3.4 Transfer Attack

We mitigate the assumption that the distribution of the target training data is similar to that of the shadow training data. In a realistic situation, it could be difficult or even impossible to secure a sufficient number of shadow data having the same distribution as the target data. Under this scenario, in [2], a transfer attack was proposed, which composes a shadow model with relatively common and similar object detection dataset. Although a shadow model has difficulty mimicking the target model's behavior owing to different statistics and appearances between two data distributions, the attack model is still expected to be able to capture the membership status of the given data.

On the other hand, the target model structure may be different. We also conducted another style of transfer attack, the shadow model structure of which differs from that of the target model.

4.4 Defense

To mitigate a membership inference against machine learning models, we propose several defense techniques.

4.4.1 Dropout

Because overfitting is a dominant reason why the target models leak their training data information, generalization techniques that prevent overfitting can help defend models against membership inferences. We adopt Dropout [95], to obtain a well-generalized model.

4.4.2 Differentially Private Algorithm

Differential privacy [24] offers a strong standard of privacy guarantees for computations involving aggregate datasets. It requires that any change to a single data point should reveal statistically indistinguishable differences from the model's output. A formal definition of differential privacy is described below: [(ϵ, δ) - Differential Privacy]

Given two neighboring datasets D and D', differing by only one record, a randomized mechanism $\mathcal{A}$ provides (ϵ, δ) - Differential Privacy if for $\forall S \subseteq Range(\mathcal{A})$,

$$\Pr[\mathcal{A}(D) \in S] \leq e^{\epsilon} \Pr[\mathcal{A}(D') \in S] + \delta. \tag{4.2}$$

We call this (ϵ, δ)-DP for short. If $\delta = 0$, $\mathcal{A}$ provides a stricter ϵ-DP. ϵ is called a privacy loss. To create a differentially private deep learning model, a differentially private stochastic gradient descent (DP-SGD) [1, 76, 93] is adopted to optimize the model. Compared to a conventional SGD optimizer, DP-SGD optimizer has two main changes to achieve the required privacy guarantee: adding Gaussian noise to gradient and gradient clipping for each minibatch sample. The specific algorithm is presented in Algorithm 2. Abadi el al. abadi2016deep showed a way to track a tight differential privacy bound of DP-SGD using moments accountant (MA). According to Yu et al. dp-publish, however, MA assumes random sampling with replacement which is impractical and is outperformed by random reshuffling [37]. Assuming sampling batches by random reshuffling, Yu et al. dp-publish showed that realistic privacy loss bound for DP-SGD is $(\rho + \sqrt{\rho \log(1/\delta)}, \delta)$-DP for $\rho = \frac{k}{2\sigma^2}$ where σ is noise scale and k is the number of epochs.

Algorithm 2: Differentially Private SGD

Input: Training examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\Sigma_i \mathcal{L}(\theta, x_i)$, learning rate η_t, group size L, noise scale σ_t gradient norm bound C

Initialize θ_0 randomly ;
for $t = 1 \ : \ T$ **do**
  data batching:
  Take a random batch of data samples $\mathbb{B}_t$ from the training dataset;
  $B = |\mathbb{B}_t|$;
  Compute gradient:
  For each $i \in \mathbb{B}_t$, $\mathbf{g}_t(x_i) \leftarrow \bigtriangledown_{\theta_t} \mathcal{L}(\theta_t, x_i)$;
  Clip gradient:
  $\hat{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/max\left(1, \frac{||\mathbf{g}_t(x_i)||_2}{C}\right)$;
  Add noise:
  $\widetilde{\mathbf{g}}_t \leftarrow \frac{1}{B}\left(\sum_i \hat{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma_t^2 C^2 \mathbb{I})\right)$;
  Descent:
  $\theta_{t+1} \leftarrow \theta_t - \eta_t \widetilde{\mathbf{g}}_t$;
end
Output : θ_T ;

4.5 Experiments

In this section, we describe the application of our method to several object detection tasks. To reduce confusion, we call the training dataset and test dataset of target and shadow models "in" and "out" data respectively. We used the Chainer framework for the object detection modules and Pytorch for the membership attack modules.

4.5.1 Target and Shadow Model Setup

Models To build target models, we train several object detection models including SSD and Faster R-CNN. For one-stage detection, the base SSD300-VGG16 and SSD512-VGG16 models use the VGG16 network as a backbone and have 300×300 images and 512×512 images as the inputs, respectively. The SSD300-Res50 model

uses ResNet50 network as a backbone. For two-stage detection, the Faster R-CNN model uses the VGG16 network as a backbone and receives images with a scale of between 600 and 800.

Datasets During the experiments, we used the datasets described earlier, i.e., VOC dataset, INRIA Pedestrian Dataset, and SynthText. According to Ahmed et al. ml-leaks, one shadow dataset is sufficient. For each dataset, D=(D^{train}, D^{test}), we split them by half into ($D^{in}_{target}, D^{out}_{target}$) and ($D^{in}_{shadow}, D^{out}_{shadow}$) to separate the target and shadow datasets. For SynthText dataset, we use the first 5,000 images with Latin characters for the target dataset and next 5,000 images for the shadow dataset.

Training To train the SSD model, we used an SGD optimizer with an initial learning rate 10^{-3}, 0.9 momentum, 0.0005 weight decay and batch size 8. We trained the model 500k iterations and dropped the learning rate by 0.1 in the 200kth, 400kth iterations. During training, we used data augmentation including horizontal flipping, color distortion, random expansion and cropping. To compare the effect of the augmentation, we also trained models that only applied flipping. To train the Faster R-CNN model, we used the same optimizer and learning rate as in the SSD and batch size 1.

Prediction In the case of a one-stage model, to see the overall distribution of the predicted bounding boxes, NMS threshold was set to 1.0. In the case of a two-stage model with two NMS layers, the RPN-NMS and the head-NMS thresholds were set to 0.7 and 1.0 respectively, because the high threshold value of RPN-NMS can cause a huge number of box proposals. The score threshold was set to 0.01.

Attack Model	Attack Method			SSD		FR	
	Aug	BT	SR	Acc	AR	Acc	AR
XGB				66.09	67.64	60.47	60.48
shallow		(O)		62.62	62.66	58.72	58.67
AlexNet		(O)		64.28	64.26	62.74	62.62
AlexNet	✓	(O)		67.55	67.55	64.30	64.22
AlexNet	✓	(O)	✓	68.30	68.24	66.59	66.49
AlexNet	✓	(U)		69.34	69.31	66.69	66.59
AlexNet	✓	(U)	✓	**71.07**	**71.02**	**67.42**	**67.34**

Table 4.1: Comparison of various attack methods. FR and XGB denote Faster R-CNN and XG-Boost. Aug, BT, SR and AR denote augmentation, box style of canvas method, score rescaling and average recall, respectively. (O) and (U) denote original and uniform box size, respectively.

4.5.2 Attack Model Setup

To perform a black-box membership inference attack, we built several attack models as presented above. For XG-Boost model, we used Python XG-Boost package[1]. XG-Boost classifier takes vectorized bounding boxes and scores as inputs and has 5 maximum depth of a tree and 450 estimators as model parameters. For CNN-based classifiers with canvas method, we built two CNN models, a simple shallow CNN model and AlexNet [56]. For Shallow CNN model, we used two convolutional networks having 64 and 128 channels and two fully connected networks having 128 and 2 units. CNN based attack model takes drawn canvas images with predicted boxes as input. For balanced training, the attack model uses the almost same number of predicted results of "in" data and "out" data of the shadow model. We applied vertical and horizontal flipping for augmentation and score rescaling presented above. We compared various canvas methods to find the most optimal attack model.

[1]https://github.com/dmlc/xgboost

Model	Dataset	iters	Target Model		Attack Model		
			test mAP	train mAP	Attack Acc	Ave. Recall	Val Acc
SSD300-VGG16(LA.)	VOC	400k	59.27	92.27	89.92	89.90	91.16
SSD300-VGG16	VOC	250k	73.88	89.30	67.88	67.92	72.20
SSD300-VGG16	VOC	500k	74.25	90.27	71.07	71.02	72.20
SSD512-VGG16	VOC	500k	76.53	91.09	71.03	70.10	73.04
SSD300-Res50	VOC	700k	66.04	85.43	73.86	73.82	75.97
Faster R-CNN	VOC	200k	72.71	88.00	62.50	62.44	64.44
Faster R-CNN	VOC	400k	71.80	90.20	67.42	67.34	64.44
SSD300-VGG16	INRIA	100k	88.20	90.90	71.40	62.95	73.21
SSD300-VGG16	SynthText	400k	88.45	90.84	66.90	66.90	68.49

Table 4.2: Attack performance on various models and datasets. LA. refers little augmentation which indicates training with only horizontal flipping. Attack Acc and Ave. recall refer attack accuracy and average recall on target models. Val Acc refers attack accuracy on shadow models.

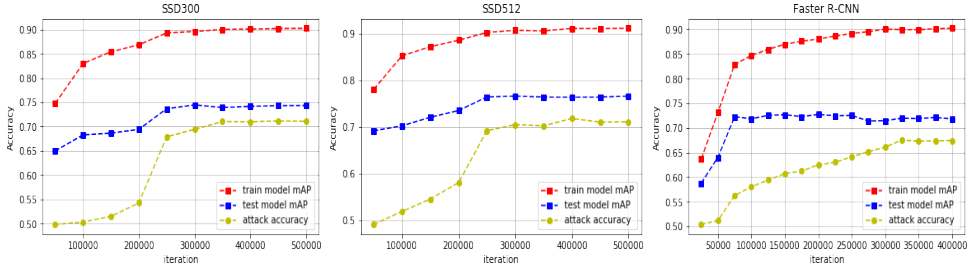


Figure 4.4: Membership inference attack results on various target models.

4.5.3 Experiment Results

Table 4.1 depicts the results of the comparisons of various attack methods. In general, AlexNet with augmentation, score rescaling and the uniform canvas method is successful on both the SSD and Faster R-CNN models. Therefore, we adopted the best performing method as the attack method in the next experiments.

To demonstrate the relationship between the membership inference and overfitting, we conducted experiments using different numbers of iterations in the model. Figure 4.4 shows that the overall attack performance increases with an increases

Model	SSD300	SSD512	FR
SSD300	74.25	68.84	61.73
SSD512	66.87	71.03	62.94
FR	60.19	57.28	67.42

Dataset	VOC	INRIA	SynthText
VOC	74.25	68.36	48.72
INRIA	74.28	71.40	50.85
SynthText	53.92	51.91	66.90

Table 4.3: Results of transfer attack over various object detection models and datasets. The x-axis represents the structure and dataset of the target models attacked and the y-axis represents that of shadow models for transfer attacks respectively. FR denotes Faster R-CNN.

in the number of iterations.

Table 4.2 shows the results of the membership inference attacks of various object detection models and datasets. The attack model is the best performing model in table 4.1. The mAP scores of the detection models are slightly smaller than their original performance because they train only half of the dataset. The evaluation metrics for the attack model are the accuracy and average recall of "in" and "out" labels. The attack model achieves a similar attack performance against the target and shadow models because the distributions of dataset and model structure are similar. Overall, the attack models achieve a high accuracy for most detection of the models and datasets. In general, large generalized errors are related to the high performance of the attack models. In the case of target models trained using the INRIA and SynthText datasets, test mAP is relatively high because the tasks are easy, although the attack models still obtain a high attack accuracy.

