Temporal Pyramid Structure for Video Frame Interpolation

Jiaqi Yang

Thesis submitted to the University of Ottawa in partial
fulfillment of the requirements for the degree of
Master of Applied Science in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The most prevalent structure in video frame interpolation involves using optical flow to guide frame warping, which typically considers only the two adjacent frames. However, these methods often fail to capture long-range temporal dependencies and often result in significant deformation in complex motion scenarios. We propose a novel Temporal Pyramid Attention (TPA) block, which employs a temporal pyramid structure to connect four frames within a sliding window for the generation of intermediate frames. The temporal pyramid structure consists of three layers to leverage multi-level features, estimate the frame window, and connect with a GRU to generate a bi-directional feature flow. Furthermore, the dual pyramid structure incorporates channel attention mechanisms, enabling the interpolation of three frames in a single process. The TPA block employs a multi-scale approach to effectively capture temporal dependencies and spatial correlations, enhancing the quality of interpolated frames. Our model achieves a state-of-the-art performance on the Vimeo90K septuplet dataset compared to existing methods using pre-trained parameters.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Jiying Zhao, for his invaluable support and guidance throughout my research journey. His insights and patience have been crucial to my progress.

I am also thankful to co-supervisor Dr. Yu Liu. Our numerous discussions have been profoundly influential, and I appreciate his engagement and intellectual contributions.

I am immensely grateful to my family and friends, who have provided me with endless encouragement and patience throughout this journey.

Lastly, I extend my thanks to everyone who was involved in any capacity in my project. This accomplishment would not have been possible without such a supportive community.

Author Contributions

In this thesis, we introduce an innovative temporal pyramid structure designed specifically for the task of video frame interpolation. Our approach involves segmenting the input video sequence into a series of window clips, allowing for the estimation of intermediate frames based on the contextual information inherent within each divided window clip. We thoroughly examine the functionality and efficacy of the temporal pyramid by analyzing the differences in feature representation before and after its application. Furthermore, we evaluate the performance of our model against state-of-the-art techniques, demonstrating that our method achieves significant advancements in video frame interpolation. The results underscore the potential of the temporal pyramid structure to enhance the accuracy and quality of interpolated video frames, marking a substantial contribution to the field.

Contents

List of Figures

List of Tables

List of Abbreviations

BiLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional neural network
ConvLSTM	Convolutional long short-term memory
CV	Computer Vision
DNN	Deep neural network
fps	frames per second
GRU	Gated recurrent unit
GT	Ground Truth
LSTM	Long short-term memory
NLP	Natural language processing
PSNR	Peak signal-to-noise ratio
RNN	Recurrent neural network
SSIM	Structural similarity index measure
TP	Temporal pyramid
VFI	Video frame interpolation
VSR	Video super resolution

1

Introduction

1.1 Motivation

In the modern digital society, video has become one of the most powerful and pervasive forms of communication. At its core, a video is a sequence of still images called frames, which creates the illusion of continuous motion when displayed in rapid succession. The

smoothness and quality of this motion depend largely on how many frames are shown per second, with higher frame rates generally resulting in more fluid and natural movement.

Slow motion video is to play a high frame rate video, like 240 frames per rate (fps) at a normal rate, like 30 fps. This technique significantly enhances the perception of details and temporal resolution, enabling a detailed analysis of fast-moving events.

There are two main approaches to obtaining high frame rate video. Using expensive high-speed cameras can capture more frames per second than normal cameras, which cannot be universally accessible. Another method is to use algorithms to intelligently generate new frames between existing ones, which is called video frame interpolation (VFI). VFI is a critical low-level task in computer vision that plays a significant role in various applications. At its core, VFI involves the generation of intermediate frames between existing frames in a video sequence. This process is essential for increasing the frame rate of videos, which can enhance the viewing experience in numerous contexts, such as sports events, animations, and cinematic productions.

The traditional method employs high-speed cameras to capture footage at elevated frame rates, which is then played back at standard speeds, thus extending the duration of the video. However, this technique necessitates the use of expensive, high-quality cameras, potentially limiting its applicability in scenarios with budget constraints. Moreover, this method applies only to videos that have already been captured, making the use of algorithms for VFI the only viable solution.

The importance of VFI has been shown in many different areas in the real world: In

sports broadcasting, for example, higher frame rates allow for smoother motion and clearer detail during fast-paced action, enabling viewers to catch every moment without motion blur. Similarly, in animation, interpolating frames can create fluid transitions, making the animation appear more lifelike and engaging. Furthermore, in the realm of virtual reality and gaming, higher frame rates contribute to a more immersive experience, reducing latency and enhancing user interaction. Beyond entertainment, VFI also has practical applications in areas such as surveillance, where improved frame rates can lead to better detection and tracking of objects. In medical imaging, it can aid in the analysis of motion in dynamic processes, providing clearer insights into patient conditions.

1.2 Challenge and research problem

However, early algorithms often simply duplicated adjacent frames or used bi-linear interpolation to extend video duration. While these methods did extend the duration, they resulted in very disjointed movements within the video, significantly detracting from the viewing experience.

Despite significant advancements in current VFI solutions, which have achieved impressive performance in terms of metrics such as peak signal-to-noise ratio(PSNR), challenges remain, particularly when the video sequence needs to be extended to $4\times$ or $8\times$ the original frame rate. This level of interpolation presents unique difficulties, as the algorithms must generate new frames that not only maintain the overall visual quality but also preserve the natural motion and fluidity of the original content.

One of the primary challenges lies in complex scenes and large motions, where deep learning models often struggle to maintain accuracy and consistency.

The current models usually fail to track the motion of fast-moving objects, and in scenes where objects need to pass through a series of obstacles or interact with one another.

Deep learning-based video interpolation methods often introduce visual artifacts where moving objects appear distorted or elastically deformed. These distortions manifest as ghosting effects, where residual traces of objects remain visible in their previous positions, creating a visible trembling or vibration effect in the interpolated video sequence. This phenomenon is particularly noticeable in regions with rapid motion or complex object trajectories

In some interpolations of street dance videos, the pivoting legs may be extremely out of shape. Another challenge in this process is the handling of great deformations, especially in human limbs during large motion. When interpolating at such high frame rates, the algorithms can struggle to accurately predict and recreate the movements of limbs or other objects, leading to artifacts that can be strange to the viewer. These artifacts, which can be considered mistakes, can manifest as unnatural stretching or distortion of objects, which greatly detracts from the realism of the video. These problems require the VFI algorithms to not only interpolate frames but also to understand the context of the motion. This means that the algorithms must be capable of discerning subtle changes in posture and movement dynamics, which can be particularly challenging when the motion involves rapid transitions or complex interactions between multiple subjects.

By employing architectures capable of capturing intricate motion patterns and temporal dependencies, such as flow-based and kernel-based methods, significant strides have been made in enhancing frame synthesis quality. However, the reliance on accurate optical flow estimation remains a critical bottleneck; inaccuracies of optical flow can lead to artifacts that detract from the overall viewing experience. Furthermore, recent innovations like generalized deformable convolution offer promising avenues by allowing models to adaptively learn motion information, which may help mitigate some limitations associated with traditional approaches. As technology continues to evolve, integrating these sophisticated methodologies could pave the way for more robust solutions that not only enhance visual fidelity but also effectively handle complex dynamic scenes, thereby transforming applications across various domains.

The main difference between current models is whether the optical flow is used to guide the frame warping.

Models which do not require the guidance of optical flow use a convolutional neural network (CNN) to construct the map from the input frames to the interpolated output frame. These approaches typically consider only the neighbouring two frames, which limits their ability to capture long-range temporal dependencies and handle complex motion patterns. As a result, these methods often struggle with artifacts and inaccurate interpolations, especially in scenarios with occlusions or large motion.

Flow-guided models have become increasingly prominent in the field of VFI. Most current models opt to train their own optical flow estimation modules, leveraging the

derived optical flow to warp adjacent frames for the interpolation task. Additionally, masks are often estimated alongside optical flow to mitigate occlusion issues. The effectiveness of these methods, however, is contingent upon the accuracy of both the optical flow estimation and the subsequent warping process. Although the relation between the two frames has been emphasized, the information from the temporal scene of the video is overlooked.

The current models will always generate intermediate frames whose objects will be stretched, shrunk and lose their original shape. These stretching and shrinking will be accumulated when users recursively use the models to make a high frame rate video, which will lead to extremely severe distortions that detract and ultimately undermine the viewer's experience.

To address these limitations and broaden the scope of application, it is imperative to develop more sophisticated models capable of generating intermediate frames that ensure shape stability. We propose a novel approach that integrates temporal coherence constraints, ensuring that the generated frames maintain the integrity of the original shapes while preserving fluid motion and decreasing the distortions in high frame rate interpolation.

1.3 Contributions

The contributions of this thesis can be summarized as:

- We introduce Temporal Pyramid (TP), an innovative architectural design that lever-

ages a temporal pyramid structure to interconnect four frames within a sliding window. This design enables the model to harness features across multiple levels, thereby facilitating the generation of bi-directional feature flow, which significantly improves interpolation accuracy, particularly in super slow-motion scenarios. Additionally, a dual-pyramid structure, integrated with a channel attention mechanism, is employed to effectively extract and utilize critical information from the feature set. This window-based approach predicts three intermediate frames from the four input frames in a single step, thereby enhancing the continuity and coherence of the video sequences.

- We employ the distilled GRU block to estimate motion using ground truth (GT) data, assisting in the training process by ensuring that motion details are accurately captured within the windowed clip. The teacher model follows the same processing pipeline as the student model, with the exception of receiving GT data and incorporating an additional GRU step within the GRU structure. The difference in hidden states between the teacher and student models is computed to guide the student model in refining its motion estimation capabilities.

- We propose a pyramid hybrid U-Net to fully exploit the temporal pyramid features. This model diverges from the traditional context block approach, which relies solely on the current neighbouring frames. Instead, the temporal pyramid features are concatenated at various layers of the U-Net, thereby providing information from multiple resolutions, which enhances the model's ability to capture and utilize temporal in-

formation across different scales.

1.4 Structure of the thesis

The outline for the rest of the thesis is as follows:

Chapter 2 provides fundamental knowledge of deep learning methodologies and the evaluation metrics utilized to assess the performance of various models

Chapter 3 presents papers related to our topic and outlines the methodologies of state-of-the-art studies. Additionally, it explains how our approach is inspired by these works.

Chapter 4 provides a detailed explanation of our methodology. We structure our model into three key components: the TP, the motion estimator, and the image reconstruction module, and describe the functionality of each component in detail.

Chapter 5 shows the details of our training and the results comparison between our model and other state-of-the-art models.

Chapter 6 provides conclusions to summarize our method and results. We discuss what can be developed in the future to improve the ability of our model.

2

Background

2.1 Deep neural network

Deep neural network (DNN) is a subset of machine learning. It has revolutionized artificial intelligence due to its remarkable ability to learn from datasets and generate high-quality outputs. The applications of DNN in computer vision have led to significant advancements

in image recognition, object detection, video processing, etc.

DNN are built to imitate how the human brain works. They consist of layers of connected nodes, or "neurons" that work together to process information in intricate ways.

The start layers of a network, the input layers, receive input data and begin the process of transforming it. This transformation happens through a feature extraction system that passes the data through deep layers, the hidden layers. The tensors from the hidden layers will be passed to output layers to generate final outputs. Each layer works to identify complex features from the data from various angles, often using elements like convolutional blocks.

The connections between neurons are weighted, and these weights get adjusted during the training process through a method called backpropagation. This process involves calculating the error in the network's predictions and subsequently updating the weights to minimize the error.

Utilizing these structures, DNNs can learn to identify patterns, transmit essential information, and produce the desired output, informed by the training datasets.

A typical DNN consists of multiple layers, where each layer performs a transformation on its inputs to contribute to the final output. The computation within a single layer of a DNN can be described by the following equation:

$$a^{(l)} = \sigma(W^{(l)}a^{(l-1)} + b^{(l)}), \tag{2.1}$$

where,

- $a^{(l)}$ is the activation of layer l,
- $W^{(l)}$ is the weight matrix for layer l,
- $b^{(l)}$ is the bias vector for layer l,
- $a^{(l-1)}$ is the activation of the previous layer $l-1$,
- σ represents the activation function, which introduces non-linearity into the network.

People have proposed different architectures like RNN and Autoencoders to expand the ability of information utilization.

2.2 RNN

RNN is a type of neural network that processes inputs in a sequential manner, where the output from a one-time step is recursively fed back as input at the subsequent time step. This enables RNNs to capture temporal dependencies within sequential data, rendering them well-suited for tasks such as speech recognition.

The recurrent dynamics of a basic Recurrent Neural Network (RNN) can be described by the following equations:

$$h_t = f(W_{hh}h_{t-1} + W_{xh}x_t + b_h), \tag{2.2}$$

where,

- h_t is the hidden state at time t,
- h_{t-1} is the hidden state at time $t-1$,
- x_t is the input at time t,
- W_{hh} is the weight matrix for the hidden state,
- W_{xh} is the weight matrix for the input,
- b_h is the bias for the hidden state,
- f is a nonlinear activation function, typically a sigmoid or tanh function.

The output at each timestep is given by:

$$y_t = W_{hy}h_t + b_y, \tag{2.3}$$

where,

- W_{hy} is the weight matrix from the hidden state to the output,
- b_y is the output bias.

RNN can dynamically adapt the internal states according to the input sequences, making them highly effective for tasks where temporal dynamics are critical. RNN also can leverage the long-range memory [1, 2]

2.2.1 LSTM

Long short-term memory (LSTM) is one kind of recurrent neural network (RNN) designed to effectively capture long-range dependencies in sequential data and avoid vanishing gradient problems. The structure achieves advantages by introducing memory cells (hidden states and cell states) with three key gate blocks: the input gate, the forget gate, and the output gate. These gates regulate the flow of information in and out of the memory cell, allowing the network to retain relevant information and keep short-term and long-term memory, respectively.

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t, \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \\
h_t &= o_t \cdot \tanh(C_t),
\end{aligned}
\tag{2.4}
$$

where,

- f_t is the activation vector of the forget gate.
- i_t is the activation vector of the input gate.
- W_f and b_f are the weight matrix and bias for the forget gate.
- W_i and b_i are the weight matrix and bias for the input gate.

