Wide Activated Separate 3D Convolution for Video Super-Resolution

by

Xiafei Yu

Thesis submitted to the University of Ottawa

In partial fulfillment of the requirements

For the M.A.Sc. degree in

Electrical and Computer Engineering

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

© Xiafei Yu, Ottawa, Canada, 2019

Abstract

Video super-resolution (VSR) aims to recover a realistic high-resolution (HR) frame from its corresponding center low-resolution (LR) frame and several neighbouring supporting frames. The neighbouring supporting LR frames can provide extra information to help recover the HR frame. However, these frames are not aligned with the center frame due to the motion of objects. Recently, many video super-resolution methods based on deep learning have been proposed with the rapid development of neural networks. Most of these methods utilize motion estimation and compensation models as preprocessing to handle spatio-temporal alignment problem. Therefore, the accuracy of these motion estimation models are critical for predicting the high-resolution frames. Inaccurate results of motion compensation models will lead to artifacts and blurs, which also will damage the recovery of high-resolution frames.

We propose an effective wide activated separate 3 dimensional (3D) Convolution Neural Network (CNN) for video super-resolution to overcome the drawback of utilizing motion compensation models. Separate 3D convolution factorizes the 3D convolution into convolutions in the spatial and temporal domain, which have benefit for the optimization of spatial and temporal convolution components. Therefore, our method can capture temporal and spatial information of input frames simultaneously without additional motion evaluation and compensation model. Moreover, the experimental results demonstrated the effectiveness of the proposed wide activated separate 3D CNN.

Acknowledgements

I would like to thank my supervisor, Professor Jiying Zhao. Thank him for supervising my research and offering me so much help. I would also like to offer my best regards to Yang Zhou, Tingyu Ye, Huizhen Zhang and all of my colleagues in our lab who helped me during my study. Lastly, I would like to thank all of my friends Nan Liu, Rui Sun, Wenbo Wu and Yuanyuan Guo for their love and help.

Dedication

This is dedicated to my mom. I appreciate very much that she always stands by me to support what I want to do. Thank her for her unconditional love and support, I would never be who I am without her.

Table of Contents

List of Tables ix

List of Figures x

Acronym xv

1 Introduction 1

1.1 Single Image Super-Resolution 1

1.1.1 Interpolation-Based Super-Resolution 2

1.1.2 Reconstruction-Based Super-Resolution 2

1.1.3 Learning-Based Super-Resolution 2

1.2 Video Super-Resolution 3

1.3 Thesis Motivations and Contributions 4

1.4 Thesis structure 5

2 Literature Review **6**

2.1 Single Image Super-Resolution 6

2.2 Video Super-Resolution 11

2.3 Summary 15

3 Convolution Neural Networks **16**

3.1 Layers 17

3.1.1 Convolutional Layers 17

3.1.2 Activation Layer 22

3.1.3 Pooling Layer 27

3.2 Training Methods 30

3.2.1 Initialization 30

3.2.2 Forward Propagation 31

3.2.3 Loss Function 32

3.2.4 Backward Propagation 33

3.2.5 Optimization 34

3.3 Typical Convolutional Architectures 37

3.3.1 AlexNet 37

3.3.2 VGGnet 38

3.3.3 ResNet 39

3.3.4 Inception . . . 40

3.4 Summary . . . 41

4 Video Super Resolution via Wide Activated Separate 3D Convolution 43

4.1 Overview . . . 43

4.2 Architecture of Model . . . 45

4.2.1 Wide Activated Separate 3D Convolutional Residual Block . . . 45

4.2.2 Temporal Features Fusion Layer . . . 49

4.2.3 High-Level Features Extraction Network . . . 50

4.2.4 HR Frame Reconstruction Layer . . . 50

4.2.5 Residual Learning . . . 51

4.3 Loss Function . . . 52

4.4 Summary . . . 52

5 Experimental Results and Evaluation 53

5.1 Environment Setup . . . 53

5.2 Experiment Settings . . . 54

5.2.1 Datasets . . . 54

5.2.2 Training and Parameters . . . 58

5.2.3 Evaluation Metrics . . . 59

5.2.4 Color Space . . . 61

5.3 Experimental Result Comparisons 63

5.3.1 Experiments Results 63

5.3.2 Results Comparison with Other methods 63

5.3.3 Results on additional real-world examples 67

5.4 Summary 71

6 Conclusions **78**

References **80**

List of Tables

5.1 The information about our training dataset. 56

5.2 The parameter size of each sub-network components. 58

5.3 PSNR and SSIM results on videos of VID4 with up-sampling scales of 2, 3 and 4. 63

5.4 Comparison of PSNR and SSIM on VID4 dataset with up-sampling scales of 2, 3 and 4. The first and last two frames are not used for calculation. Our method achieves the best performance in PSNR and SSIM. 65

5.5 Comparison of results of wide activated separate 3D convolution and original separate 3D convolution on videos of VID4 with up-sampling scale of 4. . . 69

List of Figures

2.1 The architecture of the SRCNN and FSRCNN (adopted from the original paper [1]). 7

2.2 The architecture of the ESPCN (adopted from the original paper [2]). . . . 8

2.3 The comparison of VDSR, SRResNet, EDSR and MDSR architecture (adopted from paper [3]). 9

2.4 The architecture of DenseSR (adopted from paper [3]). 10

2.5 The architecture of SRGAN (adopted from the original paper [4]). 11

2.6 Three variants architecture of VSRnet (adopted from the original paper [5]). 12

2.7 The spatial transformer motion compensation network and sub-pixel motion compensation layer (adopted from the original paper [6] and [7]). 12

2.8 The architecture of FRVSR (adopted from the original paper [8]). 13

2.9 The architecture of VSR-DUF (adopted from the original paper [9]). 13

2.10 The architecture of the temporally deformable alignment network (TDAN) (adopted from the original paper [10]). 14

2.11 The architecture of the 3D super-resolution network (3DSRnet) (adopted from the original paper [11]). 14

3.1 Convolution with stride 1 and 2. 19

3.2 Zero-padding. 20

3.3 2D convolution operation example. 20

3.4 3D convolution operations (adopted from paper [12]). 21

3.5 2D vs. 3D convolution operations. a) Applying 2D convolution on an image results in an image. b) Applying 2D convolution on a video volume (multiple frames as multiple channels) also results in an image. c) Applying 3D convolution on a video volume results in another volume, preserving temporal information of the input signal (adopted from paper [13]). 22

3.6 Sigmoid Function. 24

3.7 Tanh Function. 25

3.8 Relu Function. 26

3.9 PReLU Function. 27

3.10 ELUs Function. 28

3.11 Max pooling vs. average pooling. 29

3.12 An illustration of the architecture of AlexNet (adopted from the original paper [14]). 38

3.13 An illustration of the architecture of VGG-16. 39

3.14 The structure of a residual block. 40

3.15 The structure of naive Inception v1 (adopted from the original paper [15]). 41

4.1 The proposed wide activated separate 3D convolution network for VSR. . . 44

4.2 3D convolution and separate 3D convolution. 46

4.3 Comparison between original and wide activated 2D convolution residual blocks. 47

4.4 A residual separate 3D convolution block. 48

4.5 Concatenation layer. 49

4.6 The architecture of high-level feature extraction sub-network. 50

4.7 An illustration of sub-pixel layer with upsampling scale of 2. 51

5.1 The training loss changes along the epoches. 64

5.2 Visual comparison of wide activated separate 3D convolution network and other state-of-the-art methods on *calendar*. 65

5.3 Visual comparison of wide activated separate 3D convolution network and other state-of-the-art methods on *city*. 66

5.4 Visual comparison of wide activated separate 3D convolution network and other state-of-the-art methods on *foliage*. 66

5.5 Visual comparison of wide activated separate 3D convolution network and other state-of-the-art methods on *walk*. 67

5.6 The training loss changes of wide activated separate 3D convolution method and original activated separate 3D convolution method 68

5.7 Visual comparison of wide activated and original activated separate 3D convolution on *calendar*. 69

5.8 Visual comparison of wide activated and original activated separate 3D convolution on *city*. 70

5.9 Visual comparison of wide activated and original activated separate 3D convolution on *foliage*. 70

5.10 Visual comparison of wide activated and original activated separate 3D convolution on *walk*. 71

5.11 Visual comparison of wide activated and original activated separate 3D convolution on *hdclub_003_001*. 72

5.12 Visual comparison of wide activated and original activated separate 3D convolution on *hitachi_isee5_001*. 73

5.13 Visual comparison of wide activated and original activated separate 3D convolution on *jvc_009_001*. 74

5.14 Visual comparison of wide activated and original activated separate 3D convolution on *NYVTG_006*. 75

5.15 Visual comparison of wide activated and original activated separate 3D convolution on *PRVTG_012*. 76

5.16 Visual comparison of wide activated and original activated separate 3D convolution on *RMVTG_011*. 77

Acronyms

2D	2 Dimensional
3D	3 Dimensional
3DSRnet	3D Super-Resolution Network
AdaDelta	Adaptive Learning Rate
Adagrad	Adaptive Subgradient
Adam	Adaptive Moment
BN	Batch Normalization
CNN	Convolution Neural Network
DenseNet	Densely Connected Network
DenseSR	Dense Network for Image Super-Resolution
EDSR	Enhanced Deep Residual Network
ELU	Exponential Linear Unit

ESPCN	Efficient Sub-Pixel Convolution Network
FRVSR	Frame-Recurrent Video Super Resolution
FSRCNN	Fast Super-Resolution Convolutional Neural Network
GAN	Generative Adversarial Network
HR	High Resolution
ILSVRC	Image Large Scale Visual Recognition Challenge
JPEG	Joint Photographic Experts Group Standard
LR	Low Resolution
MDSR	Multi-Scale Deep Super-Resolution
MSE	Mean Squared Error
PIL	Python Imaging Library
PReLU	Parametric Rectified Linear Unit
PSNR	Peak-Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
ResNet	Residual Network
SGD	Stochastic Gradient Descent
SISR	Single Image Super-Resolution

SPMC	Sub-Pixel Motion Compensation
SR	Super-Resolution
SRCNN	Super-Resolution Convolutional Neural Network
SRGAN	Generative Adversarial Network for Single Image Super-Resolution
SRResNet	Super-Resolution Residual Network
SSIM	Structural Similarity
TDAN	Temporally Deformable Alignment Network
UHD	Ultra High Definition
VDSR	Very Deep Super-Resolution
VESPCN	Video Efficient Sub-Pixel Convolution Network
VGG-net	Very Deep Convolutional Networks
VSR	Video Super-Resolution
VSR-UDF	Video Super-Resolution Network using Dynamic Upsampling Filters
VSRnet	Video Super-Resolution Network

Chapter 1

Introduction

Super-Resolution (SR) is a class of techniques that enhance the resolution of an image or a video. There are two branches of SR techniques according to the amount of utilized input frames: Single Image Super-Resolution (SISR) and Video Super-Resolution (VSR).

1.1 Single Image Super-Resolution

SISR is a technique which aims to recover high quality and High Resolution (HR) image from its corresponding single low quality and Low Resolution (LR) image. Various image super-resolution methods have been proposed to implement the reconstruction process. These methods can be mainly classified into three categories [16]: interpolation-based, reconstruction-based, and learning-based.

1.1.1 Interpolation-Based Super-Resolution

Interpolation-based super-resolution methods which estimate the unknown intermediate pixel between the known pixels by using interpolation filters. There are several different interpolation filters such as Bilinear, Bicubic [17], Lanczos [18] so on. Interpolation based methods are fast but always result in over-smoothed and jagged edges as the high-frequency information of the resized image cannot be reconstructed by low-pass interpolation filters.

1.1.2 Reconstruction-Based Super-Resolution

Reconstruction-based Super-Resolution methods [19, 20, 21, 22, 23, 24] recover HR images by applying various smoothness filters when down-sample HR images to produce LR images. However, the limitation of such methods is that the proposed prior knowledge is unable to fit for arbitrary images.

1.1.3 Learning-Based Super-Resolution

Learning-based super-resolution methods exploit knowledge which is learned from a large number of external training examples to recover HR images. The knowledge that algorithms learned is the self-similarities among similar images. However, there is a limitation that the quality of reconstructed HR images is affected by the training examples. Therefore, the performance of this category of methods can suffer from insufficient training examples. Lin et al. [25] illustrates this limitation. However, this limitation can be solved with the emergence of large-scale image training dataset, such as ImageNet dataset [26] and DIV2K

dataset [27]. With the development of deep learning, deep learning based methods are becoming mainstream algorithms.

1.2 Video Super-Resolution

VSR reconstructs single HR frame from its corresponding LR version and several neighboring LR frames. There is additional information in the surrounding LR frames due to the temporal relationships between consecutive frames in a video. With the aggregation of several consecutive LR frames, the frames that are not aligned due to the movement of cameras or objects will introduce flickering artifacts into reconstructed HR frames. Therefore, aligning these consecutive frames is a challenge for VSR.

Most of the VSR methods have similar structure. A complete method contains two components: a motion estimation and compensation sub-model and an super-resolution sub-model. One limitation of these two-component methods is that the quality of recovered HR frame is heavily related to the accuracy of motion estimation and compensation model. However, the motion between frames has no ground truth, which will affect the performance of the motion estimation sub-model and the whole VSR model. Another limitation is that the motion estimation and compensation sub-model is independent of the super resolution sub-model. Implementing the two sub-models separately could result in sub-optimal performance.

1.3 Thesis Motivations and Contributions

With the technical developments in electronic displaying hardware, the resolution of displays can reach up to 4K (3840 × 2160) and 8K (7680 × 4320) Ultra High Definition (UHD). However, most of the contents still cannot match such high resolution. Therefore, VSR technique is desired to obtain high quality visual content. Moreover, in recent years, most research focuses on SISR technique and made significant performance improvements. However, the VSR technique is still under-explored. In this thesis, we only focus on VSR technique.

With the development of deep learning algorithms, the powerful convolution neural network (CNN) [14] have revolutionized computer vision including VSR task. In recent years, several neural network based methods have been proposed to make high quality reconstructions of high-resolution videos. Our proposed method also based on the CNN. Moreover, our proposed method eliminates the motion estimation and compensation submodel in order to avoid the limitations of the previous two-component models. There are two contributions in our proposed method:

One contribution is that we proposed an end-to-end VSR method which utilize separate 3D convolution in VSR task for the first time. The use of separate 3D convolution allows capturing the spatial-temporal features directly in VSR models. Our proposed method can avoid the limitations of two-component VSR models efficiently.

Another contribution is that we proposed the wide activated separate 3D convolution residual block, which can reduce the model size significantly compare to the original 3D convolution, and can capture spatial-temporal features more efficiently than the 3D con-

volution and original separate 3D convolution.

These two contributions help our method achieve the state-of-the-art results with relative small model size compared to other methods.

The following conference paper has been generated via the thesis work:

(1) Xiafei Yu and Jiying Zhao, Wide Separate 3D Convolution for Video Super Resolution, IEEE GlobalSIP 2019, Ottawa, Canada, November 11-14, 2019, 5 pages. [28].

1.4 Thesis structure

In this thesis, we focus on VSR technique and propose a novel VSR method - wide activated separate 3D convolution for video super-resolution and an improved separate 3D convolution block - wide activated separate 3D convolution residual block. The thesis is organized as follows. Chapter 1 introduces the background of SR techniques. Chapter 2 illustrates several neural network based SISR methods and VSR methods. Chapter 3 introduces the concepts of convolution neural network briefly. Chapter 4 presents the proposed VSR method. Chapter 5 shows our experimental results and compares our method with other state-of-the-art methods. Chapter 6 gives the conclusions of the thesis.

Chapter 2

Literature Review

In this chapter, we review several representative neural network based single image super-resolution and video super-resolution methods.

2.1 Single Image Super-Resolution

We use the super-resolution convolutional neural network (SRCNN) proposed by Dong et al. [29] as the benchmark since it is one of the first proposed methods which utilize deep neural network to handle single image super resolution task. The overall architecture of the SRCNN is shown in Figure 2.1. The SRCNN contains three convolutional layers. The function of these three layers are patch extraction, nonlinear mapping and reconstruction, respectively. The structure of the SRCNN is simple but the performance was demonstrated to be better than traditional methods. The SRCNN attracted public attention to apply CNN to SISR task.

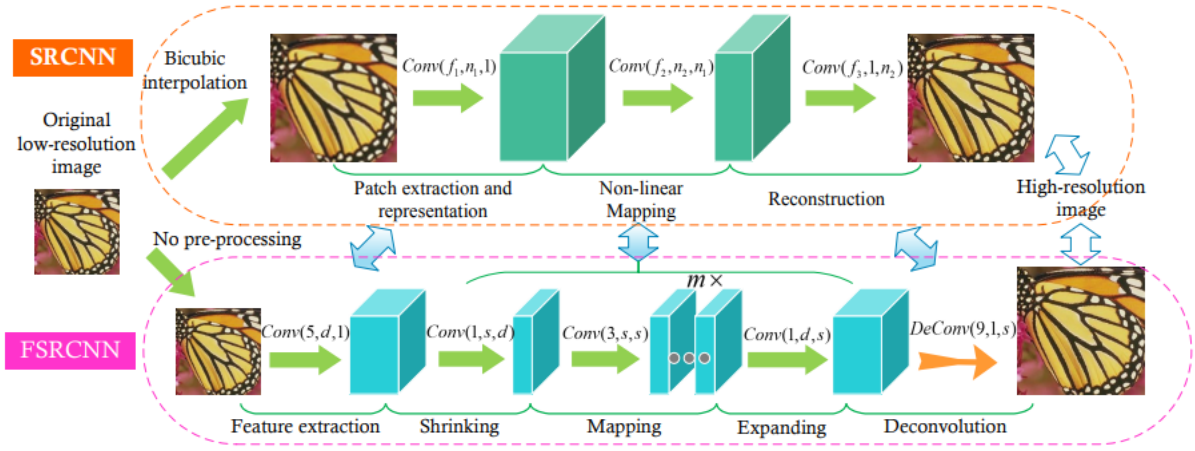


Figure 2.1: The architecture of the SRCNN and FSRCNN (adopted from the original paper [1]).

