Video Deinterlacing and Demosaicing by Deep Learning

by

Ronglei Ji

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Electrical and Electronics Engineering

KOÇ ÜNİVERSİTESİ

January 15, 2024

Video Deinterlacing and Demosaicing
by Deep Learning

Koç University
Graduate School of Sciences and Engineering
This is to certify that I have examined this copy of a doctoral dissertation by

Ronglei Ji

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____
Prof. A. Murat Tekalp (Advisor)

_____
Assoc. Prof. Aykut Erdem

_____
Assist. Prof. Zafer Doğan

_____
Prof. Hasan F. Ateş

_____
Prof. Erkut Erdem

Date: _____

To those who are being rooted and grounded in love

ABSTRACT

Video Deinterlacing and Demosaicing
by Deep Learning
Ronglei Ji
Doctor of Philosophy in Electrical and Electronics Engineering
January 15, 2024

Deinterlacing and demosaicing are commonly used techniques in the image processing pipeline for consumer video. Despite the fact that real-world video deinterlacing and demosaicing are well-suited to supervised learning from synthetically degraded data because both degradation models are known and fixed, learned video deinterlacing and demosaicing have received much less attention compared to denoising and superresolution tasks. This thesis progressively explores feature alignment, integration and reconstruction stages for both tasks tailored to their known and fixed degradation subsampling patterns.

We begin by presenting our initial work of a novel multi-field deinterlacing architecture that aligns features from adjacent fields to a reference field (to be deinterlaced) by designing novel deformable residual convolution blocks with two variants of different scales. To the best of our knowledge, this work is the first to propose fusion of multi-field features that are aligned via deformable convolutions for deinterlacing.

Next, based on our initial work, we propose a novel multi-field full frame-rate deinterlacing network, which adapts the state of-the-art superresolution approaches to the deinterlacing task. This model incorporates self attention mechanism with deformable convolution residual blocks to align features and additively integrate aligned features for reconstruction. In order to reconstruct odd and even fields directionally, separate reconstruction modules are utilized according to the parity of each reference. Our extensive experimental results demonstrate that the proposed method provides state-of-the-art deinterlacing results in terms of both numerical and perceptual performance.

Upon all these previous work, we propose a new multi-picture architecture for both video deinterlacing or demosaicing by aligning multiple supporting pictures

with missing data to a reference picture to be reconstructed, benefiting from both local and global spatio-temporal correlations in the feature space using modified deformable convolution blocks and a novel residual efficient top-k self-attention (kSA) block, respectively. Separate reconstruction blocks are used to estimate different types of missing data. Our extensive experimental results demonstrate that the proposed novel architecture provides superior results that significantly exceed the state-of-the-art for both tasks in terms of PSNR, SSIM, and perceptual quality. Ablation studies are provided to justify and show the benefit of each novel modification made to the deformable convolution and residual efficient kSA blocks.

ÖZETÇE

Derin Öğrenme ile Video Binişimsizleştirme ve Demozaikleme
Doktora Tez Başlığı
Ronglei Ji
Elektrik ve Elektronik Mühendisliği, Doktora
15 Ocak 2024

Binişimsizleştirme ve Demozaikleme, tüketici videosuna yönelik görüntü işleme hattında yaygın olarak kullanılan tekniklerdir. Her iki bozulma modeli de bilindiği ve sabitlendiğinden, gerçek dünyadaki video taramasızlaştırma ve ayrıştırma, sentetik olarak bozulmuş verilerden denetimli öğrenmeye çok uygun olmasına rağmen, öğrenilmiş video Binişimsizleştirme ve Demozaikleme, gürültü giderme ve süper çözünürlük görevleriyle karşılaştırıldığında çok daha az ilgi görmüştür. Bu tez, bilinen ve sabit bozulma alt örnekleme modellerine göre uyarlanmış her iki görev için özellik hizalama, entegrasyon ve yeniden yapılandırma aşamalarını aşamalı olarak araştırmaktadır.

Farklı ölçeklerde iki varyantla yeni deforme olabilen artık evrişim blokları tasarlayarak, bitişik alanlardaki özellikleri bir referans alanına (titreşimsizleştirilecek) hizalayan yeni bir çok alanlı Binişimsizleştirme mimarisine ilişkin ilk çalışmamızı sunarak başlıyoruz. Bildiğimiz kadarıyla bu çalışma, taramasızlaştırma için deforme edilebilir evrişimler aracılığıyla hizalanan çok alanlı özelliklerin füzyonunu öneren ilk çalışmadır.

Daha sonra, ilk çalışmamıza dayanarak, son teknoloji ürünü süper çözünürlük yaklaşımlarını Binişimsizleştirme görevine uyarlayan, yeni, çok alanlı, tam kare hızında Binişimsizleştirme ağı öneriyoruz. Bu model, özellikleri hizalamak ve yeniden yapılandırma için hizalanmış özellikleri ilave olarak entegre etmek için deforme olabilen evrişim artık bloklarıyla kişisel dikkat mekanizmasını birleştirir. Tek ve çift alanları yönlü olarak yeniden oluşturmak için her referansın paritesine göre ayrı yeniden yapılandırma modülleri kullanılır. Kapsamlı deneysel sonuçlarımız, önerilen yöntemin hem sayısal hem de algısal performans açısından son teknoloji ürünü Binişimsizleştirme sonuçları sağladığını göstermektedir.

Önceki tüm çalışmaların ardından, eksik veri içeren birden fazla destekleyici resmi, yeniden yapılandırılacak bir referans resmine hizalayarak, özellik alanındaki hem yerel hem de küresel uzay-zamansal korelasyonlardan yararlanarak, hem video Binişimsizleştirme hem de Demozaikleme için yeni bir çoklu resim mimarisi öneriyoruz. sırasıyla değiştirilmiş deforme olabilir evrişim blokları ve yeni bir artık verimli üst k kişisel dikkat (kSA) bloğu. Farklı türdeki eksik verileri tahmin etmek için ayrı yeniden yapılandırma blokları kullanılır. Kapsamlı deneysel sonuçlarımız, önerilen yeni mimarinin, PSNR, SSIM ve algısal kalite açısından her iki görev için de en son teknolojiyi önemli ölçüde aşan üstün sonuçlar sağladığını göstermektedir. Deforme olabilen evrişim ve artık verimli kSA bloklarında yapılan her yeni değişikliğin faydasını doğrulamak ve göstermek için ablasyon çalışmaları sağlanmıştır.

ACKNOWLEDGMENTS

First and foremost, I would like to express my highest respect and sincere gratitude to my advisor Prof. Murat Tekalp for his invaluable inspiration, dependable guidance and endless patience out of his professionalism and charisma, which tremendously already helped and will bring benefits to all my life and career. I also would like to thank Assoc. Prof. Aykut Erdem and Assist. Prof. Zafer Doğan for their encouragement and advice during my thesis research. I am grateful to Prof. Hasan F. Ateş and Prof. Erkut Erdem for their interests and time to serve on my thesis jury. Meanwhile, I greatly appreciate all your critical reading of this thesis.

Next, I would like to thank Prof. Engin Erzin, who introduced me into this program and then so many wonderful things could happen. Also I want to give special and profound thanks to Mr. Ufuk Yılmaz, who taught me to use the high performance clusters and helped me very timely to solve countless programming problems concerning the computing resources even in his off work time. I am so grateful as well to Nasrin Rahimi, Melissa Abache, Aslı Edwards and many many others who are in our lab, in administrative offices, in English language center, in TAships and even in dining hall for their friendship, their greetings, their accompanies, their comforting words, their encouraging smiles, their snacks and water, their leaving the lab door open for me, their inquisitive chats about my country and their assistance in using printers. It is my great honor to meet all of you in my life and to have you all in my experiences.

My everlasting love to Peijuan, Zinuo, Ziyi, Rongjun and my parents, who are the most important part of my life. I thank you all for your unconditional love and support. Finally, I acknowledge the financial support from Fung Foundation for my doctoral studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AI	Artificial Intelligence
CFA	Color Filter Array
CNN	Convolutional Neural Networks
DDPMs	Denoising Diffusion Probabilistic Models
DfRes	Deformable Convolution Residual block
Df	Deformable Convolution layer
DfConv	Deformable Convolution block
DRL	Deep Reinforcement Learning
DNN	Deep Neural Networks
GAN	Generative Adversarial Networks
kSA	top-k Self-Attention
LSTM	Long Short Term Memory
MAP	Maximum A Posterior
ML	Machine Learning
MSE	Minimum Mean Squared Error
PSNR	Peak Signal-to-Noise Ratio
RNN	Recurrent Neural Networks
SA	Self-Attention
SDES	Stochastic Differential Equations
SGD	Stochastic Gradient Descent
SGMs	Score-based Generative Models
SSIM	Structural Similarity index
VSR	Video Super-Resolution

Chapter 1

INTRODUCTION

1.1 Overview of Deep Learning

Deep learning, with its unanticipated success due to dramatic growth of computing power and availability of tremendous amounts of training data over the past decade, has drastically boosted areas such as natural language processing, speech recognition, computer vision, image/video processing including image/video restoration and super-resolution, and many other fields in both industry and academia.

Deep learning, inspired by initially mimicking how human brain behaves, works through artificial neural network which generally includes an input layer, an output layer and some hidden layers between input and output layers, where there are neuron nodes in each layer non-linearly connecting to neuron nodes in the following layer by mapping weights and bias plus nonlinear activation functions.

Compared to shallow learning models which process simpler data patterns with limited few layers or interpretable mathematical models, deep learning, usually having tens or hundreds of hidden layers, can easily handle much more complicated problems and higher level applications in real world [Dong et al., 2021]. Moreover, deep learning saves a huge portion of human efforts [Najafabadi et al., 2015] by enabling complex features that represent inputs to be extracted from raw data automatically, where hierarchical features can be constructed through layered structure.

Given a deep neural network, the initialized mapping parameters, that is, weights and bias, have to be learned to the extent of being optimal, by being updated according to a certain optimizing strategy. This learning process is usually carried out

by optimization algorithms, such as Stochastic Gradient Descent (SGD) [Sutskever et al., 2013], Adam [Kingma and Ba, 2014], and RMSprop [Hinton, 2012], which try to find a minimum point of a specific objective function.

Back-propagation is a crucial and common way to provide feedback of the model output towards the parameters in preceding layers in a deep learning model. Specifically, by calculating gradients of the particular objective error function with regard to the model parameters, the weights and biases are renewed iteratively, and the errors between the estimated output and the expectation are minimized gradually.

1.1.1 Taxonomy of Deep Learning Approaches

Deep learning is generally regarded as a sub-field of Machine Learning (ML), which is also a subset of Artificial Intelligence (AI) [Alom et al., 2019, Janiesch et al., 2021].

Upon the availability of labeled data, [Alom et al., 2019] categorized deep learning approaches into three groups: Supervised learning group including Deep Neural Networks (DNN), Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN) and so on, which utilizes labeled data as ideal expectation to adjust model parameters in order for the estimated output to be as close to expected results as possible, Semi-supervised learning group which may use Deep Reinforcement Learning (DRL) and Generative Adversarial Networks (GAN) on partially labeled data for some cases, and Unsupervised group learning internal representation within data to cluster or reduce dimension without labeled data by applying Restricted Boltzmann Machine (RBM), Auto-Encoder (AE) or generative models, such as GAN, as well. Obviously, this taxonomy is not completely exclusive, allowing variation in practical application. For instance, RNN can be classified into every other group based on the usage scenarios.

[Sarker, 2021] categorized deep learning approaches into discriminative group that uses supervised learning methods based on condition, generative group which focuses on high-order correlation and statistical distribution of data, such as GAN and AE, and a hybrid learning group which may benefit from both discriminative

and generative methods, such as CNN+LSTM and GAN+CNN.

For the reason that deep learning has been so much developed in miscellaneous areas for so many years, there cannot be a boundary-clear and all-inclusive taxonomy to cover all approaches. This thesis simply provides a general and common classification reference in order to show where our proposed model architecture stands.

In next subsection, we will look into some representative deep learning networks which are popular and widely employed, or potentially beneficial to our thesis tasks for a better understanding of our proposed architecture.

1.1.2 Convolutional Neural Network (CNN)

CNN can be traced back to as early as 1980s. [Fukushima, 1980] introduced neocognitron, which has a similar architecture to CNN but did not employ a supervised learning algorithm, recognizing input patterns based on geometrical similarity of their shapes without being affected by their positions or even by a small variation in shape or in size. [Waibel et al., 1989] presented a 1D CNN named time-delay neural network, which is independent of position in time and is used for phoneme recognition with error back-propagation algorithm for training. Both of the above two architectures can share weights along network positions or a temporal dimension, benefiting from their shift invariance property [Goodfellow et al., 2016].

[LeCun et al., 1989] constructed a CNN architecture in conjunction with stochastic gradient based back-propagation learning algorithm to recognize handwritten zip code, which contains convolution layers for automatic feature extraction with implicit pooling layers for subsampling, and fully connected layers to compile features processed by previous layers for the final output, and is the primary architecture of LeNet-5 [LeCun et al., 1998]. The LeNet-5 is a conclusion of many years of research on CNN formally defining the fundamental components of CNN architecture, and is one of the prototypes and the pioneering architectures for future CNN models, playing a significant role in advancing deep learning.

Enjoying the properties of natural signal that the local features are highly correlated and the local statistics are invariant to location, CNN utilizes local sparse

interactions to detect local patterns and shared weights to learn only one set of parameters for every location of the input, both reducing required memory for the model and improving its statistical efficiency [LeCun et al., 2015, Goodfellow et al., 2016].

Main Components of CNN:

(1) Convolutional Layer

The key mathematical principle behind a convolutional layer is the convolution operation, which involves the interaction of two signals to generate a third signal. A general form of convolution operation is given by:

$$y(t) = \int_{a=-\infty}^{+\infty} x(a)w(t-a)\,da, \tag{1.1}$$

where x(a) is one signal with variable a, and w(a) is another signal with the same variable a, y(t) is the output with regard to variable t. Here, w(t-a) indicates that signal w(a) is flipped left and right first, then shifted rightward with a distance t.

Hence, the output y(t) is the integral of x(a) with flipped and shifted w(a). If w(a) is a filtering kernel as a window, then the convolution of x(a) and w(a) can be seen as a flipped window w(a) sliding from left to right to integrate with signal x(a) at position t for an output y(t). A discrete form of convolution can be written as:

$$y(t) = \sum_{a=-\infty}^{+\infty} x(a)w(t-a). \tag{1.2}$$

The convolution is usually represented with an asterisk:

$$y(t) = (x * w)(t). \tag{1.3}$$

In a similar way, a two dimensional convolution for image or video with position variables i and j can be formulated as:

$$y(m,n) = (x * w)(m,n) = \sum_i \sum_j x(i,j)w(m-i,n-j), \tag{1.4}$$

which is equivalent to

$$y(m,n) = (w * x)(m,n) = \sum_i \sum_j w(i,j)x(m-i,n-j), \tag{1.5}$$

according to commutative property of convolution. In many practical implementation libraries, cross-correlation is optionally used by removing flipping operation in convolution [Goodfellow et al., 2016]:

$$y(m,n) = (w * x)(m,n) = \sum_i \sum_j w(i,j)x(m+i, n+j). \tag{1.6}$$

In Convolutional Neural Network, a convolutional layer is expressed as:

$$x_j^l = f(\sum_{i \in M_j^l} x_i^{l-1} * w_{ij}^l + b_j^l), \tag{1.7}$$

where $*$ denotes convolution operation, f is non-linear activation function, x_j^l is the j^{th} feature map in the l^{th} feature map layer, x_i^{l-1} is the i^{th} feature map in the $(l-1)^{th}$ feature map layer, w_{ij}^l is the kernel to weight input x_i^{l-1} for generating x_j^l, M_j^l is a mapping scheme of feature maps from previous layer to current layer, b_j^l is a bias or offset for the j^{th} feature map in the l^{th} feature map layer.

Multiple layers require that non-linear function f be utilized to break up the linearity of the CNN architecture, which can not be done through an identity activation function that would make the whole network equivalent to one layer model limiting the network to do complex tasks. Moreover, gradient based optimization algorithms desires that the activation function is continuously differentiable, so that the back-propagation can be executed in order to learn parameters. The popular nonlinear activation functions covers sigmoid function [Han and Moraga, 1995], Rectified Linear Unit (ReLU) [Nair and Hinton, 2010], Leaky Rectified Linear Unit (Leaky ReLU) [Maas et al., 2013] and many others.

Assuming an input with shape of [H, W, C] is convolved with N kernels having shape of [K, K, C], and the output should be of shape [(H-K+2P)/S+1, (W-K+2P)/S+1, N], where C is input depth or channel or number of input feature map, for gray-scale image input C is 1, for RGB color image C input is 3 and for hidden layer input C may be hundreds, H and W are input height and width respectively, K is kernel size and usually CNN uses square kernels, P is padding numbers, S is sliding stride for filter and N is numbers of filters.

There are total N kernels in one convolutional layer to produce N feature maps. Each of the N kernels has the same number C with the depth or channel of the input, so that each kernel can slide over the entire input through all channels in depth for convolution operation. Each kernel is assigned with a bias. Therefore, the total learnable parameters in one convolutional layer is N×(K×K×C+1).

The kernel size K is smaller than the height and width of the input, which causes local connections instead of full connections. Besides, the significant idea of weight/parameter sharing enables each kernel, with the same learned wights and bias, to interact/convolve with different patches across different locations in an input. Both of the above advantages decrease the required memory to store parameters and enhance its statistical efficiency [Goodfellow et al., 2016, Tekalp, 2022].

A special case is K=1, that is 1×1 convolution [Lin et al., 2013], which can be used to reduce dimension [He et al., 2016, Szegedy et al., 2015].

(2) Subsampling Layer

Following the nonlinear activation added to the output of the convolution operation, a subsampling layer is presented to make the representation/features approximately invariant/numb to small translations of the input [Goodfellow et al., 2016], that is, even if the input is translated a bit, the subsampled outputs almost do not change. The reason is that once the features are properly extracted, the exact position will not be so cared about. Subsampling layer reduces the amount of parameters by shrinking spatial sizes of the feature maps and accelerates learning process, hence to alleviate overfitting.

Subsampling is usually performed by pooling, which summarizes the activated features over a whole patch. The basic and common-used pooling operations include Max-pooling [Zhou and Chellappa, 1988], average pooling and so on. When a $p \times p$ pooling window slides across a feature map with a specified stride, over each $p \times p$ area it covers, this pooling window selects the max value or average all the values over this area to form a new feature map, depending on whether it is a Max-pooling or an average pooling operation. Therefore, no learnable parameters are required

for pooling operation.

(3) Fully Connected Layer

Fully connected layer, different from convolutional layer which locally connects the previous feature map through smaller kernels, densely connects all units in entire feature map. Assuming an input with shape of [H, W, C] is fully connected to next feature map of shape [10×C, 1, 1], then the required number of parameters is H×W×C×10×C plus 10×C biases. In fully connected layer, every output unit is connected to every input unit through a set of weight and bias. Thus, weight sharing is not need any more.

Typical CNN Architectures:

(1) LeNet

As one of the most important prototypes of CNN, LeNet laid a good foundation for further development of CNN architecture. Especially the net structure of LeNet-5 [LeCun et al., 1998], inspires the proposal of many popular CNN architectures.

LeNet-5 is actually not very deep, containing two convolutional layers with 6 and 16 kernels of size 5×5 in the first and third layer respectively, two subsampling layers with 6 and 16 pooling windows of size 2×2 in the second and fourth layer respectively, one more convolutional layer in the fifth layer with 120 kernels of size 5×5, in the last layer 84 units fully connected to the fifth layer and fully connected to 10 output units using Gaussian connection. LeNet-5 employs a squashing scaled hyperbolic tangent function as activation and Minimum Mean Squared Error (MSE) to push down penalty of correct class with Maximum A Posteriori (MAP) to pull up penalty of incorrect class [LeCun et al., 1998] as loss function.

Due to the limitation of computing power, LeNet-5 makes use of a non-complete connection scheme allowing partial kernels in the third layer to only connect partial input feature maps.

(2) AlexNet

Although some contemporary or earlier works have shown considerably improved performance in GPU driven neural networks [Oh and Jung, 2004, Chellapilla et al., 2006, Ciresan et al., 2011], AlexNet [Krizhevsky et al., 2012] stands out due to its excellent success by outperforming the runner up with a large margin and become one of the most significant works for computer vision community, having stimulated many more researchers to develop GPU based CNN architectures.

Different from many other CNN architectures which are quite similar to the pioneering shallow work [LeCun et al., 1998] having a convolutional layer immediately followed by a pooling layer, AlexNet is much deeper with 5 convolutional layers, 3 overlapping Max-pooling layers and 3 fully connected layers, and is also larger with 96 kernels of size 11×11 in the first layer. Specially, the last three convolutional layers are piled upon top of the other without pooling layers. AlexNet exploits Rectified Linear Units (ReLU) [Nair and Hinton, 2010] instead of tanh, achieving several times faster in training speed. Furthermore, data augmentation such as image translations, horizontal reflections and changing intensities of RGB channels are used, as well as dropout [Hinton et al., 2012], to decrease overfitting.

(3) Post-AlexNet

[Zeiler and Fergus, 2014] put forward an improved CNN architecture based on AlexNet, by replacing the 11×11 kernels with 7×7 kernels in the first layer and decreasing the stride of convolution to 2 from 4. To yield this enhanced architecture, the authors proposed a visualization technique using multi-layered Deconvolutional Network (deconvnet) [Zeiler et al., 2011], by revealing the input stimulus that excites each feature map at any layer and the evolution of features in training, to provide insight into the internal operation and behavior of CNN architectures for better understanding of these models.

As many researches have been done to make the AlexNet better, [Simonyan and Zisserman, 2014] investigated the CNN architecture from the significant facet of depth. The authors built their VGG nets, ranging from 11 weight layers (8 convo-

lutional layers plus 3 fully connected layers) to 19 weight layers (16 convolutional layers plus 3 fully connected layers), with kernel numbers increasing from 64 to 512 by a factor of 2, consisting of convolutional layers with kernel size 3×3, stride 1 pixel and padding 1 pixel, and of Max-pooling layers with window size 2×2 and stride 2 pixels. Different from [Krizhevsky et al., 2012, Zeiler and Fergus, 2014], VGG nets only use 3×3 kernel and state that a stack of two 3×3 convolutional layers has an equivalent receptive field of 5×5, and of three 3×3 convolutional layers has an equivalent receptive field of 7×7. Thus, small kernel size allows more non-linear activation functions to make more discriminative decisions and as well reduces amount of parameters.

[Szegedy et al., 2015] constructed a CNN architecture named GoogLeNet having 22 layers in depth and containing variable receptive fields 1×1, 3×3 and 5×5 kernels built on top of each other sequentially or in parallel. Especially the 1×1 convolutional kernels and pooling windows are put before the expensive 3×3 kernels, 5×5 kernels and fully connected layers for dimensionality reduction. Thus, GoogLeNet used about one tenth of parameters compared to AlexNet.

As going deeper, [He et al., 2016] introduced Residual Network (ResNet) to address the degradation of training accuracy (specifically, gradients vanishing), which reaches 152 layers in depth, yet with lower complexity, in contrast with VGG net. A simple ResNet block may be formulated as $\mathcal{F}(\mathrm{x}) + \mathrm{x}$. $\mathcal{F}(\mathrm{x})$ denotes a mapping through two, three or more stacked nonlinear layers such as two convolutional layers with a ReLU activation function between these two layers. x is input having the same shapes/dimensions with $\mathcal{F}(\mathrm{x})$ in order to perform an element-wise addition. In case that the channel numbers are changed, a linear projecting matrix W_p is required to adjust the channel of x to match $\mathcal{F}(\mathrm{x})$. Then this formula becomes $\mathcal{F}(\mathrm{x}) + W_p\mathrm{x}$.