Defense	test mAP	train mAP	A-acc.	p-loss
Base	74.25	90.27	71.07	∞
Dropout	74.20	89.84	70.94	∞
DP(σ=10^{-4})	74.32	88.15	68.68	$2.42\times\ 10^{10}$
DP(σ=10^{-3})	67.30	78.45	50.45	$3.87\times\ 10^{8}$

Table 4.4: Comparison of various defense methods. A-acc and p-loss denote the attack model accuracy and privacy loss respectively.

4.5.4 Transfer Attacks

Setup During the transfer attack, we used the same setup as mentioned in Section 4.5.3. We conducted a transfer attack over the SSD300, SSD512 and Faster R-CNN models and VOC dataset. We also conducted transfer attacks over VOC, INRIA, and SynthText datasets and SSD300 model.

Results Tables 4.3 list the results of the model and dataset transfer attacks. The attack model trained using the same model structure or distribution dataset showed the highest accuracy. Transfer attacks on different detection models seemed to work well. The usage of the VOC dataset to attack the INRIA dataset and vice versa achieved a good performance. This might be because these two datasets have the same common label("person") and had a few objects per image. However, a transfer attack between SynthText and the other datasets did not perform well. This could be because SynthText had little in common with VOC and INRIA and had many objects per image. Transfer attacks tend to be successful when the datasets or models are similar to each other.

4.5.5 Defense

Setup We tested the proposed defense methods against membership attacks. For the dropout, we added two dropout layers with a ratio of 0.5 before the two layers of the model. For the differentially private algorithm, we set noise scale σ=10^{-3} , 10^{-4}, gradient bound $C = 50$, and minibatch size 2. We set up a relatively small noise because the object detection model has a large number of parameters [75].We trained the SSD300 model 800k iterations for σ=10^{-3}, and 500k iterations for the others. We obtained the privacy loss with fixed δ=10^{-5}.

Results Table 4.4 shows the results of the defense methods. Dropout shows a slight drop in the attack accuracy, but it does not show a large difference. The DP(σ=10^{-4}) shows little difference from the original model with mAP, but it lowers attack accuracy meaningfully. The larger noise scale DP(σ=10^{-3}) shows some loss in accuracy, but its good defense against the attack model compensates for this.

4.6 Conclusion

In this study, we introduced new membership inference attacks against object detection models. Our proposed CNN-based attack model using the canvas method performed better than a traditional machine learning regression method. We showed that sufficiently overfitted object detection models are vulnerable to privacy leakage. A generalization error is not a guarantee of safety against an inference attack. Transfer attacks are also efficient when the models or datasets are similar. To mitigate the privacy risks, we proposed defense mechanisms that are able to reduce such risks. We showed that membership inference risks in object detection models need to be considered.

Chapter 5

Single Image Deraining

5.1 Introduction

Adverse weather conditions such as rain, haze, and snow can produce complex visual effects on natural images and videos. In particular, rain streaks, which is one of the most commonly occurring phenomena in outdoor imaging, can potentially degrade the performance in several computer vision applications. Therefore, it is imperative to develop algorithms that effectively remove rain streaks and restore pristine background scenes in vision-related tasks.

Over the past few decades, several research works have studied the removal of rain streaks from captured images. Several traditional deraining methods have suggested separating rain streaks from the clean background image based on the physical characteristics or texture appearance patterns of the rain streaks. Recently, convolutional neural network (CNN)-based methods have achieved great success in solving this problem [51, 61, 63, 88, 103, 106, 116, 117, 121, 129].

Many of the CNN-based methods utilize encoder-decoder structures, and for the

most part, they add subnetworks without fully utilizing the information generated during the encoding-decoding process. For example, to remove fine-grained rain streaks and recover rain-free backgrounds more clearly, Yu *et al.* [121] consider the encoder-decoder as a coarse deraining stage and use an additional simple network as a fine deraining stage. Wang *et al.* [103] add a residual learning branch parallel to the encoder part to form a better conditional embedding and eventually generate a much better deraining result in the decoder part. Adding these subnetworks can easily improve performance, but there is a limitation that a model becomes heavier without leveraging enough information of an original model.

There is also an effort to utilize the information that is generated within the model. In order to obtain and leverage information from other pixels for the degraded background pixels, Li *et al.* [61] and Yu *et al.* [121] exploit non-local operations. These models use a square grid with the same aspect ratio in non-local operations. However, the operations with the square grid lack an understanding of the unique properties of the rain streaks because of their vertical distribution in the rainy image, which we explore (see Figure 5.3). Consequently, these methods have difficulties in recovering details in extremely adverse weather conditions.

To address these limitations of the prior works, we present a multi-level connection and wide regional non-local block network (MCW-Net) to carefully remove rain streaks and recover background details efficiently leveraging information generated during the encoding-decoding process. The proposed MCW-Net is based on an encoder-decoder structure consisting of down-sampling and up-sampling components as depicted in Figure 5.1.

We construct multi-level connection (MLC) between multiple-scale features to efficiently utilize information across various scales without additional subnetworks

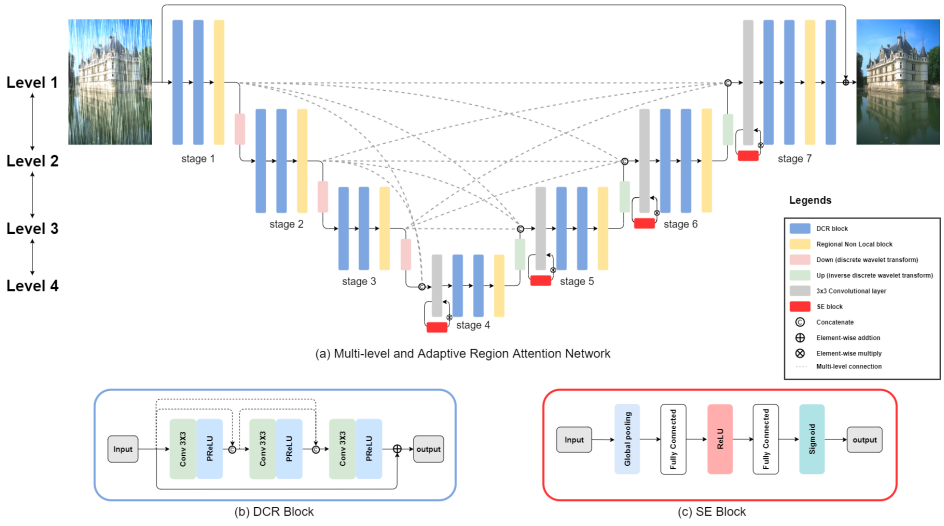


Figure 5.1: Overview of the proposed MCW-Net structure.

in the recovery of the background. We implement an interactive multi-connection that considers the interconnections between different scales. Because the features at multiple levels show different scale characteristics, direct connections rather cause adverse effects in the model. To adaptively rescale the channel-wise features in MLC, we apply a channel-wise attention layer [47] after MLC, which helps the network to focus on the useful channels. We demonstrate the importance of the channel-wise attention, and we validate that MLC plays an effective role by comparing the qualitative and quantitative results of models with and without MLC in Section 5.4.4.

In addition, we implement a non-local operation [107] to capture long-range spatial dependencies between distant pixels. We propose a wide regional non-local block (WRNL), which divides feature maps into grids of wide regions (see Figure 5.2) before performing the region-wise non-local operation. This wide grid provides a relatively more even distribution of the rain streaks by region, which facilitates

the retrieval of rich long-range background information during the recovery of the original rain-free image (see Section. 5.3.2).

Additionally, as described in [116], to prevent information loss during the sampling operation, we adopt the discrete wavelet transform (DWT) and inverse DWT (IWT) in place of the simple pooling and de-convolution operations. Unlike the pooling operation, the DWT operation is invertible via IWT, which helps to avoid information loss. In addition, rain streaks can be captured with rich frequency information via wavelet transform.

We evaluate the proposed MCW-Net on various synthetic and real-world deraining datasets and compare its performance with existing state-of-the-art methods. In particular, for real-world images, we measure the performance of the proposed method using B-FEN [112] metric dedicated to deraining quality measurement. We conduct an experiment on raindrop data, another degradation phenomenon caused by rain from the perspective of the generalization ability of the model. In addition, we validate in RainCityscape experiments that the proposed method can also help with other vision tasks such as semantic segmentation.

In summary, the contributions of this work may be summarized as follows.

1) We propose MLC to fully leverage information generated in encoding-decoding process for detail recovery without additional subnetworks. Feature information of all the scales in the down-sampling part is aggregated at each stage of the up-sampling part of the network, so it helps to recover details by preventing information loss that occurs during the sampling process. We also analyze that channel-wise attention plays an key role in the MLC.

2) We propose the WRNL, which effectively restores the background by using sufficient rain-free information in each region of widely divided grids in the input

feature maps. We experimentally demonstrated that the distribution of even rain streaks by grid helps the deraining performance.

3) We perform experiments on both synthetic and real-world rain datasets and demonstrate that the proposed method significantly outperforms existing state-of-the-art methods. We also demonstrate the excellence of the proposed method for real-world images using B-FEN, a metric dedicated to measuring deraining quality.

4) We construct joint image deraining and semantic segmentation models on the RainCityscape dataset. In addition to conventional comparisons such as the peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM), we comprehensively evaluate the contribution of the deraining model to other vision tasks.

5.2 Related Work

The single image deraining problem begins with the assumption that a rainy image consists of a background layer and a rainy layer. Several traditional training methods based on single images and videos have been proposed. Barnum *et al.* [4] reconstruct rainy images by combining the appearance model with the streak model. The appearance model identifies individual rain streaks and the streak model utilizes the statistical characteristics of rain. Chen and Hsu [12] use the low-rank model to separate the layers in a rainy image. As noted by Yang *et al.* [118], sparse coding is applied during this process to separate the rainy layer from the rainy image [20, 52, 72, 108, 135]. Further, Li *et al.* [5, 64] approach this problem using the Gaussian mixture model.

Because of the remarkable performance exhibited by deep learning-based meth-

ods, especially CNN-based ones, the potential use of deep learning in deraining has been extensively researched. Yang *et al.* [117] apply a CNN-based method for the first time and express natural images by adding atmospheric light as a component to rainy images. Fu *et al.* [29] and Fan *et al.* [28] use a single primary network that restores input images using the residual network. Based on the residual network, Li *et al.* [63] attempt to further eliminate overlapping rain streaks by organizing the context aggregate network into multiple stages. Shen *et al.* [90] consider rain streaks to be high-frequency and attempt to remove rain streaks by utilizing DWT. Yang *et al.* [116] divide the deraining process into several stages and reconstruct the image recurrently, beginning with a small portion of the image to eventually obtain the entire image.

Wang *et al.* [106] capture the spatial contextual information using a four-directional recurrent neural network with the identity matrix initialization model. Ren *et al.* [88] propose progressive ResNet to effectively remove the rain via recursive computation. Yu *et al.* [121] propose GraNet, which is designed to identify rain masks in the coarse stage using a region-aware non-local block. Subsequently, the process uses the rain masks to create the final image using another reconstruction network. To achieve pixel-wise deraining in image recovery, encoder-decoder structures have been used in certain methods. Wang *et al.* [103] propose the residual learning branch as a component of the encoder. Li *et al.* [61] enhance the performance by introducing non-local blocks into the encoder-decoder network. Among the methods that reconstruct the rainy layer to be identical to the background layer, the generative adversarial network is widely used to remove raindrops and rain streaks [62, 86, 129].