- W_C and b_C are the weight matrix and bias for the candidate cell state.
- W_o and b_o are the weight matrix and bias for the output gate.
- C_t is the cell state from the current time step.
- C_{t-1} is the cell state from the previous time step.
- $\tilde{C}_t$ is the activation vector of the cell input.
- h_{t-1} is the hidden state from the previous time step.
- h_t is the hidden state from the current time step.
- x_t is the input at the current time step.
- σ is the sigmoid activation function, squashing the output to the range $[0, 1]$.

2.2.2 Gated recurrent unit

Following the idea of LSTM, GRUs simplify the LSTM architecture while only using a hidden state to capture dependencies in sequential data. GRUs were introduced as a more efficient alternative to LSTMs, combining the forget and input gates functionality into a single update gate, which only needs to maintain one memory tensor to reduce the number of parameters.

The GRU consists of two main gates: the update and reset gates.

- The update gate determines how much information from the past needs to be passed to the bank.
- The reset gate decides how much past information to forget when processing the new input.

The operation of a GRU can be summarized as follows:

$$\begin{aligned} z_t &= \sigma(W_z \cdot [h_{t-1}, x_t] + b_z), \\ r_t &= \sigma(W_r \cdot [h_{t-1}, x_t] + b_r), \\ \tilde{h}_t &= \tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h), \\ h_t &= (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t, \end{aligned} \tag{2.5}$$

where,

- z_t: Update gate at the current time stamp.
- r_t: Reset gate at the current time stamp.
- $\tilde{h}_t$: Candidate hidden state at current time stamp.
- h_t: Hidden state at the current time stamp
- W_z and b_z are the weight matrix and bias for the update gate.
- W_r and b_r are the weight matrix and bias for the reset gate.
- W_h and b_h are the weight matrix and bias for the hidden states.

- σ: Sigmoid activation function.

Update Gate (z_t): This gate computes how much the new input should update the previous hidden state. It takes both the last hidden state (h_{t-1}) and the current input (x_t) to produce a value between 0 and 1, where 0 means completely forgetting the previous state and one means fully retaining it.

Reset Gate (r_t) determines how much past information can be forgotten. It also takes the previous hidden state and current input to compute a value between 0 and 1.

The updated hidden state is computed as a combination of the previous hidden state and the candidate hidden state. The candidate hidden state is influenced by both the current input and the previous hidden state, modulated by the reset gate. This mechanism enables the GRU to effectively integrate past and present information.

2.3 Autoencoder

Autoencoder is one kind of unsupervised learning artificial neural network, which is designed to learn the representations of input data and extract high-level features. The main objective is to transform the input into the output with the least distortion possible.[3] The number of channels in an input tensor $C \times H \times W$ may be doubled, while the resolution is halved, resulting in $4C \times \frac{H}{2} \times \frac{W}{2}$, where C, H, and W represent the dimensions of the input tensor.

The architecture of an autoencoder comprises two components: an encoder and a decoder.

- Encoder: The encoder compresses the input data into a lower-dimensional representation, known as the latent space. This process involves reducing the dimensionality of the input while retaining its essential features. The encoder consists of one or more layers of neurons that transform the input data into a compact representation.
- Decoder: The decoder takes the compressed latent representation and reconstructs it back to the original input space. The goal is to minimize the difference between the input and the reconstructed output, allowing the network to learn the underlying structure of the data.

Autoencoder performs non-linear transformations, making them more flexible and powerful for capturing complex data patterns[4, 5].

2.3.1 U-Net

The U-Net architecture comprises a symmetric encoder-decoder structure enhanced by skip connections. Within the encoder, a sequence of convolutional layers coupled with continuous down-sampling is employed to extract deep features characterized by extensive receptive fields. Subsequently, the decoder undertakes the up-sampling of these extracted features back to the resolution of the input, facilitating pixel-level semantic prediction [6].

Importantly, the skip connections play a crucial role by fusing high-resolution features from the encoder at various scales. This integration serves to mitigate the loss of spatial information inherently associated with the down-sampling process. The architectural design of U-Net, therefore, effectively balances the extraction of contextual data through down-sampling and the preservation of spatial precision necessary for detailed semantic segmentation [7]

2.4 Evaluation metrics

Following other papers on video frame interpolation (VFI), we choose to use the PSNR and Structural Similarity Index Measure (SSIM) to compare the quality of the output of the models.

2.4.1 Peak signal-to-noise ratio

The PSNR is a critical metric in various fields, particularly in optical correlators and wireless communication systems. It quantifies the level of a desired signal relative to the background noise, thereby influencing the quality and reliability of data transmission.

The PSNR represents the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation.

Although PSNR is commonly used due to its simplicity and ease of calculation, it often does not align perfectly with human visual perception. As a result, higher PSNR values

are traditionally interpreted as indicative of better quality, but they may not necessarily correspond to visually better images [8].

PSNR is calculated using the formula:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right), \tag{2.6}$$

where,

- MAX is the maximum possible pixel value of the image,
- MSE is the Mean Squared Error between the original and the compressed image.

Overall, PSNR is a fundamental concept that integrates various aspects of signal processing, measurement techniques, and communication systems. Its optimization is essential for enhancing signal quality and ensuring reliable data transmission across different applications

2.4.2 Structural similarity index measure

SSIM is a method for measuring the similarity between two images and is widely utilized in the field of image processing to evaluate the quality of processed images relative to an original, unaltered image. It was developed to provide a more perceptually relevant evaluation than simpler measures such as PSNR.

SSIM assesses image quality by evaluating changes in luminance, contrast, and structure, establishing it as a standard metric for image degradation analysis [9]

The SSIM index is defined as:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{2.7}$$

where,

- μ_x and μ_y are the average of x and y,
- σ_x^2 and σ_y^2 are the variance of x and y,
- σ_{xy} is the covariance of x and y,
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ are two variables to stabilize the division with weak denominator,
- L is the dynamic range of the pixel-values (typically this is $2^{\text{bits per pixel}} - 1$),
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.

3

Literature Review

In recent years, the field of video frame interpolation has gained significant attention because of its applications in various real-world domains, including video compression, computer graphics, etc.

Our research focuses on video frame interpolation, which primarily involves optical flow estimation, distillation, U-Net architecture, and GRU models.

3.1 Optical flow estimation

Optical flow estimation is critical to many computer vision tasks. In video frame interpolation, the optical flow-based method has shown great achievements. The optical flow takes the role of guiding the network to align the pixels between different images and feature maps. The introduction of optical flow can strongly simplify the structure of the VFI model and provide interoperability for the methods.

The first attempts to learn optical flow models involved using Markov Random Fields [10] and Field-of-Experts [11] models to capture spatial statistics. These models were limited by the lack of realistic training data and shallow architectures.

Dosovitskiy et al. proposed *FlowNet* as the first to use a deep learning network to estimate optical flow [12]. Ilg et al. expanded their idea by developing a stacked architecture to implement the warping of the second input image using intermediate optical flow [13].

Ranjan and Black employed a straightforward approach, *SpyNet*, to estimate optical flow by progressively refining the resolution from coarse to fine and from low to high, which has become the foundation for subsequent research [14].

SpyNet network takes the origin image pair as input, which will be downsampled to a series of sub-images. The optical flow will be computed by pure convolutional neural network (CNN) blocks to compute the correlation between two input images, which will be used to compute the ΔOF, which will be summed up to compute the final optical flow.

The spatial pyramid of Spynet is simple but allows the network to handle large motions

in a coarse-to-fine manner. At each level, one of the input image is warped using the current flow estimate, and the CNN predicts an increment in the flow. The warped image will be considered as the input for the next pyramid level, which is used to compute the next optical flow increment. This method does not involve explicit correlation calculations but relies on the learned capabilities of the CNNs to capture motion information. The sub-images are upsampled using bi-linear with the integrated optical flow. The resolution of ΔOF will also be refined after upsampling to form the coarse-to-fine process.

Sun et al. proposed the cost volume to convey the correlation between the two frame features as a cost volume [15]. Their *PWC-Net* is significantly smaller and easier to train compared to other models like *FlowNet2*. The cost volume the authors proposed is crucial because it stores the data-matching costs for associating a pixel with its corresponding pixels in the next frame in the neighbouring area. This is a fundamental component for stereo matching, which has motivated many subsequent works to connect computations across different layers of the feature pyramid.

Zhang et al. introduced a separable cost volume module as a replacement for traditional correlation cost volumes [16]. This module uses non-local aggregation layers to incorporate global context cues and prior knowledge.

Huang et al. discussed a masked cost-volume tokenization process, which involves dividing the raw cost map of each source pixel into patches through convolutions [17]. The core component of *FlowFormer* is its transformer-based cost-volume encoder. This step is essential for ensuring that the encoder processes only visible features during pre-

training, thereby maintaining consistency with the standard tokenization method used in *FlowFormer* for fine-tuning.

Transformer is also a popular method in optical flow estimation. Xu et al. proposed a transformer-based approach to globally compute the correlation between frames to estimate the optical flow [18]. Xu et al. developed the strategy with a cross-attention mechanism to estimate optical flow from posed images [19].

Teed et al. used a GRU to estimate optical flow by comparing the context features extracted from the original images with the current optical flow warped features [20]. Their *RAFT* extracts per-pixel features and constructs multi-scale 4D correlation volumes for all pixel pairs. The flow field is then iteratively updated through a recurrent unit that performs lookups on these correlation volumes.

Coarse to fine Pyramid is the most successful method in optical flow estimation with deep learning network [21], [22]. Although a deep pyramid can capture deeper features, it may lead to errors when the displacement is small [23]. Deep pyramid layers can aggregate too many pixels, which may significantly reduce the available information.

3.2 Video frames interpolation

The primary objective of VFI models is to generate intermediate frames between the original frames of a video sequence, thereby enhancing the visual smoothness and increasing the frame rate. The most conventional approach involves interpolating a single frame be-

tween two consecutive input frames, effectively doubling the frame rate. This method has been widely adopted due to its simplicity and effectiveness in improving video quality. However, recent advancements in VFI techniques aim to achieve more flexible and accurate interpolation across arbitrary timesteps, addressing the limitations of fixed frame rate doubling.

3.2.1 Flow free method

Tarun Kalluri et al. propose an optical flow-free end-to-end method to implement interpolation [24]. Their *FLAVR* achieved great performance with a 3D convolution kernel and used a U-Net-like structure to interpolate 6 frames in one single inference.

Meyer et al. proposed a linear blending technique for amplitudes to avoid popping artifacts, enhancing the visual quality of the interpolated images [25]. The interpolation process involves adjusting phase differences while blending amplitudes and low-frequency residuals to create smooth transitions between frames.

Shen et al. proposed *BIN* [26]. The core of the method is the pyramid module, which is capable of cyclically synthesizing clear intermediate frames. This module features an adjustable spatial receptive field and temporal scope, allowing for controllable computational complexity and restoration ability. It integrates a recurrent module to iteratively synthesize temporally smooth results without significantly increasing the model size.

3.2.2 Flow guided method

The *XVFI-Net* [27] is based on a recursive structure for bi-directional optical flow learning between two input frames to solve large motion interpolation. The scale adaptive structure can estimate optical flows from any desired coarse scale level based on motion magnitudes. The authors also introduced a high-quality dataset, X4K1000FPS, specifically designed for extreme VFI tasks, featuring 4K videos with significant motion and occlusions.

To reconstruct the intermediate frame, the optical flow-based model requires frame warping to generate the output. The two warping methods are forward warping and backward warping.

Backward warping

Jiang et al. presented a comprehensive approach to video interpolation that combines flow computation and interpolation networks [28]. Their *Super SloMo* divides the VFI task into flow estimation and intermediate frame reconstruction, which has become the basic structure of modern VFI models. The architecture also employs a U-Net design, allowing for effective learning of both optical flow and visibility maps, which are crucial for high-quality frame interpolation.

Bao et al. introduced a novel approach that leverages depth information to improve video frame interpolation [29]. They extract depth features, context features, optical flow features, and kernel features directly from the input frames. The depth features are used to enhance the estimation of optical flow. Backward warping is employed to warp the

original input frames, depth maps, and contextual features in a warping layer, preparing these features for the final U-Net-based frame synthesis.

Yu et al. introduced a Bayesian framework for the VFI task [30]. The Bayesian framework is implemented by learning posterior distributions of the intermediate optical flows and frames. This is achieved through a process of learning gradient descent, which allows for fast convergence within a few iterations.

Forward warping

Niklaus et al. used forward warping to synthesize intermediate frames [31]. To address the ambiguities in forward warping, the authors introduced a novel method called softmax splatting. This technique enables differentiable forward warping and improves the synthesis of interpolated frames by using importance metrics to clearly separate overlapping regions. The method employs an off-the-shelf optical flow estimator to derive flow fields, which are then used in conjunction with softmax splatting for forward warping the input frames.

Jin et al. proposed a network that uses a recurrent pyramid to predict intermediate frames [32]. Their *UPR-Net* combines bi-directional flow estimation and frame synthesis in a unified pyramid recurrent network. The iterative refinement is conducted in a coarse-to-fine manner, which is beneficial for high-resolution image synthesis. The number of pyramid layers may decrease if the input frames are too small, allowing the network to adapt to the resolution and mitigate detail loss.

The *UPR-Net* network also leverages bi-directional flow estimation at each pyramid

level, which helps in generating more accurate forward-warped representations. This process is iteratively refined, ensuring that the final synthesized frame is of high quality, even in challenging large motion scenarios

3.3 Distillation

Distillation has been widely recognized as an effective and highly regarded technique and serves as a strategic approach to significantly reduce the complexity and size of deep learning models while simultaneously enhancing their overall performance and efficiency. Hinton et al. proposed knowledge distillation to transfer knowledge from a large, complex model, the teacher model, to a smaller, more efficient model, the student model. During inference and deployment, only the student model is utilized [33].

Feature distillation can be utilized in computer vision tasks through a teacher-student learning paradigm, enhancing video classification by transferring knowledge to improve performance with less annotation [34].

Inspired by [35] and [33], we choose to use feature distillation. RIFE includes ground truth in the teacher training set to estimate a better optical flow.

The *RIFE* model provides the ground truth intermediate frame to the teacher model, which simplifies the estimation of $OF_{0\to0.5}$ and $OF_{0.5\to1}$ from I_0 and I_1. The teacher model may be able to estimate the two intermediate optical flows more precisely and easily. Subsequently, the distillation process transfers the features of the teacher model to the

student model. Moreover, the distillation aids the convergence of optical flow estimation. The teacher model should predict the intermediate frame as the student model does, which forms a computation between the prediction between the student model and the teacher model to compute the distillation loss.