Receptive Field

Given an input and a kernel in a convolutional layer, each convolution calculation/operation is executed over a locally restricted input area covered by this kernel,

that is, only partial locations in input can be weighted by the convolutional kernel. This locally restricted area is the receptive field for this kernel. For fully connected layers, the receptive field is the entire input. In video processing, in order to capture large motions, a sufficient receptive field is essential and required to cover adequate input locations.

In above mentioned deep neural networks, a stack of small kernels are often utilized to provide effective receptive field of a larger kernel. In addition, some improved CNNs are also proposed to enlarge receptive field to involve larger context.

Dilated convolution [Yu and Koltun, 2015], also known as Atrous convolution [Chen et al., 2017a, Chen et al., 2017b], enables the receptive field of convolutional layer to be expanded, without increasing parameters or computation, by multiplying a rate r to stride in standard convolution as formulated in Equation 1.5. Therefore, the dilated convolution can be expressed as:

$$y(m,n) = (w * x)(m,n) = \sum_i \sum_j w(i,j)x(m - r \cdot i, n - r \cdot j), \tag{1.8}$$

when $r = 1$, this formula is exactly the same with Equation 1.5 for standard convolution.

Although dilated convolution successfully enlarges the receptive field, its sampling process is still limited at fixed locations once r is provided. To tackle this issue, [Dai et al., 2017, Zhu et al., 2019] put forward deformable convolution, which allows free form deformation of the sampling grid by adding learnable offsets (Δi, Δj) to the sampling locations. Based on standard convolution in Equation 1.5, the deformable convolution is formulated as:

$$y(m,n) = (w * x)(m,n) = \sum_i \sum_j w(i,j)x(m - i + \Delta i, n - j + \Delta j), \tag{1.9}$$

Since its inception, deformable convolution has been applied to various fields. Profiting from its potential capability of capturing large motions, our work constituting this thesis is mainly based on deformable convolution.

1.1.3 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) have been demonstrated to be dominant in processing series of sequential data, such as speech, language and video frame sequences. That historic information of all the past elements in each sequence is stored in the the hidden state vector enables RNN models to learn long-term dependencies for sequential inputs.

The classic RNN can be mathematically formulated as:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b_h) \tag{1.10}$$

$$y_t = \sigma(W_y h_t + b_y), \tag{1.11}$$

where x_t, y_t and h_t denote the input, the output and hidden state containing the recurrent information at time t, separately, W_x W_y and W_h, b_x and b_y are the corresponding weights and biases for mapping to the hidden state and output, respectively, and σ is activation function.

Unfolded RNNs in time can be regarded as very deep feedforward networks in which the same weights are shared with all the layers [LeCun et al., 2015]. Consequently, when trained with back-propagation through many time steps, early classic RNNs [Jordan, 1986, Elman, 1990] face the problem of vanishing or exploding gradients as CNN does. Since gradient decent becomes progressively incapacitated as the temporal extent of dependencies stretches [Bengio et al., 1994], it is difficult to capture long-term dependencies.

Many different RNN architectures and training methods [Williams and Peng, 1990, Hochreiter and Schmidhuber, 1997, Hochreiter et al., 2001, Pascanu et al., 2013] have been proposed over the past decades to promote the practical usage of RNN. One of the most influential RNN architecture is initially presented in [Hochreiter and Schmidhuber, 1997], called Long Short Term Memory (LSTM) network, aiming at resolving the gradient vanishing problem by introducing cells and input and output gates into the vanilla RNN architecture. In the same year, another important RNN architecture Bidirectional Recurrent Neural Networks (BRNN) [Schuster and Paliwal, 1997], enabling the network to benefit from both forward and back-

ward time directions simultaneously using respective hidden layers. [Graves and Schmidhuber, 2005] combines BRNN and LSTM to present a Bidirectional LSTM network for phoneme classification. This is the first time the two basic RNN architectures are combined, which effectively outperforms unidirectional designs.

Long Short Term Memory (LSTM) network

The first main LSTM architecture is proposed in [Hochreiter and Schmidhuber, 1997] after several years cultivation [Hochreiter and Schmidhuber, 1996]. This evolved architecture enlarged the remembering capacity of RNN architecture to handle long-time-gap events by reinforcing the memory units to convey past information through time. Since then, many advanced LSTM architectures [Gers et al., 2000, Gers et al., 2002, Graves and Schmidhuber, 2005, Cho et al., 2014, Shi et al., 2015, Baytas et al., 2017] are developed based on this initial significant unit.

A mature and fundamental LSTM architecture usually denotes the LSTM with a forget gate, which can be mathematically represented as:

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \tag{1.12}$$

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \tag{1.13}$$

$$\tilde{c}_t = \tanh(W_{\tilde{c}x}x_t + W_{\tilde{c}h}h_{t-1} + b_{\tilde{c}}) \tag{1.14}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{1.15}$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \tag{1.16}$$

$$h_t = o_t \cdot \tanh(c_t), \tag{1.17}$$

where f_t, i_t and o_t are forget gate, input gate and output gate at time step t, respectively, c_t and h_t denotes memory cell state and hidden state as the output at time step t, separately, W and b are corresponding weights and bias, σ and tanh are activation functions, and '$\cdot$' denotes element-wise multiplication.

Forget gate f_t was later introduced to the original LSTM design to reset the memory state c_t once the memorized information is out of date and irrelevant [Gers et al., 2000]. When $f_t = 1$, it forgets nothing and is the same with the standard

LSTM. When $f_t = 0$, it discards all information from previous memory cell. Therefor, forget gate determines whether or not the past memorized information in cell state c_t will affect current output. In addition, [Jozefowicz et al., 2015] found that adding a positive bias to forget gate significantly improves its performance. Experiments conducted by [Greff et al., 2016] show that forget gate with the output gate plays an critical role in maintaining the performance of LSTM block.

Apart from the forget gate, [Gers and Schmidhuber, 2000] introduced peephole to directly connect memory cell to gates in order to learn precise timing. By adding $P_f \cdot c_{t-1}$, $P_i \cdot c_{t-1}$ and $P_o \cdot c_t$ to the corresponding bias b_f, b_i and b_o, respectively in Equation 1.12, 1.13 and 1.16, LSTM enables the gates to examine the present cell state, and learn exact and robust timing approaches without teacher forcing [Gers et al., 2002].

Along with the advancement of LSTM learning capacity, the parameters and computing burden are also increasing. Hence, a much simpler LSTM design, called gated recurrent unit (GRU) [Cho et al., 2014] was proposed by incorporating forget gate and input gate to be a simpler update gate, and merging memory cell state and hidden state accompanied with other modifications. The mathematical expression of GRU is:

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_{t-1} + b_r) \tag{1.18}$$

$$z_t = \sigma(W_{zx}x_t + W_{zh}h_{t-1} + b_z) \tag{1.19}$$

$$\tilde{h}_t = \tanh(W_{\tilde{h}x}x_t + W_{\tilde{h}h}(r_t \cdot h_{t-1}) + b_{\tilde{h}}) \tag{1.20}$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t, \tag{1.21}$$

where r_t is reset gate used to allow hidden states to drop irrelevant information for the future, z_t is update gate used to store past information to perform long-term interactions.

The rationale of omitting output gate in GRU was demonstrated in [Greff et al., 2016]. The forget gate and the output gate are the most crucial parts in LSTM architecture. Absence of either one hurts the performance. However, the coupling of input gate and forget gate in GRU keeps the unbounded cell state from propagating

through layers, which functions the same with the output gate.

Later, more lighter LSTM originated designs are put forward. [Dey and Salem, 2017] presented GRU1, GRU2, and GRU3 three variants of GRU to reduce parameters while retain performance. [Zhou et al., 2016, Heck and Salem, 2017] further modified GRU architecture to propose simpler one-gate designs, minimal gated unit (MGU) and its variants, having fewer parameters compared with LSTM and GRU.

Not only single LSTM unit designs are advanced, but stacked LSTM networks also have been widely applied to various areas. [Ullah et al., 2019] stacked five LSTM cells following a single normalizing layer to learn the temporal dependencies for human activity recognition. [Khodairy and Abosamra, 2021] designs a two LSTM architecture stacked in a many-to-one structure to classify driving behaviors, which achieve impressive performance. [Xia et al., 2021] constructed stacked GRU-RNN for energy load prediction by introducing a GRU variant, in which each gate is generated only by previous hidden state and bias, largely reducing the total parameters.

Bidirectional Recurrent Neural Networks

Bidirectional LSTM (BiLSTM) networks [Graves and Schmidhuber, 2005, Yu et al., 2019], compared to LSTM which only take advantage of previous time step context, can benefit from both forward and backward context, thus having been widely employed in diverse fields. [Graves and Schmidhuber, 2005] first used BiLSTM for phoneme classification. [Graves et al., 2008] made use of long-range bidirectional interdependencies of data and BiLSTM with with a connectionist temporal classification (CTC) output layer for handwriting recognition. [Sundermeyer et al., 2014] applied BiLSTM to translation modeling in order to fully utilize the source sentence. [Huang et al., 2015] applied BiLSTM to Sequence Tagging. Literature survey indicates that BiLSTM is usually utilized together with other methods, in which BiLSTM plays a role of capturing long-range information for further processing. [Liao et al., 2019] proposed a BiLSTM 3-dimensional residual network to recognize complex hand gestures obtaining high accuracy. [Agirman and Tasdemir, 2022] combine GoogLeNet and BLSTM to detect night-time fire from video.

[Chan et al., 2021] presented BasicVSR pipeline for video super-resolution (VSR) task, in which a typical bidirectional recurrent network is adopted to allow features to be temporally propagated forward and backward independently. This bidirectional propagation outperforms local propagation where the received information is confined in a local window, and unidirectional propagation where the whole sequence information accessed by different frames is not balanced.

Convolutional LSTM Network (ConvLSTM)

The convolutional LSTM [Shi et al., 2015] was built upon LSTM with a Peephole Connection, which is a fully connected LSTM (FC-LSTM) on 1-dimensional variables. Convolution operation enables ConvLSTM to learn not only temporal information but also spatial information, which indicates 3-dimensional variables. Therefore, the ConvLSTM design can be mathematically expressed as:

$$f_t = \sigma(W_{fx} * \mathcal{X}_t + W_{fh} * \mathcal{H}_{t-1} + P_f \cdot \mathcal{C}_{t-1} + b_f) \tag{1.22}$$

$$i_t = \sigma(W_{ix} * \mathcal{X}_t + W_{ih} * \mathcal{H}_{t-1} + P_i \cdot \mathcal{C}_{t-1} + b_i) \tag{1.23}$$

$$\tilde{\mathcal{C}_t} = \tanh(W_{\tilde{c}x} * \mathcal{X}_t + W_{\tilde{c}h} * \mathcal{H}_{t-1} + b_{\tilde{c}}) \tag{1.24}$$

$$\mathcal{C}_t = f_t \cdot \mathcal{C}_{t-1} + i_t \cdot \tilde{\mathcal{C}_t} \tag{1.25}$$

$$o_t = \sigma(W_{ox} * \mathcal{X}_t + W_{oh} * \mathcal{H}_{t-1} + P_o \cdot \mathcal{C}_t + b_o) \tag{1.26}$$

$$\mathcal{H}_t = o_t \cdot \tanh(\mathcal{C}_t), \tag{1.27}$$

where '$*$' denotes convolution operation and '$\cdot$' is element-wise multiplication. ConvLSTM outperforms FC-LSTM in capturing spatiotemporal interactions and has fewer parameters.

[Song et al., 2018] presented the Pyramid Dilated Bidirectional ConvLSTM (PDB-ConvLSTM) for video object detection, by building backward LSTM unit to learn deeper backward information upon spatiotemporal features learned in forward process, and incorporating pyramid dilated convolutions with two different dilate rates into LSTM to learn multi-scale features and enable features to communicate across two dilated bidirectional levels.

Based on ConvLSTM, [Wang et al., 2022b] proposed Spatiotemporal LSTM (ST-LSTM) unit featuring a zigzag memory flow, instead of horizontal memory flow in normal LSTM unit, propagating through all stacked blocks vertically and all temporal states horizontally enabling memory representations at different levels to interact.

Hybrid RNN

Many hybrid architectures are proposed in recent years to make fully use of the long-range dependencies capturing capability of RNN model. These incorporated architectures either augment the extraction of spatial dimension information or reinforce the conveying ability of the global temporal information across long-range gaps, both of which escalate and enhance the RNN architecture to handle more complex scenarios.

[Lin et al., 2020] constructed a self-attention (SA) memory module including feature aggregation part to obtain the global context information, memory updating part through a gating mechanism and output part, in order to capture long-range dependencies in both global spatial and temporal dimensions. The self-attention memory module is embedded into a regular ConvLSTM to build SA-ConvLSTM architecture for the spatiotemporal prediction.

An attention based Bidirectional CNN-RNN network [Basiri et al., 2021] was proposed for sentiment analysis. In this network, both bidirectional LSTM and bidirectional GRU are used in parallel to process long sequence dependencies in both directions. The subsequent attention mechanism is applied to enable the model to pay different attention to different words. Then, they used CNN with different kernel sizes to make features more robust to their corresponding positional variations. After these separate operations, concatenation of features and fully connected layers are used to fusion features for final output.

A hybrid CNN-RNN architecture proposed in [Nasir et al., 2021] is applied to fake news detection, where CNN is employed to extract local features and LSTM is utilized to learn long-range dependencies of these extracted local features from the

CNN outputs, for final classification.

1.1.4 Deep Generative Models

Generative models with a long history can be traced back to the 1950s [Cao et al., 2023], when sequential data such as speech and time series such as music [Zhang et al., 2023a] were generated. However, the performance of generative models did not improve significantly until deep learning got prosperous again, owing to the availability of large datasets and advancement of computing resources [Zhang et al., 2023a].

Deep generative models can be grouped into two main classes [Zhang et al., 2023a]: likelihood-based and energy-based. Likelihood-based models, including variational autoencoder [Kingma and Welling, 2013, Vahdat and Kautz, 2020], autoregressive models [Yu et al., 2022] and flow models [Papamakarios et al., 2021, Kobyzev et al., 2020], have to design a tractable density function to optimize parameters with respect to the log-likelihood of training data. Energy-based models, including generative adversarial networks [Goodfellow et al., 2014, Karras et al., 2021] and diffusion models [Rombach et al., 2022, Ruiz et al., 2023], learn simply a tractable process of sample generation [Goodfellow et al., 2020] and are more flexible to parameterize but difficult to train [Song and Kingma, 2021].

Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GAN) has attracted a lot of interest since its inception and now has been successfully applied to vision and language generating areas. GAN, first presented in [Goodfellow et al., 2014], consists of two components a discriminator (D) and a generator (G).

Given a prior distribution $p(z)$ over a noise vector $z \sim p_z(z)$, the differentiable generator function $G(z;\theta_G)$ with learnable parameters θ_G, takes z as input and generates samples which have a distribution p_G. The generator $G(z;\theta_G)$ is trained to learn the parameters θ_G to transform unstructured noise input z into samples as realistic as possible [Goodfellow et al., 2020].

The discriminator $D(input; \theta_D)$ with learnable parameters θ_D, takes training samples $x \sim p_{data}$ and generated samples $G(z) \sim p_G$ as inputs, to produce corresponding estimate and to classify x as real and and $G(z)$ as fake. D is trained to maximize the probability of assigning the correct class label to both x and $G(z)$.

The target is to make distribution p_G of the generated samples $G(z)$ approximately the same with distribution p_{data} of the observed training sample x. Therefore, D and G as rivals are playing a min-max game which can be expressed as:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)} log[D(x)] + \mathbb{E}_{z \sim p_z(z)} log[1 - D(G(z))] \tag{1.28}$$

where G is trained for generating more realistic output under the guidance of D, which is trained to indicate that the output generated by G is not real until optimized. However, GAN and fails to be in control of the generated sample types [Aldausari et al., 2022] and suffers unstable training .

A straightforward extension [Mirza and Osindero, 2014] of vanilla GAN is to condition G and D on some extra information c, which could be class labels or information from other modalities such as text. Thus, the data generation process can be directed by the extra conditioning information, and the Expression 1.28 can be extended to:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)} log[D(x|c)] + \mathbb{E}_{z \sim p_z(z)} log[1 - D(G(z|c))] \tag{1.29}$$

A more stable generative architecture, named deep convolutional generative adversarial network (DCGAN) [Radford et al., 2015] is proposed by using strided convolutions for discriminator and fractional-strided convolutions (deconvolution) for generator. Literature search shows that a large number of recent generative adversarial models utilize CNN in their generator and discriminator design, to improve performance and stabilize training process.

Sequence generative adversarial network (SeqGAN) [Yu et al., 2017b] was proposed to solve a problem of propagating gradient update from the discriminator to the generator when generating discrete sequences. In this work, LSTM is adopted as generator to handle vanishing and exploding gradient. CNN is chosen to construct the discriminator for its effectiveness int sequence classification.

Diffusion Models

Diffusion models [Sohl-Dickstein et al., 2015, Song and Ermon, 2019, Ho et al., 2020, Song et al., 2020b] have attracted a large number of interests as a new family up to date among deep generative models by breaking up the long-time dominance of generative adversarial networks (GANs), showing impressive capabilities in a wide variety of generative modeling areas [Yang et al., 2023a, Croitoru et al., 2023], such as image synthesis [Dhariwal and Nichol, 2021, Rombach et al., 2022, Saharia et al., 2022a], image/video generation [Ho et al., 2022b, Yang et al., 2023b], super-resolution [Li et al., 2022, Saharia et al., 2022c, Whang et al., 2022, Kawar et al., 2022], multi-modal tasks [Nichol et al., 2021, Ramesh et al., 2022, Saharia et al., 2022b, Ho et al., 2022a, Ceylan et al., 2023, Blattmann et al., 2023, Zhang et al., 2023b, Ruiz et al., 2023, Kawar et al., 2023, Lin et al., 2024, Chen et al., 2024] and many others [Peebles and Xie, 2023, Stypułkowski et al., 2024].

Diffusion models, based on probability, works by progressively destructing data with gradual noise injections in a forward diffusion process and then learning to reverse this process for recovering clean input from the diffused noisy data [Yang et al., 2023a, Croitoru et al., 2023]. According to the current research, diffusion models can be categorized into three formulation groups: denoising diffusion probabilistic models (DDPMs) [Sohl-Dickstein et al., 2015, Ho et al., 2020, Nichol and Dhariwal, 2021], score-based generative models (SGMs) [Song and Ermon, 2019, Song and Ermon, 2020, Vahdat et al., 2021], and stochastic differential equations (SDEs) [Song et al., 2020b].

(1) Denoising Diffusion Probabilistic Models (DDPMs)

DDPMs take advantage of two Markov chains [Yang et al., 2023a, Croitoru et al., 2023]: forward and backward Markov chain. A forward Markov chain perturbs or transforms clean input data towards approximately pure noise by diffusing predetermined noise such as Gaussian noise into the data step by step, until the data distribution is transformed to the injected noise distribution such as the predetermined Gaussian noise. In this forward diffusion process, the clean data is gradually

injected by noise and finally has the same tractable distribution with noise distribution.

A reverse backward Markov chain recovers noise-like heavily corrupted data back to original data by learning transition kernels parameterised with deep neural networks to sequentially get rid of noise until the original data is restored. Specifically, [Ho et al., 2020] used a U-Net backbone and shared parameters across time. They train the reverse process to match the actual time reversal of the forward process by minimizing the Kullback-Feebler (KL) divergence between the forward Markov chain and backward Markov chain.

(2) Score-Based Generative Models (SGMs)

Score (function) is defined as the gradient of the logarithm of a unknown probability density function with regard to its input data. Once the score function is obtained by directly training a parameterized score network [Vincent, 2011, Song et al., 2020a], Langevin dynamics [Welling and Teh, 2011] can recursively generate samples from its unknown probability density using only this score function, without estimating this unknown probability density of input data.

Based on above key idea, [Song and Ermon, 2019] proposed Noise Conditional Score Network (NCSN) to enhance score-based generative modeling by perturbing input data with various levels of noises, ranging from large enough noise to alleviate obstacles from the manifold hypothesis, to small enough noise to minimize the effect over data. Thus the score function becomes noise-level conditioned. Their proposed network also enables simultaneous estimation of scores corresponding to all noise levels using a single conditional score network. Specifically, NCSN combines U-Net [Ronneberger et al., 2015] with dilated convolution [Yu and Koltun, 2015, Yu et al., 2017a] for score network, introduces annealed Langevin dynamics to generate samples, and modifies conditional instance normalization [Dumoulin et al., 2016] to offer conditioning on noise level.

(3) Stochastic Differential Equations (SDEs)

Built upon DDPMs and SGMs, SDEs further generalizes the idea of perturbing data using multiple noise levels to an infinite number of noise levels, that is, a continuous diffusion process, which is the solution of a stochastic differential equation (SDE). The backward process with reverse-time SDE requires score function of the probability density at every time step. Therefore, a neural network, with perturbed data and the time step as input, is used to estimate the score functions and generate samples through SDE solvers [Song et al., 2020b, Croitoru et al., 2023].

1.2 Background of Deinterlacing and Demosaicing

1.2.1 Interlacing and Deinterlacing

Interlacing is a spatio-temporal sampling scheme, consisting of odd and even fields, that was introduced to strike a balance between spatial and temporal resolution because analog TV broadcasting did not support a bandwidth to deliver progressive video at a high enough frame rate.

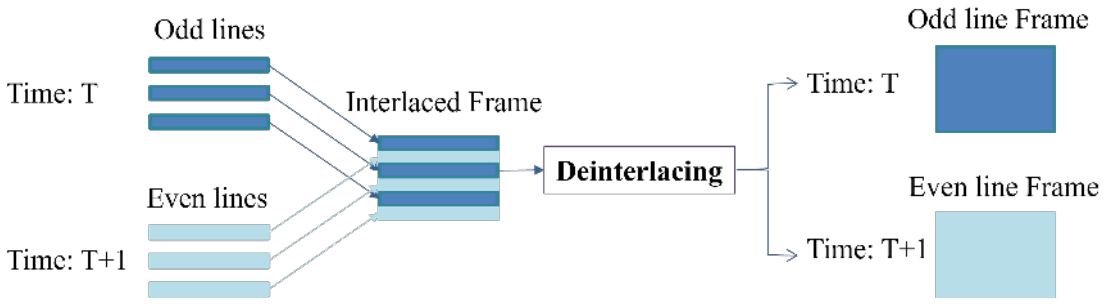


Figure 1.1: Illustration of the interlacing and deinterlacing process.

Interlaced scan refers to alternately displaying odd line field or even line field of a recorded or live dynamic scene. The interlaced scan doubles the temporal rate at the same bandwidth at the expense of vertical spatial resolution resulting in better visual perception of scenes with fast motion. Due to the time difference

of two adjacent fields, the interlaced fame, composed of these two adjacent fields, shows severe comb effect if directly displayed on a progressive device. Therefore, deinterlacing, as shown in Figure 1.1, is required to convert interlaced frame to full vertical resolution frame.

Even though almost all digital cameras today are progressive, some digital TV broadcasting is still interlaced, and there is a large amount of purely interlaced and telecined catalogue content. Hence, deinterlacing remains as a problem of interest to display interlaced broadcast on progressive displays as well as conversion of existing interlaced and telecined catalogue content for progressive broadcast.

1.2.2 Mosaicing and Demosaicing

Mosaicing refers to a color sub-sampling scheme using a color filter array (CFA) pattern that is universally employed in all single-sensor digital still and video cameras.