The problem of the SRCNN is that the input of the network are bicubic up-sampled HR images. Smoothing and bluring brought by interpolation may lead to further deviation in HR image reconstruction. Two representative papers were published to solve the problem: the fast super-resolution convolutional neural network (FSRCNN) [1] and the efficient sub-pixel convolutional network (ESPCN) [2]. The FSRCNN adopts deconvolution [30, 31] operation to up-sample the LR images adaptively. The comparison between the SRCNN and FSRCNN is shown in Figure 2.1. The ESPCN which increases resolution by rearranging the pixels of multiple LR feature maps as sub-pixels of HR images rather than by exploiting deconvolution layers. The structure of ESPCN is shown in Figure 2.2.

The very deep super resolution (VDSR) proposed by Kim et al. [32] is the first method that utilizes very deep neural network in SISR task. Inspired by the very deep convolutional networks (VGG-net) [33], the VDSR is a 20-layer model. Except the first and the last layer, each layer in the model has the same convolutional kernel size: $3 \times 3 \times 64$. However, a

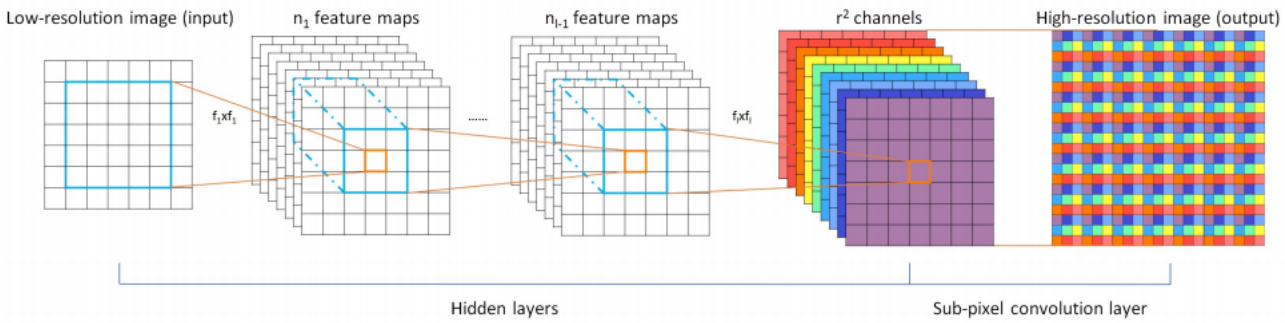
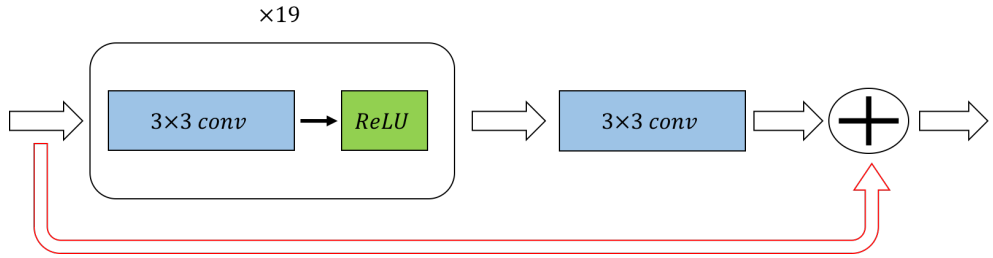


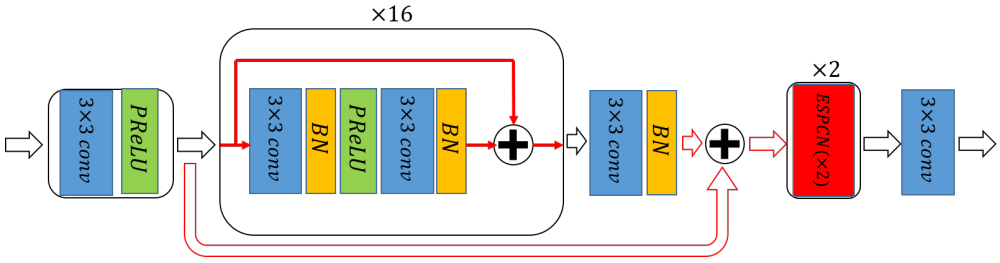
Figure 2.2: The architecture of the ESPCN (adopted from the original paper [2]).

plain architecture like VGG-net is hard to go deeper. The residual network (ResNet) [34] is a neural network that contains several residual blocks. A residual block consists of several convolution layers and a skip connection between input and output of the block. The connection structure maintains the gradients of deep layers, therefore, the ResNet can be a very deep neural network. Ledig et al. [4] proposed the super-resolution residual network (SRResNet) which adopts residual blocks to establish deeper network. Lee et al. [35] found that batch normalization (BN) [36] would be harmful for HR image construction. Hence they remove the BN part in the original residual blocks and proposed the enhanced deep residual network (EDSR) [35]. They also removed unnecessary rectified linear unit (ReLU) [37] activation layer in the original residual blocks to adapt to SISR task. The authors also proposed the multi-scale deep super-resolution (MDSR) [35] in the same paper to achieve the multi-scale architecture. The comparison of the VDSR, SRResNet, EDSR and MDSR architecture is shown in Figure 2.3

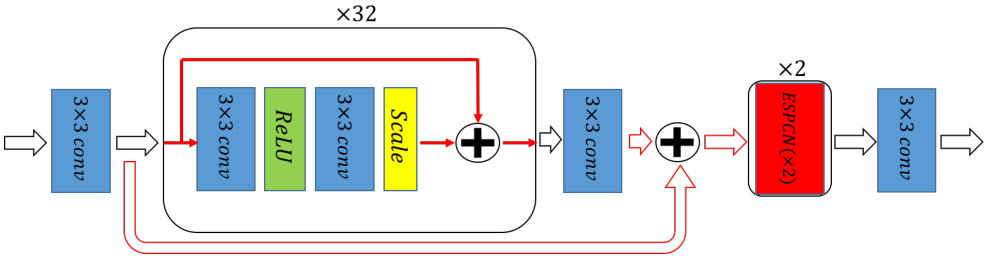
Besides the ResNet, the densely connected convolutional network (DenseNet) [38] also plays an important role in SISR task. Different from the ResNet, the DenseNets does not sum the output feature maps with the input feature maps pixel-wise but concate-



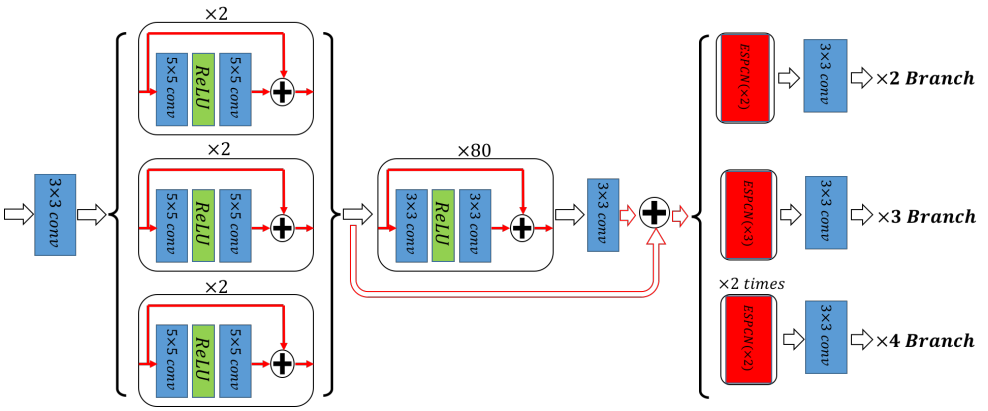
(a) VDSR



(b) SRResNet



(c) EDSR



(d) MDSR

Figure 2.3: The comparison of VDSR, SRResNet, EDSR and MDSR architecture (adopted from paper [3]).

nates all the feature maps of preceding layers and the output feature maps of the current layer image-wise. Also, each block which is made up of several layers would contains a bottleneck layer to reduce the parameter number. The residual dense network for image super-resolution (DenseSR) [39] adopts the DenseNet, concatenates all the output feature maps of convolution blocks, and outputs to the deconvolution layer. The structure of DenseSR is shown in Figure 2.4.

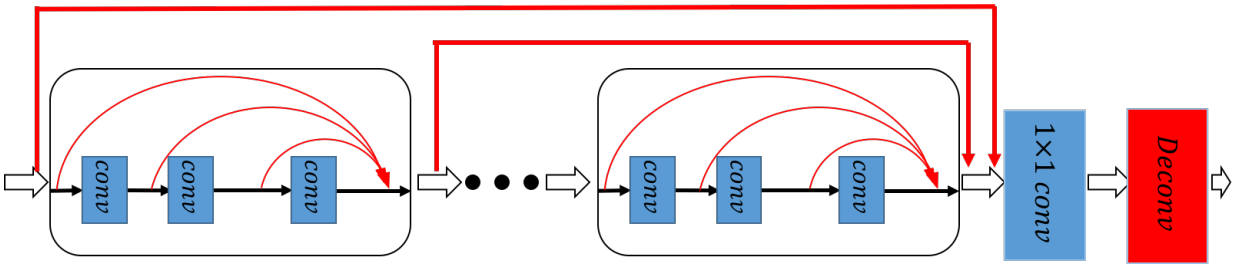
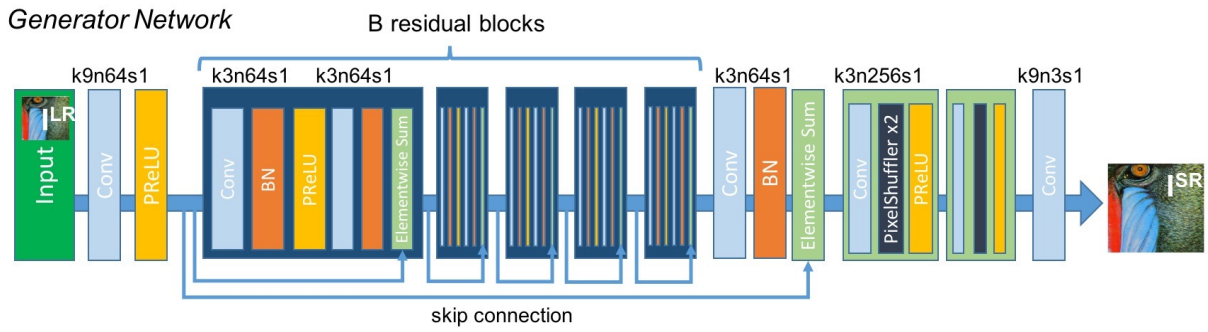
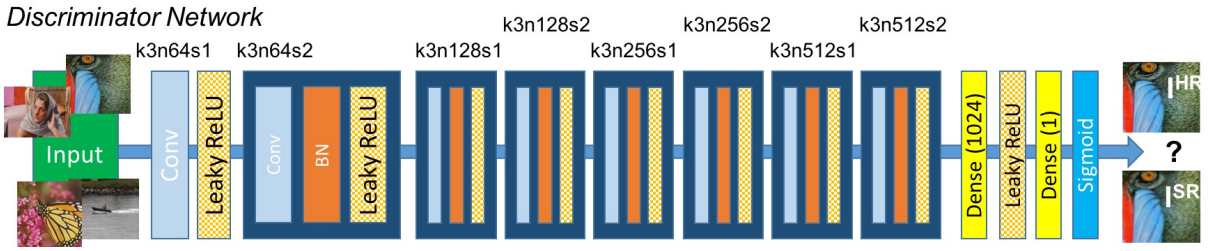


Figure 2.4: The architecture of DenseSR (adopted from paper [3]).

Another representative method is the photo-realistic single image super-resolution (SRGAN) [4], which is the first SISR method based on generative adversarial network (GAN) [40]. GAN is an architecture comprised of two sub-networks, a generative network and a discriminative network, one competing against the other. At the end, both networks reach optimization. The structure of the SRGAN is shown in Figure 2.5. Besides the novel architecture, the SRGAN also adopts perceptual loss [41] to design a new loss function, which combines the content loss (mean squared error (MSE) loss) and perceptual loss. This loss function is critical to help recover photo-realistic HR images.



(a) Generator



(b) Ddiscriminator

Figure 2.5: The architecture of SRGAN (adopted from the original paper [4]).

2.2 Video Super-Resolution

We use the video super-resolution network (VSRnet) proposed by Kappeler et al. [5] as the benchmark since it is considered to be the first proposed method which utilizes deep neural network to handle video super resolution task. Figure 2.6 shows three versions of VSRnet according to the stages where the input features are fused. They converted the video frames into the YCbCr color format and only used the luminance channel (Y) for training, testing and evaluation. They utilize the optical flow estimation algorithm [42] and adaptive motion compensation algorithm as their motion estimation and compensation model. However, these motion estimation and compensation models are independent of the super resolution model, which could result in sub-optimal performance.

To solve the problem of the VSRnet, several methods have been proposed. Caballero

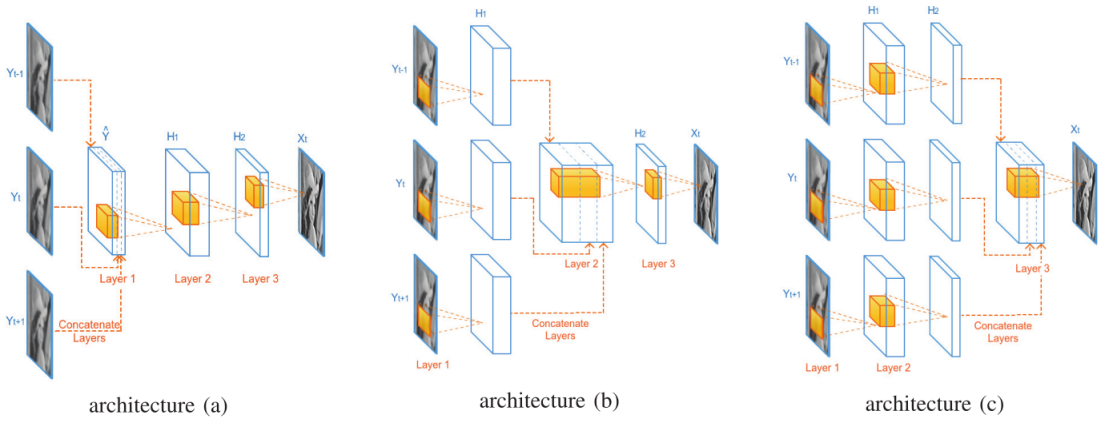
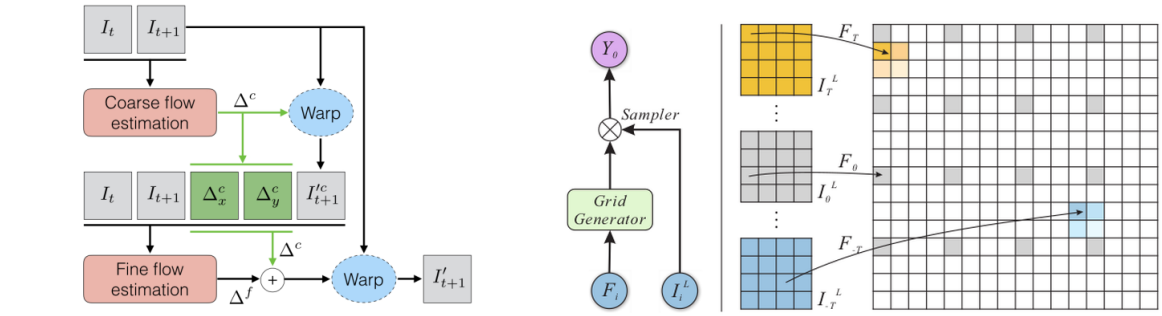


Figure 2.6: Three variants architecture of VSRnet (adopted from the original paper [5]).

et al. [6] proposed the video effecient sub-pixel convolutional network (VESPCN), which is based on the ESPCN. The VESPCN is a combination of an efficient spatial transformer motion compensation network and the ESPCN. The VESPCN is an end-to-end method as the flow estimation and compensation network and SR networks are trained jointly. Tao et al. [7] proposed another compensation method, the sub-pixel motion compensation (SPMC) layer, which can fuse information of multiple LR frames effectively. The spatial transformer network and SPMC are shown in Figure 2.7.



(a) Spatial transformer motion comepensation (b) Subpixel Motion Compensation layer

Figure 2.7: The spatial transformer motion compensation network and sub-pixel motion compensation layer (adopted from the original paper [6] and [7]).

Huang et al. [43] proposed a bidirectional recurrent architecture method to deal with VSR task . The problem is that they did not exploit any motion compensation in the model. Sajjadi et al. [8] proposed the frame-recurrent video super resolution (FRVSR) network with motion estimation and compensation network to overcome this problem. Instead of utilizing multiple LR frames, the recurrent architecture based methods exploit preceding and succeeding generated HR frames to establish current HR frame. The structure of the FRVSR is shown in Figure 2.8.

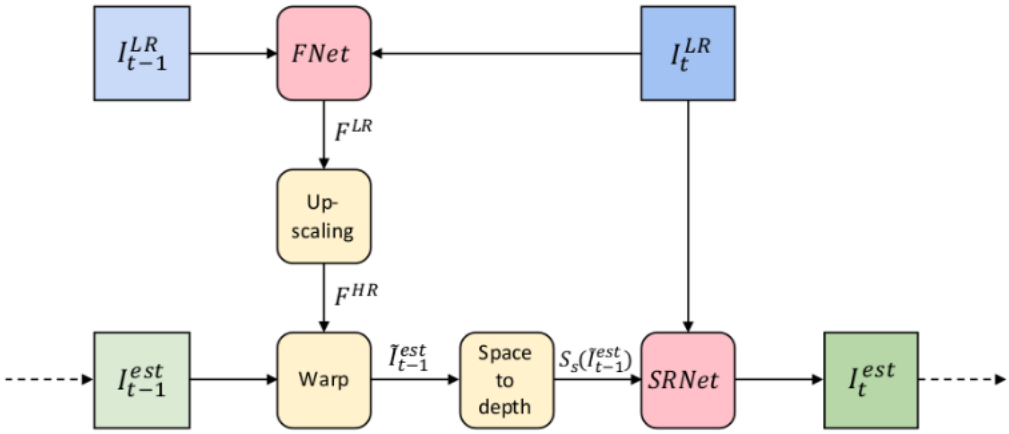


Figure 2.8: The architecture of FRVSR (adopted from the original paper [8]).

