NORTHWESTERN UNIVERSITY

COMPUTER SCIENCE DEPARTMENT

Technical Report
Number: NU-CS-2024-03

March 2024

Deep Learning Applied to Image and Video Processing

By

Xijun Wang

ABSTRACT

This dissertation introduces deep learning (DL) methods applied to image and video processing, specifically concentrating on two domains: image and video restoration, and video classification with action localization.

In the restoration domain, we introduce several innovative deep learning methodologies to address three challenges: super-resolution (SR), atmospheric turbulence (AT) correction, and motion blur (MB) removal.

Generative Adversarial Networks (GANs) have demonstrated impressive performance in addressing super-resolution challenges, because of their ability to produce visually realistic images and video frames. However, previous GAN-based models frequently suffer from undesired side effects in their outputs, such as unexpected artifacts and noise. To mitigate these artifacts and enhance the perceptual quality of the results, in Chapter 1, we propose a general method that can be effectively used in most GAN-based super-resolution models by integrating essential spatial information into the training process. We extract spatial information from the input data and integrate it into the training loss, making the corresponding loss a spatially adaptive (SA) one. We show that the proposed approach is independent of the methods employed for spatial information extraction, as well as independent of SR tasks and models. This method consistently guides the training process towards generating visually pleasing SR images and video frames, substantially reducing artifacts and noise, and ultimately leading to enhanced perceptual quality. Besides including the spatial information through training loss, we also discover incorporating it through the model framework in Chapter 2. We design a new framework that incorporates two collaborative discriminators whose aim is to jointly improve the quality of the reconstructed video sequence. While one discriminator focuses on the general properties of the images, the second one specializes in

obtaining realistically reconstructed features, such as edges. Experimental results demonstrate that the learned model outperforms current state-of-the-art models, yielding super-resolved frames with fine details, sharp edges, and reduced artifacts.

Atmospheric turbulence, a common phenomenon in daily life, arises primarily due to the uneven heating of the Earth's surface. As a result, it causes distortion and blurring in acquired images or videos, significantly affecting downstream vision tasks, especially those dependent on capturing clear, stable images or videos from outdoor environments, such as accurate object detection or recognition. It is a challenging restoration task as it consists of two types of distortions: geometric distortion and spatially variant blur. In Chapter 3, we first propose a variational inference framework AT mitigation baseline, wherein we improve the performance by learning latent prior information from the input and degradation processes. Then we design a novel deep conditional diffusion model within the variational inference framework to further enhance the perceptual quality of output images. We demonstrate that the proposed framework achieves good quantitative and qualitative results on a comprehensive synthetic AT dataset. Though existing deep learning-based methods have achieved great performance in synthetic scenarios, they invariably exhibit a performance drop when applied to real-world cases. Therefore, in Chapter 4 we further propose a real-world atmospheric turbulence mitigation method under a domain adaptation framework, which connects supervised simulated atmospheric turbulence correction with unsupervised real-world atmospheric turbulence correction. We will show our proposed method enhances performance in real-world atmospheric turbulence scenarios, improving both image quality and downstream vision tasks.

Respiratory motion and the resulting artifacts are considered to be a big problem in abdominal Magnetic Resonance Imaging (MRI). Many previous deep-learning techniques have been developed to address these respiratory motion artifacts. However, many models tend to oversmooth fine details, such as vessels in liver MRIs, while these details are most important for medical diagnosis. Thus, in Chapter 5, similar to our approach in SR, we propose a Generative Adversarial Networks (GAN)-based model for removing motion blur in abdominal MRI. We incorporate perceptual loss as part of our training loss to further enhance the perceptual quality of the images. Our model generates motion-reduced images with clearer and better fine-details, thereby providing radiologists with more realistic MRI images to aid in diagnosis.

For the classification and action localization with videos, we present deep learning methods for solving avian-solar activity classification and weakly supervised action localization.

Activity classification is essential in various real-life scenarios involving both humans and animals. The demand for precise activity classification concerning avian-solar interactions is rising, as the usage of solar energy facilities, such as photovoltaic array power stations, has been observed to impact bird species richness, behavior, and activity. However, there has been no effort to develop an automated system for monitoring and classifying avian-solar interactions. Current methods depend on human observers, which are time-consuming, resource-intensive, and prone to errors related to searcher efficiency. With the recent success of Deep Learning models in activity classification, in Chapter 6, we introduce a recurrent neural network-based model for automatically classifying six avian activities around solar energy facilities. Our model integrates crucial feature engineering metadata with video frame data, facilitating enhanced learning and more accurate activity classification. Furthermore, we address the challenge of data imbalance during training and demonstrate our model's effectiveness in detecting and classifying various activities within video tracks. Additionally, we analyze the saliency/backpropagation map of the trained proposed model and validate its decision-making rationale.

Weakly-supervised temporal action localization aims to identify and localize the action instances in untrimmed videos with only video-level action labels. Humans can adapt abstract-level knowledge about actions in various video scenarios and detect the occurrence of actions. In Chapter 7, we mimic how humans do and introduce a new perspective for locating and identifying multiple actions in a video. We propose a network named VQK-Net with a video-specific query-key attention modeling, which learns a unique query for each action category in every input video. These learned queries encapsulate abstract-level features of actions and are capable of adapting this knowledge to the target video scenario, facilitating the detection of corresponding actions along the temporal dimension. To enhance the learning of these action category queries, we leverage not only the features of the current input video but also the correlations between different videos using a novel video-specific action category query learner worked with a query similarity loss. Finally, we conduct extensive experiments on three widely adopted datasets, achieving state-of-the-art performance.

KEYWORDS

Deep Learning, image and video processing, image and video restoration, video classification with action localization

TABLE OF CONTENTS

Keywords 6

List of Figures 13

List of Tables 17

Chapter 1: Spatial Adaptive Loss Function for GAN-based Super-Resolution Models . 19

1.1 Introduction 19

1.2 Proposed Method 22

1.2.1 Extracting Spatial Information 22

1.2.2 Spatially Adaptive Pixel Loss 24

1.2.3 GAN Loss and Perceptual Loss 28

1.2.4 Spatially Adaptive Loss used in Super-Resolution Models 30

1.3 Experiments 34

1.3.1 Datasets 34

1.3.2 Training Process 36

1.3.3 Hyper-Parameters for the Loss Functions 37

1.4 Results 37

1.5 Conclusion 40

Chapter 2: Composite Discriminators for GAN-based Video Super-Resolution 44

2.1 Introduction 44

2.2 Proposed Method 45

2.2.1 A Composite Discriminator 46

2.2.2 Pixel and Feature Losses 50

2.2.3 Final Loss for Generator 51

2.3 Experiments 52

2.4 Results 54

2.5 Conclusion 55

Chapter 3: Atmospheric Turbulence Correction via Variational Deep Diffusion 56

3.1 Introduction 57

3.2 Proposed Method 58

3.2.1 Conditional Diffusion Model 58

3.2.2 Variational Inference Framework 59

3.2.3 Inference 62

3.3 Experiments 62

3.4 Results 63

3.5 Conclusion 64

Chapter 4: Real-World Atmospheric Turbulence Correction via Domain Adaptation . . 65

4.1 Introduction 66

4.2 Proposed Method 67

4.2.1 Domain Adaptation Framework 67

4.2.2 Generator Framework 71

4.2.3 Training & Testing 72

4.3 Experiments 74

4.4 Results 75

4.4.1 Image Quality 75

4.4.2 Downstream Person Identification Task 76

4.5 Conclusion 77

Chapter 5: Motion Blur Reduction in Abdominal MRIs using GANs 79

5.1 Introduction 79

5.2 Proposed Method 81

5.2.1 Motion Simulation 81

5.2.2 The Motion Reduction Pipeline 84

5.2.3 The Loss Functions 84

5.3 Experiments 86

5.4 Results 88

5.4.1 Experimental Results 88

5.4.2 Visualizations 90

5.5 Conclusion 92

Chapter 6: Avian Activity Classification Using Recurrent Networks 93

6.1 Introduction 93

6.2 Proposed Method 96

6.2.1 Avian Activity Dataset 96

6.2.2 Model Description 97

6.2.2.1 Video Bi-LSTM & Meta Bi-LSTM 97

6.2.2.2 Video-Meta Fusion Bi-LSTM 99

6.2.3 Training Objective 100

6.2.4 Data Augmentation in Metadata and Video Data 100

6.3 Experiments 100

6.4 Results 102

6.4.1 Results and Comparisons 102

6.4.2 Model Analysis 102

6.5 Conclusion 104

Chapter 7: Query-Key Attention Modeling for Weakly-Supervised Temporal Action Localization 105

7.1 Introduction 105

7.2 Proposed Method 108

7.2.1 Problem Statement 109

7.2.2 Feature Extraction 109

7.2.3 Main Pipeline Overview 110

7.2.4 Query Learner 110

7.2.5 Query-Key Attention 113

7.2.6 Training Objectives 114

7.2.7 Temporal Action Localization 117

7.3 Experiments 118

7.3.1 Experimental Settings 118

7.4 Results 120

7.4.1 Comparison with State-of-art Methods 120

7.4.2 Ablation Studies & Qualitative Results 121

7.5 Conclusion 124

References 145

LIST OF FIGURES

1.1 Deep learning on super-resolution. 19

1.2 The gray image and its edge maps (a) The gray image; (b) Edge map computed by local variance; (c) Edge map computed by Canny detection. 25

1.3 Super-resolution with GANs. 29

1.4 Pixel loss and feature loss visualization. Adapted from [31]. 30

1.5 The ESRGAN's generator architecture. It consists of a convolutional (conv) operation applied to the input image, followed by 23 basic blocks and a conv layer. A long skip connection is used, where the output feature maps will add with the output feature maps from the first conv operation. After this, up-sampling and two more conv operations are conducted to obtain the final SR image output. The basic block used here is Residual-in-Residual-Dense-Block (RRDB), which combines the multi-level residual network and dense connections. [13] 32

1.6 The ESRGAN's discriminator architecture. It consists of conv, Leaky ReLU, and batch normalization layers along the way, followed by two fully connected layers and a sigmoid activation function to obtain a probability.[13] 33

1.7 The VSRResFeatGAN's generator architecture. In the beginning, 5 conv operations are applied separately on each input bicubic interpolated frame at time t-2, t-1, t, t+1, t+2. The resulting feature maps are then concatenated together, and the combined feature map will be inputted into the following two conv layers and 15 residual blocks. Inside each residual block, there are two conv layers, each followed by a ReLU layer, and the input feature map is added to the output feature map to obtain the final output of the residual block. In the end, one more conv operation is conducted to obtain the final SR frame output at time t.[14] 35

1.8 The VSRResFeatGAN's discriminator architecture. It consists of conv, batch normalization, and Leaky ReLU layers along the way, followed by one fully connected layer and a sigmoid activation function to obtain a probability. [14] 35

1.9 Qualitative comparison of ESRGAN, SA-ESRGAN(lv), and SA-ESRGAN(canny). 41

1.10 Qualitative comparison of ESRGAN, SA-ESRGAN(lv), and SA-ESRGAN(canny). 42

1.11 Qualitative comparison of VSRResFeatGAN, SA-VSRResfeatGAN(lv), and SA-VSRResfeatGAN(canny). 43

2.1 The proposed model. 49

2.2 The proposed architecture for the generator (first row) and discriminator (second row) [48] 50

2.3 Qualitative comparison of results obtained by VSRResFeatGAN [48], SA-GAN [47], and VSRCoDiGAN on scale factor 3. The ground truth frames are shown in Fig. 2.3(a). 53

3.1 Atmospheric turbulence. Adapted from the internet. 56

3.2 Conditional denoising diffusion process. 59

3.3 Training framework of AT-VarDiff model. 60

3.4 Testing framework of AT-VarDiff model. 61

3.5 Visual comparisons of AT-DDPM [65], simple-DDPM, and AT-VarDiff. 64

4.1 Domain adaptation framework. 65

4.2 Real-world atmospheric turbulence mitigation model (Real-ATM) framework. . . . 70

4.3 Generator training framework. 73

4.4 Testing framework. 73

4.5 Visual samples of real-world AT-input, and the restored results from Syn-ATM and Real-ATM. (PIQE/BRISQUE) is shown in the parenthesis, the best score is marked in bold. 74

5.1 The motion simulation process. Three types of motion (i.e., sinusoidal motion, random motion and interweaving motion) are simulated by manipulating K-space data to generate new k-space data using DFT and iDFT. 82

5.2 Illustration of the completed motion reduction pipeline. It is composed of a generator and a discriminator. The Vgg-16 architecture [115] is used to form the perceptual loss. The CNN U-net blocks are used as the generator in our proposed GAN model. The pairs of the clean image and the image with simulated motion are used as the training input pairs. 83

5.3 example 89

5.4 example 91

6.1 Video track example. 94

6.2 Number of samples in each activity. 96

6.3 Meta Bi-LSTM, Video Bi-LSTM, Video-Meta Fusion Bi-LSTM models' architecture. 98

6.4 Saliency maps of the test video tracks. 101

6.5 The confusion matrices of Meta Bi-LSTM model trained with unbalanced (left) or balanced (right) datasets. 103

7.1 A video track contains multiple activities. 106

7.2 Action category queries are learned so as to contain action knowledge at the abstract level, which can be used to identify and detect corresponding actions in the target video. 107

7.3 Overview of our proposed VQK-Net model. $\otimes$, $\oslash$ and $\odot$ denote the element-wise multiplication, element-wise division and vector inner product. 112

7.4 (a) Query learner module. (b) Query-key attention module 114

7.5 Visualization of the learned action category queries on THUMOS14 test videos via t-SNE[205]. 123

7.6 Qualitative results on THUMOS14. The horizontal axis denotes time. The first plot is the ground truth (GT) action intervals. The remaining two plots illustrate the detection scores of ground truth action, shown in green curves, and the detected action instances using the uniform and video-specific query-key attention strategies, respectively. 125

LIST OF TABLES

1.1 Metrics comparison between ESRGAN, SA-ESRGAN (local-variance), and SA-ESRGAN (canny). The results are evaluated on Myanmar testing Dataset for scale factor 4. 39

1.2 Metrics comparison of ESRGAN, SA-ESRGAN (local-variance), and SA-ESRGAN (canny) in PSNR, SSIM, and LPIPS metrics. For the LPIPS, smaller is better. The results are evaluated on VidSet4 Dataset for scale factor 4. 39

1.3 Metrics comparison of VSRResFeatGAN, SA-VSRResFeatGAN (local-variance), and SA-VSRResFeatGAN (canny) in terms of PSNR, SSIM, and LPIPS metrics. For the LPIPS, smaller is better. The results are evaluated on VidSet4 Dataset for scale factor 4. 39

1.4 Metrics comparison between VSRResFeatGAN, SA-VSRResFeatGAN (local-variance), and SA-VSRResFeatGAN (canny). The results are evaluated on Myanmar testing Dataset for scale factor 4. 39

2.1 PSNR and SSIM comparison with state-of-the-art VSR models on the VidSet4 dataset for scale factor 3. 54

3.1 LPIPS, FID & NIQE metrics comparison on simple-DDPM, AT-VarDiff, and AT-DDPM [65]. 62

4.1 Image quality results of the Syn-ATM model on the synthetic AT testing set. 76

4.2 Image quality results of Syn-ATM and Real-ATM models on the real-world AT testing set. The best result is marked in bold. 76

4.3 Human recognition results on real-world AT videos and restored videos by Syn-ATM and Real-ATM. The best result is marked in bold. 77

5.1 Comparisons of four model performances (mean $\pm$ std) in terms of MSE, SSIM, PSNR and perceptual distance (the smaller the better) on the testing dataset. * denotes the comparison is statistically significant ($p < 0.05$). 88

5.2 Comparisons of motion reduction performance (mean $\pm$ std) in MSE, SSIM, PSNR on three different motion types of testing dataset. * denotes the comparison is statistically significant ($p < 0.05$). 90

6.1 Test accuracy of Meta Bi-LSTM, Video Bi-LSTM, and Video-Meta Fusion Bi-LSTM models. If the top 2 or top 3 predicted activities contain the given ground-truth activity, then we count it as a correct classification within the Top2 or Top3 test accuracy correspondingly. 103

7.1 The comparison with state-of-art TAL works on the THUMOS14 dataset. †refers to using additional information, such as human pose or action frequency. I3D is abbreviation for I3D features. 116

7.2 Comparisons with state-of-art works on ActivityNet1.2 dataset. AVG means the average mAP from IoU 0.5 to 0.95 with step size 0.05. 119

7.3 Comparisons with state-of-art works on ActivityNet1.3 dataset. AVG means the average mAP from IoU 0.5 to 0.95 with step size 0.05. 120

7.4 Evaluation of uniform and video-specific query learning strategies on THUMOS14. 121

7.5 Analysis of distance function used in query similarity loss on THUMOS14. 122

CHAPTER 1
SPATIAL ADAPTIVE LOSS FUNCTION FOR GAN-BASED SUPER-RESOLUTION MODELS

1.1 Introduction

Super-resolution (SR) is a conventional task in image and video processing, which aims to generate high-resolution (HR) images or videos from low-resolution (LR) counterparts. Various methodologies and algorithms have been developed to tackle this task [1]. In recent years, Deep Learning (DL) has emerged as a powerful approach in the field, demonstrating outstanding performance in various image processing tasks, including SR [2], [3]. (Please note that portions of this chapter restate text from our previously published work [4] and a journal paper currently under review.)

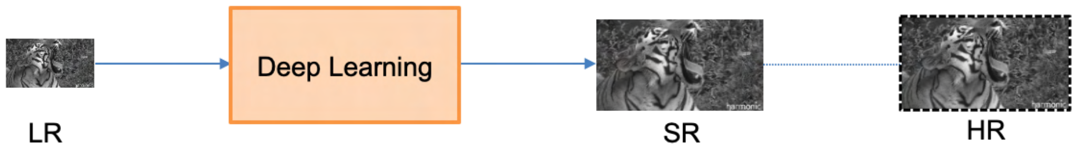


Figure 1.1: Deep learning on super-resolution.

In pioneering works employing deep neural network (DNN) on SR problems, Dong et al. [5], [6] developed the SRCNN for single image super-resolution (SISR). This approach learned end-to-end mapping from a single LR image to its HR counterpart using a Convolutional Neural Network (CNN). Subsequently, Kappeler et al. [7] proposed VSRnet, a three-layer CNN for video super-resolution (VSR). Both works have outperformed the previous state-of-art methods. Since then, numerous DNN architectures have emerged in these domains [8], [9]. Recently, the focus in this field has shifted from solely achieving accurate image or video recovery to generating visually pleasing results. To achieve this objective, generative adversarial networks [10] have been widely used in current state-of-art studies [11]–[20]. These works have demonstrated the ability of adversarial learning to guide generative outcomes toward the domain of natural images, which are perceptually authentic to humans. Consequently, this approach yields images and video frames with a realistic appearance.

The majority of outcomes produced by GAN architectures have surpassed those of non-GAN architectures in terms of perceptual quality for images and video frames. Recent studies have sought to further improve perceptual quality by integrating specific spatial information into network training. This spatial information, often crucial for enhancing visual quality, includes features like edges and textures. For example, Jiang et al. [21] proposed an edge-enhancement subnetwork to recover the high-frequency edge details of images. Wang et al. [22] and Wu et al. [23] integrated a Spatial Feature Transform layer into their network's architecture, leveraging semantic texture information from semantic segmentation maps during training. Similarly, Zhao et al. [24] developed a region-level non-local module integrated into the generative network to capture long-range feature dependencies. These studies collectively highlight the significant contribution of incorporating essential spatial information during GAN network training to produce visually pleasing outcomes. However, these methods rely on modifying or constructing new architectures or layers within their networks. Such approaches often possess limitations due to their strong dependence on specific network architectures, hindering straightforward adaptation to different network structures. In such cases, using the same approach may not reach the same performance, and many adjustments will be needed. In the current works, few have tried to include spatial information by improving the objective loss, which can be easily extended to different models regardless of their architectures, as similar loss functions are employed during training. Therefore, this chapter proposes an effective and efficient approach to enhance the training loss by integrating spatial information. This is achieved by adapting the original loss function to become spatially adaptive (SA). We demonstrate we can generate better images and video frames with higher perceptual quality after training different networks with the SA losses. Given the superior performance of GAN in the SR field, our focus is primarily on GAN models.

In state of art GAN-based super-resolution models [11]–[15], [17], [19]–[22], [24], training losses typically contain two primary components: adversarial loss and distance-based fidelity losses. The adversarial loss originates from the adversarial mechanism, initially introduced by Goodfellow et al. [10]. This loss drives the adversarial training dynamics between the generator and discriminator networks. The distance-based fidelity losses are commonly defined in pixel or feature spaces, and they can be included for different purposes. For example, most of them are used to assess the results' spatial quality [11]–[15], [17], [19], [21], [22], [24], which compute the difference between super-resolved images/frames and their corresponding HR counterparts in the low-level pixel space or the high-level feature space. While some of them are computed between adjacent estimated video frames, aiming at improving the results' temporal quality in the video super-resolution scenarios [15]. Since our goal is to further improve the results' perceptual quality, our focus lies on the losses used for discriminating the spatial quality. High-frequency details such as edges are crucial for human vision but are more challenging to recover accurately than flat areas. Distortions and artifacts are often more pronounced around these edges, impacting perceptual quality and subsequent vision applications. Such distortions and artifacts can be easily observed by human eyes and negatively impact the results' perceptual quality as well as the accuracy of subsequent vision applications. Therefore, this chapter selects edge information as the representative of spatial information. We first extract spatial (edge) information from input images or video frames. We then incorporate it into the original distance-based fidelity loss (defined in the pixel space in particular), making the new training loss a spatially adaptive(SA) one. We will show that the proposed SA loss can effectively guide the network to pay more attention to recovering those edge areas and generate a more accurate and sharper edge structure. This augmentation significantly enhances the visual impact of the ultimate super-resolved outcomes. Besides, instead of adding additional loss terms, the SA loss concept utilizes spatial information within the commonly-used pixel loss to train SR DNN models. It is also independent of the network's architecture, thus can be easily and efficiently incorporated along with different network architectures.

1.2 Proposed Method

Edge information is a crucial factor that influences image and video quality. Precise edges contribute to enhancing the perceptual quality of super-resolved images and videos. Therefore, we select edge information as the primary representative of spatial characteristics. We will first edge information from input data and integrate it into the low-level pixel training loss space, transforming the conventional pixel loss into a Spatially Adaptive (SA) one. Such SA loss can guide the models to focus more on recovering edge areas during training, thereby preserving edge information effectively. Our experiments demonstrate that employing the proposed SA loss into the training of both image and video SR GAN models results in a noticeable enhancement in the visual quality of the output images and video frames.

1.2.1 Extracting Spatial Information

As explained above, we use edge information as our spatial information. To extract edges from the input images or video frames, we utilize two different algorithms in this chapter: the local variance algorithm, generating an edge mask with values between 0 and 1, and the widely recognized Canny edge detection algorithm, which yields a binary edge mask with values of 0 or 1. Utilizing two different edge extraction algorithms stems from our hypothesis that the proposed Spatially Adaptive

(SA) loss concept remains unaffected by the choice of edge extraction methods or the resulting masks. As long as the algorithm can efficiently extract edge information from input data, the SA loss can effectively utilize the corresponding extracted edge mask. In the following, we will first describe the local variance and canny edge detection algorithms that we have used.

In this chapter, images are considered to be multi-channelized. For instance, grayscale images typically consist of one channel, while RGB-color images comprise three channels. Images in the feature domain often contain multiple channels.

1) Local variance edge detection: The local variance edge detection algorithm [25] takes a multi-channel image x with element $x(k,i,j)$ as input, where k represents the channel index and (i,j) is the pixel location within the kth channel. The local variance $\mu_{k,i,j}(x)$ at pixel location (i,j) within the kth channel is computed by:

$$\mu_{k,i,j}(x) = \sum_{(l_1,l_2)\in\Gamma_{k,i,j}} \frac{1}{|\Gamma_{k,i,j}|}(x(k,i+l_1,j+l_2) - m_{k,i,j}(x))^2, \tag{1.1}$$

where

$$m_{k,i,j}(x) = \sum_{(l_1,l_2)\in\Gamma_{k,i,j}} \frac{1}{|\Gamma_{k,i,j}|} x\left(k,i+l_1,j+l_2\right) \tag{1.2}$$

is the local mean. $\Gamma_{k,i,j}$ is the analysis window around $x(k,i,j)$, and $|\Gamma_{k,i,j}|$ denotes the number of elements in the analysis window.

The local mean $m_{k,i,j}(x)$ and local variance $\mu_{k,i,j}(x)$ compute the average and the variance of the pixels within the analysis window $\Gamma_{k,i,j}$. It is evident that the local variance $\mu_{k,i,j}(x)$ exhibits high values in high-frequency image areas such as edges, while displaying low values in flat regions. We further normalize its values to the range [0,1], getting the final edge map, defined as

$W(x)$, with elements

$$w_{k,i,j}(x) = \frac{\mu_{k,i,j}(x)}{\mu_{k,i,j}(x) + \delta}, \tag{1.3}$$

where $\delta > 0$, a tuning parameter determined experimentally. It is easy to see that in flat regions $w_{k,i,j}(x) \approx 0$, while in areas of high spatial activity like edge regions $w_{k,i,j}(x) \approx 1$. An example of the edge map computed by the local variance method for a grayscale image (k = 1) is shown in Figure 1.2(b) (The displayed edge map image is a result of scaling $W(x)$ by 255). We can see that the edge areas take higher values than the non-edge(flat) areas.

2) Canny edge detection: Since canny edge detection [26] is a well-known and typical edge extraction method, we won't further explain its details here. For our experiments, we directly utilize the Canny function from the OpenCV library [27]. The canny method outputs a binary edge map, with edge pixels assigned a value of 1 and non-edge pixels assigned a value of 0. Figure 1.2(c) depicts the resultant edge map obtained through the Canny method. Again, the edge map image shown there is scaled by 255, and we can also see that the edge areas take higher values than the non-edge (flat) areas.

Next, we demonstrate the integration of extracted edge information into the training losses of the SR models, thereby transforming them into Spatially Adaptive (SA) losses.

1.2.2 Spatially Adaptive Pixel Loss

Currently, most SR DL models utilize distance-based losses as either their primary training loss or as a component of it. Specifically, SR GAN models commonly integrate distance-based losses to regularize GAN training, often defined in both pixel and feature spaces [12]–[15], [17], [22], [19], [20], [23], [24], [28]. Our objective is to incorporate spatial information (edge information in this study) extracted from input data into the distance-based loss defined in pixel space. We name this enhanced pixel loss the spatially adaptive (SA) pixel loss. It effectively guides the model to

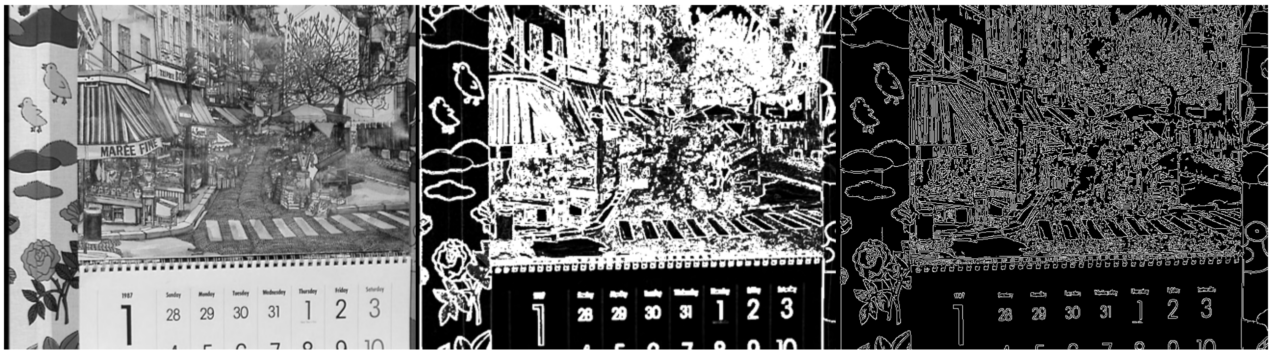

(a) (b) (c)

Figure 1.2: The gray image and its edge maps (a) The gray image; (b) Edge map computed by local variance; (c) Edge map computed by Canny detection.

produce sharper and more precise edges, thereby enhancing the visual appeal of resulting images and video frames. Subsequently, we illustrate this concept using two prevalent distance-based loss formats – the L1 norm and the Charbonnier norm. These loss functions align with those utilized in the SISR and VSR GAN models discussed in this chapter, as detailed in the subsequent subsection. The idea can be easily used for other distance-based loss formats in the same way.

The L1 norm distance loss is defined as:

$$l^1(u, v) = \sum_k \sum_i \sum_j |(u_{k,i,j} - v_{k,i,j})|, \tag{1.4}$$

and the Charbonnier distance loss is defined as:

$$\gamma(u, v) = \sum_k \sum_i \sum_j \sqrt{(u_{k,i,j} - v_{k,i,j})^2 + \epsilon^2}, \tag{1.5}$$

for the general case of two multi-channel images u and v, with elements $u_{k,i,j}$ and $v_{k,i,j}$ respectively. Index k is the channel number, e.g., k = 1 for a gray-scale image, $k \in \{1, 2, 3\}$ for a color image, and $k \in \{1, 2, 3...\}$ for an image defined in the feature space in the DL settings. ϵ is a small constant close to zero, for example, ϵ= 0.001.

The idea for distance losses is straightforward, involving the computation of dissimilarity between two images. In the context of constructing training loss for Super-Resolution Deep Learning (SR DL) models, these losses quantify disparities between synthesized SR images and ground truth HR counterparts, computed within both pixel and feature spaces. Thus, the objective of SR models is to minimize these losses, ensuring SR images closely resemble their HR counterparts.

A limitation of the defined distance-based losses is that they assign equal weight to all elements in images. However, we would like to preserve the edge information in the SR images as much as possible. Towards this end, during training the models, regions of high spatial activity areas (edge areas) should be weighted heavier than the smooth regions. Thus, we propose the following modification of the L1 and Charbonnier distance-based losses:

$$l_w^1(u, v, H) = \sum_k \sum_i \sum_j h_{k,i,j} |(u_{k,i,j} - v_{k,i,j})|, \tag{1.6}$$

$$\gamma_w(u, v, H) = \sum_k \sum_i \sum_j h_{k,i,j} \sqrt{(u_{k,i,j} - v_{k,i,j})^2 + \epsilon^2}, \tag{1.7}$$

where H is a weight matrix, including the set of weights $h_{k,i,j}$, it weighs the difference between u and v at position (k, i, j).

The following will show how to include the spatial activity information into the pixel loss by using the above-modified L1 and Charbonnier losses.

Using Equation 1.6 and 1.7, the original pixel losses for training the SR models are used to be defined as:

$$L_{pixel}^{l1} = l_w^1(hr, sr, \alpha\mathbf{1}), \tag{1.8}$$

$$L_{pixel}^{\gamma} = \gamma_w(hr, sr, \alpha\mathbf{1}), \tag{1.9}$$

where hr is the HR image, and sr is the corresponded SR image. $\mathbf{1}$ is a weight matrix whose elements are all 1. The coefficient α is a hyper-parameter, determining the weight of the pixel loss term. Such hyper-parameters balance different loss terms in the final training loss. As previously mentioned, the original pixel loss functions uniformly treat all image elements, but we want to stress the edge areas during training. We then propose the spatially adaptive (SA) pixel losses which bring the edge information in, defined as:

$$L_{SA-pixel}^{l1} = l_w^1(hr, sr, \beta W(hr)), \tag{1.10}$$

$$L_{SA-pixel}^{\gamma} = \gamma_w(hr, sr, \beta W(hr)), \tag{1.11}$$

where $W(hr)$ is the weight matrix described in Section 1.2.1, i.e., the edge map we extracted from the input HR image hr using either local variance or canny methods, and its elements are weights $w_{k,i,j}$. Recall what we have learned above about $W(hr)$: in areas of high spatial activity like edge regions, we have $w_{k,i,j}$ equals 1 or close to 1, depending on which edge extraction methods we choose. In contrast, in the flat areas, we have $w_{k,i,j}$ equals 0 or close to 0. Therefore, the values of $w_{k,i,j}$ in the edge regions are larger than the values in the flat regions. The larger the value of $w_{k,i,j}$ is, the more important the corresponding pixel (at position (k, i, j)) becomes in the function to be optimized. Because the difference calculated from these edge pixels will contribute a larger loss value to the total loss than the difference computed from the smooth pixels. Therefore, to minimize the loss value, during training, our models will consider these edge pixels as more important pixels and put more effort into recovering them. In other words, during the backward pass, the trainable parameters will be updated "consciously" towards the direction where more accurate edges can be super-resolved, and larger weight updates will be given to those parameters responsible for super-resolving these edge-like regions. Again, β is a hyper-parameter, serving as the coefficient of the

SA pixel loss used in the final training loss.