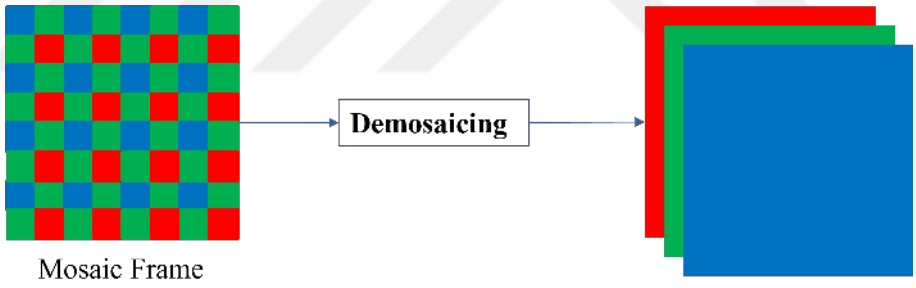


Figure 1.2: Illustration of the mosaicing and demosaicing process in Bayer pattern.

CFA is placed over pixels to filter the incoming light by wavelength range, so that received light intensities are separated into different colors according to the filtered wavelength. Normally rendered viewable color images contain red, green and blue three color channels for each color pixel. However, a raw format image recorded by a single-chip digital image sensor only have one color for each pixel and the colors vary across pixels. Since each pixel only has a single color channel, this raw format image is a mosaic image. Bayer filter, as the most commonly used CFA, arranges the color pattern to determine the proportions and positions of red, green

and blue colors. For the reason that human eyes are more sensitive to green light, Bayer pattern distributes twice as many green pixels as red or blue to mimic the color perception of the human eyes. Therefore, in a Bayer pattern mosaic image, there are half green, one quarter red and one quarter blue pixels.

Demosaicing, as shown in Figure 1.2, is used to reconstruct the missing colors for each pixel from a subsampled mosaic image, in order to form a full color image with red (R), green (G) and blue (B) three channels for each pixel.

1.3 Motivation and Thesis Outline

1.3.1 Motivation

Deinterlacing and demosaicing are commonly used in the image processing pipeline for consumer video. Yet, the industry employs simple methods for demosaicing, deinterlacing, telecine, and 50-to-60 Hz and vice versa conversion. Over the years, many conventional solutions have been proposed by the academia for demosaicing [Losson et al., 2010], intra-field deinterlacing [Kim et al., 2007, Yoo and Jeong, 2002], inter-field deinterlacing [Kwon et al., 2003], and spatio-temporal super-resolution [Shechtman et al., 2005]. However, these solutions do not employ learned image/video priors; hence, introduce blurring or disturbing color artifacts, edge artifacts, and/or flicker.

On the other hand, real-world demosaicing and deinterlacing are well-suited for supervised learning from synthetic data, because the forward (degradation) model is pre-determined and is fixed for these upsampling problems. The forward model for both problems can be expressed as

$$\mathbf{y} = \mathbf{S}\mathbf{x} + \mathbf{v} \tag{1.30}$$

where $\mathbf{x}$ is the ideal progressive video, $\mathbf{y}$ denotes the subsampled video, $\mathbf{v}$ is observation noise, and $\mathbf{S}$ is the subsampling matrix defined by the interlace or known CFA pattern.

The literature on learned video demosaicing and deinterlacing benefiting from *both local and global spatio-temporal correlations* is scarce. Moreover, the litera-

ture treats demosaicing and deinterlacing as separate problems although they share a common forward model (1.30). In this work, we propose a single novel multi-picture architecture (for both problems) that integrates both local and global spatio-temporal features extracted by modified deformable convolution blocks and efficient residual top-k self-attention blocks, respectively, and employs separate reconstruction blocks to estimate the missing pixels.

1.3.2 Thesis Outline

The rest of this thesis is organized as: We review related works and our main contributions in Chapter 2. Chapter 3, published in [Ji and Tekalp, 2021], introduces an initial learning architecture for deinterlacing task with feature alignment by using deformable residual convolution blocks. Chapter 4, published in [Ji and Tekalp, 2022], presents an improved multi-field deinterlacing architecture by employing a combination of deformable convolution residual blocks and self-attention blocks, which ranks first at submission time on MSU benchmark [MSU, 2022]. Based on the work in Chapter 3 and Chapter 4, Chapter 5 further modifies and improves deinterlacing architecture to propose a new multi-picture learning architecture for both deinterlacing and demosaicing tasks and extensive experiments show that our model yields state-of-the-art results and generalizes better than all existing methods for both tasks when trained and tested on different datasets. Chapter 6 concludes this thesis.

Chapter 2

RELATED WORKS AND CONTRIBUTIONS

We review related works on learned deinterlacing and demosaicing in Section 2.1.1 and Section 2.1.2, respectively. Prior works on deformable convolutions and integration of local and global features are reviewed in Section 2.1.3 and Section 2.1.4, respectively, in order to more clearly identify our novel contributions on these fundamental blocks. Section 2.2 summarizes the novelty and contributions of this thesis.

2.1 Related works

2.1.1 Learned Deinterlacing

Early works either consider only intra-frame deinterlacing [Zhu et al., 2017, Akyüz et al., 2020] or do not employ feature-level alignment of supporting fields [Bernasconi et al., 2020, Zhao et al., 2022a]; hence, their results are sub-optimal in the presence of large motion. There are few recent works that consider feature-level field alignment using deformable convolutions for the deinterlacing task. One of the first works was by us [Ji and Tekalp, 2021]. Other papers that use deformable convolutions for feature-level field alignment for deinterlacing have concurrently appeared [Liu et al., 2021, Ji and Tekalp, 2022, Zhao et al., 2022b].

[Liu et al., 2021] aimed at exploring the spatial temporal correlations within two successive fields by using deformable convolution for aligning to be interpolated fields, followed by a couple of channel attention residual blocks for the direct mapping to the missing field feature. However, their work discards frames farther from the present one and is regraded as intra-frame deinterlacing without long-dependencies consideration. [Zhao et al., 2022b] utilized vertical pixel shuffle to restore vertical resolution and then employed U-Net structure enabling large receptive fields and

multiple scales to estimate offsets for deformable convolution in order to align features. Different from the two works, our later conference paper [Ji and Tekalp, 2022] proposed to align a couple of successive given fields using deformable convolution residual blocks for spatial feature alignment and then to combine the aligned features with features globally processed by vanilla self-attention (SA) block for final reconstruction. Recent work [Gao et al., 2023], closely based on [Chan et al., 2022], proposed a bidirectional U-Net-like feature alignment using deformable convolution and feature refinement architecture at multi-scales guided by optical flow.

In addition to CNN alignment based deinterlacing works mentioned above, [Yeh et al., 2022] proposed VDNet by using deformable convolution to align base deinterlaced images through spatial line average and temporal line average techniques and then combining deformable convolution with bidirectional RNN to optimize temporal feature and resize feature map. [Song et al., 2023] presented a transformer based deinterlacing approach to better utilize the inter-frame movement correlation by self-attention mechanism.

This thesis extends and advances the preliminary results in [Ji and Tekalp, 2022]. In particular, we replace the self-attention (SA) block with an efficient residual top-k SA block for global feature processing. We also modify the deformable convolution residual blocks in [Ji and Tekalp, 2022] by removing the redundant residual connection and adding a convolution layer between the two deformable convolution layers to obtain better performance. We elaborate on the novelty of our paper in more detail in Section 2.2.

2.1.2 Learned Image and Video Demosaicing

Deep learned models provide important quantitative and visual performance gains over conventional solutions.

Early approaches for still images employed multi-model [Tan et al., 2018] or multi-stage [Wang et al., 2020] deep convolutional networks (CNN) to reconstruct and refine the G, R and B channels,separately. [Zhang et al., 2022a] employed a deep CNN denoiser prior for plug-and-play restoration. [Feng et al., 2021] applies

convolution-attention network to multispectral demosaicing. [Zhang et al., 2022b] proposed a spatially adaptive convolutional network by feeding the color filter array (CFA) pattern to the network, which inspired us to propose the separate reconstruction blocks in our work.

Furthermore, some joint approaches are proposed to jointly solve demosaicing and denoising problems, or to jointly handle demosaicing and superresolution problems. [Sharif et al., 2021] proposed a deep network leveraging the depth and spatial attention to jointly tackle both demosaicing and denoising problems. [Zhang et al., 2024] presented a deep guided attention network to jointly study the problem of denoising and demosaicing, considering both high intensity and high sampling rate of green channel for both problems, by first training denoising and demosaicing networks sequentially and then learning jointly. [Xu et al., 2020] designed a joint demosaicing and superresolution network built upon deep residual-dense structure with channel attention to facilitate information flow for final reconstruction, preceded by a pre-demosaicing step. [Qian et al., 2022] investigated the effect of the combining order in pipeline on performance and suggested that the pipeline of denoising, superresolution and demosaicing yields better performance in sequential processing. Moreover, their experimental results show the superiority of joint processing over sequential processing by incorporating the same residual in residual dense block [Wang et al., 2018] into different comparing schemes. However, [Guo et al., 2023] argues that the sequential combination of denoising and demosaicing will be old-fashioned because of the joint work with the overwhelming deep learning. Partial reasons are that deep learning based processing chains are data and device specific, and computational efficiency and burden are also challenging. In our latest architecture, instead of adding heavy-weight denoising layers, we utilize top-k selection to filter out redundant elements and remove irrelevant information.

Multi-picture demosaicing that take spatio-temporal color correlations into account have only very recently been started to be considered for burst image and video demosaicing. [Kokkinos and Lefkimmiatis, 2019] applied convolutional iterative network on burst images by aligning each supporting picture to the reference

using enhanced correlation coefficient. [Ehret et al., 2019] employed an inverse compositional algorithm to register frames when performing fine-tuning on burst images. [Tan et al., 2022] presented a two-stage CNN model for joint demosaicing and denoising of burst Bayer images. [Dewil et al., 2023] proposed recurrent networks with motion compensation for joint video denoising and demosaicing.

In Chapter 5, we propose a common multi-picture architecture for both video demosaicing and deinterlacing that perform local and global feature alignment by registering supporting pictures with the reference using modified deformable convolutions and top-k attention, respectively. We also show that using separate reconstruction blocks for R, G, and B channels provides better results than conditioning on the CFA pattern as done in [Zhang et al., 2022b].

2.1.3 Deformable Convolution for Video Processing

Deformable convolution, with a larger and more flexible receptive field compared to regular convolution, was first proposed for object detection and semantic segmentation [Dai et al., 2017] and then was developed by [Zhu et al., 2019]. In deformable convolution, pixel positions under the kernel are displaced according to learned offsets $\Delta p_{(i,j)}$. Hence, the deformable convolution can be expressed as:

$$y(p_{(m,n)}) = \sum_{(i,j)} w(i,j) \cdot x(p_{(m-i,n-j)} + \Delta p_{(i,j)}), \tag{2.1}$$

where $p_{(m,n)}$ is the current pixel position, $x(p_{(m,n)})$ is the pixel at position $p_{(m,n)}$, $w(i,j)$ are kernel weights, and for a 3×3 deformable convolution $i, j \in (-1, 0, 1)$.

It has later been adopted for video processing tasks, such as video super-resolution (VSR) [Tian et al., 2020, Wang et al., 2019, Ying et al., 2020, Lee et al., 2022], frame rate up-conversion [Cheng and Chen, 2021, Danier et al., 2022, Li et al., 2023a, Lei et al., 2023], next-frame prediction [Yılmaz and Tekalp, 2021], video compression [Chen et al., 2022, Yılmaz et al., 2023] and video restoration tasks [Jiang et al., 2022, Wu et al., 2023, Liu et al., 2022]. Temporal alignment of supporting pictures with a reference picture at the feature level using deformable convolution without using optical flow was first proposed in TDAN [Tian et al., 2020]. This was

extended to the pyramid, cascading, and deformable convolution alignment sub-network in EDVR [Wang et al., 2019] and later was adopted by many other VSR tasks.

In Chapter 3, Chapter 4 and Chapter 5, we modified the offset estimation method in deformable convolutions tailoring it to demosaicing and deinterlacing tasks, and proposed an initial deformable convolution residual (DfRes) block and a modified deformable convolution (DfConv) block for feature-level picture alignment.

2.1.4 Integrating Local and Global Features via Self-Attention

Given the success of non-local image processing and the ability of self-attention (SA) to exploit long range interactions, there is growing interest in using SA layers either as stand-alone primitives or to augment ConvNets for image processing. Attention augmented convolutional networks combining both convolutional and SA layers by concatenating respective feature maps has been proposed [Bello et al., 2019]. However, the memory requirement and computational complexity of modeling global dependency by dot-product attention are quadratic with respect to picture size, which prohibits its application to high-resolution images and large videos.

Recently, several works have emerged proposing efficient SA implementations [Tay et al., 2020]. Shen *et al.* proposed an efficient implementation of SA with linear complexity [Shen et al., 2021]. A later work [Xia et al., 2022] further incorporates a Gaussian approximation to decompose the exponential function of queries and keys in order to change the order of multiplications.

Both [Shen et al., 2021] and [Xia et al., 2022] utilize all tokens in matrix multiplications, which contributes to the computation cost as well as introducing a noise effect due to irrelevant tokens. Recently, [Wang et al., 2022a] proposed an efficient k-NN self-attention to keep only the top k similar tokens instead of all tokens in the attention weight map in order to benefit from long range dependencies while filtering out irrelevant tokens.

In Chapter 5, we combine these ideas with some improvements on vanilla SA in Chapter 4, to propose the residual efficient kSA block: i) we show that the

kSA block with a residual connection yields better performance than without a residual connection, ii) different from [Bello et al., 2019], we show that combining features computed by deformable convolution and kSA blocks by addition rather than concatenation yields better performance, iii) we propose an efficient top-k SA by changing multiplication order to reduce training and test time while preserving the performance.

2.2 Thesis contributions

This section provides a condensed contributions of our thesis with respect to each chapter.

Chapter 1 presents an concise overview of deep learning, starting from a brief taxonomy of deep learning approaches and continuing with three widely employed neural networks including CNN, RNN and generative networks with respective fundamentals and pillar architectures, together with an primary introduction to deinterlacing and demosaicing, in order to lay a preliminary foundation for further work of our thesis, and concludes with our motivation for thesis subject.

Chapter 2 reviews related literature background on learned deinterlacing, learned demosaicing, deformable convolution and self-attention, all of which contribute to our main work.

Chapter 3 proposes to use deformable convolution for deinterlacing feature alignment for the first time [Ji and Tekalp, 2021]. While deformable convolution has shown excellent feature registration among multiple frames in image restoration and SR, prior works on deep deinterlacing have not registered features of neighboring fields with the reference field. We designed an initial deinterlacing architecture by presenting Deformable Residual Convolution Blocks (DfRes) with two variants of different sizes, to align each neighboring field feature to the reference field feature for further reconstruction. An indicator was also proposed to distinguish whether the reference field is an odd filed or even field in order to properly form the final progressive output frame.

Chapter 4 proposes an improved deinterlacing architecture [Ji and Tekalp, 2022]

by combining self-attention mechanism in parallel with the DfRes blocks to enable better multi-field feature alignment and fusion. The separate reconstruction module is utilized to restore even and odd fields separately from high level features under the guidance of indicator flag. The overall performance is significantly improved over those in [Ji and Tekalp, 2021].

Chapter 5 proposes a common multi-picture architecture for both video demosaicing and deinterlacing (since both are video upsampling problems with fixed degradation pattern) featuring modified deformable convolution layers and efficient residual top-k SA layers to additively combine local and global features, respectively, and separate reconstruction blocks for each channel to be upsampled.

Specifically, We proposed an enhanced deformable convolution (DfConv) blocks for feature alignment based on the DfRes blocks proposed in [Ji and Tekalp, 2022] by removing the redundant residual connection and inserting a regular convolution layer in between the two deformable convolution layers (Df). We replace the SA block in [Ji and Tekalp, 2022] with novel efficient residual top-k SA block, which can reduce the training and test time to about half compared to [Ji and Tekalp, 2022] while retaining performance. We demonstrate the superiority of adding the features aligned by the deformable convolution layers and self-attention layers (in parallel) over concatenating them as proposed in [Bello et al., 2019]. We show the advantage of using separate reconstruction blocks for each picture to be reconstructed according to its particular subsampling pattern than using a single reconstruction block for all picture types.

Ablation studies and extensive experiments validate the state of the art results, as well as the effectiveness and generalizability, of our model for both demosaicing and deinterlacing tasks.

Chapter 6 summarizes the work presented in this thesis.

Chapter 3

LEARNED MULTI-FIELD DEINTERLACING WITH FEATURE ALIGNMENT VIA DEFORMABLE RESIDUAL CONVOLUTION BLOCKS

3.1 Contributions

Prior works on deep deinterlacing have not registered features of neighboring fields with the reference field. It has been shown that deformable convolution can perform excellent feature registration among multiple frames [Tian et al., 2020, Wang et al., 2019] in image restoration and SR. While it is not straightforward how to align features of neighbor frames with the reference frame in frame interpolation, since the reference frame is non-existent, it is possible to align features in deinterlacing because existing lines in the reference field serve as anchors to align features of neighbor frames. To the best of our knowledge, the work [Ji and Tekalp, 2021] in this chapter is the first to propose aligning features using deformable convolution for deinterlacing at paper submission time.

We also propose two alternative deformable residual convolution block designs in Section 3.2.2 and Section 3.2.3, where the latter has smaller number of parameters.

3.2 Proposed Method

We first present our overall deinterlacing network architecture. We, then, introduce two alternative designs for the proposed deformable residual convolution block used in our architecture.

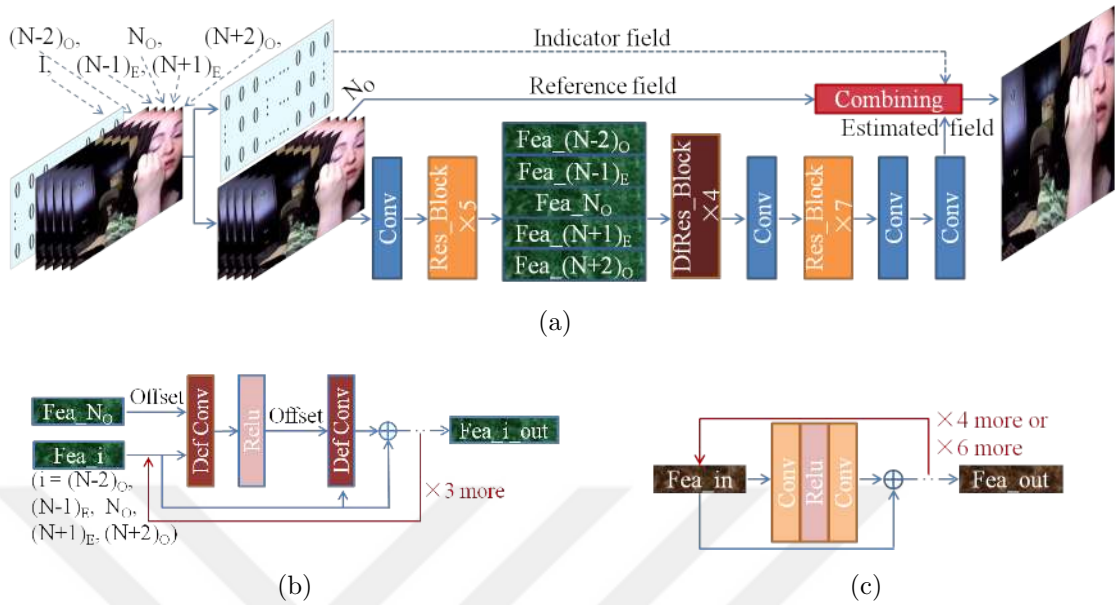


Figure 3.1: a) The proposed SR deinterlacing network with five input data fields $(N-2)_O, (N-1)_E, N_O, (N+1)_E, (N+2)_O$, and one indicator field (I), where the reference field for alignment is N_O, the field to be estimated is N_E. b) Details of a DfRes block used for feature alignment (shown by the brown box in (a)). c) Details of a residual block depicted by orange boxes in (a).

3.2.1 Proposed Deinterlacing Network Architecture

The overall deinterlacing architecture is shown in Figure 3.1 (a) The input consists of an odd number of data fields (e.g., we use 5 fields, $(N-2)_O, (N-1)_E, N_O, (N+1)_E, (N+2)_O$, when the reference field is N_O) and an indicator field (I) stacked together. Although the reconstructed fields $(N-1)_O$ and $(N-2)_E$ would be available at test time since they are in the past of the current field N_E to be estimated, we chose not to use them in order to avoid potential error propagation. The indicator is set to 1 when the reference field is an even field; or, it is set to 0 when the reference field is an odd field. The indicator is separated from the data fields before inputting data fields to the model. The indicator is later used when forming the output progressive frames as explained below.

The data fields are first fed into five regular residual blocks (Res_Block ×5) to

extract features. These features are then fed into the alignment module to align the features of the supporting fields to those of the reference field using four deformable residual convolution blocks (DfRes_Block ×4). Then, the aligned features are stacked and processed by ×7 regular Res_Blocks to generate the estimated field.

Finally, we combine the reference field and estimated opposite parity field with the help of the indicator field to form progressive output frame as shown in Figure 3.3 (d) (f) (g). If the indicator field is 0 (1), we place the estimated field in the even (odd) lines of the output progressive frame.

3.2.2 Deformable Residual Convolution (DfRes) Block

In the literature, the offsets for deformable convolution layers are learned by applying regular convolutions directly to concatenated input feature maps [Dai et al., 2017, Tian et al., 2020, Ying et al., 2020, Cheng and Chen, 2021]. In our experiments, we found that learning offsets by applying deformable convolution only on the reference field produces better performance especially in terms of peak signal-to-noise ratio (PSNR), which is one of our contributions.

Our proposed deformable residual convolution (DfRes) block, shown in Figure 3.1 (b) employs two deformable convolution (DefConv) layers instead of regular convolutions in order to align features of supporting fields with those of the reference field. In the first DefConv layer, we treat the reference field features as the offset and perform DefConv on features of one supporting field with this offset. Then, the result is activated by a ReLU layer and is fed into the second DefConv layer as the offset. The red arrow denotes that there are a total of (3+1) DfRes blocks, where each time, we align one supporting field to the reference field.

3.2.3 An Alternative DfRes Block Design

While exploring different architectures, we observed that if we apply Conv-ReLU-DefConv on the difference between the reference field and each supporting field to compute offsets, we obtain performance similar to that of DfRes block, shown in Figure 3.1 (b) with less parameters. This new block is called DfRes_v block,

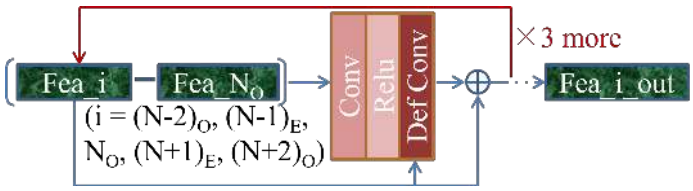


Figure 3.2: DfRes_v, a variant of the proposed DfRes block.

and it is depicted in Figure 3.2. In particular, network with DfRes_v block has 1,741,219 parameters, while network with DfRes blocks has 2,239,747 parameters. In Section 3.3, we provide SR deinterlacing results with both DfRes and DfRes_v blocks.

3.3 Experimental Evaluation

In this section, we first introduce dataset generation, evaluation methods, and details of training procedure in Section 3.3.1. Then, we compare our proposed network architecture using DfRes blocks with other state-of-the-art methods in terms of PSNR and SSIM metrics and visual quality in Section 3.3.2. Finally, we discuss generalization of our pre-trained network to other datasets via cross-validation in Section 3.3.3.