Yang *et al.* [119] propose the fractal band learning network based on frequent

band recovery. Wang *et al.* [104] propose an interpretable deep network based on a convolutional dictionary network. Jiang *et al.* [51] use the images of various sizes as the input to the model. A multi-scale pyramid structure is used to promote cooperative representation. Deng *et al.* [21] propose two-branch parallel networks, in which one branch performs rain removal and the other branch detail recovery. In [109], newly formulated rain streaks transmission maps, vapor transmission maps, and atmospheric lights are respectively learned by three different networks. Zhang *et al.* [130] propose a paired rain removal network, which exploits both stereo images and semantic information. Zamir *et al.* [126] propose a multi-stage progressive architecture with a supervised attention module for image restoration.

Chen *et al* [7] present an image processing transformer (IPT). IPT covers different several tasks such as super-resolution, denoising, and deraining based on the transformer method. The authors augment ImageNet images to low-resolution, noised, and rainy images via corresponding filters and then pre-train the IPT with each set. Yue *et al* [123] propose a dynamic rain generator to mimic the rain streaks in the video. The rain streaks in generated videos are removed by a deep learning-based model called derainer.

Zhang *et al* [132] exploit the low to high-level features and attention operation to restore the hazed images. Their intuition is that the low-level features contribute to recovering finer details and the high-level features represent the shape of the object or abstract semantic information. The utilization of hierarchical features and the attention mechanism are similar to one of our strategies, multi-level connection. However, their work fuses the lower-level features only in the most-down-sampled features, whereas we consider all the features captured in the down-sampling phase on every up-sampling phase.

5.3 Proposed Network

In this section, we describe the architecture of the proposed MCW-Net, which is is based on a U-Net-like structure whose overview is depicted in Figure 5.1. As is apparent from the figure, we divide the levels according to the size of the feature map and define a set of blocks as a stage.

The proposed MCW-Net consists of an encoder part and a decoder part. The first three stages form the encoder part, and the remaining four stages the decoder part. We propose MLC, which connect all outputs of the encoder to all inputs of the decoder. MLC enables more diverse scale features to be used during the restoration process. Each stage of MCW-Net is composed of two densely connected residual (DCR) blocks, each of which consists of three convolution layers followed by PReLU [100] (refer Figure 5.1(b)) and one WRNL block. To adaptively rescale channel-wise features after concatenating the multi-level features, a squeeze-and-excitation (SE) block is added in front of each decoder stage. A 1×1 convolutional layer follows the SE block to adjust the number of channels.

5.3.1 Multi-Level Connection

In the usual U-Net-like network, connections exist only between features corresponding to the same level. Such a structure cannot make use of multiple scale information during the recovery of low-level features in the decoder. However, single image deraining is a low-level vision task that requires richer range scale features to restore the details in the image. Inspired by [96, 98, 105], we formulate MLC to aggregate the features of all the levels. At each stage of the up-sampling part of the network, features from all scales in the down-sampling part is aggregated. These

multi-scale features provide a wider range of information from simple patterns (*e.g.*, corners or edge/color conjunctions) in its lower-level to more complex high-level features (*e.g.*, significant variation and object-specific features). They encourage more delicate deraining because rainy pixels in the image are recovered referencing semantic context and details from other intact pixels. However, simply fusing various features might cause necessary information weighted insufficiently in conjunction with more weight on less helpful information at the current up-sampling stage. The attention mechanism allows the model to focus more on significant channels among several channels, and for this reason, it is essential when connecting features of multiple levels.

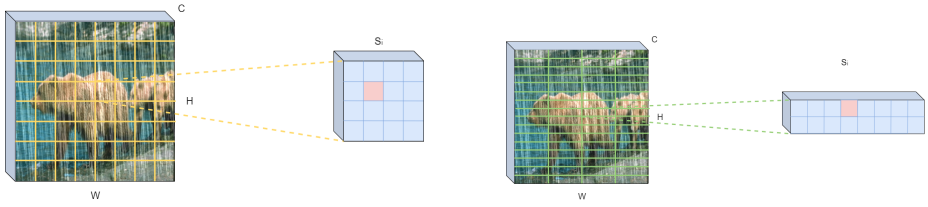
Formally, let E^l_{out} be the output features at level l ($l = 1, 2, 3$) in the encoder part. At each level l ($l = 1, 2, 3, 4$) in the decoder part, the input feature D^l_{in} is given as:

$$D^l_{concat} = (\bigoplus_{i=1}^{3} H^l_i(E^i_{out})) \oplus H_{up}(D^{l+1}_{out}) \tag{5.1}$$

$$D^l_{in} = W_{1\times1}(f_{SE}(D^l_{concat})) \tag{5.2}$$

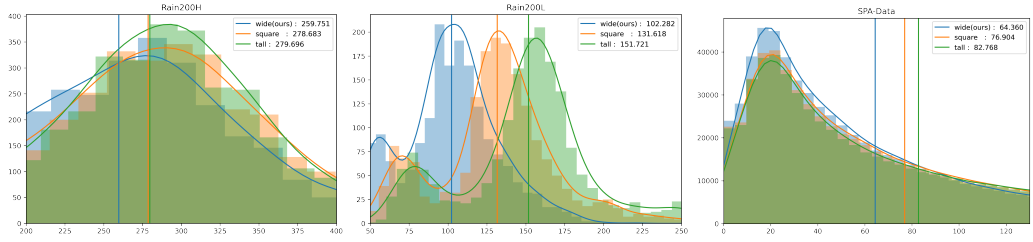
where $\oplus$ denotes the concatenation operation, $H_{up}(\cdot)$ denotes the up-sampling operation, D^l_{out} denotes the output feature of the decoder part at level l, $W_{1\times1}$ denotes the 1×1 convolution layer, and $f_{SE}(\cdot)$ denotes the SE block discussed above. $H^l_i(\cdot)$ denotes the sampling operation from level i to l. In other words, H^l_i is the down-sampling by $l - i$ times, identity, and up-sampling by $i - l$ times operations if $l > i$, $l = i$, and $l < i$, respectively. We set $D^5_{in} = 0$ for convenience.

Without MLC, high-level features cannot be used during the processing of low-level features and vice versa. This approach helps the network to exploit various



(a) (b)

Figure 5.2: Examples of patches of the input feature of the regional non-local block. (a) Square patch, (b) Wide rectangular patch. Every pixel in a patch refers to every pixel in the patch.



(a) Rain200H [117] (b) Rain200L [117] (c) SPA-data [106]

Figure 5.3: Analysis of rain streak distributions in various region types. The x-axis represents the standard deviation between the number of rain pixels in the patches in each image. The y-axis represents the number of images and vertical bars are the means of each dataset standard deviation. The distribution of the images according to the standard deviation is represented by histograms. We approximate the probability density function of the histogram by using kernel density estimation. As can be seen in the figures, the wide region has the smallest standard deviation mean on all the datasets, so it can be interpreted that each patch of the wide region has the evenest background information.

scale representations in recovering large-scale features. To find the correct correspondence between the feature shapes at different scales, we apply discrete wavelet transforms (DWT or IWT), as described in Section 5.3.3, for the down-sampling and up-sampling operations.

5.3.2 Wide Regional Non-Local Block

We denote the input feature to the WRNL as $X \in \mathbb{R}^{H \times W \times C}$. We divide X into a $a \times b$ grid of patches $\{X^k\}, (k = 1, ..., K = ab)$ where K is the number of patches. The grid division is illustrated in Figure 5.2. The linear embedding processes for X^k to generate the output Z^k are formulated as follows.

$$\Phi(X^k)_i^j = \phi(X_i^k, X_j^k) = \exp\{\theta(X_i^k)\psi(X_j^k)^T\} \tag{5.3}$$

$$\theta(X_i^k) = X_i^k W_\theta, \psi(X_i^k) = X_i^k W_\psi, G(X)_i^k = X_i^k W_g \tag{5.4}$$

where X_i^k denotes the feature X^k at position $i = 1, ..., HW/ab$. The learnable weight matrices W_θ, W_ϕ, and W_g have the dimensions of $C \times L$, $C \times L$, and $C \times C$, respectively. In practice, $L = C/2$ is used. The regional non-local operation can be expressed as follows:

$$Z_i^k = \frac{1}{\delta_i(X^k)} \sum\nolimits_{j \in S_i} \Phi(X^k)_i^j \, G(X^k)_i \,, \quad \forall i \,, \tag{5.5}$$

where $\delta_i(X^k) = \sum_{j \in S_i} \phi(X_i^k, X_j^k)$ denotes the correlation between X_i^k and each X_j^k in S_i, and Z_i^k denotes the output feature Z^k at position i. S_i denotes a set of patch positions. If $a > b$, then the patch is wider than when $a = b$. Therefore, we call the patch a wide rectangular patch, a square patch, and a tall rectangular patch if $a > b$, $a = b$, and $a < b$, respectively. In the WRNL block, we set the $a \times b$ grids to 16×4 , 8×2, 4×1, and 4×1 at levels 1, 2, 3, and 4, respectively.

Analysis

Given that the non-local block recovers a specific pixel based on the information of other pixels in the patch, it is necessary to have sufficient background information in each patch. The regional non-local block uses the background information sufficiently if the rain streaks are evenly distributed between the patches. However, we observe that the rain streaks are not evenly distributed between square patches in the images used in the previous deraining research [61, 121]. Because of the predominantly vertical distribution of rain steaks, we expect that wide rectangular patches have a more even distribution of the streaks than square and tall rectangular patches.

The distribution of the rain streaks is confirmed through experiments. Wide rectangular, square, and tall rectangular patches are prepared by dividing the height and width of the image into 16×4, 8×8 and 4×16 grids respectively. It should be noted that (a) in Figure 5.2 contains an 8×8 grid of patches, and (b) in Figure 5.2 contains 16×4 grid of patches. We consider pixels as rain streaks if the difference between the pixels in x_{input} and x_{gt} exceeds a certain threshold. The standard deviation between the number of rain pixels in the patches included in each image is depicted in Figure 5.3. Wide rectangular patches are observed to exhibit smaller standard deviation values compared to square and tall rectangular patches, which implies an even distribution of rain across all patches. This results in the effective recovery of the image because the usable background information within each patch is also distributed evenly as shown in Table 5.7.

5.3.3 Discrete Wavelet Transform

To prevent information loss, we adopt the discrete wavelet transform for the sampling operation. In particular, We use 2D Haar wavelet which is widely used in image processing.

The proposed network uses DWT and IWT for down-sampling and up-sampling, respectively. In particular, we adopt the Haar transform, which is simple and widely used method in image processing [35, 68, 85, 90, 116]. The Haar transform is calculated based on the filter $\mathbf{f}_{LL}$, $\mathbf{f}_{LH}$, $\mathbf{f}_{HL}$ and $\mathbf{f}_{HH}$ as follows:

$$\mathbf{f}_{LL} = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \mathbf{f}_{LH} = \frac{1}{4}\begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}, \mathbf{f}_{HL} = \frac{1}{4}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, \mathbf{f}_{HH} = \frac{1}{4}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \quad (5.6)$$

Given that $\mathbf{f}_{LL}$ is identical to average pooling, LL achieves local translation invariance by reducing the size of the feature map (Equation 5.6). LH, HL, and HH contain edge information. In particular, as LH contains vertical edge information, the features of the rain streaks can be effectively obtained from it. The IWT operation during the up-sampling process is the inverse operation of the DWT.