However, we still have one problem left for the SA pixel losses defined in Equation 1.10 and 1.11. When we are using the edge map $W(hr)$ as our weight matrix, in the flat areas, we have $w_{k,i,j}$ equals 0 or close to 0, which means we almost completely ignore to super-resolved the flat regions, and this is not appropriate. Therefore, in practice, we consistently retain a certain level of original pixel loss when forming the SA pixel loss, so the final SA pixel losses we used are:

$$L^{l1}_{SA-pixel} = l^1_w(hr, sr, \alpha\mathbf{1} + \beta W(hr)), \tag{1.12}$$

$$L^{\gamma}_{SA-pixel} = \gamma_w(hr, sr, \alpha\mathbf{1} + \beta W(hr)), \tag{1.13}$$

and the reason for including the $\alpha\mathbf{1}$ term is to ensure that the loss does not ignore flat regions in the images.

1.2.3 GAN Loss and Perceptual Loss

Following the state-of-the-art methods in super-resolution for both classical and perceptual loss functions [15], [29] we use a GAN-based training to produce frames of high perceptual quality.

Adapting the GAN formulation first introduced in [10] to VSR results in solving the adversarial min-max problem

$$\begin{aligned}\min_{\theta}\max_{\phi} L_{GAN}(\phi,\theta) = \mathbb{E}_x\left[\log D_\phi(hr)\right] + \\ \mathbb{E}_Y\left[\log\left(1 - D_\phi\left(G_\theta\left(lr\right)\right)\right)\right],\end{aligned} \tag{1.14}$$

where lr denotes the LR input, D_ϕ refers to the discriminator with trainable parameters ϕ, and G_θ represents the generator network with trainable parameters θ. Therefore, $G_\theta(lr)$ represents the

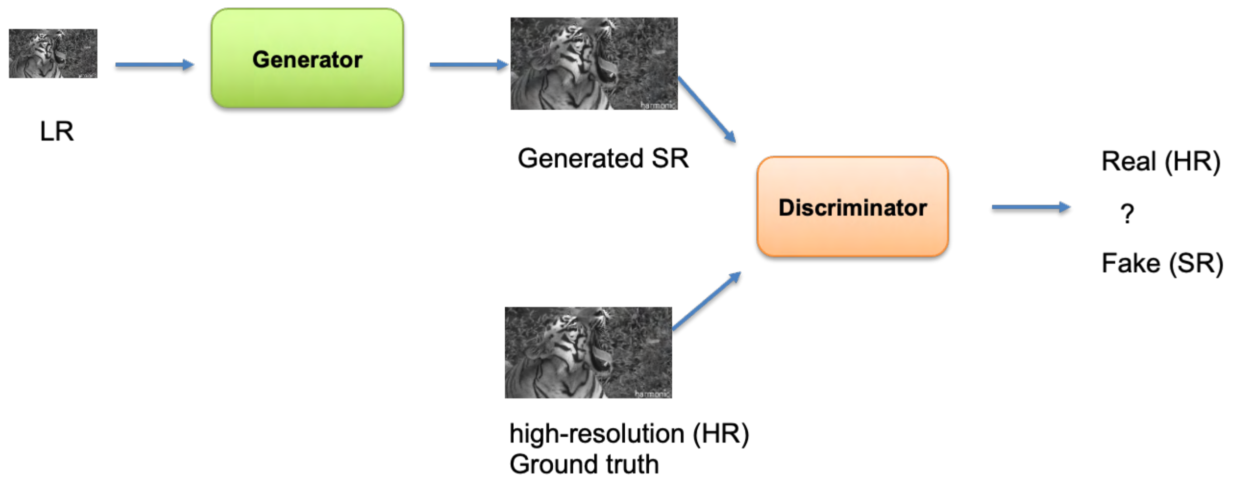


Figure 1.3: Super-resolution with GANs.

output of the generator, which corresponds to the super-resolved image sr.

Because the GAN-based model could be unstable during training and is not easy to converge appropriately. Typically, training the generator can be more challenging than training the discriminator. Therefore, in addition to the adversarial loss, two other distance-based losses, namely the pixel and perceptual loss, are commonly employed when training the generator, as depicted in Figure 1.4.

We have defined the pixel loss, which calculates the distance between the generated SR image and the HR image in pixel space. Regarding the perceptual loss, it measures the distance between the generated SR image and the HR image in a perceptual space, which is defined by the middle layer output of a pre-trained discriminative CNN model when taking the HR and SR as inputs, respectively:

$$L_{percep} = \gamma(\psi(hr), \psi(G_\theta(lr))), \tag{1.15}$$

where the feature space denoted as $\psi(\cdot)$ is computed from the activations of the intermediate layers of the VGG network [30]. Without loss of generality, we adopt the Charbonnier distance format

γ, which can be easily adapted to other distance formats.

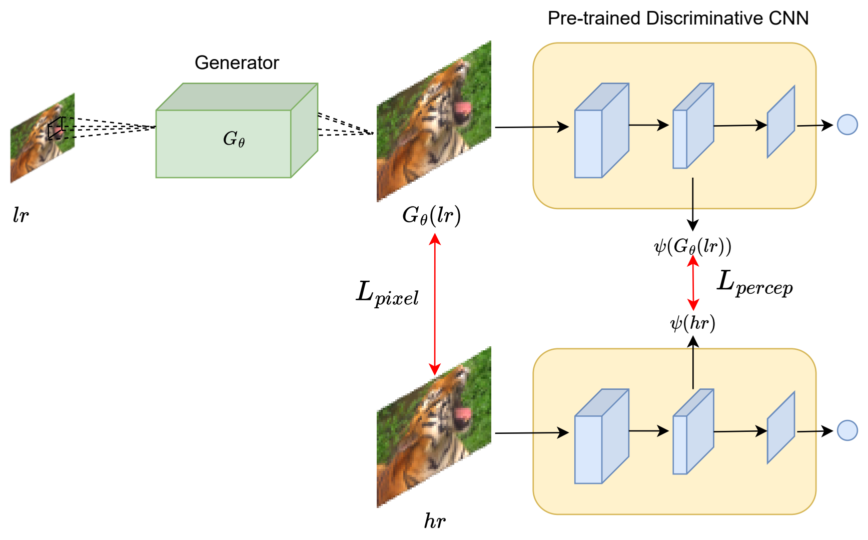


Figure 1.4: Pixel loss and feature loss visualization. Adapted from [31].

1.2.4 Spatially Adaptive Loss used in Super-Resolution Models

The spatially adaptive loss can be effectively used to improve the results regardless of the chosen SR models. In this chapter, we use two widely utilized SR GAN models as illustrative instances to apply the SA loss, one for image SR (ESRGAN [13]), the other for video SR (VSRResFeatGAN [14]). They both adopted the GANs framework and used perceptual loss during training. We will show that enhancing their initial training loss with the SA loss leads to a subsequent augmentation in the visual quality of the output results. The results comparison will be shown in Section 1.4.

1) SA loss for single image super-resolution model: ESRGAN adopts the GAN framework. Accordingly, the model comprises two networks: a generator and a discriminator. The generator's architecture is shown in Figure 1.5. It takes an LR image as input and outputs the corresponding SR image. The discriminator's architecture is shown in Figure 1.6. Its input is either an HR image

or an SR image generated from the generator, and its output is the probability of whether the input is a genuine HR image or otherwise. More details can be found in [13].

The original training loss used to train the ESRGAN contains the adversarial(GAN) loss, the perceptual loss, and the pixel loss [13]:

$$Loss^{esr} = \alpha L_{GAN} + L_{percep} + L_{pixel}, \tag{1.16}$$

and the pixel loss used here is:

$$l_w^1(x, G_\theta(y), \beta\mathbf{1}). \tag{1.17}$$

It follows the definition from Equation 1.8, where x is the ground truth HR image, y is the corresponding LR image, and it is also the input of generator network G with trainable parameter θ. Therefore $G_\theta(y)$ represents the output SR image. This pixel loss evaluates the L1 norm distance between the super-resolved image $G_\theta(y)$ and the ground truth image x in the pixel space, and all elements in the images are equally weighted. The coefficients assigned to the GAN and pixel loss terms are represented as α and β, respectively. The coefficient for the perceptual loss term is fixed to 1.

The new training loss we used to retrain the ESRGAN replaces the original pixel loss with the proposed SA one and remains the same adversarial loss and perceptual loss used in [13]. Therefore, the final SA training loss we used to train the model is defined as:

$$Loss^{SA-esr} = \alpha L_{GAN} + L_{percep} + L_{SA-pixel}, \tag{1.18}$$

and the SA pixel loss used here is:

$$l_w^1(x, G_\theta(y), \beta_1 \mathbf{1} + \beta_2 W(x)), \tag{1.19}$$

which follows the definition of SA pixel loss from Equation 1.12. We name the model trained with this SA training loss (Equation 1.18) the SA-ESRGAN.

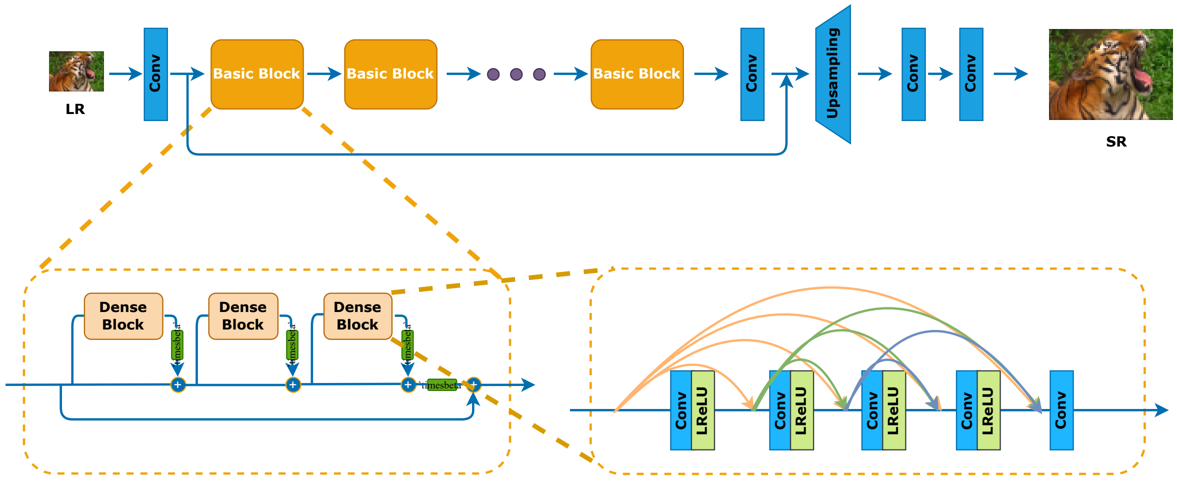


Figure 1.5: The ESRGAN's generator architecture. It consists of a convolutional (conv) operation applied to the input image, followed by 23 basic blocks and a conv layer. A long skip connection is used, where the output feature maps will add with the output feature maps from the first conv operation. After this, up-sampling and two more conv operations are conducted to obtain the final SR image output. The basic block used here is Residual-in-Residual-Dense-Block (RRDB), which combines the multi-level residual network and dense connections. [13]

2) SA loss for video super-resolution model: VSRResFeatGAN model also contains a generator and a discriminator. The generator's architecture is shown in Figure 1.7. Since the VSRResFeatGAN is used for video super-resolution task, it also takes temporal information in. Rather than using a single frame, its generator takes five consecutive frames as input, which are the bicubic interpolation on the LR frames at times t-2, t-1, t, t+1, and t+2. The resulting output is the corresponding SR frame at the central time instance, t. The discriminator's architecture is shown

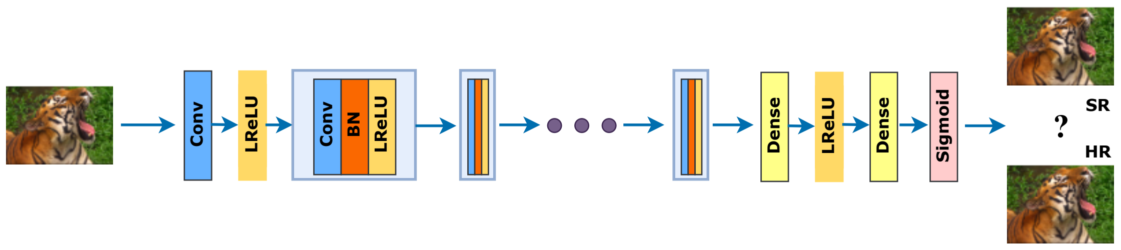


Figure 1.6: The ESRGAN's discriminator architecture. It consists of conv, Leaky ReLU, and batch normalization layers along the way, followed by two fully connected layers and a sigmoid activation function to obtain a probability.[13]

in Figure 1.8. Its input is either an HR frame or an SR frame generated from the generator, and its output is the probability of whether the input is a real HR frame or not. More details can be found in [14].

The original training loss used to train the VSRResFeatGAN also contains three terms: the adversarial(GAN) loss, the perceptual loss, and the pixel loss [14]:

$$Loss^{vsr} = \alpha_1 L_{GAN} + \alpha_2 L_{percep} + L_{pixel}, \tag{1.20}$$

and the pixel loss used here is:

$$\gamma_w(x, G_\theta(Y), \beta\mathbf{1}), \tag{1.21}$$

it follows the definition from Equation 1.9, where x is the ground truth HR frame. Y is the bicubic-interpolated frames sequence, and it is also the input of the generator network G with trainable parameter θ, and $G_\theta(Y)$ represents the output SR frame at time t. This pixel loss evaluates the Chabonnier norm distance between the super-resolved frame $G_\theta(Y)$ and the ground truth frame x in the pixel space, and all elements in the images are equally weighted. α_1, α_2, and β are the coefficients for GAN, perceptual, and pixel loss terms.

Like what we have done with the SISR model (ESRGAN), the new training loss we used to retrain the VSRResFeatGAN replaces the original pixel loss with the corresponding SA one and remains the same adversarial loss and perceptual loss used in [14]. Therefore, the final SA training loss we used to train the model is defined as:

$$Loss^{SA-vsr} = \alpha_1 L_{GAN} + \alpha_2 L_{percep} + L_{SA-pixel}, \tag{1.22}$$

and the SA pixel loss used here is:

$$\gamma_w(x, G_\theta(Y), \beta_1 \mathbf{1} + \beta_2 W(x)), \tag{1.23}$$

which aligns with the definition of the SA pixel loss from Equation 1.13. We name the model trained with this SA training loss (Equation 1.22) the SA-VSRResFeatGAN.

1.3 Experiments

This section shows the details of training Single Image Super-Resolution models, including ESRGAN and SA-ESRGAN, as well as the Video Super-Resolution models, comprising VSRResFeatGAN and SA-VSRResFeatGAN.

1.3.1 Datasets

The training dataset used for ESRGAN [13] and SA-ESRGAN is DIV2K [32], and the validation dataset used for them is Set14 [33]. The dataset used for VSRResFeatGAN [14] and SA-VSRResFeatGAN is Myanmar video dataset [34]. Myanmar video dataset contains 59 video sequences, and we take 53 of them to make the training (80%) and validation (20%) datasets. The rest 6 are used for testing. We followed the same data generation approaches described in [13] and [14], respectively. All experiments are performed under the scale factor 4 between LR and HR image pairs.

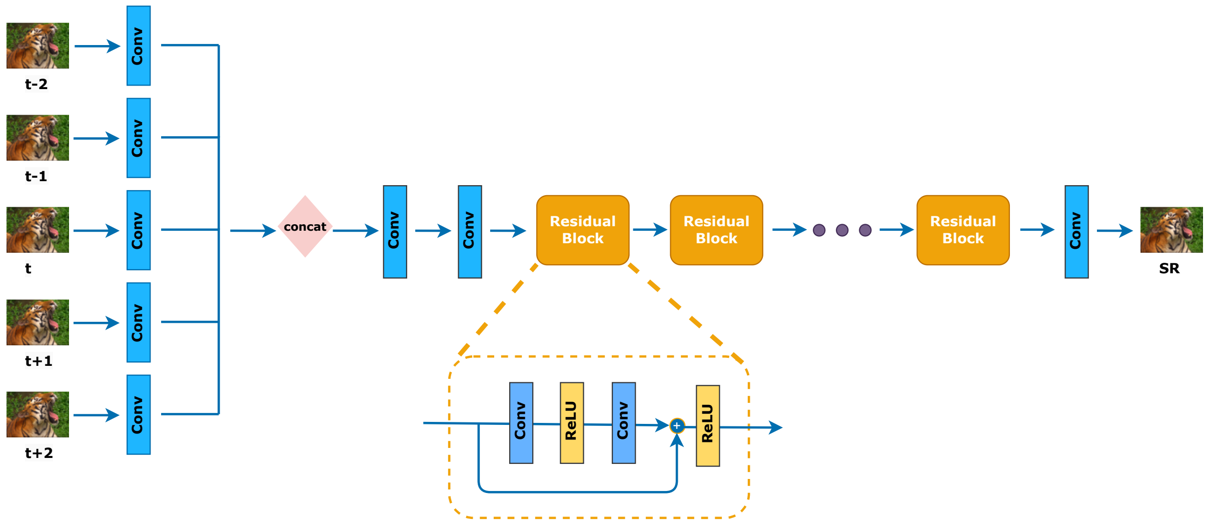


Figure 1.7: The VSRResFeatGAN's generator architecture. In the beginning, 5 conv operations are applied separately on each input bicubic interpolated frame at time t-2, t-1, t, t+1, t+2. The resulting feature maps are then concatenated together, and the combined feature map will be inputted into the following two conv layers and 15 residual blocks. Inside each residual block, there are two conv layers, each followed by a ReLU layer, and the input feature map is added to the output feature map to obtain the final output of the residual block. In the end, one more conv operation is conducted to obtain the final SR frame output at time t.[14]

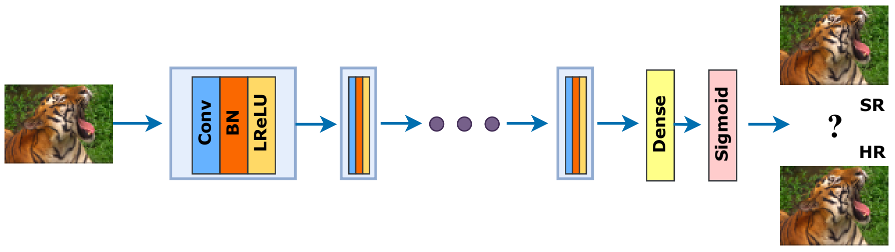


Figure 1.8: The VSRResFeatGAN's discriminator architecture. It consists of conv, batch normalization, and Leaky ReLU layers along the way, followed by one fully connected layer and a sigmoid activation function to obtain a probability. [14]

1.3.2 Training Process

To train the ESRGAN and SA-ESRGAN models, we first initialize their generators with the PSNR-oriented pre-trained model, provided by the author of ESRGAN [13]. Such initialization facilitates proper convergence during the following GAN-based training phase. Within the GAN-based training, the ESRGAN model's generator is optimized using the loss function defined in Equation 1.16, while the SA-ESRGAN model's generator is optimized using the loss function defined in Equation 1.18. For both models, the learning rates for the generator and discriminator are initialized to 10^{-4}, then halved at [50k, 100k, 200k, 300k] iterations (k = 10^3). To guarantee fair comparisons, we set the maximum training iteration number as 500k and use the minimum validation loss as the termination criterion for both models. The validation loss is computed by Equation 1.16 and Equation 1.18 on the validation dataset, respectively. The optimizer we use is Adam [35] with batch size 16. The generator and discriminator are updated alternately.

The training processes of the VSRResFeatGAN and SA-VSRResFeatGAN models are similar. We first initialize the generator with the PSNR-oriented pre-trained model, provided by the author of VSRResFeatGAN [14]. In the following GAN-based training process, for the VSRResFeatGAN model, the generator is trained using the loss function in Equation 1.20, and for the SA-VSRResFeatGAN model, the generator is trained using the loss function in Equation 1.22. For both models, the learning rates for the generator and discriminator are set to 10^{-4}. We set the maximum training epoch number to 40 and also use the minimum validation loss as the training termination criterion for both models. The validation loss is computed using Equation 1.20 and Equation 1.22 on the validation dataset, respectively. The optimizer we use is Adam with batch size 64. The generator and discriminator are updated alternately.

1.3.3 Hyper-Parameters for the Loss Functions

For training the ESRGAN and VSRResFeatGAN models, we choose the same coefficients used in [13] and [14] respectively: α = 0.005, β = 0.01 in Equation 1.16 and 1.17 when training the ESRGAN model. α_1 = 0.001, α_2 = 0.998 and β = 0.001 in Equation 1.20 and 1.21 when training the VSRResFeatGAN model.

During the training of the SA-ESRGAN and SA-VSRResFeatGAN models, we have determined that when the SA loss component constitutes approximately 15% of the total loss value, it exhibits the ability to exert a substantial and appropriate influence on the generator, leading to the production of sharper and more distinct edges in the generated outcomes. Based on this observation, we have set the coefficients for the loss terms as follows: for SA-ESRGAN model, the generator is trained employing the loss outlined in Equation 1.18 and Equation 1.19, with α_1 = 0.005, β_1 = 0.01, and β_2 = 20. For SA-VSRResFeatGAN model, its generator is trained using the loss function outlined in Equation 1.22 and Equation 1.23, with α_1 = 0.001, α_2 = 0.998 and β_1 = 0.001, and we set β_2 = 5 when using the local variance edge extraction method to compute weight matrix W, while β_2 = 1.5 when using the canny edge extraction method. In both models, β_2 governs the influence of the SA loss component; its values are decided experimentally so that the SA loss part took up around 15% of the total training loss, respectively.

1.4 Results

In this section, we show the impact of the proposed SA loss on the outcomes. By doing so, we will provide both qualitative and quantitative comparisons between the SISR model and the VSR model, trained both with and without the SA loss, i.e., ESRGAN vs. SA-ESRGAN and VSRResFeatGAN vs. SA-VESRResFeatGAN.

In Table 1.2 and 1.1, we show the comparison of ESRGAN and SA-ESRGAN in terms of

PSNR, SSIM, and the Learned Perceptual Image Patch Similarity (LPIPS) [36] metrics for VidSet4 and Myanmar test datasets. LPIPS is a new standard to compare the perceptual similarity between a reference image and a distorted one with a CNN. The author found out this perceptual distance provides results consistent with human judgment [37]. We also observed that it was consistent with the human's opinion regarding the sharpness of the produced super-resolved images. For the SA-ESRGAN model, based on the edge detection algorithm used to compute the weight matrix $W(x)$ in Equation 1.19, we categorize the model into SA-ESRGAN (lv) and SA-ESRGAN (canny), i.e., they are trained with the local variance-based SA loss and the Canny detection–based SA loss, respectively. We can see that for the Vidset 4 test dataset (Table 1.2), the SA-ESRGAN models surpass the ESRGAN model across all metrics. Similar trends emerge for the Myanmar test dataset (Table 1.1), where all SA-ESRGAN models outperform the ESRGAN model in terms of the LPIPS metric. This fact verifies our proposal that using the SA loss for the training can help guide the SISR model to generate images with higher perceptual quality. The qualitative comparisons are shown in Figure 1.9-1.10. We can see that the SA-ESRGAN yields more precise super-resolution of edges and fine details with reduced noise than the ESRGAN.

Similarly, for the VSR models, we show the quantitative comparison of VSRResFeatGAN and SA-VSRResFeatGAN in Table 1.3 and 1.4. Again, SA-VSRResFeatGAN models surpass the VSRResFeatGAN model almost for all the metrics' values. Especially for the LPIPS metric, it consistently improves after SA loss is used, which verifies our proposal that the use of the SA loss during training can guide the VSR model towards generating images with higher perceptual quality and better visual feelings. The qualitative comparisons are shown in Figure 1.11. We can observe that the SA-VSRResFeatGAN generates more accurate super-resolved edges and fine details with less noise compared with the VSRResFeatGAN.

	PSNR	**SSIM**	**LPIPS**
ESRGAN	29.13	0.8445	0.0398
SA-ESRGAN (lv)	29.23	0.8444	0.0387
SA-ESRGAN (canny)	**29.75**	**0.8561**	**0.0357**

Table 1.1: Metrics comparison between ESRGAN, SA-ESRGAN (local-variance), and SA-ESRGAN (canny). The results are evaluated on Myanmar testing Dataset for scale factor 4.

	PSNR	**SSIM**	**LPIPS**
ESRGAN	21.53	0.6349	0.1022
SA-ESRGAN (lv)	22.14	0.6581	**0.0953**
SA-ESRGAN (canny)	**22.32**	**0.6674**	0.0955

Table 1.2: Metrics comparison of ESRGAN, SA-ESRGAN (local-variance), and SA-ESRGAN (canny) in PSNR, SSIM, and LPIPS metrics. For the LPIPS, smaller is better. The results are evaluated on VidSet4 Dataset for scale factor 4.

	PSNR	**SSIM**	**LPIPS**
VSRResFeatGAN	24.71	0.7199	0.1045
SA-VSRResFeatGAN (lv)	**25.09**	**0.7344**	**0.0992**
SA-VSRResFeatGAN (canny)	24.74	0.7192	0.1027

Table 1.3: Metrics comparison of VSRResFeatGAN, SA-VSRResFeatGAN (local-variance), and SA-VSRResFeatGAN (canny) in terms of PSNR, SSIM, and LPIPS metrics. For the LPIPS, smaller is better. The results are evaluated on VidSet4 Dataset for scale factor 4.

	PSNR	**SSIM**	**LPIPS**
VSRResFeatGAN	32.22	0.8887	0.0545
SA-VSRResFeatGAN (lv)	**32.48**	**0.8918**	0.0540
SA-VSRResFeatGAN (canny)	32.36	0.8890	**0.0539**

Table 1.4: Metrics comparison between VSRResFeatGAN, SA-VSRResFeatGAN (local-variance), and SA-VSRResFeatGAN (canny). The results are evaluated on Myanmar testing Dataset for scale factor 4.

1.5 Conclusion

In this chapter, we introduced the Spatially Adaptive (SA) loss for training SR models, incorporating spatial information extracted from input images/video frames into the training process. We specifically emphasize edges as a crucial spatial feature. Models trained with the SA loss exhibit the ability to produce super-resolved images/video frames with improved perceptual quality.

We have shown that the proposed method is independent of the edge information extraction approach, as long as it effectively detects the edges. Utilizing extracted edge information to construct the SA loss and retraining the SR model accordingly consistently yielded improved results. These enhancements were evident in better metric values, sharper edges, improved reconstruction of fine details, and a notable reduction in noise. Additionally, we have shown that the SA loss benefits both SISR and VSR GAN models, consistently enhancing their performance. This underscores the adaptability and effectiveness of our proposed SA loss across various scenarios, guiding SR models toward generating visually appealing outcomes.

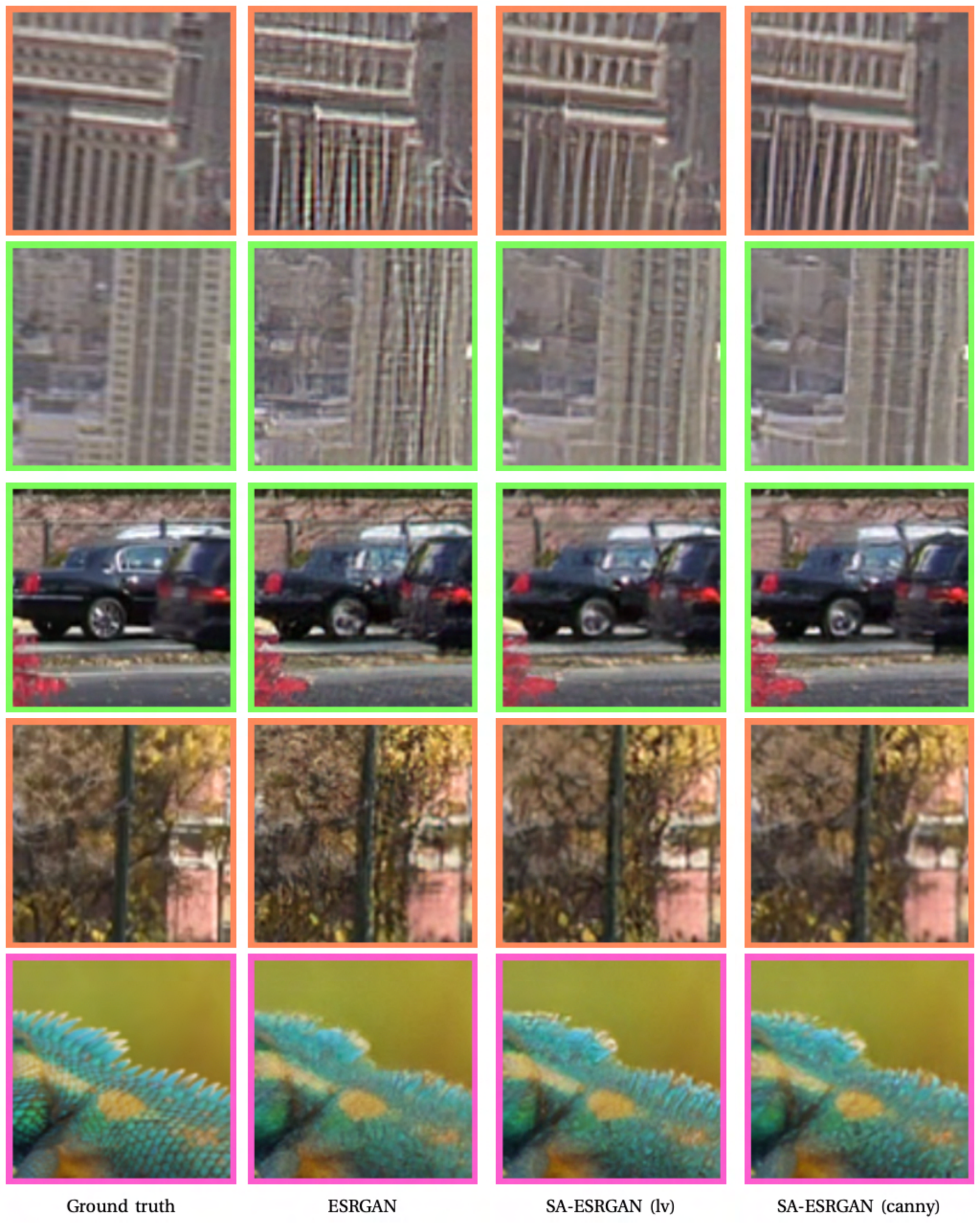


Figure 1.9: Qualitative comparison of ESRGAN, SA-ESRGAN(lv), and SA-ESRGAN(canny).