Ziwei Liu et al. align teacher and student features through channel-wise transformation, improving performance in tasks like image classification, object detection, instance segmentation, and semantic segmentation [36].

3.4 Temporal structure

Some have combined video frame interpolation and video frame super-resolution into a single task and tried to leverage the temporal and spatial connections between video frames [37–39]. They proposed their method based on the belief that continuous space-time video super-resolution (VSR) simultaneously enhance spatial resolution and temporal information of video sequences.

However, the temporal importance is not emphasized in most computer vision tasks, which degrades the video problem to a 2-image problem. Many papers discuss the arbitrary time stamp interpolation [40]. Arbitrary timestamp training typically adheres to the same structure as intermediate timestamp training [35]. Arbitrary timestamps are generated by configuring a random time gap parameter to select the GT from the inner frames of a video sequence, using the first and last frames as inputs. The optical flow is subsequently

adjusted according to the time gap parameter to accurately capture motion at the chosen timestamp.

LSTM structure has been applied in many computer vision tasks, most of which perform a coarse-to-fine process to enhance the accuracy of the optical flow. *RAFT* used GRU to generate the optical flow from coarse to fine [20]. Khan et al. used GRU and attention to extract features from the input image[41].

Tatsunami et al. selected LSTM to capture long-range dependencies [42]. BiLSTM layers are used for spatial information, which process input sequences vertically and horizontally, thereby enhancing memory efficiency and throughput. Multilayer perceptrons facilitate the mixing of channel features.

Xiang et al. proposed a unified one-stage framework that simultaneously learns temporal interpolation and spatial super-resolution, overcoming the limitations of two-stage methods [37]. They use deformable ConvLSTM to align temporal features to convey the temporal features. This framework effectively leverages local and global temporal contexts, addressing large motion issues in video sequence

Chan et al. used bi-directional flow to estimate the video sequences frame by frame and pass the extracted temporal features to the next frame [38, 39]. Their *BasicVSR* method used optical flow to guide deformable alignment, which enhances the offset diversity while overcoming instability issues. This approach is motivated by the strong relation between deformable alignment and flow-based alignment, leading to sharper and more detailed feature alignment. The motion features of the video sequences are stored and updated

by the CNN blocks. In [39], they developed their original idea to connect the features from the adjacent two frames before and after the frame, which constructs the temporal relation of the video sequences. The updated *BasicVSR++* relaxed the assumption of the first-order Markov property in *BasicVSR* by adopting a second-order connection. This approach facilitates more robust information aggregation from different spatial-temporal locations, improving the model's performance in occluded and fine regions.

3.4.1 BiLSTM

Bidirectional long short-term memory (BiLSTM) is a powerful neural network architecture utilized in various NLP and signal processing tasks. It improves the network's capacity to capture context information from both past and future data inputs, rendering it particularly effective for sequential data analysis. In the CV domain, especially in VFI, BiLSTM has seen limited application due to insufficient emphasis on continuous video context information.

Wang et al proposed a BiLSTM method to address the challenge of classifying text sentiments. The optimal model integrates Word2Vec embeddings with a BiLSTM architecture, enhancing the ability to capture sequential text intricacies. The model's performance improves with the addition of more BiLSTM layers, allowing for better identification of contextual clues [43].

3.5 Artifacts in warped frames

Although warping has achieved considerable progress in VFI, artifact reduction in the final results remains a challenge.

One common method for blending the intermediate frame is to use bi-linear sampling, which can blur the frames and damage the quality of the result. In the arbitrary timestamp interpolation task, the blending parameter is used to control the position of the intermediate frame, which is adjusted with the required timestamp to balance the information from the warped frames [35].

Between two consecutive frames, a pixel from the previous frame may shift to a new location in the current frame. If no other pixel moves into the pixel's original position, the original area will not be updated during backward warping, leaving it unchanged. This results in the pixel being visible in both its new and original locations. This situation can lead to ghosting and artifacts [44]. Forward warping can mitigate this issue, but it also has some problems like leaving a hole in the image [45], which we will discuss in Chapter 4.

To generate a smooth and precise intermediate frame from the warped frames, context modules are utilized to provide additional information for the image generation blocks. The context features can accompany the warped frames for intermediate frame prediction [31, 45]. Huang et al. reused the warped frames to extract context information for the final reconstruction [35]. The output of the reconstruction block can be used by the warped

frames as a correction to refine the blending [32].

4

Temporal Pyramid Based Video Frame Interpolation

Our model follows the design of an optical flow-guided VFI network. We propose a TP block to provide additional temporal information, enhancing the details of the intermediate frames. Our approach achieves a state-of-the-art performance in PSNR.

4.1 Outline

We propose a GRU-based window clip model to leverage temporal information for a better intermediate frame construction. This model aims to enhance the quality of generated frames by effectively capturing the dynamics of motion and context within the window clip instead of two frames.

We made two assumptions to tackle the problem: 1). The motion between frames can be considered linearly moving, and 2). The contextual information within the window clip is crucial enough to predict the intermediate frame accurately.

Based on the two assumptions, we developed a novel architecture that integrates spatial and temporal features, allowing for a more nuanced understanding of the scene dynamics. The GRU structure aligns each frame one by one with channel attention to provide context information for the subsequent frame, enhancing the model's ability to focus on relevant details while minimizing noise from less critical elements. This approach improves the quality of the generated frames and ensures that the transitions appear smooth and coherent, reflecting the proper motion of the objects within the scene.

Optical flow has been chosen to guide the warping of frames to simulate linear motion, thereby enabling a more accurate representation of movement and facilitating the preservation of spatial relationships throughout the sequence. The optical flow estimation adheres to established methods, drawing on prior research to estimate optical flow solely from two adjacent frames. This approach utilizes proven techniques for capturing motion

dynamics between these specific frames, ensuring a more accurate representation of movement within the sequence. We compare forward and backward warping to determine which method produces superior visual fidelity and computational efficiency results, particularly in 4× and 8× slow-motion scenarios.

Our proposed temporal context-based U-Net structure combines temporal features with spatial information, enhancing the model's capability to model intricate details and transitions in motion, thus maintaining the consistency of objects during interpolation. The U-Net block follows the traditional layer combination in conjunction with our temporally aligned features across different layers of the feature pyramid. The overview of our method is illustrated in Fig. 4.1. The model takes a 4 frames window clip as input. The TP extracts the features using a pyramid CNN and extracts the temporal information with our bi-directional GRU structure. The extracted temporal features are passed to a channel attention block to emphasize the important channel inside it. The input frames will be sent to the optical flow estimator to generate the optical flow, which is used to backwarp the adjacent frames. These generated features will be aggregated in the final temporal pyramid U-Net to predict the intermediate frames. During training, we used feature distillation to optimize the training by comparing the difference between the GRU features from the teacher model and from the student model.

In the following, we will introduce data augmentation in Section 4.2, TP in Section 4.3, 4.4 and 4.5, image warping in Section 4.6, optical flow estimation in Section 4.7, distillation in Section 4.8, temporal U-Net in Section 4.9.

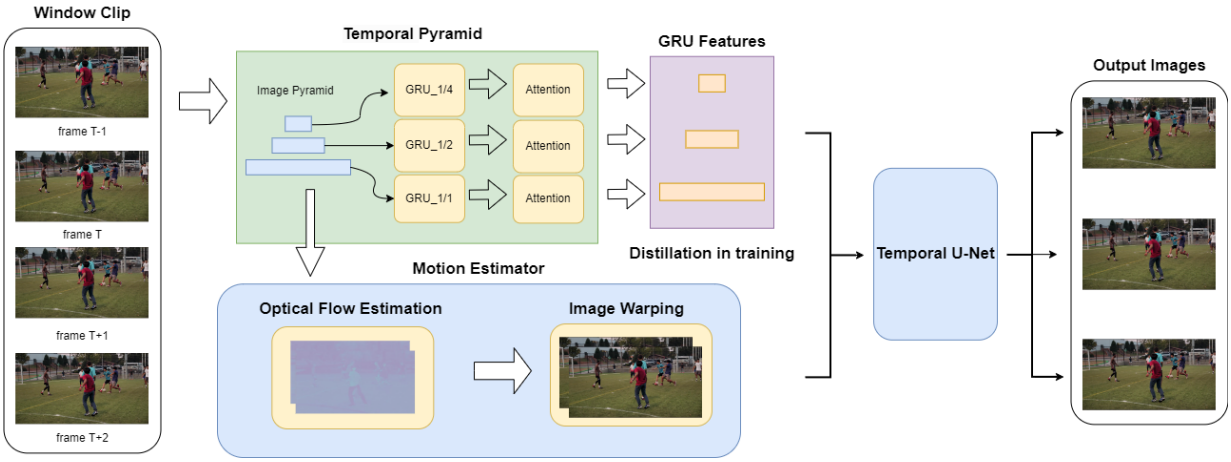


Figure 4.1: Overview of TP model.

4.2 Feature engineering and temporal stamp setting

4.2.1 Feature engineering

Feature engineering, such as normalization, resizing, and data augmentation, plays a crucial role in most DNN models; These techniques can help to reduce noise, enhance convergence speed, and mitigate overfitting, ultimately leading to improved performance.

We also applied preprocessing for our training dataset. The fundamental spatial transformation is used to extend the variety of the training dataset. The images of the training dataset will be flipped and rotated, and the values of RGB will be exchanged randomly to simulate different moving situations. Each training unit has seven frames in the Vimeo90K

Septuplet. In most traditional video interpolation models, the seven frames will be used three times to continuously take "one test, one ground truth, one test" combination as training input. However, this method cannot fully leverage the advantages of multi-frame datasets. This kind of usage just considers the 7-frame dataset as the superposition of the 3-frame dataset, which will waste the time continuity between the seven frames.

4.2.2 Temporal stamp setting

In VFI, the temporal stamp refers to the time interval between frames. In arbitrary interpolation tasks, the temporal stamp before and after the intermediate frame may vary, which lacks significance in real-world applications [27, 35]. Moreover, two-frame interpolation often fails to capture the temporal context of the video sequence. We discuss these limitations in Longer time period and Longer time stamp.

Longer time period

Most models in VFI only take three neighbouring frames as input, where GT is the middle one. This strategy has been proven to be useful in many models. However, when dealing with video sequences, the information between the frames may reveal the connection among the whole video sequence, which cannot be accessed if only three consecutive frames are considered. To address this issue, we introduce an RNN structure and a temporal window to connect multiple frames effectively. Dataset *Vimeo90K Septuplet* provides seven frames as an image sequence [46]. As a result, the temporal window is designed to comprise

seven frames, a configuration that will be applied across other datasets with long image sequences.

Longer time stamp

In other models targeting arbitrary time stamp interpolation, the index of the frames inside one training set may be shuffled so that the model can take inputs with different time gaps. However, arbitrary interpolation is not considered an essential target because the demand for motion at specific moments between two frames is not a necessary or practical requirement in modern industrial video processing. The requirement of interpolating multiple frames can be accomplished by recurrently applying the model to the newly generated frame sequence. However, using a 7-frame set to simulate an adjustable time period of motion is inspiring. As the first frame to be taken is fixed as the frame f_0, the time gap between the frames can be more than the default setting of the dataset.

The interval can be extended by skipping more than one frame between frame f_0 and frame f_1. This transformation increases the original small motion into a larger motion, making it more challenging for the model to process.

However, we choose to use a window clip and keep the one-image interval length to focus on the temporal context in our training

4.3 Bi-direction pyramid structure

The pyramid strategy has been an established technique in computer vision for years. The pyramid structure is often considered a type of multi-scale signal representation, effectively introducing information at various observation levels. These different layers of our TP are created by applying downsampling to the original image. In traditional computer vision, Gaussian and Laplacian pyramids have succeeded significantly and have been integrated into deep-learning neural networks to guide models. Researchers have also extensively worked with CNN blocks to leverage information across different resolutions. While these pyramid structures have been successfully applied to deep learning models, their primary focus on spatial hierarchies often limits their ability to capture the temporal dynamics crucial for our task. Therefore, we have used a feature pyramid to prepare the feature tensor for the GRU structure.

We design a three-level pyramid to capture temporal and spatial features at different scales. The bi-direction strategy is also applied to solve the occlusion of objects in the video sequence.

As a result, we choose to use bi-linear downsampling on the original input images to obtain feature tensors with dimensions of $H \times W$, $H/2 \times W/2$, and $H/4 \times W/4$, where H and W are the height and width of the original image. The CNN encoder will be applied to this multi-scale approach, allowing us to effectively integrate fine and coarse features, thereby enhancing the model's ability to comprehend complex motion patterns over time.

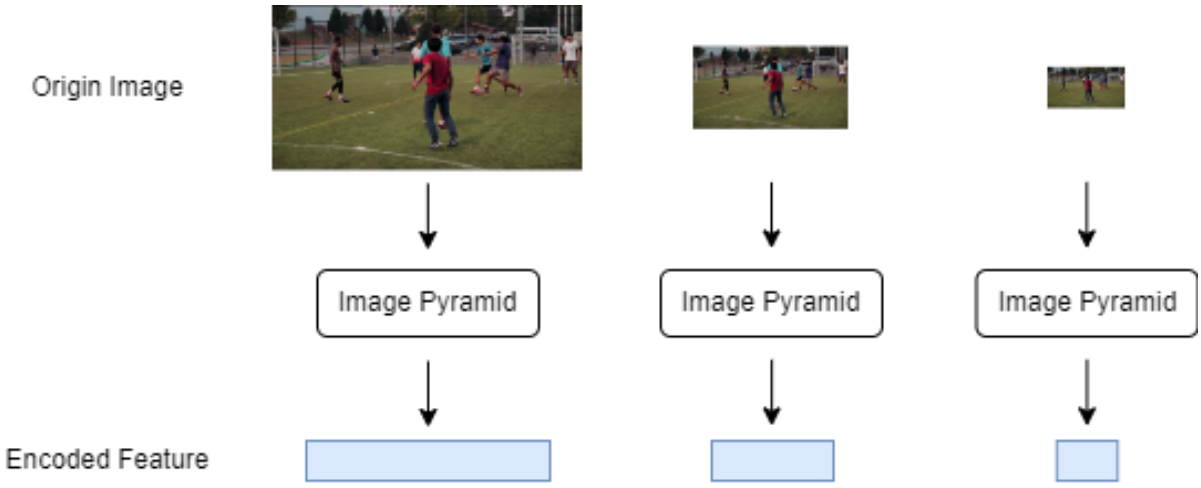


Figure 4.2: Image pyramid.

