BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

NEURAL NETWORKS FOR VIDEO QUALITY ENHANCEMENT

ZLEPŠOVÁNÍ KVALITY VIDEA POMOCÍ NEURONOVÝCH SÍTÍ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **MATEJ SIROVATKA**
AUTOR PRÁCE

SUPERVISOR **Ing. MICHAL HRADIŠ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2024

Bachelor's Thesis Assignment

155408

Institut:	Department of Computer Graphics and Multimedia (DCGM)
Student:	**Sirovatka Matej**
Programme:	Information Technology
Title:	**Neural Networks for Video Quality Enhancement**
Category:	Image Processing
Academic year:	2023/24

Assignment:

1. Familiarize yourself with basics of video processing using artificial neural networks.
2. Review contemporary video enhancement methods which use neural networks.
3. Propose a method or propose an extension of an existing multi-frame video enhancement method.
4. Prepare a suitable data set for experiments.
5. Implement the proposed method and conduct experiments on the dataset.
6. Evaluate the achieved results and discuss future development possibilities.
7. Create a concise video presentation outlining your work, its objectives, and results.

Literature:
- Liang et al.: VRT: A Video Restoration Transformer, arXiv preprint arXiv:2201.12288, 2022.
- Isobe et al.: Video Super-resolution with Temporal Group Attention. arXiv:2007.10595. 2020.
- Cao et al.: Video Super-Resolution Transformer, arXiv:2106.06847v3, 2023.
- Yonggui Zhu and Guofang Li: A Lightweight Recurrent Grouping Attention Network for Video Super-Resolution, arXiv:2309.13940, 2023.

Requirements for the semestral defence:
Goals 1-4.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Hradiš Michal, Ing., Ph.D.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	9.5.2024
Approval date:	17.4.2024

Abstract

In this thesis, a new method for video super-resolution is proposed. The method is based on the idea of using deformable convolutional layers together with optical flow to align features from multiple sequential video frames. This novel module is then used in a U-Net-like deep neural network to predict high-resolution frames. The proposed method is evaluated on a dataset containing real-life scenes and compared to other methods. Multiple different configurations of the proposed method are tested and the results are analyzed. The results of the experiments show promising results, with the model outperforming bilinear interpolation, and single-frame methods. Multiple different architectures of the feature alignment module together with the rest of the U-Net architecture are tested, showing that using Vgg19 as the encoder of the U-Net gives the best results.

Abstrakt

Cieľom tejto práce je vytvoriť novú metódu super rozlíšenia na zlepšenie kvality videa. Táto metóda je založená na myšlienke použitia deformovateľných konvolučných vrstiev a optického toku na zarovnanie príznakov z viacerých po sebe následujúcich snímkov videa. Táto metóda je následne použitá v neuronovej sieti založenej na U-Net architektúre na predikciu snímkov vo vysokom rozlíšení. Vyhodnotenie je prevedené na datasete obsahujúcom snímky z reálneho života a porovnané s inými metódami. Testované sú rôzne konfigurácie navrhnutej metódy a výsledky sú analyzované. Výsledky experimentov ukazujú sľubné výsledky, pričom model prekonáva bilineárnu interpoláciu a metódy založené na jednom snímku. Testované sú rôzne architektúry modulu zarovnávania príznakov spolu s celou architektúrou U-Net, pričom sa ukazuje, že použitie Vgg19 ako enkóderu dáva najlepšie výsledky.

Keywords

video quality enhancement, video super-resolution, optical flow, deformable convolutions, neural networks, deep learning, supervised learning, U-Net

Klíčová slova

zlepšenie kvality videa, super rozlíšenie videa, optický tok, deformovateľné konvolúcie, neurónové siete, hlboké učenie, učenie s učiteľom, U-Net

Reference

SIROVATKA, Matej. *Neural Networks for Video Quality Enhancement.* Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Michal Hradiš, Ph.D.

Rozšířený abstrakt

Hlavným cieľom tejto práce je vytvoriť novú architektúru neurónovej siete pre super rozlíšenie videa. Cieľom je preskúmať existujúce architektúry a navrhnúť nový modul na zarovnanie príznakov z viacerých sekvenčných snímkov videa. Tento modul je založený na použití deformovateľných konvolučných vrstiev a optického toku.