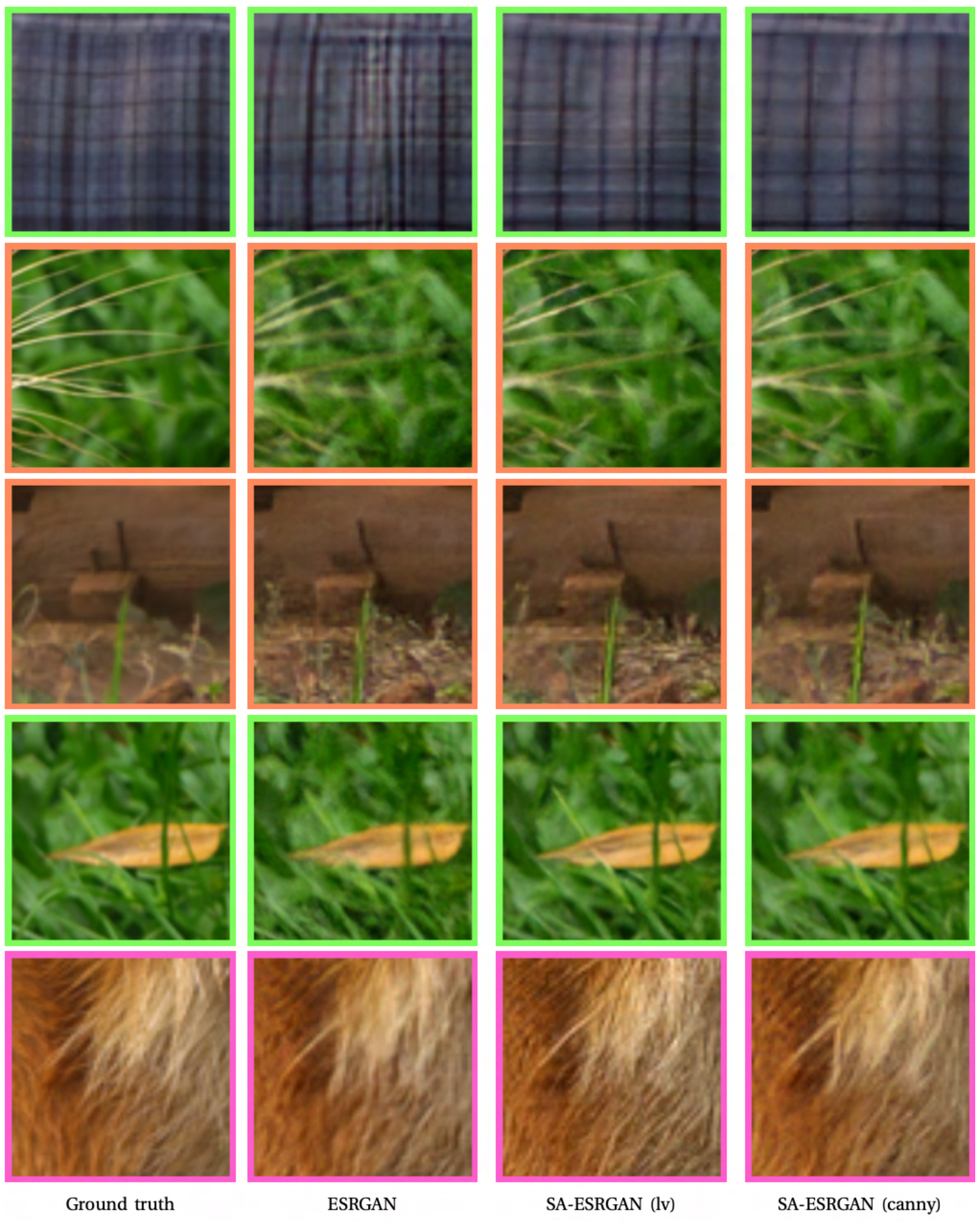


Figure 1.10: Qualitative comparison of ESRGAN, SA-ESRGAN(lv), and SA-ESRGAN(canny).

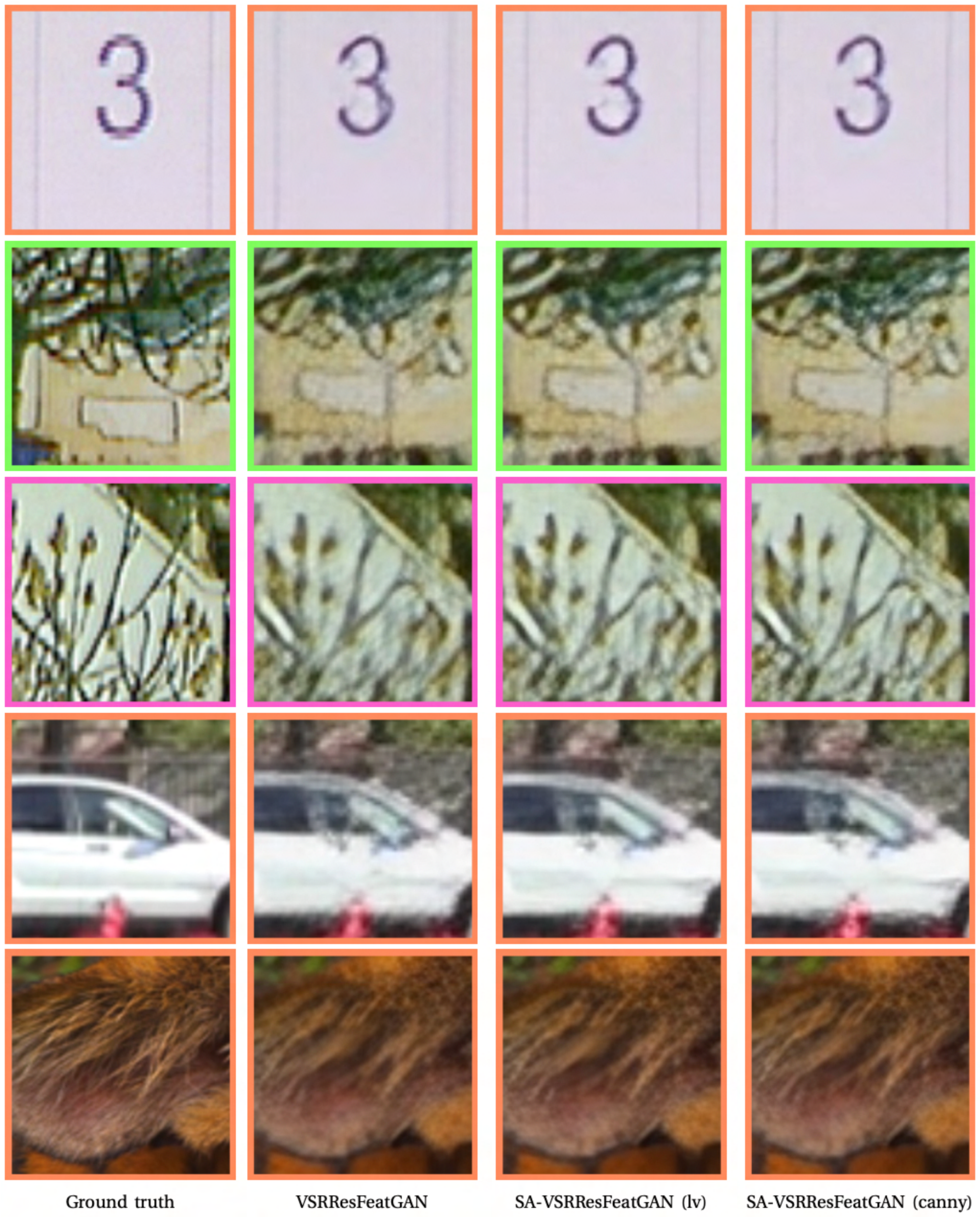


Figure 1.11: Qualitative comparison of VSRResFeatGAN, SA-VSRResfeatGAN(lv), and SA-VSRResfeatGAN(canny).

CHAPTER 2
COMPOSITE DISCRIMINATORS FOR GAN-BASED VIDEO SUPER-RESOLUTION

Besides including the spatial information through training loss, we also discovered to incorporate it through the model framework. We design a second discriminator tailored to discern edge information. By employing this additional discriminator, we exert more pressure on our generator to produce edge information closer to reality. The alterations in the loss function are primarily in the adversarial losses due to the inclusion of the second discriminator, which focuses specifically on the edges of input images. (Please note that this chapter revisits text and figures from our previously published work [38].)

2.1 Introduction

One of the fundamental problems in image and video processing is Video Super-Resolution (VSR), aiming to restore High-Resolution (HR) video sequences from Low-Resolution (LR) counterparts. Recent advancements in Super-Resolution (SR) suggest that learning-based methods yield more realistic images compared to model-based techniques [39], [40]. Deep Neural Networks have emerged as the preferred tool for these learning-based approaches [41]–[43].

Generative Adversarial Networks (GANs) [44], capable of learning complex distributions from samples, have gained attention in the VSR domain. Researchers adopt GAN-based training over classical Mean Squared Error (MSE) to encourage networks to produce solutions resembling natural videos [45]–[48]. Many GAN-based approaches integrate feature-based perceptual losses to enhance the perceptual quality of frames. Furthermore, recent GAN-based VSR methods enhance their performance by integrating useful information, such as spatial [47] or temporal information

[45], [46], [48], during training. For instance, findings from [47] demonstrate that integrating spatial information into the training objective function (pixel and perceptual losses) enhances the sharpness of frames while reducing artifacts and noise. These approaches typically augment the training objective function by either adding or enhancing loss terms [46], [47], or by refining the generator network [45], [46], [48]. No published findings in VSR have attempted to integrate this information by enhancing the discriminator's capability significantly. GAN-based VSR approaches have predominantly employed the traditional two-player GAN framework introduced in [44], comprising a single generator and discriminator in a min-max game. Recent GAN studies [49], [50] have proposed alternative GAN frameworks, demonstrating their effectiveness in overcoming several modeling limitations of traditional GANs.

In this chapter, we introduce a novel GAN framework for addressing VSR challenges. During training, a composite discriminator is employed. The model leverages spatial information to generate frames with enhanced edge reconstruction and visual quality. The convergence of the proposed approach is established. By training with this new GAN framework, we outperform current state-of-art methods [45], [47], [48] which are trained using the traditional GAN framework with no additional spatial information included in the discriminator.

2.2 Proposed Method

The model proposed in [45], [48] sometimes produces SR frames with blurred edges and noise in high-frequency areas. A limitation of this model is that the spatial activity of image regions is not specifically taken into account during training. It is, however, clear that edge information plays a very important role in the quality of the reconstruction and that edge regions are more difficult to super-resolve than flat-regions. We propose the use of a composite discriminator which includes a novel edge sharpness enforcing collaborative discriminator to obtain realistic edges. Our model makes use of high frequency information extracted from frames in our training datasets and forces the generator to specifically take into account edge (high spatial activity) areas. By doing so, the generator is forced to produce crisper edges and fewer artifacts.

2.2.1 A Composite Discriminator

GANs [44] learn to generate samples from a specific data distribution through an adversarial training procedure. In the traditional GAN approach for image generation, a *generator* network learns to generate an image given a latent random vector z at its input. The learning of the generator is guided by an auxiliary network, a *discriminator*, which is simultaneously trained to distinguish between images generated by the generator and images from the training dataset. Given a generator $G(\mathrm{z})$, with latent variable z to be defined later, the discriminator is trained to distinguish between real and fake images, i.e., it outputs $D(\mathrm{x}) = 1$ when x is sampled from the training dataset of natural images and $D(G(\mathrm{z})) = 0$ when the images are produced by the generator. On the other hand, the generator is trained to make the discriminator believe that its generated images $G(\mathrm{z})$ are real, i.e., trained to assign to the discriminator output a probability $D(G(\mathrm{z})) = 1$. As a result of this adversarial training, the generator eventually converges to a solution which the discriminator fails to identify as ”fake”, which generally implies successful learning of the image manifold by the generator.

In [45], [48] we propose the use of the powerful generative property of GANs in VSR. Using GAN-based instead of MSE-based training enables the models to obtain frames of much higher perceptual quality. The original GAN setting was modified by inputting the sequence of input low-resolution frames Y to the generator instead of a random vector z. This is similar to the use of GANs in still image super-resolution [51], where a single LR image is provided as input to the generator. The generator is adversarially trained to super-resolve the input LR frames in a way that the discriminator cannot distinguish between the reconstructed HR frames, $\hat{\mathrm{x}} = G(\mathrm{Y})$ and real HR images. The GAN formulation first introduced in [44] was adapted to VSR by solving:

$$\begin{aligned}\min_{\theta}\max_{\phi} \mathrm{L}_{\mathrm{GAN}}(\phi,\theta) = &\mathrm{E}_{\mathrm{x}}[\log \mathrm{D}_{\phi}(\mathrm{x})] \\ &+ \mathrm{E}_{\mathrm{Y}}[\log(1-\mathrm{D}_{\phi}(\mathrm{G}_{\theta}(\mathrm{Y})))],\end{aligned} \tag{2.1}$$

where x is the center HR frame of dimensions $N \times N$, Y is a short sequence of LR input frames around its LR version y, each of dimensions $N \times N$ (notice that the LR images are bicubically upsampled), D_{ϕ} is the discriminator network with trainable parameters ϕ and G_{θ} is the generator network with trainable parameters θ, where here these parameters correspond to the learnable convolutional kernels of our networks.

The above model is the basis of all the so far based GAN formulation, but a close look at the optimization function indicates that the better discrimination could be achieved by using a model of the form

$$\begin{aligned}&\min_{\theta}\max_{\phi,\phi'} \mathrm{L}_{\mathrm{GAN}}(\phi,\phi',\theta) = \mathrm{E}_{\mathrm{x}\sim\mathrm{p}_{\mathrm{x}}(\mathrm{x})}[\log(\mathrm{D}_{1\phi}^{\lambda}(\mathrm{x})\mathrm{D}_{2\phi'}^{1-\lambda}(\mathrm{x}))] \\ &+ \mathrm{E}_{\mathrm{Y}\sim\mathrm{p}_{\mathrm{Y}}(\mathrm{Y})}[\log((1-\mathrm{D}_{1\phi}(\mathrm{G}_{\theta}(\mathrm{Y})))^{\lambda}(1-\mathrm{D}_{2\phi'}(\mathrm{G}_{\theta}(\mathrm{Y})))^{1-\lambda})],\end{aligned} \tag{2.2}$$

where two different discriminators $D_{1\phi}$ and $D_{2\phi'}$ with parameters ϕ and ϕ', respectively, are used and $0 < \lambda < 1$.

Notice that following the approach in [44], it can be easily shown that fixing θ

$$\begin{aligned}&\max_{\phi,\phi'} \mathrm{L}_{\mathrm{GAN}}(\phi,\phi',\theta) \\ &= \mathrm{E}_{\mathrm{x}\sim\mathrm{p}_{\mathrm{x}}(\mathrm{x})}\left[\log\frac{\mathrm{p}_{\mathrm{x}}(\mathrm{x})}{\mathrm{p}_{\mathrm{x}}(\mathrm{x})+\mathrm{p}_{\mathrm{g}}(\mathrm{x})}\right] + \mathrm{E}_{\mathrm{x}\sim\mathrm{p}_{\mathrm{g}}(\mathrm{x})}\left[\log\frac{\mathrm{p}_{\mathrm{g}}(\mathrm{x})}{\mathrm{p}_{\mathrm{x}}(\mathrm{x})+\mathrm{p}_{\mathrm{g}}(\mathrm{x})}\right],\end{aligned} \tag{2.3}$$

where $p_g(\mathrm{x})$ is the probability distribution induced on x by Y. Furthermore, following the approach in [44] it can be shown that

$$D_{1\phi}(\mathrm{x}) = \mathrm{D}_{2\phi'}(\mathrm{x}) = \frac{\mathrm{p_x(x)}}{\mathrm{p_x(x)} + \mathrm{p_g(x)}}. \tag{2.4}$$

and also that the global minimum on θ in Eq. (2.2) is achieved when and only when $p_g(\cdot) = p_{\mathrm{x}}(\cdot)$. However, the network architecture of $D_{1\phi}(\mathrm{x})$ will prevent it from always satisfying Eq. (2.4). For instance, this discriminator may concentrate on detecting important image properties but may not be capable of detecting all of them. This is similar to the case when prior models are used to restore images. For instance, horizontal filters are used to regularize horizontal differences but these filers are insensitive to vertical ones. In GAN terminology, a discriminator may be good at detecting certain fake note properties (or even be good on some notes) but not all of them. In this chapter we approach the possible limited capabilities of $D_{1\phi}$ by introducing a second discriminator that, in our case, concentrates on the quality of high frequency areas and, in particular, on producing realistic edges. To do so we redefine $D_{2\phi'}(\cdot) = D_{2\phi'}(W\cdot)$, where W denotes a high pass filter. Notice that other definitions of $D_{2\phi'}(\cdot)$ are possible, but we will concentrate here on recovering high spatial activity areas.

Let us now provide a graphical description of our collaborative model. In Fig. 2.1, the W operator is a spatial information extractor. We use the method illustrated in [52][47] to define the W operator, which is consistent with the

masking property of the human visual system, according to which noise is visible in flat regions but not visible at edges. The output of the W operator is a weighted image (edge focusing map), where pixels in areas of high spatial activity, like edge regions, have much larger values than those

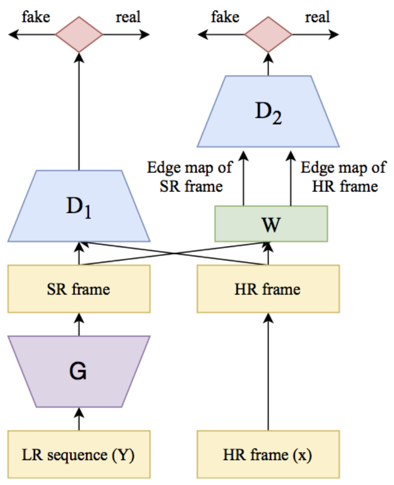


Figure 2.1: The proposed model.

in flat regions (in this work, all the pixel values are normalized to the range [0,1] during training). In the rest of the chapter, the weighted image generated after the application of operator W will be simply referred to as the edge map.

We adopt the VSRResNet architecture proposed in [45], [48] as our generator. The architecture is shown in Fig. 2.2. It is based on 15 residual blocks, each block containing two convolutional layers with kernels of size 3 by 3, with a Rectified Linear Unit (ReLU) activation function after each convolution step.

We adopt the same architecture for both discriminators D_{1_ϕ} and $D_{2_{\phi'}}$ (used in [48]), shown in Fig. 2.2. The network is composed of three convolution layers followed by a fully connected layer and a sigmoid operation. However, they are provided with different inputs. The input to discriminator D_{1_ϕ} are super resolved and HR frames, while the input to discriminator $D_{2_{\phi'}}$ are the corresponding edge maps of the super resolved and HR frames. Besides of fooling the original discriminator D_{1_ϕ}, the generator has also to fool the discriminator $D_{2_{\phi'}}$, so the edge map of the generated super resolved frame has to be realistic, close to the edge map of its corresponding HR frame. As a result, generated frames have more accurate edges with less noise and fewer

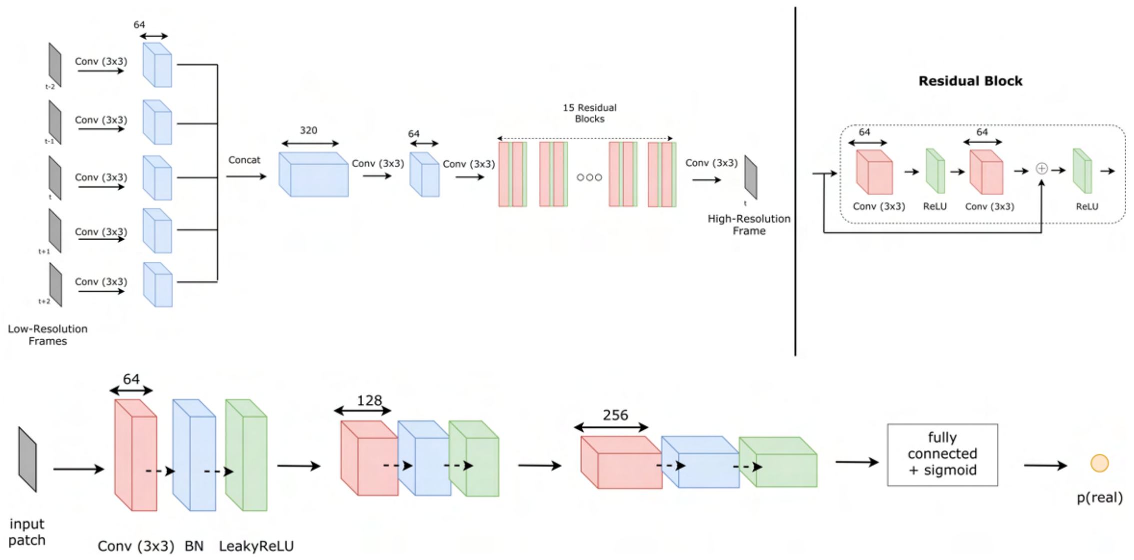


Figure 2.2: The proposed architecture for the generator (first row) and discriminator (second row) [48]

artifacts and both discriminators collaborate to obtain better images. We name our proposed model the Collaborative Discriminator GAN (CoDiGAN), when applied to VSR we will denote it by VSRCoDiGAN.

2.2.2 Pixel and Feature Losses

To regularize undesired artifacts that may escape the collaborative model, similar to Chapter 1, we use two distance-based regularizers, defined in pixel and feature spaces, respectively.

Let us consider the Charbonnier loss, defined as

$$\gamma(u,v) = \sum_k \sum_i \sum_j \sqrt{(u_{k,i,j} - v_{k,i,j})^2 + \epsilon^2}, \tag{2.5}$$

where u and v are multichannel images with elements $u_{k,i,j}$ and $v_{k,i,j}$, respectively, where k denotes channel (for instance, $k = 1$ for a gray-scale image and $k = 1, 2, 3$ for a color image), i, j

denotes pixel location and $\epsilon > 0$. The pixel-wise loss only depends on low-level pixel information, and it is defined as the Charbonnier loss of the difference of two frames in pixel space, that is, $\sum_{(\mathrm{x},\mathrm{Y})\in\mathrm{T}} \gamma\left(\mathrm{x}, \mathrm{G}_\theta\left(\mathrm{Y}\right)\right)$, where x and Y are sampled from the training dataset T. The perceptual loss in feature space computes differences between high-level image feature representations extracted from pre-trained convolutional neural networks. In this chapter, we choose our feature space to be the representation space obtained from extracting the feature maps from the third and fourth convolution layer of the VGG network defined in [53], denoted as $VGG\left(\cdot\right)$ in this chapter. Therefore, the feature loss is defined as $\sum_{(\mathrm{x},\mathrm{Y})\in\mathrm{T}} \gamma\left(VGG\left(\mathrm{x}\right), VGG\left(G_\theta\left(\mathrm{Y}\right)\right)\right)$. In the next section, we show the final loss for training the generator.

2.2.3 Final Loss for Generator

Combining the losses defined in the previous sections, our generator has to effectively minimize adversarial losses together with pixel and feature losses, thus our final loss function becomes:

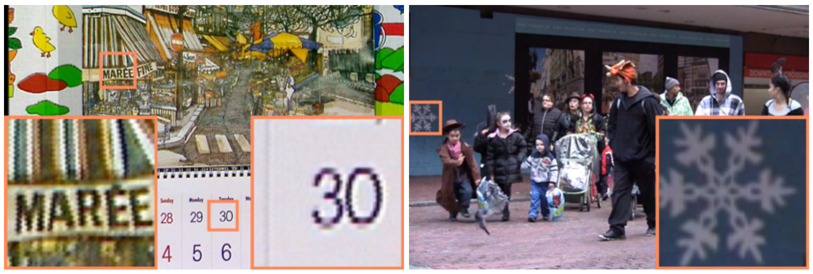
$$
\begin{aligned}
\mathrm{L}_{\text{final}}(\theta) = {} & \alpha_1 \left[\mathbb{E}_Y \left[-\log D_{1\phi}\left(G_\theta\left(\mathrm{Y}\right)\right) - \log D_{2\phi'}\left(W\left(G_\theta\left(\mathrm{Y}\right)\right)\right)\right]\right] \\
& + \alpha_2 \sum_{(\mathrm{x},\mathrm{Y})\in\mathrm{T}} \gamma\left(\mathrm{x}, G_\theta\left(\mathrm{Y}\right)\right) \\
& + \left(1 - \alpha_1 - \alpha_2\right) \sum_{(\mathrm{x},\mathrm{Y})\in\mathrm{T}} \gamma\left(VGG\left(\mathrm{x}\right),\ VGG\left(G_\theta\left(\mathrm{Y}\right)\right)\right),
\end{aligned} \tag{2.6}
$$

where $0 < \alpha_1, \alpha_2$ and $\alpha_1 + \alpha_2 < 1$, We have fixed λ to $1/2$. In the next section, we show that this model improves the quality of the super resolved video.

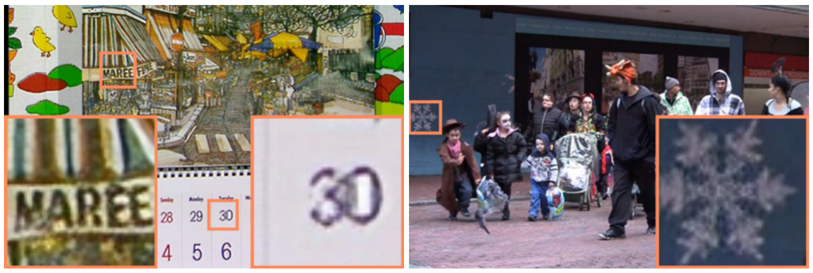
2.3 Experiments

We synthesized the training dataset of HR/LR-sequence pairs from the Myanmar video sequence. Our training dataset consists of nearly 1 million pairs, where each sample in the training dataset is composed of five extracted 36×36 LR patches at times $\mathrm{t} - 2$, $\mathrm{t} - 1$, t, $\mathrm{t} + 1$, and $\mathrm{t} + 2$, and its corresponding 36×36 HR patch at time t. The LR frames were computed using bicubic downsampling followed by bicubic upsampling to bring them to the same spatial extent as the original HR patch.

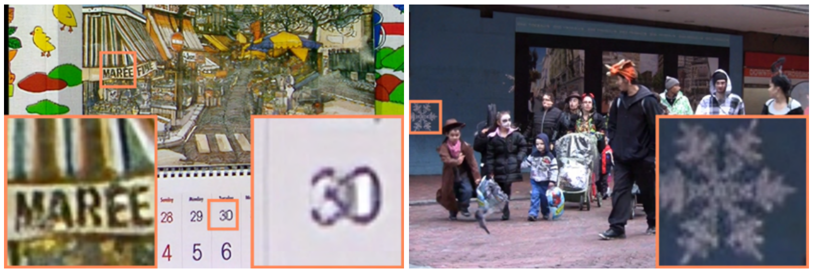
To ensure convergence of generator and discriminator loss functions, it is critical for the generator network to start at a reasonable θ at the beginning of the training [48]. Thus, prior to beginning the adversarial training, we first trained the generator network for 100 epochs with the traditional pixel based MSE loss using the ADAM [54] optimizer and a batch size of 64. For this pre-training, the initial learning rate was set to 10^{-3} and it was then further divided by a factor of 10 at the 50th and 75th epochs of the training. We train our generator for the scale factor 3. Using the weights of this pre-trained generator as initial weights, we trained our spatially adaptive collaborative GAN model with the loss functions defined in (2.6) for 30 epochs, setting the learning rate to 10^{-4} for the generator and both discriminators. The weight decay was set to 10^{-3} for the discriminators and 10^{-4} for the generator. We use the ADAM [54] optimizer and a batch size of 64. The values of α_1 and α_2 used in (2.6) were determined experimentally. We found their optimal values to be: $\alpha_1 = 0.0005$, $\alpha_2 = 0.001$. The parameter ϵ in (2.5) is set to 0.001. We found out that 30 epochs were appropriate for our model to converge.



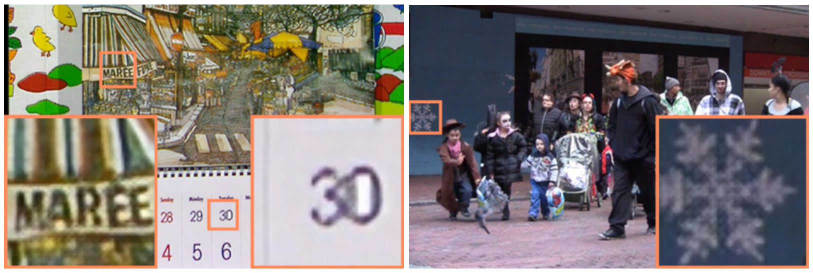
(a) Ground Truth



(b) VSRResFeatGAN



(c) SA-GAN



(d) VSRCoDiGAN

Figure 2.3: Qualitative comparison of results obtained by VSRResFeatGAN [48], SA-GAN [47], and VSRCoDiGAN on scale factor 3. The ground truth frames are shown in Fig. 2.3(a).

2.4 Results

As we have already indicated, we trained our model on the Myanmar dataset. In order to check whether our model could also work well in different datasets, we tested it on the VidSet4 dataset [55], a commonly used dataset for testing VSR models, which contains 4 scenarios.

We compared our proposed VSRCoDiGAN model with the current state-of-the-art video super-resolution models for VidSet4 test dataset. More specifically, we compared it with VSRResFeat-GAN [48] which uses a similar adversarial training approach as ours but with a single discriminator, that is, without explicitly using spatial information into account. Our second model used for comparison is the spatially adaptive GAN (SA-GAN) for the VSR problem [47], which incorporates spatial information into pixel and feature losses and is trained using the traditional single discriminator GAN framework.

Table 2.1: PSNR and SSIM comparison with state-of-the-art VSR models on the VidSet4 dataset for scale factor 3.

	VSRResFeatGAN	SA-GAN	VSRCoDiGAN
	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
calendar	23.40/0.8033	23.59/**0.8130**	**23.63**/0.8086
city	27.23/0.7832	27.48/**0.7925**	**27.57**/0.7869
foliage	25.29/0.7544	25.74/0.7754	**26.37/0.7974**
walk	30.20/0.9182	30.40/0.9213	**30.64/0.9230**
Average	26.53/0.8148	26.80/0.8256	**27.06/0.8289**

Table 2.1 reveals that both VSRCoDiGAN and SA-GAN outperform the state-of-the-art VS-RResFeatGAN model in all scenarios (calendar, city, foliage, walk). This suggests that including spatial information into training helps to improve the quality of generated frames. Furthermore, we observe that our VSRCoDiGAN model performs better than SA-GAN in most cases, which indicates that it is beneficial to formulate the training with collaborative discriminators to explictly

incorporate high frequency spatial information in the learning process. Also, as table 2.1 indicates, our method improves the visual quality of the generated frames for all scenarios. A qualitative comparison is shown in Fig. 2.3. Considering the zoomed in regions in the frames (numbers and letters in the first column and the star in the second column), we can clearly observe that our VSRCoDiGAN model generates more accurately super-resolved edges (closer to those in ground truth frames) with fewer artifacts and less noise compared to VSRResFeatGAN and SA-GAN. We conclude from these quantitative and qualitative results that including a spatially adaptive discriminator to incorporate spatial information into the GAN training has a significant positive impact on the quality of the resulting frames.

2.5 Conclusion

In this chapter, we have shown that our VSRCoDiGAN model, which incorporates a novel spatially adaptive collaborative discriminator to explicitly integrate high-frequency spatial information into training. The model exhibits significantly fewer artifacts and more accurate edges. We established the zero-sum game property of our model and provided justification for its selective discrimination approach.

CHAPTER 3
ATMOSPHERIC TURBULENCE CORRECTION VIA VARIATIONAL DEEP DIFFUSION

From this chapter, we begin to delve into the atmospheric turbulence correction task. Atmospheric turbulence is a common phenomenon in our daily experiences. For instance, when driving on a hot day, distant landscapes may appear blurred and distorted. It results primarily from turbulent airflow caused by the interaction of hot and cold air masses. When light traverses these regions, it becomes distorted. Subsequently captured by detectors, such as human eyes or cameras, resulting images exhibit blurring and distortion. Such distortions adversely impact subsequent computer vision tasks, including human or object detection and recognition. Our objective is to mitigate atmospheric turbulence effects by producing cleaner images or videos, thereby improving the performance of the following vision tasks. (Please note this chapter restates text and figures from our previously published work [56]).