In each pyramid layer, we use 4 Convolution and Leaky Relu blocks as the pyramid encoder to capture the feature maps at different resolutions, ensuring that the model retains critical spatial information while progressively abstracting the data.

Following the other works on optical flow-based VFI models, we duplicate the feature extraction process with the reversed input order, which forms a back run of the video window clip. An object that becomes obscured over time will reappear from behind the object that was obscuring it when the sequence is played in reverse. This phenomenon can assist the model in capturing occlusion. If a region in the forward flow is inconsistent with the backward flow, it may indicate occlusion, where part of the scene becomes hidden or obstructed. By analyzing these discrepancies, the model can better infer the underlying structure of the scene and improve its predictions for future frames [47].

4.4 GRU

To connect the information from the whole window clip, we choose to use GRU to connect the images together and pass the hidden states to represent the temporal context information for the following blocks.

4.4.1 Why GRU

In the studies by Chan et al., [38] and [39], features extracted from images are manipulated using optical flow for the VSR task. This process involves integrating temporal information from two neighbouring frames into the current frame to enrich it with contextual details. This technique enhances the frame's informational content, providing a more robust basis for super-resolution processing.

However, due to the inherent differences between VSR and VFI, our approach encounters a specific challenge: it lacks access to the actual motion information of the intermediate frame. Consequently, the estimated motion cannot be verified directly using the current frame alone. This limitation necessitates alternative strategies to approximate motion with sufficient accuracy, impacting the overall efficacy of frame interpolation.

Consequently, it is crucial to meticulously manage the information that is incorporated into the model. Utilizing a forget gate is essential to selectively retain pertinent motion details while discarding irrelevant data. Therefore, we have opted to employ a GRU structure. This choice not only facilitates precise control over the context information but

also enhances the computational efficiency of the process. The GRU's architecture, which integrates update and reset gates, allows for a dynamic adjustment of information flow, thereby optimizing the handling of motion data for our specific requirements.

4.4.2 The usage of GRU

To effectively capture the temporal information of a video sequence and connect the frames within the clip window, we use a GRU to leverage the temporal context during inference. We note the process as Equ. (4.1).

$$H_n = \mathbf{GRU}(X, H_{n-1}), \tag{4.1}$$

where,

- X is the input frame at the current time step,
- H_n is the hidden state at the current time step, which will be passed to the next time step.
- H_{n-1} is the hidden state from the previous time step.

As previously mentioned, we consider four frames as the input set for our model. Correspondingly, three hidden states will be estimated in each direction of the GRU. At the same time, we omit the first output hidden state as it lacks the relationship between the two

frames we are processing and instead reflects the relationship between the zero initialization and the first frame. The remaining three hidden states provide motion information to refine the reconstruction of the intermediate frame in subsequent blocks. Additionally, the bi-directional architecture offers information on the reversed motion process. The process can be noted as Equ. (4.2).

$$\begin{aligned} H_f &= \mathbf{GRU}(X_0, X_1, X_2, X_3), \\ H_b &= \mathbf{GRU}(X_3, X_2, X_1, X_0), \end{aligned} \tag{4.2}$$

where,

- X_i is the input frame with index i, where $i = 0, 1, 2, 3$.
- H_f is the hidden state list that contains the three hidden states in the forward direction.
- H_b is the hidden state list that contains the three hidden states in the backward direction.

We only select three out of four hidden states in each direction because the hidden states describe the motion between two input frames. But the GRU takes zero and the first input frame as the first input, which does not contain the motion relation.

After extracting the six feature tensors from the window clip, we select only two hidden states—one from the forward estimation and one from the backward estimation—for the subsequent blocks of our model. We choose the hidden states that share the same index in

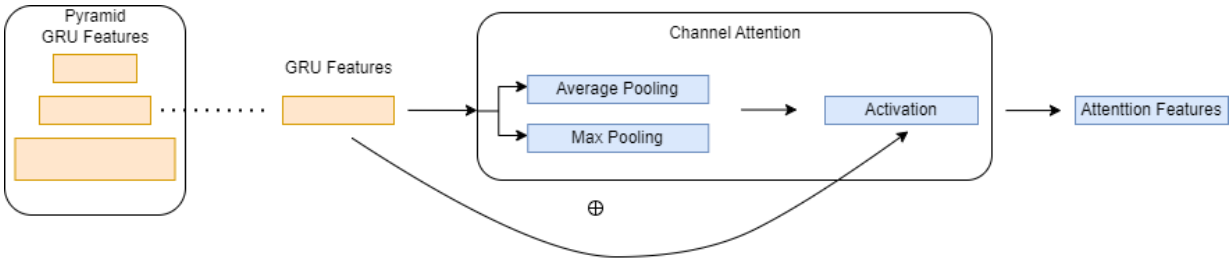


Figure 4.3: Channel attention.

the secret state list. This pair of frames can capture the entire motion of the window clip, as their memory is expected to contain the essential motion dynamics. This ensures that our model remains efficient while leveraging the complete sequence information. Information from the future and the past is isolated to prevent any potential leakage that might distort the learning process, allowing the model to focus exclusively on the relevant temporal features.

4.5 Channel attention

Attention block is one of the most popular modules in deep learning networks nowadays. Attention modules have been used successfully. The temporal features from the hidden states of GRU may not be appropriate for the following reconstruction process.

As a result, we choose to use channel attention to extract and activate the temporal features. The channel attention follows Equ. (4.3):

$$\begin{aligned} \mathbf{Att}_c(H) &= \sigma\left(\mathbf{AvgPool}(H) + \mathbf{MaxPool}(H)\right), \\ X_{att} &= H + H \times \mathbf{Att}_c(H), \end{aligned} \tag{4.3}$$

As shown in Fig. 4.3, we follow the design of Squeeze-and-Excitation Networks (*SE*) channel attention model [48] to build our channel attention block to enhance the representation of these hidden states, enabling the model to weigh the importance of different features dynamically. GRU features can be enhanced or weakened by this channel attention block.

In the channel attention block, the three layers of GRU features are separated and each of the features is passed through the max pooling and average pooling layer to compute the parameters of channel attention, which are used to compute the feature attention with the origin features as shown in Equ. (4.3).

4.6 Optical flow estimation

The optical flow-guided approach to VFI has demonstrated significant advancements in the field. In light of these achievements, we have elected to adopt this method as the foundation for constructing our model.

We evaluated the two most popular methods for estimating optical flow and opted to follow the design outlined in [35]. Additionally, we compared our model's performance with and without the use of pretrained parameters from *RIFE*. The details of the methods

and the comparison are outlined in the following sections.

4.6.1 Coarse-to-fine estimation

The coarse-to-fine estimation has been chosen in a lot of models[14, 32]. The idea of the coarse-to-fine structure is simple and can be shown as the following Algorithm 1.

Algorithm 1 Optical Flow Computation via Pyramid Levels.

Input: D_1, D_2 ▷ Initial images at the finest pyramid level
Output: OF ▷ Computed optical flow between D_1 and D_2
Initialize $OF \leftarrow 0$
for i in PYR_LEVEL **do** ▷ Iterate over each level of the image pyramid
    $\Delta OF \leftarrow \text{Corr}(D_{1i}, D_{2i})$ ▷ Compute correlation between downsampled images
    $D_{1(i+1)} \leftarrow \text{UpSample}(D_{1i})$ ▷ Upsample the first image to the next level
    $D_{2(i+1)} \leftarrow \text{UpSample}(D_{2i})$ ▷ Upsample the second image to the next level
    $D_{1(i+1)} \leftarrow W(D_{1(i+1)}, \Delta OF)$ ▷ Warp the upsampled first image using ΔOF
end for
$OF \leftarrow OF + \Delta OF$ ▷ Accumulate the computed optical flow

where,

- D_1 and D_2: Represent the original images that have been downsampled. These images are inputs to the algorithm and are processed through multiple levels of a pyramid structure.
- i: Index representing the current level of the pyramid. The level $i = 0$ corresponds to the images that have been downsampled PYR_LEVEL times.

- PYR_LEVEL: Denotes the number of layers in the pyramid. Each level corresponds to a different resolution of the images, with the highest level ($i = 0$) being the most downsampled version.

- **UpSample**: Upsampling function that upsamples the downsampled image to a higher resolution. Bi-linear interpolation is widely used.

- W: Represents the warping function. This function warps an image by applying a transformation defined by the delta optical flow ΔOF. The specific method of warping will be discussed in detail in subsequent sections.

- ΔOF: The delta optical flow computed between the corresponding downsampled images at the current pyramid level. This flow is used to adjust the alignment of images in subsequent steps.

- OF: The cumulative optical flow is computed as the sum of delta optical flows across all pyramid levels. This represents the overall motion between the initial images across the entire sequence of operations.

It is crucial to judiciously select the number of pyramid layers used in image processing. As noted by Xu et al. [49], downsampling associated with a deep pyramid layer inevitably results in a loss of information. Furthermore, if the motion between frames is smaller than the grid size of the downsampled resolution, accurate tracking of such motion becomes unfeasible. This limitation highlights the need for a balanced approach in determining the pyramid depth to preserve essential details while still enabling effective motion tracking.

Pyramid Layer	Slow	Medium	Fast
2	38.05/0.9861	35.21/0.97204	30.98/0.9325
3	38.22/0.9865	35.63/0.9734	32.55/0.9462
4	38.21/0.9864	35.60/0.9732	32.75/0.9475

Table 4.1: Qualitative (PSNR/ SSIM) test results across different motion speeds at various pyramid layers.

In [32], the author introduced an adaptive pyramid approach, which tailors the number of pyramid layers according to the resolution of the input images. We conducted tests on this method using images of consistent resolution from the Vimeo90k dataset [46]. The results are presented in Table 4.1

The images utilized in the experiments were of resolution 256×448. Based on the test results, it was observed that three pyramid layers yield the best performance. Utilizing deeper layers was found to detrimentally affect the final image quality. In our model, we choose three layers to balance the computation cost and the model performance.

4.6.2 Problem of fixed pyramid layer

Because our model is trained and validated on a single dataset, the optimal layer of the motion extractor is determined to be three. However, in real-world applications, the three-layer estimator may not always produce the best motion features, especially for large images. A dynamic layer selector may provide a better solution to this problem when the layer number is automatically chosen as the shape of the image changes [32].

4.6.3 RIFE

Based on the coarse-to-fine method, Huang et al. proposed this intermediate optical flow-based structure. Traditionally, it is assumed that the motion between two frames is linear, such that the optical flows from time t to $t_{0.5}$ and from $t_{0.5}$ to t_1 are equivalent to half of the optical flow from t to $t+1$. In contrast, their method directly estimates both $OF_{0\to0.5}$ and $OF_{1\to0.5}$ simultaneously, using these intermediate optical flows to warp the two frames respectively.

The modified algorithm can be shown as Algorithm 2.

Algorithm 2 Intermediate Optical Flow Estimation of RIFE.

Input: D_{0i}, D_{1i} ▷ Initial images at the finest pyramid level
Output: $\Delta OF_{0\to0.5}, \Delta OF_{0.5\to1}$ ▷ Optical flows for each half interval
for i in PYR_LEVEL **do** ▷ Iterate over each pyramid level
    $\Delta OF_{0\to0.5}, \Delta OF_{0.5\to1} \leftarrow \text{Corr}(D_{0i}, D_{1i})$ ▷ Compute correlations
    $D_{0(i+1)} \leftarrow W(D_{0(i+1)}, \Delta OF_{0\to0.5})$ ▷ Warp image D_{0i} using $\Delta OF_{0\to0.5}$
    $D_{1(i+1)} \leftarrow W(D_{1(i+1)}, \Delta OF_{1\to0.5})$ ▷ Warp image D_{1i} using $\Delta OF_{1\to0.5}$
    $D_{0(i+1)} \leftarrow \text{UpSample}(D_{0i})$ ▷ Upsample image D_{0i}
    $D_{1(i+1)} \leftarrow \text{UpSample}(D_{1i})$ ▷ Upsample image D_{1i}
end for
$\Delta OF_{0\to0.5} \leftarrow \sum \Delta OF_{0\to0.5}$ ▷ Sum optical flows for final $0 \to 0.5$
$\Delta OF_{0.5\to1} \leftarrow \sum \Delta OF_{0.5\to1}$ ▷ Sum optical flows for final $0.5 \to 1$

4.7 Motion-based frame warping

The optical flows estimated in the previous blocks are used to guide the motion estimation of the intermediate frames. There are usually two kinds of warping methods in VFI tasks: forward warping and backward warping. We have applied the tests for both forward warping and backward warping for our model to get a better final intermediate frame prediction.

It is also important to choose what should be warped by the optical flow. Some authors choose to warp the original input frames [35], which preserves the connection between the warped images and the real-world inputs. However, feature warping has become a more popular option in recent approaches due to its greatly improved results [32].

We conducted tests to evaluate the choice of warping methods and warping tensors. The results will be presented in Chapter 5.

4.7.1 Backward warping

Backward warping is a crucial technique used in video frame interpolation, particularly in deep-learning models designed to generate intermediate frames between existing video frames. The key idea of backward warping is to map pixels from the target frame, the intermediate frame in our task, to the source frames, the input frames, rather than the more intuitive forward warping, where pixels are mapped from the source to the target.

The optical flow **OF** should be preserved as $\tilde{\mathbf{OF}}$ during backward warping to track the pixels in the source frames. We then add $\tilde{\mathbf{OF}}$ and the index grid **G** to obtain the sampling guidance map $\mathbf{G}_w$, which has the same height and width as the input tensors. Using the guidance map, the pixels in the target tensor can select corresponding pixels from the source tensor. Since the optical flow values may be decimal, this can result in non-integer indices within the source tensor. Consequently, bi-linear sampling is applied to obtain the final prediction.

The process can be described as Equ. (4.4):

$$\begin{aligned} \tilde{\mathbf{OF}} &= -\mathbf{OF}, \\ \mathbf{G}_w &= \tilde{\mathbf{OF}} + \mathbf{G}, \\ W &= \mathbf{G}_w(\mathbf{T}_s), \end{aligned} \tag{4.4}$$

where,

- $\mathbf{G}_w$ is the basic index grid.
- $\mathbf{T}_s$ is the source tensor.
- W is the warped tensor.

The process can be illustrated in Fig. 4.4

The value of the blue pixel in Fig. 4.4 is derived from the red pixel. The optical flow vector (a, b) denotes the warping process. The center point is shifted according to the reversed optical flow to acquire the red pixel.