3.3.1 Experimental Settings

Datasets and Evaluation Methods

We used UCF101 [Soomro et al., 2012], REDS [Nah et al., 2019] and Vimeo-90K [Xue et al., 2019] datasets to demonstrate the effectiveness of our method and compare it with other methods. We separated ucf101 dataset into training and test sets according to [Soomro et al., 2012] and tested only on ApplyEyeMakeup class. In the REDS dataset, we used clip00, clip11, clip15 and clip20 (called REDS4) for testing and the remaining clips for training [Wang et al., 2019]. In Vimeo-90K dataset, we used the training and test septuplets as stated [Xue et al., 2019]. We also used Vid4 [Sajjadi et al., 2018] dataset for testing.

In all cases, synthetic interlaced videos are generated by extracting odd and even fields of video frames as shown in Figure 3.3 (a)-(b), where N_O and N_E represent odd and even fields of frame N.

We use peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) to quantitatively evaluate results of all methods. We also provide visuals zooming in on details for subjective evaluation of the results.

Training Procedure

We stack five adjacent data fields $(N-2)_O$, $(N-1)_E$, N_O, $(N+1)_E$ and $(N+2)_O$, when the field to be estimated is N_E, to feed into the model as shown in Figure 3.3 (c)-(d). We train our model for 4,000 iterations with batch-size 32 in each iteration and the input patch size is 64×80. For data augmentation, we randomly flip fields horizontally without rotation so as to keep field sizes in each batch the same.

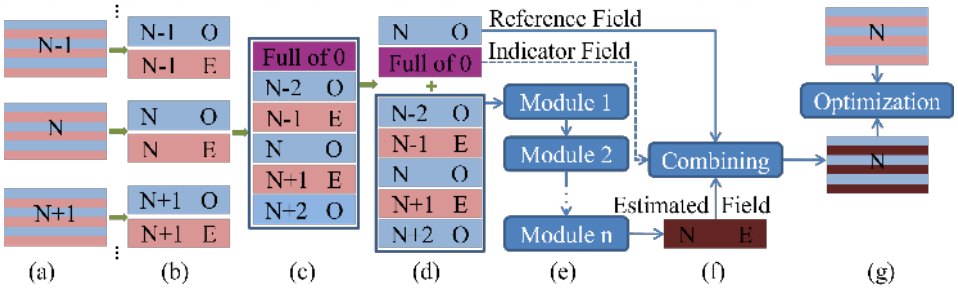


Figure 3.3: Overview of data processing during training.

We use the loss function

$$\begin{aligned} L =& MSE(Y^{pred}, Y^{GT}) + \lambda_{TV} \cdot TV(Y^{pred}) \\ &+ (1 - SSIM(Y^{pred}, Y^{GT})) \end{aligned} \tag{3.1}$$

where Y^{pred} and Y^{GT} are deinterlaced and ground truth frames, respectively, $MSE(\cdot)$ is the mean square error, $TV(\cdot)$ denotes the total variation regularizer, and λ_{TV} is a scale factor, which is set to 2.0×10^{-3} in all experiments.

Table 3.1: Number of parameters for different methods.

DfRes	**DfRes_v**	**EDVR**	**TDAN**	**DUF**
2,239,747	1,741,219	4,916,515	2,081,830	1,608,503

Table 3.2: Comparison of PSNR (top) and SSIM (bottom) for different methods on multiple datasets. The best and second best are marked with Red and Blue color respectively.

Dataset	**DfRes**	**DfRes_v**	EDVR	TDAN	DUF
UCF101	40.08	39.47	39.39	39.50	31.81
	0.990	0.989	0.9875	0.9879	0.869
REDS4	32.18	32.60	32.37	32.47	30.46
	0.927	0.934	0.930	0.935	0.881
Vimeo	33.74	33.98	33.79	33.61	31.49
	0.939	0.942	0.936	0.935	0.885

The optimizer and learning rate settings are the same as those in EDVR [Wang et al., 2019]. All experiments were conducted using Pytorch over tesla_t4.

In Section 3.3.2, we keep the training set up the same for all other networks used for comparison.

3.3.2 Comparison with the State of the Art

We compare our proposed DfRes and DfRes_v methods with three state-of-the-art methods: EDVR [Wang et al., 2019], TDAN [Tian et al., 2020] and DUF [Jo et al., 2018]. We do not modify these three methods except that we change the upsampling step to direct combination of reference field and estimated field specifically for deinterlacing as shown in Figure 3.3 (f)-(g). The number of parameters in these models is shown in Table 3.1. We also perform a comparison with RTDVD [Zhu

Table 3.3: Comparison with RTDVD [Zhu et al., 2017] on Vid4 dataset.

Test on Vid4	RTDVD		**DfRes**	
	PSNR	SSIM	PSNR	SSIM
Folder calendar	25.33	0.934	26.39	0.893
Folder city	29.54	0.947	31.69	0.931
Folder foliage	31.88	0.968	31.50	0.933
Folder walk	34.66	0.983	33.85	0.970
Average	30.35	0.958	30.86	0.932

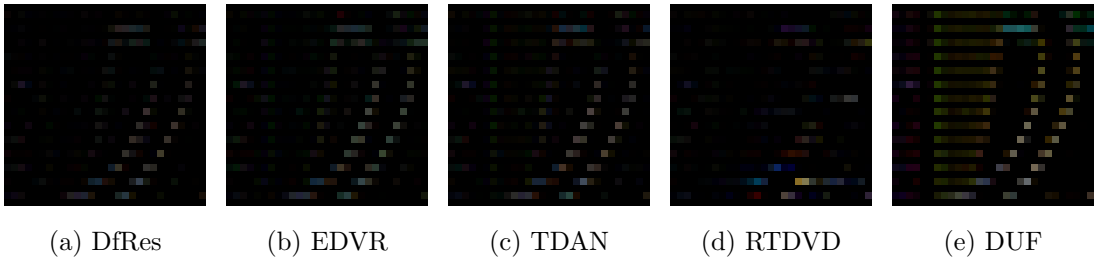
et al., 2017] on Vid4 dataset using their pre-trained model.

The quantitative comparison results on UCF101, REDS and Vimeo-90K datasets are shown in Table 3.2. On these three datasets, our proposed DfRes and DfRes_v methods outperform all other methods in terms of PSNR. Table 3.3 shows the comparison with RTDVD [Zhu et al., 2017] and our method has higher average PSNR value especially on clip calendar and city.

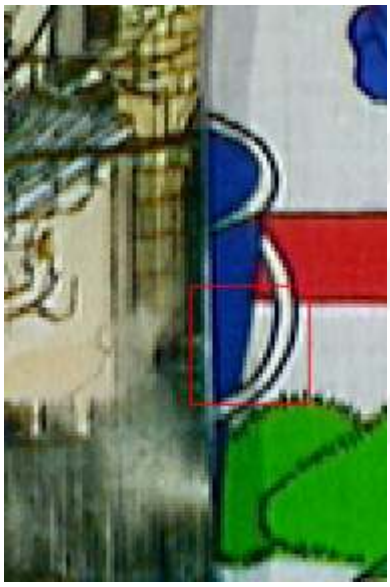
Visual comparisons are shown in Figure 3.4. Frames in (a)-(e) show the differences between estimated frame and the progressive frame. The darker the deinterlaced frame, the closer to the progressive frame. Images in (f) and (g) zoom in the same area of resulting deinterlaced frames for better visual comparison. Resulting frames show that the proposed DfRes method produces less strip noises and color distortion.

3.3.3 Generalization to Other Datasets

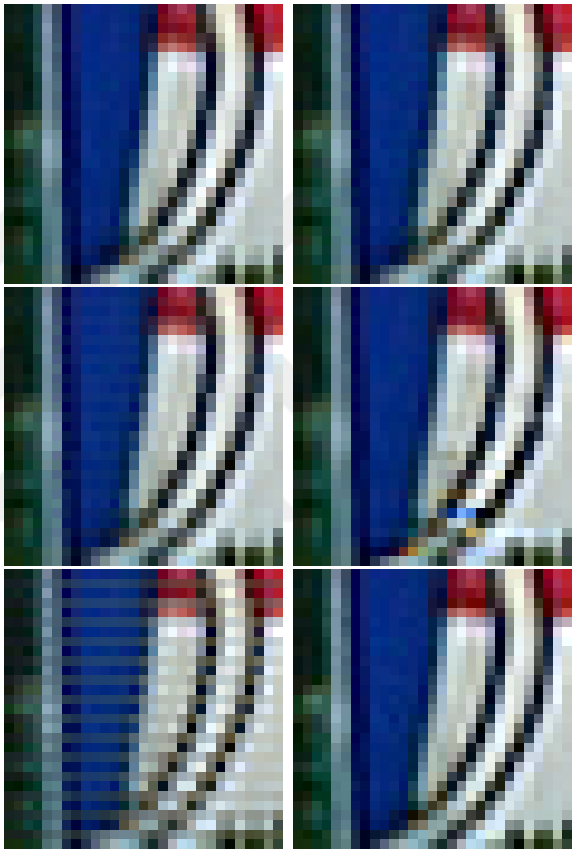
In order to demonstrate generalizability of our models, we test the models that are trained on ucf101, REDS and Vimeo-90k, on the other two datasets and Vid4 [Sajjadi et al., 2018]. Table 3.4 shows that our method in general generalizes well but the performance varies by the amount of motion in the training and test sets.



(a) DfRes (b) EDVR (c) TDAN (d) RTDVD (e) DUF



(f)



(g)

Figure 3.4: Visual evaluation of a deinterlaced frame from Vid4 dataset. (a)-(e) differences between the actual and reconstructed progressive frames. (f) Progressive ground truth frame. (g) Zooming in the red box. From upper left to lower right: DfRes, EDVR, TDAN, RTDVD [Zhu et al., 2017], DUF and ground truth.

Table 3.4: Generalization of the proposed DfRes and DfRes_v methods trained and tested on different datasets (Top: PSNR, Bottom: SSIM).

Train on	**DfRes:** Test on				**DfRes_v:** Test on			
	UCF101	REDS4	Vimeo	Vid4	UCF101	REDS4	Vimeo	Vid4
UCF101	40.08	32.47	37.60	30.86	39.47	32.47	37.23	30.54
	0.990	0.933	0.974	0.932	0.989	0.934	0.972	0.929
REDS	33.53	32.18	34.78	29.04	34.59	32.60	35.56	29.41
	0.898	0.927	0.938	0.902	0.900	0.934	0.944	0.912
Vimeo	34.66	27.22	33.74	28.75	34.81	27.60	33.98	28.73
	0.961	0.818	0.939	0.895	0.961	0.827	0,942	0.896

3.4 Conclusion

In this chapter, we introduce a new deformable residual convolution (DfRes) block for feature alignment, and a new network for video deinterlacing with feature alignment. Experimental results demonstrate that the proposed network yields state-of-the-art performance and DfRes block can be employed to effectively deinterlace videos.

We observe that the PSNR values depend on the amount and type of motion in the test sequences, e.g., ucf101 ApplyEyeMakeup class has still background and small motion in single direction; hence, yields the highest PSNR values.

Chapter 4

MULTI-FIELD DEINTERLACING USING DEFORMABLE CONVOLUTION RESIDUAL BLOCKS AND SELF-ATTENTION

4.1 Contributions

This chapter advances the state-of-the-art in learned deinterlacing as follows:

1) Different from our early work [Ji and Tekalp, 2021], we employ a self-attention module in parallel to the DfRes blocks and show that this new architecture provides improved performance by enabling better multi-frame feature alignment and fusion.

2) We propose to utilize separate reconstruction module to restore even and odd fields separately, with which the overall performance of all methods presented here are significantly improved over those in [Ji and Tekalp, 2021].

3) We provide ablation and generalization results to show the effectiveness of proposed model over competing learned deinterlacing methods.

4.2 Proposed Deinterlacing Architecture

4.2.1 Overview of the Proposed Architecture

The proposed overall architecture is depicted in Figure 4.1 (a). The input to the field feature alignment stage consists of an odd number of fields centered about the reference field. We use an indicator bit, which is 1 when the reference field is an even field; or 0 when the reference field is an odd field, when forming the output progressive frames.

Each data field is first mapped into 64 feature channels via the convolution layer Conv_1 and next processed by five regular residual blocks. These features are then

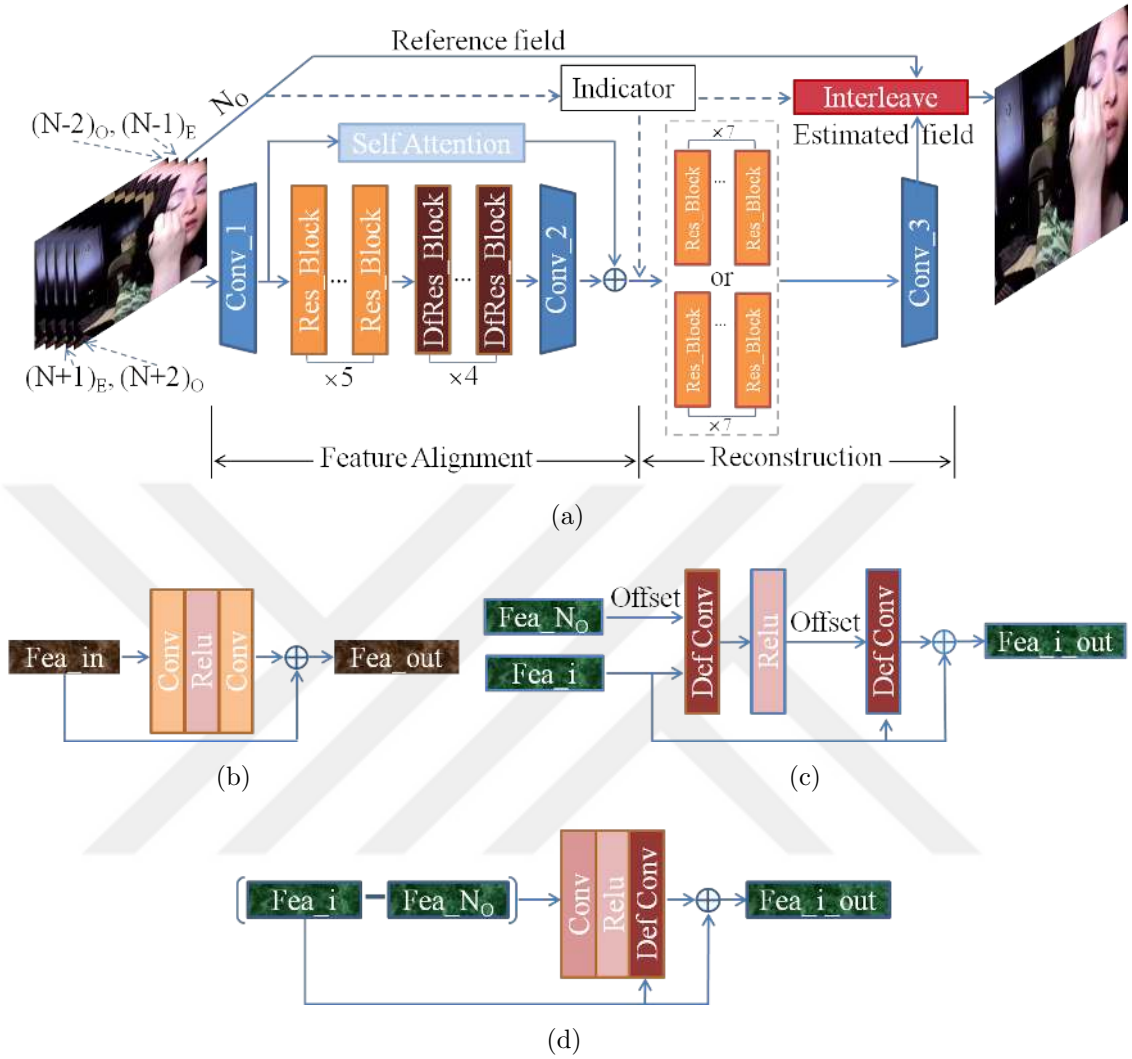


Figure 4.1: (a) The proposed deinterlacing network with five input fields $(N-2)_O, (N-1)_E, N_O, (N+1)_E, (N+2)_O$, where the reference field for alignment is N_O, the field to be estimated is N_E. (b) Block diagram of a standard residual block depicted by orange boxes in (a). (c) Block diagram of a DfRes block shown by the brown box in (a). (d) Block diagram of a differential DfRes (ΔDfRes) block. Fea_i represents each input field. Fea_i_out is the corresponding output feature after one DfRes (ΔDfRes) block and it will replace Fea_i to be aligned through next DfRes (ΔDfRes) block.

fed into four separate alignment modules to align the features of each supporting field to those of the reference field using deformable convolution residual blocks

(DfRes_Block ×4). The aligned features are then stacked and they are fused and reduced to 64 channels through Conv_2.

In addition, we employ an SA module in parallel to the DfRes blocks to enhance the performance of feature registration. The 64-channel feature tensors processed by the SA module and DfRes module in parallel are added together to form the final 64-channel aligned feature tensor. Experiments demonstrate the effectiveness of the proposed strategy.

The 64-channel aligned feature tensor is input to the reconstruction stage, where it is processed by two branches of 7 regular residual blocks each. The last Conv_3 layer maps 64 channels to 3 channels to generate the estimated field.

Finally, we interleave the available reference field and estimated opposite parity field using the indicator bit to form progressive output frames as shown in Figure 4.2 (d) (f) (g). If the indicator field is 0 (1), we place the estimated field in the even (odd) lines of the output progressive frame.

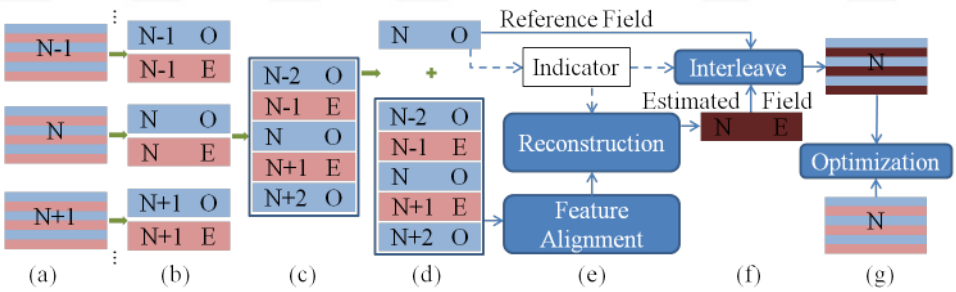


Figure 4.2: Overview of data processing during training. (a)-(b) synthetic interlaced videos are generated by extracting odd and even fields of video frames, where N_O and N_E represent odd and even fields of frame N. (c) input fields. (d)-(g) deinterlacing process.

4.2.2 Field Alignment using Deformable Convolution Blocks

Deformable convolution residual (DfRes) block [Ji and Tekalp, 2021], shown in Figure 4.1 (c), employs two deformable convolution layers instead of regular convo-

lutions in order to align features of supporting fields with those of the reference field.

ΔDfRes Block [Ji and Tekalp, 2021] has similar performance but less parameters compared to DfRes block, and it is depicted in Figure 4.1 (d).

4.2.3 Field Alignment using Efficient Self-Attention

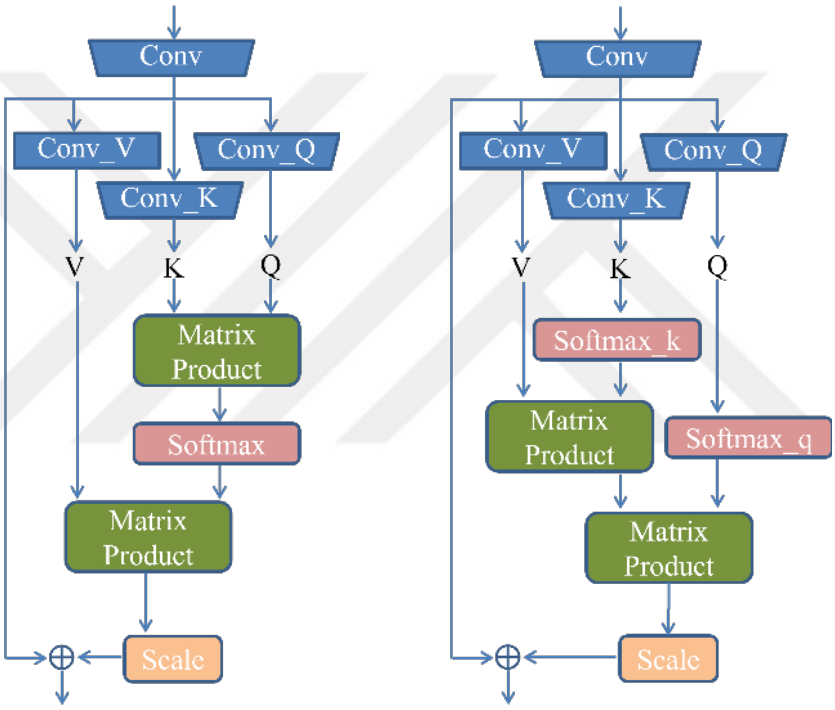


Figure 4.3: (a) Block diagram of the self-attention module, where Softmax is defined in Equation 4.2. (b) Block diagram of the efficient self-attention module, where Softmax_k and Softmax_q are defined in Equation 4.4.

The SA module shown in dashed light blue box in Figure 4.1 (a) and depicted in detail in Figure 4.3 (a), is parallel to feature extraction Res_Block ×5 module and feature alignment DfRes module. The data fields with n pixels in each field are first mapped to 64 channel feature space via regular convolution operations (Conv_1 in Figure 4.1 (a) and Conv in Figure 4.3 (a)). Then, queries ($Q \in R^{8\times n}$), keys

(K $\in R^{8\times n}$) and values (V $\in R^{64\times n}$) are computed by regular convolution operations Conv_Q, Conv_K and Con_V separately, where the output feature channels of Conv_Q and Conv_K are $\frac{1}{8}$ times of those input feature channels as shown with blue trapezoids in Figure 4.3 (a), and the input and output channels of Con_V are the same as shown with blue rectangle in Figure 4.3 (a). After obtaining Q, K and V matrix, we multiply Q and K by transposing Q and feed the result into softmax operation $\rho(Q^T \times K)$ for normalization. Finally, we multiply V and the transpose of the normalized softmax output to gain an initial output of Self Attention Module. The final output of Self Attention Module is the original 64 channel feature resulted from Conv in Figure 4.3 (a) plus the scaled initial output as shown in plus sign in Figure 4.3 (a). Hence, the Self Attention Module operation can be expressed in the following equation:

$$SA(Q, K, V) = V \times (\rho(Q^T \times K))^T \cdot Scale, \tag{4.1}$$

where T denotes matrix transpose operation and $\times$ matrix product. Scale is a learnable factor for scaling initial output. We use softmax function for normalization:

$$Softmax: \ \rho(Y) = \sigma_{row}(Y), \tag{4.2}$$

where σ_{row} applies softmax row-wise.

In dealing with the huge computation cost of self attention, we utilize the following approximate equation in test [Shen et al., 2021]:

$$ESA(Q, K, V) = V \times \rho_k(K^T) \times \rho_q(Q) \cdot Scale, \tag{4.3}$$

where ρ_k and ρ_q are softmax functions utilized to normalize K and Q features. They are defined as:

$$\begin{aligned} Softmax\_k: \ \rho_k(Y) &= \sigma_{col}(Y), \\ Softmax\_q: \ \rho_q(Y) &= \sigma_{row}(Y), \end{aligned} \tag{4.4}$$

where σ_{col} and σ_{row} apply softmax column-wise and row-wise, respectively.