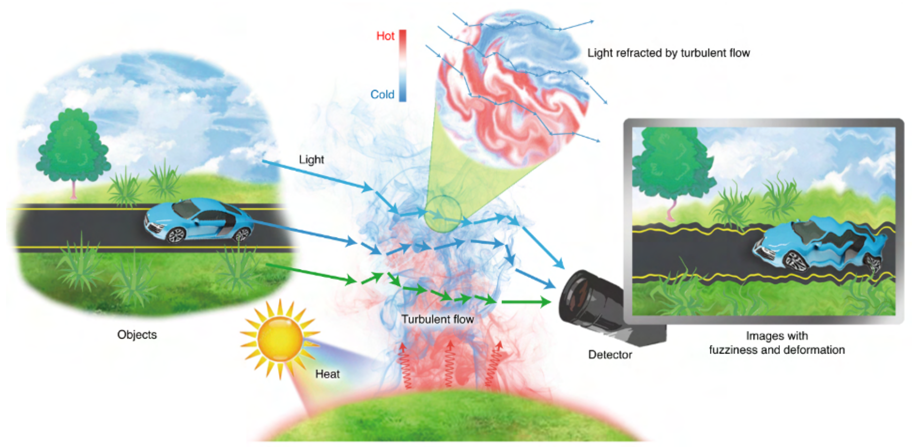


Figure 3.1: Atmospheric turbulence. Adapted from the internet.

3.1 Introduction

Atmospheric turbulence, a common phenomenon in daily life, iprimarily arises from the uneven heating of the Earth's surface. This phenomenon can lead to significant blur and perceptual degradation. Consequently, it can significantly affect the performance of subsequent downstream vision tasks, such as detection, recognition, and so on. Unlike other imaging inverse problems, atmospheric turbulence degradation contains a mixture of geometrical distortion, spatially variant blur, and noise, which makes AT more challenging to mitigate. Earlier works in AT correction mainly focus on optics and lucky imaging algorithms [57]. In recent years, with the development of deep-learning (DL) algorithms for solving various inverse problems [58], some works have proposed DL-based AT removal methods [59], [60]. The availability of a fast AT simulation algorithm that preserves essential turbulence statistics, as presented in [61], has made large-scale data-driven DL training for AT correction feasible [59]. In this chapter, we also adopt the simulation method in [61] to construct our training and testing datasets.

Recently, deep diffusion models have emerged for image generation [62]. As a likelihood-based algorithm, this algorithm exhibits greater stability during training compared to generative adversarial networks (GAN) and does not suffer from mode collapse. Diffusion models have proven successful in diverse vision tasks, including image synthesis [63] and super-resolution [63], [64]. A recent study applied diffusion models for restoring faces affected by atmospheric turbulence [65]. The authors employed a pre-trained diffusion model from the super-resolution task and adapted it for face atmospheric turbulence (AT) correction. Nevertheless, no prior research has explored the use of diffusion models in generic scene atmospheric turbulence (AT) correction. In this chapter, we propose employing a diffusion model to eliminate atmospheric turbulence in generic scenes, yielding visually superior results. Moreover, in contrast to prior methods [64], [65] that only rely on the input low-quality image for guiding diffusion models, we leverage a variational inference image restoration framework [66] to learn latent features relevant to task-specific prior information from both the input and the degradation process. Subsequently, we incorporate this acquired knowledge as a conditioning factor into the diffusion models, enabling them to adapt their behavior based on both the degraded input image and the task-specific prior information, thereby enhancing their performance.

3.2 Proposed Method

In this section, we present the proposed AT-VarDiff model. We begin by explaining the conditional denoising diffusion process of our model in Section 3.2.1. Then, we introduce the variational framework used to obtain the condition encoding the task-specific information in Section 3.2.2. Finally, in Section 3.2.3, we illustrate the inference during testing.

3.2.1 Conditional Diffusion Model

Our training dataset contains N image pairs $\{y_i, x_i\}_{i=1}^{N}$, where y_i represents the AT degraded image and x_i the corresponding ground-truth image. As shown in Figure 3.1, our model aims at learning the data distribution $p(x|y, c)$ by a stochastic iterative refinement process, which maps the input degraded image y and the learned latent prior information c to the ground-truth image x. The forward/diffusion process (from right to left) gradually adds Gaussian noise, denoted by $q(x_t|x_{t-1})$. Our goal is to reverse the diffusion process (from left to right) by gradually recovering the image from the input Gaussian noise with conditions, which corresponds to learning the reverse process of a fixed Markov Chain of length T conditioned on y and c. More specifically, starting from a pure Gaussian noise image $x_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, the model learns the conditional transition distribution $p_\theta(x_{t-1}|x_t, y, c)$ and iteratively denoises the image for T steps, generating the target image x_0 in the end, such that $x_0 \sim p(x|y, c)$.

The overall training framework of the AT-VarDiff model is shown in Figure 3.3. Following the model design in denoising diffusion probabilistic model (DDPM) [62], the architecture of our conditional diffusion module is a U-Net [67] based on a wide ResNet [68], denoted as ϵ_θ. Training is performed by optimizing the usual variational bound on the negative log-likelihood, and the corresponding objective function can be simplified to [62], [63]:

$$\mathcal{L}_{diff} = \mathbb{E}_{x,y,c,\varepsilon,t}[||\epsilon - \epsilon_\theta(x_t, t, y, c)||_2^2], \tag{3.1}$$

with t is uniformly sampled from $\{1, ..., T\}$. According to Equation 1, ϵ_θ takes as input the noisy target image x_t, time step t, the AT degraded image y, and the learned latent prior information c to provide an estimate of the noise ϵ. The details of obtaining c are discussed in the following Section.

3.2.2 Variational Inference Framework

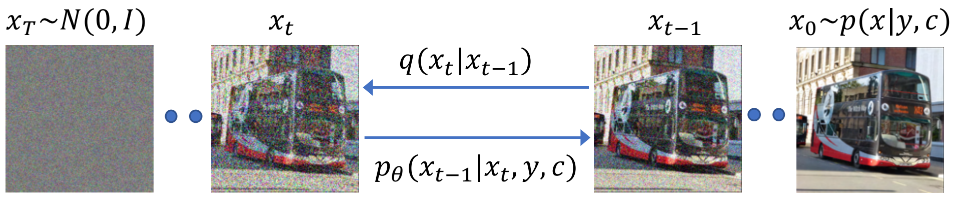


Figure 3.2: Conditional denoising diffusion process.

The current conditional diffusion models used for solving image restoration tasks like super-resolution [63], [64], and face AT correction [65] only use the input degraded image as the condition. No work in the literature has employed any other task-specific prior information or domain-knowledge to further enhance the conditioning progress. According to [66], providing additional

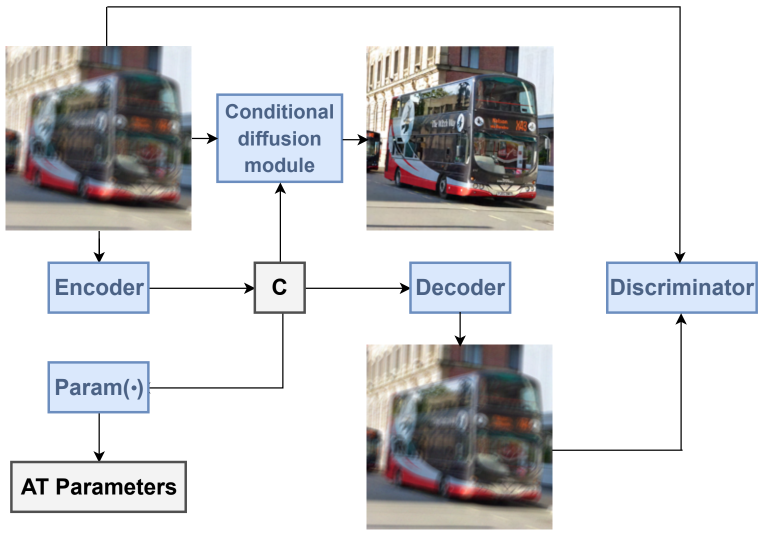


Figure 3.3: Training framework of AT-VarDiff model.

information can be interpreted as dividing a complex distribution into simpler sub-distributions that will eventually make network training easier and the results more accurate, since the number of possible solutions would be reduced. In this chapter, we propose to use a variational inference framework to extract the latent task-specific prior information from the input and the degradation process and use the extracted feature as an additional condition to guide the diffusion model.

As shown in Figure 3.3, we refer to a variational autoencoder (VAE) based framework [66] to learn the latent feature c from the input degraded image y and the AT degradation parameters. To achieve this goal, the objective we use here contains three parts: the VAE loss, the adversarial loss, and the AT degradation parameters' loss.

The VAE loss contains the fidelity term and the reconstruction term, that is,

$$\mathcal{L}_{vae} = D_{KL}(q_{e_\psi}(c|y)||p(c)) + ||y - \hat{y}||_2^2. \tag{3.2}$$

The first term is the fidelity term; it measures the fidelity of c extracted from the encoder e_ψ, whose

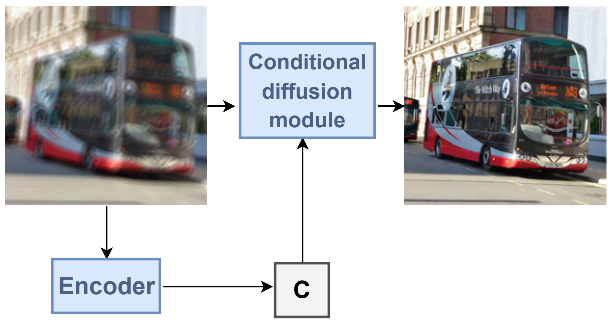


Figure 3.4: Testing framework of AT-VarDiff model.

input is the degraded image y. It is represented as the KL divergence of the approximate posterior $q_{e_\psi}(c|y)$ from the prior $p(c)$. We select the prior $p(c)$ as a standard Gaussian distribution. The second term is the reconstruction term, and we adopt the pixel-wise mean squared error (MSE) distance between the input degraded image y and the output $\hat{y}$ of the decoder d_φ. In addition, we utilize a GAN [69] to better learn the input degraded image distribution, in which an additional discriminator is jointly trained to discriminate the generated $\hat{y}$ and the true degraded image y. Therefore, we also include an adversarial loss: $\mathcal{L}_{adv} = -log(D(\hat{y}))$, and the corresponding loss for the discriminator D is: $\mathcal{L}_{disc} = -log(D(y)) - log(1 - D(\hat{y}))$.

Finally, we would like the latent feature c to contain knowledge from the AT degradation process. Therefore, we add a degradation loss defined as:

$$\mathcal{L}_{degrad} = ||\phi_{at} - \hat{\phi_{at}}||_2^2, \tag{3.3}$$

where ϕ_{at} represents the ground-truth AT degradation parameters from the pre-trained AT simulator [61]. $\hat{\phi_{at}} = Param(c)$ represents the estimated AT degradation parameters, and is the output of a small network (parameter estimation module) $Param(\cdot)$ taking c as input, as shown in Figure 3.3.

Therefore, the final objective used for training our AT-VarDiff model is defined as

$$\mathcal{L} = \mathcal{L}_{diff} + \lambda_1 \mathcal{L}_{vae} + \lambda_2 \mathcal{L}_{adv} + \lambda_3 \mathcal{L}_{degrad}, \tag{3.4}$$

where $\lambda_1, \lambda_2, \lambda_3$ are hyper-parameters.

3.2.3 Inference

The testing framework of our model is shown in Figure 3.4. During testing, we followed the DDPM's denoising sampling procedure (Algorithm 2 in [62]) conditioned on the input degraded image y and the learned task-specific latent feature c to generate the output restored image. During both training and testing, we perform the conditioning via concatenation, and we set $T = 1000$ for all the experiments.

3.3 Experiments

Table 3.1: LPIPS, FID & NIQE metrics comparison on simple-DDPM, AT-VarDiff, and AT-DDPM [65].

	AT-DDPM [65]	**Simple-DDPM**	**AT-VarDiff (Ours)**
LPIPS $\downarrow$	0.2150	0.1923	**0.1094**
FID $\downarrow$	80.05	60.87	**32.69**
NIQE $\downarrow$	10.15	6.65	**6.46**

We use the AT simulator in [61] to simulate the effect of AT on the REDS dataset [70], and the hyper-parameter of the simulator (D/r_0) is chosen randomly in the range [0.5, 2.0]. Our synthetic training dataset has one million AT degraded and clean image pairs. We use another 2500 synthetic AT degraded images as the testing dataset.

For our encoder module, we use 5 2D-convolution (2D-conv) layers with ReLU activation and one down-sampling layer after the first conv layer. For the decoder module, we use 5 2D-conv layers with ReLU activation and one up-sampling layer after the first conv layer. For our parameter estimation module, we simply use 2 2D-conv layers with LeakyReLU activation. The discriminator is formed by 11 2D-conv layers with LeakyReLU activation and spectral normalization [71].

During training, we augment the training data by random cropping (160×160), random vertical and horizontal flips, and random transposing. We train our model for 200 epochs with 1500 iterations per epoch, and we set the batch size to 16. We use the Adam optimizer [35] with a weight decay of 0, and we set the initial learning rate to $1e-4$ and gradually reduced it to $5e-6$ during training utilizing the cosine annealing schedule [72]. The hyper-parameters λ_1, λ_2, and λ_3 used in our final training objective (Equation 3.4) are set to 0.1, 0.1, and 0.5, respectively. All the experiments are performed on a single NVIDIA Quadro RTX 8000 GPU.

3.4 Results

To evaluate our model, we first the Fréchet Inception Distance (FID) [73] and the Learned Perceptual Image Patch Similarity (LPIPS) [36] metrics, which are measures of similarity between two sets of images. We also use the Naturalness Image Quality Evaluator (NIQE score) [74] to evaluate how likely the generated image is to be a naturally occurring one. These three metrics are shown to correlate well with the human judgment of visual quality. Lower metric values indicate higher image quality. In Table 3.1, we show the quantitative results of our proposed AT-VarDiff model and compare it to using a pure conditional DDPM-based diffusion model like the approach used in [65], i.e., this simple-DDPM model is only built with the conditional diffusion module and is only conditioned on the input degraded image y. We can see that our AT-VarDiff model improves on all metrics, demonstrating the effectiveness of our proposed variational conditional diffusion

Figure 3.5: Visual comparisons of AT-DDPM [65], simple-DDPM, and AT-VarDiff.

framework. We also compare with the pre-trained AT-DDPM model from [65] in the table. Our proposed approach consistently exhibits much better visual clarity, far fewer artifacts, and higher quality across the testing dataset. A visual example can be seen in Figure 3.5.

3.5 Conclusion

In this chapter, we propose the variational deep diffusion model AT-VarDiff to restore images degraded by atmospheric turbulence. Our approach utilizes the diffusion process to mitigate atmospheric turbulence (AT) in generic scenes. Additionally, we employ a variational inference framework to extract latent task-specific prior information from both the input and the AT degradation. Furthermore, we incorporate the extracted features as an additional condition to guide the diffusion model's behavior. We demonstrate that our proposed method yields superior results and outstanding visual quality, outperforming the current state-of-art.

CHAPTER 4
REAL-WORLD ATMOSPHERIC TURBULENCE CORRECTION VIA DOMAIN ADAPTATION

Up to this point, our AT correction model has been trained using synthetic data, and there's always a performance drop when applying the model to another dataset, particularly when confronted with real-world AT data. Additionally, in real-world scenarios, there is no ground-truth clean data available for AT-degraded videos. Consequently, to mitigate the performance degradation of our model on real data and to maximize the utilization of labeled synthetic datasets, we employ a domain adaptation method and propose a real-world AT mitigation model. (Please note this chapter r

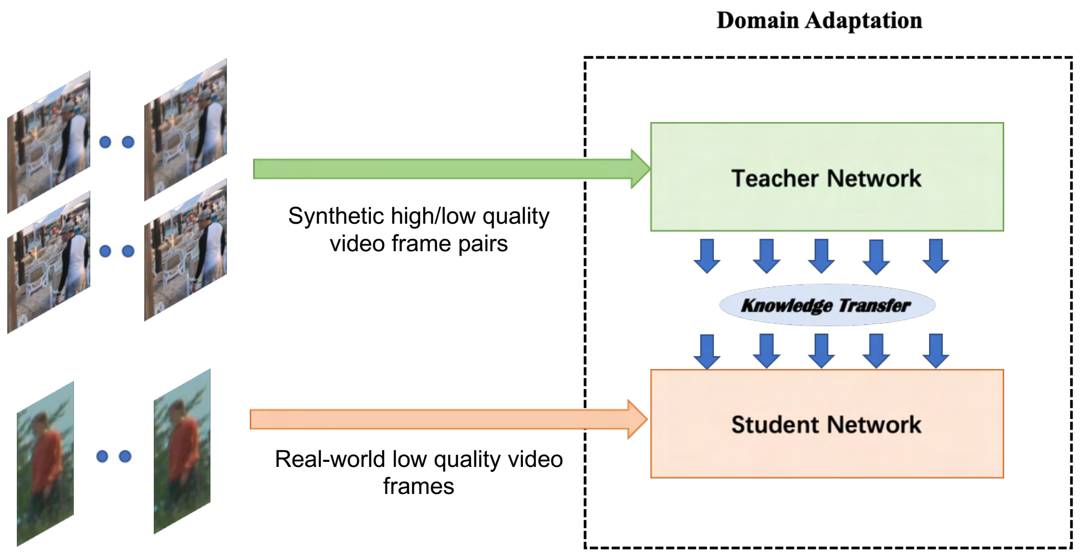

Domain Adaptation
Teacher Network
Synthetic high/low quality video frame pairs
Knowledge Transfer
Student Network
Real-world low quality video frames

Figure 4.1: Domain adaptation framework.

4.1 Introduction

Atmospheric turbulence (AT) introduces numerous challenges such as image/video distortion, blurring, and diminished accuracy in subsequent vision tasks like object detection, recognition, and tracking. For instance, the task of identifying a person appearing in an image or video is important in many surveillance applications, and such surveillance tasks usually involve imaging over long distances, which are more susceptible to the effects of atmospheric turbulence [76], [77]. Therefore, multiple methods have been studied in the literature for AT mitigation. Early attempts at AT correction centered on adaptive optics-based methods [78]–[80], which typically demand costly and intricate hardware setups. Therefore, numerous image-processing-based methods were also developed [57], [81]–[83]. These methods often involve fusing complementary clear regions across frames using the lucky fusion process, followed by deconvolution to restore the output [76].

Recently, with the development of deep-learning (DL) algorithms for solving various inverse problems [31], an increasing number of studies have proposed DL-based AT removal methods. A deep learning-based approach is introduced in [84] employing an effective nearest neighbors-based method for registration and an uncertainty-based network for restoration. A physics-inspired transformer model for imaging through atmospheric turbulence is introduced in [59]. The authors in [85] presented a multi-frame image restoration transformer tailored for addressing atmospheric turbulence. The authors in [60] employ a two-stage deep adversarial network aimed at minimizing atmospheric turbulence, where the first stage focuses on reducing geometrical distortion, while the second stage targets minimizing image blur. A method leveraging well-trained Generative Adversarial Networks (GANs) as effective priors, learning to restore images within the semantic space context is proposed in [86]. Reference [87] proposes a variational deep-learning approach for video atmospheric turbulence mitigation. Finally, references [56], [65], [88] propose diffusion-based models for atmospheric turbulence restoration and generate high-quality images. All these current models are trained using synthetically generated data and tested on synthetic or real-world data. The performance of the existing models always decreases when applied to real-world images or videos. Therefore, motivated by the success of domain adaptation techniques in real-world image super-resolution and blurring tasks [89]–[91], we propose in this chapter to employ domain adaptation to train the model using a combination of synthetic and real-world data. As shown in Figure 4.1, we use a teacher-student framework and perform knowledge transfer from supervised learning to unsupervised learning. The teacher network is trained with a synthetic dataset containing simulated AT degraded frames paired with their corresponding ground-truth clean frames, enabling supervised learning. The student network is only trained with the AT degraded dataset from the real world, and no ground-truth clean frames are used, therefore the learning is unsupervised.

4.2 Proposed Method

In Section 4.2.1, we will first describe the design of the domain adaptation framework for our real-world AT mitigation model along with the associated objective functions. In Section 4.2.2, we will delve into the design of our generators and their associated objective functions. Lastly, the summarized objective function during training, the training and testing procedures are outlined in Section 4.2.3.

4.2.1 Domain Adaptation Framework

The framework of our real-world atmospheric turbulence mitigation (Real-ATM) model is depicted in Figure 4.2. We employ a teacher-student (T-S) framework and facilitate knowledge transfer from supervised to unsupervised learning. Specifically, the teacher network is trained using a synthetic dataset comprising simulated AT-degraded frames and their corresponding ground-truth clean frames, thus learning in a supervised manner. The student network is trained solely on the AT-degraded dataset from the real world, and there are no ground-truth clean frames for them.

We adopt the GAN-based [10] training to generate frames of high perceptual quality. First, for training the teacher component, we utilize pairs of synthetic AT-degraded and clean frames. The degraded frame y_T is fed into the teacher generator and outputs the restored one $\hat{x}_T$. The teacher discriminator is trained to differentiate between its input being the restored frame $\hat{x}_T$ generated by the teacher generator and the real frame x_T from the corresponding ground-truth clean frame. The Reproduce Net (R-Net) is trained to regenerate the degraded frame $\hat{y}_T$, taking the restored frame $\hat{x}_T$ as input.

The objective function for training the teacher part first contains a pixel distance loss and a perceptual distance loss, measuring the distance between the restored frame and its corresponding ground-truth clean frame:

$$L_{dist}^{T} = \|x_T - G_\theta(y_T)\|_2^2 + \|\psi(x) - \psi(G_\theta(y_T))\|_2^2, \tag{4.1}$$

where x_T denotes the ground-truth clean frame, y_T represents the corresponding synthetic degraded frame, and G_θ denotes the teacher generator, so $G_\theta(y_T)$ represents the output of the generator, which is the restored frame $\hat{x}_T$. The feature space denoted as $\psi(\cdot)$ is computed from the activations provided by the 3^{rd} and 4^{th} convolution layers of the VGG network [30].

Following [10], the adversarial losses for our AT mitigation task are shown as follows: The teacher generator network minimizes the following loss with respect to θ

$$L_{gen}^{T} = \mathbb{E}_{y_T}\left[-\log D_{\phi^T}\left(G_\theta\left(y_T\right)\right)\right], \tag{4.2}$$

and the teacher discriminator network D_{ϕ^T} minimizes the following loss with respect to ϕ^T

$$\begin{aligned} L_{dis}^T = & \mathbb{E}_{x_T}\left[-\log D_{\phi^T}\left(x_T\right)\right] + \\ & \mathbb{E}_{y_T}\left[-\log\left(1 - D_{\phi^T}\left(G_\theta\left(y_T\right)\right)\right)\right]. \end{aligned} \tag{4.3}$$

A reproduce match loss is used to quantify the distance between the degraded frame reproduced by the R-Net and the true degraded frame:

$$L_{rm}^T = \|y_T - \hat{y}_T\|_2^2. \tag{4.4}$$

During the training of the student component, we utilize real-world AT data, which consists solely of degraded frames. During training, the student generator takes the real-world AT degraded frame y_S as input and outputs the restored frame $\hat{x}_S$. We have the student generator share weights with the teacher generator, constituting one of the knowledge transfer steps between the teacher and the student.

The student discriminator D_{ϕ^S} is trained to distinguish between the real clean frame x_S and the fake restored frame $\hat{x}_S$. Note that since we do not have the real clean frames, we utilize the clean frame x_T from the dataset used for training the teacher part. This is feasible because the goal of the discriminator is to distinguish between data from two distributions: samples from the clean data distribution and those from the fake data distribution. Furthermore, the student discriminator's weights are partially shared with the teacher discriminator's, representing the second stage of knowledge transfer. Finally, the restored frame $\hat{x}_S$ is also fed into the R-Net, which regenerates the degraded frame $\hat{y}_S$. The R-Net here serves as the third knowledge transfer stage.

The losses associated with the student part include the adversarial losses for the student gener-

Figure 4.2: Real-world atmospheric turbulence mitigation model (Real-ATM) framework.

ator and discriminator:

$$L_{gen}^{S} = \mathbb{E}_{y_S}\left[-\log D_{\phi^S}\left(G_\theta\left(y_S\right)\right)\right], \tag{4.5}$$

$$\begin{aligned} L_{dis}^{S} = &\mathbb{E}_{x_T}\left[-\log D_{\phi^S}\left(x_T\right)\right] + \\ &\mathbb{E}_{y_S}\left[-\log\left(1 - D_{\phi^S}\left(G_\theta\left(y_S\right)\right)\right)\right], \end{aligned} \tag{4.6}$$

and the reproduce match loss between the regenerated degraded frame and the real degraded frame:

$$L_{rm}^{S} = \|y_S - \hat{y}_S\|_2^2. \tag{4.7}$$

Up to this point, we have gained a better understanding of how our model works via the knowledge transfer from supervised learning to unsupervised learning. Specifically, the T-S knowledge transfer relies on a shared generator network, partial weight sharing of dual discriminators, and a reproduce network.

4.2.2 Generator Framework

Drawing inspiration from the variational inference framework employed in restoration models [56], [66], [87], the detailed framework of the teacher and student generators is shown in Figure 4.3. During training, the Decoupled Dynamic Filter (DDF)-based [92] network aims at generating the restored frame from the degraded input frame conditioned on the latent feature c. This c is learned via a variational autoencoder (VAE), which aims at learning to reconstruct the degraded frames, thereby imbuing c with image-prior information acquired from the degraded data.

Additionally, we learn from the literature that incorporating the information from the degradation process during training could further improve the model's performance. Therefore, we employ another CNN-based network to predict the atmospheric turbulence degradation parameters. Consequently, c also encompasses prior information learned from the AT degradation procedure.

The losses associated with this VAE-based design contain a fidelity term and a reconstruct term. The fidelity term is the KL divergence of the learned posterior and the prior, which is a standard normal distribution. The reconstruction term measures the distance between the reconstructed degraded frame and the real degraded frame.

Thus, for the teacher part, the VAE loss is defined as:

$$L_{vae}^{T} = D_{KL}(q_{e_{\psi}}(c_T|y_T)||p(c_T)) + ||y_T - \hat{y}_T^G||_2^2, \tag{4.8}$$

where the first term is the fidelity term; it measures the fidelity of c_T extracted from the encoder e_{ψ}, whose input is the degraded image y_T. It is represented as the KL divergence of the approximate posterior $q_{e_{\psi}}(c_T|y_T)$ from the prior $p(c_T)$. We select the prior $p(c_T)$ as a standard Gaussian distribution. The second term is the reconstruction term, and we adopt the pixel-wise mean squared error (MSE) distance between the input degraded image y_T and the output $\hat{y}_T^G$ of the decoder d_{φ}.

Similarly, the VAE loss for the student is:

$$L_{vae}^{S} = D_{KL}(q_{e_\psi}(c_S|y_S)||p(c_S)) + ||y_S - \hat{y}_S^G||_2^2. \tag{4.9}$$

Finally, we would like the latent feature c to encompass knowledge about the AT degradation process. Remember that only the teacher knows the synthetic AT degradation process. Therefore, we add a degradation loss for the teacher part, defined as:

$$L_{degrad}^{T} = ||\phi_{at} - \hat{\phi_{at}}||_2^2, \tag{4.10}$$

where ϕ_{at} represents the ground-truth AT degradation parameters from the pre-trained AT simulator [61]. $\hat{\phi_{at}} = Param(c_T)$ represents the estimated AT degradation parameters and is the output of a small network (parameter estimation module) $Param(\cdot)$ taking c_T as input, as shown in Figure 4.3.

4.2.3 Training & Testing

Since we utilize a GAN-based training scheme, there is a generation procedure and a discrimination procedure, and they are trained alternatively [10]. To summarize, the objective function during training contains both the teacher part and the student part. Specifically, during the generation procedure, the objective function is formulated as follows:

$$\mathcal{L}_g = L^T + L^S, \tag{4.11}$$

where

$$L^T = L_{dist}^T + \lambda_1 L_{gen}^T + \lambda_2 L_{rm}^T + \lambda_3 L_{vae}^T + \lambda_4 L_{degrad}^T, \tag{4.12}$$

and

$$L^S = \lambda_1 L^S_{gen} + \lambda_5 L^S_{rm} + \lambda_3 L^S_{vae}, \tag{4.13}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ and λ_5 are hyper-parameters. During the discrimination procedure, the objective function is formulated as:

$$\mathcal{L}_d = L^T_{dis} + L^S_{dis}. \tag{4.14}$$

During testing, only the generator network is needed. Illustrated in Figure 4.4, the degraded frame is fed into the encoder module to obtain the latent feature c. Then, c and the degraded frame are input into the DDF module to produce the restored output.

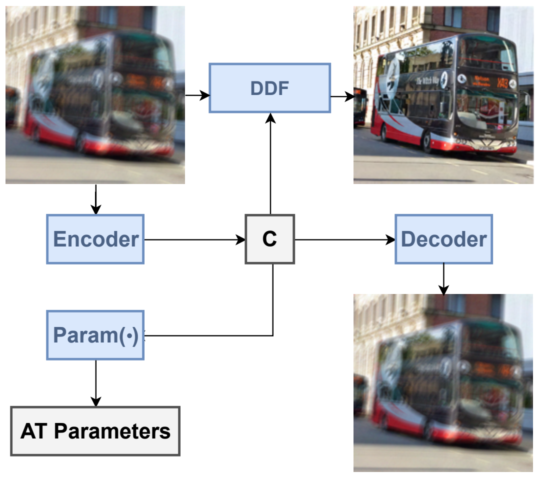


Figure 4.3: Generator training framework.

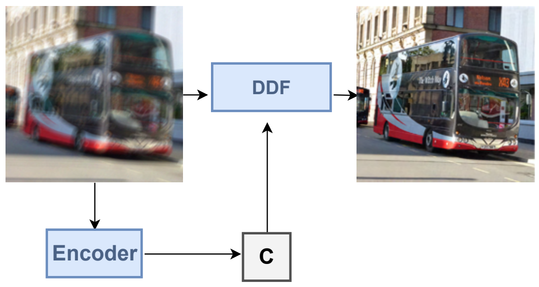


Figure 4.4: Testing framework.

4.3 Experiments

To get the synthetic training dataset for training the teacher part, we use the AT simulator in [61] to simulate the effect of AT on the REDS dataset [70], and the hyper-parameter of the simulator (D/r_0) is chosen randomly in the range [0.5, 2.0]. Our synthetic training dataset comprises 10^6 AT-degraded and clean image pairs. We use another 2500 synthetic AT-degraded images as the synthetic testing dataset. For the student part training, we use 18 AT-degraded videos from the training set of the real-world dataset BRIAR [93]. We use another 54 AT-degraded videos from the testing set of BRIAR as the real-world testing dataset.

To construct our teacher and student generators, as shown in Figure 3, for the encoder module, we use 5 2D-convolution (2D-conv) layers with ReLU activation and one down-sampling layer after the first conv layer. For the decoder module, we use 5 2D-conv layers with ReLU activation and one up-sampling layer after the first conv layer. For the DDF-based module, we use 1 2D-conv layer followed by 8 DDF bottleneck blocks [92] and 2 2D-conv layers. For our parameter estimation module, we simply use 2 2D-conv layers with LeakyReLU activation.