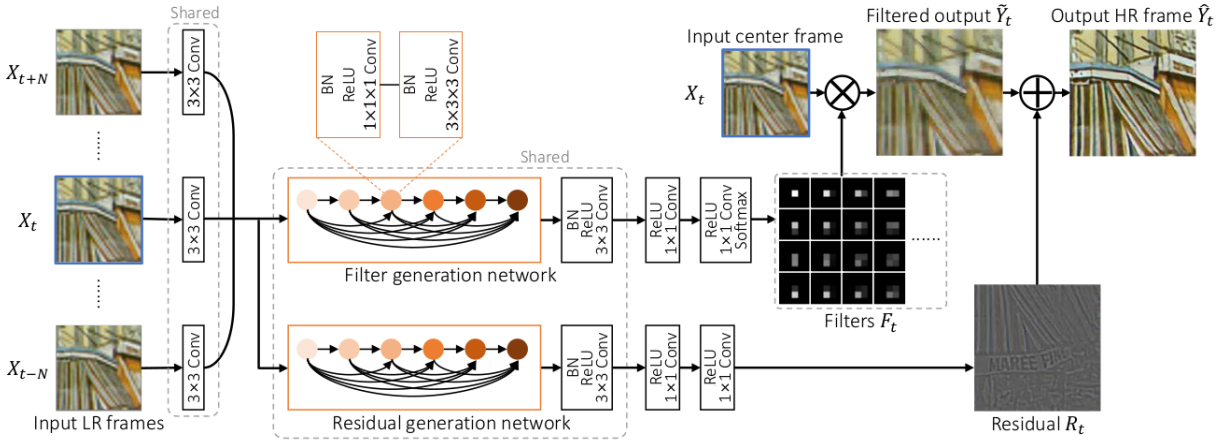


Figure 2.9: The architecture of VSR-DUF (adopted from the original paper [9]).

However, sufficiently high-quality motion estimation is not easy to obtain even with state-of-the-art optical flow estimation networks. Even with accurate motion fields, the image-wrapping based motion compensation will produce artifacts around image structures, which may be propagated into the final reconstructed HR frames. Several recent papers described architectures based on feature-wise alignment rather than optical flow compensation, which would avoid drawbacks of utilizing optical flow compensation in VSR task.

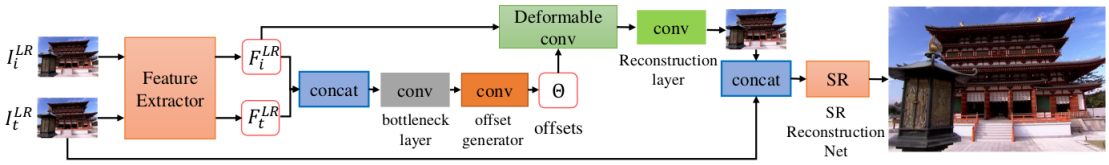


Figure 2.10: The architecture of the temporally deformable alignment network (TDAN) (adopted from the original paper [10]).

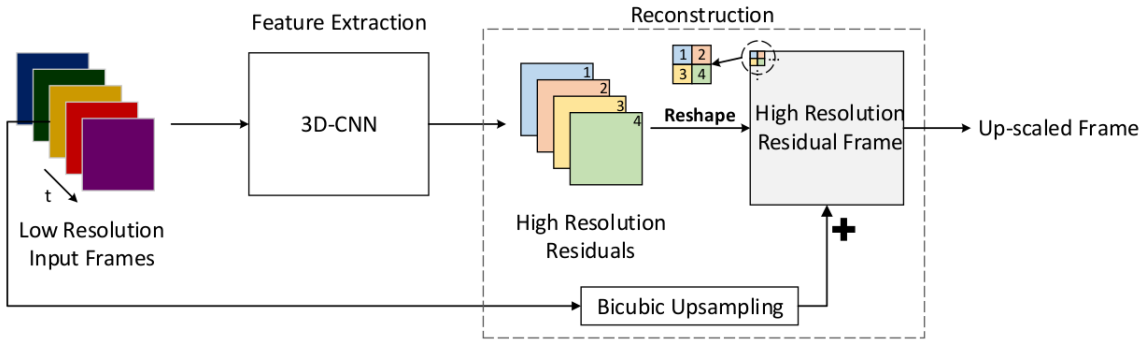


Figure 2.11: The architecture of the 3D super-resolution network (3DSRnet) (adopted from the original paper [11]).

Jo et al. [9] proposed the deep video super-resolution network using dynamic upsampling filters (VSR-UDF) which exploits dynamic filters [44] to generate dynamic upsampling filters according to the different context of input LR frames (architecture is shown

in Figure 2.9). Zhou et al. [45] also proposed a similar network based on dynamic upsampling filters. Tian et al. [10] uses deformable convolutional networks [46] to adaptively align feature maps rather than align LR frames (architecture is shown in Figure 2.10). This could avoid the drawbacks of using optical flow to compensate between input frames. Kim et al. [11] proposed to use 3D CNN [12] in VSR task. 3D CNNs can capture temporal information between input frames rather than compensate it (architecture is shown in Figure 2.11). Based on 2 dimensional (2D) convolution, 3D CNN introduces convolutional operation in temporal domain, combined with the convolution in spatial domain. The 3D convolution is better for spatial-temporal feature extraction.

2.3 Summary

In this chapter, we illustrated several CNN based SISR and VSR methods. Most of previous CNN based VSR methods focus on exploiting motion estimation and compensation models to better reconstruct HR frames. Unlike these works, we will propose a novel separate 3D convolution residual block to avoid the exploitation of motion estimation and compensation in VSR.

Chapter 3

Convolution Neural Networks

With the explosive development of Artificial Intelligence, deep learning algorithms (deep neural networks) have had a monumental growth in rencent years. Currently, deep neural networks provide the best solutions in the majority of Computer Vision and Natural Language Processing tasks. CNN, as one of the most important categories in neural networks, has achieved success in the domain of Computer Vision. CNNs also have proven to be very effective in super resolution tasks. A deep CNN is a sequence of layers, all of which could be divided into different categories with distinct functions. Typically, there are three main categories of layers: convolutional layer, activation layer and pooling layer. In this chapter, we will introduce these layers in Sec. 3.1. Then, we will present the training methods in Sec. 3.2. Finally, we will list several typical deep convolutional architectures in Sec. 3.3.

3.1 Layers

A deep CNN contains hundreds or thousands of layers, these layers are cascaded, which means the input of a layer is the output of previous layer. Typically, a CNN usually contains three kinds of layers: convolutional layers, activation layers and pooling layers.

3.1.1 Convolutional Layers

Convolutional layers are derived from convolution operation. According to the operating dimension, the convolutional layers could divided into 2D and 3D convolutional layers. There are some expansive structures in both 2D and 3D convolutional layers. We illuminate two kinds of structures utilized in our frameworks: deformable 2D convolutional layer and separate 3D convolutional layer.

Basically, the inputs of a CNN are considered to be images. A gray image only has one channel, and the value in each pixel indicates the brightness of this spot. While a color image has three channels, red, green and blue, these three channels stack over each other to represent a color image. The value of each pixel is ranging from 0 to 255. Every image can be represented as a matrix of pixel values. The matrix shape of images are w $\times$ h $\times$ c, where w is the width, h is the height and c is the channel number of images.

Usually, The convolution operation can be expressed by a function:

$$f(x) = W * x + b \tag{3.1}$$

where W is the weight (value) of convolutional kernel, $*$ represent the convolution operation, x is the input of convolutional kernel and b is the bias, which is a constant. The W

and b are trainable, and such trainable variables are named as parameters in CNNs.

2 Dimensional Convolutional Layer

2 dimensional convolution layers indicate that the convolution is operating in the spatial domain. Each convolutional layer consists of a set of convolution kernels. The size of 2D convolution kernels could be n $\times$ m, where n is the width of kernels and m is the height of kernels. Normally, the n and m are equal in covolutional layers. There are two other significant parameters: *c_in* and *c_out*. *c_in* is the channel of input images/features and *c_out* is the output features/images channel number. Therefore, the n $\times$ m is the perception field of each kernel, *c_out* is the amount of the kernels used in the layer.

In CNNs, convolutional layers cascade with each other generally, which means the output channel number (*c_out*) of current layer should match for the input channel number (*c_in*) of succeeding layer.

During the forward propagation, each kernel slides (convolves) over the input features along width and height directions to output a 2D feature map. The value of each position in output features is computed by the dot products between the kernel and the corresponding perception field in input features. Each convolutional layer will contain multiple kernels that have the ability to extract diverse visual features.

There are two unique concepts in convolutional layers: padding and stride.

The stride is the step of sliding the kernel each time. The default convolutional stride is 1 which means sliding the kernel one pixel each time. The another common stride is 2, with which the kernel slides 2 pixels each time in height or width direction. The stride

could be any positive integer, though stride larger than 2 is uncommon in practice. Figure 3.1 shows convolution with the stride 1 and stride 2.

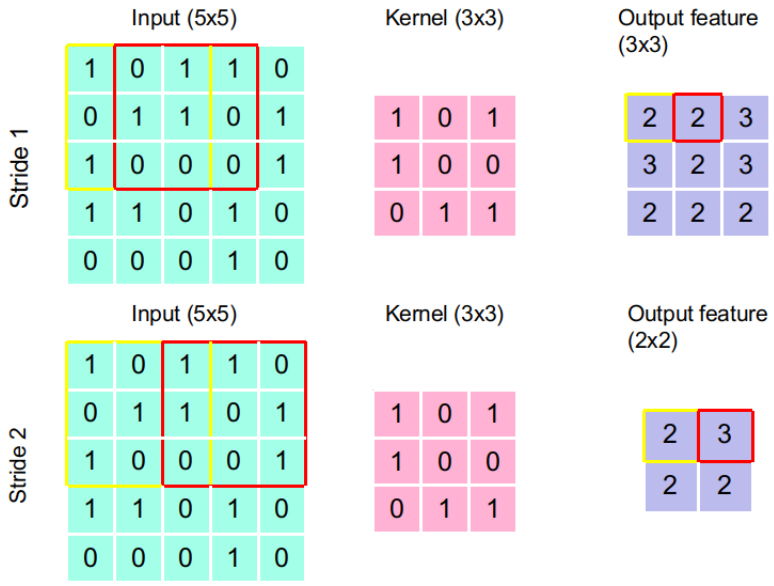


Figure 3.1: Convolution with stride 1 and 2.

According to the above formula, the output size of each layer is smaller than input size even using stride 1. With the increasing number of convolutional layers, the output size of layers shrunk constantly. To avoid this problem, padding is introduced into convolutional layers. Padding in convolutional layers means padding the input features/images with zeros around the border. Figure 3.2 shows the input feature map and the padded input feature when the size of zero padding is 1. The spatial size of output can be controlled by changing zero-padding size. Commonly, we would like to keep the spatial size of the input and output the same. The following formula shows how to calculate the output size by given input size, kernel size, stride and padding size.

$$O = (N - F + 2P)/S + 1 \tag{3.2}$$

where O is the output size, N is the input size, F is the kernel size, P is the padding size and S is the stride.

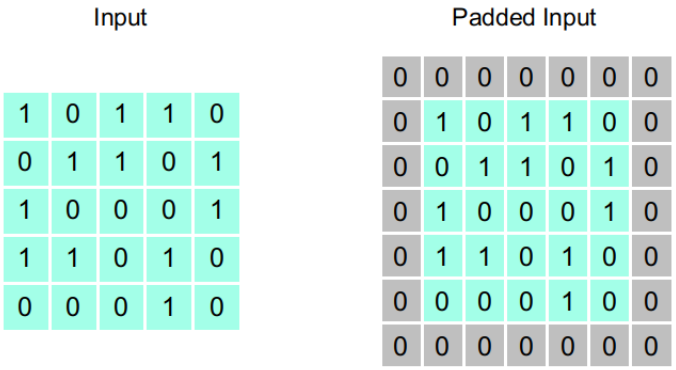


Figure 3.2: Zero-padding.

Figure 3.3 shows a pretty simple example of the 2D convolution operation with stride 1 and no padding.

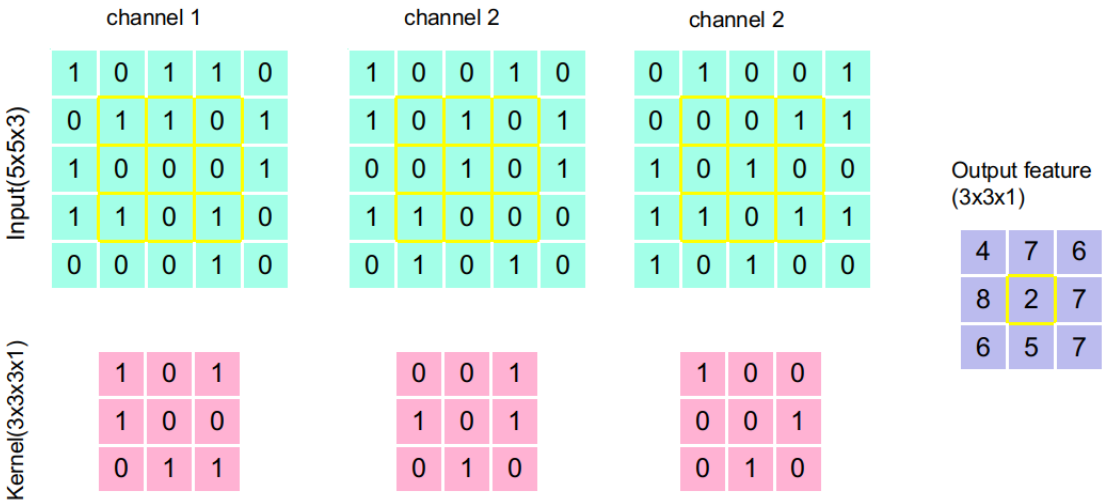


Figure 3.3: 2D convolution operation example.

3 Dimensional Convolutional Layer

The 3D convolutional networks were proposed by Ji et al. [12] to solve action recognition tasks. The 3D CNN is extended from 2D CNN. 2D convolutional layer processes convolution operation in spatial domain, thus 2D CNNs only have ability to capture spatial information rather than temporal information. The 3D convolutional layers add convolution operation in temporal domain in order to capture temporal information of a set of consecutive images (video). Figure 3.4 shows the principle of 3D convolution operation in temporal domain. Therefore, The size of 3D convolution kernels could be n × m × t, where n and m are the width and height of spatial perception field, and t is the input length of temporal convolution for each convolution operation. *c_in* and *c_out* have the same meaning as 2D convolutional layers. Several papers have proved that the 3D convolution is more powerful in video processing than 2D convolution. Figure 3.5 compares the difference between 2D convolution and 3D convolution.

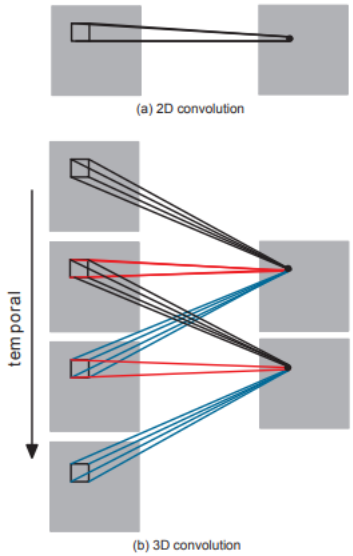


Figure 3.4: 3D convolution operations (adopted from paper [12]).

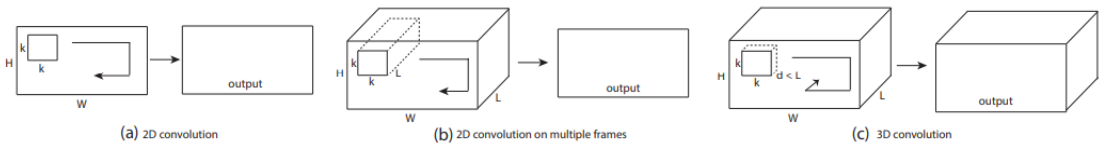


Figure 3.5: 2D vs. 3D convolution operations. a) Applying 2D convolution on an image results in an image. b) Applying 2D convolution on a video volume (multiple frames as multiple channels) also results in an image. c) Applying 3D convolution on a video volume results in another volume, preserving temporal information of the input signal (adopted from paper [13]).

3.1.2 Activation Layer

Usually, a convolutional layer is followed by an activation layer (nonlinear layer), which aims to introduce nonlinearity to the network. The convolution operation is a linear operation. Without any nonlinearity introduced, the network would remain a linear function, which will lead to a poor network computation ability. With the introduction of activation layer, the final function of the network is nonlinear, which ensures that the network could approximate complex functions. The functions utilized in activation layers are termed as activation functions. There are several commonly used activation functions: Sigmoid, Tanh, ReLU, Leaky ReLU and exponential linear unit (ELU).

Sigmoid The Sigmoid function is an S shape function (shown in Figure 3.6), which converts any real-valued input number into a number range from 0 to 1. The sigmoid function has frequent usage in the beginning of neural networks development since it has a nice imitation of the working principle of a neuron. The degree of activation of neurons

range from not activated at all (0) to completely activated at a maximum value (1). The formula of Sigmoid is

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

Recent years, the sigmoid function has been gradually abandoned in neural networks. It has two major disadvantages: Activation saturation problem and the problem that the outputs are not zero-centered.

The activation saturation problem occurs when the the absolute value of input is larger than 5, the output of the neuron is saturated which results in that the gradient at these regions is almost zero. This problem could lead to gradient vanishing problem which means the network update weights (in CNN is the value of convolutional kernels) pretty slow in learning process.

Another problem is that the output of Sigmoid function is not zero-centered. The output of Sigmoid function is always positive. Thus, the input of succeeding layers is always positive, then the gradients of weights in these layers will become either all positive or negative. This could introduce undesirable zig-zagging dynamics in the gradient updates for the weights.