Efficient Self Attention Module as shown in Figure 4.3 (b) reduces test time to the level of models without self attention module with only about 0.02 dB PSNR loss, which generates a close approximation due to the non-linearity of softmax function. We call the DfRes module based deinterlacing network with this parallel Self Attention Module DfRes_SA.

4.2.4 Frame Reconstruction Module

We propose to employ two same reconstruction blocks in parallel (i.e., Res_Block $\times$7, right two orange box in Fig. 4.1 (a), and details are in Fig. 4.1 (b)) to reconstruct even and odd fields separately in each data batch, which can restore fields more pertinently and directionally and thus produce better performance than the model with only one reconstruction block. We call these two parallel reconstruction blocks Separate Reconstruction Module. We utilize this proposed module in all three proposed DfRes designs, which are DfRes, ΔDfRes and DfRes_SA, and In Section 4.3, we will compare deinterlacing results from these three designs.

4.3 Experimental Evaluation

4.3.1 Experimental Settings

Dataset. We used UCF101 [Soomro et al., 2012], REDS [Wang et al., 2019, Nah et al., 2019] and Vimeo-90K [Xue et al., 2019] datasets with the same separation in [Ji and Tekalp, 2021] to demonstrate the effectiveness of our method and compare it with other methods. We also used Vid4 [Sajjadi et al., 2018] dataset for testing.

Evaluation Methods. Peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) are used to quantitatively evaluate results and zooming in on details for subjective evaluation of the results.

Training Procedure. All the training settings are the same with [Ji and Tekalp, 2021] except that we train our model for 150,000 iterations using Pytorch over tesla_v100 GPUs and that in the loss function in [Ji and Tekalp, 2021] we replace

$(1-SSIM(Y^{pred}, Y^{GT}))$ with $0.1 \cdot Cb(Y^{pred}, Y^{GT})$, where $Cb(\cdot)$ is Charbonnier Loss.

4.3.2 Comparison with the State-of-the-Art methods

We compare our proposed DfRes, ΔDfRes and DfRes_SA deinterlacing methods with three state-of-the-art SR methods: EDVR [Wang et al., 2019], TDAN [Tian et al., 2020] and DUF [Jo et al., 2018]. We plug modules of these three comparison networks into our deinterlacing architecture by replacing modules in Figure 4.2 (e) with corresponding modules of the comparison networks. We do not modify these comparison methods except that we change the upsampling step to direct combination of reference field and estimated field specifically for deinterlacing as shown in Figure 4.2 (f)-(g).

The quantitative comparison results in Table 4.1 show that on three datasets, our proposed DfRes and DfRes_SA methods outperform all other methods in terms of PSNR and SSIM.

Visual comparisons in Figure 4.4, which include EDVR method without attention module (EDVR_woTSA), show that our proposed DfRes_SA, DfRes and ΔDfRes methods produce sharper frames and EDVR method plugged into our deinterlacing

Table 4.1: Comparison of PSNR (top) and SSIM (bottom) for different methods on multiple datasets. The best and the second best are marked with red and blue color respectively.

Datasets	**DfRes_SA**	**DfRes**	EDVR	TDAN	DUF
UCF101	51.92	51.48	51.23	47.74	49.75
	0.99931	0.9993	0.9992	0.998	0.999
Vimeo	44.35	43.45	42.15	40.73	40.66
	0.9949	0.994	0.991	0.9889	0.989
REDS	37.00	36.51	33.20	30.09	31.36
	0.984	0.982	0.964	0.944	0.955

Table 4.2: Comparison of regular offset estimation and proposed DfRes offset estimation on UCF101 dataset.

Methods	Regular + SA	DfRes_SA
PSNR	51.89	51.92
SSIM	0.99929	0.99931

Table 4.3: Ablation study of proposed method.

Modules	DfRes+SA	only DfRes	only SA
PSNR	51.92	51.48	50.23
SSIM	0.99931	0.99925	0.99907

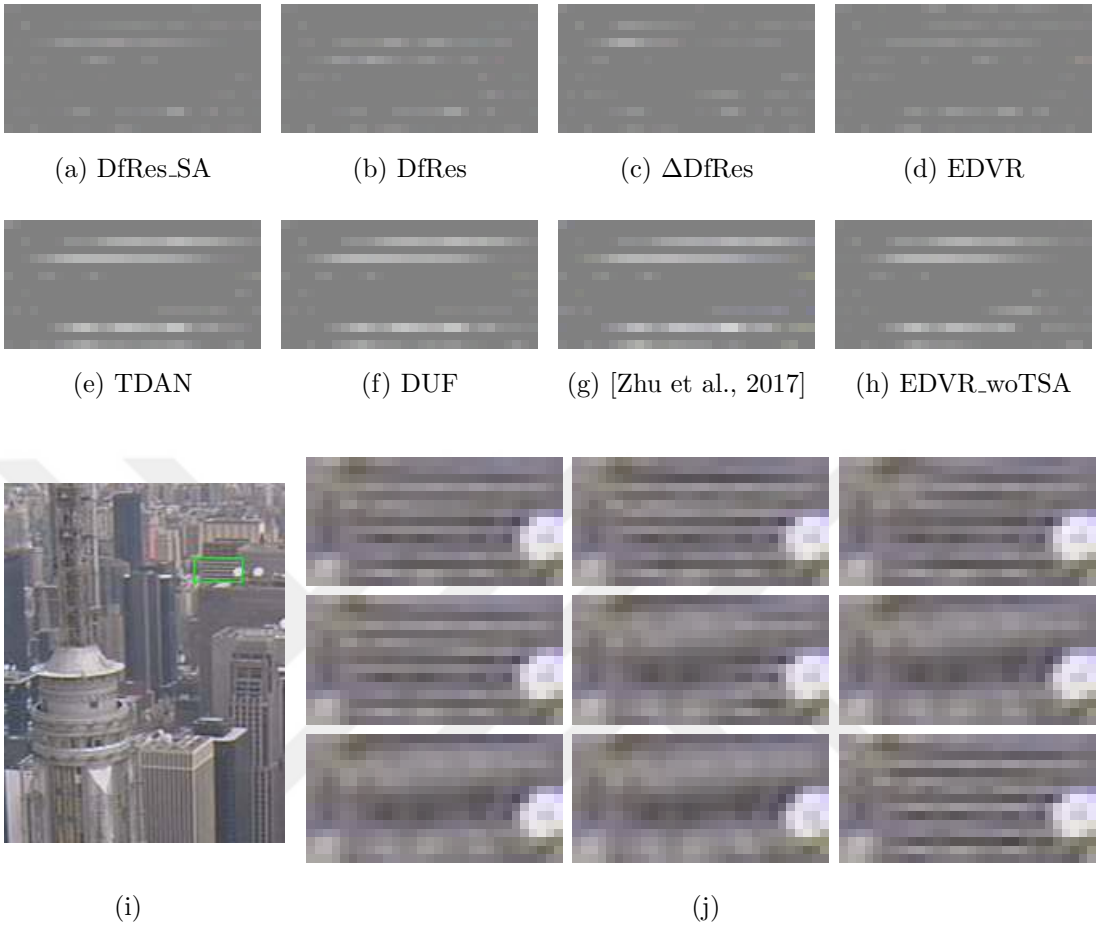
architecture can also yield comparable visual effect.

In [Dai et al., 2017, Tian et al., 2020, Wang et al., 2019], offsets are estimated by regular convolution operation on the concatenation of neighboring field and reference field. In Table 4.2, we demonstrate the improvement of proposed DfRes offset estimation over regular one.

Ablation study in Table 4.3 shows that Deformable Convolution and Self Attention module can complement each other for feature alignment to improve performance.

Visual comparisons shown in Figure 4.4 illustrate that our proposed methods produce sharper results than [Zhu et al., 2017], which is also neural network based full frame rate deinterlacing methods.

More quantitative and qualitative comparison results concerning proposed methods for same category deinterlacing methods can be found on MSU benchmark website [MSU, 2022].



(a) DfRes_SA (b) DfRes (c) ΔDfRes (d) EDVR

(e) TDAN (f) DUF (g) [Zhu et al., 2017] (h) EDVR_woTSA

(i) (j)

Figure 4.4: Visual evaluation of a deinterlaced frame from Vid4 dataset. (a)-(h) Differences between the actual and reconstructed progressive frames. (i) Progressive ground truth frame. (j) Zooming in the green box. From upper left to lower right: DfRes_SA (PSNR: 37.29), DfRes (PSNR: 37.15), ΔDfRes (PSNR: 37.25), EDVR (PSNR: 36.93), EDVRwoTSA (PSNR: 36.21), TDAN (PSNR: 35.41), DUF (PSNR: 36.08), [Zhu et al., 2017] (PSNR: 29.31), and ground truth.

4.3.3 Evaluation of Generalization Performance

Table 4.4 shows that our method in general generalizes well but the performance varies by the amount of motion in the training and test sets. Extensive experimental results reveal that both training and test dataset affect the performance of deinterlacing methods.

Table 4.4: Generalization results (Top: PSNR, Bottom: SSIM) of the proposed DfRes, ΔDfRes and DfRes_SA methods trained (Training datasets from top to bottom: UCF101, Vimeo and REDS) and tested (Test datasets from left to right: UCF101, Vimeo and REDS4) on different datasets.

DfRes_SA: Test on			**DfRes:** Test on			**ΔDfRes:** Test on		
UCF	Vimeo	REDS4	UCF	Vimeo	REDS4	UCF	Vimeo	REDS4
51.92	40.78	34.86	51.48	40.59	34.40	51.46	38.99	34.39
0.999	0.990	0.975	0.999	0.990	0.972	0.999	0.984	0.972
46.63	44.35	36.00	46.06	43.45	35.20	46.11	43.71	35.39
0.998	0.995	0.980	0.998	0.994	0.976	0.998	0.994	0.977
38.04	42.39	37.00	37.26	42.04	36.51	37.78	41.82	36.49
0.9919	0.9933	0.9835	0.9917	0.9928	0.9817	0.9915	0.9924	0.9816

4.4 Conclusion

We introduce a new network architecture for video deinterlacing with feature alignment by combining deformable convolution residual block and self attention, which can incorporate state of the art superresolution approaches for deinterlacing task. Experimental results demonstrate that the proposed network yields state-of-the-art performance and can be adopted to effectively deinterlace videos.

Chapter 5

A NEW MULTI-PICTURE ARCHITECTURE FOR LEARNED VIDEO DEINTERLACING AND DEMOSAICING WITH PARALLEL MODIFIED DEFORMABLE CONVOLUTION AND EFFICIENT TOP-K SELF-ATTENTION BLOCKS

5.1 Novelty and Advancing the State of the Art

This chapter advances the state-of-the-art in multi-picture learned video demosaicing and deinterlacing as follows:

1) We propose a common multi-picture architecture for video demosaicing and deinterlacing (since both are video upsampling problems) featuring modified deformable convolution layers and efficient residual top-k SA layers to additively combine local and global features, respectively, and separate reconstruction blocks for each channel to be upsampled.

2) We modify the DfRes blocks proposed in [Ji and Tekalp, 2022] by removing the redundant residual connection and inserting a regular convolution layer in between the two deformable convolution layers (Df). We show that the modified Deformable Convolution blocks (DfConv) yield better results than DfRes blocks.

3) We replace the SA block in [Ji and Tekalp, 2022] with novel efficient residual top-k SA block. We show that changing multiplication order can reduce the training and test time to about half compared to [Ji and Tekalp, 2022] while retaining performance, and that adding a residual connection in the top-k SA block yields a remarkable boost in the performance.

4) We demonstrate that adding the features aligned by the deformable convolu-

tion layers and self-attention layers (in parallel) instead of concatenating them as proposed in [Bello et al., 2019] yields better results.

5) We show that using separate reconstruction blocks for each picture to be reconstructed according to its particular subsampling pattern yields superior performance compared to using a single reconstruction block for all picture types.

6) Our results are the state-of-the-art for both demosaicing and deinterlacing tasks. We provide ablation studies on the value of k and also to show how much improvement comes from each of the proposed modifications, as well as extensive results to show that our architecture provides superior generalization performance over competing methods for both tasks when trained and tested on different datasets.

5.2 Proposed Deinterlacing and Demosaicing Architecture

We first present the rationale and an overview of our proposed architecture in Section 5.2.1. We introduce the main blocks of the proposed architecture: deformable convolution blocks, efficient residual top-k SA block, and separate reconstruction blocks, in Section 5.2.2 to Section 5.2.4, respectively.

5.2.1 Rationale and Overview

The rationale for a common architecture for both tasks is that they are both video upsampling problems (with different but fixed subsampling patterns), which would benefit from spatio-temporal modeling of features. Hence, a common architecture modeling both local and global spatio-temporal interactions using deformable convolutions and self attention, respectively, and separate reconstruction modules for different fields/channels should serve both tasks well.

The proposed multi-field video upsampling network, depicted in Figure 5.1(a), consists of two stages: i) feature alignment and integration stage, where local features of supporting pictures are computed and aligned with those of the current reference picture using modified deformable convolution blocks, and then integrated additively with the global features computed by efficient residual top-k self-attention layer in parallel, and ii) reconstruction stage with separate reconstruction blocks

(two for deinterlacing and three for demosaicing) to estimate missing pixels in each subsampled component.

The input to the alignment stage consists of an odd number of pictures centered

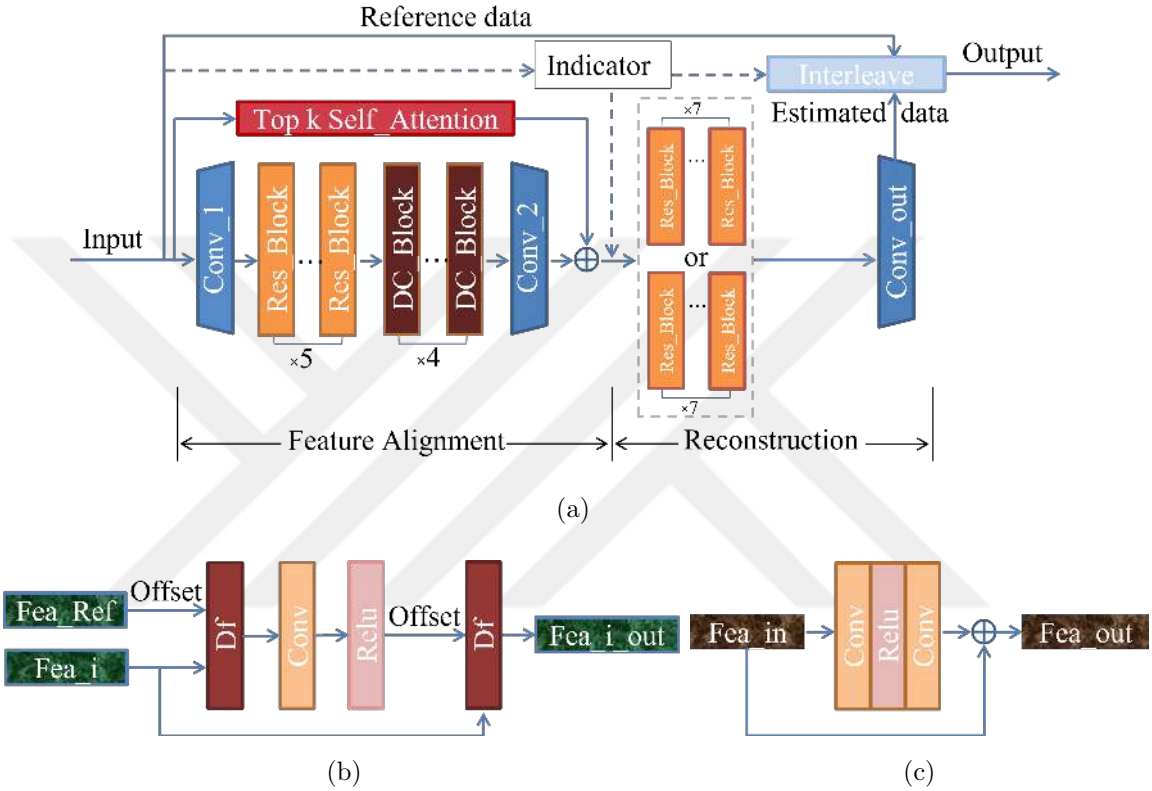


Figure 5.1: The proposed multi-picture network architecture. (a) The overall architecture, where DfConv blocks and top-k self-attention block extract and align local and global spatio-temporal features, respectively. The aligned features are integrated additively. The reconstruction stage consists of separate reconstruction blocks (two 7 × Res_Blocks for deinterlacing and three 7 × Res_Blocks for demosaicing) according to the subsampling pattern. The solid arrows are data flow and the dashed arrows are indicator (not containing data information) flow. (b) Block diagram of a DfConv block (DC_Block, brown boxes in (a)), where Fea_i denotes input features and Fea_i_out denotes the corresponding output features, which will be input to the next DfConv block. (c) Block diagram of a standard residual block (Res_Block, orange boxes in (a)).

about the reference picture. In the deinterlacing task, for example, if we were to estimate an even field N_E using 5 fields, then the reference field is N_O and the input fields would be $(N-2)_O, (N-1)_E, N_O, (N+1)_E, (N+2)_O$. Although the reconstructed fields $(N-1)_O$ and $(N-2)_E$ would be available at test time since they are in the past of the current field to be estimated, we chose not to use them in order to avoid potential error propagation. In the demosaicing task, we have successive R, G, B pictures with missing pixels as input. We use an indicator flag to indicate the type (i.e., odd/even or R/G/B) of the picture with missing lines/pixels according to the subsampling pattern.

It is well-known that deformable convolutions can achieve excellent temporal alignment of local features in video restoration and SR tasks [Tian et al., 2020, Wang et al., 2019]. It is also well-known that self attention (SA) expands the receptive field to the entire picture. However, the combination of two has been seldom used to integrate local and global features for image/video upsampling tasks. Our previous work [Ji and Tekalp, 2022] uses the standard vanilla SA, which involves all tokens into the calculations, including a number of irrelevant tokens in high resolution image/video processing that inject noise into the attention results. It has been observed that this negatively affects the training process [Wang et al., 2022a]. In this work, we employ top-k SA (kSA) block, which selects the most relevant top-k (not all) tokens from the attention feature map, to enhance the performance of feature registration and integration. We also noticed that adding a residual connection to top-k SA block provides a significant performance improvement. The 64-channel local and global feature tensors processed by the DfConv and kSA block branches in parallel are added together to form the integrated 64-channel feature tensor. This architecture is different from previous use of SA to augment ConvNets [Bello et al., 2019] in computer vision tasks, where they connect convolutional and SA layers in series and concatenate respective features. We demonstrate the effectiveness of our proposed strategy in Section 5.3.

The 64-channel aligned and integrated feature tensor is input to the reconstruction stage, where it is processed by one of the separate reconstruction blocks as

determined by the indicator flag. Each reconstruction block has 7 regular residual blocks. The last Conv_out layer maps the 64 channel feature tensor to 3 channels to generate the reconstructed pixel samples. Finally, the estimated pixel samples are interleaved with the available reference pixel samples for deinterlacing task or with the other two estimated channels for demosaicing task to reconstruct the output frame.

5.2.2 Modified Deformable Convolution Blocks

Each input picture sequence with missing data is first mapped to a 64-channel feature tensor via the convolution layer Conv_1 and next processed by five regular residual blocks to refine the features. We employ deformable convolution (DfConv) blocks (shown in Figure 5.1(a) brown boxes as DC_Block), to align spatio-temporal features due to their flexible receptive fields. The processing chain is depicted in Figure 5.1(a), where we use 5× standard Res_Blocks for feature extraction and 4× DfConv blocks for feature alignment. In the following, we propose a new DfConv block structure that yields better performance than using the standard deformable convolution blocks [Tian et al., 2020] and deformable convolution residual blocks (DfRes) [Ji and Tekalp, 2022] for feature alignment.

In the literature, offsets for deformable convolution are learned by applying regular convolutions directly to concatenated input feature maps [Dai et al., 2017, Tian et al., 2020, Cheng and Chen, 2021, Ying et al., 2020]. Clearly, the quality of learned offsets directly affects the performance of deformable convolution. We found that learning the offsets by using deformable (rather than standard) convolutions with a refining convolution layer (different from DfRes [Ji and Tekalp, 2021] and DfRes+SA [Ji and Tekalp, 2022] blocks) yields better performance.

Our proposed Deformable Convolution (DfConv) block, shown in Figure 5.1(b), employs two deformable convolution layers (Df) with a regular convolution layer between them, in order to align features of supporting pictures with those of the reference picture. In the first Df layer, we treat the reference picture features as offsets and perform Df on features of each supporting picture separately with these offsets.

The results are refined through a regular convolution layer and a ReLU activation, and then fed into the second Df layer as offsets. There are a total of 4 DfConv blocks, where each time, we align one of the supporting pictures to the reference picture. Here, the difference from our prior work [Ji and Tekalp, 2022] is that, firstly, we remove the redundant residual connection which we observe degrades the performance, and secondly, we add a regular convolution layer in between two Df layers, which refines the offsets. The effectiveness of these two modifications is demonstrated in the top row of Table 5.5.

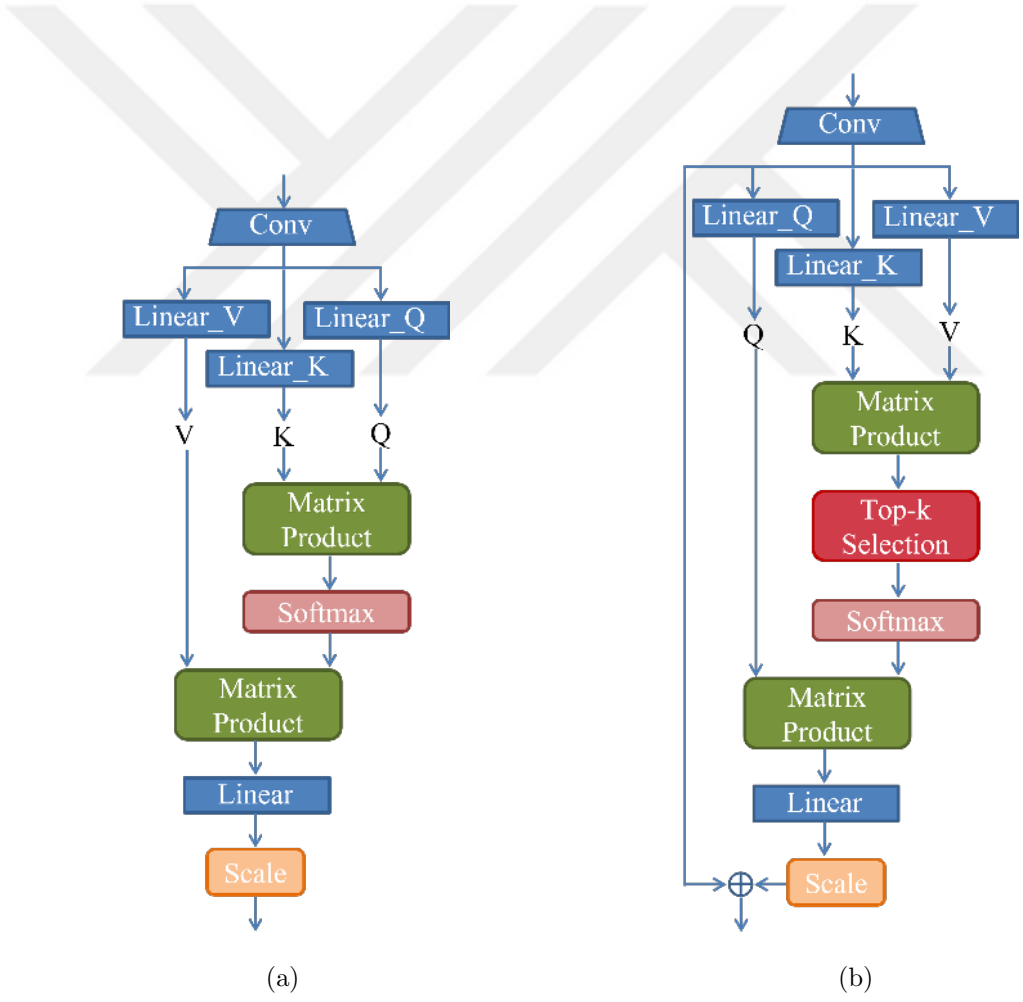


Figure 5.2: Self-attention (SA) blocks. (a) Block diagram of the standard SA block. (b) Block diagram of the proposed efficient residual top-k SA block, where top k selection operator is defined by Equation 5.4.