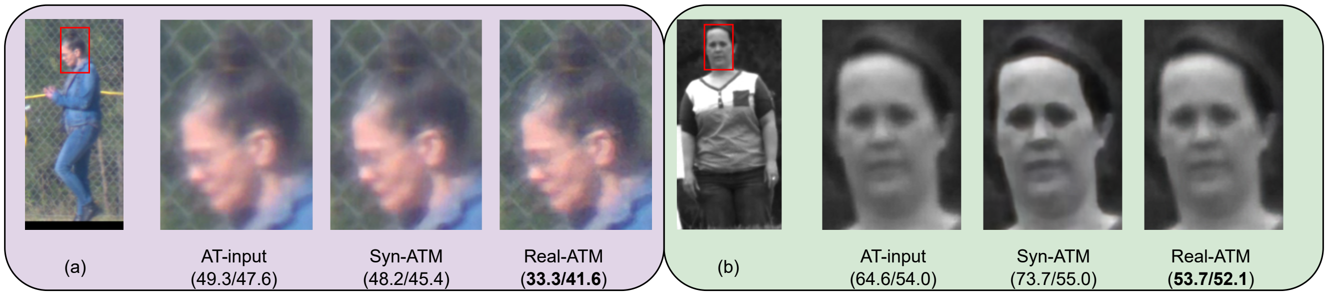


Figure 4.5: Visual samples of real-world AT-input, and the restored results from Syn-ATM and Real-ATM. (PIQE/BRISQUE) is shown in the parenthesis, the best score is marked in bold.

To construct the teacher and student discriminators, we use 11 2D-conv layers with LeakyReLU activation and spectral normalization [71]. The partial weight sharing of the teacher and student discriminators occurs from the 6th to the 9th convolutional layers.

During training, we augment the training data by random cropping (160×160), random vertical and horizontal flips, and random transposing. We train our model for 200 epochs with 1500 iterations per epoch, and we set the batch size to 16. We use the Adam optimizer [35] with no weight decay, and we set the initial learning rate to $1e-4$ and gradually reduced it to $5e-6$ during training utilizing the cosine annealing schedule [72]. The hyper-parameters λ_1, λ_2, λ_3, λ_4, and λ_5 used in Equation 4.12 and Equation 4.13 are set to 1e-3, 5e-1, 1e-1, 2e-1, and 2.5e-1, respectively. All the experiments are performed on two NVIDIA Quadro RTX 8000 GPUs.

4.4 Results

In this section, we will show that our proposed method enhances performance in real-world atmospheric turbulence (AT) scenarios, benefiting both image quality and the downstream person identification task.

4.4.1 Image Quality

In Table 4.1, we show the Peak Signal to Noise Ratio (PSNR) and the Structural Similarity Index Measure (SSIM) results of the model trained only with the AT synthetic dataset (Syn-ATM), i.e., only the teacher part is trained. We can see that the current model has achieved noticeable improvement on the synthetic testing set. Due to the absence of ground-truth counterparts in the real-world testing set, we adopt the no-reference image quality metrics: Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) and Perception-based Image Quality Evaluator (PIQE) to evaluate the image quality (lower metric values indicate higher image quality). As shown in Table 4.2, for the Syn-ATM model, the image quality does not improve much when we test it on the real-world AT testing set. This observation aligns with our explanation that synthesized turbulent images are insufficient in capturing all real-world turbulence characteristics. Despite achieving remark-

	PSNR↑	**SSIM↑**
AT-input	24.19	0.7263
Syn-ATM	**30.61**	**0.8168**

Table 4.1: Image quality results of the Syn-ATM model on the synthetic AT testing set.

	BRISQUE↓	**PIQE↓**
AT-input	47.67	63.39
Syn-ATM	46.99	68.89
Real-ATM	**44.90**	**49.95**

Table 4.2: Image quality results of Syn-ATM and Real-ATM models on the real-world AT testing set. The best result is marked in bold.

able performance in simulated scenarios, a performance drop occurs when applying the models to real-world cases. Conversely, our proposed real-ATM model obtains improved results on both metrics, demonstrating the effectiveness of our proposed method in filling up the domain gap between synthetic and real-world space. Some real-world visual examples are shown in Figure 4.5. We could see that the real-ATM model provides better visual quality. Besides, we observed that the synthetic-ATM model can sometimes generate results with artificial characteristics and does not look like a naturally occurring image, as shown in Figure 4.5 (b). This discrepancy arises because the model is solely trained on the synthetic AT dataset, failing to capture the full complexity of real-world AT distributions.

4.4.2 Downstream Person Identification Task

According to previous works from the literature [94]–[96], image and video restoration as a fundamental low-level vision task can significantly improve the visual quality and benefit many downstream computer vision tasks. However, there is a complexity in real-world applications that might not be solved by quality alone. Sometimes the enhancement of the perceptual quality of images

	Top-20↑	**Top-15↑**
AT-input	18.5%	14.8%
Syn-ATM	14.8%	7.4%
Real-ATM	**25.9%**	**16.7%**

Table 4.3: Human recognition results on real-world AT videos and restored videos by Syn-ATM and Real-ATM. The best result is marked in bold.

does not necessarily improve the performance on vision tasks or the enhancement strategies need to be tailored to specific applications rather than just improving image quality alone. Therefore, in this section, we investigate the potential benefits of our proposed restoration network for downstream tasks. Specifically, we evaluate our model on a human recognition task to validate the necessity of restoration. For evaluation, we use the real-world person identification model, ShARc [97], applied to our real-world testing set comprising 54 videos from the BRIAR testing set. As shown in Table 4.3, top-M represents whether the person could be correctly identified within the top M predictions made by the identification model. When applying the identification model to the restored videos generated from the Syn-ATM, the identification results do not improve. When applying the identification model to the videos restored from the real-ATM, the identification results significantly improve, demonstrating the effectiveness of our proposed method in benefiting the downstream vision task, in addition to improving the image quality alone.

4.5 Conclusion

In this chapter, we introduce a real-world atmospheric turbulence mitigation method. Our model utilizes a domain adaptation teacher-student framework, bridging supervised synthetic AT mitigation with unsupervised real-world AT mitigation. This strategy improves our capacity to utilize information from labeled synthetic datasets while minimizing the domain gap between synthetic and real-world data. Our ultimate model, real-ATM, not only enhances image quality in restoring real-world atmospheric turbulence data but also aids downstream vision tasks.

CHAPTER 5
MOTION BLUR REDUCTION IN ABDOMINAL MRIS USING GANS

In this chapter, we look into the motion blur reduction task. Similar to the approach used for super-resolution, we employ a GAN-based method to eliminate motion blur in abdominal MRIs.(Please note this chapter restates text and figures from our previously published work [98])

5.1 Introduction

Magnetic resonance imaging (MRI) is a noninvasive test used to diagnose medical conditions. It is also an invaluable diagnostic tool in early diagnosis and evaluation of liver focal lesions and tumors [99]. However, involuntary body movements and respiration during imaging sessions often lead to motion artifacts, especially in individuals unable to hold their breath adequately, resulting in significant image degradation [100]. Given the crucial role of abdominal anatomy in accurate diagnosis, the need for repeat scans due to such artifacts burdens hospitals with additional time and costs [101]. Therefore, a more efficient method needs to be developed to reduce abdominal motion and artifacts automatically in real-time.

Deep learning methods, particularly Convolutional Neural Networks (CNNs), have demonstrated significant success in addressing various inverse problems in image processing [31], including denoising [102], [103], super-resolution [6], [104], [105], and motion artifact reduction [106], [107]. However, outcomes from these models often suffer from blurring and over-smoothing due to the filtering effects of CNN kernels. Notably, in the medical domain, where image quality is crucial for accurate diagnosis, there is a growing demand for cleaner, sharper, and more realistic images. Recently, Generative Adversarial Networks (GANs) [10] have emerged as a powerful tool for learning the underlying distribution of training data, leading to more natural-looking images with improved fine-detail reconstruction compared to pure CNN models [31], [108]. Therefore, in this chapter, we propose a GAN-based model to address our motion reduction task.

To enhance the perceptual quality of generated images, we incorporate the perceptual loss [109] into our GAN-based model's training loss. It is a distance-based loss defined in the feature space from a pre-trained discriminative CNN model, so it contributes to optimizing the model in the feature space instead of the pixel space. The perceptual loss has been included as a regularizer to the adversarial loss in a number of the state-of-the-art GAN-based models, which shows promising results with respect to improving the image visual quality in many other image processing tasks, like denoising[110] and super-resolution[111], [112].

GAN-based models are generally challenging to train due to their complexity. Hence, starting with a solid foundation is crucial, particularly for the generator, which faces a more complex task than the discriminator. Rather than training both components from scratch, we initially pre-train our generator using a pixel-wise distance loss (i.e., mean square error). Subsequently, we employ this pre-trained model as the starting point for the generator in subsequent GAN training steps. This chapter shows that this pre-training technique further facilitates the model to have a better motion reduction performance.

This chapter begins by simulating three types of motion artifacts to replicate real abdominal motion. Then, we propose a GAN-based model for reducing motion artifacts. To train this model effectively, we incorporate the perceptual loss into our training loss function. Additionally, we add another pre-training step to enhance the perceptual quality of the results, thereby improving the performance of GAN training.

5.2 Proposed Method

In this section, we first describe a method to simulate motion artifacts via the K-space and generate three different motion types for abdominal MRIs. Then we introduce the whole process of our GAN-based motion reduction pipeline and explain the algorithms and loss functions we use for training the GAN model.

5.2.1 Motion Simulation

Pairs of images (i.e., clean images and images with simulated motion) are needed for training purposes. The motion is simulated by manipulating the clean MRI image in the K-space. K-space represents the 2D Fourier transform of the MRI images, where low spatial frequencies (i.e., close to the center of the K-space) contain contrast and signal to noise ratio information and high spatial frequencies (i.e., close to the peripheral regions of the K-space) contain image resolution information [113]. Therefore, by manipulating the K-space data, we can obtain different types and varying severity of the simulated motion artifacts.

Three types of motion artifacts are simulated following [114] in order to mimic the most realistic body motions, as shown in Fig. 5.1. The clean images I_c are first converted to the K-space, $\mathcal{F}(I_c)$, by taking their Discrete Fourier Transform (DFT). They can be converted back to the image space by taking their inverse DFT (iDFT). Images of the first motion type I_{m_1} simulate the periodic respiratory patterns of motion. They are generated by adding phase shifting errors $\phi(k)$ in a sinusoidal wave, i.e.,

$$I_{m_1} = \mathcal{F}^{-1}[\mathcal{F}(I_c)e^{-j\phi(k)}], \tag{5.1}$$

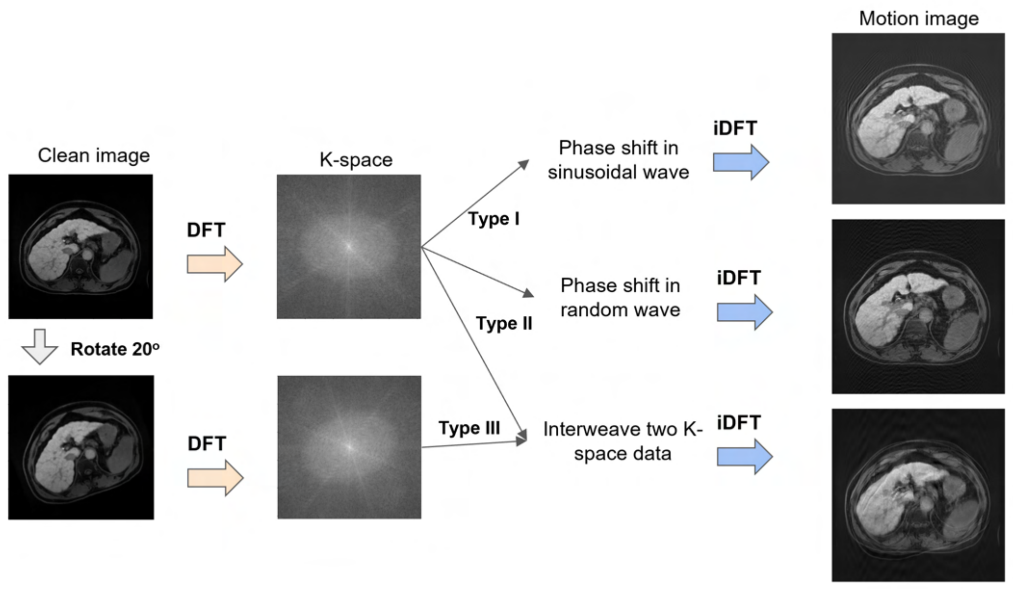

Motion image
Clean image
K-space
DFT
Rotate 20°
DFT
Type I
Type II
Type III
Phase shift in sinusoidal wave
Phase shift in random wave
Interweave two K-space data
iDFT
iDFT
iDFT

Figure 5.1: The motion simulation process. Three types of motion (i.e., sinusoidal motion, random motion and interweaving motion) are simulated by manipulating K-space data to generate new k-space data using DFT and iDFT.

and

$$\phi(k) = \begin{cases} 0 & \text{if } |k| \leq k_m \\ 2\pi k \Delta \sin(\alpha k + \beta) & \text{if } |k| > k_m \end{cases}, \tag{5.2}$$

where k is the K-space coordinate in the phase-encoding direction ($-\pi < k < \pi$), k_m is the number of center k-space lines that are kept unchanged, which is randomly selected in $[\pi/10, \pi/2]$, Δ denotes the number of shift pixels (The larger the Δ, the more severe the simulated motion), and α in $[0.5, 5]$ Hz and β in $[0, \pi/4]$ are the frequency and phase of the sinusoidal wave, respectively.

Similarly, to simulate the non-periodic and irregular respiratory pattern, images of the second motion type I_{m_2} are generated by adding phase shifting errors $\phi(k)$ in a random wave as in Eq.5.1 and keeping the center $\%5 \sim \%10$ of the K-space lines unchanged with no phase encoding, while $\phi(k)$ is randomly selected to be close to the peripheral K-space lines in chance of $\%10 \sim \%50$.

In terms of the severe motion artifacts caused by the body movement, another motion type I_{m_3} is introduced by interweaving K-space data across the clean and the rotated images, where the rotated image I_r is created by rotating a clean image I_c with a random angle in $[-25^o, 25^o]$. Both images are converted to K-space and a new K-space data, K_{new}, can be generated according to:

$$K_{new} = \begin{cases} \mathcal{F}(I_r) & \text{if } k_i \leqslant |k| \leqslant k_{i+j} \\ \mathcal{F}(I_c) & \text{if } otherwise \end{cases}, \tag{5.3}$$

where $j \in [0, 128]$ is the number of K-space lines k_j from I_r and $i \in [0, 384]$ is the value of a K-space location in I_c where we want to interweave with the K-space data from I_r. The total number of K-space lines is 512, which equals to the size of the MRI in the image space. The motion image I_{m_3} can then be converted back from K_{new} by using the iDFT. The selections of i and j simulate different severities of the motion.

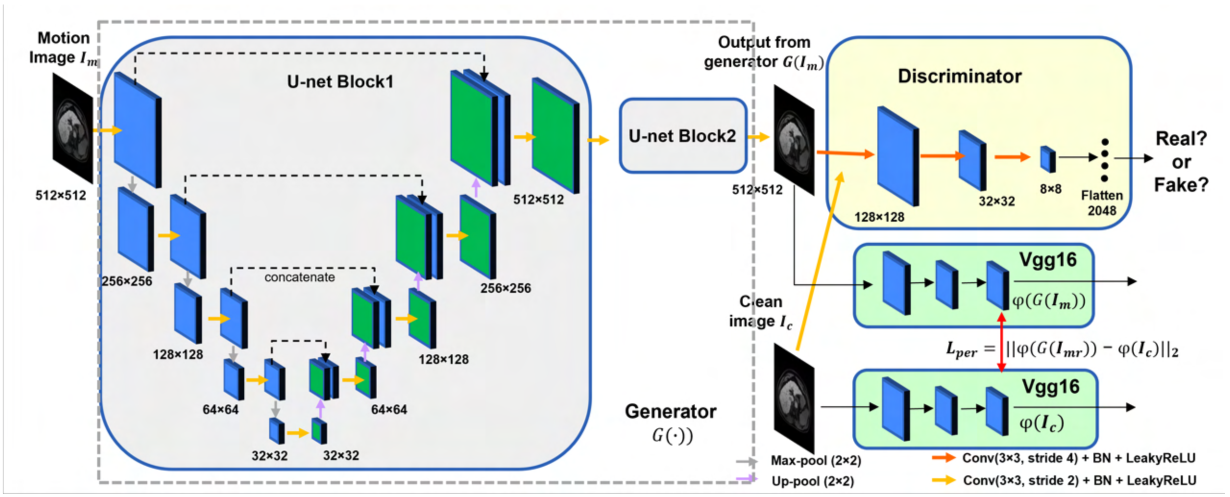


Figure 5.2: Illustration of the completed motion reduction pipeline. It is composed of a generator and a discriminator. The Vgg-16 architecture [115] is used to form the perceptual loss. The CNN U-net blocks are used as the generator in our proposed GAN model. The pairs of the clean image and the image with simulated motion are used as the training input pairs.

5.2.2 The Motion Reduction Pipeline

Because the GAN model[10] favors solutions that look more like realistic images, in this chapter, we use a GAN-based motion reduction model. The complete pipeline of the motion reduction process is illustrated in Fig. 5.2.

The GAN architecture includes a generator and a discriminator, where the generator is used to reduce most of the motion artifacts and the discriminator aims to distinguish if the input image is the clean image (i.e., True) or the output image from the generator (i.e., Fake), so as to further help the generator perform motion reduction. In other words, in order to fool the discriminator, our generator will be trained to output images that look like the ground truth (clean) images as much as possible. The generator is made up of two U-net like blocks [107], each including an encoder to extract the motion patterns and a decoder to restore the image to its original size. The discriminator contains 3 blocks, where each block combines convolutional, batch normalization, and activation function LeakyReLU (Conv+BN+LeakyReLU) operations. Finally, a fully-connected layer and a sigmoid function are used in the end to output a probability of the input image to be real or fake. If the input image is real, the probability should be close to 1, and if it is fake, the probability should be close to 0. Additional details about the generator and the discriminator can be found in Fig. 5.2.

5.2.3 The Loss Functions

The loss function L_{gen} used for training the generator in our GAN model is a combination of the pixel loss L_{pix}, the perceptual loss L_{per}, and the adversarial generator loss L_{GAN_G}, that is,

$$L_{gen} = L_{pix} + \lambda_1 L_{per} + \lambda_2 L_{GAN_G}, \tag{5.4}$$

where λ_1, λ_2 are hyper-parameters, representing the relative importance of the three loss terms. L_{pix} is the Mean Absolute Error (MAE) loss defined in the pixel space, that is,

$$L_{pix} = \frac{1}{hwc}||I_{mr} - I_c||_1, \tag{5.5}$$

where I_{mr} is the output of the generator G, I_c is the corresponding ground truth clean image, h and w represent the height and the width of images, respectively, and c represents the number of image channels.

L_{per} is the perceptual MSE (Mean Squared Error) loss defined in the feature space, that is,

$$L_{per} = \frac{1}{hwc'}||\phi_i(I_{mr}) - \phi_i(I_c)||_2, \tag{5.6}$$

where $\phi_i(.)$ is the feature map generated from layer i of the pre-trained Vgg-16 model [115], and c' represents the number of feature map channels. The Vgg-16 model used to calculate the perceptual loss is pre-trained on the ImageNet dataset [116].

Finally, L_{GAN_G} is the adversarial loss of the generator, which is defined as

$$L_{GAN_G} = \mathbb{E}_{I_m}[-logD(G(I_m))], \tag{5.7}$$

where D represents the discriminator, I_m is the input image with motion artifacts, and $G(I_m)$ is the output image of the generator G, i.e., $G(I_m) = I_{mr}$.

In terms of the discriminator, the training loss is the adversarial loss of the discriminator, which is defined as

$$L_{dis} = \mathbb{E}_{I_c}[-logD(I_c)] + \mathbb{E}_{I_m}[-log(1 - D(G(I_m)))]. \tag{5.8}$$

5.3 Experiments

In total, 202 clean abdominal MRI 512 × 512 images from 15 subjects are included in this study. 8 subjects are used for training (112 images), 3 subjects for validation (40 images) and 4 subjects (50 images) for testing. Since three types of motion artifacts with different motion severities are generated from the clean images, we have in total of 3672 training images, 1044 validation images, and 500 testing images.

In order to train our proposed model, there are two training stages. In the first stage, we pre-train the generator network only, using the pixel loss function defined in Equation 5.5. In the second stage, we train the GAN model, where the generator's weights are first initialized by the pre-trained model from the first stage. Next, the generator and discriminator are trained jointly. The loss function used to train the generator is defined in Equation 5.4 and the discriminator's training loss is defined in Equation 5.8. Besides initializing the generator's weights from a reasonable starting point, in order to ensure the stability and proper convergence of the GAN training, we also need to prevent our discriminator from learning too fast. Otherwise, the generator can never fool the discriminator, thus causing the failure of the adversarial training. Therefore, we train the generator 4 times more than the discriminator. Specifically, after training the generator and discriminator in the first training epoch, the discriminator is fixed and will be updated again until the 5th epoch. The same process repeats in all the following training epochs.

Some ablation studies have been completed to show the effects of the proposed GAN architecture, the effectiveness of the perceptual loss and the importance of the pre-training step. In this study, we compare four models in their performances of reducing artifacts and motion: **1)** the CNN generator only (CNN-onlyG), **2)** the GAN model with no-pretrained generator (GAN-nopreG), **3)** the GAN model with pre-trained generator (GAN-preG) and **4)** the perceptual GAN model with pre-trained generator (percepGAN-preG). All four models are trained and tested on the same dataset.

The CNN-onlyG model uses only the denoising CNN generator, and it is trained with the pixel loss defined in Equation 5.5. Besides, it is trained with smaller image patches (64 × 64), which are randomly selected from the full-size images (512 ×512). Later on, this fully trained model is used as the pre-trained generator in model GAN-preG and model percepGAN-preG. The GAN-nopreG model uses the GAN model with the generator trained from scratch, and it does not use the perceptual loss, so it only uses the pixel and adversarial losses, i.e., the combination of Equations 5.5 and 5.7. The GAN-preG model also does not use the perceptual loss. Finally, the percepGAN-preG model is our proposed model which uses the perceptual loss during training, and the perceptual loss is calculated from the $15^{th}, 16^{th}$ and 17^{th} layers of the Vgg-16 model ($i = 15, 16, 17$ in Equation 5.6). The generator's training loss function is defined in Equation 5.4 with $\lambda_1 = 0.6$ and $\lambda_2 = 0.8$, which are defined experimentally.

All GAN models are trained with full-size images. All training processes use the Adam optimizer [117] with an initial learning rate of 0.0001. The batch size of the CNN-onlyG model is 16 and of the rest GAN-related models are 2. All training epochs are set to 50 with early stopping with a patience of 5. The training time takes an average of 4.3 hours for each experiment and testing a single image takes less than 0.5s. All training and testing processes are performed using Tensorflow 2.0 in Python 3.8 on two GPUs (NVIDIA Titan V).

The mean squared error (MSE), the structural similarity index (SSIM) [118], the peak signal-to-noise ratio (PSNR) [119] and a perceptual distance metric [120] are measured on the testing data to compare the similarity between the model output and the clean image. A paired t-test is performed to compare the objective performance of each of these four models. The comparison is statistically significant if the $p < 0.05$. All statistical analyses are performed using the Scikitsklearn and NumPy packages in Python 3.8.

Table 5.1: Comparisons of four model performances (mean ± std) in terms of MSE, SSIM, PSNR and perceptual distance (the smaller the better) on the testing dataset. * denotes the comparison is statistically significant ($p < 0.05$).

Metrics/Models	CNN-onlyG	GAN-nopreG	GAN-preG	**percepGAN-preG**
MSE	0.012 ± 0.007	0.005 ± 0.002	0.004 ± 0.002	**0.002** ± **0.001***
SSIM	0.857 ± 0.089	0.913 ± 0.014	0.928 ± 0.026	**0.942** ± **0.022***
PSNR	25.944 ± 2.613	29.409 ± 1.799	31.409 ± 1.995	**33.894** ± **2.435***
Perceptual Distance	0.229 ± 0.035	0.066 ± 0.029	0.069 ± 0.032	**0.015** ± **0.006***

5.4 Results

5.4.1 Experimental Results

We compare the quantitative results among four models using the same testing dataset, as shown in Table 5.1. The results show that the percepGAN-preG model significantly outperforms other models in all metrics with SSIM of 0.942 ± 0.022, MSE of 0.002 ± 0.001, PSNR of 33.894 ± 2.435 and perceptual distance of 0.015 ± 0.006. In addition, the overall performance of the GAN model is better than that of the CNN denoising model, indicating the important role of the adversarial training in this motion reduction task. Furthermore, the model with pre-trained weights (GAN-preG) achieves a better motion reduction performance (SSIM: 0.928 ± 0.026, PSNR: 31.409 ± 1.995) compared with the model trained from scratch (GAN-nopreG) (SSIM: 0.913 ± 0.014, PSNR: 29.409 ± 1.799). Finally, our proposed percepGAN-preG model outperforms the GAN-preG model, indicating that the perceptual loss further improves the generated images' visual quality.

Table 5.2 compares the performance of our proposed model on the three types of simulated

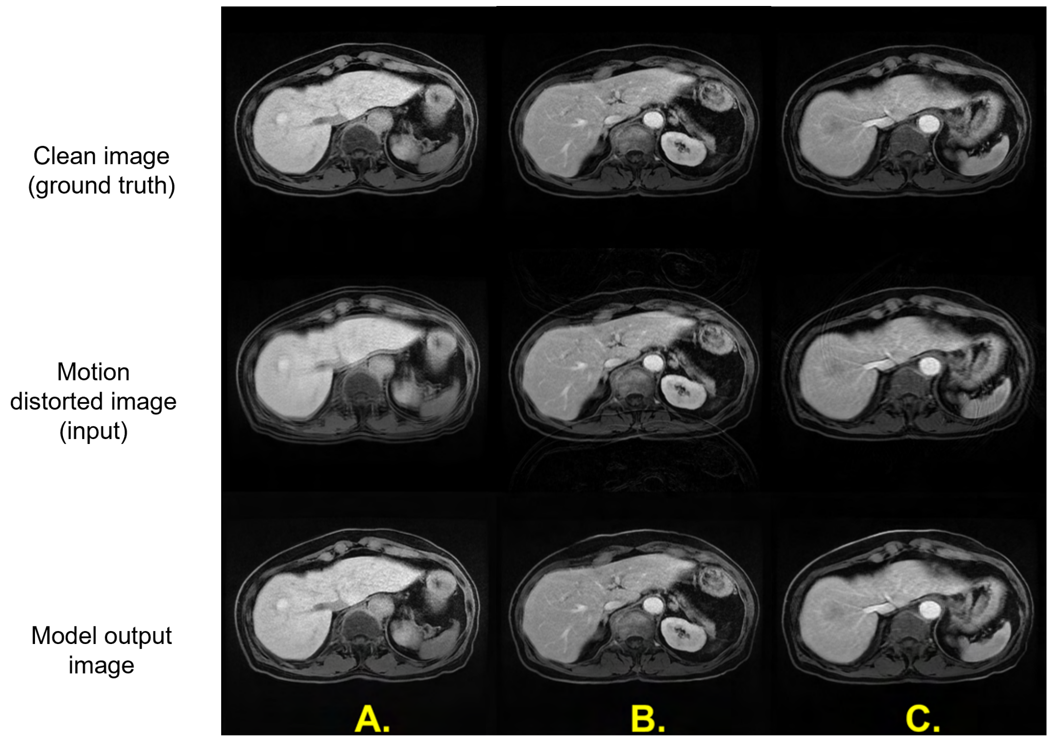


Figure 5.3: Motion reduction examples by the proposed model on three types of motion simulation: A. random motion; B. interweaving motion and C. sinusoidal motion. The first row: clean images; second row: images with specific simulated motion; third row: motion reduced images by the proposed model.

motion data. The model achieves the best motion reduction performance on the sinusoidal type of motion, as shown in Fig. 5.3C (SSIM: 0.953 ± 0.019, PSNR: 36.763 ± 2.145), followed by the random one (Fig. 5.3A, SSIM: 0.938 ± 0.037, PSNR: 31.087 ± 3.224) and the interweaving one (Fig. 5.3B, SSIM: 0.944 ± 0.023, PSNR: 34.932 ± 2.758). The sinusoidal type of motion simulates the periodic breathing patterns and the interweaving type of motion simulates the body movement patterns. The results are in line with our simulation purposes, i.e., the random motion is the most complex type of motion that represents the non-periodic and irregular respiratory patterns.

Table 5.2: Comparisons of motion reduction performance (mean ± std) in MSE, SSIM, PSNR on three different motion types of testing dataset. * denotes the comparison is statistically significant ($p < 0.05$).

Metrics/Motion Types	Type I (Sinusoidal)	Type II (Random)	Type III (Interweave)
MSE	0.002 ± 0.001*	0.004 ± 0.002	0.002 ± 0.002
SSIM	0.953 ± 0.019*	0.938 ± 0.037	0.944 ± 0.023
PSNR	36.763 ± 2.145*	31.087 ± 3.224	34.932 ± 2.758

5.4.2 Visualizations

Figure 5.3 illustrates three representative types of the simulated motion and their corresponding motion reduced images after our proposed percepGAN-preG model. It is clear that our proposed model is able to reduce most of the motion artifacts and at the same time, preserve the tissue details in the abdominal MRIs. Figure 5.4 provides two more examples to further compare the different motion reduction performances among the four compared models. The CNN-onlyG model uses the pixel-to-pixel level MSE loss, leading to over-smoothing and blurring problems, which agree with the observations in study [120]. However, this problem can be alleviated by introducing a GAN model with a discriminator to distinguish between the real and fake images so as to fine-tune the generator and improve the image sharpness. In addition, the perceptual loss computes the difference of two images with a large receptive field, providing another way to restore specific anatomical structures, so the percepGAN-preG model achieves the highest visual quality again. The important role of the pre-trained weights used in the generator is addressed in this chapter. As shown in Fig. 5.4, although as we observe, more motion artifacts can be reduced by the GAN-nopre model compared to the CNN-onlyG model, the reconstruction shows a brighter contrast compared with the ground truth. In this model, the generator is initialized with the random weights so it needs to learn all basic image information from scratch, including the image contrast and the abdominal anatomy. The generator, therefore, will output the fake images with many unreal artifacts that can be easily distinguished by the discriminator, thus the discriminator's job becomes considerably easier, and the generator can hardly be trained to compete with the discriminator afterwards. Therefore the adversarial training process may not be able to converge to a proper solution. The GAN model with a pre-trained generator (GAN-preG) avoids this problem by already having a good starting point for the generator, thus the discriminator needs to be better trained to distinguish the real and fake images, i.e., to better learn the true distribution of real images, and in return, the generator needs to generate images that look more like the real images' distribution to fool the discriminator.

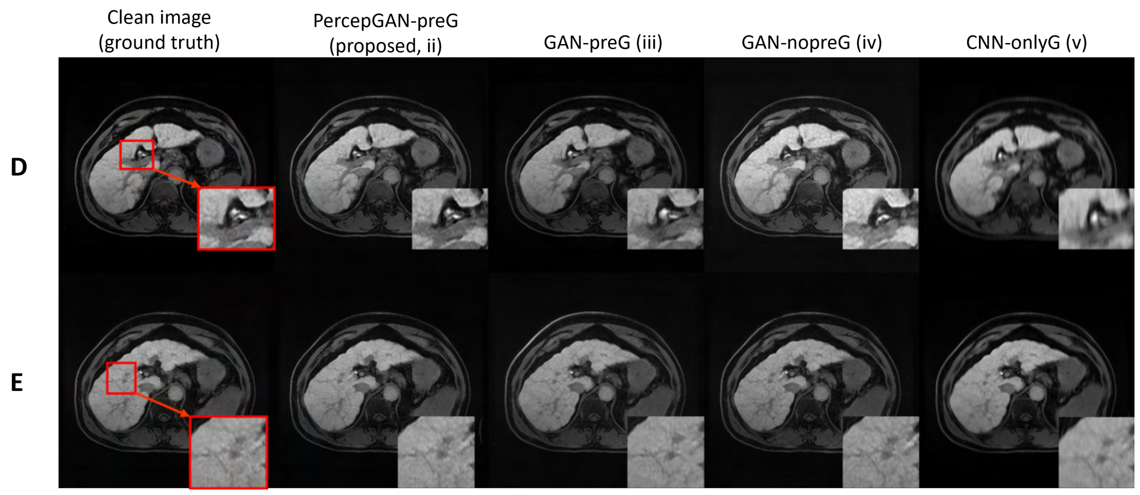


Figure 5.4: Comparison of motion reduction performance among four models and two different abdominal images (D and E). They are the proposed perceptual-GAN model with a pre-trained generator (column ii), the GAN model with a pre-trained generator (column iii), the GAN model with no pre-trained generator (column iv) and the CNN denoising model (column v).