Zväčšenie kvality a rozlíšenia videa je dôležitým problémom v oblasti spracovania videa. Použitie neurónových sietí na tento účel je veľmi populárne a existuje mnoho architektúr, ktoré dosahujú dobré výsledky. Problémom a zároveň výzvou je použitie informácií z viacerých snímkov videa na produkciu snímku vo vyššom rozlíšení a kvalite. Častým problémom je zarovnanie príznakov z okolitých snímkov videa. Súčasné metódy často používajú metódy bez explicitného zarovnania príznakov, čo môže viesť k chybám v predikcii. V tejto práci je navrhnutý nový modul, ktorý explicitne zarovnáva príznaky z viacerých snímkov videa pomocou optického toku predikovaného predtrénovanou neurónovou sieťou RAFT. Optický tok medzi predikovaným a podpornými snímkami videa je následne použitý ako posun pre deformovateľné konvolučné vrstvy, ktorých zorné pole sa týmto spôsobom posúva na miesto so zodpovedajúcimi príznakmi v podpornom snímku videa. Tento modul je následne použitý v neurónovej sieti založenej na U-Net architektúre na predikciu snímkov vo vyššom rozlíšení. Výsledkom práce su experimenty porovnávajúce rôzne konfigurácie navrhnutej metódy a porovnávanie s inými metódami a takisto skripty na trénovanie a vyhodnocovanie rozšíreného modelu.

Architektúra a jej variácie boli boli trénované a testované na dátovej sade REDS pozostávajúcej z 300 sekvencií po 100 snímkoch, kde snímky nízkej kvality mali rozlíšenie 180×320 pixelov, a úlohou neurónovej siete bolo predikovať odpovedajúce snímky v rozlíšení 720×1280 pixelov. Vstupom siete boli 3 po sebe následujúce snímky videa a optický tok medzi stredným a predchádzajúcim, respektíve následujúcim snímkom a výstupom siete bol prostredný snímok vo vyššom rozlíšení. Pomocou enkóderu U-Net architektúry boli extrahované príznaky v rôznych rozlíšeniach pre každý snímok. Príznaky podporných (predchádzajúci a následujúci) snímkov boli následne spracované deformovateľnými konvolučnými vrstvami, do ktorých bol vloženy optický tok ako posun. Toto zarovanie prebehlo na každej vrstve architektúry, z dôvodu zachovania informácií o zarovnaní na rôznych úrovniach príznakov. Výsledné príznaky boli následne spojené s príznakmi stredného snímku a ich dimenzionalita bola znížená konvolučnými vrstvami. Týmto spôsobom bolo dosiahnute aby množstvo príznakov sedelo s množstvom príznakov jedného snímku. Výsledné príznaky boli následne spracované dekóderom siete, ktorý predikoval finálne príznaky snímku. Tieto príznaky boli následne spracované sekvenciou reziduálnych konvolučných vrstiev a nakoniec spracované konvolučnými vrstvami a PixelShuffle vrstvami, ktoré zvýšili rozlíšenie príznakov na požadované rozlíšenie. Na tréning siete bola použitá stredná kvadratická chyba a na vyhodnocovanie kvality boli použité metriky PSNR a SSIM. Výsledky experimentov ukazujú, že navrhnutá metóda dosahuje lepšie výsledky ako bilineárna interpolácia a metódy založené na jednom snímku. Výsledky sú analyzované a diskutované a sú navrhnuté možné zlepšenia.

Experimenty boli vykonané na rôznych konfiguráciách navrhnutej metódy, kde boli testované rôzne architektúry modulu zarovnávania príznakov spolu s celou architektúrou U-Net. Výsledky ukazujú, že použitie Vgg19 ako enkóderú poskytuje najlepšie výsledky, kvôli vyššej dimenzionalite príznakov na výstupe enkóderu oproti architektúre používajúcej ResNet34 ako enkóder (32 kanálov vs 16 kanálov). Vďaka vyššej dimenzionalite príznakov je možné zachovať viac informácií o zarovnaní príznakov z podporných snímkov, čo vedie k lepším výsledkom. Následne boli vykonané experimenty, kde optický tok nebol priamo

vložený ako posun do deformovateľných konvolučných vrstiev, ale bol predtým spracovaný 1 konvolučnou vrstvou, ktorej úlohou boli zmeniť dimenzie optického toku aby sedeli s dimenziami pre posuny deformovateľných konvolučných vrstiev. Výsledky tohto experimentu preukázali lepšie výsledky ako priamo vložený optický tok.