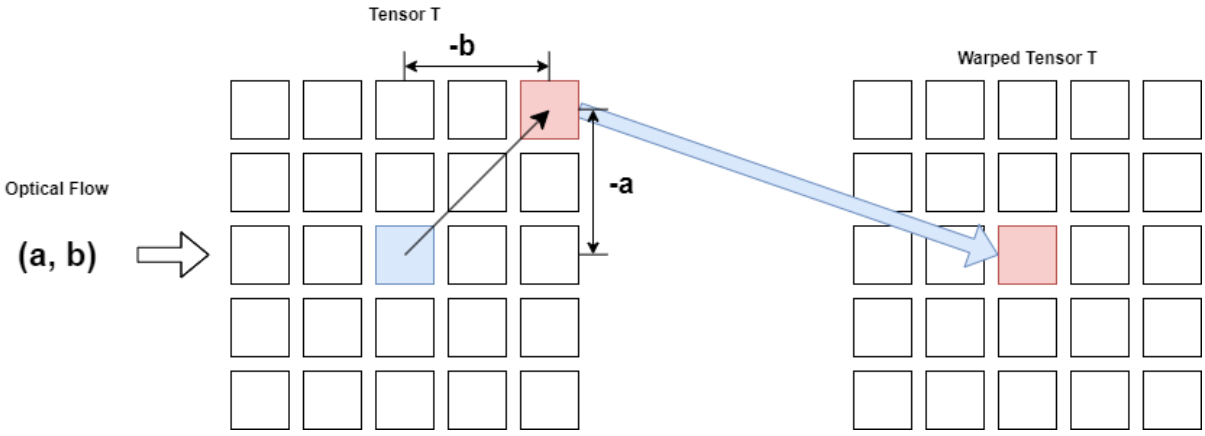


Figure 4.4: Optical flow backwarping.

Backward warping avoids this issue since every pixel in the target frame is explicitly mapped to a source pixel. It is generally more efficient in computational terms since it avoids redundant calculations and can be implemented in a way that minimizes interpolation errors.

4.7.2 Forward warping

The value of the warped pixel is derived from the center pixel in Fig. 4.5. The optical flow vector (c, d) denotes the warping process. The pixel at the center point is moved to the pixel at the corner according to the optical flow to acquire the target pixel.

Forward warping is another technique used in video frame interpolation, which involves directly mapping pixels from source frames to the target frame. While it may seem more intuitive, forward warping presents some unique challenges, particularly when implemented

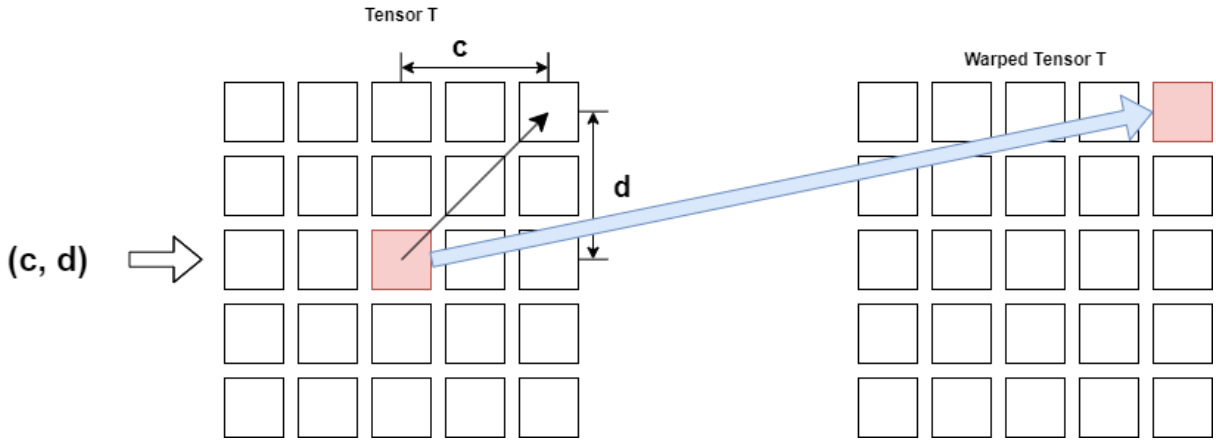


Figure 4.5: Optical flow forward warping.

in deep learning models.

Instead of pulling pixels from the target frame, forward warping pushes pixels from the source frames to the new frame. Forward warping is conceptually simpler and aligns more directly with the process of motion in physical space.

The process can be described as Equ. (4.5):

$$\begin{aligned} \mathbf{G}_w &= \tilde{\mathbf{OF}} + \mathbf{G}, \\ W &= \mathbf{G}_w(\mathbf{T}_s), \end{aligned} \tag{4.5}$$

Because of the direct selection from the source frames, not every pixel of the target frame may be filled. Holes may appear in areas where pixels from the source frames cannot be warped. This issue arises for the following reasons [31, 32]:

1. Occlusion between frames: Certain pixels may be absent in other frames, making it

impossible to align a pixel to another location.

2. Overlap: In cases of motion such as bending, objects and pixels may stack at the same position, resulting in blank areas where there are no corresponding pixels.

3. Coarse optical flow: The estimation module cannot guarantee the correctness of every single value of the optical flow, potentially leading to errors in the warping process.

However, the benefits of forward warping are obvious. Forward warping can be particularly effective when combined with a sophisticated optical flow estimation that accurately captures the movement between frames, which is critical in real-time applications.

The process of forward warping can be illustrated in Fig. 4.5

4.8 Distillation

As discussed in Chapter 2, the hidden states of the GRU structure are utilized to capture the motion between two consecutive frames. However, this motion represents a coarse transition from time $t-1$ to time t, which does not directly correspond to the intermediate frame at time $t_{0.5}$.

Based on this idea, we can extend the simulation to include more information in the procedure. We upgraded the origin four-frame motion to a seven-frame motion, which halves the time gap between each motion to form a more detailed motion estimation.

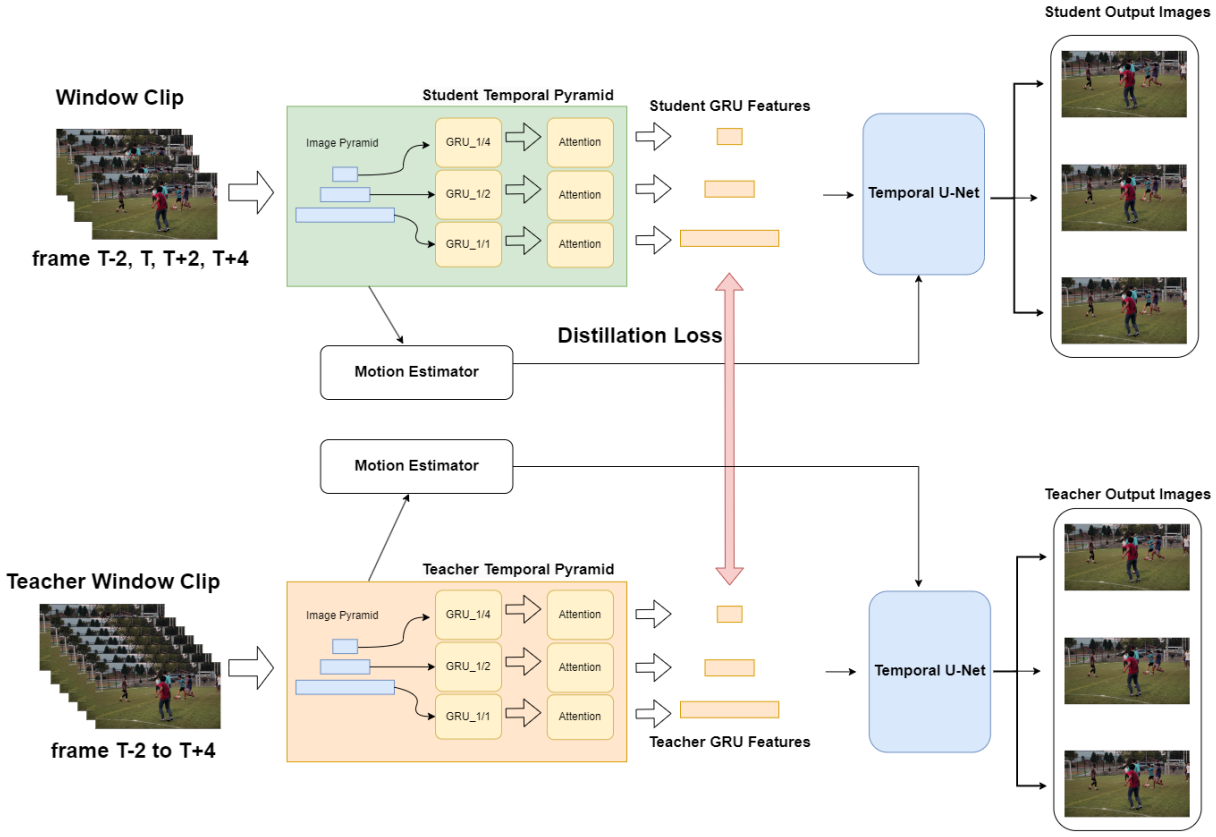


Figure 4.6: Distillation structure.

The process of distillation is illustrated in Fig. 4.6. As shown in the left part of the figure, the teacher model takes all the seven input frames, both the images that the student model receives and GT. The features of GRU will be compared to pass the distillation loss $\mathcal{L}_{distill}$ to update the parameters. Other parts of the model are the same but trained separately. The teacher model needs to generate the intermediate frame and to be compared with GT to update as the student model does. In the inference, only the student model is utilized.

To enhance the precision of the motion simulation, knowledge distillation can be employed to transfer information from a transfer set, which contains the ground truth of the interpolation, to the distilled model. The transfer set can incorporate all seven frames of the training septuplet to train the teacher model. Consequently, the GRU structure can represent a more detailed motion process, accounting for the transitions from time $t-1$ to time $t-0.5$ and from time $t-0.5$ to time t. The process of teacher GRU block can be described as:

$$HTea_{t-1\to t-0.5} = \text{GRU}(GT, HTea_{t-1}), \tag{4.6}$$

$$HTea_{t-0.5\to t} = \text{GRU}(X, HTea_{t-1\to t-0.5}), \tag{4.7}$$

$$HTea_t = HTea_{t-0.5\to t}, \tag{4.8}$$

where,

- GT is the ground truth frame of the current interval.
- $HTea_{t-1}$ is the hidden states in the previous interval between time $t-2$ and time $t-1$.
- $HTea_{t-1\to t-0.5}$ is the hidden state between time $t-1$ and time $t-0.5$, which introduce the $t-0.5$ by taking GT as input.
- $HTea_{t-0.5\to t}$ is the hidden state between time $t-0.5$ and time t.

We duplicate the structure to create both the teacher and student models within our overall architecture. Except for the GRU block, all other components remain identical.

The hidden state of the feature at the final time step will be used to compute the distillation loss. This process aims to minimize the discrepancy between the outputs of the student GRU block and the teacher GRU block, thereby guiding the student GRU's weight matrix to effectively predict the entire motion trajectory without relying on the intermediate frame.

Due to the variations in channel dimensions across layers, the distillation loss must be weighted appropriately to achieve balance. The objective is to ensure that each layer contributes equally to the distillation process. Therefore, the distillation function is defined

as:

$$\begin{aligned}\mathcal{L}_{\text{distill}} = &(H_{\text{Tea}_{d0}} - H_{d0}) \\ &+ 0.5 \times (H_{\text{Tea}_{d2}} - H_{d2}) \\ &+ 0.25 \times (H_{\text{Tea}_{d4}} - H_{d4}),\end{aligned} \tag{4.9}$$

The $\mathcal{L}_{\text{distill}}$ will contribute to the final loss function during updates. A parameter α will be applied to control the influence of the distillation process. Following previous works, we set it to 0.001.

The teacher model also needs to estimate the intermediate frame as the student model does to compute the difference between the teacher-predicted image and the ground truth, which is used to compute the loss $\mathcal{L}_{\text{tea}}$. The quality of the teacher model cannot be guaranteed if the teacher model is not updated with the student model.

4.9 Temporal pyramid U-Net

The structure of our temporal U-Net is illustrated as Fig. 4.7. The traditional U-Net structure is limited to using the information within the input features, which may fail to incorporate features from other types of information. The features to be processed can only be introduced at the beginning of the U-Net structure. Models like *RIFE* [35] have used a context block that reuses current frames and optical flow, but this approach still cannot be independent of what has already been sent to the U-Net block, potentially leading to

wasted computation and redundant features. Therefore, we aim to provide the base U-Net structure with more extensive features.

We connect the resolution changes between the pyramid layers and the U-Net steps to incorporate temporal information, thereby optimizing the reconstruction of the intermediate frame. The entire process can be described as Equ. (4.10):

$$\begin{aligned} \text{U-Net}_0 &= \text{CNN}(I_t, I_{t-1}, I'_t, I'_t, OF_f, OF_b), \\ \text{U-Net}_n &= \text{CNN}(\text{concat}(U - Net_0, H_n)), \end{aligned} \tag{4.10}$$

In the upper part of the U-Net block, the dimensions of the features are halved, while the lower part concatenates the corresponding layers from the upper block. Both the forward and backward temporal GRU features are incorporated into the U-Net layer, connecting the features of the current adjacent frames with the entire video window clip. With this architecture, the optical flow warping can be isolated by the temporal features, while the final reconstruction can still access all contextual information.

Overall our temporal U-Net takes the corresponding temporal features as input. The downsampling of the U-Net structure will fit the different resolutions of the pyramid layers.