5.3.4 Loss Function

We define the loss function L as follows.

$$\mathcal{L} = \|x_{gt} - f(x_{input})\|_1 + \|x_{gt} - f(x_{input})\|_2 \quad (5.7)$$

where x_{input} denotes the input rainy image, x_{gt} denotes the corresponding rain-free image, and f denotes the return of the MCW-Net output with respect to x_{input}. We use L1+L2 loss because it shows the slightly better performance, but

our method does not appear to be sensitive to the loss.

5.4 Experiments

In this section, we present the dataset used in this study and describe the details of the experimental setting. We present two versions of the proposed MCW-Net: a small model and a large model. The architecture of the two models is same except for the number of channels. The small model has eight times fewer channels than the large model. We conduct a quantitative and qualitative evaluation of the proposed method and compare its performance with state-of-the-art methods. An ablation study is conducted to confirm the significance of each component introduced in Section 5.3.

5.4.1 Datasets and Evaluation Metrics

Table 5.1: Synthetic and real-world datasets

Datasets	Train	Test	Type
Rain200L [117]	1,800	200	synthetic
Rain200H [117]	1,800	200	synthetic
Rain800 [129]	700	100	synthetic
Rain1200 [128]	12,000	1,200	synthetic
RainCityscapes [16, 48]	9,432	1,188	synthetic
SPA-Data [106]	640k	1,000	real-world
Yang *et al.* [117]	-	15	real-world
DQA [112]	-	206	real-world
Raindrop [86]	861	58 (A)/249 (B)	real-world

Five synthetic datasets (Rain200H, Rain200L, Rain800, Rain1200, RainCityscapes) and three real-world datasets (SPA-Data, Yang *et al.* [117], DQA [112]) are used

to evaluate the performance of the proposed method. As pointed out by Ren *et al.* [88], certain overlaps of background exist between the training and test datasets in the Rain100H and Rain100L datasets. Therefore, we evaluate our model using the updated Rain200H and Rain200L datasets, which do not share backgrounds with the corresponding training datasets. Because the absence of ground truth data makes quantitative evaluation impossible, the real-world dataset of [117] is evaluated qualitatively using the Rain200H-trained weights. In addition, Raindrop [86] dataset is used to evaluate the raindrop removal performance of the proposed method. We compare the performance of the proposed method with nine state-of-the-art single-image deraining methods.

We employ PSNR and SSIM [110] metrics for quantitative quality assessment. All the PSNRs reported in the following experimental results are calculated for RGB channels. Some previous works filter the derained RGB images into YCbCr space and then evaluate PSNR only for the Y channel to focus on the luminance. However, because most of the high-level vision algorithms commonly receive the RGB image as an input, we consider evaluation of well-recovered rainy image in RGB space is more appropriate and helpful for other vision tasks. Additionally, we employ the dedicated B-FEN metric [112] to measure the deraining quality of deraining algorithms.

5.4.2 Datasets and Experiment Details

For all the datasets, we randomly crop 256×256 patch from each input image. During the training, we set the batch size to 4 and use the Adam optimizer. For the large model, we set the learning rate to be 10^{-4} and train our model for 200 epochs on the Rain200H, Rain200L, and Rain800 datasets, 100 epochs on the

Table 5.2: Average PSNR and SSIM comparison on the synthetic datasets Rain200H [117], Rain200L [117], Rain800 [129], Rain1200 [128], and real-world dataset SPA-Data [106]. The highest values are indicated in red and the second-highest values are indicated in blue. Note that IPT uses rainy-augmented ImageNet pre-trained weight, but proposed methods and other comparable models do not. So we manually train the IPT from sketch only with the provided dataset.

Method	JORDER [117] (CVPR' 2017)	RESCAN [63] (ECCV' 2018)	SPANet [106] (CVPR' 2019)	PReNet [88] (CVPR' 2019)	ReHEN [118] (MM' 2019)	RCDNet [104] (CVPR' 2020)
Rain200L	36.95/0.979	36.94/0.980	35.60/0.974	36.28/0.979	38.57/0.983	35.28/0.971
Rain200H	22.05/0.727	26.62/0.841	26.32/0.858	27.64/0.884	27.48/0.863	26.18/0.835
Rain800	22.24/0.776	24.09/0.841	24.37/0.861	22.83/0.790	26.96/0.854	24.59/0.821
Rain1200	24.32/0.862	32.48/0.910	32.38/0.920	30.40/0.891	32.64/0.914	32.23/0.910
SPA-Data	35.72/0.978	36.99/0.967	38.53/0.987	35.68/0.942	38.65/0.974	41.47/0.983
Params	4,169,024	499,668	283,716	168,963	298,263	3,166,355
Method	DRD-Net [21] (CVPR' 2020)	MPRNet [126] (CVPR' 2021)	IPT [7] (CVPR' 2021)	IPT [7] (w/ pretraining)	MCW-Net (Ours-small)	MCW-Net (Ours-large)
Rain200L	37.15/0.987	37.87/0.983	37.08/0.980	40.32/0.989	39.19/0.986	39.92/0.988
Rain200H	28.16/0.920	27.63/0.872	27.03/0.955	-	29.31/0.901	30.70/0.922
Rain800	26.32/0.902	25.93/0.832	25.64/0.833	-	28.39/0.876	28.42/0.876
Rain1200	-	32.91/0.916	20.12/0.691	-	33.17/0.922	33.70/0.928
SPA-Data	-	44.89/0.989	17.75/0.515	-	42.81/0.986	46.88/0.991
Params	5,230,214	3,637,303	115,333,723	115,333,723	2,158,586	129,539,018

Rain1200 and the RainCityscapes datasets, 3 epochs on the SPA-Data dataset, and 500 epochs on the Raindrop dataset. For the small model, we set the learning rate to be 5×10^{-4} and train our model for 500 epochs on the Rain200H, Rain200L, and Rain800 datasets, 100 epochs on the Rain1200 and the RainCityscapes datasets, 5 epochs on the SPA-Data dataset, and 750 epochs on the Raindrop dataset.

5.4.3 Evaluations

Results on Synthetic Datasets

As mentioned in Section 5.4, the proposed MCW-Net is evaluated on four synthetic datasets, and the performance is compared to eight state-of-the-art methods. The quantitative results on the synthetic datasets are presented in Table 5.2.

For other models to be compared, if a metric is not provided in the original paper, we train the models with their default settings and report the results to the comparison table. Otherwise, we report the better result between the provided metric in the original paper and the result obtained by our re-trained model. If reproduction is not possible, we directly copy the provided result to the comparison table.

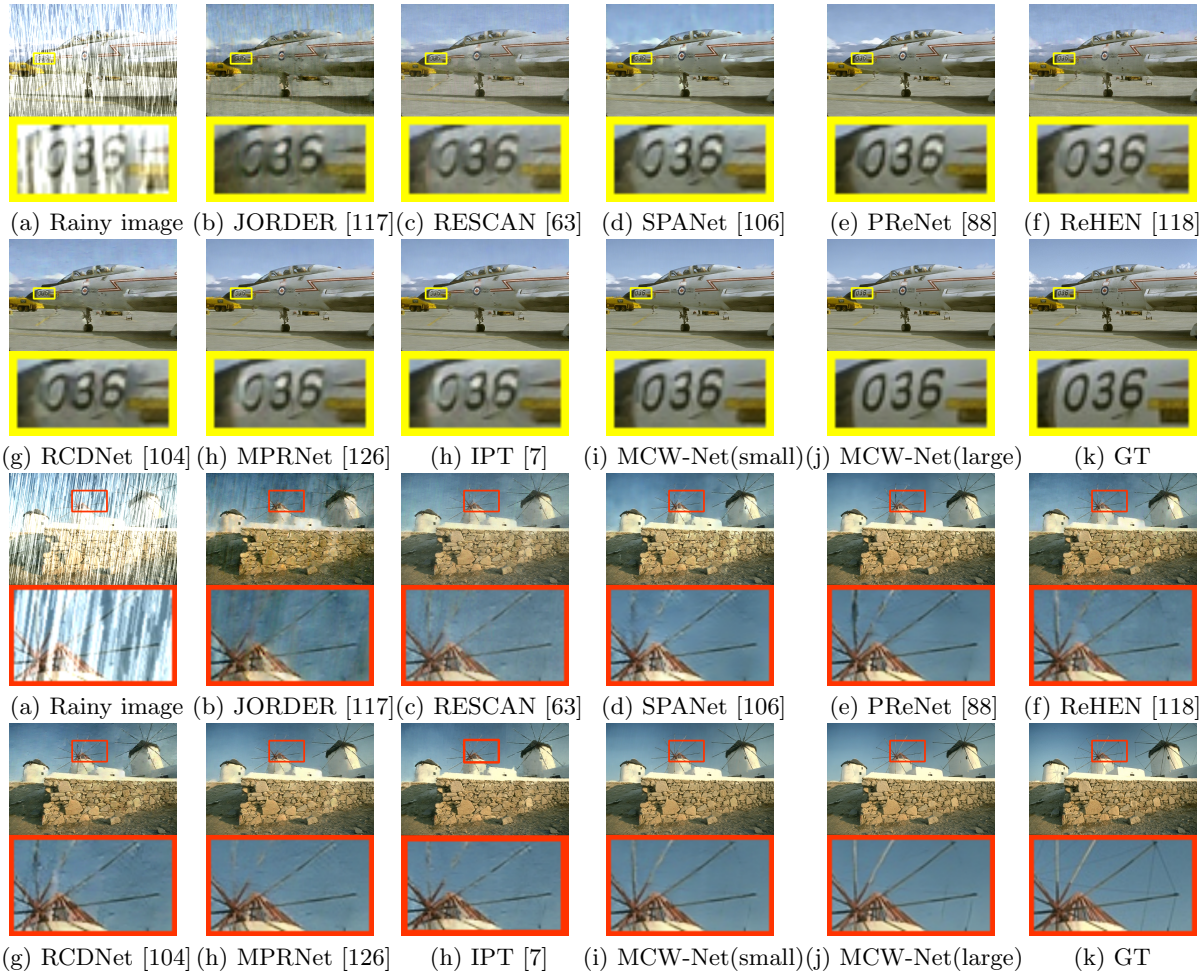
As is evident from the data, the proposed MCW-Net (large) achieves remarkable improvement over existing state-of-the-art methods with respect to the PSNR and SSIM metrics across all synthetic datasets, and MCW-Net (small) follows right behind.

The original inputs, the ground truth, and the qualitative results for Rain200H are shown in Figure 5.4. As shown in the yellow boxes of Figure 5.4, MCW-Net (small) clearly restores the number compared to other methods, and MCW-Net (large) restores the digits surprisingly similar to ground truth. In the red boxes of Figure 5.4, MCW-Net (small) restores the sky and spokes of the windmill cleanly compared to other methods but failed to recover lines, while MCW-Net (large) restores some of the lines.