5.2.3 Efficient Residual Top-k Self-Attention Block

It is desirable to exploit the global receptive field of the SA module [Vaswani et al., 2017] for global feature processing. However, the vanilla implementation of SA, shown in Figure 5.2(a), employs all tokens, which injects noise into the training process in high resolution image/video processing tasks by including irrelevant tokens in the computations, thus limiting the performance and increasing the computation burden. We mitigate this problem by implementing an efficient residual top-k SA (EkSA) implementation, shown in Figure 5.2(b), which is a modified version of the approach proposed by [Wang et al., 2022a].

In the standard SA implementation, each picture sequence with n pixels is mapped to 64 feature channels via regular convolution layers (Conv in Figure 5.2(a)) for feature extraction. Then, queries (Q $\in \mathbb{R}^{n\times 64}$), keys (K $\in \mathbb{R}^{n\times 64}$) and values (V $\in \mathbb{R}^{n\times 64}$) are separately computed by linear mapping layers Linear_Q, Linear_K and Linear_V, where the numbers of output channels of Linear_Q, Linear_K and Linear_V (shown by blue rectangles in Figure 5.2(a)) are the same as those of the input channels respectively. After obtaining Q, K and V matrices, we multiply Q and K transpose to form the attention weight map, and normalize the result by the softmax operation $\rho(Q \times K^T)$. Then, we multiply the normalized softmax output and V to obtain an initial attention result. Finally, the initial attention result is mapped through a linear layer and scaled to yield the final attention output. Hence, the standard SA operation can be summarized by:

$$SA(Q, K, V) = L[\rho(Q \times K^T) \times V] \cdot Scale, \tag{5.1}$$

where T denotes matrix transpose, $\times$ is matrix product, L represents linear mapping and $Scale$ is a learnable scale factor. The softmax normalization

$$\rho(Y) = \sigma_{row}(Y), \tag{5.2}$$

is applied row-wise.

Multiple inputs bring in replicated and redundant information since many elements are similar to each other across frames. In order to eliminate irrelevant redun-

dant tokens and accelerate training, we instead implement the top-k SA model [Wang et al., 2022a] given by:

$$kSA(Q, K, V) = L[\rho(T_k[Q \times K^T]) \times V] \cdot Scale, \tag{5.3}$$

where

$$[T_k(A)]_{ij} = \begin{cases} A_{ij}, & if \quad A_{ij} \in top-k(row\ j), \\ -\infty, & \text{otherwise.} \end{cases} \tag{5.4}$$

is utilized to select the top k elements (tokens) row-wise. More details can be found in [Wang et al., 2022a]. The selection of k relevant tokens, shown by the red box in Figure 5.2(b), follows computation of the attention weight map.

While removing irrelevant tokens improves the performance of the model successfully, this simple implementation of top-k self-attention is not computationally efficient due to the fact that we only set the irrelevant tokens to zeros instead of removing them thoroughly as [Bolya et al., 2022] and [Li et al., 2023b], which decrease the numbers of tokens by merging tokens or reducing the length of sequences and to some extent weaken the global dependencies. In order to improve the computational efficiency significantly, inspired by [Shen et al., 2021] and [Xia et al., 2022], we exchange multiplication order by first multiplying V^T and K to generate the attention weight map. Thus, the attention weight map matrix is reduced to 64 $\times$ 64 from n $\times$ n. Hence, the efficient top-k SA (EkSA) can be expressed by:

$$EkSA(Q, K, V) = L[Q \times \rho(T_k[V^T \times K])] \cdot Scale, \tag{5.5}$$

The efficiency and effectiveness of this top-k SA design, shown in Figure 5.2(b), are demonstrated in Table 5.2 and Table 5.6, respectively.

Different from [Wang et al., 2022a], we found that adding a residual connection between initial features and the scaled output, shown by the plus sign in Figure 5.2(b), significantly improves the overall test performance. We demonstrate the effectiveness of the residual connection in the middle row of Table 5.5.

5.2.4 Separate Reconstruction Blocks

We experimentally determine that using separate reconstruction blocks, called the SepRecon architecture, shown in Figure 5.1(a), brings benefits in the case of multiple known and fixed degradation models. In SepRecon block, each Res_Block is depicted in Figure 5.1(c). Specifically, we employ two reconstruction modules in parallel with 7 $\times$ standard Res_Blocks in each branch (shown by two rows of orange blocks on the right in Figure 5.1(a)) for the deinterlacing task to separately reconstruct the missing even and odd fields, and three reconstruction modules in parallel with 7 $\times$ standard Res_Blocks in each branch for the demosaicing task to reconstruct each of R, G, and B channels separately. The SepRecon architecture restores the missing lines and pixel values directionally according to the specific degradation model; thus, producing superior performance compared to a single reconstruction block for both even and odd fields or three color channels. The advantage of the SepRecon architecture is demonstrated by an ablation study in Section 5.3.

5.3 Experimental Evaluation

We start by discussing the experimental procedures, such as datasets, evaluation methods, and training details in Section 5.3.1. Then, we present comparison of our proposed DfConv+EkSA models with the state-of-the-art both quantitatively in terms of PSNR and SSIM metrics and qualitatively by visual quality in Section 5.3.2. Next, we present ablation studies in Section 5.3.3 to demonstrate the improvements obtained by using each of the proposed modifications such as DfConv blocks, the efficient top-k SA module, etc.. Finally, we show that the generalization performance of our models is superior to competing methods when trained and tested on different datasets via cross-validation in Section 5.3.4.

5.3.1 Experimental Procedures

Datasets

we selected widely and publicly used datasets which cover different resolutions and amounts of motions to demonstrate our proposed architecture and to compare with other methods. We used combinations of UCF101 [Soomro et al., 2012], REDS [Nah et al., 2019], Vimeo-90K [Xue et al., 2019] and Vid4 [Sajjadi et al., 2018] sets for training and testing.

The UCF101 video dataset (240 × 320) was separated into training and test sets according to [Soomro et al., 2012] and we tested only on ApplyEyeMakeup test class.

In REDS video dataset (720 × 1280), clip00, clip11, clip15 and clip20 (called REDS4) were used for testing and the remaining clips were used for training [Wang et al., 2019].

In Vimeo-90K video dataset (256 × 448), we utilized the training and test septuplets as stated in [Xue et al., 2019].

There is no training set for the Vid4 [Sajjadi et al., 2018] dataset (576 × 704, 480 × 704), and we used Vid4 only for testing. We tested all compared methods with their best publicly available trained models for both deinterlacing and demosaicing tasks on Vid4.

In addition, two well-known video clips, *SpongeBob* (480 × 720) and *Mr. Bean's Holiday* (1920 × 1080), were used as real-world data to validate the effectiveness of our models.

Interlaced videos are generated by splitting ground-truth progressive videos into odd and even fields and retaining only odd or even fields from alternating frames.

Mosaic video sequences are obtained by keeping pixels in R, G, and B channels according to the Bayer pattern and setting other pixel values to zeros.

Both these acquired interlaced and mosaic sequences are directly input into corresponding proposed models and the immediate scaled outputs are straightforwardly compared without any other preprocessing or postprocessing steps.

Evaluation Methods

We use peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) [Wang et al., 2004] for quantitative evaluation of results. We also provide visuals zooming in on details for subjective evaluation of the results.

Training Details

We input five adjacent fields, $(N-2)_O$, $(N-1)_E$, N_O, $(N+1)_E$ and $(N+2)_O$ to estimate field N_E, or $(N-2)_E$, $(N-1)_O$, N_E, $(N+1)_O$ and $(N+2)_E$ to estimate field N_O for the deinterlacing task, and mosaiced R, G, B arrays for frames $(N-2)$, $(N-1)$, N, $(N+1)$ and $(N+2)$ to restore frame N for the demosaicing task.

We use the loss function:

$$\begin{aligned} L =& MSE(Y^{pred}, Y^{GT}) + 0.1 \cdot Char(Y^{pred}, Y^{GT}) \\ &+ \lambda_{TV} \cdot TV(Y^{pred}) \end{aligned} \tag{5.6}$$

where Y^{pred} and Y^{GT} are estimated and ground truth frames, respectively, $MSE(\cdot)$ is the mean square error, $Char(\cdot)$ denotes the Charbonnier loss [Wang et al., 2019], $TV(\cdot)$ denotes the total variation regularizer [Zhu et al., 2017], and λ_{TV} is a scale factor that is set to 2.0×10^{-3} in all experiments.

We use the Adam optimizer with the initial learning rate 4×10^{-4} and Cosine Annealing scheme, the same as in [Ji and Tekalp, 2022]. We train all models for 150,000 iterations with mini-batch size equal to 32 for deinterlacing and 24 for demosaicing tasks, and the input patch size is set to 64×80.

5.3.2 Comparison with the State-of-the-Art

Selected models

We compare our proposed DfConv+EkSA model with DfRes [Ji and Tekalp, 2021], DfRes+SA [Ji and Tekalp, 2022], DeT [Song et al., 2023], Gao *et al.* [Gao et al., 2023] and RTDVD [Zhu et al., 2017] for the deinterlacing task, and with DPIR [Zhang et al., 2022a], BJDD [Sharif et al., 2021], Kokkinos [Kokkinos and Lefkimmiatis, 2019], NTSDCN [Wang et al., 2020] and Ehret [Ehret et al., 2019] for the demosaicing

Table 5.1: Number of parameters for different models.

Model	**DfRes**	**DfRes+SA**	**DfConv+EkSA**
Deinterlacing	2,756,739	2,784,276	2,943,235
Demosaicing	-	-	3,460,227

Table 5.2: Test Time (sec.) for one output frame with different models (Top: Deinterlacing on UCF101 dataset, Bottom: Demosaicing on Vid4 dataset).

DfConv+EkSA	DfConv+kSA	DfRes+SA	DfRes	DeT	RTDVD
0.08	0.32	0.17	0.08	0.14	0.09

DfConv+EkSA	DPIR	Kokkinos	NTSDCN	Ehret	BJDD
0.65	0.50	5.22	0.97	0.14	0.01

task, where RTDVD [Zhu et al., 2017], DPIR and NTSDCN are designed for single picture input, and comparison with Gao *et al.* [Gao et al., 2023] is only literally stated in Section 5.3.4 due to unavailability of codes. We also provide comparisons with the classical video superresolution architectures (EDVR [Wang et al., 2019], TDAN [Tian et al., 2020], DUF [Jo et al., 2018]) trained on our deinterlacing and demosaicing video datasets. In line with the availability of training and test codes of different compared methods, we retrain DfRes, DfRes+SA, EDVR, TDAN and DUF with the same training datasets, and adopt the best available pre-trained models of other methods for validation.

As a measure of model complexity, the numbers of parameters in the compared models are very close to each other as shown in Table 5.1. Table 5.2 shows the computational speed (test run time) competitiveness of our DfConv+EkSA model.

Table 5.3: Comparison of models for the Video deinterlacing task on multiple datasets (Top: PSNR, Bottom: SSIM).

Datasets \ Methods	**Ours**	DfRes+SA	DfRes	DeT	RTDVD
UCF101	52.34	51.92	51.48	43.46	36.96
	0.9994	0.9993	0.9993	0.9951	0.9859
Vimeo-90K	44.96	44.35	43.45	42.22	38.81
	0.9956	0.9949	0.9935	0.9850	0.9740
REDS	38.04	37.00	36.51	34.08	33.45
	0.9870	0.9835	0.9817	0.9513	0.9455

Table 5.4: Comparison of models for the Video demosaicing task on Vid4 dataset and on REDS4 dataset (Top: PSNR, Bottom: SSIM).

Test on:	**Ours**	DPIR	BJDD	Kokkinos	NTSDCN	Ehret
Vid4	39.44	37.91	37.55	34.05	32.77	31.59
	0.9897	0.9816	0.9819	0.9834	0.9868	0.9522
REDS4	48.04	43.89	44.54	32.37	36.51	-
	0.9982	0.9895	0.9918	0.9534	0.9975	-

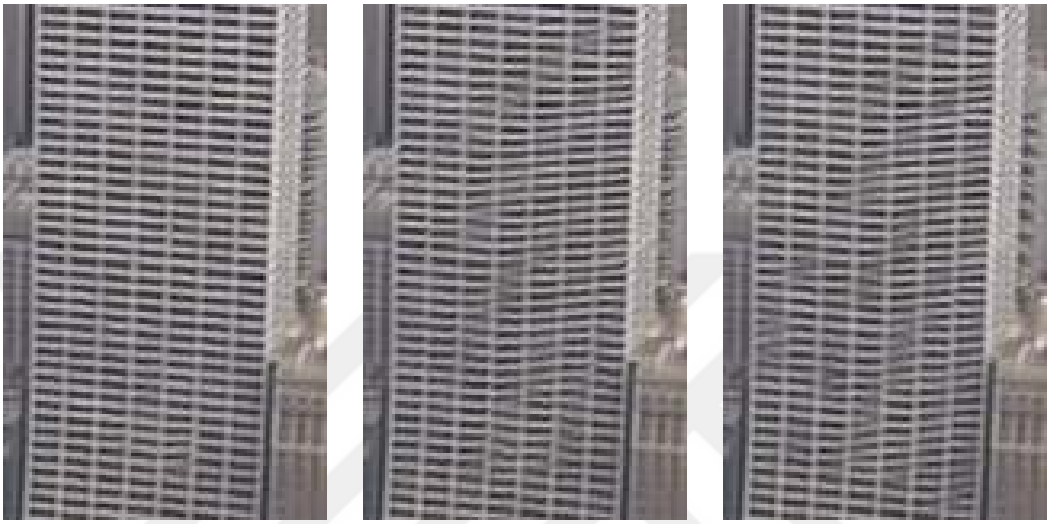
Quantitative Performance Comparison

Quantitative comparison results for the deinterlacing and demosaicing tasks are provided in Table 5.3 and Table 5.4, respectively.

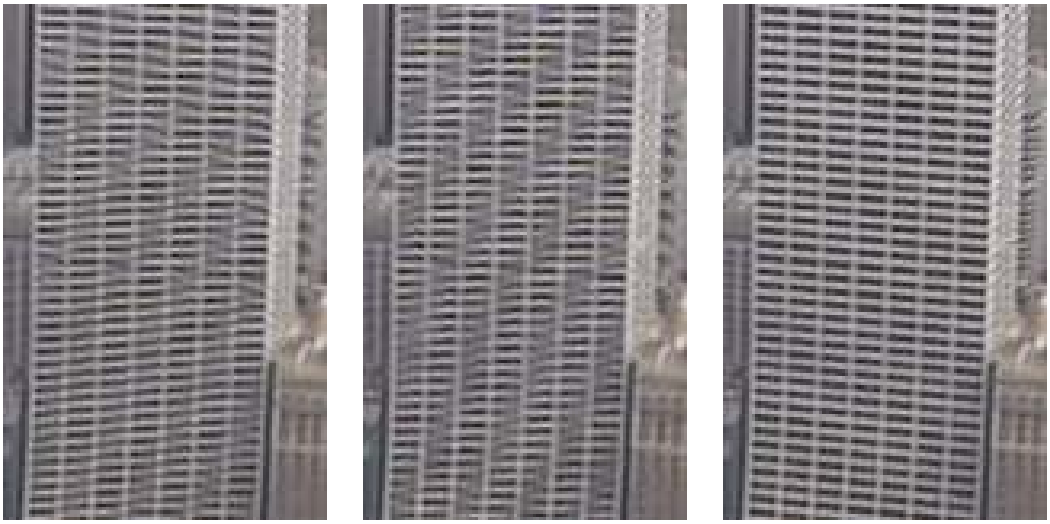
Table 5.3 shows that our DfConv+EkSA model with the new offset estimation and efficient residual top-k self attention modules trained for the deinterlacing task is clearly superior to the competing methods on multiple datasets. Hence, our proposed DfConv+EkSA model is the state-of-the-art in the deinterlacing task.

Table 5.4 shows that our DfConv+EkSA model trained for the demosaicing task

also outperforms the state-of-the-art demosaicing methods on both Vid4 and REDS4 datasets. Hence, it is also the state-of-the-art in the demosaicing task by a large margin.

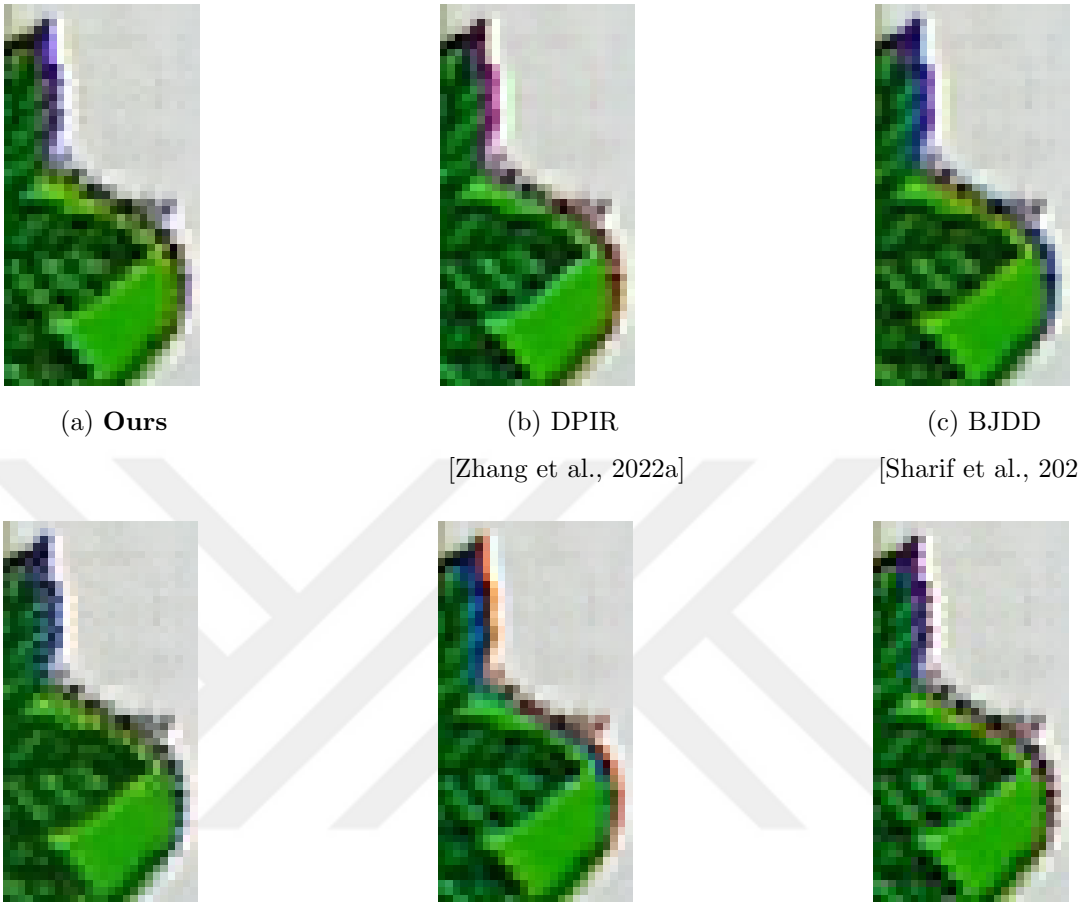


(a) **Ours** (b) DfRes+SA [Ji and Tekalp, 2022] (c) DfRes [Ji and Tekalp, 2021]



(d) DeT [Song et al., 2023] (e) Ref [Zhu et al., 2017] (f) GT

Figure 5.3: (a)-(e) are deinterlaced results on a frame (Vid4) with different models; (f) is the ground truth frame.

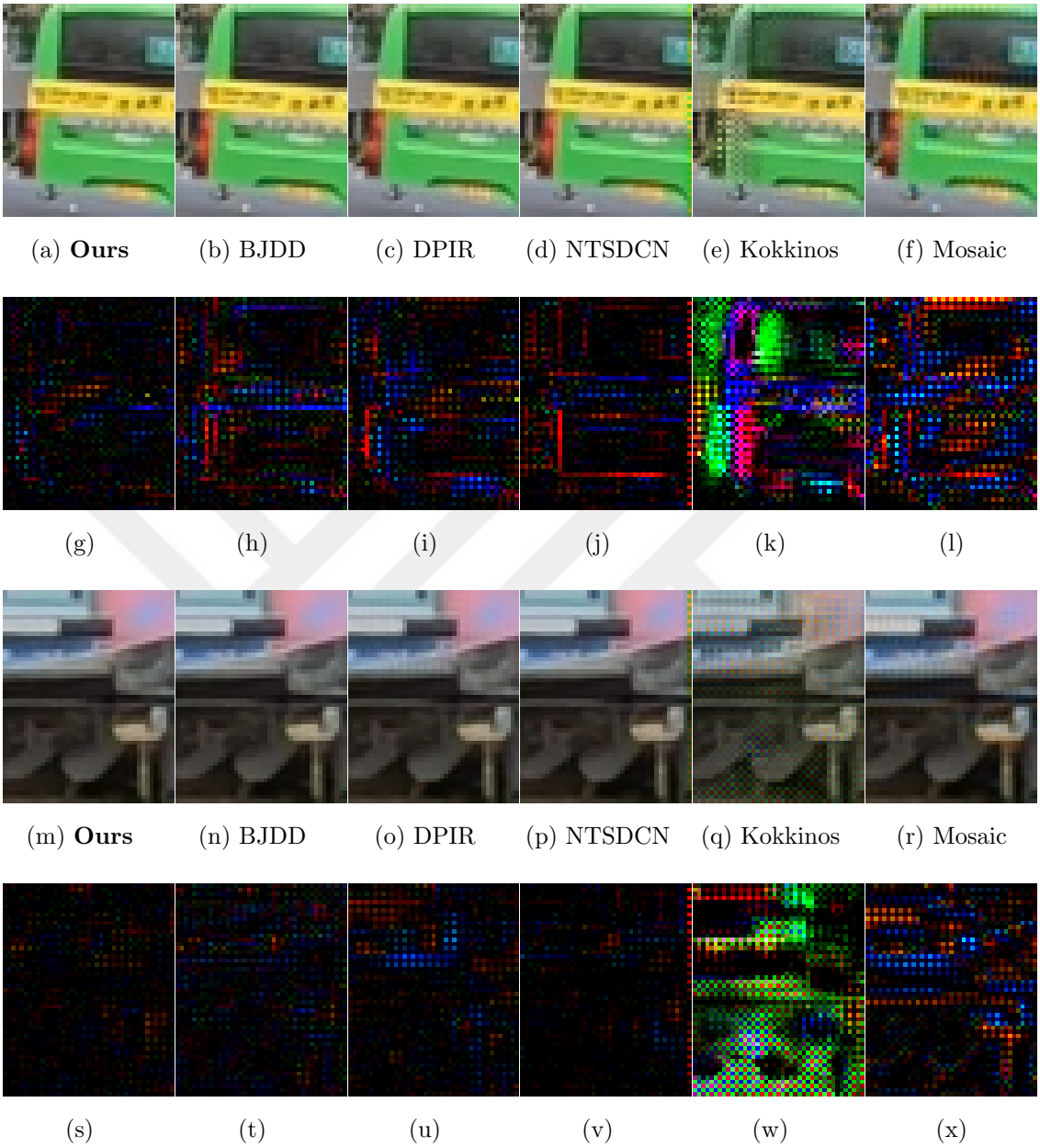


(a) **Ours** (b) DPIR [Zhang et al., 2022a] (c) BJDD [Sharif et al., 2021]

(d) Kokkinos [Kokkinos and Lefkimmiatis, 2019] (e) NTSDCN [Wang et al., 2020] (f) GT

(g) **Ours** (h) DPIR (i) BJDD (j) Kokkinos (k) NTSDCN (l) GT (m)

Figure 5.4: (a)-(e) and (g)-(k) are crops from a demosaiced frame (Vid4) with different models; (f) and (l) are ground truth crops. The crops (a)-(f) and (g)-(l) correspond to the yellow and red boxes marked on (m).