4.9.1 Output refinement

The difference between our synthesis method and the alignment-based synthesis method is illustrated in Fig. 4.8. The alignment based method estimates the refinement using features extracted and processed by other modules. This refinement is then added to the blended

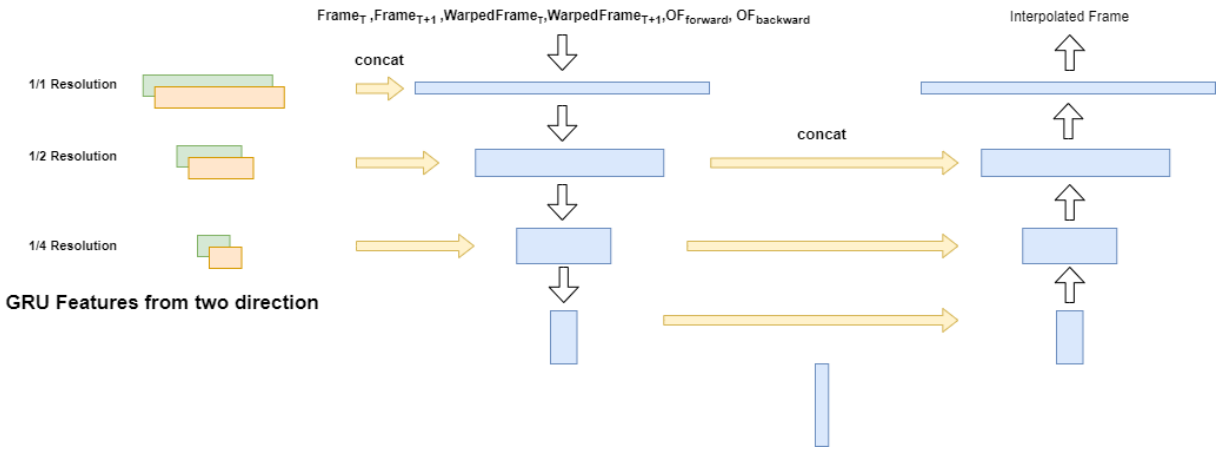


Figure 4.7: Temporal U-Net.

result of the two warped images to synthesize the final intermediate output. Consequently, the quality of the optical flow used in warping has a significant impact on the quality of the outputs.

The output range of a neuron's computation, such as $(-1, 1)$ or $(0, 1)$, is determined by the activation function employed within the neural network [50]. Commonly, RGB pixel values, which originally range from $(0, 255)$, are mapped to $(0, 1)$. This normalization of pixel values is crucial as it enhances the numerical stability of neural networks, improves the efficiency of learning, and contributes to the robustness of the training models. Normalization ensures that the input data are on a standardized scale, which is particularly beneficial in handling variations in input data and expediting the convergence during training [51]. Additionally, the blending process can introduce blurriness due to the overlap of the two images, an issue that becomes more pronounced in videos with large motion.

Some models employ a strategy where the overlay of two aligned images is computed,

followed by adding the result of the final synthesis block [35, 52]. In alignment-based methods, adjacent frames are aligned and reconstructed separately. The results from these computations are then aggregated to produce the final intermediate frame. In contrast, our method processes the intermediate features consecutively to derive the final intermediate frame, allowing for a more integrated and continuous transformation across frames.

However, this method introduces potential issues in the final result estimation.

- While this approach can reduce computational costs and is straightforward, it may lead to value overflow issues. Specifically, since both the blended image and the computation result are scaled between 0 and 1, simply adding them can result in values exceeding this range, thus leading to overflow or out-of-bounds errors. In our experiments based on the methodology of [52], the maximum value observed in the output was 1.2, which exceeds the nominal upper boundary by 20%. To adapt the tensor for image conversion, clamping is employed to restrict the value range to conform to standard image formats. However, this clamping approach leads to significant information loss. More critically, it disrupts the intrinsic relationships among pixels, potentially resulting in substantial degradation of image quality.

- The additive contribution of the final synthesis block may be relatively minor, potentially exerting an insufficient influence on the base-aligned image. Consequently, the quality of the output is predominantly dependent on the image alignment method.

To solve this problem, we have opted to integrate a temporal U-Net block directly within a CNN for reconstructing the final result. The chosen activation function for this

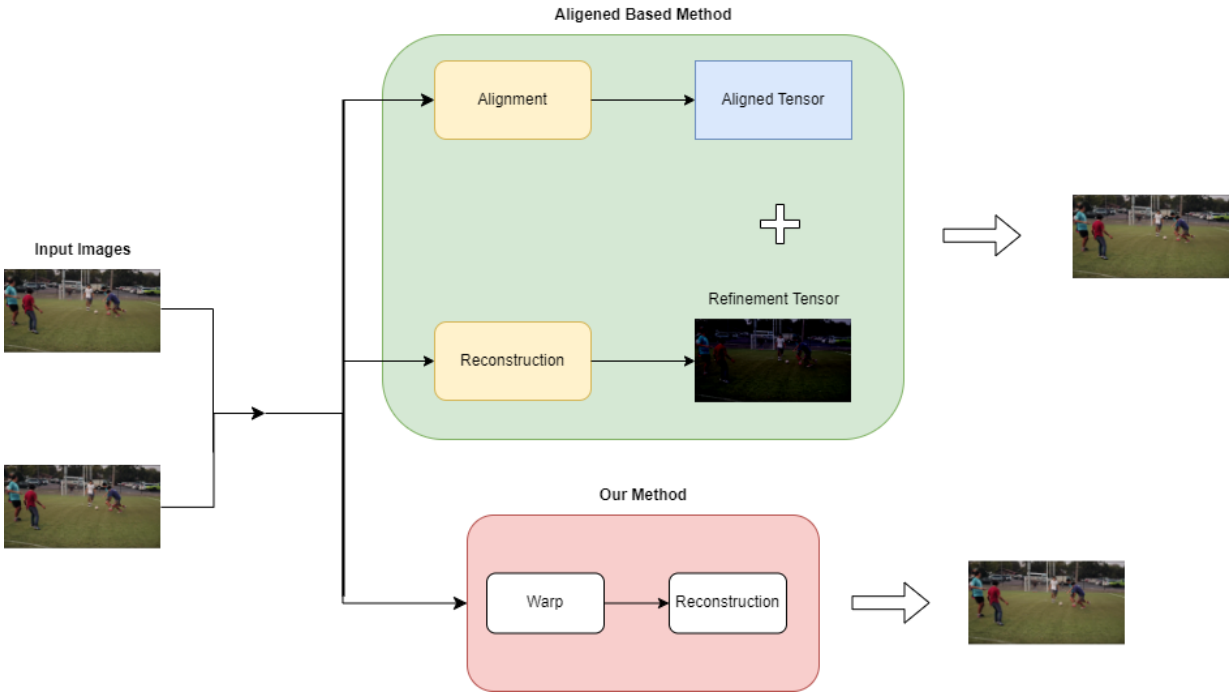


Figure 4.8: Synthesis method comparison.

configuration is the sigmoid function. This approach allows for effective management of output values, ensuring they remain within the desired range of $(0, 1)$, thereby preserving the integrity and quality of the image data[53].

5

Experiments

In this chapter, we will discuss the details of our training experiments and validation tests. Ablation tests are conducted to evaluate the effect of the TP on the VFI task.

5.1 Implementation details

5.1.1 Training strategy

We train our model on the Vimeo 90K training set. We chose AdamW with weight decay 10^{-4}. The training for the final version of our model is conducted in NVIDIA RTX GeForce 24GB. To balance training speed and performance, the training data set is cropped to frame patches 224×224 to accelerate training speed with batch size 20. The images of the training dataset are flipped and rotated, and the values of RGB can be exchanged randomly to simulate different moving situations.

As we mentioned in Section 4.6, the optical flow estimation can be greatly influenced by the layer number of the estimation pyramid. In our three-layer structure, the resolution of the coarsest layer only holds 18×18 pixels, which may only track general pixel motion and patterns, which may fail to estimate a clear optical flow map for large images.

5.1.2 Loss function

We choose to use a linear loss combination of the loss of the teacher model $\mathcal{L}_{tea}$, the loss of the student model $\mathcal{L}_{rec}$ and the distillation loss $\mathcal{L}_{distil}$.

$$\mathcal{L} = \mathcal{L}_{tea} + \mathcal{L}_{rec} + \lambda_d \mathcal{L}_{distill}, \tag{5.1}$$

We set $\lambda_d = 0.01$ in order to balance the scale of losses.

λ_d should not be set to a very high value to avoid overfitting. Because the teacher model and student model do not actually process the same input data, it might damage the quality of the final output if the student model is forced to learn everything from the teacher model.

The $\mathcal{L}_{tea}$ and $\mathcal{L}_{rec}$ are combined loss of robust L1 loss and Census transform loss which follow:

$$\mathcal{L} = \mathcal{L}_{robust} + \mathcal{L}_{Census}, \tag{5.2}$$

The $\mathcal{L}_{robust}$ is less sensitive to outliers than the standard L2 loss:

$$\mathcal{L}_{robust} = \frac{1}{N} \sum \sqrt{(I_{pi} - GT_i)^2 + \epsilon^2}, \tag{5.3}$$

where,

- I_{pi} denote the pixel value at position p in predict image I_p.
- GT_q denote the pixel value at a neighboring position q in ground truth image GT within the window.

$\mathcal{L}_{Census}$ is the Census transform loss of the predicted image and the ground truth. The Census transformation follows:

$$\text{Census}(\mathbf{I})_p = \prod_{q \in \mathcal{N}(p)} \mathbf{1}(\mathbf{I}_q < GT_p), \tag{5.4}$$

where,

- q and p are the positions of pixels in the image.
- $\mathcal{N}(p)$ represents the set of all neighbouring pixels q within the window centred at p.

The Census loss is the average Hamming distance between the Census-transformed patches of the predicted image I_p and the ground truth GT:

$$\mathcal{L}_{\text{Census}} = \text{avg} \sum (\text{Census}(I_p) - \text{Census}(GT)), \tag{5.5}$$

5.1.3 Dataset

We used *Vimeo90K Septuplet* [46] for training and used *Vimeo90K*, *X4K1000FPS* [27], *SNU-FILM* [54], and *Parkour* [55–57] to compare the performance with state-of-the-art models.

Vimeo90K Septuplet

Our model is trained on the *Vimeo90K* septuplet dataset with the clip window setup. The septuplet dataset contains 91,701 sequences, each consisting of 7 consecutive frames. The three even-numbered frames are used as ground truth. We followed the test settings from the original test list and [58] to evaluate the performance of our model across different movement speeds.

X4K1000FPS

X4K1000FPS dataset is a high-frame-rate video dataset designed for tasks such as VFI and VSR [27]. It contains $4K$ resolution videos captured at 1000 fps. We clipped the same 384×384 area of each image and separated the original dataset into a series of 7-frame groups to do the comparison.

SNU-FILM

SNU-FILM dataset is designed for training and evaluating video frame interpolation models. It consists of high-quality videos across various frame rates, resolutions, and scenes [52].

SNU-FILM can be separated into a series of tests based on the time stamp. However, we simply separate the original dataset into a series of 7-frame groups to suit our window clip setting.

Parkour

The Parkour dataset contains 28 RGB videos capturing human subjects performing four typical parkour techniques. These are highly dynamic motions with rich contact interactions with the environment [59], which makes it very challenging for VFI tasks. We clipped the same 384×384 area of each image and separated the original dataset into a series of 7-frame groups to do the comparison.

5.2 Temporal feature validation

The innovative aspect of our work lies in the utilization of a temporal pyramid to augment the network with continuity information of images. Subsequently, we demonstrate the network's effectiveness by comparing the features before and after entering the temporal pyramid.

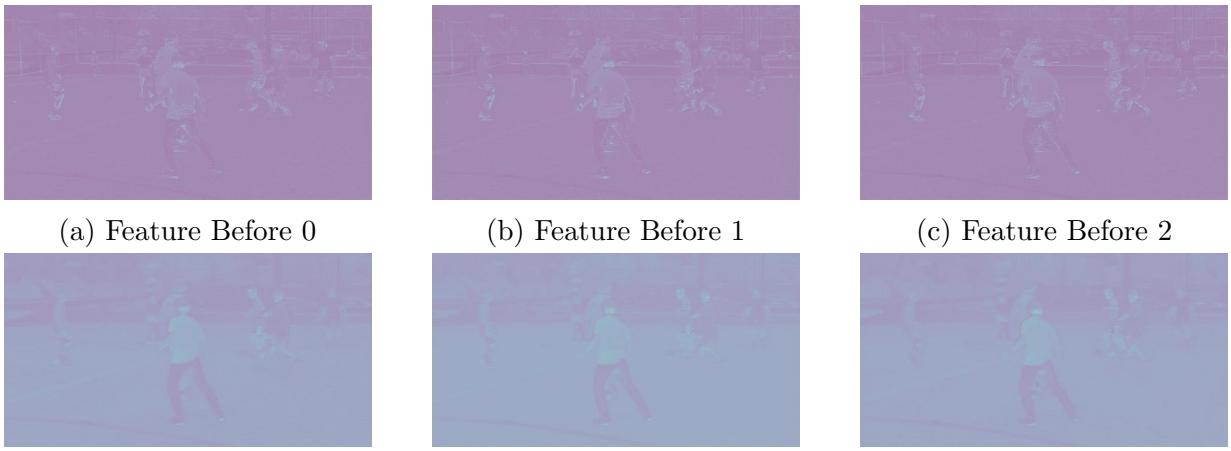
(a) Original Image 0 (b) Original Image 1 (c) Original Image 2 (d) Original Image 3

Figure 5.1: Original input images for temporal feature comparison at the original resolution.

In Fig. 5.1, we show the original images, Frame 0, Frame 1, Frame 2 and Frame 3, for the temporal feature comparison. We use three different features, each corresponding to one of the three temporal intervals. We compare the difference between the feature heatmap before and after entering TP.

In the original resolution heatmaps shown in Fig. 5.2 and Fig. 5.3, we observe an emphasis on the finer details of the patterns. Notably, in Fig. 5.3, the pants in motion appear significantly brighter than the background, aiding the network in distinguishing moving objects from the static background. This contrast facilitates the estimation of a clear intermediate frame. The feature differences in the original resolution are primarily associated with the objects in the frames. The areas where the soccer players and trees

are located have brighter colours, indicating that the temporal features have effectively captured these objects.



(a) Feature Before 0 (b) Feature Before 1 (c) Feature Before 2

(d) Feature After 0 (e) Feature After 1 (f) Feature After 2

Figure 5.2: Comparison of backward temporal features before TP and backward features after TP at the original resolution.