Vo výsledkoch experimentov je vidieť, že navrhnutá metóda dosahuje lepšie výsledky ako bilineárna interpolácia a metódy založené na 1 snímku. Najlepšie výsledky dosahuje architektúra s Vgg19 ako enkóderom a s predspracovaným optickým tokom. Ďalšie experimenty, na rôznych architektúrach, ako napríklad GAN siete alebo metódy používajúce koncept pozornosti, by mohli priniesť ešte lepšie výsledky.

Neural Networks for Video Quality Enhancement

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Michal Hradiš, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Matej Sirovatka
May 9, 2024

Acknowledgements

I would like to thank my supervisor Ing. Michal Hradiš, Ph.D. for his guidance and support during the preparation of this thesis.

Contents

Chapter 1

Introduction

Video quality enhancement is a process of improving the quality of a video by removing visual imperfections or improving its resolution. In recent years, the rise of deep learning has brought many new methods and architectures, that are able to surpass the classical methods.

This work focuses on the research of the existing methods and architectures and the development of a new method for video quality enhancement, concretely for super-resolution – a task of increasing the resolution combined with quality improvement. A common problem in a video super-resolution with neural networks is how to align matching features from the neighboring frames. A novel module is proposed, that uses a combination of deformable convolutions and optical flow for this task. Experiments with this module are then conducted on a U-Net [44] architecture to evaluate its performance. Different variations of this module, together with different U-Net architectures are tested and compared with state-of-the-art methods.

The experiments have shown promising results, outperforming both classical upsampling using bilinear interpolation and learned upsampling using U-Net architecture without this module. These results show, that the proposed module is able to align the features of the neighboring frames, which leads to better results. Therefore, this module can be used in other architectures and tasks, where the alignment of the features is crucial, yielding even better results.

Starting with Chapter 2, which covers the basics of image quality enhancement, different problems and methods are described, whether using neural networks or not, used for each task in image quality enhancement. In Chapter 3, an overview of the state-of-the-art methods is given with a focus on the recent and most successful architectures such as U-Net, EDVR [52] and RVRT [32] together with a description of commonly used datasets and metrics. In Chapter 4, a problem of video super-resolution and motivation for this thesis is described in detail, followed by the proposed solution design. Chapter 5 covers the implementation details, such technologies used and the structure of the source code. Lastly, in Chapter 6, results from different experiments are shown and discussed.

Chapter 2

Image Quality Enhancement

This chapter focuses on the methods used to improve the quality of images. These methods then can be mapped to video, when the video is treated as a sequence of images. It covers the most common and simple non-neural network methods, as well as the metrics used to evaluate these methods. It also gives an overview of the most common neural network architectures used for image quality enhancement.

2.1 Image Denoising

Image denoising can be thought of as a process of removing noise and visual imperfections from an image. Noise can be of various forms and can be caused by various factors. The most common causes of noise are inadequate lighting, low-quality camera sensors or movement and motion. Some noise can also be caused because of image compression, which is commonly used to reduce the size of images. During the compression, some information is lost, which can cause visual imperfections, such as edge and detail smoothing, blocking artifacts, etc.

2.1.1 Non-neural network methods

Before the widespread use of neural networks, image denoising was done using methods relying on various mathematical models [12, 9]. Some of the most common methods are:

- **Median filter**: This method replaces each pixel's value with the median value of the pixels in the neighborhood. This method is very simple and fast, but it can cause blurring of the edges.
- **Gaussian filter**: A Gaussian convolution kernel is used to blur the image. This is especially useful when trying to remove Gaussian noise. Such as the median filter, this method also can cause blurring of the edges.
- **Bilateral filter**: Method, similarly to the Gaussian filter uses a convolution kernel with weights that are commonly sampled from the Gaussian distribution. Crucially, the weights are not only dependent on the spatial distance but also on the radiometric distance (e.g., color intensity, depth distance, etc.). This helps to preserve sharp edges, while still removing the noise.
- **Non-local means** [7]: Based on the idea that the image contains many similar patches, this method replaces each pixel's value with the weighted average of pixels

located in similar patches across the image. These weights are directly proportional to the similarity of the patches. This is very effective for preserving detailed textures, which are often lost when using other methods.