Tanh The Tanh function is proposed to solve the second problem of the Sigmoid function. The output of Tanh is zero-centered, which is shown in Figure 3.7. It limits the $(+\infty, -\infty)$ to the range [-1, 1]. The formula of Tanh is shown below:

$$Tanh(x) = \frac{Sinh(x)}{cosh(x)} \tag{3.4}$$

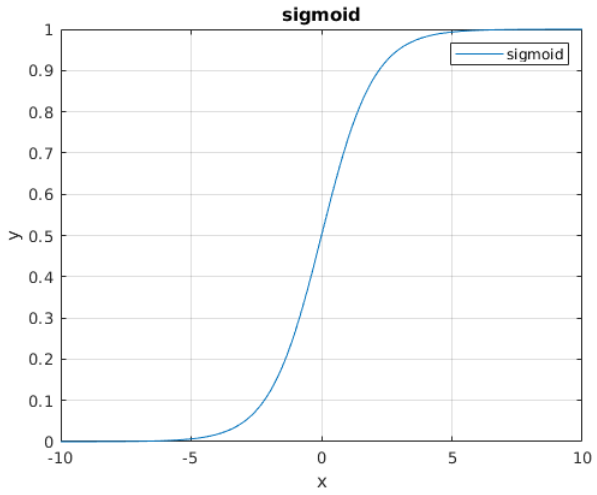


Figure 3.6: Sigmoid Function.

where Sinh(x) and Cosh(x) are defined as:

$$Sinh(x) = \frac{1 - e^{-2x}}{2e^{-x}} \tag{3.5}$$

$$Cosh(x) = \frac{1 + e^{-2x}}{2e^{-x}} \tag{3.6}$$

Then, Equ. (3.4) can be written as

$$Tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{3.7}$$

According to Equ. (3.3), Equ. (3.7) can be written as:

$$Tanh(x) = 2Sigmoid(2x) - 1 \tag{3.8}$$

Actually, the Tanh function is a scaled Sigmoid function. It could solve the none zero-centering problem, though the activation saturation problem remaining.

ReLU The ReLU function has become popular in recent years as it solves the activation saturation problem in Sigmoid function properly. It defined as below:

$$f(x) = max(0, x) \tag{3.9}$$

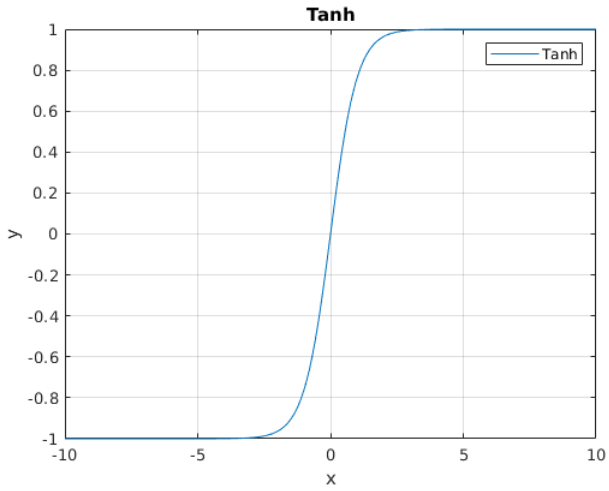


Figure 3.7: Tanh Function.

Figure 3.8 shows ReLU function. Even though the positive part is linear, the ReLU function is nonlinear in nature. There are several advantages and disadvantages:

- Krizhevsky et al. [14] found that ReLU activation function could accelerate the convergence speed of the stochastic gradient descent method which is the algorithm used for training neural networks. The speed of convergence with using ReLU is six times faster than the speed with Sigmoid and Tanh functions.

- The formula of ReLU is simple compared to Sigmoid and Tanh functions. Thus, the implementation of ReLU is much easier than Sigmoid and Tanh that involve expensive operations.

- However, ReLU can result in neurons completely inactivated during training. For a neuron, when the input is a negative number, then the output is 0 if the activation function is ReLU, which will result in the inactivation of previous neurons according to the chain rule. The performance of networks would be terrible if high percentage of

network neurons keep inactivated during the training process. Therefore, the learning rate should be cautiously selected to avoid massive neurons from being inactivated.

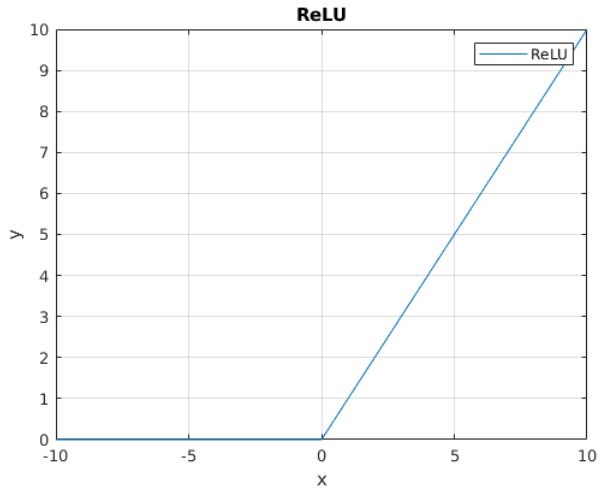


Figure 3.8: Relu Function.

PReLU and ELU Parametric rectified linear unit (PReLU) proposed by He et al.[47] is one of the improved methods based on ReLU to solve the problem that ReLU produces unactivated neurons. The output of PReLU is the product of the input and a small positive number when input is smaller than 0. The formula of PReLU is:

$$PReLU = \begin{cases} x & x \geq 0 \\ \alpha * x & x < 0 \end{cases} \tag{3.10}$$

where α is a small positive constant, which is also a parameter need to be selected properly. Figure 3.9 shows PReLU when α equal to 0.1. Theoretically, PRelu could avoid producing unactivated neurons effectively. However, multiple experiments have proved that the PReLU cannot outperform the ReLU in all situations.

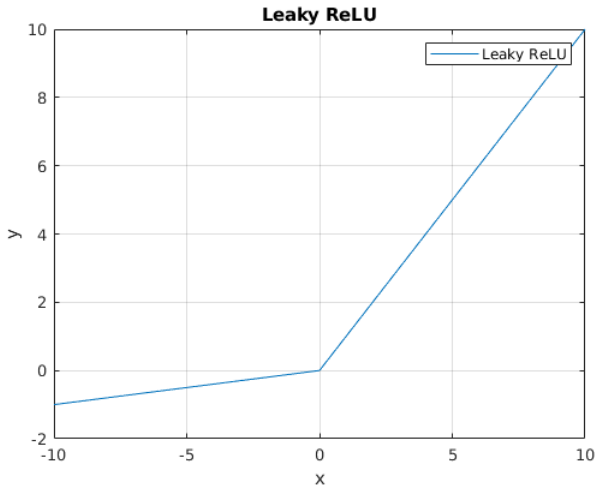


Figure 3.9: PReLU Function.

The ELU [48] is another method attempting to improve ReLU function. Different from PReLU, when inputs are smaller than 0, the ELU uses an exponential expression. The formula of ELU is:

$$ELU = \begin{cases} x & x \geq 0 \\ \alpha * (e^x - 1) & x < 0 \end{cases} \tag{3.11}$$

where α is a small positive constant, which is also a parameter need to be selected properly. The exponential function ensures that the mean of outputs of ELU is closer to zero. It is proved that ELU could help networks obtain higher classification accuracy than ReLU. Figure 3.10 shows the ELU when α is equal to 0.5.

3.1.3 Pooling Layer

It is common to insert a pooling layer between successive activation layers in a convolutional network. The output of activation layers is fed into a pooling layer. The function of pooling layers is down-sampling. With the increasing number of pooling layers, the spatial size of

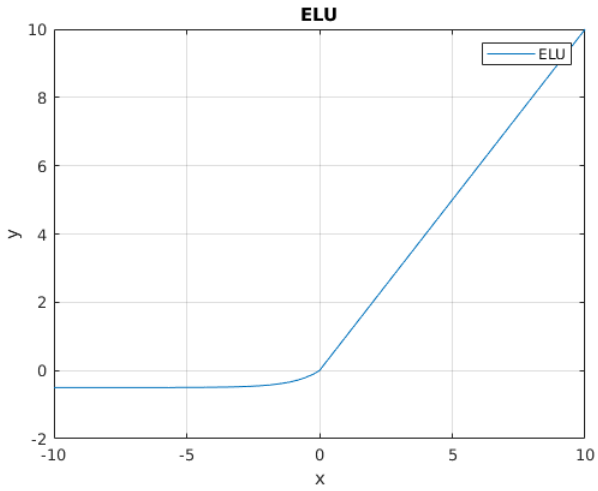


Figure 3.10: ELUs Function.

the feature maps decreases gradually. Pooling layers can help to reduce the number of training parameters and avoid overfitting.

The pooling operation is similar to the convolution operation. The pooling is applied on one patch at a time, then slide the pooling kernel over the input feature maps along width and height directions. There are two types of pooling layers: max pooling layer and average pooling layer. Literally, the max pooling layers preserve the largest value from each patch of the input feature map and discard the rest of the information in that patch. The average pooling layer generates the average value in each patch and discard others.

The most common form of utilizing a pooling layer is that the kernel size is 2×2 and stride is 2. In this case, the pooling operation would take a single value over 4 values. Figure 3.11 shows the max pooling and mean pooling with an input size of 4×4, a kernel size of 2×2 and a stride of 2. In general, the output size of pooling layer can be computed as:

$$O = (N - F)/S + 1 \tag{3.12}$$

where O is the output size, N is the size of input, F is the size of pooling kernel and S is the stride.

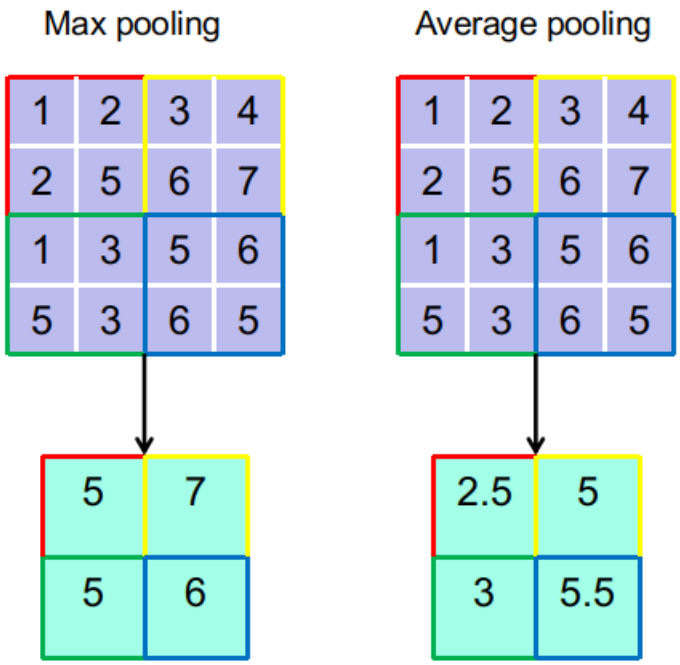


Figure 3.11: Max pooling vs. average pooling.

However, there is a problem in pooling layers. Only one value in each patch is preserved, the majority of information in a patch is lost after pooling layers. Moreover, the pooling layer introduces zero learnable parameters since it operates a fixed function for every input.

Therefore, Springenberg et al. [49] proposed an all convolutional net to overcome the problem. They found that pooling layer can simply be replaced by a convolutional layer with increased stride. This method introduces the flexibility of down-sampling method by replacing the fixed pooling function with the learnable kernels in convolutional layers, which is beneficial for the computation of networks.

3.2 Training Methods

The training process of neural networks is the process of obtaining the optimal set of parameters in networks.

3.2.1 Initialization

There are several initialization methods, some of them are basic initialization methods while others are sophisticated techniques that help CNNs achieve better performance.

Zero initialization Literally, zero initialization is initializing parameters to zero. However, we generally initialize biases to 0 but not weights. When all the weights are initialized to 0, the result of every input is the same, thus the loss is the same for each input. With the constant loss, all weights keep the same value in the whole training process, which means the networks actually learn nothing.

Random initialization At the early stage of development of neural networks, weights are initialized with random numbers. However, the selection of weights is important. When weights are initialized to large values, the output of convolutional layers are also high. The updating steps are large, which will result in unstable training. When the weights are initialized to pretty small values, the updating steps will be too tiny to achieve the optimization rapidly.

Xavier initialization Xavier initialization [50] is proposed to combine with Tanh activation function. By initializing with Xavier algorithm, the distribution of weights in certain

layer is a normal distribution with zero mean and standard deviation is defined as:

$$STD_{Xavier} = \sqrt{\frac{2}{n_{in} + n_{out}}} \tag{3.13}$$

where STD_{Xavier} is the standard deviation, n_{in} and n_{out} are respectively the number of input and output channels of the layer.

The Xavier initialization algorithm ensures that the initial weights are neither too much bigger than 1, nor too much less than 1. Proper initialization methods help to avoid gradients vanishing and exploding problem, also help to accelerate the training speed of networks.

He initialization He initialization [47] is similar to Xavier initialization, and He initialization is more suitable for networks with ReLU activation function. By initializing with He algorithm, the distribution of weights in a certain layer is a normal distribution with zero mean and the standard deviation is defined as:

$$STD_{He} = \sqrt{\frac{2}{n_{in}}} \tag{3.14}$$

where STD_{He} is the standard deviation, n_{in} is the number of input channels of the layer.

3.2.2 Forward Propagation

Forward-propagation is the process of inferring output according to the input and networks. The input data pass through the network layers and the output of the network is obtained straightforwardly. The final output can be represented as:

$$F(x) = f_n(f_{n-1}...f_i(...f_1(x))) \tag{3.15}$$

where x is the input of networks, the f_i is function of each network layer.

3.2.3 Loss Function

After obtaining output of networks through forward propagation, the output needs to be evaluated by a certain standard which is loss function in machine learning. Therefore, the loss function is a performance metric on how close the output of the network and the desired values are. The output of loss function is a positive number. The smaller loss value represents the better performance of network. There are three commonly used loss functions: L1 loss, L2 loss and cross entropy.

L1 loss. L1 loss is the mean of absolute errors, which is defined as:

$$L1 = \frac{1}{N}\sum_{i=1}^{N}|y_i - F(x_i)| \tag{3.16}$$

where x_i is the ith input data, N is the amount of input data, y_i is the corresponding expected output and F(x_i) is the output of network.

L2 loss. L2 loss is mean squared error, which is defined as:

$$L2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - F(x_i))^2 \tag{3.17}$$

where x_i is the ith input data, N is the amount of input data, y_i is the corresponding expected output and F(x_i) is the output of network.

Cross entropy. Cross-entropy is commonly used to quantify the divergence (difference) between two probability distributions. It usually used in classification tasks. It is defined

as:

$$CrossEntropy = \sum_{i=1}^{N} p(x_i) log(q(x_i)) \tag{3.18}$$

where p(x) and q(x) are the true and predictive probability distributions respectively. x_i is the ith input data.

3.2.4 Backward Propagation

Since Rumelhart et al. [51] proved that backward propagation enables faster neural network learning rate than previous methods, the backward propagation algorithm became the mainstream algorithm of learning in neural networks. Backward propagation means that the errors which are calculated by loss function are propagated through the network in backward direction to alter the parameters of networks.

The backward propagation algorithm utilizes gradients of parameters to alter these parameters. The gradient is composed of a set of partial derivatives with respect to variables related to the function, and each value of element represents the greatest increase/decrease rate of the function with respect to the corresponding variable.

The derivations of the back-propagation algorithm are calculated based on the chain rule of calculus. Chain rule is a method help to find the derivatives of variables in the nested equations. For example, if a nested function is defined as $F(x) = f_2(f_1(x))$, then the derivative of x is calculated by:

$$\frac{dF}{dx} = \frac{df_2}{df_1} \cdot \frac{df_1}{x} \tag{3.19}$$

The calculation of derivative of loss with respect to any weight in the network is easy

according to the chain rule. Supposing E is the loss, s_j^k is the output of convolutional layer of node j in layer k, a_j^k is output of activation layer of node j in layer k and w_{ij}^k is the weight of the link between the node i in layer k-1 and node j in layer k, then the partial derivative of loss with respect to w_{ij}^k is calculated by:

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial a_j^k} \cdot \frac{\partial a_j^k}{\partial s_j^k} \cdot \frac{\partial s_j^k}{w_{ij}^k} \tag{3.20}$$

With the backward propagation, the partial derivative of loss with respect to any weight in the network could be calculated easily. With these partial derivatives, the next step is to update corresponding weights.

3.2.5 Optimization

Optimizing a neural network means minimizing the loss of the neural network. Thus, the optimization of a network is to find a set of weights that ensure that the loss of the network reaches the lowest value. Optimization algorithms in CNNs are helping us to minimize the loss effectively. There are several useful optimization algorithms. We explicate stochastic gradient descent (SGD), SGD with momentum, adaptive subgradient method (Adagrad), adaptive learning rate method (AdaDelta) and adaptive moment estimation (Adam) in the following paragraphs.

Gradient Descent is the foundation of optimization algorithms in neural networks. When utilizing gradient descent to update parameters in neural networks, the speed of minimizing the loss of networks could be the fastest. The formula of gradient descent is defined as:

$$\Theta = \Theta - \eta \cdot \bigtriangledown J(\Theta) \tag{3.21}$$

where Θ are the parameters, η is the learning rate and $\bigtriangledown J(\Theta)$ are the Gradient of loss $J(\theta)$ with respect to parameter Θ.

SGD. SGD update parameters for each training data. It is faster than gradient descent, and is defined as:

$$\Theta = \Theta - \eta \cdot \bigtriangledown J(\Theta; x_i, y_i) \tag{3.22}$$

where x_i is one of the training input and y_i is the corresponding desired output.

The parameters will be updated for each training data. Though frequent updating could help discover better local minima, it also results in fluctuation during optimizing process.