Results on Real-world Datasets

For further general verification of the proposed method, additional experiments are conducted on two real-world datasets. To estimate the performance of the other state-of-the-art models, we employ the same way as quantitative evaluation of synthetic data. On the SPA-Data, MCW-Net exhibits quantitatively outstanding performance compared to the other state-of-the-art methods.

To confirm the effectiveness of the method trained using synthetic rainy images



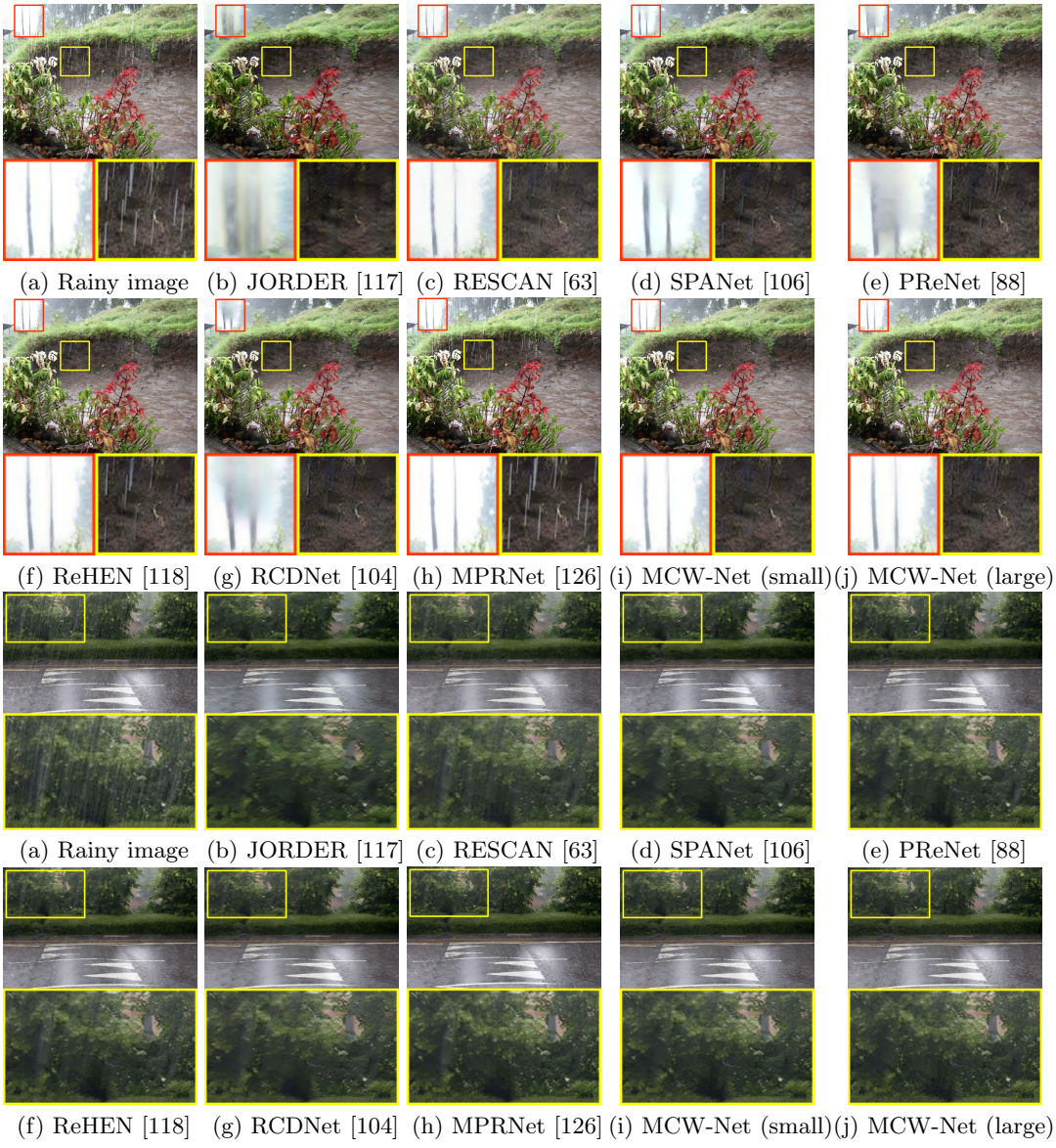
(a) Rainy image (b) JORDER [117] (c) RESCAN [63] (d) SPANet [106] (e) PReNet [88] (f) ReHEN [118]

(g) RCDNet [104] (h) MPRNet [126] (h) IPT [7] (i) MCW-Net(small) (j) MCW-Net(large) (k) GT

(a) Rainy image (b) JORDER [117] (c) RESCAN [63] (d) SPANet [106] (e) PReNet [88] (f) ReHEN [118]

(g) RCDNet [104] (h) MPRNet [126] (h) IPT [7] (i) MCW-Net(small) (j) MCW-Net(large) (k) GT

Figure 5.4: Results obtained via several state-of-the-art methods on the Rain200H [117] images. The outputs of MCW-Net exhibit no traces of rain streaks on both image samples. MCW-Net also recovers the most detailed images.



(a) Rainy image (b) JORDER [117] (c) RESCAN [63] (d) SPANet [106] (e) PReNet [88]

(f) ReHEN [118] (g) RCDNet [104] (h) MPRNet [126] (i) MCW-Net (small) (j) MCW-Net (large)

(a) Rainy image (b) JORDER [117] (c) RESCAN [63] (d) SPANet [106] (e) PReNet [88]

(f) ReHEN [118] (g) RCDNet [104] (h) MPRNet [126] (i) MCW-Net (small) (j) MCW-Net (large)

Figure 5.5: Results obtained via several state-of-the-art methods on the Yang *et al.* [117] images. Among state-of-the-art methods, MCW-Net is the only one that restore the detail of the images while removing the rain streaks.

in removing real rain streaks, qualitative experiments are conducted on images presented by Yang *et al.* [117]. To compare the model performances under fair conditions, only Rain200H is used during the training process. As shown in Figure 5.5, MCW-Net generates satisfactory results with respect to both the removal of the rain streaks and the restoration of the details in the background. The small and large versions of MCW-Net recover the details of the columns in the red box and remove the rain streaks in the yellow box better compared to other models. Although detail recovery and rain removal are the trade-off for other models, MCW-Net succeeds in both. MCW-Net also recovers the cleanest background for another image sample. The yellow box shows the output of MCW-Net exhibits no traces of rain streaks while they are left in the result of the other models.

Results on Authentic Rain Images with the Dedicated Metric

PSNR and SSIM estimate how the recovered output image closes to the target image. Therefore, several low-level vision tasks (*e.g*, denoising, super-resolution, deraining) conventionally exploit these metrics as an evaluation tool. Nonetheless, one might argue that PSNR and SSIM are general-purpose quality metrics so they are limited to concentrating only on the deraining ability. Besides, they need target images to be calculated and thus cannot be applied on target-absent authentic rainy images.

To handle this issue, we additionally evaluate the proposed method via a measurement called B-FEN [112], which accurately evaluates the deraining quality using a bi-directional feature embedding network. A higher B-FEN score represents better perceptual quality, which indicates that the model not only effectively removes rain streaks but also well preserves the original rain-free image. We train all the

Table 5.3: Comparison results of the various methods on DQA dataset in B-FEN [112] metric (higher is better).

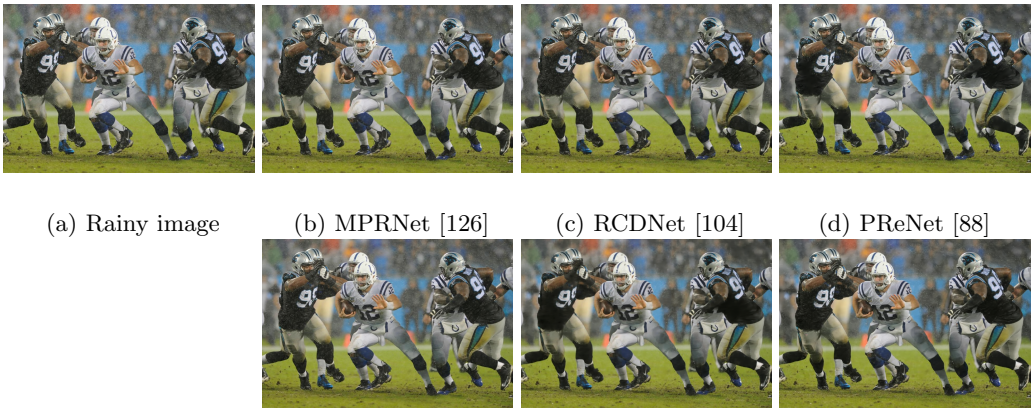
Methods	B-FEN
original	0.2997
MPRNet	0.3051
RCDNet	0.3101
PreNet	0.3139
SPANet	0.3154
MCW-Net (ours-small)	0.3287
MCW-Net (ours-large)	0.3222

comparable models and the proposed method on SPA-Data and evaluate the DQA dataset [112]. Since DQA is a real-world testing image set, real-world SPA-Data can guide the model to capture the properties of authentic rain streaks.

As shown in Table 5.3, our model achieves the highest B-FEN score. One thing to note is that the small version of MCW-Net has a higher B-FEN score than the large version of MCW-Net. B-FEN is a subjective opinion-aware metric, and opinion-making participants may tend to focus more on rain streaks removal than background restoration. From this point of view, it may lead to possible inconsistent results different from that of other objective opinion-unaware metrics.

Results on Raindrop data

Raindrops, which are a commonly observed phenomenon in conjunction with rain, also might degrade the performance in computer vision applications. Even though we design the proposed method to remove rain streaks in images, we explore the model's generalizability with the raindrop image dataset. The experimental results are reported in Table 5.4 and Figure 5.7. In the evaluation, we use the weight of the AGAN model provided by the author. We calculate PSNR and SSIM metrics



(a) Rainy image (b) MPRNet [126] (c) RCDNet [104] (d) PReNet [88]

(e) SPANet [106] (g) MCW-Net (small) (j) MCW-Net (large)

Figure 5.6: Results obtained via several state-of-the-art methods on the DQA images. Among state-of-the-art methods, MCW-Net is the only one that restore the detail of the images while removing the rain streaks. Based on the area of left-most person in (g) and (j), the small version removed rain better than the larger version, consistent with the results of the B-FEN score. However, the small version remove all the wrinkles on clothes, and the large version preserve them, so the large version achieves a better restoration of details.

Table 5.4: Average PSNR and SSIM comparision on Raindrop dataset.

Dataset	Testset A		Testset B	
	PSNR	SSIM	PSNR	SSIM
Eigen13 [25]	23.74	0.788	-	-
Pix2Pix [50]	28.15	0.855	-	-
PreNet [88]	28.58	0.913	-	-
AGAN [86]	30.55	0.910	24.43	0.795
MCW-Net (ours-small)	29.96	0.906	24.91	0.800
MCW-Net (ours-large)	30.77	0.918	25.17	0.809

(a) Rain drop image (b) AGAN [86] (c) MCW-Net (small) (d) MCW-Net (large) (e) GT

(a) Rain drop image (b) AGAN [86] (c) MCW-Net (small) (d) MCW-Net (large) (e) Clean

Figure 5.7: Results obtained via several state-of-the-art methods on the Raindrop images. Images in the first ans second rows are from testset A and testset B, respectively.

in RGB channels as in other experiments.