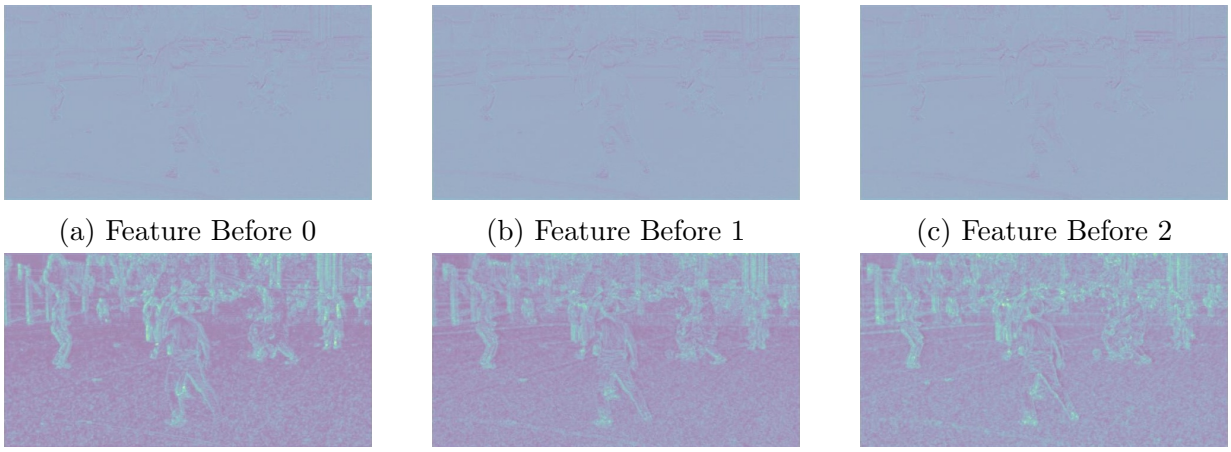
In the 1/4 resolution heatmaps, Fig. 5.4 and Fig. 5.5, the boundaries are emphasized, which can help the network to maintain the consistency of the objects. The temporal feature at quarter resolution captures more detailed boundary information. When comparing the two legs of the soccer player positioned at the center of the image, it is evident that the right leg appears brighter than the left, as the captured sequence shows significantly more motion in the right leg compared to the left.

Because our model is specially designed for video sequence interpolation, we cannot use the commonly used datasets, which consist of triplet image groups. We make comparisons with the state-of-the-art models on the valid dataset of Vimeo90K Septuplet.

(a) Feature Before 0 (b) Feature Before 1 (c) Feature Before 2

(d) Feature After 0 (e) Feature After 1 (f) Feature After 2

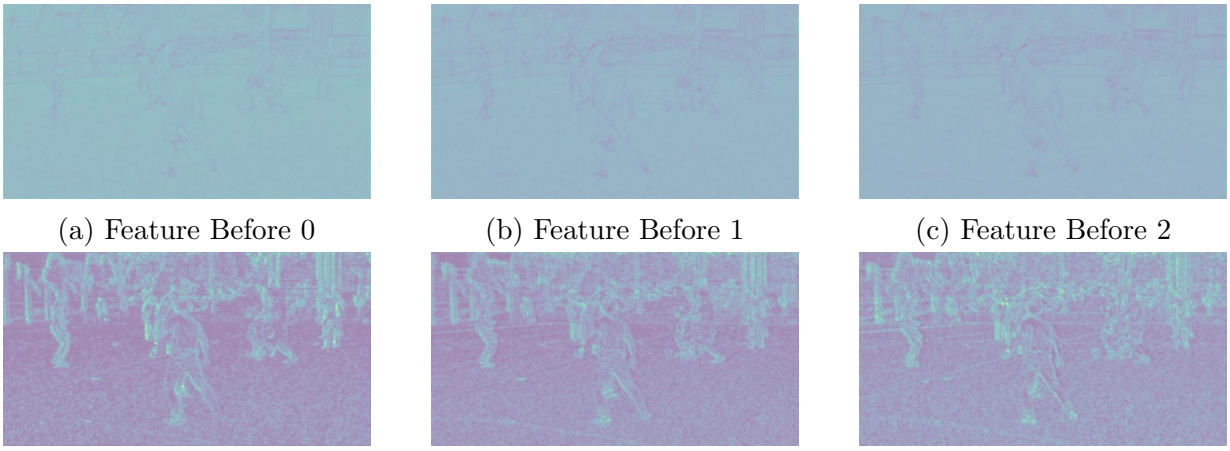
Figure 5.3: Comparison of forward temporal features before TP and backward features after TP at the original resolution.



(a) Feature Before 0 (b) Feature Before 1 (c) Feature Before 2

(d) Feature After 0 (e) Feature After 1 (f) Feature After 2

Figure 5.4: Comparison of backward temporal features before TP and backward features after TP at the 1/4 resolution.



(a) Feature Before 0 (b) Feature Before 1 (c) Feature Before 2

(d) Feature After 0 (e) Feature After 1 (f) Feature After 2

Figure 5.5: Comparison of forward features before TP and backward features after TP at the 1/4 resolution.

Model	Vimeo90K	Slow	Medium	Fast
NoDistillation	35.79/0.971	37.98/0.984	35.63/0.973	32.86/0.948
NoTemporal	23.45/0.511	23.17/0.519	23.48/0.512	23.36/0.481
RIFE Pretrained	36.43/0.968	39.21/0.983	36.24/0.970	35.33/0.971
NoPretrained	36.73/0.972	39.28/0.9842	36.68/0.972	33.99/0.947

Table 5.1: Vimeo90K Ablation Test.

5.3 Ablation test

We compare our model with its ablation versions to demonstrate their necessity. We trained our ablation models to test the effect of the temporal module and distillation module. GRU block and distillation block are removed and trained respectively.

The ablation test is conducted using an input image resolution of 224×224, which

is smaller than the original resolution. This resolution was chosen because it matches the training resolution, and is expected to yield the best performance during validation. Consequently, the metrics demonstrate better performance compared to the results in 5.1.

We show our result comparison on *Vimeo90K Septuplet* where our models are trained. The models in the ablation test are:

- NoDistillation: The distillation part is removed while other parts are the same.
- NoTemporal: The whole TP part is removed, which also removes the GRU features comparison for the distillation structure.
- RIFE Pretrained: The structure is identical to the final version of our model, but the motion estimator uses pretrained parameters from the *RIFE* model and remains unchanged during training.
- NoPretrained: The structure is the final version of our model and trained without any modifications without the usage of the pretrained parameters.

The first two versions, NoDistillation and NoTemporal, reveal that both the TP and distillation blocks significantly enhance the quality of the output.

In the version, RIFE Pretrained, we use the pretrained parameters of the optical flow estimation block from *RIFE* [35]. This version performs best in the Fast subset comparison but is weaker than NoPretrained in the Medium, Slow, and overall performance, indicating that its stability is inferior to the latter.

The model, NoPretrained, achieves better performance than RIFE Pretrained and is chosen as the final version of our model.

5.4 Metrics Comparison to state-of-the-art models

We compare the performance of our model with other traditional VFI models. The models we select are representative:

- *IFRNet* [60] extracted pyramid features from input frames and used intermediate flow to backward warping, which is similar to our model. The model *IFRNet-large* is the extended version of *IFRNet*, where feature channels from the first to the fourth pyramid levels are doubled.
- *SoftSplat* employed a forward warping technique, where our approach utilizes backward warping to achieve alignment between the two images.
- *CAIN* implemented channel attention similar to our model; however, in their methodology, the attention of the two images is concatenated.
- *RIFE* incorporated a distillation process, and we adopted their design for the motion estimator.
- *VFIFormer* use the popular vision transformer structure to leverage the correlation among patches in the images.

- *UPR-Net* demonstrated superior overall performance in comparison to other open source VFI models, rendering it a compelling target for our study. The models *UPR-Net-large* and *UPR-Net-LARGE* are extended versions of *UPR-Net*, where all feature channels of the base version are scaled by factors of 1.5 and 2.0, respectively.

The parameters in other models will be multiplied by three because the parameters in our model are responsible for the entire three intermediate frame generations.

Model	Slow	Medium	Fast	Parameters (M)
IFRNet	35.88/0.986	32.95/0.972	29.68/0.9455	15.0
IFRNet-large	36.22/0.987	33.13/0.926	29.96/0.949	17.4
SoftSplat	37.68/0.984	35.05/0.970	32.07/0.940	23.1
CAIN	36.90/0.983	34.43/0.969	31.12/0.936	42.78
RIFE	37.74/0.985	35.17/0.972	32.20/0.943	29.4
VFIFormer	36.21/0.987	33.33/0.974	30.41/0.951	72.3
UPR-Net	38.22/0.987	35.63/0.973	32.55/0.946	5.1
UPR-Net-large	38.39/0.987	35.84/0.973	32.74/0.947	11.1
UPR-Net-LARGE	36.60/0.987	36.02/0.975	32.91/0.948	19.7
Ours	38.00/0.983	35.67/0.973	33.03/0.948	42.93

Table 5.2: Vimeo90K comparison metrics at different speeds. red represents the highest PSNR, brown represents the second highest, and blue represents the third highest.

We compare our model against state-of-the-art models using the Vimeo90K Septuplet validation test [46]. Following the approach outlined in [58], we partition the original dataset into slow, medium, and fast segments to evaluate the performance of the models across different scenarios. The result can be seen in Table 5.2 and Table 5.3. In Table 5.2, our model achieves the highest PSNR in the subset Fast. In Table 5.3, *UPR-Net-large*

performed the highest PSNR on *X4K1000FPS*. Model *VFIFormer* ranked first on *SNU-FILM* and *PARKOUR*. However, our model performs at a competitive level, ranking within the top 3 across all datasets except *X4K1000FPS*.

Model	Vimeo90K	X4K1000FPS	SNU-FILM	PARKOUR
IFRNet	32.81/0.971	43.00/0.989	37.02/0.986	21.64/0.814
IFRNet-large	33.00/0.973	43.22/0.989	37.23/0.987	21.97/0.819
SoftSplat	35.13/0.967	42.27/0.985	36.59/0.984	22.14/0.806
CAIN	34.43/0.966	40.98/0.983	36.71/0.985	21.81/0.784
RIFE	35.25/0.970	42.55/0.987	36.42/0.984	22.63/0.703
VFIFormer	33.36/0.973	43.13/0.989	37.31/0.987	22.81/0.823
UPR-Net	35.69/0.972	43.23/0.989	36.94/0.986	22.11/0.805
UPR-Net-large	35.89/0.973	43.35/0.989	36.93/0.986	22.18/0.808
UPR-Net-LARGE	36.07/0.973	43.32/0.989	36.94/0.986	22.31/0.813
Ours	35.78/0.972	42.25/0.988	37.11/0.987	22.25/0.810

Table 5.3: Comparison metrics in different datasets (PSNR/ SSIM). red represents the highest PSNR, brown represents the second highest, and blue represents the third highest.

In Fig. 5.6, we compare the performance of different models in terms of the number of parameters against PSNR from Table 5.3. Because the parameters in other models are only used to predict one intermediate frame from the adjacent frames. As a result, in Tabel 5.2 and Fig. 5.6, we tripled the number of the parameters of the other two-frame interpolation models to balance the difference. We can see that our model is capable of producing high-quality interpolated frames while maintaining an acceptable number of parameters.

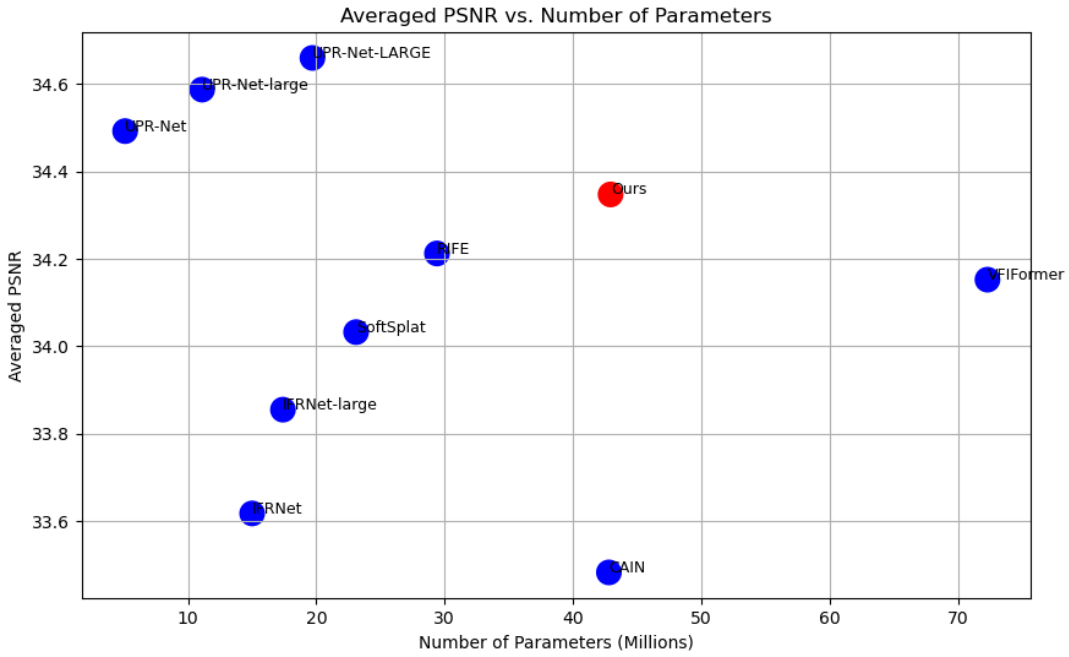


Figure 5.6: Averaged PSNR vs. Parameters in Triple Image Set Interpolation in comparison metrics.

5.5 Visual output comparison

Visual comparisons are shown in Fig. 5.7, Fig. 5.8, Fig. 5.9, and Fig. 5.10. We can see that our model has achieved the best performance in PSNR in this comparison. However, we observe that PSNR does not fully capture the visual quality of images. In our comparisons, both *UPR-Net* and *VFIFormer* produce well-converged boundaries and effectively handle artifacts.