SGD with momentum. A technique called momentum is introduced to SGD in order to accelerate and stabilize the convergence of networks. SGD with momentum [52] exploits the previous update to calculate the current update. The formula of each update is defined as:

$$\bigtriangleup \Theta_t = \alpha \cdot \bigtriangleup \Theta_{t-1} + \eta \cdot \bigtriangledown J(\Theta) \tag{3.23}$$

where α is a fraction and named as momentum term, η is the learning rate and $\bigtriangledown J(\Theta)$ are the Gradient of loss.

Thus, the parameter update formula is $\Theta = \Theta - V(t)$. However, SGD with momentum still has a problem. At the end of training process, the network almost reach to the minima, the momentum remains high value and the training speed is relatively too fast. At that point, we would like to slow down the training speed of approaching the minima in order to

avoid missing the minima. Therefore, the adaptive learning rate optimization algorithms are proposed.

Adagrad. Adagrad [53] is one of the adaptive learning rate optimization algorithms. In the previous optimization algorithms, all network parameters are updated at the same time as all parameters have the same learning rate. In Adagrad, each parameter at each time step has a different learning rate. the parameter updating formula at time step t is:

$$\Theta_{i,t+1} = \Theta_{i,t} - \frac{\eta}{\sqrt{G_{ii,t} + \epsilon}} \cdot g_{i,t} \tag{3.24}$$

where η is the initial learning rate, $g_{i,t}$ is the gradient of Θ_i at time step t and ϵ is a small number used to prevent division by 0. Also $G_{ii,t}$ is a function of previous gradients with respect to Θ_i:

$$G_{ii,t} = \sum_{\tau=1}^{t} g_{i,\tau}^2 \tag{3.25}$$

Though the learning rate is different for every parameter in Adagrad, it still has a problem that the learning rate is always decaying along with training process. Then Adadelta [54] is proposed to solve the problem.

Adadelta. Instead of utilizing the summation of all past square gradients directly, Adadelta calculates the sum recursively. the gradient sum $G_{ii,t}$ at time step t is defined as:

$$G_{ii,t} = \alpha \cdot G_{ii,t-1} + (1 - \alpha) \cdot g_{i,t}^2 \tag{3.26}$$

where α is the momentum, it controls the percentage of previous gradients are being used. $g_{i,t}$ is the gradient of Θ_i at time step t. After obtaining the gradient sum $G_{ii,t}$, the parameter update formula is the same as Equ. (3.24).

Adam. The previous optimization algorithms calculate different learning rate for different parameter, also the fixed momentum is introduced into these algorithms. Adam [55] makes a further improvement. It computes adaptive learning rate and adaptive momentum value for each parameter. The formula for the parameter updating is defined as:

$$\Theta_{t+1} = \Theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{3.27}$$

where ϵ is a small number used to prevent division by 0. $\hat{m}_t$ and $\hat{v}_t$ are defined as:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.28}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.29}$$

where m_t and v_t are values of the first moment (mean) and the second moment (variance) of the gradients respectively. β_1^t and β_2^t are the decay rate of the first moment and the second moment respectively.

3.3 Typical Convolutional Architectures

A huge amount of convolutional architectures have been proposed since the first successful convolution network has been proposed in 2012. In this section, we make an inventory of the most typical and powerful neural network architectures.

3.3.1 AlexNet

AlexNet [14] is one of the first successful and popular convolution networks that solve image classification task in computer vision. AlexNet was first proposed by Krizhevsky

in Image Large Scale Visual Recognition Challenge (ILSVRC) [56] of 2012, This network significantly outperformed the other methods which using traditional image classification algorithms. The network achieved a top-5 error of 15.3%, more than 10.8 percentage points lower than that of the runner up. Figure 3.12 illustrates the network. As shown in the figure, it contains 5 convolutional layers to extract features, 3 max pooling layers to reduce the number of training parameters and 3 fully-connected layers which are usually used for classification tasks and not suitable for generation tasks. The contribution is huge as bringing deep neural networks out to the public.

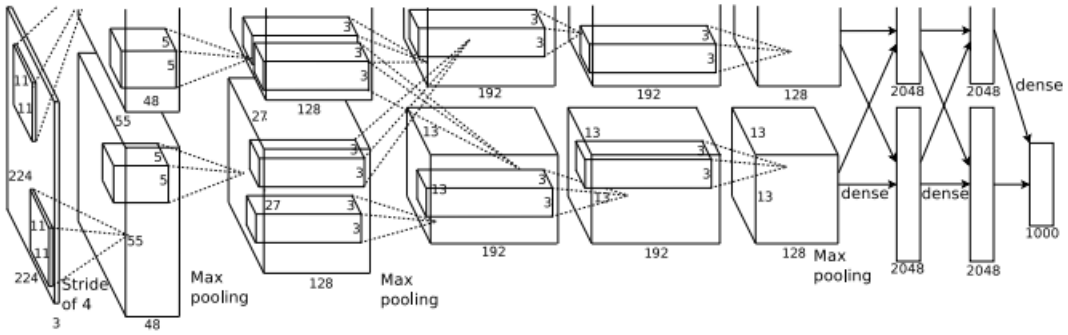


Figure 3.12: An illustration of the architecture of AlexNet (adopted from the original paper [14]).

3.3.2 VGGnet

VGGnet is proposed by Simonyan et al. [33] in ILSVRC [56] of 2014. There are two branches, VGG-19 and VGG-16. These two networks won the first and second place of the competition. The 19 and 16 represent the number of layers of learnable parameters in the networks. The architecture of VGG-16 is illustrated in Figure 3.13, which contains 13 convolutional layers for extracting features, 5 max pooling layers to reduce the number of

training parameters and the 3 fully-connected layers for classification. The contributions of the work are:

First, the VGGnet use 3 × 3 convolutional kernels stacked on top of each other can replace large sized convolutional kernel. Moreover, these stacked 3 × 3 kernels are more effective than large sized convolutional kernel.

Second, it proved that increasing convolutional depth is more powerful than increasing convolutional width. This encourages scientists to build deeper neural networks.

Finally, it proved that using pre-training in small neural networks, which can help to train larger neural networks.

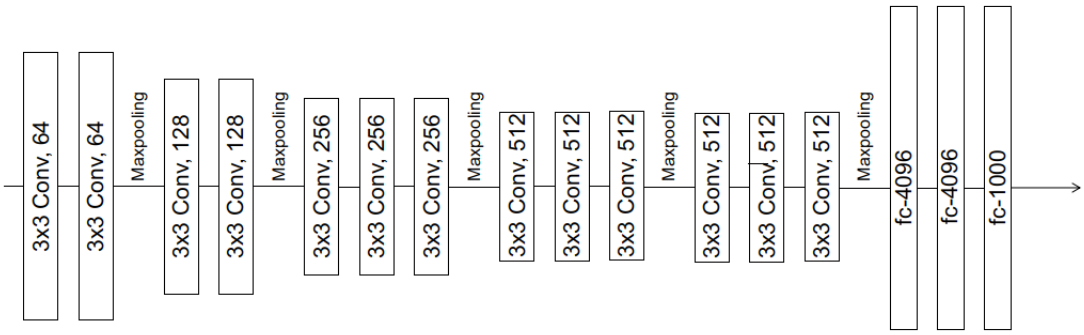


Figure 3.13: An illustration of the architecture of VGG-16.

3.3.3 ResNet

In traditional neural networks, the output of each layer feeds into the following layer directly. In principle, deeper network should have better performance. However, the traditional networks with larger number of layers usually have worse perforamce than their shallower counterparts, which is irrational. The reason is that with the increasing number

of layers, the gradient vanishing problems occur, which is harmful for the performance of networks. In order to solve the problem, He et al. [34] proposed Residual network, which consists of several residual blocks. In these blocks, they proposed to add a skip connection between input of block and output of block to preserve the gradients to avoid gradient vanishing problem and skip several extra layers to avoid overfitting. Figure 3.14 illustrates the architecture of residual block, which stacks convolution layer, BN layer and activation layer. BN layer further helps to avoid gradient vanishing and explosion problems.

The ResNet achieves outstanding performance and is widely adopted by deep learning networks rapidly. The depth of neural networks get impressively increased, some neural networks contain up to hundreds of convolutional layers.

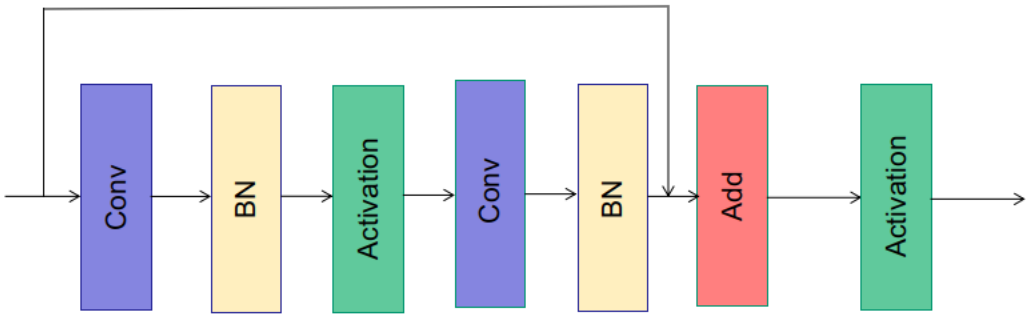


Figure 3.14: The structure of a residual block.

3.3.4 Inception

The Inception network series is proposed by Google, and it contains 4 versions, from v1 [15] to v4. They mentioned a problem: the objects of interest are different for each input image, therefore, the fixed size convolutional kernel is insufficient. They proposed a network that can choose the right kernel size automatically. Each layer of Inception network contains

multiple convolutional kernels of different sizes, which can capture features of different sizes. The output of each layer is the concatenation of the outputs of these kernels. Figure 3.15 shows the naive version of Inception v1.

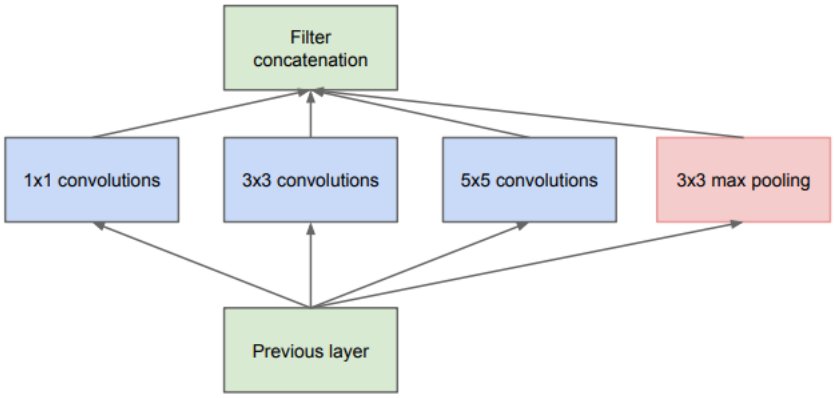


Figure 3.15: The structure of naive Inception v1 (adopted from the original paper [15]).

In the v2 version [36], they added an extra 1×1 convolution kernel before these convolutional kernels of different sizes to reduce the number of parameters. Moreover, they added BN layers to accelerate the network convergence speed. In the v3 version [57], they factorized the convolutional kernels of large sizes into stacked 3×3 kernels as described in VGG network. In the v4 version [58], they adopted residual learning by adding residual connections between the input and output of each layer.

3.4 Summary

In this chapter, we introduced important concepts of convolution neural network, including different functional layers, network training methods and several typical convolution networks. Our proposed activated separate 3D convolution network is an improved convo-

lution network which is based on 2D and 3D convolutions.

Chapter 4

Video Super Resolution via Wide Activated Separate 3D Convolution

We propose a wide activated separate 3D convolution based neural network for VSR. First, we give the overview of our proposed framework in Sec. 4.1. Then, we illustrate the sub-networks in our method in Sec. 4.2. Finally, we define the loss function used for our training in Sec. 4.3.

4.1 Overview

Suppose I_t^{LR} is the tth LR video frame, and the resolution of I_t^{LR} is $H \times W \times C$, where H, W and C are the height, width and number of channels of the frame, respectively. The resolution of the corresponding HR video frame I_t^{HR} is $sH \times sW \times C$, where s is the up-sampling scale. Video super-resolution methods aim to recover the high quality

HR video frame I_t^{HR} from the reference LR frame I_t^{LR} and $2N$ supporting frames { I_{t-N}^{LR}, I_{t-N+1}^{LR},...,I_{t+N-1}^{LR}, I_{t+N}^{LR} }.

The structure of our proposed method is illustrated in Figure 4.1. It is an end-to-end neural network that consists of four main sub-networks: temporal feature extraction, temporal feature fusion, high-level feature extraction and HR frame reconstruction.

The temporal feature extraction sub-network aims to extract temporal and low-level spatial information of the input reference (center) LR frame and the supporting (surrounding) LR frames. It contains 5 wide activated separate 3D convolutional residual blocks. The temporal feature fusion sub-network aims to fuse these extracted temporal features. It is a 2D convolutional layer. The high-level features extraction sub-network is to capture the high-level missing components of the LR frames compared to the HR frame, which consists of 22 wide activated 2D convolutional residual blocks. The HR frame reconstruction layer is to reconstruct the HR frame by using the LR feature maps, we adopt the sub-pixel convolution layer proposed by Shi et al. [2]. Besides, we also adopt residual learning, by adding the bicubically up-sampled reference LR frame with the output of our network directly to help recover the HR frame better.

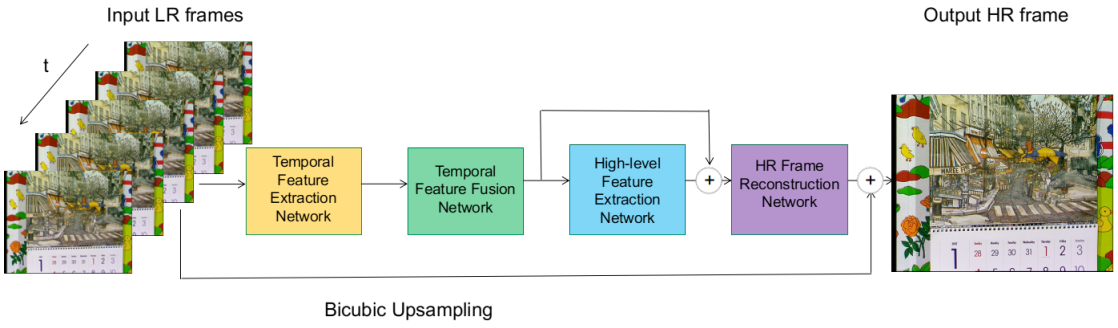


Figure 4.1: The proposed wide activated separate 3D convolution network for VSR.

4.2 Architecture of Model

4.2.1 Wide Activated Separate 3D Convolutional Residual Block

Separate 3D Convolutional Layer

We introduced the traditional 3D convolution in Sec. 3.1. However, the 3D convolution has several problems:

Firstly, the computation is expensive. Because 3D convolutional layers operate convolution not only in the horizontal and vertically direction but also in the temporal domain. The 3D convolutional kernel size is $n \times n \times t$, where $n \times n$ is the size of convolutional kernel in spatial domain and t is the size of convolutional kernel in temporal domain. Therefore, the parameter number of the 3D convolutional kernel is t times that of the 2D convolutional kernel. With more trainable parameters, the network is more complex and the computation is more expensive. Therefore, the optimization of the network is harder to reach.

Secondly, it is difficult to build a very deep 3D convolutional network because of the massive parameters of each 3D convolutioanl layer. The depth of a network has been proven much important in deep neural networks. As a result, 3D CNNs has shown worse performance than some deeper 2D CNNs in several tasks, though the parameter size of the 3D CNN model is equal or even larger than that of the 2D CNN model.

Finally, the 3D CNNs cannot be fine-tuned by a pre-trained model. All 3D CNNs must be trained from scratch, which makes 3D CNNs lose the power of transfer learning and pre-training. Therefore, the training of 3D CNNs is more difficult than that of 2D CNNs.

To solve the above mentioned problems, the separate 3D convolution [59] is an improved 3D convolution model. The separate 3D convolution decomposes a 3D convolution into a 2D spatial convolution and a 1D temporal convolution. There are 3 types of connection methods: 2D spatial convolution followed by 1D temporal convolution, 2D spatial convolution followed by 1D temporal convolution with skip connection, and 2D spatial convolution and 1D temporal convolution are connected in parallel. Our work is inspired by the separate 3D convolution structure proposed by Tran et al. [60], which connect the 2D spatial convolution and 1D temporal convolution in series. Figure 4.2 shows the structure of the 3D convolution and the separate 3D convolution. When the size of the 3D convolution is $n \times n \times t$, the size of the 2D spatial and the 1D temporal convolutional kernel of the separate 3D convolution are $n \times n \times 1$ and $1 \times 1 \times t$, respectively.

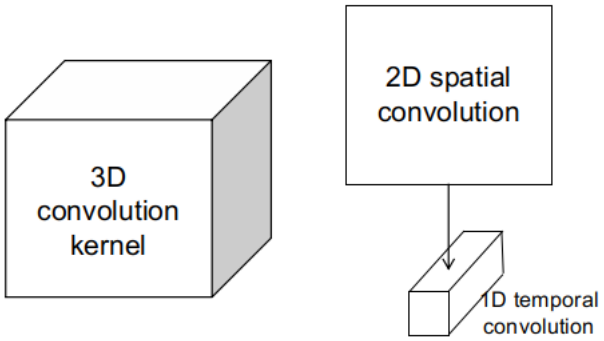
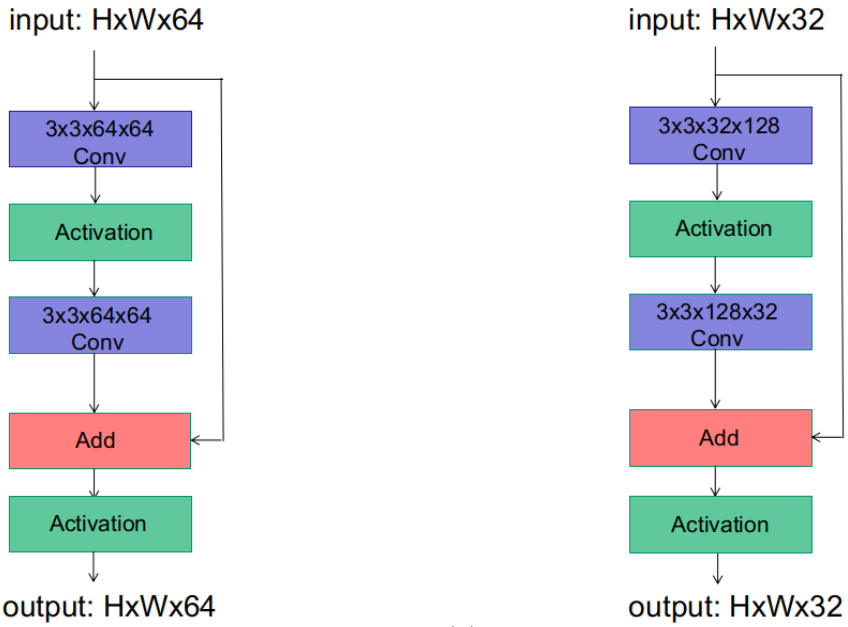


Figure 4.2: 3D convolution and separate 3D convolution.