5.4.4 Ablation Study

We conduct an ablation study to demonstrate the significance of all the methods used in the MCW-Net architecture. MCW-Net (large) and Rain200H dataset are used for the ablation study. We conduct three experiments and report the average values. All the evaluations are dedicated to the proposed method without Cutmix. Because Cutmix is the data augmentation strategy and hence is not directly related

Table 5.5: Ablation study on the various strategies presented in Section 5.3.

WRNL	DWT	MLC	Cutmix	PSNR	SSIM
				28.12	0.906
✓				29.49	0.911
✓	✓			30.22	0.917
✓	✓	✓		30.62	0.921
✓	✓	✓	✓	30.70	0.922

to ablation about the model structure.

Ablation study on strategies employed

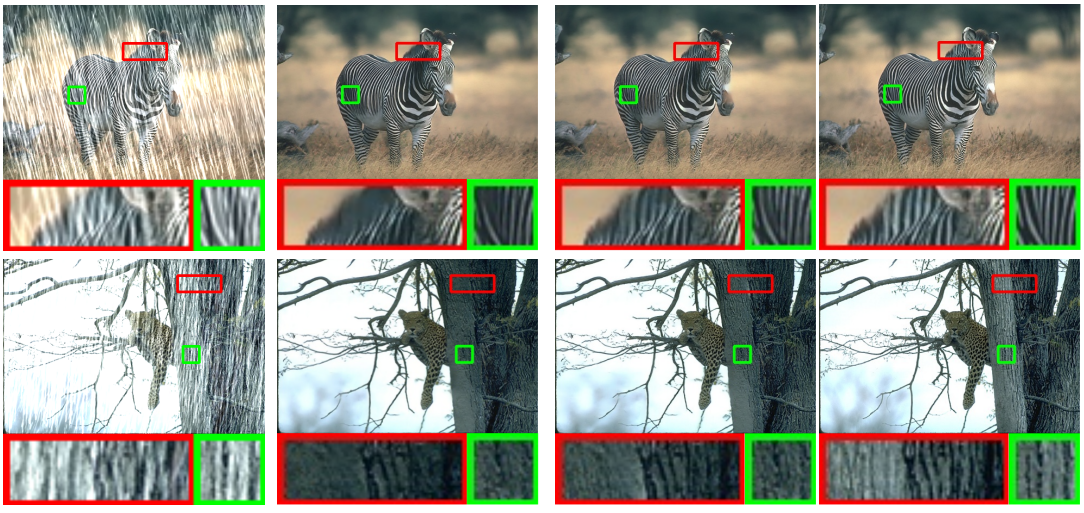
An ablation investigation is conducted to evaluate the performance of the proposed strategies. The baseline model is constructed with two DCR blocks corresponding to each stage and the 2×2 max pooling and pixel shuffle operation are adopted as the down-sampling and up-sampling operations, respectively. As evident from Table 5.5, each strategy contributes to the performance improvement.

Ablation study on MLC

To show the importance of channel-wise attention to MLC, we evaluate the performance using MLC with channel-wise attention and MLC with other commonly used fusing operations, addition and concatenation. As shown in Table 5.6, MLC with addition or concatenation rather degrade the performance. We analyze that this result occurs because additional information which is messy and unorganized rather interferes with the decoding process. Considering channel-wise attention serves as an indication of which information should be referenced more importantly at the current level decoding process, so MLC with channel-wise attention improves the performance of the model.

Table 5.6: Ablation study on MLC, where C.A. denotes channel-wise attention. The experiments are conducted on the proposed method without Cutmix.

	PSNR	SSIM
No MLC	30.22	0.917
MLC with concatenation	30.07	0.913
MLC with addition	30.26	0.916
MLC with C.A. (SE)	30.62	0.921



(a) Input (b) MCW-Net (w.o. MLC) (c) MCW-Net (w. MLC) (d) GT

Figure 5.8: Qualitative ablation study on MLC. MCW-Net (large) is used for the study. In the first row, we can see that the zebra pattern in the red and green box is not well restored without MLC. However, model with MLC restored the pattern in the red and green box well. In the second row, we can also see that the model with MLC better restored the tone and texture of the tree than the model without MLC. As a result, we can confirm that MLC plays a certain role in recovering detail as intended.

In addition, we conduct a qualitative ablation study to see if MLC actually helps to restore the details as intended, and the results are shown in Figure 5.8. The results confirm that MLC effectively does detail recovery as well as quantitative improvement. Details of the results are described in the caption of Figure 5.8.

Table 5.7: Ablation study on types of regional non-local blocks.

Dataset	Region Type	PSNR	SSIM
Rain200H	Tall Rectangle	30.08	0.916
	Square	30.14	0.915
	Wide Rectangle	30.62	0.921
Rain200L	Tall Rectangle	39.86	0.987
	Square	39.87	0.988
	Wide Rectangle	39.92	0.988
SPA-DATA	Tall Rectangle	42.78	0.987
	Square	42.96	0.987
	Wide Rectangle	43.05	0.987

Ablation study on non-local block region types

We evaluate the performance using square, tall, and wide-type regional non-local blocks on Rain200H, Rain200L, and SPA-DATA datasets. The results presented in Table 5.7 demonstrate that the wide-type regional non-local block achieves the best performance.

Ablation study on various sampling operations

To compare the performance of various sampling operations, we evaluate the performance using mean pooling, 1x1 convolution, and the discrete wavelet transform. The results presented in Table 5.8 demonstrate that the discrete wavelet transform has the best performance.

5.4.5 Applications for Other Tasks

We investigate the effect of the deraining model on improving the performance of high-level vision applications such as semantic segmentation. Because rain streaks

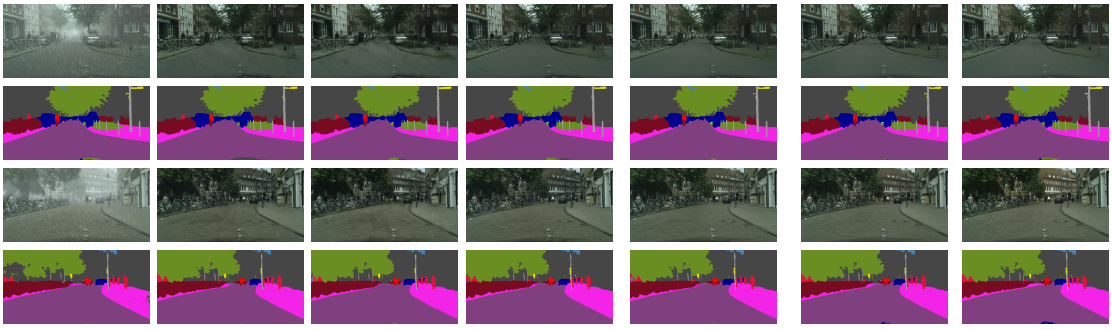
Table 5.8: Ablation study on various sampling methods. Note that three different sampling operations are compared on the proposed method without MLP and Cutmix.

Sampling Operation	PSNR	SSIM
Mean Pooling	29.50	0.909
1×1 conv.	29.80	0.911
DWT & IWT	30.62	0.921

Table 5.9: Comparison results of joint deraining and semantic segmentation on RainCityscape dataset comprising three rain intensities ($\alpha \in \{0.01, 0.02, 0.03\}$ where α denotes the intensity of the rain streaks). We use DeepLabV3+ [8] for semantic segmentation. We compare the models that show an improvement in the semantic segmentation performance which is measured as mIOU metric. avg. in the metric column denotes average value of all α.

Metric	Rainy	ReHEN	PReNet	RCDNet	MPR-Net	MCW-Net	MCW-Net (Large)	Clean
PSNR	15.55	23.47	28.88	25.51	25.91	**33.94**	**35.82**	∞
SSIM	0.826	0.916	0.972	0.958	0.964	**0.981**	**0.987**	1.000
mIOU (avg.)	0.6254	0.4833	0.7636	0.7402	0.7516	**0.7679**	**0.7728**	0.7810
mIOU (α=0.01)	0.5528	0.6724	0.7765	0.7641	0.7626	**0.7743**	**0.7773**	
mIOU (α=0.02)	0.4816	0.6284	0.7652	0.7439	0.7509	**0.7703**	**0.7750**	
mIOU (α=0.03)	0.4171	0.5777	0.7492	0.7140	0.7414	**0.7590**	**0.7663**	

can degrade the visibility of objects under complex weather conditions, the incorporation of effective image enhancement would be helpful in several vision models. To this end, we apply the public semantic segmentation model DeepLabV3+ [8] on the Cityscape dataset [16]. Hu *et al.* [48] synthesized rain streaks on the Cityscape dataset with different rain intensities α ($\alpha \in \{0.01, 0.02, 0.03\}$). Quantitative results for the improvement of the semantic segmentation accuracy in addition to the deraining performance are reported in Table 5.9. The qualitative comparison is shown in Figure 5.9.



(a) Rainy (b) PReNet (c) RCDNet (d) MPR-Net (e) Ours (small) (f) Ours (large) (g) Clean

Figure 5.9: Examples of joint deraining and semantic segmentation. The first row denotes the deraining results on the RainCityscape dataset. The second row denotes the semantic segmentation results obtained by DeepLabV3+ [8].

5.4.6 Analysis on multi-level features

To achieve insight as to how much each level contributes in deraining process for each connection, we measure the feature importance of channel-wise attention in SE layer at each connection.We measure the feature importance as follows:

As presented in Section 5.3.1, the features of each level in the encoder part are aggregated at level l in the decoder part,

$$D^l_{concat} = (\bigoplus_{i=1}^{3} H^l_i(E^i_{out})) \oplus H_{up}(D^{l+1}_{out}) \tag{5.8}$$

$$= \tilde{D}^l_1 \oplus \tilde{D}^l_2 \oplus \tilde{D}^l_3 \oplus \tilde{D}^l_{normal}, \tag{5.9}$$

where $\tilde{D}^l_i$ and $\tilde{D}^l_{normal}$ are the results of $H^l_i(E^i_{out})$ and $H_{up}(D^{l+1}_{out})$, respectively. Note that D^{l+1}_{out} means the output of previous layer. Afterwards, D^l_{concat} is fed into SE layer f_{SE} as

$$f_{SE}(D^l_{concat}) = f_{SE}([\tilde{D}^l_1, \tilde{D}^l_2, \tilde{D}^l_3, \tilde{D}^l_{normal}]) \tag{5.10}$$

$$= [\hat{D}^l_1, \hat{D}^l_2, \hat{D}^l_3, \hat{D}^l_{normal}], \tag{5.11}$$

where, each $\hat{D}^l_i$ can be considered as corresponding output of $\tilde{D}^l_i$ because f_{SE} is channel-wise attention operation. Now, we obtain the feature importance by applying L_2 norm and normalization to each of them.

$$\tilde{\lambda}^l_i = \frac{\|\tilde{D}^l_i\|_2}{\sum_j \|\tilde{D}^l_j\|_2},\ i = 1, 2, 3, normal \tag{5.12}$$

$$\hat{\lambda}^l_i = \frac{\|\hat{D}^l_i\|_2}{\sum_j \|\hat{D}^l_j\|_2},\ i = 1, 2, 3, normal, \tag{5.13}$$

where $\tilde{\lambda}^l_i$ and $\hat{\lambda}^l_i$ denote the feature importance before and after SE layer, respectively.