(a) **Ours** (b) BJDD (c) DPIR (d) NTSDCN (e) Kokkinos (f) Mosaic

(g) (h) (i) (j) (k) (l)

(m) **Ours** (n) BJDD (o) DPIR (p) NTSDCN (q) Kokkinos (r) Mosaic

(s) (t) (u) (v) (w) (x)

Figure 5.5: More demosaicing results on REDS4 data. (f) and (r) are mosaic frames. (a)-(e) and (m)-(q) are demosaiced results. (g)-(l) and (s)-(x) are 10 times pixel differences between (a)-(f) and (m)-(r), and GT, respectively.

Visual Performance Comparison

It is generally accepted that PSNR/SSIM results do not always correlate well with visual evaluations. To this effect, we also provide visual comparisons. Figure 5.3

(a) **Ours** (b) DeT (c) DfRes+SA (d) DfRes (e) RTDVD (f) Interlaced

(g) **Ours** (h) DeT (i) DfRes+SA (j) DfRes (k) RTDVD (l) Interlaced

Figure 5.6: Deinterlacing results on real-world data. (f) and (l) are interlaced frames, where even and odd fields are weaved. (a)-(e) and (g)-(k) are deinterlaced results for the bottom field of the corresponding interlaced frames.

show deinterlaced frames with different models. Our method produces almost no stripe distortions compared to other two methods.

Figure 5.4 (a)-(e) and (g)-(k) show demosaiced frames with different models. Close inspection of (a)-(e) show that our DfConv+EkSA model produces less color distortion and (g)-(k) show that our method generates sharper images compared to other methods which look over-smoothed due to denoising.

Figure 5.5 shows more visual results for the demosaicing task on REDS4 dataset. We can see that (c) and (o) bring in slight vertical strip noise when zoomed in on plate and white area; (d) and (p) produce color distortion while dealing with edge information; while (e) and (q) fail to demosaic frames successfully. Due to the pixel level operation in demosaicing, it is hard to visually tell the differences between (a) and (b), or (m) and (n). However, with the help of difference images from (g) to (l) and (s) to (x), we can easily find that our method produces less differences from ground truth frames as shown in (g) and (s). In overall contrast, our proposed method demosaics input frames well avoiding generating strip noise or

color distortion as shown in (a), (g), (m) and (s).

Figure 5.6 shows visual results on two real-world video clips. Our proposed method (a) can relieve severe comb-teeth caused by interlacing (f) compared with other methods through (b)-(e). In addition, we can see that (i) and (j) effectively preserve the letters but generate artificial patterns with the horizontal lines; (k) deinterlaces the lines well but yields white dot noise on the top arcs of letters s, o and d, and fades top lines of letters h, l and d. Instead, our proposed model can deinterlace both the letters and the horizontal lines properly without producing additional noise. DeT [Song et al., 2023] deinterlaces (h) well with indistinguishable visual effect compared to ours (g). This is because of the simple texture pattern of (h) in contrast to more complex texture patterns in (b) and Fig. 5.3 (d) where DeT introduces noises.

In summary, visual results demonstrate that our proposed model keeps genuine details well without introducing extra noise or artificial patterns in a variety of video clips.

5.3.3 Ablation Studies

We conducted the following ablation studies to demonstrate the effect of each component of the proposed architecture shown in Figure 5.1 on the performance of our model: i) the structure of new DfConv blocks, ii) the use of top-k SA vs. SA block both with or without residual connection, iii) the effect of different values of k on the performance of top-k SA, iv) combining features aligned by DfRes and SA through addition vs. concatenation, v) the structure of the reconstruction module. The results of these ablations, summarized in Table 5.5 and Table 5.6 on the UCF101 video dataset, are discussed in detail in the following:

i) The structure of the new DfConv blocks, i.e, estimating offsets by using DfConv vs. DfRes [Ji and Tekalp, 2022] and using two Df layers with a convolution layer in between vs. only two Df layers for feature alignment. In our prior work [Ji and Tekalp, 2022], we show that DfRes offset estimation, which applies deformable convolution residual block only to the reference picture, yields performance im-

Table 5.5: Ablation studies for the deinterlacing task: Top row: Comparison of DfConv vs. DfRes blocks, and integration of features by addition + vs. concatenation |. Middle row: Comparison of kSA vs. SA module for $k = 100$ with and without residual connection. Bottom row: Comparison of separate reconstruction modules with 7 Res_Blocks each vs. single module with 0, 7, 14 Res_Blocks.

Alignment/Integration	DfConv+SA	Df+SA	DfRes+SA	DfRes|SA
PSNR	52.22	52.16	51.92	51.80
SSIM	0.99934	0.99933	0.99931	0.99931
Self-Attention Block	kSA $w\backslash$ res	kSA $wo\backslash$ res	SA $w\backslash$ res	SA $wo\backslash$ res
PSNR	52.32	51.52	52.22	51.56
SSIM	0.99936	0.99924	0.99934	0.99922
Reconstruction Stage	SepRecon	0	7	14
PSNR	51.92	50.98	51.37	51.23
SSIM	0.99931	0.99921	0.99923	0.99917

Table 5.6: Ablation study on the value of k for top-k self attention.

k values for DfConv+kSA	**50**	100	150	all
PSNR	52.34	52.32	52.30	52.22
SSIM	0.99936	0.99936	0.99935	0.99934
k values for DfConv+EkSA	32	**50**	64	-
PSNR	52.32	52.34	52.28	-
SSIM	0.99936	0.99936	0.99936	-

provement compared to the offset estimation used in TDAN [Tian et al., 2020] and EDVR [Wang et al., 2019], which applies regular convolution layers to concatenation of neighboring and reference pictures. In this paper, we show, in the top part of Table 5.5, that about 0.3 dB further PSNR improvement can be obtained by our new

DfConv offset estimation method compared to DfRes offset estimation. Moreover, the top row of Table 5.5 also shows that Df block without a residual connection (Df+SA) yields better performance than DfRes block with a residual connection (DfRes+SA), and that Df layers with a convolution layer in between (the proposed DfConv+SA) produces the best performance.

ii) The use of kSA vs. SA block both with or without residual connection: Then, in the middle row of Table 5.5, we compare top-k Self Attention (kSA, where k is set equal to 100) and vanilla SA modules, and both with residual connection and without residual connection. Results indicate that a residual connection in both kSA and SA modules can significantly improve performance and kSA is better than SA.

iii) The effect of different values of k on the performance of top-k SA: Table 5.6 shows $k = 50$ is the best value for both DfConv+kSA model which utilizes the normal multiplication order as depicted in Equation 5.3, and DfConv+EkSA model which changes the multiplication order as given in Equation 5.5. Moreover, this table also shows that changing multiplication order does not affect the performance when k is set equal to 50 for top-k SA, which is different from standard SA [Ji and Tekalp, 2022].

iv) Combining features aligned by DfRes and SA through addition vs. concatenation: The paper [Ji and Tekalp, 2022] also shows that the features computed by the SA module complements the features aligned by the deformable convolution blocks to yield significantly improved performance. While SA alone does not provide good performance, it yields significantly improved performance when combined with the features computed by the deformable convolution blocks. We show, at the right part of the top row in Table 5.5 that combining deformable convolution blocks and SA module by addition (DfRes+SA) is superior to concatenation (DfRes|SA).

v) The structure of the reconstruction module: We investigate the improvement that comes from using the SepRecon module instead of a single reconstruction module with 7 or 14 residual blocks or no residual blocks at all. The bottom row of Table 5.5 shows that there is more than 0.5 dB gain in using the SepRecon module

with 7 residual blocks in each branch.

Table 5.7: Generalization performance of our DfConv+EkSA model trained and tested on different datasets for deinterlacing and demosaicing tasks (Top: PSNR, Bottom: SSIM).

Train on	**Deinterlacing:** Test on			**Demosaicing:** Test on		
	UCF101	Vimeo-90K	REDS4	UCF101	Vimeo-90K	REDS4
UCF101	52.34	41.49	35.40	52.64	43.91	44.50
	0.9994	0.9917	0.9771	0.9994	0.9944	0.9967
Vimeo-90K	47.57	44.96	36.67	46.99	48.01	45.76
	0.9985	0.9956	0.9824	0.9980	0.9975	0.9974
REDS	38.71	43.00	38.04	44.64	43.69	48.04
	0.9932	0.9941	0.9870	0.9968	0.9942	0.9982
YOUKU	48.73	40.14	34.38	50.18	44.06	44.61
	0.9989	0.9890	0.9725	0.9991	0.9948	0.9968

5.3.4 Evaluation of Generalization Performance

In order to evaluate the generalization ability as to different datasets of our architecture, we train different models on the training sets UCF101, REDS, Vimeo-90k, and YOUKU [Youku, 2019], and cross-test them on other testsets. Table 5.7 shows that our models generalize well to different testsets but their performance may vary according to the amount of motion present in the training and test sets.

We also trained super-resolution based methods adapted for deinterlacing according to the comparison strategy in [Ji and Tekalp, 2022] on three different datasets and only tested on the Vid4 dataset [Sajjadi et al., 2018] to evaluate the best training dataset (out of three) for the deinterlacing task. Table 5.8 shows that the PSNR and SSIM generalization performance of all models (except one) are the best when they are trained on the Vimeo dataset. This table also demonstrates the high train-

Table 5.8: Comparative generalization results for the deinterlacing task tested on Vid4 dataset (Top: PSNR, Bottom: SSIM).

Train on	**Ours**	DfRes_SA	DfRes	ΔDfRes	EDVR	TDAN	DUF
UCF101	34.61	34.17	33.62	33.23	33.35	32.27	32.43
	0.9790	0.9771	0.9751	0.9712	0.9738	0.9687	0.9691
Vimeo-90k	34.85	34.49	33.60	33.93	33.69	33.32	33.46
	0.9795	0.979	0.9753	0.9760	0.9756	0.974	0.9748
REDS	34.71	34.12	33.58	33.56	30.39	26.47	31.44
	0.9785	0.9777	0.9756	0.9758	0.9525	0.9148	0.9650

ing stability (less fluctuation of PSNR and SSIM) of our model on different training datasets compared to other methods on these three training datasets. Moreover, our method has an overall performance superiority over all other methods.

It is worthy to note that the SSIM results tested on Vimeo dataset trained on Vimeo and REDS datasets shown in Table 5.7 are higher than those tested on Vimeo dataset in the second table of the state of the art deinterlacing paper [Gao et al., 2023]. Moreover, the SSIM results tested on Vid4 dataset of our proposed methods shown in Table 5.8 are also higher than those tested on Vid4 dataset in paper [Gao et al., 2023].

Extensive experimental results reveal that both training and test datasets affect the performance of models for both tasks, i.e., training the same network on different training datasets produces different performances on the same testset, and a model trained on a single training dataset produces different performances on different testsets.

5.4 Conclusion

We introduce a new learned model, called DfConv+EkSA, that achieves the state-of-the-art performance in both video deinterlacing and demosaicing tasks. The pro-

posed design consists of two main stages: namely, feature alignment and integration stage and separate reconstruction stage, each containing novelties that contribute to the performance improvements as demonstrated by the ablation studies. The feature alignment and integration stage combines local spatio-temporal features aligned by novel modified deformable convolution (DfConv) blocks and global features processed by a novel efficient top-k self-attention (EkSA) block additively, in order to benefit from both local and non-local video features. In the reconstruction stage, we use separate reconstruction modules (7 blocks each) for different type of pictures, which outperforms using a single reconstruction module with 7 or 14 blocks. The model design choices we made are justified by ablation studies.

Extensive experimental results demonstrate that the proposed model with the proposed design novelties yields the state-of-the-art performance and generalization ability (across different datasets) for both tasks. This work demonstrates that the proposed learned demosaicing and deinterlacing models should be an integral part of consumer video production and processing pipelines.

Chapter 6

CONCLUSION

In this thesis, we progressively proposed a multi-picture learned architecture with parallel Deformable Convolution blocks and top- k Self-Attention block for both video deinterlacing and video demosaicing tasks.

We first introduced the fundamentals background of our work, including a succinct overview of deep learning approaches to show the brief progression of deformable convolution evolved from a basic mathematical convolution operation, and the basic concepts of interlacing to deinterlacing and mosaicing to demosaicing. We, then, reviewed the literature background of deinterlacing and demosaicing to have a general picture of the current development of both tasks, as well as related works of the deformable convolution and self attention.

Within such a context, we proposed for the first time an alignment based deinterlacing architecture by designing a Deformable Convolution Residual (DfRes) block to align each neighboring field to the reference field in the feature domain. Experimental results demonstrated the superiority of our proposed alignment based network over non-alignment based architecture.

Upon this initial deinterlacing architecture, we added self-attention mechanism in parallel to the DfRes block in order to strength the global information capturing capability of our deinterlacing architecture. In addition, we utilized the separate reconstruction module to reconstruct the even and odd fields separately according to the parity of the reference field. Experimental results showed the effectiveness of our improved architecture over competing learned deinterlacing methods and the overall performance was significantly improved over that of the initial deinterlacing architecture.

Finally, we presented a common learned multi-picture architecture for both video

deinterlacing and video demosaicing tasks taking advantage of the fixed degradation pattern of both upsampling video tasks. This new architecture, named DfConv+EkSA, consists of novel modified deformable convolution (DfConv) blocks to align spatio-temporal features and novel efficient top-k self-attention (EkSA) block to process global features, in order to enable the architecture to additively benefit from both local and non-local video dependencies. To reconstruct different channels of both tasks, we used separate reconstruction modules (SepRecon, 7 residual blocks each) to restore each channel with one SepRecon module from high-level features. Thus, two SepRecon modules are required for deinterlacing task and three for demosaicing task, effectiveness of which were justified by ablation studies.

Extensive experimental results demonstrate that the proposed architecture with the proposed design novelties yields the state-of-the-art performance and generalization ability (across different datasets) for both tasks.

The future work can be done using generative networks conditioned on the given information for both deinterlacing and demosaicing tasks based on the fact that interlaced frames have half known pixels and mosaic frames have one third known pixels.

BIBLIOGRAPHY

[Agirman and Tasdemir, 2022] Agirman, A. K. and Tasdemir, K. (2022). Blstm based night-time wildfire detection from video. *PLoS one*, 17(6):e0269161.

[Akyüz et al., 2020] Akyüz, A. O. et al. (2020). Deep joint deinterlacing and denoising for single shot dual-iso hdr reconstruction. *IEEE Trans. Image Process.*, 29:7511–7524.

[Aldausari et al., 2022] Aldausari, N., Sowmya, A., Marcus, N., and Mohammadi, G. (2022). Video generative adversarial networks: a review. *ACM Computing Surveys (CSUR)*, 55(2):1–25.

[Alom et al., 2019] Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A., and Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and architectures. *electronics*, 8(3):292.

[Basiri et al., 2021] Basiri, M. E., Nemati, S., Abdar, M., Cambria, E., and Acharya, U. R. (2021). Abcdm: An attention-based bidirectional cnn-rnn deep model for sentiment analysis. *Future Generation Computer Systems*, 115:279–294.

[Baytas et al., 2017] Baytas, I. M., Xiao, C., Zhang, X., Wang, F., Jain, A. K., and Zhou, J. (2017). Patient subtyping via time-aware lstm networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 65–74.

[Bello et al., 2019] Bello, I., Zoph, B., Vaswani, A., Shlens, J., and Le, Q. V. (2019). Attention augmented convolutional networks. In *IEEE/CVF Int. Conf. Comput. Vis.*, pages 3286–3295.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Bernasconi et al., 2020] Bernasconi, M., Djelouah, A., Hattori, S., and Schroers, C. (2020). Deep deinterlacing. In *SMPTE Annual Technical Conf. Exhibition*.

[Blattmann et al., 2023] Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S. W., Fidler, S., and Kreis, K. (2023). Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575.

[Bolya et al., 2022] Bolya, D., Fu, C.-Y., Dai, X., Zhang, P., Feichtenhofer, C., and Hoffman, J. (2022). Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*.

[Caballero et al., 2017] Caballero, J., Ledig, C., Aitken, A., Acosta, A., Totz, J., Wang, Z., and Shi, W. (2017). Real-time video super-resolution with spatio-temporal networks and motion compensation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4778–4787.

[Cao et al., 2023] Cao, Y., Li, S., Liu, Y., Yan, Z., Dai, Y., Yu, P. S., and Sun, L. (2023). A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt. *arXiv preprint arXiv:2303.04226*.

[Ceylan et al., 2023] Ceylan, D., Huang, C.-H. P., and Mitra, N. J. (2023). Pix2video: Video editing using image diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23206–23217.

[Chan et al., 2021] Chan, K. C., Wang, X., Yu, K., Dong, C., and Loy, C. C. (2021). Basicvsr: The search for essential components in video super-resolution and beyond. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4947–4956.

[Chan et al., 2022] Chan, K. C., Zhou, S., Xu, X., and Loy, C. C. (2022). Basicvsr++: Improving video super-resolution with enhanced propagation and alignment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5972–5981.

[Chellapilla et al., 2006] Chellapilla, K., Puri, S., and Simard, P. (2006). High performance convolutional neural networks for document processing. In *Tenth international workshop on frontiers in handwriting recognition*. Suvisoft.

[Chen et al., 2017a] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017a). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

[Chen et al., 2017b] Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017b). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

[Chen et al., 2024] Chen, M., Laina, I., and Vedaldi, A. (2024). Training-free layout control with cross-attention guidance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5343–5353.

[Chen et al., 2022] Chen, Z., Lu, G., Hu, Z., Liu, S., Jiang, W., and Xu, D. (2022). Lsvc: A learning-based stereo video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6073–6082.

[Cheng and Chen, 2021] Cheng, X. and Chen, Z. (2021). Multiple video frame interpolation via enhanced deformable separable convolution. *IEEE Trans. Pattern Anal. Mach. Intell.*

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representa-

tions using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Ciresan et al., 2011] Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *Twenty-second international joint conference on artificial intelligence*. Citeseer.

[Croitoru et al., 2023] Croitoru, F.-A., Hondru, V., Ionescu, R. T., and Shah, M. (2023). Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Dai et al., 2017] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. (2017). Deformable convolutional networks. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 764–773.

[Danier et al., 2022] Danier, D., Zhang, F., and Bull, D. (2022). Enhancing deformable convolution based video frame interpolation with coarse-to-fine 3d cnn. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1396–1400. IEEE.

[Dewil et al., 2023] Dewil, V., Courtois, A., Rodríguez, M., Ehret, T., Brandonisio, N., Bujoreanu, D., Facciolo, G., and Arias, P. (2023). Video joint denoising and demosaicing with recurrent cnns. In *Winter Conf. on Appl. Comput. Vis. (WACV)*.

[Dey and Salem, 2017] Dey, R. and Salem, F. M. (2017). Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE.

[Dhariwal and Nichol, 2021] Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794.

[Dong et al., 2021] Dong, S., Wang, P., and Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Review*, 40:100379.

[Dumoulin et al., 2016] Dumoulin, V., Shlens, J., and Kudlur, M. (2016). A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*.

[Ehret et al., 2019] Ehret, T., Davy, A., Arias, P., and Facciolo, G. (2019). Joint demosaicking and denoising by fine-tuning of bursts of raw images. In *IEEE/CVF Int. Conf. Comp. Vis. (ICCV)*, pages 8868–8877.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

[Feng et al., 2021] Feng, K., Zhao, Y., Chan, J. C.-W., Kong, S. G., Zhang, X., and Wang, B. (2021). Mosaic convolution-attention network for demosaicing multispectral filter array images. *IEEE Transactions on Computational Imaging*, 7:864–878.

[Fukushima, 1980] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.

[Gao et al., 2023] Gao, Z., Song, M., Schroers, C., and Zhang, Y. (2023). Revitalizing legacy video content: Deinterlacing with bidirectional information propagation. *arXiv preprint arXiv:2310.19535*.

[Gers and Schmidhuber, 2000] Gers, F. A. and Schmidhuber, J. (2000). Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 189–194. IEEE.

[Gers et al., 2000] Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471.

[Gers et al., 2002] Gers, F. A., Schraudolph, N. N., and Schmidhuber, J. (2002). Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

[Goodfellow et al., 2020] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

[Graves et al., 2008] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2008). A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868.

[Graves and Schmidhuber, 2005] Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.

[Greff et al., 2016] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2016). Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232.

[Guo et al., 2023] Guo, Y., Jin, Q., Morel, J.-M., and Facciolo, G. (2023). How to best combine demosaicing and denoising? *Inverse Problems and Imaging*, pages 0–0.

[Han and Moraga, 1995] Han, J. and Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks*, pages 195–201. Springer.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Heck and Salem, 2017] Heck, J. C. and Salem, F. M. (2017). Simplified minimal gated unit variations for recurrent neural networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1593–1596. IEEE.

[Hinton, 2012] Hinton, G. (2012). Rmsprop. Website. `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`.

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[Ho et al., 2022a] Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J., et al. (2022a). Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*.

[Ho et al., 2020] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

[Ho et al., 2022b] Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., and Salimans, T. (2022b). Cascaded diffusion models for high fidelity image generation. *The Journal of Machine Learning Research*, 23(1):2249–2281.

[Hochreiter et al., 2001] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

[Hochreiter and Schmidhuber, 1996] Hochreiter, S. and Schmidhuber, J. (1996). Lstm can solve hard long time lag problems. *Advances in neural information processing systems*, 9.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Huang et al., 2015] Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

[Janiesch et al., 2021] Janiesch, C., Zschech, P., and Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3):685–695.

[Ji and Tekalp, 2021] Ji, R. and Tekalp, A. M. (2021). Learned multi-field de-interlacing with feature alignment via deformable residual convolution blocks. In *Int. Conf. Visual Comm. and Image Proc. (VCIP)*, pages 1–5. IEEE.

[Ji and Tekalp, 2022] Ji, R. and Tekalp, A. M. (2022). Multi-field de-interlacing using deformable convolution residual blocks and self-attention. In *IEEE Int. Conf. Image Process. (ICIP)*, pages 901–905.

[Jiang et al., 2022] Jiang, B., Xie, Z., Xia, Z., Li, S., and Liu, S. (2022). Erdn: Equivalent receptive field deformable network for video deblurring. In *European Conference on Computer Vision*, pages 663–678. Springer.

[Jo et al., 2018] Jo, Y., Oh, S. W., Kang, J., and Kim, S. J. (2018). Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit*, pages 3224–3232.

[Jordan, 1986] Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Eighth Annual Conference of the Cognitive Science Society, 1986*, pages 513–546.

[Jozefowicz et al., 2015] Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2342–2350, Lille, France. PMLR.