Wide 2D Convolutional Residual Blocks

As introduced in Sec. 3.3.3, residual blocks could solve the gradient vanishing problem and work well in several computer vision tasks. However, the original residual blocks are not suitable for generative tasks. Therefore, several improved residual blocks are proposed

to suit for generation tasks. We adopt an improved 2D convolutional residual blocks, wide 2D convolution residual block [61] in our model. The wide 2D convolution residual block has wider intermediate feature maps and narrower output feature maps while the original residual block has the same channel size from input to output of the block. Figure 4.3 shows the comparison of the original residual block and the wide activated residual block. These two blocks have the same number of parameters. For the original residual block, the input, intermediate and output layers have the same number of channels. And for the wide activated residual block, the intermediate layer has larger number of channels than the input and output layers. It has been proved that the performance of wide 2D residual blocks is better than the original residual blocks though the wide activated blocks and original blocks have the same parameter complexity [62].



(a) Original 2D convolution residual block

(b) Wide activated 2D convolution residual block

Figure 4.3: Comparison between original and wide activated 2D convolution residual blocks.

Wide 3D Convolutional Residual Blocks. Inspired by the wide activated residual blocks and separate 3D convolution, we propose a novel wide activated separate 3D residual block. Firstly, we eliminate the unnecessary activation layer of the separate 3D convolution residual blocks, then combine the wide activation theory with the separate 3D convolution. Our proposed model has wider intermediate layer which is generated by 2D convolution along spatial domain and narrower output which is generated by 1D convolution along temporal domain. Our model has wider activated layer than the original 3D convolution. Figure 4.4 shows an example of the architecture of our proposed blocks. To establish the temporal feature extraction sub-network, we stack 5 of these wide activated separate 3D residual blocks to extract abundant temporal features from the input reference LR frame and supporting frames effectively. In each block, the 2D convolutional layer in spatial domain has wider output channel and the 1D convolutional layer in temporal domain has narrower output channel.

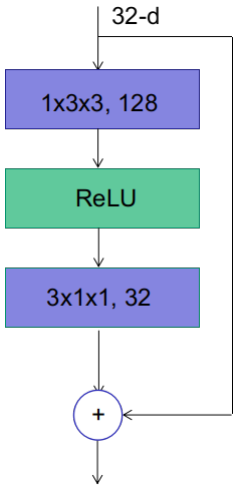


Figure 4.4: A residual separate 3D convolution block.

4.2.2 Temporal Features Fusion Layer

We concatenate all the output feature maps of the final 3D blocks along time dimension. That means we convert the feature maps with size $H \times W \times (2N+1) \times C$ to the feature maps with size $H \times W \times (2N+1)C$ where H and W are the height and width of feature maps, (2N+1) is the number of input LR frames and C is the output channel number. Figure 4.5 illustrates the principle of concatenation layer. The concatenated feature maps are followed by a 2D convolution layer to fuse the temporal information of these feature maps.

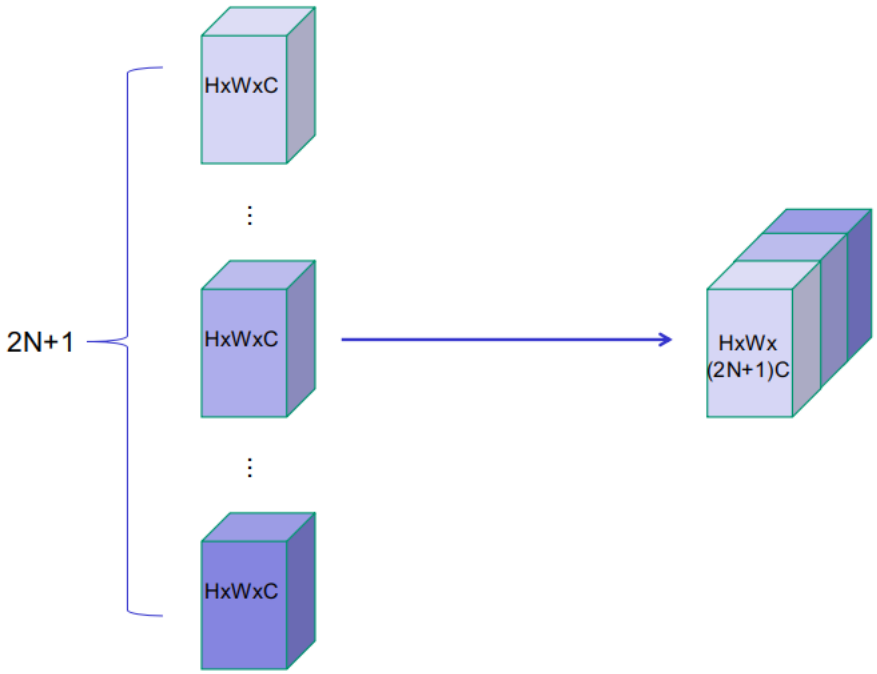


Figure 4.5: Concatenation layer.

4.2.3 High-Level Features Extraction Network

We use a high-level feature extraction network to capture the missing information of the LR frame compared to an HR frame. After obtaining the fused LR features, the high level features could be captured by that network. The network consists of 22 2D residual blocks with wide activation. In these blocks, we use ReLU as activation function. Therefore, the architecture of the sub-network is illustrated in Figure 4.6.

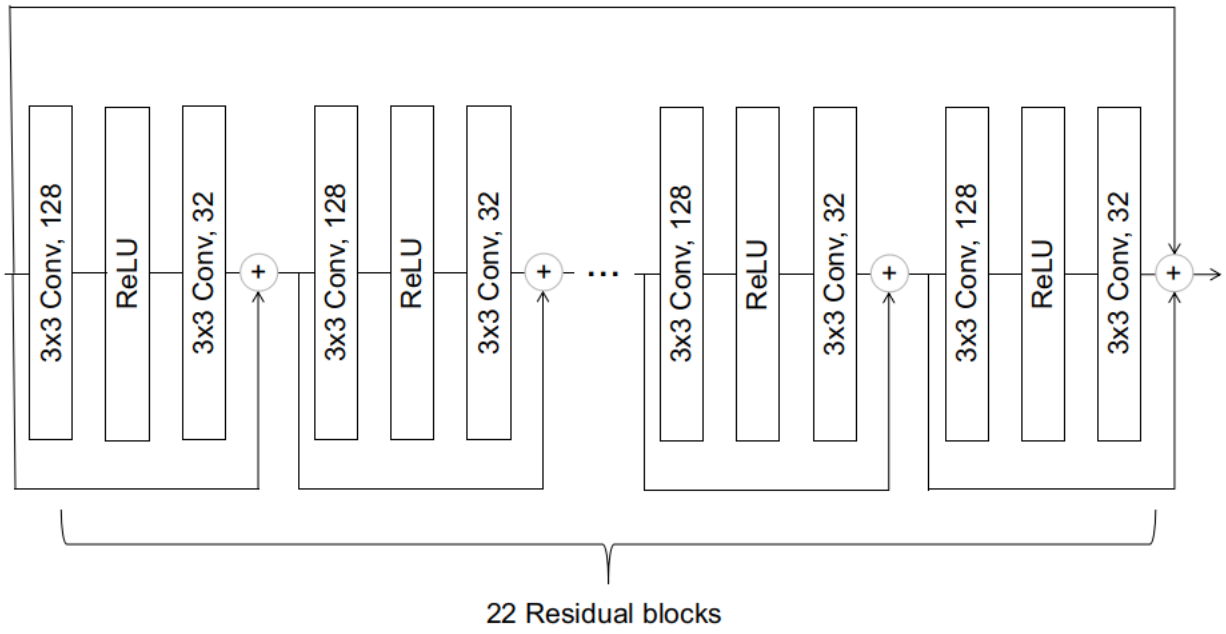


Figure 4.6: The architecture of high-level feature extraction sub-network.

4.2.4 HR Frame Reconstruction Layer

After extracting the high-level features of the LR input frames, we utilize a sub-pixel convolution layer [2] to reconstruct the HR frame from the LR feature maps. Actually, the sub-pixel convolution is a depth-to-space transformation, which arranges the pixels in

multiple LR feature maps as sub-pixels and merges these sub-pixels into the pixels of the HR frame. For up-sampling scale is s, the size of HR frame is $sH \times sW \times c$ when the size of input feature maps is $H \times W \times s^2c$. Figure 4.7 illustrates the HR frame reconstruction layer.

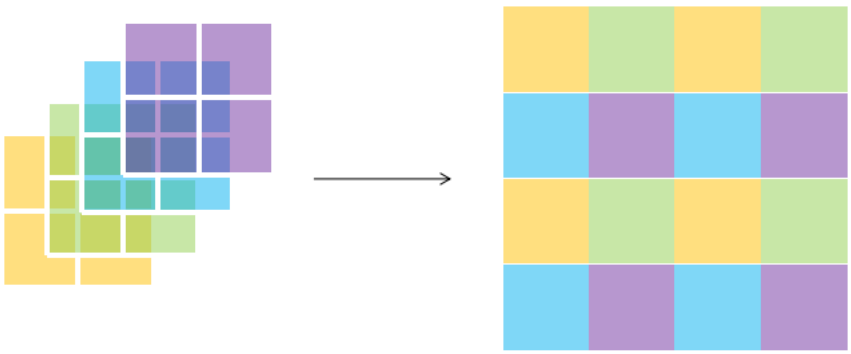


Figure 4.7: An illustration of sub-pixel layer with upsampling scale of 2.

4.2.5 Residual Learning

HR frames usually contain various content, including low and high frequency components. However, in LR frames, the majority of low frequency components is usually preserved and the majority of high frequency components is lost. Therefore, the VSR networks focus on recovering the missing high frequency components. As illustrated in Figure 4.1, we use bicubic interpolation method to up-sample the LR reference frame and add the up-sampled image to the output of the network directly. This operation ensures that the network focus on learning the residual compositions between the ground truth and the up-sampled image, which is beneficial for the network learning. It not only eases the training of network also solves the gradient vanishing and exploding problem.

4.3 Loss Function

We use the spatial MSE as loss function to train our model, which is defined as:

$$L_{sr} = \frac{1}{N}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M}[I_{sr}(i,j) - I_{gt}(i,j)]^2 \tag{4.1}$$

where N and M are the height and width of HR frame, $I_{sr}(i,j)$ is the pixel value of location (i,j) in the reconstructed HR frame and $I_{gt}(i,j)$ is the pixel value of the same location (i,j) in the ground truth HR frame.

The loss function is a function that maps the network evaluation to a real number, representing the average squared difference between the reconstructed pixels values and the actual pixels values. The optimization of the network is the process to minimize the loss value by tuning parameters.

4.4 Summary

In this chapter, we presented our proposed VSR method. The proposed methods based on wide activated separate 3D convolution residual block and wide activated 2D convolution residual block. Comparing to the original activated separate 3D convolution and the original activated 2D convolution, the wide activated separate 3D convolution and wide activated 2D convolution allow introducing more non-linear calculation units into network while have the same numbers of parameters as the original activated counterparts. We also defined our training loss which is used to evaluate the training process.

Chapter 5

Experimental Results and Evaluation

In this chapter, we will introduce our experiments. First, we introduce the environment of our experiments in Sec. 5.1. Then, we introduce the experimental settings in detail in Sec. 5.2. Finally, we compare our proposed method with other state-of-the-art VSR methods in Sec. 5.3.

5.1 Environment Setup

We use a PC with an Intel Core i7-7700K CPU and a NVIDIA Geforce GTX 1080Ti GPU. The PC also contains 16 GB DDR4 system memory to store our training and testing dataset and 11 GB GPU-dedicated memory to store our CNN models. We implement our VSR architecture based on the Tensorflow platform [63]. TensorFlow is an end-to-end open source machine learning platform produced by Google. We deploy our implementations on an operating system of Ubuntu version 18.04.2. NVIDIA proprietary driver and CUDA

toolkit are installed to support GPU accelerated computation.

5.2 Experiment Settings

5.2.1 Datasets

Training Datasets

The training datasets have a great influence on the performance of deep learning based networks. Therefore, a large-scale and content-rich dataset is important for training high-performance networks. For SISR task, there are several publicly available large-scale datasets: a dataset contains 291 images, which is consisting of 91 images from Dong et al. [64] and 200 images from Martin et al. [65]. The other dataset is consisting of images randomly selected from ImageNet [14]. The last dataset is the DIV2K [27], which contains 1000 2K-resolution HR images.

Similarly, a large-scale dataset that contains sufficient and high-quality HR videos is significant in training high-performance VSR networks. However, there is no publicly availabe dataset in VSR task. The previous methods [6, 8, 10, 66] usually collect their own datasets. Jo et al. [66] collected 351 videos from the Internet, including wildlife, activity and landscape contents. Then, they cropped the selected videos to produce the patches of 144 $\times$ 144 resolution. And they also sampled 160,000 ground truth patches as training dataset. Tian et al. [10] used Vimeo Super-Resolution dataset [67] as their training dataset, which contains 64,612 training samples, each of which contains seven consecutive frames with size of 448 $\times$ 256. Sajjadi et al. [8] produced their high-quality

dataset by selecting 40 high-resolution videos downloaded from vimeo.com. They firstly down-sampled the original videos by a factor of 2 and then extracted patches of size 256 $\times$ 256 to generate the HR videos.

Thanks to Sajjadi et al. [8] for providing the URLs of their used videos on their website, even though some videos are not accessible right now. The rest of videos are sufficient for us to generate our large-scale video dataset. We collected 31 videos from the video list on the website of Sajjadi. 16 of them are 4K videos, 9 of them are 1080P videos and the rest are 720P videos. We down-sampled all the 31 videos by a factor of 2 and split them into fragments that do not include sudden scene changes by using FFmpeg. We extracted 1898 fragments in these videos and cropped them into patches with size of 384 $\times$ 384 resolution randomly to generate ground truth HR fragments. For each cropped fragments, we have to make sure that the fragments conatain sufficient amount of motion. The first 30 consecutive frames of each fragment are our training dataset. In total, the training dataset contains 56,940 ground truth HR video frames. Table 5.1 shows the basic information of our training dataset.

Testing Datasets

Almost all the previous papers used the same test dataset VID4 [68]. The dataset consists of four test videos, each of which contains different type of contents:

- *Calendar* has plenty of hand-drawn texture and text content.

- *City* has buildings with rich details of different scales.

Table 5.1: The information about our training dataset.

Information	Training dataset
Ground truth resolution	384×384
Codec	H.264
Frame rate	30
Frames per fragment	30
Total frames	56,940

- *Foliage* has fast-moving vehicles and thin tree branches.

- *Walk* contains a large number of human faces and clothes.

As mentioned in Sec. 4.1, a VSR network usually takes (2N + 1) consecutive LR frames as input frames. The preceding N LR frames and succeeding N LR frames are the supporting frames and the center frame is the reference frame. The ground truth is the HR frame in the center. The input frames are selected by a sliding window whose length is (2N + 1) and moving step length is 1. In order to ensure that each reference LR frame has sufficient number of supporting frames, the first N frames and the last N frames should not be regarded as reference frames. Therefore, we only use $M - 2N$ frames as reference frames when the fragment contains M frames.

The previous methods usually exploit 3, 5 and even 7 consecutive LR frames to reconstruct the HR frame. In our experiments, the size of input LR frames is 5 because the separate 3D convolution convolves along the temporal domain using a convolution kernel

of size 3. Moreover, 5 is the most appropriate size of input LR frames for the trade-off between time complexity and performance. Therefore, the first and last 2 frames of each video are not used as center LR frames.

Down-sampling Methods

In our experiments, we test on the situation where up-sampling scale s is 2, 3 and 4. The HR frames are patches with resolution of 384×384. The LR input frames are patches with resolutions of 192×192, 128×128 and 96×96, according to the different up-sampling scales. We use the *imresize* function in Matlab to generate LR frames. The *imresize* function utilizes bicubic down-sampling method to down-sample HR frames. Moreover, the function has an *antialiasing* model to prevent aliasing when shrinking an image. Also, different Python libraries have different default down-sample methods. OpenCV uses bilinear as the default down-sampling method and Python Imaging Library (PIL) uses nearest neighbour as the default down-sampling method. In scikit-image library of Python, the resize function uses bilinear down-sampling method as the default. The difference between this library and the other two is its *antialiasing* model which is similar to *imresize* function in Matlab while none of the previous two libraries has the *antialiasing* model. We denote this direct down-sampling method as BI in our thesis.

Another method is to blur HR frames by a Gaussian kernel and then down-sample the blurred HR frames by selecting every 4-th pixel in each spatial direction. We denote this down-sampling method as BD in our thesis.

In our experiments, we adopt the first down-sampling method to obtain our LR frames.

Table 5.2: The parameter size of each sub-network components.