We calculate the feature importance before and after the SE layer for Rain200H and SPA-DATA datasets as described above, and we report the results in Figure 5.10. We find that feature importance is evenly distributed before the SE layer, but more diversely distributed after the SE layer. Combining such results with Table 5.6 and Figure 5.8, we assume that the channel-wise attention guided via SE operation has a crucial contribution to deraining. Furthermore, unspecified distribution of feature importance before the SE layer could cause performance degradation, implying that simple connections such as addition and concatenation could be detrimental to the performance. In this respect, we suggest that the SE layer emphasizes more meaningful features for recovering rainy images at each level.

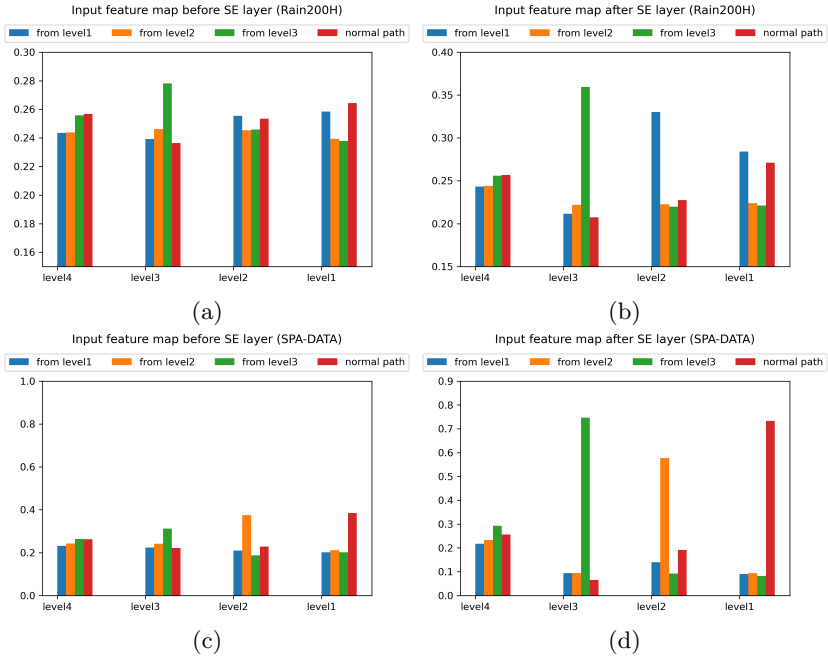


Figure 5.10: Intensity analysis of channel-wise attentions at each MLC on Rain200H and SPA-DATA datasets.

5.5 Conclusion

In this study, we present the multi-level connections and an adaptive regional attention network structure for single-image deraining. The proposed MCW-Net adaptively aggregates features via connections between multiple levels and the SE block in the background recovery. To utilize rich long-range rain-free background information in the deraining process, we propose a novel WRNL. The proposed method outperforms existing state-of-the-art methods. In particular, the network restores the details of the input image and almost completely removes rain streaks on both the synthesized and the real-world datasets. Furthermore, additional experiments demonstrate that MCW-Net contributes to other vision tasks by enhancing images degraded under bad weather conditions.

Bibliography

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, *Deep learning with differential privacy*, in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 308–318.

[2] S. Ahmed, Z. Yang, H. Mathias, B. Pascal, F. Mario, and M. Backes, *Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models*, arXiv preprint arXiv:1806.01246, (2018).

[3] P. Baldi and K. Hornik, *Neural networks and principal component analysis: Learning from examples without local minima*, Neural networks, 2 (1989), pp. 53–58.

[4] P. C. Barnum, S. Narasimhan, and T. Kanade, *Analysis of rain and snow in frequency space*, IJCV, 86 (2010), p. 256.

[5] J. Bossu, N. Hautière, and J.-P. Tarel, *Rain or snow detection in image sequences through use of a histogram of orientation of streaks*, IJCV, 93 (2011), pp. 348–367.

[6] D. Chen, J.-P. Mei, C. Wang, Y. Feng, and C. Chen, *Online knowledge distillation with diverse peers*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 3430–3437.

[7] H. Chen, Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu, and W. Gao, *Pre-trained image processing transformer*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 12299–12310.

[8] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, *Encoder-decoder with atrous separable convolution for semantic image segmentation*, in ECCV, 2018.

[9] T. Chen and C. Guestrin, *Xgboost: A scalable tree boosting system*, in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, ACM, 2016, pp. 785–794.

[10] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A simple framework for contrastive learning of visual representations*, in International conference on machine learning, PMLR, 2020, pp. 1597–1607.

[11] X. Chen and K. He, *Exploring simple siamese representation learning*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 15750–15758.

[12] Y.-L. Chen and C.-T. Hsu, *A generalized low-rank appearance model for spatio-temporally correlated rain streaks*, in ICCV, 2013.

[13] J. H. Cho and B. Hariharan, *On the efficacy of knowledge distillation*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 4794–4802.

[14] I. Chung, S. Park, J. Kim, and N. Kwak, *Feature-map-level online adversarial knowledge distillation*, in International Conference on Machine Learning, PMLR, 2020, pp. 2006–2015.

[15] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, *Describing textures in the wild*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 3606–3613.

[16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, *The cityscapes dataset for semantic urban scene understanding*, in CVPR, 2016.

[17] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems, 2 (1989), pp. 303–314.

[18] N. Dalal and B. Triggs, *Histograms of oriented gradients for human detection*, 2005.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.

[20] L.-J. Deng, T.-Z. Huang, X.-L. Zhao, and T.-X. Jiang, *A directional global sparse model for single image rain removal*, Applied Mathematical Modelling, 59 (2018), pp. 662–679.

[21] S. Deng, M. Wei, J. Wang, Y. Feng, L. Liang, H. Xie, F. L. Wang, and M. Wang, *Detail-recovery image deraining via context aggregation networks*, in CVPR, 2020.

[22] T. Ding, D. Li, and R. Sun, *Suboptimal local minima exist for wide neural networks with smooth activations*, Mathematics of Operations Research, (2022).

[23] Y. Dong, Q.-A. Fu, X. Yang, T. Pang, H. Su, Z. Xiao, and J. Zhu, *Benchmarking adversarial robustness on image classification*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 321–331.

[24] C. Dwork, *A firm foundation for private data analysis*, Communications of the ACM, 54 (2011), pp. 86–95.

[25] D. Eigen, D. Krishnan, and R. Fergus, *Restoring an image taken through a window covered with dirt or rain*, in Proceedings of the IEEE international conference on computer vision, 2013, pp. 633–640.

[26] E. Englesson and H. Azizpour, *Efficient evaluation-time uncertainty estimation by improved distillation*, arXiv preprint arXiv:1906.05419, (2019).

[27] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, *The pascal visual object classes (voc) challenge*, IJCV, 88 (2010), pp. 303–338.

[28] Z. Fan, H. Wu, X. Fu, Y. Huang, and X. Ding, *Residual-guide network for single image deraining*, in ACM MM, 2018.

[29] X. Fu, J. Huang, D. Zeng, Y. Huang, X. Ding, and J. Paisley, *Removing rain from single images via a deep detail network*, in CVPR, 2017.

[30] S. Gidaris, A. Bursuc, G. Puy, N. Komodakis, M. Cord, and P. Perez, *Obow: Online bag-of-visual-words generation for self-supervised learning*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 6830–6840.

[31] Y. Gong, L. Liu, M. Yang, and L. Bourdev, *Compressing deep convolutional networks using vector quantization*, arXiv preprint arXiv:1412.6115, (2014).

[32] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[33] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, arXiv preprint arXiv:1412.6572, (2014).

[34] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al., *Bootstrap your own latent-a new approach to self-supervised learning*, Advances in neural information processing systems, 33 (2020), pp. 21271–21284.

[35] T. Guo, H. Seyed Mousavi, T. Huu Vu, and V. Monga, *Deep wavelet prediction for image super-resolution*, in CVPR Workshops, 2017.

[36] A. Gupta, A. Vedaldi, and A. Zisserman, *Synthetic data for text localisation in natural images*, in CVPR, 2016, pp. 2315–2324.

[37] M. Gürbüzbalaban, A. Ozdaglar, and P. Parrilo, *Why random reshuffling beats stochastic gradient descent*, arXiv preprint arXiv:1510.08560, (2015).

[38] S. Han, H. Mao, and W. J. Dally, *Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding*, arXiv preprint arXiv:1510.00149, (2015).

[39] F. He, B. Wang, and D. Tao, *Piecewise linear activations substantially shape the loss surfaces of neural networks*, arXiv preprint arXiv:2003.12236, (2020).

[40] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, *Momentum contrast for unsupervised visual representation learning*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 9729–9738.

[41] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn*, in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.

[42] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[43] B. Heo, M. Lee, S. Yun, and J. Y. Choi, *Knowledge distillation with adversarial samples supporting decision boundary*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 3771–3778.

[44] G. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, arXiv preprint arXiv:1503.02531, (2015).

[45] K. Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural networks, 4 (1991), pp. 251–257.

[46] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, arXiv preprint arXiv:1704.04861, (2017).

[47] J. Hu, L. Shen, and G. Sun, *Squeeze-and-excitation networks*, in CVPR, 2018.

[48] X. Hu, C.-W. Fu, L. Zhu, and P.-A. Heng, *Depth-attentional features for single-image rain removal*, in CVPR, 2019.

[49] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[50] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, *Image-to-image translation with conditional adversarial networks*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1125–1134.

[51] K. Jiang, Z. Wang, P. Yi, C. Chen, B. Huang, Y. Luo, J. Ma, and J. Jiang, *Multi-scale progressive fusion network for single image deraining*, in CVPR, 2020.

[52] L.-W. Kang, C.-W. Lin, and Y.-H. Fu, *Automatic single-image-based rain streaks removal via image decomposition*, TIP, 21 (2011).

[53] K. Kawaguchi, *Deep learning without poor local minima*, Advances in neural information processing systems, 29 (2016).

[54] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li, *Novel dataset for fine-grained image categorization: Stanford dogs*, in Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC), vol. 2, Citeseer, 2011.

[55] A. Krizhevsky, G. Hinton, et al., *Learning multiple layers of features from tiny images*, (2009).

[56] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in NIPS, 2012, pp. 1097–1105.

[57] M. Kubat, *Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7.*, The Knowledge Engineering Review, 13 (1999), pp. 409–412.

[58] S. Laine and T. Aila, *Temporal ensembling for semi-supervised learning*, arXiv preprint arXiv:1610.02242, (2016).

[59] T. Laurent and J. Brecht, *The multilinear structure of relu networks*, in International conference on machine learning, PMLR, 2018, pp. 2908–2916.

[60] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural networks, 6 (1993), pp. 861–867.

[61] G. Li, X. He, W. Zhang, H. Chang, L. Dong, and L. Lin, *Non-locally enhanced encoder-decoder network for single image de-raining*, in ACM MM, 2018.

[62] R. Li, L.-F. Cheong, and R. T. Tan, *Heavy rain image restoration: Integrating physics model and conditional adversarial learning*, in CVPR, 2019.

[63] X. Li, J. Wu, Z. Lin, H. Liu, and H. Zha, *Recurrent squeeze-and-excitation context aggregation net for single image deraining*, in ECCV, 2018.

[64] Y. Li, R. T. Tan, X. Guo, J. Lu, and M. S. Brown, *Rain streak removal using layer priors*, in CVPR, 2016.