[Karras et al., 2021] Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. (2021). Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863.

[Kawar et al., 2022] Kawar, B., Elad, M., Ermon, S., and Song, J. (2022). Denoising diffusion restoration models. *Advances in Neural Information Processing Systems*, 35:23593–23606.

[Kawar et al., 2023] Kawar, B., Zada, S., Lang, O., Tov, O., Chang, H., Dekel, T., Mosseri, I., and Irani, M. (2023). Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6007–6017.

[Khodairy and Abosamra, 2021] Khodairy, M. A. and Abosamra, G. (2021). Driving behavior classification based on oversampled signals of smartphone embedded sensors using an optimized stacked-lstm neural networks. *IEEE Access*, 9:4957–4972.

[Kim et al., 2007] Kim, W., Jin, S., and Jeong, J. (2007). Novel intra deinterlacing algorithm using content adaptive interpolation. *IEEE Trans. Consum. Electron.*, 53(3):1036–1043.

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

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Kobyzev et al., 2020] Kobyzev, I., Prince, S. J., and Brubaker, M. A. (2020). Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979.

[Kokkinos and Lefkimmiatis, 2019] Kokkinos, F. and Lefkimmiatis, S. (2019). Iterative residual cnns for burst photography applications. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pages 5929–5938.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[Kwon et al., 2003] Kwon, O., Sohn, K., and Lee, C. (2003). Deinterlacing using directional interpolation and motion compensation. *IEEE Trans. Consum. Electron.*, 49(1):198–203.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Lee et al., 2022] Lee, Y., Cho, S., and Jun, D. (2022). Video super-resolution method using deformable convolution-based alignment network. *Sensors*, 22(21):8476.

[Lei et al., 2023] Lei, P., Fang, F., Zeng, T., and Zhang, G. (2023). Flow guidance deformable compensation network for video frame interpolation. *IEEE Transactions on Multimedia*.

[Li et al., 2022] Li, H., Yang, Y., Chang, M., Chen, S., Feng, H., Xu, Z., Li, Q., and Chen, Y. (2022). Srdiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, 479:47–59.

[Li et al., 2023a] Li, H.-D., Yin, H., Liu, Z.-H., and Huang, H. (2023a). Enhanced spatial-temporal freedom for video frame interpolation. *Applied Intelligence*, 53(9):10535–10547.

[Li et al., 2023b] Li, Y., Min, K., Tripathi, S., and Vasconcelos, N. (2023b). Svitt: Temporal learning of sparse video-text transformers. In *Proceedings of the IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pages 18919–18929.

[Liao et al., 2019] Liao, Y., Xiong, P., Min, W., Min, W., and Lu, J. (2019). Dynamic sign language recognition based on video sequence with blstm-3d residual networks. *IEEE Access*, 7:38044–38054.

[Lin et al., 2013] Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

[Lin et al., 2024] Lin, S., Liu, B., Li, J., and Yang, X. (2024). Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5404–5411.

[Lin et al., 2020] Lin, Z., Li, M., Zheng, Z., Cheng, Y., and Yuan, C. (2020). Self-attention convlstm for spatiotemporal prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11531–11538.

[Liu et al., 2022] Liu, J., Zhou, M., and Xiao, M. (2022). Deformable convolution dense network for compressed video quality enhancement. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1930–1934. IEEE.

[Liu et al., 2021] Liu, Y., Zhang, X., Wang, S., Ma, S., and Gao, W. (2021). Spatial-temporal correlation learning for real-time video deinterlacing. In *IEEE Int. Conf. Multimedia Expo (ICME)*, pages 1–6.

[Losson et al., 2010] Losson, O., Macaire, L., and Yang, Y. (2010). Comparison of color demosaicing methods. *Advances in Imaging and Electron Physics*, 162:173–265.

[Maas et al., 2013] Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA.

[Mirza and Osindero, 2014] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

[MSU, 2022] MSU (2022). Msu deinterlacer benchmark. Website. `https://videoprocessing.ai/benchmarks/deinterlacer.html`.

[Nah et al., 2019] Nah, S., Baik, S., Hong, S., Moon, G., Son, S., Timofte, R., and Mu Lee, K. (2019). Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog. Workshops*.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

[Najafabadi et al., 2015] Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., and Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of big data*, 2(1):1–21.

[Nasir et al., 2021] Nasir, J. A., Khan, O. S., and Varlamis, I. (2021). Fake news detection: A hybrid cnn-rnn based deep learning approach. *International Journal of Information Management Data Insights*, 1(1):100007.

[Nichol et al., 2021] Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. (2021). Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*.

[Nichol and Dhariwal, 2021] Nichol, A. Q. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR.

[Oh and Jung, 2004] Oh, K.-S. and Jung, K. (2004). Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314.

[Papamakarios et al., 2021] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680.

[Pascanu et al., 2013] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr.

[Peebles and Xie, 2023] Peebles, W. and Xie, S. (2023). Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205.

[Qian et al., 2022] Qian, G., Wang, Y., Gu, J., Dong, C., Heidrich, W., Ghanem, B., and Ren, J. S. (2022). Rethinking learning-based demosaicing, denoising, and super-resolution pipeline. In *2022 IEEE International Conference on Computational Photography (ICCP)*, pages 1–12. IEEE.

[Radford et al., 2015] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[Ramesh et al., 2022] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3.

[Rombach et al., 2022] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). 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*, pages 234–241. Springer.

[Ruiz et al., 2023] Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. (2023). Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510.

[Saharia et al., 2022a] Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., and Norouzi, M. (2022a). Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10.

[Saharia et al., 2022b] Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. (2022b). Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494.

[Saharia et al., 2022c] Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. (2022c). Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726.

[Sajjadi et al., 2018] Sajjadi, M. S., Vemulapalli, R., and Brown, M. (2018). Frame-recurrent video super-resolution. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pages 6626–6634.

[Sarker, 2021] Sarker, I. H. (2021). Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6):420.

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

[Sharif et al., 2021] Sharif, S., Naqvi, R. A., and Biswas, M. (2021). Beyond joint demosaicking and denoising: An image processing pipeline for a pixel-bin image sensor. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 233–242.

[Shechtman et al., 2005] Shechtman, E., Caspi, Y., and Irani, M. (2005). Space-time super-resolution. *IEEE Trans. on Patt. Anal. Mach. Intell.*, 27(4):531–545.

[Shen et al., 2021] Shen, Z., Zhang, M., Zhao, H., Yi, S., and Li, H. (2021). Efficient attention: Attention with linear complexities. In *IEEE/CVF Winter Conf. Appl. Comp. Vis*, pages 3531–3539.

[Shi et al., 2015] Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28.

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

[Sohl-Dickstein et al., 2015] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR.

[Song et al., 2023] Song, C., Li, H., Zheng, D., Wang, J., Jiang, Z., and Yang, B. (2023). Transformer-based video deinterlacing method. In *International Conference on Neural Information Processing*, pages 357–369. Springer.

[Song et al., 2018] Song, H., Wang, W., Zhao, S., Shen, J., and Lam, K.-M. (2018). Pyramid dilated deeper convlstm for video salient object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 715–731.

[Song and Ermon, 2019] Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

[Song and Ermon, 2020] Song, Y. and Ermon, S. (2020). Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448.

[Song et al., 2020a] Song, Y., Garg, S., Shi, J., and Ermon, S. (2020a). Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR.

[Song and Kingma, 2021] Song, Y. and Kingma, D. P. (2021). How to train your energy-based models. *arXiv preprint arXiv:2101.03288*.

[Song et al., 2020b] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020b). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

[Soomro et al., 2012] Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

[Stypułkowski et al., 2024] Stypułkowski, M., Vougioukas, K., He, S., Zięba, M., Petridis, S., and Pantic, M. (2024). Diffused heads: Diffusion models beat gans on talking-face generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5091–5100.

[Sundermeyer et al., 2014] Sundermeyer, M., Alkhouli, T., Wuebker, J., and Ney, H. (2014). Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 14–25.

[Sutskever et al., 2013] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[Tan et al., 2018] Tan, D. S., Chen, W.-Y., and Hua, K.-L. (2018). Deepdemosaicking: Adaptive image demosaicking via multiple deep fully convolutional networks. *IEEE Trans. Image Process.*, 27(5):2408–2419.

[Tan et al., 2022] Tan, H., Xiao, H., Liu, Y., and Zhang, M. (2022). Two-stage cnn

model for joint demosaicing and denoising of burst bayer images. *Computational Intelligence and Neuroscience*, 2022.

[Tay et al., 2020] Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. (2020). Efficient transformers: A survey. *ACM Computing Surveys (CSUR)*.

[Tekalp, 2022] Tekalp, A. M. (2022). *Deep Learning for Image/Video Restoration and Super-resolution*. Now Foundations and Trends.

[Tian et al., 2020] Tian, Y., Zhang, Y., Fu, Y., and Xu, C. (2020). TDAN: temporally-deformable alignment network for video super-resolution. In *IEEE/CVF Conf. Comput. Vis. Patt. Recog.*, pages 3360–3369.

[Ullah et al., 2019] Ullah, M., Ullah, H., Khan, S. D., and Cheikh, F. A. (2019). Stacked lstm network for human activity recognition using smartphone data. In *2019 8th European workshop on visual information processing (EUVIP)*, pages 175–180. IEEE.

[Vahdat and Kautz, 2020] Vahdat, A. and Kautz, J. (2020). Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679.

[Vahdat et al., 2021] Vahdat, A., Kreis, K., and Kautz, J. (2021). Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

[Vincent, 2011] Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

[Waibel et al., 1989] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. (1989). Phoneme recognition using time-delay neural networks. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37:328 – 339.

[Wang et al., 2022a] Wang, P., Wang, X., Wang, F., Lin, M., Chang, S., Li, H., and Jin, R. (2022a). KVT: k-nn attention for boosting vision transformers. In *European Conf. on Computer Vision (ECCV), Tel Aviv, Israel, October 23–27, 2022, Part XXIV*, pages 285–302. Springer.

[Wang et al., 2019] Wang, X., Chan, K. C., Yu, K., Dong, C., and Change Loy, C. (2019). Edvr: Video restoration with enhanced deformable convolutional networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, pages 0–0.

[Wang et al., 2018] Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., and Change Loy, C. (2018). Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0.

[Wang et al., 2022b] Wang, Y., Wu, H., Zhang, J., Gao, Z., Wang, J., Philip, S. Y., and Long, M. (2022b). Predrnn: A recurrent neural network for spatiotemporal predictive learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2208–2225.

[Wang et al., 2020] Wang, Y., Yin, S., Zhu, S., Ma, Z., Xiong, R., and Zeng, B. (2020). NTSDCN: new three-stage deep convolutional image demosaicking network. *IEEE Trans. Circuits and Syst. Video Technol.*, 31(9):3725–3729.

[Wang et al., 2004] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612.

[Welling and Teh, 2011] Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688.

[Whang et al., 2022] Whang, J., Delbracio, M., Talebi, H., Saharia, C., Dimakis, A. G., and Milanfar, P. (2022). Deblurring via stochastic refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16293–16303.

[Williams and Peng, 1990] Williams, R. J. and Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501.

[Wu et al., 2023] Wu, Z., Zhang, K., Sun, C., Xuan, H., and Yan, Y. (2023). Flow-guided deformable alignment network with self-supervision for video inpainting. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

[Xia et al., 2022] Xia, B., Hang, Y., Tian, Y., Yang, W., Liao, Q., and Zhou, J. (2022). Efficient non-local contrastive attention for image super-resolution. In *Proceedings of the AAAI Conf. Artif. Intell.*, volume 36, pages 2759–2767.

[Xia et al., 2021] Xia, M., Shao, H., Ma, X., and de Silva, C. W. (2021). A stacked gru-rnn-based approach for predicting renewable energy and electricity load for smart grid operation. *IEEE Transactions on Industrial Informatics*, 17(10):7050–7059.

[Xu et al., 2020] Xu, X., Ye, Y., and Li, X. (2020). Joint demosaicing and super-resolution (jdsr): Network design and perceptual optimization. *IEEE Transactions on Computational Imaging*, 6:968–980.

[Xue et al., 2019] Xue, T., Chen, B., Wu, J., Wei, D., and Freeman, W. T.

(2019). Video enhancement with task-oriented flow. *Int. Journal Comput. Vis.*, 127(8):1106–1125.

[Yang et al., 2023a] Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., and Yang, M.-H. (2023a). Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39.

[Yang et al., 2023b] Yang, R., Srivastava, P., and Mandt, S. (2023b). Diffusion probabilistic modeling for video generation. *Entropy*, 25(10):1469.

[Yeh et al., 2022] Yeh, Y.-C., Dy, J., Huang, T.-M., Chen, Y.-Y., and Hua, K.-L. (2022). Vdnet: video deinterlacing network based on coarse adaptive module and deformable recurrent residual network. *Neural Computing and Applications*, 34(15):12861–12874.

[Yılmaz and Tekalp, 2021] Yılmaz, M. A. and Tekalp, A. M. (2021). DFPN: Deformable frame prediction network. In *2021 IEEE Int. Conf. Image Process.(ICIP)*, pages 1944–1948. IEEE.

[Yılmaz et al., 2023] Yılmaz, M. A., Ulas, O. U., and Tekalp, A. M. (2023). Multi-scale deformable alignment and content-adaptive inference for flexible-rate bi-directional video compression. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 2475–2479. IEEE.

[Ying et al., 2020] Ying, X., Wang, L., Wang, Y., Sheng, W., An, W., and Guo, Y. (2020). Deformable 3d convolution for video super-resolution. *IEEE Signal Process. Lett.*, 27:1500–1504.

[Yoo and Jeong, 2002] Yoo, H. and Jeong, J. (2002). Direction-oriented interpolation and its application to de-interlacing. *IEEE Trans. on Consum. Electron.*, 48(4):954–962.

[Youku, 2019] Youku (2019). Youku video enhancement and super resolution dataset. Website. `https://tianchi.aliyun.com/dataset/dataDetail?dataId=39568`.

[Yu and Koltun, 2015] Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

[Yu et al., 2017a] Yu, F., Koltun, V., and Funkhouser, T. (2017a). Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480.

[Yu et al., 2022] Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., et al. (2022). Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5.

[Yu et al., 2017b] Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017b). Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

[Yu et al., 2019] Yu, Y., Si, X., Hu, C., and Zhang, J. (2019). A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270.

[Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer.

[Zeiler et al., 2011] Zeiler, M. D., Taylor, G. W., and Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *2011 international conference on computer vision*, pages 2018–2025. IEEE.

[Zhang et al., 2023a] Zhang, C., Zhang, C., Zheng, S., Qiao, Y., Li, C., Zhang, M., Dam, S. K., Thwal, C. M., Tun, Y. L., Huy, L. L., et al. (2023a). A complete survey on generative ai (aigc): Is chatgpt from gpt-4 to gpt-5 all you need? *arXiv preprint arXiv:2303.11717*.

[Zhang et al., 2022a] Zhang, K., Li, Y., Zuo, W., Zhang, L., Van Gool, L., and Timofte, R. (2022a). Plug-and-play image restoration with deep denoiser prior. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(10):6360–6376.

[Zhang et al., 2023b] Zhang, L., Rao, A., and Agrawala, M. (2023b). Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847.

[Zhang et al., 2022b] Zhang, T., Fu, Y., and Li, C. (2022b). Deep spatial adaptive network for real image demosaicing. *AAAI Conf. Artif. Intell.*, 36(3):3326–3334.

[Zhang et al., 2024] Zhang, T., Fu, Y., and Zhang, J. (2024). Deep guided attention network for joint denoising and demosaicing in real image. *Chinese Journal of Electronics*, 33(1):1–9.

[Zhao et al., 2022a] Zhao, Y., Jia, W., and Wang, R. (2022a). Rethinking deinterlacing for early interlaced videos. *IEEE Trans. Circuits and Syst. Video Technol.*, 32(7):4872–4878.

[Zhao et al., 2022b] Zhao, Y., Ma, Y., Chen, Y., Jia, W., Wang, R., and Liu, X. (2022b). Multi-frame joint enhancement for early interlaced videos. *IEEE Trans. Image Process.*

[Zhou and Chellappa, 1988] Zhou and Chellappa (1988). Computation of optical flow using a neural network. In *IEEE 1988 international conference on neural networks*, pages 71–78. IEEE.

[Zhou et al., 2016] Zhou, G.-B., Wu, J., Zhang, C.-L., and Zhou, Z.-H. (2016). Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, 13(3):226–234.

[Zhu et al., 2017] Zhu, H., Liu, X., Mao, X., and Wong, T.-T. (2017). Real-time deep video deinterlacing. *arXiv preprint arXiv:1708.00187*.

[Zhu et al., 2019] Zhu, X., Hu, H., Lin, S., and Dai, J. (2019). Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9308–9316.

Appendix A

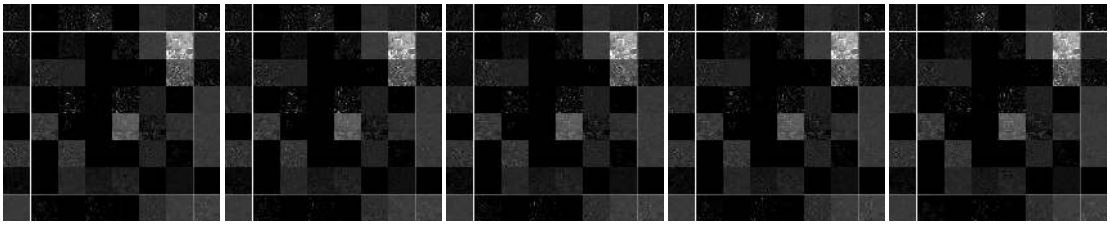
A.1 FLOPS and Model Sizes

Floating-point Operations Per Second (FLOPS) expresses the computer performance by measuring the numbers of floating-point calculations one system can perform in a second. Model size can be indicated by the amounts of learnable parameters in a corresponding model. Table A.1 shows the FLOPS on two input sizes of Vid4 dataset and parameter amounts for each model.

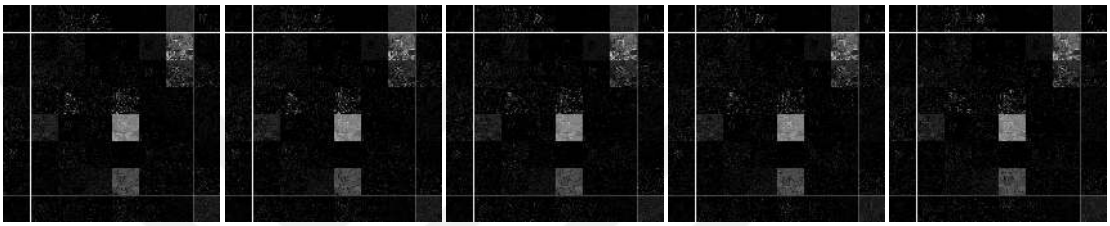
Table A.1: FLOPS and sizes of different models on Vid4 dataset, where EkSA represents DfConv+EkSA.

Deinterlace	EkSA	DfRes+ESA	DfRes	DeT	RTDVD
Parameters	2.94324M	2.78428M	2.75674M	8.18531M	58.496k
Input size	(6,3,288,704)	(6,3,288,704)	(6,3,288,704)	(3,3,288,704)	(3,576,704)
FLOPS	813.331G	702.233G	695.184G	1272.68G	141.622G
Input size	(6,3,240,704)	(6,3,240,704)	(6,3,240,704)	(3,3,240,704)	(3,480,704)
FLOPS	666.357G	585.194G	579.32G	1060.57G	118.018G
Demosaic	EkSA	NTSDCN	BJDD	Kokkinos	DPIR
Parameters	3.46023M	1.253443M	3.47006M	380.375K	188.163K
Input size	(5,3,576,576)	(3,576,576)	(3,576,576)	(5,3,576,576)	(3,576,576)
FLOPS	1759.55G	833.488G	174.676G	125.938G	62.5554G
Input size	(5,3,480,480)	(3,480,480)	(5,3,480,480)	(5,3,480,480)	(3,480,480)
FLOPS	1175.2G	578.81G	121.302G	87.4568G	43.4412G

A.2 Feature Visualization

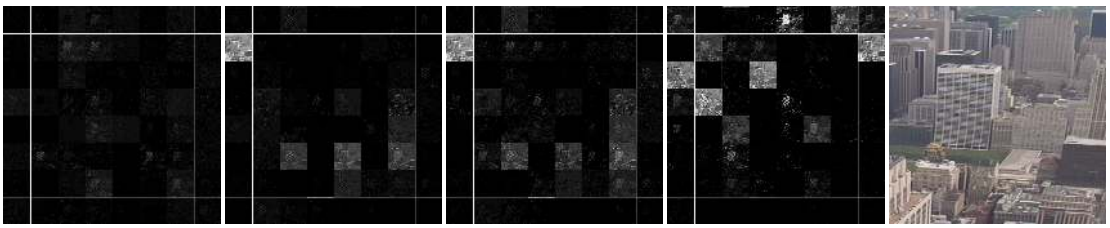


(a) Features after first convolution layer Conv_1



(b) Extracted features after ×5 Res_Block

(c) Neighboring features aligned to the reference feature before Conv_2



(d) DfConv (e) EkSA (f) Integration (g) SepRecon (h) Output field

Figure A.1: Visualized feature for deinterlacing task, where (a)-(c) have size of (5, 64, 288, 288), (d)-(g) have size of (64, 288, 288), and (b) has size of (3, 288, 288). Channels 64 are arranged to be an 8×8 gray scale image matrix. Frame number 5 is displayed as 5 separate image matrices.

We display feature maps as image matrices in Figure A.1, where features are taken after being processed by different blocks in Figure 5.1. Figure A.1 (a)-(b) show low level features which are similar in pattern. Features processed by DfConv

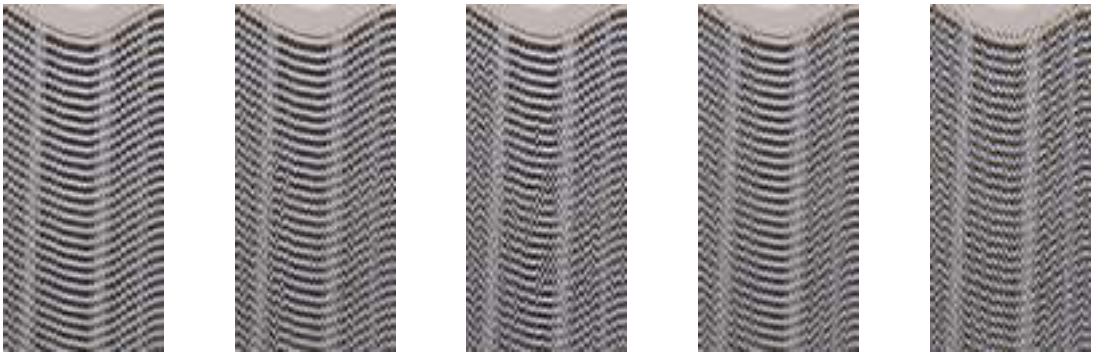
in (c) and (d) and by EkSA in (e) show different but complementary details.

A.3 Temporal Profiles

(a) Horizontal temporal profiles on calendar clip (74 frames).

(b) Vertical temporal profiles on calendar clip (74 frames).

(c) Horizontal temporal profiles on city clip (60 frames).



(d) Vertical temporal profiles on city clip (60 frames).

Figure A.2: Temporal profiles on Vid4 dataset for deinterlacing task, from left to right: DfConv_EkSA, DfRes_SA, DfRes, DeT, RTDVD.

Temporal profiles [Caballero et al., 2017] are generated by stacking a line of pixels across multiple frames at fixed width or height to show the temporal consistency.

Figure A.2 shows that our three methods produce more accurate results through time axis and DfConv_EkSA generates finer details than the other two methods.