5.5 Conclusion

In this chapter, a novel GAN-based CNN model trained with perceptual loss is developed for automatic motion reductions of abdominal MRIs. Three types of motion simulation methods are proposed to mimic the real motions. By comparing four different models, our proposed model proves that using the GAN-based model with the pre-training step and the perceptual loss can greatly reduce motion artifacts and preserve medical image details. This real-time application of the methodology can play a significant role in improving future clinical diagnosis and it can be potentially applied to other medical motion types.

CHAPTER 6
AVIAN ACTIVITY CLASSIFICATION USING RECURRENT NETWORKS

From this chapter, we move into the second area of this dissertation, deep learning in classification with videos. In this chapter, we will solve an avian activity classification task. The initial motivation of this project is to protect the birds from colliding with the solar panels. During the daytime, Strong sunlight causes reflections on these panels, potentially misleading birds into colliding with them, resulting in injuries or fatalities. (Please note this chapter restates text and figures from our accepted work, which will be published at the 2024 International Conference on Computing and Artificial Intelligence (ICCAI 2024).)

6.1 Introduction

The success of deep learning (DL) methods in computer vision has opened promising avenues for video-based activity classification, encompassing both human and animal behaviors [121]–[128]. However, animal activity classification poses unique challenges, including unpredictable behaviors and limited datasets. Many state-of-the-art DL models for video-based animal activity classification adopt a dual-stream approach, utilizing RGB raw frames and corresponding optical flow as inputs. For instance, [129] and [130] employed convolutional neural networks (CNNs) to extract features from RGB frames and optical flow, merging these streams to predict cattle or mouse behaviors. Similarly, Schindler et al. [131] extended this approach to classify a wider range of wild animal activities. Given the sequential nature of video data, recurrent neural networks (RNNs) such as Long Short-Term Memory (LSTM) networks have been favored, as demonstrated by [132], who used an LSTM network to predict salmon feeding activities. This chapter focuses on video-based animal activity classification, specifically avian/bird activity. We propose a DL-based method, designing a Bidirectional LSTM (Bi-LSTM) model tailored for avian/bird activity classification.

Solar energy is widely recognized as a crucial green energy source, with solar panels increasingly deployed in open areas. However, these solar facilities can impact bird communities in various ways, including species richness, migrations, and activities [133]–[135]. Understanding the interactions between birds and solar infrastructure is essential for the sustainable deployment of utility-scale solar facilities. Currently, monitoring avian-solar interactions and classifying avian activities rely on human observers, which is costly, time-consuming, and prone to errors. It also poses safety concerns for observers working outdoors. Therefore, our study aims to develop a DL model capable of automatically classifying interactions between birds and solar panels.

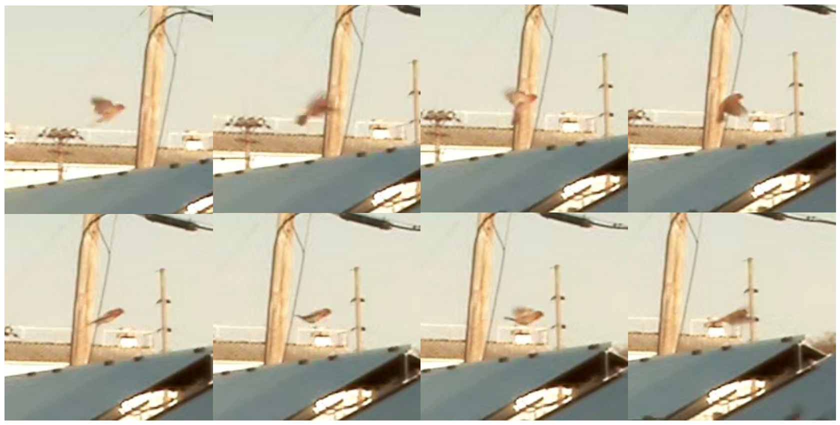


Figure 6.1: Video track example.

DL methods for studying bird behavior via video data are scarce in the current literature. Existing DL-based bird activity classification focuses mainly on acoustic signals [136]–[138]. Since March 2020, researchers at Argonne National Laboratory (Argonne) have collected over 6,000 hours of daytime video at five operational solar energy facilities using high-definition true-color video cameras. By processing these videos, Argonne has generated and labeled bird-centered video tracks of avian activities around the solar energy facilities, such as fly over above, fly-through, and perching. In this chapter, we train our proposed model using this avian activity video dataset. Several challenges must be addressed, such as the significant imbalance between activity categories, a common issue in many classification tasks [139], [140]. Moreover, compared to state-of-the-art video-based animal activity classification, our avian activity classification is more challenging. Previous works in video-based animal activity classification involved no camera movement and relatively static backgrounds [129]–[132]. However, in our avian activity dataset, video tracks are bird-centered, and birds often move quickly, resulting in rapidly changing backgrounds. Additionally, distant bird objects may appear tiny. To address these challenges, we integrate not only video track frames but also critical engineering features (metadata) provided by Argonne with each video track into our proposed model, leading to improved predictions.

Additionally, few DL works in video-based activity classification analyze their model's interpretability, which assesses whether the trained models make logical decisions. Ideally, the model should focus on areas where the activity occurs, for example, the areas around the object whose activity is being classified throughout the video. These analyses are common and crucial in state-of-the-art DL image-based classification methods [141]–[143]. In summary, in this chapter: 1) We propose a novel DL method for video-based avian activity classification. Our Video-Meta Fusion Bi-LSTM model classifies six bird activities using both video data and additional engineered features (metadata). 2) We effectively handle the challenge of training with an imbalanced dataset. 3) After training, we verify that our model focuses on the relevant areas by computing and examining saliency/backpropagation maps. 4) To the best of our knowledge, this study is the first to utilize RNNs for video-based avian activity classification while also analyzing the decision-making rationale of the proposed DL model in animal activity classification.

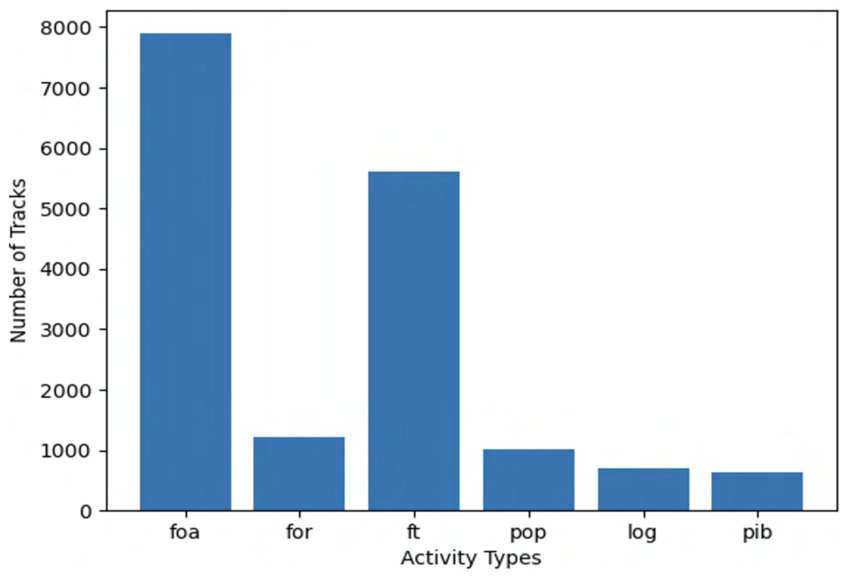


Figure 6.2: Number of samples in each activity.

6.2 Proposed Method

6.2.1 Avian Activity Dataset

Argonne has provided us with an avian activity dataset that consists of video ”tracks” and their corresponding feature engineering features (metadata). Each track is a bird-centered sequence of frames for a single bird. An example can be found in Figure 6.1. The metadata, containing the bird’s trajectory (x/y coordinate information) and its moving speed, is pre-computed by Argonne researchers and directly provided within the dataset for each track.

Based on the activities of birds around the solar energy facilities, the dataset is categorized into six activities:

1. Fly over above (foa): a bird flying high above solar panels.

2. Fly over reflection (for): a bird’s reflection flying over panels.

3. Fly through (ft): a bird flying near solar panels.

4. Perch on panel (pop): a bird flying into the frame lands on any part of the panel structure, or a bird on the panel flies away.

5. Land on the ground (log): a bird flew in and landed on the ground, a bird on the ground flew away, or a bird is moving on the ground.

6. Perch in background (pib): a bird flying into the frame lands on objects other than panels, or a bird on a non-panel object flies away.

Figure 6.2 shows the number of samples for each activity category. We can clearly see a heavy imbalance in the dataset. There are two major activities - the foa and ft, and four minor activities.

6.2.2 Model Description

Before proposing the final fusion model, we first introduce two sub-models that utilize either the metadata sequence input (Meta Bi-LSTM) or the video sequence input (Video Bi-LSTM).

6.2.2 Video Bi-LSTM & Meta Bi-LSTM

The Video Bi-LSTM architecture takes the video frame sequence as input and predicts the probability for each activity category. As shown in Figure 6.3(b), after the video frame sequence is inputted into the model, we first use convolutional, normalization and pooling layers to extract the deep features from them. These features are then flattened into vectors and passed through a Bi-LSTM block (composed of two stacked bidirectional LSTM layers, as shown in Figure 6.3(d)), followed by fully-connected dense layers. Finally, the output layer utilizes a softmax function to provide the predicted probability for each activity category. The Video Bi-LSTM model can be

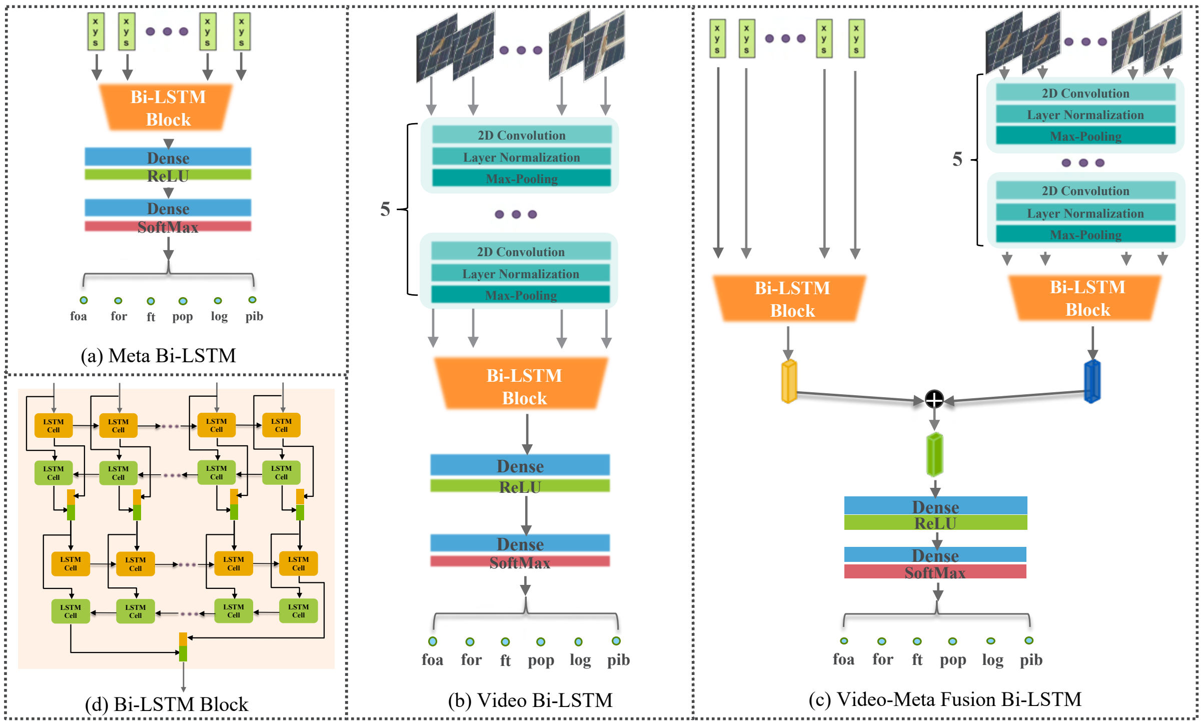


Figure 6.3: Meta Bi-LSTM, Video Bi-LSTM, Video-Meta Fusion Bi-LSTM models' architecture.

described as:

$$p = f_{video}(\boldsymbol{v}), \tag{6.1}$$

where $\boldsymbol{v}$ denotes the input video frame sequence, p is the output vector of the predicted probabilities for C activity categories, and $f_{video}(\cdot)$ denotes the Video Bi-LSTM model.

In the dataset, some bird video tracks can be very long and last for numerous frames, and we would like our model to get information from both past and future frames during training. Therefore, we employ bidirectional LSTM layers rather than pure LSTM layers when building the models [144].

As explained in Section 1, incorporating engineered features can be beneficial for our task. The current video tracks are bird-centered, which loses the general bird trajectory information in the larger camera field of view. Since the x/y coordinate information represents the trajectory of the bird's movement, and the speed of the bird can differ for different activities along the trajectories, we choose to include the x and y coordinates and the bird's speed as the critical metadata. Figure 6.3(a) shows the Meta Bi-LSTM model, which solely uses the metadata sequence as input. That is, the input of this model is a sequence of 3-dimensional vectors (x-coordinate, y-coordinate, speed). The Meta Bi-LSTM model can be described as:

$$p = f_{meta}(\boldsymbol{m}), \tag{6.2}$$

where $\boldsymbol{m}$ denotes the input metadata sequence, p represents the output vector containing the predicted probabilities for C activity categories, and $f_{meta}(\cdot)$ denotes the Meta Bi-LSTM model.

6.2.2 Video-Meta Fusion Bi-LSTM

The final proposed model, Video-Meta Fusion Bi-LSTM (VM Bi-LSTM), is shown in Figure 6.3(c). It incorporates two input streams: the video sequence and the metadata sequence. It fuses these two streams after the Bi-LSTM blocks, where two feature representations are captured and learned from the metadata and video frames along the temporal dimension. The addition of these two deep feature vectors is then input into the following dense layers. The Fusion Bi-LSTM model can be described as:

$$p = f_{fusion}(\boldsymbol{v}, \boldsymbol{m}), \tag{6.3}$$

where the VM Bi-LSTM model, denoted as $f_{fusion}(\cdot)$, takes both the video frame sequence $\boldsymbol{v}$ and its corresponding metadata sequence $\boldsymbol{m}$ as inputs. p is the output vector of the predicted probability for C activity categories. The proposed VM Bi-LSTM model can then learn to classify the activity by combining the information from engineered features and raw video sources. In section 4, we show that the proposed fusion model can achieve better results than the two sub-models.

6.2.3 Training Objective

We use the categorical cross-entropy loss as the loss function for training the Video Bi-LSTM, Meta Bi-LSTM, and VM Bi-LSTM models:

$$\mathcal{L} = -\sum_{c=1}^{C} y_c log(p_c), \tag{6.4}$$

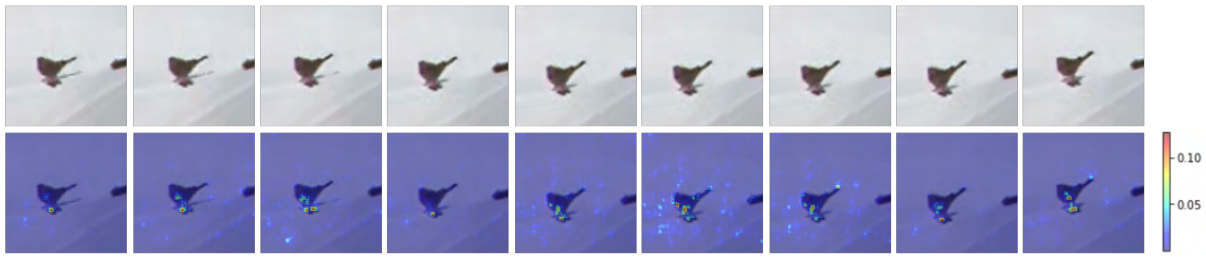
where $[y_1, y_2, ...y_C]$ is the one-hot ground-truth activity label vector with C activity categories, and p_c is the predicted probability for the cth activity category.

6.2.4 Data Augmentation in Metadata and Video Data

Our models are trained with a heavily unbalanced dataset, and this causes severe issues if we directly use it to train our models. Therefore, we need to balance the dataset. In general, we use up-sampling methods for the four minor activities: we apply image augmentation on each frame in the video tracks, like changes in brightness, saturation, or contrast. Since we are augmenting the video tracks instead of the single images, we use the same augmentation process on all frames within each video track to maintain temporal consistency along these frames. The augmentation process can differ from track to track. Besides, we keep the same metadata values for those augmented tracks as the tracks they are upsampled from.

6.3 Experiments

This section presents the experimental results and comparisons among the Video Bi-LSTM, Meta Bi-LSTM, and VM Bi-LSTM models on the avian activity dataset. We use the saliency map to


(a) A perching on panel (pop) test video track clip (top row). The corresponding saliency heat maps overlayed onto the original video frames (bottom row).

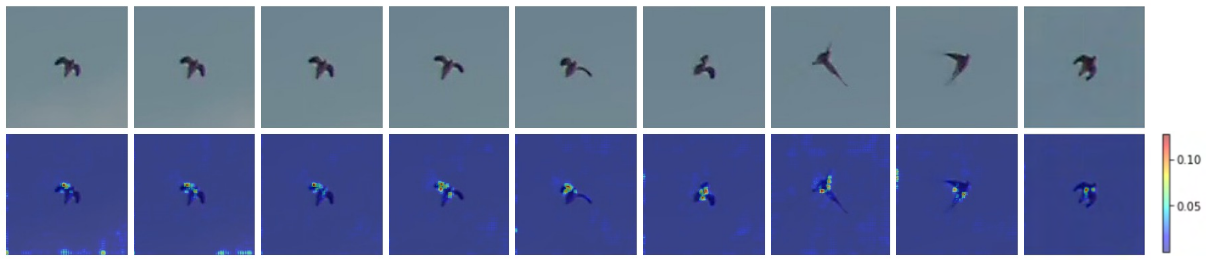

(b) A fly over above (foa) test video track clip (top row). The corresponding saliency heat maps overlayed onto the original video frames (bottom row).

Figure 6.4: Saliency maps of the test video tracks.

analyze our model and validate the rationality of its predictions. Lastly, we assess the performance of our model. **Dataset**. The avian activity dataset provided to us by Argonne is not publicly available. It consists of a total of 17,059 bird-centered video tracks, with the two major activities (foa and ft) accounting for over half of the dataset. The metadata information is pre-computed by Argonne researchers and provided to us directly within the dataset for each track. When utilizing this dataset, We separate the dataset into training (80%), validation (10%), and testing (10%) sets. As described in section 6.2.4, during training the Video Bi-LSTM and Meta Bi-LSTM models, we perform data augmentation to up-sample the four minor activities (for, pop, log, and pib) to 2000 tracks each and randomly down-sample the two major activities to 2000 tracks each. In the case of the VM Bi-LSTM model, which is larger than the two sub-models and has more trainable parameters, we slightly increase the up-sampling and down-sampling numbers to 2500 tracks each category. The video track frames have dimensions of 200×200 pixels. During training, validating, and testing, we center-crop the frames to 100×100 pixels. **Train & test process**. During training, we set the total epochs to 40. We use the validation loss to monitor the training process and apply early stopping with patience 10. Adam is used as the optimizer with the learning rate equal to 0.001. The batch size is chosen to be one as the length of video tracks in the dataset varies a lot.

6.4 Results

6.4.1 Results and Comparisons

To demonstrate the limitations of training models with an unbalanced dataset, we present a comparison of confusion matrices in Figure 6.5 between the Meta Bi-LSTM models trained on the original unbalanced dataset and the balanced dataset. When trained with the unbalanced dataset, we can see that the model can only learn to classify the two major activities (foa, ft), while considering all the other minor activities as foa or ft, as shown in Figure 6.5. After using the balanced dataset, the model now makes predictions across six activities, i.e., it successfully recognizes that there are six classes rather than just two.

Using the balanced dataset, we train and compare the performance of Video Bi-LSTM, Metadata Bi-LSTM, and Video-Meta Fusion Bi-LSTM models. Table 6.1 presents the comparison of test accuracy. We can see that the fusion model performs the best, it can reach 76.9%, 89.9%, and 95.2% accuracy in Top1, Top2, and Top3 test accuracy, respectively. This comparison confirms our intuition that fusing the critical metadata with the raw video input enables DL models to leverage more information, enhance their learning process, and improve their ability to distinguish between different activities.

6.4.2 Model Analysis

We use the saliency/backpropagation maps [141] to validate that our trained model reasonably classifies each activity. The saliency maps indicate the areas of the frames that the model focuses

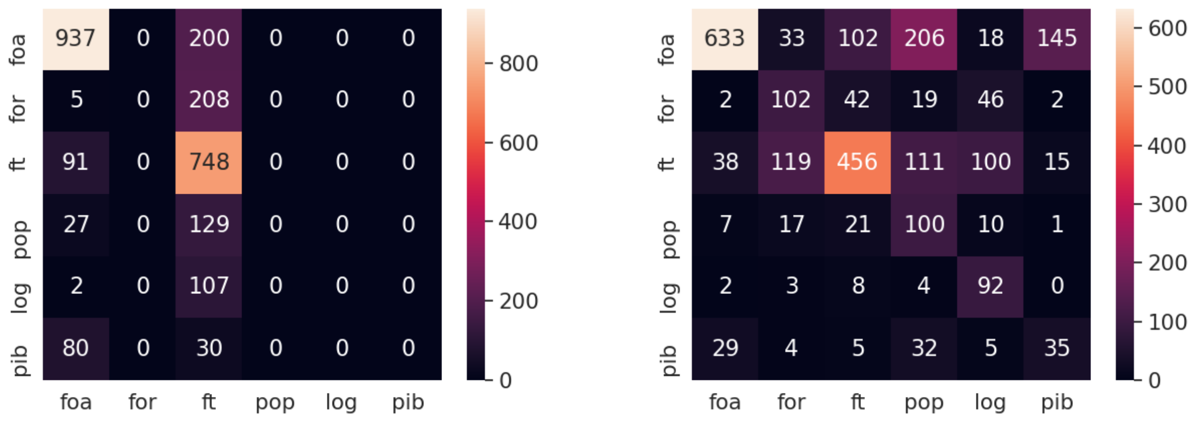


Figure 6.5: The confusion matrices of Meta Bi-LSTM model trained with unbalanced (left) or balanced (right) datasets.

Table 6.1: Test accuracy of Meta Bi-LSTM, Video Bi-LSTM, and Video-Meta Fusion Bi-LSTM models. If the top 2 or top 3 predicted activities contain the given ground-truth activity, then we count it as a correct classification within the Top2 or Top3 test accuracy correspondingly.

	Top1	**Top2**	**Top3**
Meta Bi-LSTM	55.3%	78.2%	89.9%
Video Bi-LSTM	72.5%	86.4%	94.0%
Video-Meta Fusion Bi-LSTM	**76.9%**	**89.9%**	**95.2%**

on when making decisions. For an input video sequence with N contiguous frames, denoted as F_i ($i = 1, 2, ..., N$), its saliency map W_i for each frame is calculated according to

$$W_i = |\frac{\partial \boldsymbol{s}_c}{\partial F_i}|, \tag{6.5}$$

where $\boldsymbol{s}_c$ is the scalar score of class c - the model's output before the final SoftMax layer for the cth class. The derivative in the equation calculates the gradient of $\boldsymbol{s}_c$ at (each pixel of) F_i. The computed saliency maps indicate the areas/pixels in the video frames which affect the c class score the most.

In Figure 6.4, we show the saliency maps of a pop test video track and a foa test video track which are correctly classified by the trained VM Bi-LSTM model's top-1 predicted activity. We can see that the trained model primarily directs its attention towards the bird object in most frames. Moreover, for the pop video track, the model also focuses on the areas of the bird reflected on the solar panels, and this is undoubtedly reasonable. As in a lot of the pop tracks, the bird perches and stays on the panels, and the reflection on the panel is an important factor in determining if the bird is staying/perching on the panels.

6.5 Conclusion

In this chapter, we introduce the Video-Meta Fusion Bi-LSTM model for avian activity classification. Our model integrates information from raw video RGB frames and feature engineering metadata when building the model. This fusion approach yields enhanced test accuracy compared to both the Video Bi-LSTM and Meta Bi-LSTM models. Furthermore, we apply a consistent data augmentation technique to address dataset imbalance. Examination of saliency heatmaps reveals that our model makes decisions in a logical manner.

CHAPTER 7
QUERY-KEY ATTENTION MODELING FOR WEAKLY-SUPERVISED TEMPORAL ACTION LOCALIZATION

In the latter part of our study, we observed that each video track may exhibit multiple activities, like the one shown in Figure 7.1, it contains both the fly-over-above and perch-on-panel activity. But only the perch-on-panel label is given on this video track. Besides, instead of only giving out the activity classification on the video level, we are also curious to find out when each activity happens during the videos. Specifically, we aim to determine the start and end timestamps of these actions. So, we'll need to tackle the weakly-supervised temporal action localization problem. Due to the lack of time and human resources to re-label the avian-solar video dataset, we leverage publicly available datasets to address these issues. (Please note this chapter restates text and figures from our work [145], submitted for journal peer review.)

7.1 Introduction

In recent years, video analysis has been a rapidly developing topic due to the explosive growth of video data used in various real-world applications, especially in the field of video temporal action localization (TAL). The reason for this is that long untrimmed videos contain more interesting foreground activity and useless background activity, and they are more common than short trimmed videos. TAL is a highly challenging task that aims at predicting the start and end times of all action instances and identifying their categories in untrimmed videos. Many works have been done in a fully-supervised manner, where both the video-level labels and the temporal boundary annotations are provided during training [146]–[150]. In contrast, the weakly-supervised temporal action

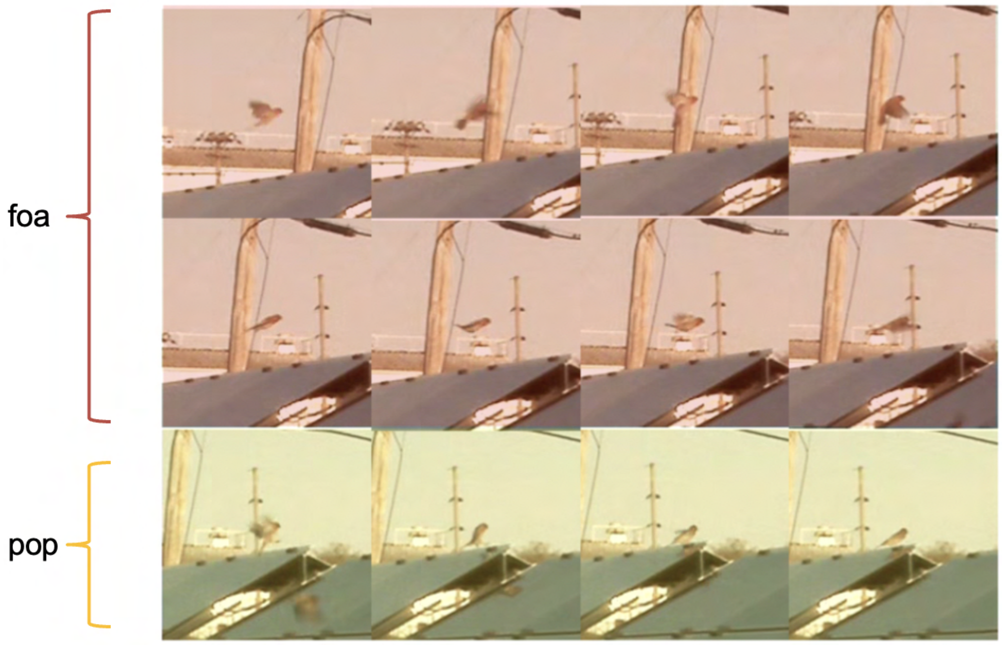


Figure 7.1: A video track contains multiple activities.

localization (WTAL) task attempts to rely only on video-level labels to localize action instances, which can significantly relieve the high cost of manually annotating the temporal boundaries.

Similar to other weakly-supervised video understanding tasks [151]–[153], many WTAL methods employ a multiple instance learning (MIL) strategy [154]–[160]. This approach involves computing segment-level class probability scores, aggregating top scores for each class as video-level class scores, and utilizing video-level classification losses for optimization based on provided video-level labels.

With the success of Transformer [161] in many computer vision tasks [162], [163], some recent TAL works build their models based on Transformer's encoder-decoder framework [164]–[166] and achieve good results. However, all these works aim to learn a set of action queries corresponding to the latent representations of a set of time areas (action proposals). Few works attempt to solve the TAL task in such a way that the abstract-level knowledge of each action category is

Figure 7.2: Action category queries are learned so as to contain action knowledge at the abstract level, which can be used to identify and detect corresponding actions in the target video.

learned and used to recognize and detect the corresponding actions in various video scenes, just like how humans do. The closest work to this idea is STPN[167]. However, it is limited to learning a uniform set of weight parameters for action categories using a fully connected layer.

In this chapter, we present a new video-specific query-key attention mechanism and propose our VQK-Net model based on it. Our high-level idea is illustrated in Fig. 7.2. More specifically, we propose to learn the video-specific action category queries that can be adapted in different video scenarios and simultaneously maintain the action core knowledge features used to detect actions in the videos, i.e., the learned action category queries contain abstract knowledge of actions, and they are tailored to the target video scene to optimize the application of this knowledge. To accomplish this, we propose incorporating video features into the action category queries learning process for two reasons: 1) Since the same action can appear differently in different videos, integrating input video information could help learn the action category queries to better fit into different video scenarios. 2) Some action knowledge can be hidden in the video. Therefore, video features can help the model learn the action's core knowledge features. We achieve this by referring to the cross-attention in Transformer's decoder design.

However, one issue we've overlooked is that video features may confuse the model when learning action category queries for actions not present in the input video—meaning there's no useful information about these categories in the input. To address this, we propose a query similarity loss. The rationale behind this approach is that for any two videos containing the same action category, their corresponding action category queries should resemble each other since they both learn the abstract-level knowledge features of that action. This loss function compels the model to learn the core knowledge features of actions by leveraging correlations between videos of similar action categories. Additionally, besides explicitly teaching the model about action categories in the video through video-level classification loss, the query-similarity loss implicitly provides similar information. Consequently, action category queries corresponding to actions present in the video are better learned and enhanced under the guidance of query similarity loss, thereby mitigating the impact of action category queries not present in the video.

The two-stage model training strategy is widely used to solve the TAL task. In the initial feature extraction stage, a pre-trained feature extractor (e.g., I3D [168]), typically trained on a large trimmed dataset (e.g., Kinetics) for general video action classification tasks, extracts video features from the untrimmed video input. Subsequently, a temporal localization model is trained using the extracted video features. In this chapter, we adopt this two-step training strategy as well.