[65] Z. Li and D. Hoiem, *Reducing overconfident errors outside the known distribution*, (2018).

[66] S. Liang, Y. Li, and R. Srikant, *Enhancing the reliability of out-of-distribution image detection in neural networks*, arXiv preprint arXiv:1706.02690, (2017).

[67] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Microsoft coco: Common objects in context*, in European conference on computer vision, Springer, 2014, pp. 740–755.

[68] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, *Multi-level wavelet-cnn for image restoration*, in CVPR Workshops, 2018.

[69] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, *Ssd: Single shot multibox detector*, in ECCV, Springer, 2016, pp. 21–37.

[70] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, *Understanding membership inferences on well-generalized learning models*, arXiv preprint arXiv:1802.04889, (2018).

[71] H. Lu and K. Kawaguchi, *Depth creates no bad local minima*, arXiv preprint arXiv:1702.08580, (2017).

[72] Y. Luo, Y. Xu, and H. Ji, *Removing rain from a single image via discriminative sparse coding*, in ICCV, 2015.

[73] A. Malinin and M. Gales, *Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness*, Advances in Neural Information Processing Systems, 32 (2019).

[74] A. Malinin, B. Mlodozeniec, and M. Gales, *Ensemble distribution distillation*, arXiv preprint arXiv:1905.00076, (2019).

[75] H. B. McMahan and G. Andrew, *A general approach to adding differential privacy to iterative training procedures*, arXiv preprint arXiv:1812.06210, (2018).

[76] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, *Learning differentially private recurrent language models*, arXiv preprint arXiv:1710.06963, (2017).

[77] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, *Improved knowledge distillation via teacher assistant*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 5191–5198.

[78] M. P. Naeini, G. Cooper, and M. Hauskrecht, *Obtaining well calibrated probabilities using bayesian binning*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29, 2015.

[79] A. S. Nemirovskij and D. B. Yudin, *Problem complexity and method efficiency in optimization*, (1983).

[80] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, *Reading digits in natural images with unsupervised feature learning*, (2011).

[81] Q. Nguyen, *On connected sublevel sets in deep learning*, in International Conference on Machine Learning, PMLR, 2019, pp. 4790–4799.

[82] A. Niculescu-Mizil and R. Caruana, *Predicting good probabilities with supervised learning*, in Proceedings of the 22nd international conference on Machine learning, 2005, pp. 625–632.

[83] N. Passalis and A. Tefas, *Learning deep representations with probabilistic knowledge transfer*, in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 268–284.

[84] H. Petzka and C. Sminchisescu, *Non-attracting regions of local minima in deep and wide neural networks*, Journal of Machine Learning Research, 22 (2021), pp. 1–34.

[85] P. Porwik and A. Lisowska, *The haar-wavelet transform in digital image processing: its status and achievements*, Machine graphics and vision, 13 (2004), pp. 79–98.

[86] R. Qian, R. T. Tan, W. Yang, J. Su, and J. Liu, *Attentive generative adversarial network for raindrop removal from a single image*, in CVPR, 2018.

[87] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, in CVPR, 2016, pp. 779–788.

[88] D. Ren, W. Zuo, Q. Hu, P. Zhu, and D. Meng, *Progressive image deraining networks: a better and simpler baseline*, in CVPR, 2019.

[89] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, in NIPS, 2015, pp. 91–99.

[90] L. Shen, Z. Yue, Q. Chen, F. Feng, and J. Ma, *Deep joint rain and haze removal from a single image*, in ICPR, IEEE, 2018.

[91] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, *Membership inference attacks against machine learning models*, in 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 3–18.

[92] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv preprint arXiv:1409.1556, (2014).

[93] S. Song, K. Chaudhuri, and A. D. Sarwate, *Stochastic gradient descent with differentially private updates*, in 2013 IEEE Global Conference on Signal and Information Processing, IEEE, 2013, pp. 245–248.

[94] I. G. Sprinkhuizen-Kuyper and E. J. Boers, *A local minimum for the 2-3-1 xor network*, IEEE Transactions on Neural Networks, 10 (1999), pp. 968–971.

[95] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, JMLR, 15 (2014), pp. 1929–1958.

[96] K. Sun, B. Xiao, D. Liu, and J. Wang, *Deep high-resolution representation learning for human pose estimation*, in CVPR, 2019.

[97] M. Tan and Q. Le, *Efficientnet: Rethinking model scaling for convolutional neural networks*, in International Conference on Machine Learning, PMLR, 2019, pp. 6105–6114.

[98] M. Tan, R. Pang, and Q. V. Le, *Efficientdet: Scalable and efficient object detection*, arXiv preprint arXiv:1911.09070, (2019).

[99] A. Tarvainen and H. Valpola, *Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results*, Advances in neural information processing systems, 30 (2017).

[100] L. Trottier, P. Gigu, B. Chaib-draa, et al., *Parametric exponential linear unit for deep convolutional neural networks*, in ICMLA, IEEE, 2017.

[101] L. Venturi, A. S. Bandeira, and J. Bruna, *Spurious valleys in one-hidden-layer neural network optimization landscapes*, Journal of Machine Learning Research, 20 (2019), p. 133.

[102] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, *The caltech-ucsd birds-200-2011 dataset*, (2011).

[103] G. Wang, C. Sun, and A. Sowmya, *Erl-net: Entangled representation learning for single image de-raining*, in ICCV, 2019.

[104] H. Wang, Q. Xie, Q. Zhao, and D. Meng, *A model-driven deep neural network for single image rain removal*, in CVPR, 2020.

[105] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, et al., *Deep high-resolution representation learning for visual recognition*, arXiv preprint arXiv:1908.07919, (2019).

[106] T. Wang, X. Yang, K. Xu, S. Chen, Q. Zhang, and R. W. Lau, *Spatial attentive single-image deraining with a high quality real rain dataset*, in CVPR, 2019.

[107] X. Wang, R. Girshick, A. Gupta, and K. He, *Non-local neural networks*, in CVPR, June 2018.

[108] Y. Wang, S. Liu, C. Chen, and B. Zeng, *A hierarchical approach for rain or snow removing in a single color image*, TIP, 26 (2017), pp. 3936–3950.

[109] Y. Wang, Y. Song, C. Ma, and B. Zeng, *Rethinking image deraining via rain streaks and vapors*, arXiv preprint arXiv:2008.00823, (2020).

[110] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, et al., *Image quality assessment: from error visibility to structural similarity*, TIP, 13 (2004), pp. 600–612.

[111] X. Wei, S. Liang, X. Cao, and J. Zhu, *Transferable adversarial attacks for image and video object detection*, arXiv preprint arXiv:1811.12641, (2018).

[112] Q. Wu, L. Wang, K. N. Ngan, H. Li, F. Meng, and L. Xu, *Subjective and objective de-raining quality assessment towards authentic rain image*, IEEE Transactions on Circuits and Systems for Video Technology, 30 (2020), pp. 3883–3897.

[113] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, *Adversarial examples for semantic segmentation and object detection*, in ICCV, 2017, pp. 1369–1378.

[114] P. Xu, K. A. Ehinger, Y. Zhang, A. Finkelstein, S. R. Kulkarni, and J. Xiao, *Turkergaze: Crowdsourcing saliency with webcam based eye tracking*, arXiv preprint arXiv:1504.06755, (2015).

[115] T.-B. Xu and C.-L. Liu, *Data-distortion guided self-distillation for deep neural networks*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 5565–5572.

[116] W. Yang, J. Liu, S. Yang, and Z. Guo, *Scale-free single image deraining via visibility-enhanced recurrent wavelet learning*, TIP, 28 (2019).

[117] W. Yang, R. T. Tan, J. Feng, J. Liu, Z. Guo, and S. Yan, *Deep joint rain detection and removal from a single image*, in CVPR, 2017.

[118] W. Yang, R. T. Tan, S. Wang, Y. Fang, and J. Liu, *Single image deraining: From model-based to data-driven and beyond*, arXiv preprint arXiv:1912.07150, (2019).

[119] W. Yang, S. Wang, D. Xu, X. Wang, and J. Liu, *Towards scale-free rain streak removal via self-supervised fractal band learning.*, in AAAI, 2020.

[120] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, *Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop*, arXiv preprint arXiv:1506.03365, (2015).

[121] W. Yu, Z. Huang, W. Zhang, L. Feng, and N. Xiao, *Gradual network for single image de-raining*, in ACM MM, 2019.

[122] L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng, *Revisiting knowledge distillation via label smoothing regularization*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[123] Z. Yue, J. Xie, Q. Zhao, and D. Meng, *Semi-supervised video deraining with dynamical rain generator*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 642–652.

[124] C. Yun, S. Sra, and A. Jadbabaie, *Small nonlinearities in activation functions create bad local minima in neural networks*, arXiv preprint arXiv:1802.03487, (2018).

[125] S. Yun, J. Park, K. Lee, and J. Shin, *Regularizing class-wise predictions via self-knowledge distillation*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13876–13885.

[126] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, *Multi-stage progressive image restoration*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 14821–14831.

[127] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, *Barlow twins: Self-supervised learning via redundancy reduction*, in International Conference on Machine Learning, PMLR, 2021, pp. 12310–12320.

[128] H. Zhang and V. M. Patel, *Density-aware single image de-raining using a multi-stream dense network*, in CVPR, 2018.

[129] H. Zhang, V. Sindagi, and V. M. Patel, *Image de-raining using a conditional generative adversarial network*, IEEE transactions on circuits and systems for video technology, (2019).

[130] K. Zhang, W. Luo, W. Ren, J. Wang, F. Zhao, L. Ma, and H. Li, *Beyond monocular deraining: Stereo image deraining via semantic understanding*, in ECCV, 2020.

[131] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, *Be your own teacher: Improve the performance of convolutional neural networks via self distillation*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3713–3722.

[132] X. Zhang, T. Wang, W. Luo, and P. Huang, *Multi-level fusion and attention-guided cnn for image dehazing*, IEEE Transactions on Circuits and Systems for Video Technology, (2020).

[133] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, *Deep mutual learning*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4320–4328.

[134] Y. Zhou and Y. Liang, *Critical points of linear neural networks: Analytical forms and landscape properties*, in International Conference on Learning Representations, 2018.

[135] L. Zhu, C.-W. Fu, D. Lischinski, and P.-A. Heng, *Joint bi-layer optimization for single-image rain streak removal*, in ICCV, 2017, pp. 2526–2534.

국문초록

본 학위 논문은 심층 신경망의 손실 표면에 대하여 다룬다. 심층 신경망의 손실 함수는 볼록 함수와 같이 나쁜 국소점을 가지는가? 조각적으로 선형은 활성함수를 가지는 경우에 대해서는 잘 알려였지만, 일반적인 매끄러운 활성함수를 가지는 심층 신경망에 대해서는 아직까지 알려지지 않은 것이 많다. 본 연구에서는 나쁜 국소점이 일반적인 매끄러운 활성함수에서도 존재함을 보인다. 이것은 심층 신경망의 손실 표면에 대한 이해에 부분적인 설명을 제공해 줄 것이다. 추가적으로 본 논문에서는 학습 이론, 사생활 보호적인 기계 학습, 컴퓨터 비전 등의 분야에서의 심층 신경망의 다양한 응용을 선보일 예정이다.

주요어휘: Deep learning, neural network, local minimum

학번: 2015-22567