Sub-networks	Number of parameters	
Temporal feature Extraction	Separate 3D	$\begin{pmatrix} 1 \times 3 \times 3 \times 1 \times 32 \\ 3 \times 1 \times 1 \times 32 \times 32 \end{pmatrix} \times 1$
	Wide separate 3D	$\begin{pmatrix} 1 \times 3 \times 3 \times 32 \times 128 \\ 3 \times 1 \times 1 \times 128 \times 32 \end{pmatrix} \times 5$
Temporal feature Fusion	2D	$3 \times 3 \times 128 \times 32$
High-level feature Extraction	Wide 2D residual	$\begin{pmatrix} 3 \times 3 \times 32 \times 128 \\ 3 \times 3 \times 128 \times 32 \end{pmatrix} \times 22$
HR reconstruction	2D	$3 \times 3 \times 32 \times scale^2$

For the second down-sample method, almost all previous works did not publish their sizes of Gaussian kernel, so it is hard to compare the results on a fair basis. Therefore, we do not use the second down-sampling method for our experiments.

5.2.2 Training and Parameters

As mentioned, we perform different up-sampling scales to test the efficiency of our method. Therefore, the HR reconstruction sub-network has different number of output channels corresponding to different up-sampling scales. We did a serial of experiments and referred to several previous papers [6, 8, 10, 66] to choose the optimum parameters. We set each training sequence contains 5 consecutive LR frames. We implement our network by Tensorflow

[63] and utilize NVIDIA 1080TI GPU to accelerate training process.

We use Adam as our optimizer, we use the default setting: $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The mini-batch size of training is set to 16, which means there are 16 groups of LR sequences input into the network in one time. The learning rate of training is initialized to 10^{-4} for all layers and decreases half for every 100 epochs.

All the weights are initialized with random normal distributions with standard deviation equal to $1e^{-2}$ and all biases are initialized to 0.

To clarify the parameters of each layer in our proposed network, Table 5.2 shows the parameters in each sub-network of the whole network.

5.2.3 Evaluation Metrics

There are two kinds of image quality evaluation metrics in VSR task: peak-signal-to-noise ratio (PSNR) and structural similarity (SSIM) [69].

The PSNR is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation. The PSNR evaluates the mean squared error between ground truth images and reconstructed HR images, and higher PSNR represents less noise in reconstructed image. The expression of PSNR is defined as:

$$PSNR = 10log_{10}\left(\frac{MAX^2}{MSE}\right) \tag{5.1}$$

where MAX is the maximum value of each pixel in the image, and it is usually equal to 255 for color and gray images. MSE is defined in Equ. (4.1).

The PSNR is an objective evaluation metric, however, it has a bias to over-smoothed images as it evaluates the squared error. It tends to give higher scores to these reconstructed images that remove not only noise but also a part of the high frequency textures.

The SSIM is a method for evaluating the perceived quality of reconstructed frames. It takes the similarity of high frequency components (such as lines, edges) into consideration. Therefore, the reconstructed images have to remove noises and preserve the high frequency components of HR images to get higher SSIM score. The expression of SSIM is defined as:

$$SSIM(x,y) = [l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma}] \tag{5.2}$$

where x and y are two different images, $l(x,y)$, $c(x,y)$, $s(x,y)$ are the three comparison measurements. $l(x,y)$ is the luminance, $c(x,y)$ is the contrast and $s(x,y)$ is the structure measurments. The α, β and γ are the importance factor, and they are usually eauql to 1. The three comparison measurements are defined below:

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{5.3}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{5.4}$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{5.5}$$

where μ_x and μ_y are the average of x and y, σ_x and σ_y are the standard deviation of x and y, σ_{xy} is the cross-covariance of x and y. c_1 and c_2 are two variables to stabilize the division with weak denominator, c_3 is equal to $\frac{c_2}{2}$ by default. Also, c_1 and c_2 are defined as:

$$c_1 = (k_1 L)^2 \tag{5.6}$$

$$c_2 = (k_2 L)^2 \tag{5.7}$$

where L is the maximum value of each pixel in the image, it is usually equal to 255 for color and gray images. k_1 and k_2 are two decimals smaller than 1.

The SSIM is more subjective, it tends to look for images that have high perceptive quality, better quality for human eyes. However, the computation is complex. Therefore, it is always used together with the PSNR to evaluate the quality of reconstruction.

5.2.4 Color Space

For the utilization of color space information of input LR frames, there are two different formats: RGB and YCbCr. In our experiments, we convert images to YCbCr format and only utilize the luminance (Y) information, because humans are more sensitive to brightness changes than color changes, and, the luminance channel contains majority of information of images. As a result, only the luminance (Y) component is used for evaluation.

The mainstream conversion standard of RGB-YCbCr is ITU-R BT.601. The standard is implemented in Matlab and scikit-image library of Python. The YCbCr format images can be obtained by the following equations:

$$Y = 16 + \frac{65.738 \cdot R}{255} + \frac{129.057 \cdot G}{255} + \frac{25.064 \cdot B}{255} \tag{5.8}$$

$$Cb = 128 - \frac{37.945 \cdot R}{255} - \frac{74.494 \cdot G}{255} + \frac{112.439 \cdot B}{255} \tag{5.9}$$

$$Cr = 128 + \frac{112.439 \cdot R}{255} - \frac{94.154 \cdot G}{255} - \frac{18.285 \cdot B}{255} \tag{5.10}$$

Generally, the value of each pixel is range from 0 and 255, where 0 represents the black and 255 represents white. However, the values of Y channel according to the above formula is ranging from 16 to 235. The values below 16 are called footroom and those larger than

235 are called headroom. Both of them are reserved for possible overshooting in analog video equipment.

Another conversion standard is JPEG, which is widely implemented in Python libraries, such as OpenCV, Scipy and PIL. The definition is much different from the ITU-R BT.601 standard. The conversion are defined as:

$$Y = 0 + 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{5.11}$$

$$Cb = 128 - 0.168736 \cdot R - 0.331264 \cdot G + 0.5 \cdot B \tag{5.12}$$

$$Cr = 128 + 0.5 \cdot R - 0.418658 \cdot G - 0.081312 \cdot B \tag{5.13}$$

The numerical values of PSNR and SSIM are different when using different conversion standards to evaluate the same reconstructed HR frame. According to the experimental results of previous methods, they use ITU-R BT.601 as conversion standard and extract the Y channel to calculate the values of PSNR and SSIM.

In our experiments, we only exploit luminance channel for training and testing (JPEG standard), we do not normalize the pixel values as all pixel values in an image have the same scale (range from 0 to 255). However, the same as other methods, we evaluate our method on luminance channel which is produced by ITU-R BT.601 standard.

Table 5.3: PSNR and SSIM results on videos of VID4 with up-sampling scales of 2, 3 and 4.

Scale		Calendar	City	Foliage	Walk
×2	PSNR	28.98	32.45	30.64	33.54
	SSIM	0.94	0.92	0.92	0.95
×3	PSNR	25.59	29.63	27.98	31.17
	SSIM	0.85	0.86	0.85	0.90
×4	PSNR	22.80	27.96	26.39	29.31
	SSIM	0.78	0.81	0.77	0.87

5.3 Experimental Result Comparisons

5.3.1 Experiments Results

In this part, we will show results of our proposed method on VID4. Figure 5.1 shows the MSE loss changes during the training process. The results of our proposed method on four videos of VID4 dataset are shown in Table 5.3

5.3.2 Results Comparison with Other methods

Table 5.4 shows the comparison results of our proposed wide separate 3D convolution for VSR method with other state-of-the-art video super-resolution methods on the VID4 dataset with up-sampling scales of 2, 3 and 4. The first and last two frames are not used in

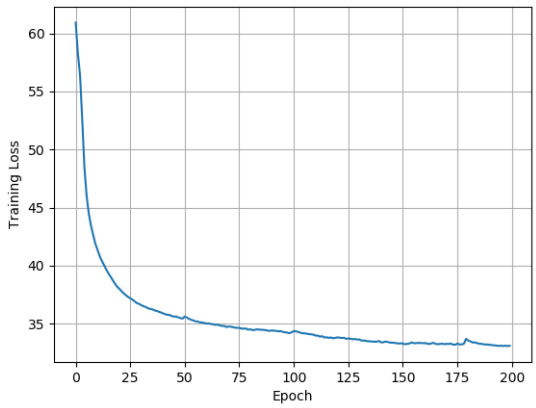


Figure 5.1: The training loss changes along the epoches.

our evaluation. As shown in Table 5.4, our proposed method outperforms the state-of-the-art methods with an average of 0.41 dB in PSNR and 0.013 in SSIM when the up-sampling scale of 4. Also, our method achieves the best result with the up-sampling scales of 2 and 3. The visual comparisons when the up-sampling scale is 4 are shown in Figures 5.2 through 5.5. For Figure 5.2, the text are recovered more clearly by our proposed method. For Figure 5.3, the edges of the windows and detailed textures are recovered more realistically by our proposed method. For Figure 5.4, the hubs of the car are recovered more clearly by our method. For Figure 5.5, the human faces are recovered more realistically by our method.

We also demonstrate the effectiveness of the proposed wide activated separate 3D residual block and the original activated separate 3D residual block. We compare the training loss when train the wide activated method and original activated method with similar parameter sizes and exactly the same training setups. Figure 5.6 shows the loss changes of wide activated separate 3D method and original activated separate 3D method during

Table 5.4: Comparison of PSNR and SSIM on VID4 dataset with up-sampling scales of 2, 3 and 4. The first and last two frames are not used for calculation. Our method achieves the best performance in PSNR and SSIM.

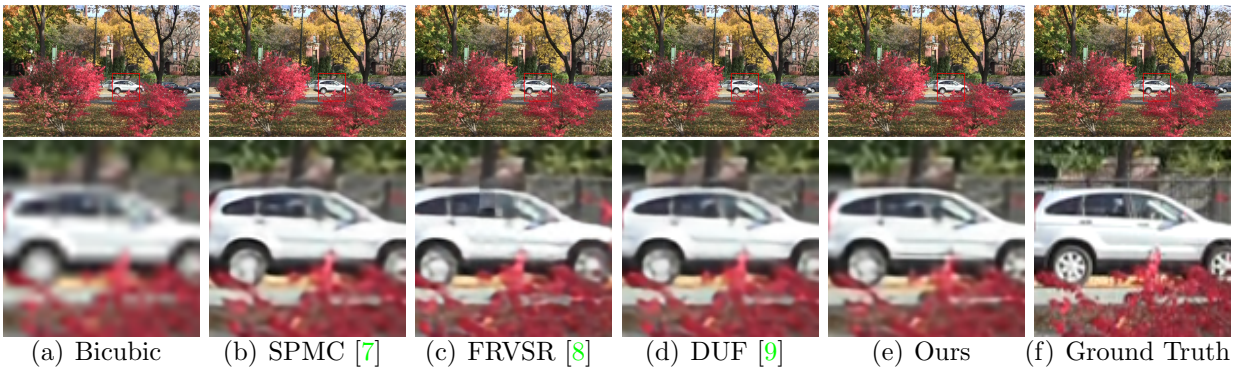
Scale	Average	Bicubic	VSRnet [5]	VESPCN [6]	SPMC [7]	FRVSR [8]	DUF [9]	Ours
×2	PSNR	28.43	31.30	-	-	-	-	**31.40**
	SSIM	0.868	0.928	-	-	-	-	**0.933**
×3	PSNR	25.28	26.79	27.25	27.49	-	-	**28.60**
	SSIM	0.733	0.697	0.845	0.840	-	-	**0.865**
×4	PSNR	23.81	24.73	25.34	25.88	26.17	26.21	**26.62**
	SSIM	0.634	0.697	0.730	0.774	0.798	0.798	**0.811**

(a) Bicubic (b) SPMC [7] (c) FRVSR [8] (d) DUF [9] (e) Ours (f) Ground Truth

Figure 5.2: Visual comparison of wide activated separate 3D convolution network and other state-of-the-art methods on *calendar*.

(a) Bicubic (b) SPMC [7] (c) FRVSR [8] (d) DUF [9] (e) Ours (f) Ground Truth

Figure 5.3: Visual comparison of wide activated separate 3D convolution network and other state-of-the-art methods on *city*.



(a) Bicubic (b) SPMC [7] (c) FRVSR [8] (d) DUF [9] (e) Ours (f) Ground Truth

Figure 5.4: Visual comparison of wide activated separate 3D convolution network and other state-of-the-art methods on *foliage*.

(a) Bicubic (b) SPMC [7] (c) FRVSR [8] (d) DUF [9] (e) Ours (f) Ground Truth

Figure 5.5: Visual comparison of wide activated separate 3D convolution network and other state-of-the-art methods on *walk*.

training process. Table 5.5 shows the comparison results of the wide activated separate 3D convolution and original separate 3D convolution on VID4 with up-sampling scale of 4. The visual comparisons are shown in Figures 5.7 to 5.10. We can see from the Figure 5.7 to Figure 5.10, the text, shape of windows, hub of car and the texture of cloth are recovered more realistically by our wide activated separate 3D convolution than the original activated separate 3D convolution.

5.3.3 Results on additional real-world examples

To enrich our experiments and verify the effectiveness of our method, we evaluate our method with an additional real-world test dataset which is provided by Tao et al. [7]. They collect the dataset from several commercial videos, which contain abundant natural-world and urban scenes. The test dataset consists 30 different sequence and each of them contains 31 consecutive frames. They down-sample all the original frames to generate patches with resolution of 960 $\times$ 540 as ground truth. Then they use bicubic down-

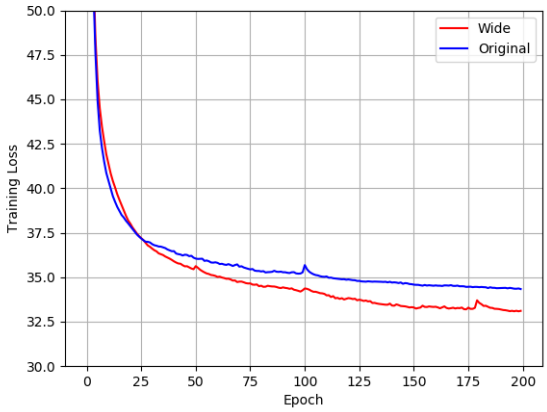


Figure 5.6: The training loss changes of wide activated separate 3D convolution method and original activated separate 3D convolution method

.

sampling method to generate LR frames. The first and last two frames of each sequence should not be considered as reference LR frames in order to ensure each reference frame have four supporting frames.

We test our method on the test dataset with the up-sampling scale of 4. Due to space limitations, visual comparison on only one frame of the six videos of the test dataset are shown in Figure 5.11 to Figure 5.16. These figures show the results of the original activated separate 3D convolution and wide activated separate 3D convolution. Our method has higher PSNR and SSIM for Figure 5.11 to Figure 5.14. For Figure 5.15, our method and original activated separate 3D convolution method have the same PSNR. And for Figure 5.16, our method has higher SSIM. We can see from the Figure 5.11 and 5.12, the horizontal edges of the building and the texture of the armour are recovered more clearly by our method. For Figure 5.13 and 5.14, the shadow of the handrail and the texture of window are recovered more realistically by our method. For Figures 5.15 and 5.16, the

Table 5.5: Comparison of results of wide activated separate 3D convolution and original separate 3D convolution on videos of VID4 with up-sampling scale of 4.

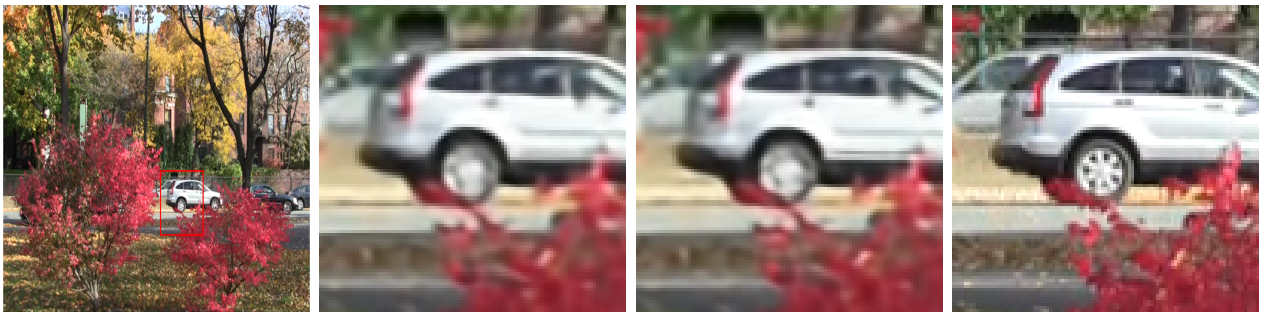
VID4	Wide activated separate 3D		Original activated separate 3D	
	PSNR	SSIM	PSNR	SSIM
Calendar	22.80	0.78	22.80	0.78
City	27.96	0.81	27.88	0.80
Foliage	26.39	0.77	26.30	0.76
Walk	29.31	0.87	29.03	0.86

(a) Full image (b) Original activated (c) Wide activated (d) Ground truth

Figure 5.7: Visual comparison of wide activated and original activated separate 3D convolution on *calendar*.

(a) Full image (b) Original activated (c) Wide activated (d) Ground truth

Figure 5.8: Visual comparison of wide activated and original activated separate 3D convolution on *city*.



(a) Full image (b) Original activated (c) Wide activated (d) Ground truth

Figure 5.9: Visual comparison of wide activated and original activated separate 3D convolution on *foliage*.

(a) Full image (b) Original activated (c) Wide activated (d) Ground truth

Figure 5.10: Visual comparison of wide activated and original activated separate 3D convolution on *walk*.

text and the face of the statue are recovered more clearly by our method.

5.4 Summary

In this chapter, we showed the experimental settings and the experimental results. We compared our method with other state-of-the-art methods in terms of PSNR and SSIM on VID4 dataset with up-sampling scales of 2,3 and 4. The comparisons demonstrate that our method outperforms other methods. We compared our method with other methods on real-word examples to reinforce the demonstration. In addition, we did experiment to compare the wide activated separate 3D convolution and the original activated separate 3D convolution. The results demonstrated the superiority of our method.