7.2 Proposed Method

In this section, we present a comprehensive explanation of the proposed VQK-Net model for WTAL. We first formulate the WTAL problem in Section 7.2.1 and describe the feature extraction in Section 7.2.2. Then we provide an overview of the main pipeline of VQK-Net in Section 7.2.3. After that, we delve into the key components of the model: query learner and query similarity loss in Section 7.2.4, and query-key attention module in Section 7.2.5. Finally, we detail the training objective functions in Section 7.2.6 and how the temporal action localization is performed in Section 7.2.7. The overview of our model is shown in Fig. 7.3.

7.2.1 Problem Statement

We formulate the WTAL problem as follows: During training, for a video **x**, only its video-level label is given, denoted as $\mathbf{y} = [y_1, y_2, ..., y_{C+1}]$, where C+1 is the number of action categories and the $(C+1)$-th class is the background category. An action can occur multiple times in the video, and $y_i = 1$ only if there is at least one instance of the i-th action category in the video. During testing, given a video **x**, we aim at detecting and classifying all action proposals temporally, denoted as $\mathbf{x}_{pro} = \{(t_s^j, t_e^j, c^j, \varepsilon^j)\}_{j=1}^{r(\mathbf{x})}$, where $r(\mathbf{x})$ is the number of action proposals for video **x**, and $t_s^j, t_e^j, c^j, \varepsilon^j$ denote the start time, the end time, the predicted action category and the classification score of the predicted action category, respectively.

7.2.2 Feature Extraction

Following the previous work in [154], for each input video **x**, we split it into multi-frame segments, each segment containing a fixed number of frames. To handle the variation of video lengths, a fixed number of T segments are sampled from each video. Following the two-stream strategy used in action recognition [168], [169], we extract the segment-level RGB and flow features vectors $\mathbf{x}_{rgb} \in \mathbb{R}^{D/2}$ and $\mathbf{x}_f \in \mathbb{R}^{D/2}$ from a pre-trained extractor, i.e., I3D, with dimension $D = 2048$. At the end of the feature extraction procedure, each video **x** is represented by two matrices $X_{rgb} \in \mathbb{R}^{T\times(D/2)}$ and $X_f \in \mathbb{R}^{T\times(D/2)}$, denoting the RGB and flow features for the video, respectively.

7.2.3 Main Pipeline Overview

Fig. 7.3 shows the main pipeline of our proposed VQK-Net model. For an input video **x**, we refer to the mutual learning scheme [157] to learn the probability of each segment being foreground from two stream features X_{rgb} and X_f: as shown in Fig. 7.3, we first employ three convolution layers with LeakyRelu activations in between and a sigmoid function on X_{rgb} to get the segment-level foreground probability distribution $\mathbf{s}_{rgb} \in \mathbb{R}^T$, and the same to obtain $\mathbf{s}_f \in \mathbb{R}^T$ with X_f . We average them to get the final $\mathbf{s} \in \mathbb{R}^T$: $\mathbf{s} = \frac{\mathbf{s}_{rgb}+\mathbf{s}_f}{2}$.

Then, we first directly concatenate RGB and flow features in the feature dimension, i.e., concatenate X_{rgb} and X_f to form $X \in \mathbb{R}^{T\times D}$, and input X to two convolution layers with LeakyReLU activations in between to learn the final fusion feature $\hat{X} \in \mathbb{R}^{T\times D}$. The query learner module then takes $\hat{X}$ and C+1 randomly initialized learnable action category query embeddings, which can be stacked to form a category query matrix $Q_{init} \in \mathbb{R}^{(C+1)\times D}$, as inputs. In this module, we refer to the Transformer decoder's design [161] with our proposed query similarity loss to learn the final category query matrix $\hat{Q} \in \mathbb{R}^{(C+1)\times D}$, which contains the learned action category queries for C+1 classes. Finally, we feed $\hat{X}$ through a convolution layer to learn the final video features $\hat{K} \in \mathbb{R}^{T\times D}$ of the input video, used as the video key. The learned query matrix $\hat{Q}$ and learned key matrix $\hat{K}$ will be input to the following query-key attention module to produce the temporal class activation map (T-CAM) $A \in \mathbb{R}^{(C+1)\times T}$. The details are discussed in the following Sections.

7.2.4 Query Learner

The query Learner is an essential part of our VQK-Net model. It learns the video-specific action category queries by exploiting both the video features and the correlation between different videos. The final learned queries will be used to query and detect the corresponding actions along the temporal dimension in the input video.

Structure. As we explain in Section 7.1, different videos have different scenarios, so it is beneficial to learn the video-specific action category queries that can best match the input video. Given the input video **x**, we proposed to include the input video features $\hat{X}$ into learning action category queries instead of just learning the action category queries for all the videos based on Q_{init}. In addition, learning action category queries for specific videos provides the possibility of using correlations between videos to further enhance the learned action category queries.

To include the features learned from the input video, we refer to the Transformer decoder's design. The head attention operation function $f_h(\cdot)$ used in our query learner is defined as:

$$f_h(Q, K, V) = HW_O, \tag{7.1}$$

where

$$H = f_a(QW_Q, KW_K, VW_V), \tag{7.2}$$

and

$$f_a(Q, K, V) = \varsigma(\frac{QK^\top}{\sqrt{D}})V. \tag{7.3}$$

Q, K, V are three input matrices, and W_Q, W_K, W_V and $W_O \in \mathbb{R}^{D\times D}$ are learnable parameter matrices. $\varsigma(\cdot)$ takes a matrix as input, and it denotes that each row of its input is normalized using the softmax operation.

As shown in Fig. 7.4(a), in our query learner, a head attention operation will first operate on the initial action category query matrix Q_{init} itself, i.e., $f_h(Q_{init}, Q_{init}, Q_{init})$. After that, a residual connection and Layer Normalization will be used to output Q_1. The video feature $\hat{X}$ will be used in the second head attention operation to adapt action category queries with the video-specific discriminated features, i.e., $f_h(Q_1, \hat{X}, \hat{X})$. The final output of query learner module is the learned

Figure 7.3: Overview of our proposed VQK-Net model. $\otimes$, $\oslash$ and $\odot$ denote the element-wise multiplication, element-wise division and vector inner product.

action category query matrix $\hat{Q}$ for the input video $\mathbf{x}$.

Query similarity Loss. To improve the learned action category queries and achieve better performance, we exploit the correlation between videos and propose a query similarity loss: For the k-th action category, we define a set V_k that contains all the videos in the training set that has this action in their ground-truth labels. For any two videos $\mathbf{x}_i$ and $\mathbf{x}_j$ in V_k, their learned action category query matrices are $\hat{Q}_i$ and $\hat{Q}_j$, and the rows of these matrices $\{\hat{\mathbf{q}}_i^c\}_{c=1}^{C+1}$ and $\{\hat{\mathbf{q}}_j^c\}_{c=1}^{C+1}$ are the learned query vectors for C+1 categories, respectively. Ideally, we would like the k-th category query vectors from these two sets, i.e., $\hat{\mathbf{q}}_i^k$ and $\hat{\mathbf{q}}_j^k$, to have similar representations, because they should contain the same abstract knowledge features for the k-th action category. The query similarity loss is defined as:

$$\mathcal{L}_{QS} = \frac{1}{C+1} \sum_{k=1}^{C+1} \frac{1}{\binom{|V_k|}{2}} \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in V_k \\ \mathbf{x}_i \neq \mathbf{x}_j}} d(\hat{\mathbf{q}}_i^k, \hat{\mathbf{q}}_j^k), \tag{7.4}$$

where $d(\mathbf{e}_1, \mathbf{e}_2)$ is the cosine distance:

$$d(\mathbf{e}_1, \mathbf{e}_2) = 1 - \frac{\mathbf{e}_1 \cdot \mathbf{e}_2}{\|\mathbf{e}_1\| \|\mathbf{e}_2\|}, \tag{7.5}$$

where $\mathbf{e}_1$ and $\mathbf{e}_2$ are two input vectors, $(\cdot)$ is the inner product and $\|\cdot\|$ is the magnitude. The smaller the cosine distance is, the more similar the feature vectors are.

7.2.5 Query-Key Attention

Finally, for the input video $\mathbf{x}$, we have its final learned action category query matrix $\hat{Q}$ and its video features $\hat{K}$ (used as the final video key). As shown in Fig. 7.4(b), each learned action category vector (a row of $\hat{Q}$) will be used to query on the video key $\hat{K}$ at each time step by the vector inner product, and the output value is the attention weight of the corresponding action occurring at a time step. The higher the weight, the more likely that action occurs. Our query-key attention operation is defined as:

$$\psi(Q, K) = \frac{QK^\top}{\sqrt{D}}, \tag{7.6}$$

where Q and K are two input matrices.

The temporal class activation map (T-CAM) A will be computed as:

$$A = \psi(\hat{Q}, \hat{K}), \tag{7.7}$$

which contains the attention weight for each action along the temporal dimension (T). The softmax operation will be performed on T-CAM to calculate some training losses that we illustrate in Section 7.2.6, *e.g.*, the video-level classification loss. The effect of extremely small gradient will possibly be made after the softmax function, since the inner products could grow large in magnitude with a large value of D. Therefore, as defined in Eq. (7.6), we scale the value by $1/\sqrt{D}$ to counteract this effect.

Figure 7.4: (a) Query learner module. (b) Query-key attention module

7.2.6 Training Objectives

We adopt the top-k multiple instance learning strategy [154] to compute the video-level classification loss. Given a training video **x**, since we only have its video-level class ground-truth label, we will use the segment-level scores from its learned T-CAM A to first obtain the video-level class scores by aggregating the top k values along the temporal dimension for each class in A, i.e., aggregating top k values in each row of A:

$$v_c = \frac{1}{k} \max_{\substack{U \subset A_c \\ |U|=k}} \sum_{i=1}^{k} U_i, \tag{7.8}$$

where A_c is a set containing T attention weight values from the c-th row of A. U_i is the i-th element in the set U. We set $k = \max(1, \lfloor \frac{T}{m} \rfloor)$, and m is a hyper-parameter.

After that, we calculate the probability mass function (pmf) over all the action classes by applying softmax operation along the class dimension:

$$p_c = \frac{exp(v_c)}{\sum_{c'=1}^{C+1} exp(v_{c'})}, \tag{7.9}$$

where $c = 1, 2, ..., C + 1$.

The video-level classification loss is computed as the cross-entropy loss between the ground-truth pmf and the predicted pmf:

$$\mathcal{L}^{A}_{VCLS} = -\sum_{c=1}^{C+1} y_c log(p_c), \tag{7.10}$$

where $[y_1, y_2, ...y_C, y_{C+1}]$ is the normalized ground-truth vector, and the background activity is fixed to be a positive class since it always exists in the untrimmed videos.

Following the previous work[158], in order to better recognize the background activity and reduce its impact during inference, we apply the learned **s** (defined in Section 7.2.3) to suppress the background segments on the T-CAM A and obtain the background-suppressed T-CAM: $\hat{A} = \mathbf{s} \otimes A$, in which **s** element-wise multiplies on every row of A. We then also calculate the video-level classification loss $\mathcal{L}^{\hat{A}}_{VCLS}$ on $\hat{A}$, and the background is fixed as a negative class now since it is suppressed. Our final video-level classification loss is denoted as: $\mathcal{L}_{VCLS} = \mathcal{L}^{A}_{VCLS} + \mathcal{L}^{\hat{A}}_{VCLS}$.

As described in Section 7.2.3, we adopt the mutual learning scheme [157] to learn the segment-level probabilities of being foreground action from both the RGB and flow input streams, and $\mathbf{s}_{rgb}$ and $\mathbf{s}_f$ should align with each other as they both represent the foreground probability of each segment along the temporal dimension T, so a mutual learning loss is used as:

$$\mathcal{L}_{ML} = \frac{1}{2}(\|\mathbf{s}_{rgb} - \eta(\mathbf{s}_f)\|_2^2 + \|\eta(\mathbf{s}_{rgb}) - \mathbf{s}_f\|_2^2), \tag{7.11}$$

where $\|\cdot\|_2$ is the L2 norm, and $\eta(\cdot)$ stops the gradient of its input, so that $\mathbf{s}_{rgb}$ and $\mathbf{s}_f$ can be treated

Table 7.1: The comparison with state-of-art TAL works on the THUMOS14 dataset. †refers to using additional information, such as human pose or action frequency. I3D is abbreviation for I3D features.

Supervision	Method	mAP@IoU(%)							AVG mAP(%)		
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.1:0.5	0.3:0.7	0.1:0.7
Fully	TAL-Net [170] (CVPR'18)	59.8	57.1	53.2	48.5	42.8	33.8	20.8	52.3	39.8	45.1
	GTAN [171] (CVPR'19)	69.1	63.7	57.8	47.2	38.8	-	-	55.3	-	-
	VSGN [172] (ICCV'21)	-	-	66.7	60.4	52.4	41.0	30.4	-	50.2	-
	RefactorNet [173] (CVPR'22)	-	-	70.7	65.4	58.6	47.0	32.1	-	54.8	-
Weakly†	3C-Net [174] (ICCV'19)	59.1	53.5	44.2	34.1	26.6	-	8.1	43.5	-	-
	PreTrimNet [175] (AAAI'20)	57.5	54.7	41.4	32.1	23.1	14.2	7.7	41.0	23.7	23.7
	SF-Net [176] (ECCV'20)	71.0	63.4	53.2	40.7	29.3	18.4	9.6	51.5	30.2	40.8
	BackTAL [177] (TPAMI'22)	-	-	54.4	45.5	36.3	26.2	14.8	-	35.4	-
Weakly (I3D)	STPN [167] (CVPR'18)	52.0	44.7	35.5	25.8	16.9	9.9	4.3	35.0	18.5	27.0
	Nguyen *et at* [178] (ICCV'19)	64.2	59.5	49.1	38.4	27.5	17.3	8.6	47.7	28.2	37.8
	ACSNet [179] (AAAI'21)	-	-	-	42.7	32.4	22.0	-	-	-	-
	HAM-Net [158] (AAAI'21)	65.9	59.6	52.2	43.1	32.6	21.9	12.5	50.7	32.5	39.8
	UM [180] (AAAI'21)	67.5	61.2	52.3	43.4	33.7	22.9	12.1	51.6	32.9	41.9
	FAC-Net [181] (ICCV'21)	67.6	62.1	52.6	44.3	33.4	22.5	12.7	52.0	33.1	42.2
	AUMN [182] (CVPR'21)	66.2	61.9	54.9	44.4	33.3	20.5	9.0	52.1	32.4	41.5
	CO_2-Net [157] (MM'21)	70.1	63.6	54.5	45.7	38.3	26.4	13.4	54.4	35.7	44.6
	Zhao *et at* [183] (Multimed. Tools Appl.'22)	64.3	57.9	48.4	38.9	29.7	-	-	47.8	-	-
	BaM+ACGNet [184] (AAAI'22)	68.1	62.6	53.1	44.6	34.7	22.6	12.0	52.6	33.4	42.5
	MMSD [185] (TIP'22)	69.7	64.3	54.6	45.0	36.4	23.0	12.3	54.0	34.3	43.6
	DCC [159] (CVPR'22)	69.0	63.8	55.9	45.9	35.7	24.3	13.7	54.1	35.1	44.0
	FTCL [186] (CVPR'22)	69.6	63.4	55.2	45.2	35.6	23.7	12.2	53.8	34.4	43.6
	ASM-LOC [187] (CVPR'22)	71.2	65.5	57.1	46.8	36.6	25.2	13.4	55.4	35.8	45.1
	Huang *et at* [188] (CVPR'22)	71.3	65.3	55.8	47.5	38.2	25.4	12.5	55.6	35.9	45.1
	DELU [189] (ECCV'22)	71.5	66.2	56.5	47.7	**40.5**	27.2	15.3	56.5	37.4	46.4
	AMS [190] (TMM'22)	69.1	62.3	52.7	42.8	33.1	23.1	13.0	52.0	32.4	42.3
	Xia *et at* [191] (Multimed. Tools Appl.'23)	60.2	54.5	46.9	27.6	28.2	-	-	45.5	-	-
	F3-Net [192] (TMM'23)	69.4	63.6	54.2	46.0	36.5	-	-	53.9	-	-
	VQK-Net (ours)	**72.0**	**66.5**	**57.6**	**48.8**	40.3	**28.1**	**15.7**	**57.0**	**38.1**	**47.0**

as pseudo-labels of each other.

Based on the assumption that an action is detected from a sparse subset of the video segments [167], a sparsity loss $\mathcal{L}_{Sparse}$ is used for the segment-level probabilities $\mathbf{s}_{rgb}, \mathbf{s}_f$, and $\mathbf{s}$:

$$\mathcal{L}_{SP} = \frac{1}{3}(\|\mathbf{s}_{rgb}\|_1 + \|\mathbf{s}_f\|_1 + \|\mathbf{s}\|_1). \tag{7.12}$$

Moreover, since $\mathbf{s}_{rgb}, \mathbf{s}_f, \mathbf{s}$ are the learned segment-level probabilities of being foreground action, they should oppositely align with the probability distribution of the background class, which is learned from the query-key attention operation, i.e., the $(C+1)$-th row of A after it is applied by softmax operation along the column (class) dimension, denoted as $\mathbf{a} = column\_softmax(A)[C+1,\ :\,] \in \mathbb{R}^T$. We use the guide loss [158] to fulfill this goal:

$$\mathcal{L}_G = \frac{1}{3}(\|\mathbf{1} - \mathbf{a} - \mathbf{s}_{rgb}\|_1 + \|\mathbf{1} - \mathbf{a} - \mathbf{s}_f\|_1 + \|\mathbf{1} - \mathbf{a} - \mathbf{s}\|_1), \tag{7.13}$$

where $\|\cdot\|_1$ is the $l1$ norm, and $\mathbf{1} \in \mathbb{R}^T$ is a vector with all element values equal to 1.

We also adopt the co-activity similarity loss $\mathcal{L}_{CAS}$ [154] that uses the video features $\hat{X}$ and suppressed T-CAM $\hat{A}$ to better learn the video features and T-CAM [1].

Finally, we train our proposed VQK-Net model using the following joint loss function:

$$\mathcal{L} = \mathcal{L}_{VCLS} + \alpha\mathcal{L}_{QS} + \mathcal{L}_{ML} + \beta\mathcal{L}_G + \mathcal{L}_{CAS} + \gamma\mathcal{L}_{SP}, \tag{7.14}$$

where α, β, and γ are the hyper-parameters.

7.2.7 Temporal Action Localization

During testing time, given a video **x**, we first calculate the video-level possibility of each action category occurring in the video from background-suppressed T-CAM $\hat{A}$. We set a threshold to discard the categories with probabilities less than the threshold (set to 0.2 in our experiments). For the remaining action classes, we threshold on the segment-level foreground probability distribution **s** to get rid of the background segments and obtain the category-agnostic action proposals by selecting the continuous components from the remaining segments. We calculate the proposal's classification score ε by using the outer-inner score [193] on $\hat{A}$. To enrich the proposal pool with proposals in different scale levels, we use multiple thresholds to threshold on **s**. The soft non-maximum

[1]More details of the co-activity similarity loss a can be found in [154].

suppression is performed for overlapped proposals.

7.3 Experiments

7.3.1 Experimental Settings

Datasets & Evaluation metrics. We evaluate our approach on three widely used action localization datasets: THUMOS14 [194], ActivityNet1.2 [195], and ActivityNet1.3 [195].

THUMOS14 contains 200 validation videos and 213 test videos of 20 action categories. It is a challenging benchmark. The videos inside have various lengths, and the actions frequently occur (on average, 15.5 activity instances per video). We use the validation videos for training and the test videos for testing.

ActivityNet1.2 dataset has 4819 training videos, 2383 validation videos and 2489 test videos of 100 action classes. It contains around 1.5 activity instances per video. Since the ground-truth annotations for the test videos are withheld for the challenge, we utilize the validation videos for testing.

ActivityNet1.3 dataset has 10024 training videos, 4926 validation videos, and 5044 test videos of 200 action classes. It contains around 1.6 activity instances per video. Since the ground-truth annotations for the test videos are withheld for the challenge, we utilize the validation videos for testing.

We evaluate our method with the mean average precision (mAP) at various intersections over union (IoU) thresholds. We utilize the officially released valuation code to calculate the evaluation metrics [195].

Implementation details. In this work, we sample the video streams into non-overlapping 16 frames segments and apply the I3D network [168] pre-trained on Kinetics[204] to extract the 1024-

Table 7.2: Comparisons with state-of-art works on ActivityNet1.2 dataset. AVG means the average mAP from IoU 0.5 to 0.95 with step size 0.05.

Supervision	Method	mAP@IoU (%)			
		0.5	0.75	0.95	AVG
Fully	SSN [196]	41.3	27.0	6.1	26.6
Weakly†	SF-Net [176]	37.8	24.6	10.3	22.8
	Lee *et al* [197]	44.0	26.0	5.9	26.8
	BackTAL[177]	41.5	27.3	14.4	27.0
Weakly(I3D)	DGAM [198]	41.0	23.5	5.3	24.4
	HAM-Net [158]	41.0	24.8	5.3	25.1
	UM [180]	41.2	25.6	6.0	25.9
	ACSNet [179]	41.1	26.1	**6.8**	26.0
	CO_2-Net [157]	43.3	26.3	5.2	26.4
	AUMN [182]	42.0	25.0	5.6	25.5
	D2Net [199]	42.3	25.5	5.8	26.0
	Zhao *et at* [183]	32.7	21.5	8.4	20.9
	VQK-Net (ours)	**44.5**	**26.6**	5.1	**26.8**

dimensional segment-level RGB and flow features. For a fair comparison, we do not finetune the feature extractor. During the training stage, we randomly sample $T = 500$ segments for the THUMOS14 dataset and $T = 60$ segments for the ActivityNet1.2 and ActivityNet1.3 datasets. During the evaluation stage, all segments are taken. The values of α, β, and γ used in Eq. (7.14) were determined experimentally. We found their optimal values to be: $\alpha = 5, \beta = 0.8$, and $\gamma = 0.8$ for the THUMOS14 dataset, and $\alpha = 10, \beta = 0.8$, and $\gamma = 0.8$ for ActivityNet1.2 and ActivityNet1.3 datasets. To determine k in Eq. (7.8), m is set to 7 for the THUMOS14 dataset, 4 for the ActivityNet1.2 dataset, and 6 for the ActivityNet1.3 dataset.

At the training stage, we sample 10 videos as a batch. In each batch, there are at least three pairs of videos such that each pair has at least one action category in common. We use the Adam optimizer [35] with a learning rate of 0.00005 and weight decay rate of 0.001 for THUMOS14, a

Table 7.3: Comparisons with state-of-art works on ActivityNet1.3 dataset. AVG means the average mAP from IoU 0.5 to 0.95 with step size 0.05.

Supervision	Method	mAP@IoU (%)			
		0.5	0.75	0.95	AVG
Fully	SSN [196]	39.1	23.5	5.5	24.0
	PCG-TAL [200]	44.3	29.9	5.5	28.9
Weakly(I3D)	STPN [167]	29.4	16.9	2.6	-
	TSCN [201]	35.3	21.4	5.3	21.7
	UM [180]	41.2	25.6	6.0	25.9
	ACSNet [179]	36.3	24.2	5.8	23.9
	AUMN [182]	38.3	23.5	5.2	23.5
	TS-PCA [202]	37.4	23.5	5.9	23.7
	UGCT [203]	39.1	22.4	5.8	23.8
	FACNet [181]	37.6	24.2	6.0	24.0
	FTCL [186]	40.0	24.3	**6.4**	24.8
	Huang *et at* [188]	40.6	24.6	5.9	25.0
	Xia *et at* [191]	36.2	22.6	5.4	22.4
	VQK-Net (ours)	**42.4**	**26.4**	5.5	**26.3**

learning rate of 0.00003 and weight decay rate of 0.0005 for ActivityNet1.2 and ActivityNet1.3. For action localization, we use multiple thresholds from 0.1 to 0.9 with a step of 0.08, and we perform soft non-maximum suppression with an IoU threshold of 0.7. All the experiments are performed on a single NVIDIA Quadro RTX 8000 GPU.

7.4 Results

7.4.1 Comparison with State-of-art Methods

In Table 7.1, Table 7.2, and Table 7.3, we compare our method with the existing state-of-art weakly-supervised methods and some fully-supervised methods. For the THUMOS14 dataset. We show mAP scores at different IoU thresholds from 0.1 to 0.7 with a step size of 0.1. Our VQK-Net model outperforms recent weakly-supervised approaches and establishes new state-of-the-art results on most IoU metrics. Moreover, our model outperforms some fully-supervised TAL methods and even some recent methods using additional weak supervisions, such as human pose or action frequency. For the ActivityNet1.2 and ActivityNet1.3 datasets, our method also reach state-of-art performance and outperforms some recent fully-supervised methods and the recent methods with additional weak supervisions. These results indicate the effectiveness of our proposed method.

Table 7.4: Evaluation of uniform and video-specific query learning strategies on THUMOS14.

Exp	0.1	0.2	0.3	0.4	0.5	0.6	0.7	AVG mAP (%)	
								(0.1:0.5)	(0.1:0.7)
Uniform	69.7	64.3	55.6	46.7	39.0	26.2	13.8	55.1	45.1
Video-specific	**72.0**	**66.5**	**57.6**	**48.8**	**40.3**	**28.1**	**15.7**	**57.0**	**47.0**

7.4.2 Ablation Studies & Qualitative Results

Analysis on query learning strategies. In the process of designing the query-key (q-k) attention mechanism, we investigate different strategies to learn our action categories queries. The performance of uniform and video-specific strategies was evaluated on the THUMOS14 dataset in Table 7.4. In the table, we first show the results of using the uniform strategy. In this experiment, the model does not include the video features in learning and learns a set of uniform action category queries for all the videos, i.e., the learned queries are not video-specific. The model simply relies on the learnable initial query embeddings, and we do not use the query learner module (Fig. 7.4(a)) in Fig. 7.3. The query similarity loss is not applicable in this case because it relies on the correlation of videos.

While with the video-specific strategy, the model learns the video-specific action category queries, as described in Fig. 7.3 and Section 7.2.4. From the table, it can be observed that the

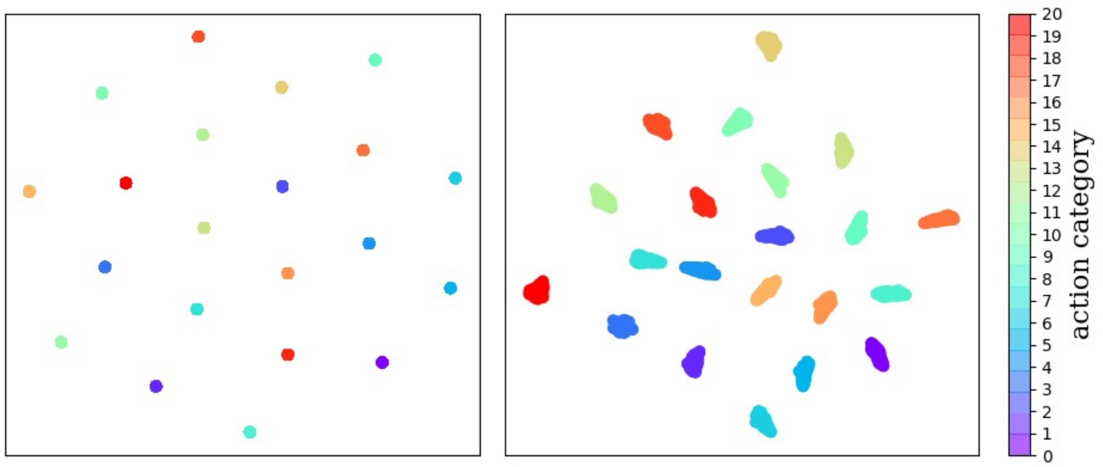
Table 7.5: Analysis of distance function used in query similarity loss on THUMOS14.

Exp	AVG mAP (%)	
	(0.1:0.5)	(0.1:0.7)
Cosine	**57.0**	**47.0**
Jensen-Shannon	55.9	45.7
Euclidean	55.1	45.1
Manhattan	54.8	44.8

video-specific query learning strategy outperforms the uniform strategy quantitatively.

Fig. 7.5(b) shows the visualization of VQK-Net's learned action category queries for C+1 categories (including background) on test videos of THUMOS14 (C = 20), where the video-specific query learning strategy is used. Fig. 7.5(a) shows the learned action category queries using the uniform query learning strategy. From Fig. 7.5(b), we can observe that there are 21 clusters of video-specific action queries for all test videos. This observation aligns with our hypothesis: the learned 21 category queries for each input video contain the abstract action knowledge features of 21 action categories, respectively, and compared to the uniform learning strategy where all videos have the same 21 action category queries (Fig. 7.5(a)), video-specific action category queries have the ability to variant based on different input video scenes, to work optimally under the target video scenario while maintaining the core action knowledge features used to detect and identify actions in the target videos.

We show some representative examples in Fig. 7.6. For each example, the top row represents the ground truth localization. The uniform and video-specific correspond to the experiments from Table 7.4. From Fig. 7.6, we can see that the video-specific query-key attention strategy predicts better localization against the uniform query-key attention strategy, demonstrating the effectiveness of the video-specific query-key attention modeling. Besides the increased precision in the localization, the video-specific approach can correct some missing detections from the uniform



(a) Uniform (b) Video-specific

Figure 7.5: Visualization of the learned action category queries on THUMOS14 test videos via t-SNE[205].

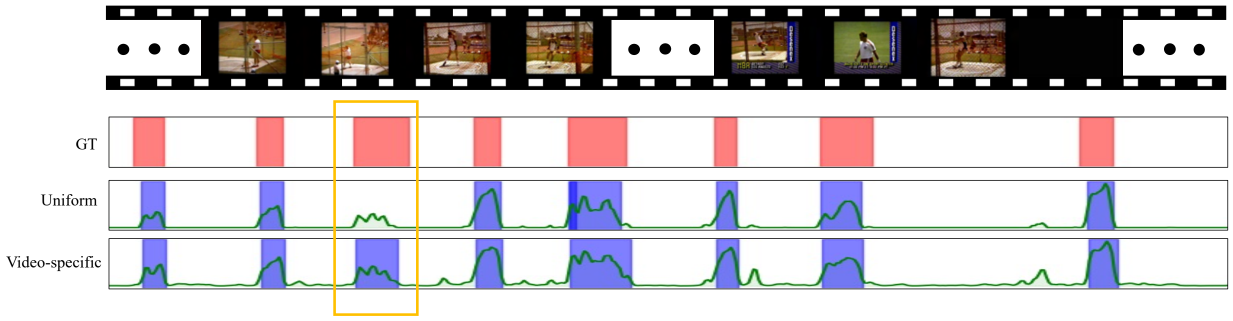
approach. In addition, even though some examples have frequent action occurrences, our VQK-Net model successfully detects all the action instances, which shows the ability to handle dense action occurrences.

Analysis of the distance function in query similarity loss. In Table 7.5, we present the analysis of the distance function used in the query similarity loss (Eq. (7.4)). We can see that the cosine similarity distance performs the best, and the Jensen-Shannon distance is the second, while the Euclidean and Manhattan have a poor performance. This result aligns with the nature of our learned queries. Since the VQK-Net learns the video-specific action queries that could fit under different scenarios, the learned queries should not be precisely identical among different videos, as illustrated in the comparison in Fig. 7.5. Therefore, using absolute distance such as Euclidean, Manhattan, etc., will not be appropriate.

7.5 Conclusion

This chapter introduces the VQK-Net model, which emulates human action localization by employing video-specific query-key attention modeling. VQK-Net learns video-specific action category queries that contain abstract-level action knowledge and can adapt to the target video scenario. We utilize these learned action categories to identify and localize the corresponding activities in different videos. We devise a novel video-specific action category query learner accompanied by a query similarity loss, which steers the query learning process leveraging video correlations. Our method demonstrates state-of-the-art performance on WTAL benchmarks.