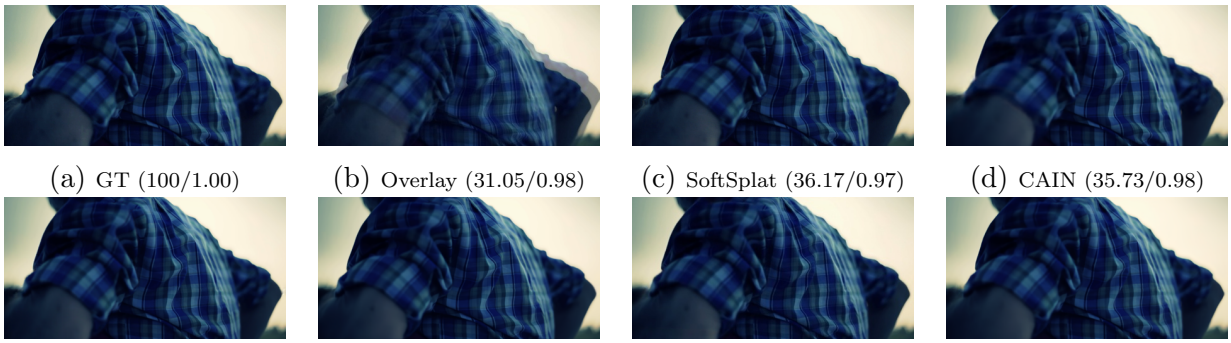
(a) GT (100.00/1.00) (b) Overlay (31.7/0.84) (c) SoftSplat (35.32/0.95) (d) CAIN (35.21/0.96)

(e) RIFE (35.66/0.96) (f) VFIFormer (36.65/0.98) (g) UPR-Net (35.86/0.98) (h) Ours (36.67/0.98)

Figure 5.7: Comparison of different models for Extreme SNU-Film dataset (PSNR/SSIM).

Especially in Fig. 5.7, the girl's hair has a boundary that is very close to the wall with a similar color in the background, making it difficult to fully separate the hair from the wall in the interpolated frame. In our result (h), the artifact blurs the gap between the two, significantly degrading the boundary of the wall. However, both *CAIN* (d) and *UPR-Net* (g) clearly preserve the gap. Nevertheless, the PSNR values for both of these results are relatively lower than ours, which we believe is because our model demonstrates superior

performance in capturing finer details.



(a) GT (100/1.00) (b) Overlay (31.05/0.98) (c) SoftSplat (36.17/0.97) (d) CAIN (35.73/0.98)

(e) RIFE (36.46/0.98) (f) VFIFormer (36.65/0.98) (g) UPR-Net (35.86/0.98) (h) Ours (36.67/0.98)

Figure 5.8: Comparison of different models for Vimeo90K Fast dataset (PSNR/SSIM).

(a) GT (100/1.00) (b) Overlay (32.00/0.77) (c) SoftSplat (34.20/0.94) (d) CAIN (33.33/0.93)

(e) RIFE (34.40/0.95) (f) VFIFormer (34.89/0.96) (g) UPR-Net (33.18/0.93) (h) Ours (35.25/0.97)

Figure 5.9: Comparison of different models for Vimeo90K Fast dataset (PSNR/SSIM).

(a) GT (100/1.00) (b) Overlay (30.10/0.70) (c) SoftSplat (34.83/0.93) (d) CAIN (34.04/0.91)

(e) RIFE (34.95/0.93) (f) VFIFormer (35.58/0.94) (g) UPR-Net (34.95/0.93) (h) Ours (35.66/0.94)

Figure 5.10: Comparison of different models for Vimeo90K Medium dataset (PSNR/SSIM).

6

Conclusions and Future Work

6.1 Conclusion

In conclusion, we have successfully explored the innovative application of a temporal-based attention pyramid feature extractor for VFI, particularly in high frame rate scenarios and slow-motion videos. Through research and experimentation, we have demonstrated that

our model effectively capitalizes on the temporal changes inherent in video sequences, thereby enhancing the quality and accuracy of interpolated frames. The quantitative comparison in Chap 5 can prove that our model can have a stable performance in various conditions, making it a robust solution for real-world applications in VFI.

We have designed a GRU-based attention pyramid feature extractor that capitalizes on temporal changes within video sequences for effective frame interpolation. Our approach is specifically tailored to high frame rate scenarios, with a particular emphasis on interpolating frames in slow-motion videos.

Our TP offers a novel approach to handling temporal information in image processing models. By linking frames in a video sequence into windowed clips, we enhance the integration of spatial and temporal information, thereby improving the model's ability to capture dynamic changes across frames.

We conducted a comparative analysis of our model against state-of-the-art models and ablation tests. The results affirm that our model meets the current standards, indicating promising applications in advanced video frame interpolation tasks.

6.2 Future work

While our TP has achieved significant progress in the VFI task, there remains considerable room for further improvement in our work.

- In [61], deformable convolution is employed to capture the motion within video se-

quences, which could be beneficial for TP in extracting more detailed information from window clips. Determining the optimal placement and usage of deformable convolution could pose a significant challenge but also offer a substantial improvement in constructing a better contextual reference for intermediate frame reconstruction.

- Currently, our model is trained solely on the Vimeo90K dataset. While this dataset includes a wide variety of general scenarios, it lacks emphasis on specific motions, such as human body movements and sports games. This limitation may restrict the practical applicability of our model and fail to fully realize its potential. Training on multiple datasets could be a beneficial strategy to enhance the model's versatility and effectiveness in diverse real-world applications.

- The temporal structure could be adapted to other transformations to capitalize on the context provided by video sequences. Currently, we isolate each window clip from others, confining computations within the clip itself. Considering interactions between clips can be promising for further development of our methodology.

References

1. Jordan, M. I. in *Advances in psychology* 471–495 (Elsevier, 1997).
2. Elman, J. L. Finding structure in time. *Cognitive science* **14,** 179–211 (1990).
3. Zhang, Y. *A better autoencoder for image: Convolutional autoencoder* in *ICONIP17-DCEC. Available online: http://users. cecs. anu. edu. au/Tom. Gedeon/conf/ABCs2018/paper/ABCs2018_paper_58. pdf (accessed on 23 March 2017)* (2018).
4. Ladjal, S., Newson, A. & Pham, C.-H. A PCA-like autoencoder. *arXiv preprint arXiv:1904.01277* (2019).
5. Pham, C.-H., Ladjal, S. & Newson, A. PCA-AE: Principal Component Analysis Autoencoder for Organising the Latent Space of Generative Networks. *Journal of Mathematical Imaging and Vision,* 1–17 (2022).
6. Li, X. *et al.* H-DenseUNet: hybrid densely connected UNet for liver and tumor segmentation from CT volumes. *IEEE transactions on medical imaging* **37,** 2663–2674 (2018).
7. Cao, H. *et al. Swin-unet: Unet-like pure transformer for medical image segmentation* in *European conference on computer vision* (2022), 205–218.
8. Horé, A. & Ziou, D. *Image Quality Metrics: PSNR vs. SSIM* in *2010 20th International Conference on Pattern Recognition* (2010), 2366–2369.
9. Brunet, D., Vrscay, E. R. & Wang, Z. On the mathematical properties of the structural similarity index. *IEEE Transactions on Image Processing* **21,** 1488–1499 (2011).

10. Roth, S. & Black, M. J. On the spatial statistics of optical flow. *International Journal of Computer Vision* **74,** 33–50 (2007).

11. Piao, D., Menon, P. G. & Mengshoel, O. J. *Computing probabilistic optical flow using markov random fields* in *Computational Modeling of Objects Presented in Images. Fundamentals, Methods, and Applications: 4th International Conference, CompIMAGE 2014, Pittsburgh, PA, USA, September 3-5, 2014 4* (2014), 241–247.

12. Dosovitskiy, A. *et al. Flownet: Learning optical flow with convolutional networks* in *Proceedings of the IEEE international conference on computer vision* (2015), 2758–2766.

13. Ilg, E. *et al. Flownet 2.0: Evolution of optical flow estimation with deep networks* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 2462–2470.

14. Ranjan, A. & Black, M. J. *Optical flow estimation using a spatial pyramid network* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 4161–4170.

15. Sun, D., Yang, X., Liu, M.-Y. & Kautz, J. *Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 8934–8943.

16. Zhang, F., Woodford, O. J., Prisacariu, V. A. & Torr, P. H. *Separable flow: Learning motion cost volumes for optical flow estimation* in *Proceedings of the IEEE/CVF international conference on computer vision* (2021), 10807–10817.

17. Shi, X. *et al. Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 1599–1610.

18. Xu, H., Zhang, J., Cai, J., Rezatofighi, H. & Tao, D. *Gmflow: Learning optical flow via global matching* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), 8121–8130.

19. Xu, H. *et al.* Unifying flow, stereo and depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

20. Teed, Z. & Deng, J. *Raft: Recurrent all-pairs field transforms for optical flow* in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16* (2020), 402–419.

21. Hofinger, M. *et al. Improving optical flow on a pyramid level* in *European Conference on Computer Vision* (2020), 770–786.

22. Guan, S., Li, H. & Zheng, W.-S. *Unsupervised learning for optical flow estimation using pyramid convolution lstm* in *2019 IEEE international conference on multimedia and expo (ICME)* (2019), 181–186.

23. Brox, T., Bregler, C. & Malik, J. *Large displacement optical flow* in *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), 41–48.

24. Kalluri, T., Pathak, D., Chandraker, M. & Tran, D. *Flavr: Flow-agnostic video representations for fast frame interpolation* in *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (2023), 2071–2082.

25. Meyer, S., Wang, O., Zimmer, H., Grosse, M. & Sorkine-Hornung, A. *Phase-based frame interpolation for video* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), 1410–1418.

26. Shen, W. *et al. Blurry video frame interpolation* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), 5114–5123.

27. Sim, H., Oh, J. & Kim, M. *Xvfi: extreme video frame interpolation* in *Proceedings of the IEEE/CVF international conference on computer vision* (2021), 14489–14498.

28. Jiang, H. *et al. Super slomo: High quality estimation of multiple intermediate frames for video interpolation* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 9000–9008.

29. Bao, W. *et al. Depth-aware video frame interpolation* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), 3703–3712.

30. Yu, Z. *et al. Deep bayesian video frame interpolation* in *European Conference on Computer Vision* (2022), 144–160.

31. Niklaus, S. & Liu, F. *Softmax splatting for video frame interpolation* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), 5437–5446.

32. Jin, X. *et al.* UPR-Net: A Unified Pyramid Recurrent Network for Video Frame Interpolation. *International Journal of Computer Vision,* 1–15 (2024).

33. Hinton, G., Vinyals, O. & Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

34. Çelik, A., Küçükmanísa, A. & Urhan, O. Feature distillation from vision-language model for semisupervised action classification. *Turkish Journal of Electrical Engineering and Computer Sciences* (2023).

35. Huang, Z., Zhang, T., Heng, W., Shi, B. & Zhou, S. *Real-time intermediate flow estimation for video frame interpolation* in *European Conference on Computer Vision* (2022), 624–642.

36. Liu, Z., Wang, Y. & Chu, X. A Simple and Generic Framework for Feature Distillation via Channel-wise Transformation. *arXiv.org* (2023).

37. Xiang, X. *et al. Zooming slow-mo: Fast and accurate one-stage space-time video super-resolution* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), 3370–3379.

38. Chan, K. C., Wang, X., Yu, K., Dong, C. & Loy, C. C. *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* (2021), 4947–4956.

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

40. Zhang, H., Ren, D., Yan, Z. & Zuo, W. Arbitrary Timestep Video Frame Interpolation with Time-Dependent Decoding. *Mathematics* **12,** 303 (2024).

41. Khan, R. *et al.* A deep neural framework for image caption generation using gru-based attention mechanism. *arXiv preprint arXiv:2203.01594* (2022).

42. Tatsunami, Y. & Taki, M. Sequencer: Deep lstm for image classification. *Advances in Neural Information Processing Systems* **35,** 38204–38217 (2022).

43. Wang, W. Text Sentiment Classification Method Based on Bilstm. *Highlights in Business, Economics and Management* **21,** 679–687 (2023).

44. Liu, Z., Yeh, R. A., Tang, X., Liu, Y. & Agarwala, A. *Video frame synthesis using deep voxel flow* in *Proceedings of the IEEE international conference on computer vision* (2017), 4463–4471.

45. Niklaus, S. & Liu, F. *Context-aware synthesis for video frame interpolation* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 1701–1710.

46. Xue, T., Chen, B., Wu, J., Wei, D. & Freeman, W. T. Video enhancement with task-oriented flow. *International Journal of Computer Vision* **127,** 1106–1125 (2019).

47. Yang, Y., Wang, Y., Lian, J. & Fang, Z. *Pyramid Channel Attention Combined Adaptive Multi-column Network for SISR* in *International Forum on Digital TV and Wireless Multimedia Communications* (2022), 180–190.

48. Zhong, Z. *et al. Squeeze-and-attention networks for semantic segmentation* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), 13065–13074.

49. Xu, L., Jia, J. & Matsushita, Y. Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **34,** 1744–1757 (2011).

50. Apicella, A., Donnarumma, F., Isgrò, F. & Prevete, R. A survey on modern trainable activation functions. *Neural Networks* **138,** 14–32 (2021).

51. Sane, P. & Agrawal, R. *Pixel Normalization from Numeric Data as Input to Neural Networks: For Machine Learning and Image Processing* in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)* (2017), 2221–2225.

52. Choi, M., Kim, H., Han, B., Xu, N. & Lee, K. M. *Channel attention is all you need for video frame interpolation* in *Proceedings of the AAAI Conference on Artificial Intelligence* **34** (2020), 10663–10671.

53. Han, J. & Moraga, C. *The influence of the sigmoid function parameters on the speed of backpropagation learning* in *International workshop on artificial neural networks* (1995), 195–201.

54. Choi, M., Kim, H., Han, B., Xu, N. & Lee, K. M. *Channel Attention Is All You Need for Video Frame Interpolation* in *AAAI* (2020).

55. Li, Z. *et al. Estimating 3D Motion and Forces of Person-Object Interactions from Monocular Video* in *Computer Vision and Pattern Recognition (CVPR)* (2019).

56. Maldonado, G., Bailly, F., Souères, P. & Watier, B. Angular momentum regulation strategies for highly dynamic landing in Parkour. *Computer Methods in Biomechanics and Biomedical Engineering* **20,** 123–124 (2017).

57. Maldonado, G. *Analysis and generation of highly dynamic motions of anthropomorphic systems : application to parkour* PhD thesis (Université Paul Sabatier - Toulouse III, 2017).

58. Xu, G. *et al. Temporal Modulation Network for Controllable Space-Time Video Super-Resolution* in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2021).

59. Maldonado, G., Bailly, F., Souères, P. & Watier, B. Angular momentum regulation strategies for highly dynamic landing in Parkour. *Computer Methods in Biomechanics and Biomedical Engineering* **20,** 123–124 (2017).

60. Kong, L. *et al. Ifrnet: Intermediate feature refine network for efficient frame interpolation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 1969–1978.

61. Shi, Z., Liu, X., Shi, K., Dai, L. & Chen, J. Video Frame Interpolation via Generalized Deformable Convolution. *IEEE Transactions on Multimedia* **24,** 426–439 (2022).