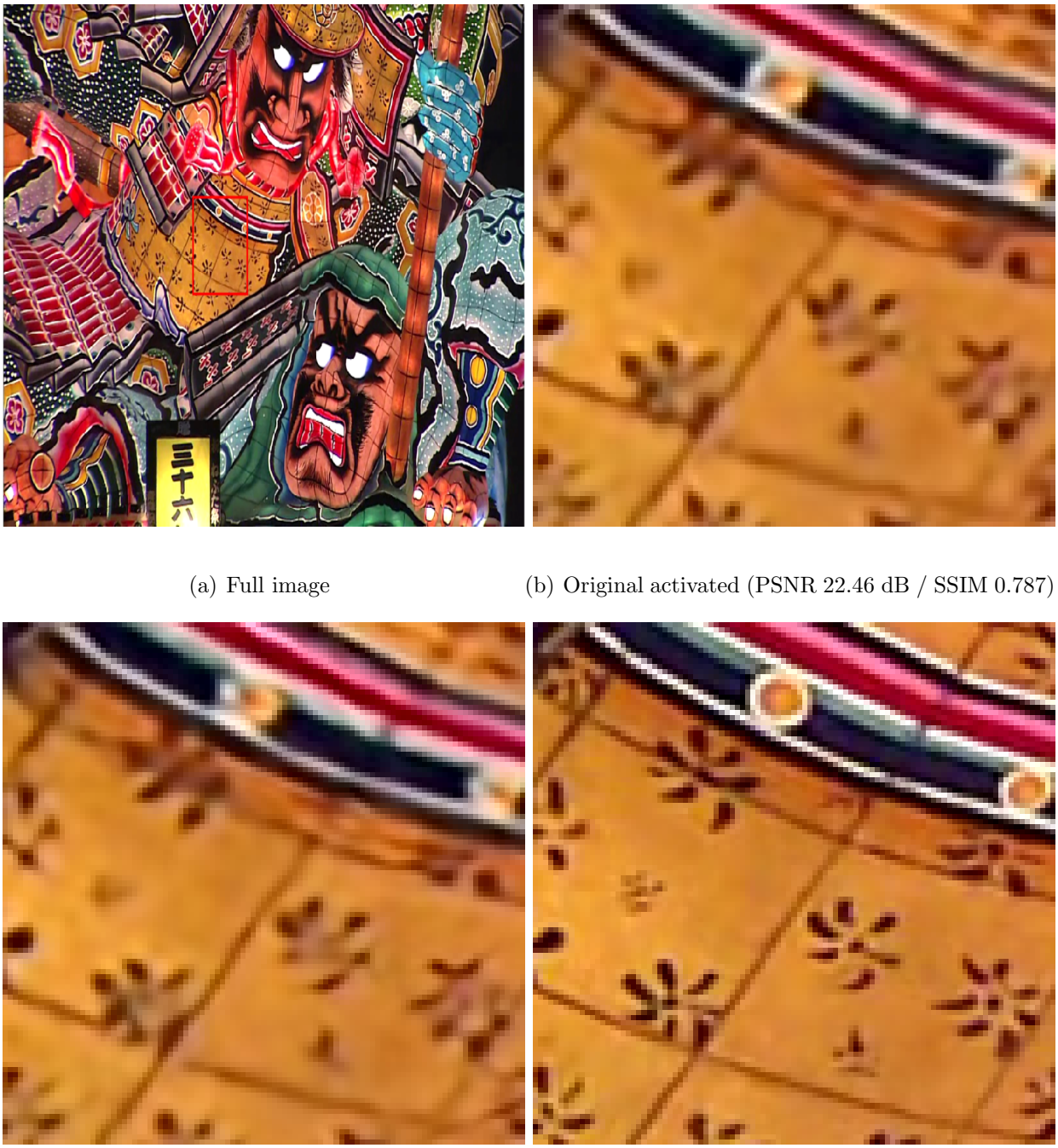
(a) Full image

(b) Original activated (PSNR 20.70 dB / SSIM 0.667)

(c) Wide activated (PSNR **20.76** dB / SSIM **0.669**)

(d) Ground truth

Figure 5.11: Visual comparison of wide activated and original activated separate 3D convolution on *hdclub_003_001*.



(a) Full image

(b) Original activated (PSNR 22.46 dB / SSIM 0.787)

(c) Wide activated (PSNR **22.75** dB / SSIM **0.800**)

(d) Ground truth

Figure 5.12: Visual comparison of wide activated and original activated separate 3D convolution on *hitachi_isee5_001*.

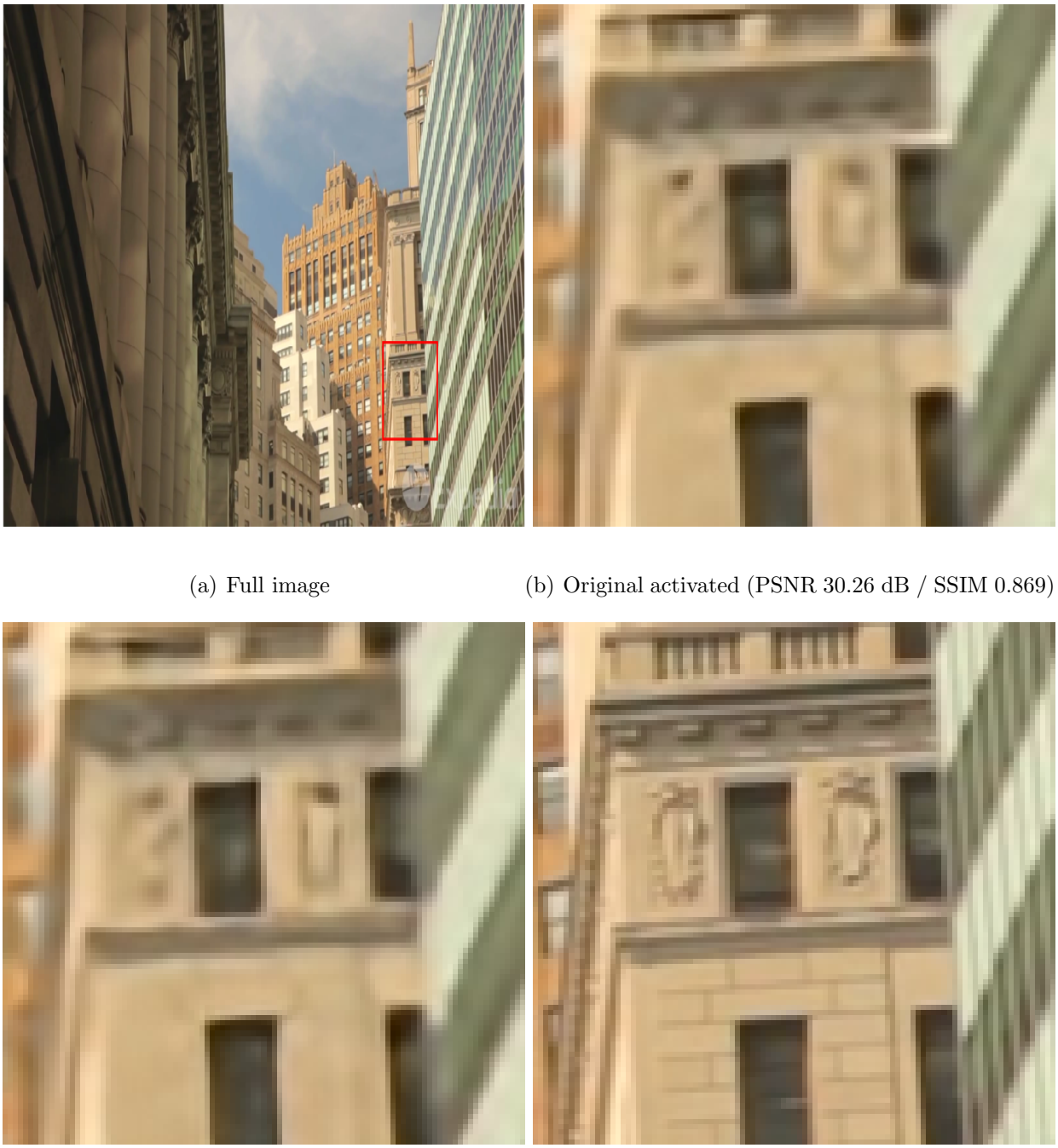
(a) Full image

(b) Original activated (PSNR 27.84 dB / SSIM 0.857)

(c) Wide activated (PSNR **27.90** dB / SSIM **0.860**)

(d) Ground truth

Figure 5.13: Visual comparison of wide activated and original activated separate 3D convolution on *jvc_009_001*.



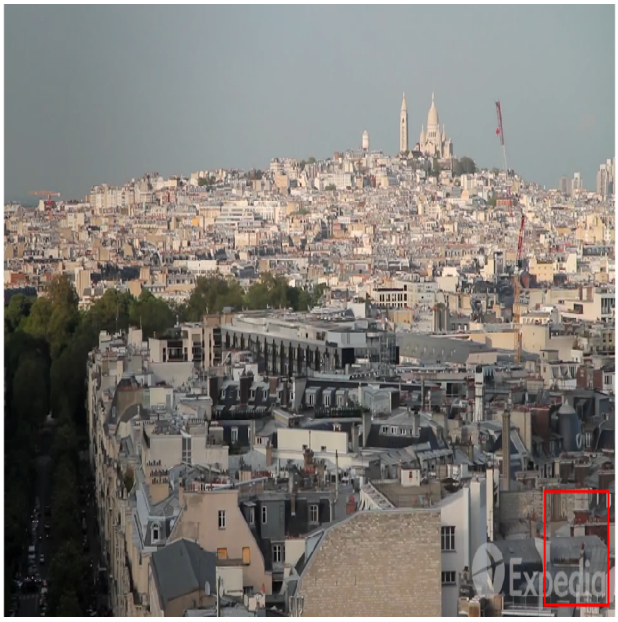
(a) Full image

(b) Original activated (PSNR 30.26 dB / SSIM 0.869)

(c) Wide activated (PSNR **30.41** dB / SSIM **0.874**)

(d) Ground truth

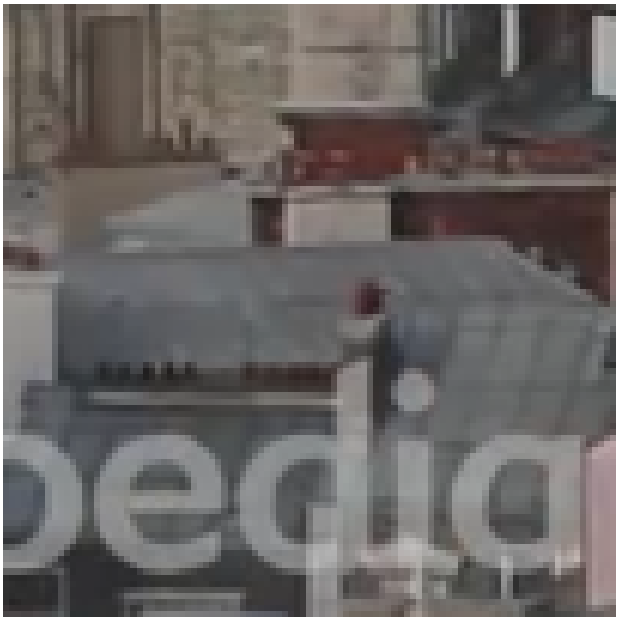
Figure 5.14: Visual comparison of wide activated and original activated separate 3D convolution on *NYVTG_006*.



(a) Full image

(b) Original activated (PSNR **25.95** dB / SSIM **0.769**)

(c) Wide activated (PSNR **25.95** dB / SSIM 0.768)



(d) Ground truth

Figure 5.15: Visual comparison of wide activated and original activated separate 3D convolution on *PRVTG_012*.

(a) Full image

(b) Original activated (PSNR **25.38** dB / SSIM 0.750)

(c) Wide activated (PSNR 25.36 dB / SSIM **0.751**)

(d) Ground truth

Figure 5.16: Visual comparison of wide activated and original activated separate 3D convolution on *RMVTG_011*.

Chapter 6

Conclusions

In this thesis, we proposed a novel convolution network based method for VSR task: the wide activated separate 3D convolution residual block. The proposed method aims to overcome the limitations of previous methods.

- We introduced the separate 3D convolution into VSR problem for the first time. We proposed a improved separate 3D convolution - wide activated separate 3D convolution residual block. With introducing the wide activated separate 3D convolution, we added more activated units into the network while maintained the same number of parameters as the original counterpart. The experiments proved that the wide activated separate 3D convolution network outperforms the original activated separate 3D convolution network in temporal information capture.

- Most of previous methods utilized a 2-component structure: a super-resolution model combined with a motion estimation and compensation model. Unlike these previous methods, our proposed wide activated separate 3D convolution network for VSR

only contains an end-to-end super-resolution model. The wide activated separate 3D convolution contains convolution operation in the spatial domain as well as in the temporal domain, which allows the network have the capability of extracting the spatial-temporal feature directly. Compared with these two-component models, our method can be trained as a whole, which ensures the network to reach the optimum easily. The comparisons in terms of PSNR and SSIM on VID4 test dataset with up-sampling scales of 2,3 and 4 demonstrated that the wide activated separate 3D convolution blocks outperform these VSR methods that utilize motion estimation and compensation models.

In the future, we would like to further enhance our wide activated separate 3D convolution. We will try to replace the 3D convolution along spatial domain with the deformable convolution, which have the ability to adapt to the geometric variations of objects. The deformable convolution can gradually align the input LR frames, which can help to better utilize the temporal information.

References

[1] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European Conference on Computer Vision*, pages 391–407. Springer, 2016.

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

[3] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, and Qingmin Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 2019.

[4] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4681–4690, 2017.

[5] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2(2):109–122, 2016.

[6] Jose Caballero, Christian Ledig, Andrew Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4778–4787, 2017.

[7] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *IEEE International Conference on Computer Vision*, pages 4472–4480, 2017.

[8] Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6626–6634, 2018.

[9] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3224–3232, 2018.

[10] Yapeng Tian, Yulun Zhang, Yun Fu, and Chenliang Xu. Tdan: Temporally deformable alignment network for video super-resolution. *arXiv preprint arXiv:1812.02898*, 2018.

[11] Soo Ye Kim, Jeongyeon Lim, Taeyoung Na, and Munchurl Kim. 3DSRnet: Video super-resolution using 3D convolutional neural networks. *arXiv preprint arXiv:1812.09079*, 2018.

[12] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

[13] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *IEEE International Conference on Computer Vision*, pages 4489–4497, 2015.

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

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[16] Sabyasachi Moitra. Single-image super-resolution techniques: A review. *International Journal for Science and Advance Research in Technology*, 3:271–283, 04 2017.

[17] Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160, 1981.

[18] Claude E Duchon. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology*, 18(8):1016–1022, 1979.

[19] Simon Baker and Takeo Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):1167–1183, 2002.

[20] Shengyang Dai, Mei Han, Wei Xu, Ying Wu, and Yihong Gong. Soft edge smoothness prior for alpha channel super resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 7, pages 1–8. Citeseer, 2007.

[21] Shengyang Dai, Mei Han, Wei Xu, Ying Wu, Yihong Gong, and Aggelos K Katsaggelos. Softcuts: a soft edge smoothness prior for color image super-resolution. *IEEE Transactions on Image Processing*, 18(5):969–981, 2009.

[22] Raanan Fattal. Image upsampling via imposed edge statistics. *ACM Transactions on Graphics (TOG)*, 26(3):95, 2007.

[23] Jian Sun, Zongben Xu, and Heung-Yeung Shum. Image super-resolution using gradient profile prior. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[24] Jian Sun, Zongben Xu, and Heung-Yeung Shum. Gradient profile prior and its applications in image super-resolution and enhancement. *IEEE Transactions on Image Processing*, 20(6):1529–1542, 2011.

[25] Zhouchen Lin, Junfeng He, Xiaoou Tang, and Chi-Keung Tang. Limits of learning-based superresolution algorithms. *International Journal of Computer Vision*, 80(3):406–420, 2008.

[26] J. Deng, W. Dong, R. Socher, L. Li, and and. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[27] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[28] Yu Xiafei and Zhao Jiying. Wide separate 3D convolution for video super resolution. In *7th IEEE Global Conference on Signal and Information Processing*, 2019.

[29] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision*, pages 184–199. Springer, 2014.

[30] Matthew D Zeiler, Graham W Taylor, Rob Fergus, et al. Adaptive deconvolutional networks for mid and high level feature learning. In *International Conference on Computer Vision*, volume 1, page 6, 2011.

[31] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[32] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1646–1654, 2016.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

[35] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 136–144, 2017.

[36] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[37] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[38] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[39] Tong Tong, Gen Li, Xiejie Liu, and Qinquan Gao. Image super-resolution using dense skip connections. In *IEEE International Conference on Computer Vision*, pages 4799–4807, 2017.

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

[41] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[42] Marius Drulea and Sergiu Nedevschi. Total variation regularization of local-global optical flow. In *2011 International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 318–323. IEEE, 2011.

[43] Yan Huang, Wei Wang, and Liang Wang. Bidirectional recurrent convolutional networks for multi-frame super-resolution. In *Advances in Neural Information Processing Systems*, pages 235–243, 2015.

[44] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pages 667–675, 2016.

[45] Yang Zhou, Xiaohong Liu, Lei Chen, and Jiying Zhao. Video super-resolution via dynamic local filter network. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 41–45. IEEE, 2018.

[46] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *IEEE International Conference on Computer Vision*, pages 764–773, 2017.

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

[48] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[49] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

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

[51] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1, 1988.

[52] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013.

[53] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[54] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[55] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[56] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[57] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[58] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, 2017.

[59] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3D residual networks. In *IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[60] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[61] Jiahui Yu, Yuchen Fan, Jianchao Yang, Ning Xu, Zhaowen Wang, Xinchao Wang, and Thomas Huang. Wide activation for efficient and accurate image super-resolution. *arXiv preprint arXiv:1808.08718*, 2018.

[62] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.

[63] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

[64] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016.

[65] David Martin, Charless Fowlkes, Doron Tal, Jitendra Malik, et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. 2001 International Conference on Computer Vision, 2001.

[66] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[67] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, pages 1–20, 2017.

[68] Ce Liu and Deqing Sun. A Bayesian approach to adaptive video super resolution. In *2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 209–216. IEEE, 2011.

[69] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.