(a) An example of CliffDiving action



(b) An example of HammerThrow action

(c) An example of ThrowDiscus action

Figure 7.6: Qualitative results on THUMOS14. The horizontal axis denotes time. The first plot is the ground truth (GT) action intervals. The remaining two plots illustrate the detection scores of ground truth action, shown in green curves, and the detected action instances using the uniform and video-specific query-key attention strategies, respectively.

REFERENCES

[1] H. Chen, X. He, L. Qing, *et al.*, “Real-world single image super-resolution: A brief review,” *Information Fusion*, vol. 79, pp. 124–145, 2022.

[2] K. Chauhan, S. Patel, M. Kumhar, *et al.*, “Deep learning-based single-image super-resolution: A comprehensive review,” *IEEE Access*, 2023.

[3] H. Liu, Z. Ruan, P. Zhao, *et al.*, “Video super-resolution based on deep learning: A comprehensive survey,” *Artificial Intelligence Review*, vol. 55, no. 8, pp. 5981–6035, 2022.

[4] X. Wang, A. Lucas, S. Lopez-Tapia, X. Wu, R. Molina, and A. K. Katsaggelos, “Spatially adaptive losses for video super-resolution with gans,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 1697–1701.

[5] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in *European conference on computer vision*, Springer, 2014, pp. 184–199.

[6] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[7] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, “Video super-resolution with convolutional neural networks,” *IEEE Transactions on Computational Imaging*, vol. 2, no. 2, pp. 109–122, 2016.

[8] J. Kim, J. K. Lee, and K. M. Lee, “Accurate image super-resolution using very deep convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1646–1654.

[9] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, “Enhanced deep residual networks for single image super-resolution,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, vol. 1, 2017, p. 3.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[11] M. S. Sajjadi, B. Scholkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4491–4500.

[12] C. Ledig, L. Theis, F. Huszár, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[13] X. Wang, K. Yu, S. Wu, *et al.*, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[14] A. Lucas, S. Lopez-Tapia, R. Molina, and A. K. Katsaggelos, "Generative adversarial networks and perceptual losses for video super-resolution," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3312–3327, 2019.

[15] E. Pérez-Pellitero, M. S. Sajjadi, M. Hirsch, and B. Schölkopf, "Photorealistic video super resolution," *arXiv preprint arXiv:1807.07930*, 2018.

[16] M. Zhang and Q. Ling, "Supervised pixel-wise gan for face super-resolution," *IEEE Transactions on Multimedia*, vol. 23, pp. 1938–1950, 2020.

[17] S. López-Tapia, A. Lucas, R. Molina, and A. K. Katsaggelos, "A single video super-resolution gan for multiple downsampling operators based on pseudo-inverse image formation models," *Digital Signal Processing*, vol. 104, p. 102 801, 2020.

[18] A. Lugmayr, M. Danelljan, F. Yu, L. Van Gool, and R. Timofte, "Normalizing flow as a flexible fidelity objective for photo-realistic super-resolution," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 1756–1765.

[19] J. Guerreiro, P. Tomás, N. Garcia, and H. Aidos, "Super-resolution of magnetic resonance images using generative adversarial networks," *Computerized Medical Imaging and Graphics*, p. 102 280, 2023.

[20] J. Song, H. Yi, W. Xu, X. Li, B. Li, and Y. Liu, "Esrgan-dp: Enhanced super-resolution generative adversarial network with adaptive dual perceptual loss," *Heliyon*, vol. 9, no. 4, 2023.

[21] K. Jiang, Z. Wang, P. Yi, G. Wang, T. Lu, and J. Jiang, “Edge-enhanced gan for remote sensing image superresolution,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 8, pp. 5799–5812, 2019.

[22] X. Wang, K. Yu, C. Dong, and C. C. Loy, “Recovering realistic texture in image super-resolution by deep spatial feature transform,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 606–615.

[23] X. Wu, A. Lucas, S. Lopez-Tapia, *et al.*, “Semantic prior based generative adversarial network for video super-resolution,” in *2019 27th European Signal Processing Conference (EUSIPCO)*, IEEE, 2019, pp. 1–5.

[24] J. Zhao, Y. Ma, F. Chen, *et al.*, “Sa-gan: A second order attention generator adversarial network with region aware strategy for real satellite images super resolution reconstruction,” *Remote Sensing*, vol. 15, no. 5, p. 1391, 2023.

[25] S. N. Efstratiadis and A. K. Katsaggelos, “Adaptive iterative image restoration with reduced computational load,” *Optical engineering*, vol. 29, no. 12, pp. 1458–1468, 1990.

[26] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[27] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[28] M. Cheon, J.-H. Kim, J.-H. Choi, and J.-S. Lee, “Generative adversarial network-based image super-resolution using perceptual content losses,” in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[29] A. Lucas, A. K. Katsaggelos, S. L. Tapia, and R. Molina, “Generative adversarial networks and perceptual losses for video super-resolution,” in *ICIP*, 2018.

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

[31] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos, “Using deep neural networks for inverse problems in imaging: Beyond analytical methods,” *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 20–36, 2018.

[32] E. Agustsson and R. Timofte, “Ntire 2017 challenge on single image super-resolution: Dataset and study,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 126–135.

[33] R. Zeyde, M. Elad, and M. Protter, “On single image scale-up using sparse-representations,” in *International conference on curves and surfaces*, Springer, 2010, pp. 711–730.

[34] *Myanmar 60p, harmonic inc. (2014)*, `http://www.harmonicinc.com/resources/videos/4k-video-clip-center`.

[35] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[36] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[37] K. Zhang, W. Zuo, S. Gu, and L. Zhang, “Learning deep cnn denoiser prior for image restoration,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017, pp. 2808–2817.

[38] X. Wang, A. Lucas, S. Lopez-Tapia, X. Wu, R. Molina, and A. K. Katsaggelos, “A composite discriminator for generative adversarial network based video super-resolution,” in *2019 27th European Signal Processing Conference (EUSIPCO)*, IEEE, 2019, pp. 1–5.

[39] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, “Video super-resolution with convolutional neural networks,” *IEEE Transactions on Computational Imaging*, vol. 2, no. 2, pp. 109–122, 2016.

[40] K. Zhang, W. Zuo, S. Gu, and L. Zhang, “Learning deep cnn denoiser prior for image restoration,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2808–2817.

[41] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia, “Video super-resolution via deep draft-ensemble learning,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 531–539.

[42] D. Li and Z. Wang, “Video super-resolution via motion compensation and deep residual learning,” *IEEE Transactions on Computational Imaging*, 2017.

[43] O. Makansi, E. Ilg, and T. Brox, “End-to-end learning of video super-resolution with motion compensation,” in *German Conference on Pattern Recognition*, Springer, 2017, pp. 203–214.

[44] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[45] A. Lucas, A. K. Katsaggelos, S. L. Tapia, and R. Molina, “Generative adversarial networks and perceptual losses for video super-resolution,” in *ICIP*, 2018.

[46] E. Pérez-Pellitero, M. S. M. Sajjadi, M. Hirsch, and B. Schölkopf, “Photorealistic video super resolution,” in *Workshop and Challenge on Perceptual Image Restoration and Manipulation (PIRM) at the 15th European Conference on Computer Vision (ECCV)*, 2018.

[47] X. Wang, A. Lucas, Lopez Tapia, X. Wu, R. Molina, and A. K. Katsaggelos, “Spatially adaptive losses for video-superresolution with gans,” in *IEEE Int. Conference on Acoustics, Speech, and Signal Processing, in press*, 2018.

[48] A. Lucas, S. Lopez Tapia, R. Molina, and A. K. Katsaggelos, “Generative adversarial networks and perceptual losses for video super-resolution,” *IEEE Trans. on Image Processing*, 2019.

[49] T. D. Nguyen, T. Le, H. Vu, and D. Phung, “Dual discriminator generative adversarial nets,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17, 2017, pp. 2667–2677, ISBN: 978-1-5108-6096-4.

[50] I. Durugkar, I. Gemp, and S. Mahadevan., “Generative multi-adversarial networks,” in *Int. Conference on Learning Representations*, 2017.

[51] C. Ledig, L. Theis, F. Huszar, *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 105–114, 2017.

[52] S. N. Efstratiadis and A. K. Katsaggelos, “Adaptive iterative image restoration with reduced computational load,” *Optical engineering*, vol. 29, no. 12, pp. 1458–1469, 1990.

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

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. arXiv: `1412.6980`.

[55] C. Liu and D. Sun, "A bayesian approach to adaptive video super resolution," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 2011, pp. 209–216.

[56] X. Wang, S. López-Tapia, and A. K. Katsaggelos, "Atmospheric turbulence correction via variational deep diffusion," in *2023 IEEE 6th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, IEEE, 2023, pp. 1–4.

[57] X. Zhu and P. Milanfar, "Removing atmospheric turbulence via space-invariant deconvolution," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 157–170, 2012.

[58] L. Wang, Y. Guo, L. Liu, Z. Lin, X. Deng, and W. An, "Deep video super-resolution using hr optical flow estimation," *IEEE Transactions on Image Processing*, vol. 29, pp. 4323–4336, 2020.

[59] Z. Mao, A. Jaiswal, Z. Wang, and S. H. Chan, "Single frame atmospheric turbulence mitigation: A benchmark study and a new physics-inspired transformer model," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIX*, Springer, 2022, pp. 430–446.

[60] S. N. Rai and C. Jawahar, "Removing atmospheric turbulence via deep adversarial learning," *IEEE Transactions on Image Processing*, vol. 31, pp. 2633–2646, 2022.

[61] Z. Mao, N. Chimitt, and S. H. Chan, "Accelerating atmospheric turbulence simulation via learned phase-to-space transform," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 759–14 768.

[62] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[63] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[64] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, "Image super-resolution via iterative refinement," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[65] N. G. Nair, K. Mei, and V. M. Patel, "At-ddpm: Restoring faces degraded by atmospheric turbulence using denoising diffusion probabilistic models," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 3434–3443.

[66] J. W. Soh and N. I. Cho, "Variational deep image restoration," *IEEE Transactions on Image Processing*, vol. 31, pp. 4363–4376, 2022.

[67] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, Springer, 2015, pp. 234–241.

[68] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[69] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[70] S. Nah, S. Baik, S. Hong, *et al.*, "Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[71] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[72] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[73] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[74] A. Mittal, R. Soundararajan, and A. C. Bovik, "Making a "completely blind" image quality analyzer," *IEEE Signal processing letters*, vol. 20, no. 3, pp. 209–212, 2012.

[75] X. Wang, S. López-Tapia, and A. K. Katsaggelos, “Real-world atmospheric turbulence correction via domain adaptation,” *arXiv preprint arXiv:2402.07371*, 2024.

[76] N. G. Nair, K. Mei, and V. M. Patel, “A comparison of different atmospheric turbulence simulation methods for image restoration,” in *2022 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2022, pp. 3386–3390.

[77] N. Chimitt and S. H. Chan, “Simulating anisoplanatic turbulence by sampling intermodal and spatially correlated zernike coefficients,” *Optical Engineering*, vol. 59, no. 8, pp. 083 101–083 101, 2020.

[78] J. E. Pearson, “Atmospheric turbulence compensation using coherent optical adaptive techniques,” *Applied optics*, vol. 15, no. 3, pp. 622–631, 1976.

[79] M. C. Roggemann and B. M. Welsh, *Imaging through turbulence*. CRC press, 2018.

[80] R. K. Tyson and B. W. Frazier, *Principles of adaptive optics*. CRC press, 2022.

[81] M. Aubailly, M. A. Vorontsov, G. W. Carhart, and M. T. Valley, “Automated video enhancement from a stream of atmospherically-distorted images: The lucky-region fusion approach,” in *Atmospheric Optics: Models, Measurements, and Target-in-the-Loop Propagation III*, SPIE, vol. 7463, 2009, pp. 104–113.

[82] N. Anantrasirichai, A. Achim, N. G. Kingsbury, and D. R. Bull, “Atmospheric turbulence mitigation using complex wavelet-based fusion,” *IEEE Transactions on Image Processing*, vol. 22, no. 6, pp. 2398–2408, 2013.

[83] N. Chimitt, Z. Mao, G. Hong, and S. H. Chan, “Rethinking atmospheric turbulence mitigation,” *arXiv preprint arXiv:1905.07498*, 2019.

[84] N. G. Nair and V. M. Patel, “Confidence guided network for atmospheric turbulence mitigation,” in *2021 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2021, pp. 1359–1363.

[85] X. Zhang, Z. Mao, N. Chimitt, and S. H. Chan, “Imaging through the atmosphere using turbulence mitigation transformer,” *IEEE Transactions on Computational Imaging*, 2024.

[86] K. Mei and V. M. Patel, “Ltt-gan: Looking through turbulence by inverting gans,” *IEEE Journal of Selected Topics in Signal Processing*, 2023.

[87] S. López-Tapia, X. Wang, and A. K. Katsaggelos, "Variational deep atmospheric turbulence correction for video," in *2023 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2023, pp. 3568–3572.

[88] A. Jaiswal, X. Zhang, S. H. Chan, and Z. Wang, "Physics-driven turbulence image restoration with stochastic refinement," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 12 170–12 181.

[89] H. Liu, B. Li, M. Lu, and Y. Wu, "Real-world image deblurring via unsupervised domain adaptation," in *International Symposium on Visual Computing*, Springer, 2023, pp. 148–159.

[90] W. Wang, H. Zhang, Z. Yuan, and C. Wang, "Unsupervised real-world super-resolution: A domain adaptation perspective," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 4318–4327.

[91] L. Wang and K.-J. Yoon, "Semi-supervised student-teacher learning for single image super-resolution," *Pattern Recognition*, vol. 121, p. 108 206, 2022.

[92] J. Zhou, V. Jampani, Z. Pi, Q. Liu, and M.-H. Yang, "Decoupled dynamic filter networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6647–6656.

[93] D. Cornett, J. Brogan, N. Barber, *et al.*, "Expanding accurate person recognition to new altitudes and ranges: The briar dataset," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 593–602.

[94] H. Yang, "Rethinking image and video restoration: An industrial perspective," *restoration*, vol. 3, p. 20,

[95] H. Talebi and P. Milanfar, "Learning to resize images for computer vision tasks. arxiv 2021," *arXiv preprint arXiv:2103.09950*,

[96] B. Zhang, Y. Guo, R. Yang, *et al.*, "Darkvision: A benchmark for low-light image/video perception," *arXiv preprint arXiv:2301.06269*, 2023.

[97] H. Zhu, W. Zheng, Z. Zheng, and R. Nevatia, "Sharc: Shape and appearance recognition for person identification in-the-wild," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 6290–6300.

[98] Y. Wu, X. Wang, and A. K. Katsaggelos, "Motion artifact reduction in abdominal mris using generative adversarial networks with perceptual similarity loss," in *17th International Symposium on Medical Information Processing and Analysis*, SPIE, vol. 12088, 2021, pp. 142–150.

[99] S. Namasivayam, D. R. Martin, and S. Saini, "Imaging of liver metastases: MRI," *Cancer Imaging*, vol. 7, no. 1, pp. 2–9, Feb. 1, 2007.

[100] M. Balafar, "Review of noise reducing algorithms for brain MRI images," *ijtpe*, vol. 4, pp. 54–59, Dec. 1, 2012.

[101] S. H. Ali, M. E. Modic, S. Y. Mahmoud, and S. E. Jones, "Reducing clinical MRI motion degradation using a prescan patient information pamphlet," *American Journal of Roentgenology*, vol. 200, no. 3, pp. 630–634, Mar. 2013.

[102] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[103] K. Zhang, W. Zuo, and L. Zhang, "Ffdnet: Toward a fast and flexible solution for cnn-based image denoising," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4608–4622, 2018.

[104] W. Shi, J. Caballero, F. Huszár, *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.

[105] M.-I. Georgescu, R. T. Ionescu, and N. Verga, "Convolutional neural networks with intermediate loss for 3d super-resolution of ct and mri scans," *IEEE Access*, vol. 8, pp. 49 112–49 124, 2020.

[106] B. A. Duffy, W. Zhang, H. Tang, *et al.*, "Retrospective correction of motion artifact affected structural mri images using deep learning of simulated motion," 2018.

[107] J. Liu, M. Kocak, M. Supanich, and J. Deng, "Motion artifacts reduction in brain MRI by means of a deep residual network with densely connected multi-resolution blocks (DRN-DCMB)," *Magnetic Resonance Imaging*, vol. 71, pp. 69–79, Sep. 1, 2020.

[108] Y. Chen, F. Shi, A. G. Christodoulou, Y. Xie, Z. Zhou, and D. Li, "Efficient and accurate mri super-resolution using a generative adversarial network and 3d multi-level densely con-

nected network," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2018, pp. 91–99.

[109] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*, Springer, 2016, pp. 694–711.

[110] Q. Yang, P. Yan, Y. Zhang, *et al.*, "Low-dose ct image denoising using a generative adversarial network with wasserstein distance and perceptual loss," *IEEE transactions on medical imaging*, vol. 37, no. 6, pp. 1348–1357, 2018.

[111] C. Ledig, L. Theis, F. Huszar, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[112] A. Lucas, S. López-Tapia, R. Molina, and A. K. Katsaggelos, "Generative adversarial networks and perceptual losses for video super-resolution," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3312–3327, 2019.

[113] D. Moratal, A. Vallés-Luch, L. Martí-Bonmatí, and M. Brummer, "K-space tutorial: An MRI educational tool for a better understanding of k-space," *Biomedical Imaging and Intervention Journal*, vol. 4, no. 1, e15, Jan. 1, 2008.

[114] M. Herbst, J. Maclaren, C. Lovell-Smith, *et al.*, "Reproduction of motion artifacts for performance analysis of prospective motion correction in MRI," *Magnetic Resonance in Medicine*, vol. 71, no. 1, pp. 182–190, 2014.

[115] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778, ISBN: 978-1-4673-8851-1.

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

[117] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: `1412.6980 [cs.LG]`.

[118] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*,

vol. 13, no. 4, pp. 600–612, Apr. 2004, Conference Name: IEEE Transactions on Image Processing.

[119] A. Horé and D. Ziou, “Image quality metrics: PSNR vs. SSIM,” in *2010 20th International Conference on Pattern Recognition*, ISSN: 1051-4651, Aug. 2010, pp. 2366–2369.

[120] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[121] N. Chinpanthana and Y. Liu, “Human activities of daily living recognition with graph convolutional network,” in *Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence*, 2020, pp. 305–310.

[122] Y. Li, Y. Li, and Y. Gu, “Channel-wise spatial attention with spatiotemporal heterogeneous framework for action recognition,” in *Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence*, 2020, pp. 334–338.

[123] H. A. Ullah, S. Letchmunan, M. S. Zia, U. M. Butt, and F. H. Hassan, “Analysis of deep neural networks for human activity recognition in videos–a systematic literature review,” *IEEE Access*, 2021.

[124] J.-C. Hou, A. McGonigal, F. Bartolomei, and M. Thonnat, “A self-supervised pre-training framework for vision-based seizure classification,” in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 1151–1155.

[125] A. Dubey, N. Lyons, A. Santra, and A. Pandey, “Xai-bayeshar: A novel framework for human activity recognition with integrated uncertainty and shapely values,” in *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2022, pp. 1281–1288.

[126] A. Ray, M. H. Kolekar, R. Balasubramanian, and A. Hafiane, “Transfer learning enhanced vision-based human activity recognition: A decade-long analysis,” *International Journal of Information Management Data Insights*, vol. 3, no. 1, p. 100 142, 2023.

[127] B. Natarajan, R. Elakkiya, R. Bhuvaneswari, K. Saleem, D. Chaudhary, and S. H. Samsudeen, “Creating alert messages based on wild animal activity detection using hybrid deep neural networks,” *IEEE Access*, 2023.

[128] Z. Xing, Q. Dai, H. Hu, J. Chen, Z. Wu, and Y.-G. Jiang, "Svformer: Semi-supervised video transformer for action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 18 816–18 826.

[129] A. Fuentes, S. Yoon, J. Park, and D. S. Park, "Deep learning-based hierarchical cattle behavior recognition with spatio-temporal information," *Computers and Electronics in Agriculture*, vol. 177, p. 105 627, 2020.

[130] M. Kopaczka, D. Tillmann, L. Ernst, J. Schock, R. Tolba, and D. Merhof, "Assessment of laboratory mouse activity in video recordings using deep learning methods," in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2019, pp. 3673–3676.

[131] F. Schindler and V. Steinhage, "Identification of animals and recognition of their actions in wildlife videos using deep learning techniques," *Ecological Informatics*, vol. 61, p. 101 215, 2021.

[132] H. Måløy, A. Aamodt, and E. Misimi, "A spatio-temporal recurrent network for salmon feeding action recognition from underwater videos in aquaculture," *Computers and Electronics in Agriculture*, vol. 167, p. 105 087, 2019.

[133] A. K. Sisodia *et al.*, "Impact of bird dropping deposition on solar photovoltaic module performance: A systematic study in western rajasthan," *Environmental Science and Pollution Research*, vol. 26, no. 30, pp. 31 119–31 132, 2019.

[134] E. Visser, V. Perold, S. Ralston-Paton, A. C. Cardenal, and P. G. Ryan, "Assessing the impacts of a utility-scale photovoltaic solar energy facility on birds in the northern cape, south africa," *Renewable energy*, vol. 133, pp. 1285–1294, 2019.

[135] R. J. Mustafa, M. R. Gomaa, M. Al-Dhaifallah, and H. Rezk, "Environmental impacts on the performance of solar photovoltaic systems," *Sustainability*, vol. 12, no. 2, p. 608, 2020.

[136] M. Lasseck, "Acoustic bird detection with deep convolutional neural networks," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, 2018, pp. 143–147.

[137] A. Thakur, A. Pankajakshan, and P. Rajan, "Learned aggregation in cnn: All-conv net for bird activity detection," in *Detection and Classification of Acoustic Scenes and Events 2018 Workshop*, 2018.

[138] M. Anderson, J. Kennedy, and N. Harte, "Low resource species agnostic bird activity detection," in *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, IEEE, 2021, pp. 34–39.

[139] S. Pouyanfar, Y. Tao, H. Tian, S.-C. Chen, and M.-L. Shyu, "Multimodal deep learning based on multiple correspondence analysis for disaster management," *World Wide Web*, vol. 22, no. 5, pp. 1893–1911, 2019.

[140] S. Pouyanfar, S.-C. Chen, and M.-L. Shyu, "Deep spatio-temporal representation learning for multi-class imbalanced data classification," in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, IEEE, 2018, pp. 386–393.

[141] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *In Workshop at International Conference on Learning Representations*, Citeseer, 2014.

[142] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[143] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," *Advances in neural information processing systems*, vol. 31, 2018.

[144] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[145] X. Wang and A. K. Katsaggelos, "Video-specific query-key attention modeling for weakly-supervised temporal action localization," *arXiv preprint arXiv:2305.04186*, 2023.

[146] F. Long, T. Yao, Z. Qiu, X. Tian, J. Luo, and T. Mei, "Gaussian temporal awareness networks for action localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[147] P. Chen, C. Gan, G. Shen, W. Huang, R. Zeng, and M. Tan, "Relation attention for temporal action localization," *IEEE Transactions on Multimedia*, vol. 22, no. 10, pp. 2723–2733, 2019.

[148] K. Xia, L. Wang, S. Zhou, N. Zheng, and W. Tang, "Learning to refactor action and co-occurrence features for temporal action localization," in *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2022, pp. 13 884–13 893.

[149] Z. Zhu, L. Wang, W. Tang, Z. Liu, N. Zheng, and G. Hua, "Learning disentangled classification and localization representations for temporal action localization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 2, 2022.

[150] K. Xia, L. Wang, Y. Shen, S. Zhou, G. Hua, and W. Tang, "Exploring action centers for temporal action localization," *IEEE Transactions on Multimedia*, 2023.

[151] J.-C. Feng, F.-T. Hong, and W.-S. Zheng, "Mist: Multiple instance self-training framework for video anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 14 009–14 018.

[152] F.-T. Hong, X. Huang, W.-H. Li, and W.-S. Zheng, "Mini-net: Multiple instance ranking network for video highlight detection," in *European Conference on Computer Vision*, Springer, 2020, pp. 345–360.

[153] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6479–6488.

[154] S. Paul, S. Roy, and A. K. Roy-Chowdhury, "W-talc: Weakly-supervised temporal activity localization and classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 563–579.

[155] M. Rashid, H. Kjellstrom, and Y. J. Lee, "Action graphs: Weakly-supervised action localization with graph convolution networks," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 615–624.

[156] P. Lee, Y. Uh, and H. Byun, "Background suppression network for weakly-supervised temporal action localization," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 11 320–11 327.

[157] F.-T. Hong, J.-C. Feng, D. Xu, Y. Shan, and W.-S. Zheng, "Cross-modal consensus network for weakly supervised temporal action localization," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 1591–1599.

[158] A. Islam, C. Long, and R. Radke, “A hybrid attention mechanism for weakly-supervised temporal action localization,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 1637–1645.

[159] J. Li, T. Yang, W. Ji, J. Wang, and L. Cheng, “Exploring denoised cross-video contrast for weakly-supervised temporal action localization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 914–19 924.

[160] Y. Zhao, H. Zhang, Z. Gao, W. Gao, M. Wang, and S. Chen, “A novel action saliency and context-aware network for weakly-supervised temporal action localization,” *IEEE Transactions on Multimedia*, 2023.

[161] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[162] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, “Vivit: A video vision transformer,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6836–6846.

[163] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.

[164] M. Nawhal and G. Mori, “Activity graph transformer for temporal action localization,” *arXiv preprint arXiv:2101.08540*, 2021.

[165] X. Liu, Q. Wang, Y. Hu, X. Tang, S. Bai, and X. Bai, “End-to-end temporal action detection with transformer,” *arXiv preprint arXiv:2106.10271*, 2021.

[166] M. Li, H. Wu, Y. Liu, H. Liu, C. Xu, and X. Li, “W-art: Action relation transformer for weakly-supervised temporal action localization,” in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 2195–2199.

[167] P. Nguyen, T. Liu, G. Prasad, and B. Han, “Weakly supervised action localization by sparse temporal pooling network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6752–6761.

[168] J. Carreira and A. Zisserman, “Quo vadis, action recognition? a new model and the kinetics dataset,” in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[169] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1933–1941.

[170] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar, "Rethinking the faster r-cnn architecture for temporal action localization," in *proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1130–1139.

[171] F. Long, T. Yao, Z. Qiu, X. Tian, J. Luo, and T. Mei, "Gaussian temporal awareness networks for action localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 344–353.

[172] C. Zhao, A. K. Thabet, and B. Ghanem, "Video self-stitching graph network for temporal action localization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 658–13 667.

[173] K. Xia, L. Wang, S. Zhou, N. Zheng, and W. Tang, "Learning to refactor action and co-occurrence features for temporal action localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 884–13 893.

[174] S. Narayan, H. Cholakkal, F. S. Khan, and L. Shao, "3c-net: Category count and center loss for weakly-supervised action localization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8679–8687.

[175] X.-Y. Zhang, H. Shi, C. Li, and P. Li, "Multi-instance multi-label action recognition and localization based on spatio-temporal pre-trimming for untrimmed videos," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 12 886–12 893.

[176] F. Ma, L. Zhu, Y. Yang, *et al.*, "Sf-net: Single-frame supervision for temporal action localization," in *European conference on computer vision*, Springer, 2020, pp. 420–437.

[177] L. Yang, J. Han, T. Zhao, T. Lin, D. Zhang, and J. Chen, "Background-click supervision for temporal action localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 9814–9829, 2021.

[178] P. X. Nguyen, D. Ramanan, and C. C. Fowlkes, "Weakly-supervised action localization with background modeling," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5502–5511.

[179] Z. Liu, L. Wang, Q. Zhang, *et al.*, "Acsnet: Action-context separation network for weakly supervised temporal action localization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 2233–2241.

[180] P. Lee, J. Wang, Y. Lu, and H. Byun, "Weakly-supervised temporal action localization by uncertainty modeling," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 1854–1862.

[181] L. Huang, L. Wang, and H. Li, "Foreground-action consistency network for weakly supervised temporal action localization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8002–8011.

[182] W. Luo, T. Zhang, W. Yang, *et al.*, "Action unit memory network for weakly supervised temporal action localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9969–9979.

[183] M. Zhao, Z. Hu, S. Li, S. Bi, and Z. Sun, "Mask attention-guided graph convolution layer for weakly supervised temporal action detection," *Multimedia Tools and Applications*, pp. 1–18, 2022.

[184] Z. Yang, J. Qin, and D. Huang, "Acgnet: Action complement graph network for weakly-supervised temporal action localization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 3090–3098.

[185] L. Huang, L. Wang, and H. Li, "Multi-modality self-distillation for weakly supervised temporal action localization," *IEEE Transactions on Image Processing*, vol. 31, pp. 1504–1519, 2022.

[186] J. Gao, M. Chen, and C. Xu, "Fine-grained temporal contrastive learning for weakly-supervised temporal action localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 999–20 009.

[187] B. He, X. Yang, L. Kang, Z. Cheng, X. Zhou, and A. Shrivastava, "Asm-loc: Action-aware segment modeling for weakly-supervised temporal action localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 925–13 935.

[188] L. Huang, L. Wang, and H. Li, "Weakly supervised temporal action localization via representative snippet knowledge propagation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3272–3281.

[189] M. Chen, J. Gao, S. Yang, and C. Xu, "Dual-evidential learning for weakly-supervised temporal action localization," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IV*, Springer, 2022, pp. 192–208.

[190] C. Ju, P. Zhao, S. Chen, *et al.*, "Adaptive mutual supervision for weakly-supervised temporal action localization," *IEEE Transactions on Multimedia*, 2022.

[191] H.-f. Xia and Y.-z. Zhan, "Deep cascaded action attention network for weakly-supervised temporal action localization," *Multimedia Tools and Applications*, pp. 1–19, 2023.

[192] M. Moniruzzaman and Z. Yin, "Feature weakening, contextualization, and discrimination for weakly supervised temporal action localization," *IEEE Transactions on Multimedia*, 2023.

[193] Z. Shou, H. Gao, L. Zhang, K. Miyazawa, and S.-F. Chang, "Autoloc: Weakly-supervised temporal action localization in untrimmed videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 154–171.

[194] Y.-G. Jiang, J. Liu, A. Roshan Zamir, *et al.*, *THUMOS challenge: Action recognition with a large number of classes*, `http://crcv.ucf.edu/THUMOS14/`, 2014.

[195] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015, pp. 961–970.

[196] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin, "Temporal action detection with structured segment networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2914–2923.

[197] P. Lee and H. Byun, "Learning action completeness from points for weakly-supervised temporal action localization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 648–13 657.

[198] B. Shi, Q. Dai, Y. Mu, and J. Wang, "Weakly-supervised action localization by generative attention modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1009–1019.

[199] S. Narayan, H. Cholakkal, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, "D2-net: Weakly-supervised action localization via discriminative embeddings and denoised acti-

vations," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 608–13 617.

[200] R. Su, D. Xu, L. Sheng, and W. Ouyang, "Pcg-tal: Progressive cross-granularity cooperation for temporal action localization," *IEEE Transactions on Image Processing*, vol. 30, pp. 2103–2113, 2020.

[201] Y. Zhai, L. Wang, W. Tang, Q. Zhang, J. Yuan, and G. Hua, "Two-stream consensus network for weakly-supervised temporal action localization," in *European conference on computer vision*, Springer, 2020, pp. 37–54.

[202] Y. Liu, J. Chen, Z. Chen, B. Deng, J. Huang, and H. Zhang, "The blessings of unlabeled background in untrimmed videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6176–6185.

[203] W. Yang, T. Zhang, X. Yu, T. Qi, Y. Zhang, and F. Wu, "Uncertainty guided collaborative training for weakly supervised temporal action detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 53–63.

[204] W. Kay, J. Carreira, K. Simonyan, *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[205] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

DEEP LEARNING APPLIED TO IMAGE AND VIDEO PROCESSING

Approved by:

Date Approved: February 13, 2024