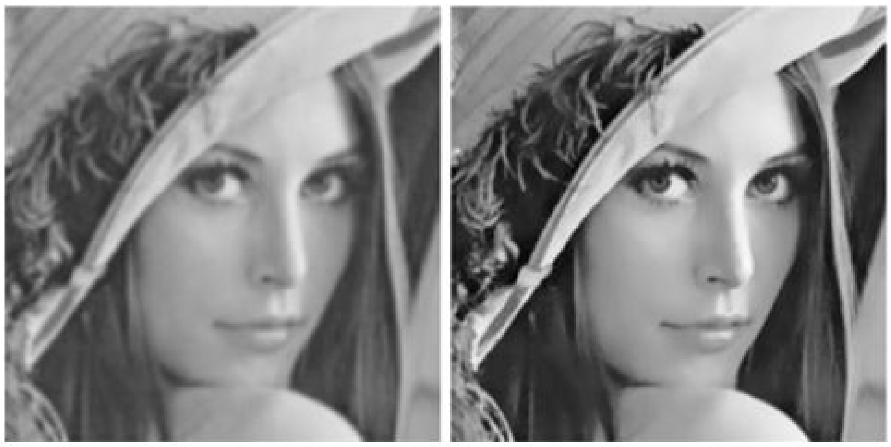


Figure 2.1: Example of image denoising using Gaussian and Bilateral (left to right) denoising kernels. Image taken from [14].

2.1.2 Neural network methods

With the rise of deep learning, many different architectures have been proposed and used for image restoration, slowly surpassing classical methods [48]. These range from simple convolutional neural networks (CNNs) to more complex architectures, such as generative adversarial networks (GANs).

- **U-Net** [44]: Architecture originally designed for medical segmentation, but has been widely used for other image-based tasks as well. It consists of an encoder, that compresses the input image into a lower-dimensional space, and a decoder, that generates the output image from this representation. Residual connections are used to connect the encoder and decoder, which helps to preserve the detailed textures and edges. This architecture is discussed in more detail in Section 3.3.1.

- **Denoising Convolutional Neural Networks (DnCNN)** [58]: This is a type of CNN, that is specifically designed for image denoising. Interesting thing about this architecture is, that rather than predicting the clean image directly, it predicts the residual image, that is the difference between the clean and the noisy image.

- **Autoencoders**: First proposed in the early 1990s by Kramer *et al.* [27], but became widely used with the concept of generative models of data. A neural network, that is used to learn efficient representation of the input data. This works by compressing the input data into a lower-dimensional space and then reconstructing the original data from this representation. By training the network on pairs of clean and noisy images, the network can learn to map noisy images to clean ones, effectively removing the noise and other visual imperfections.

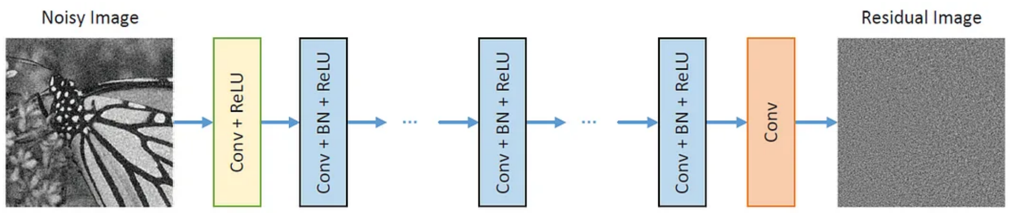


Figure 2.2: Example of DnCNN architecture. Taken from [58].

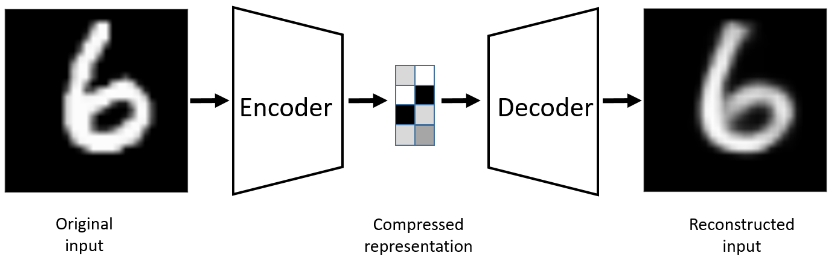


Figure 2.3: A schema of an Autoencoder architecture. Taken from [6].

- **Generative Adversarial Networks (GANs)** [15]: GANs are a type of neural network, that consists of two networks, a generator and a discriminator. The generator is responsible for generating the clean image, meanwhile the discriminator is responsible for distinguishing between the original clean image and its generated counterpart. Through this adversarial process, the generator learns to produce highly realistic denoised images, however, they can often produce nonexistent details.

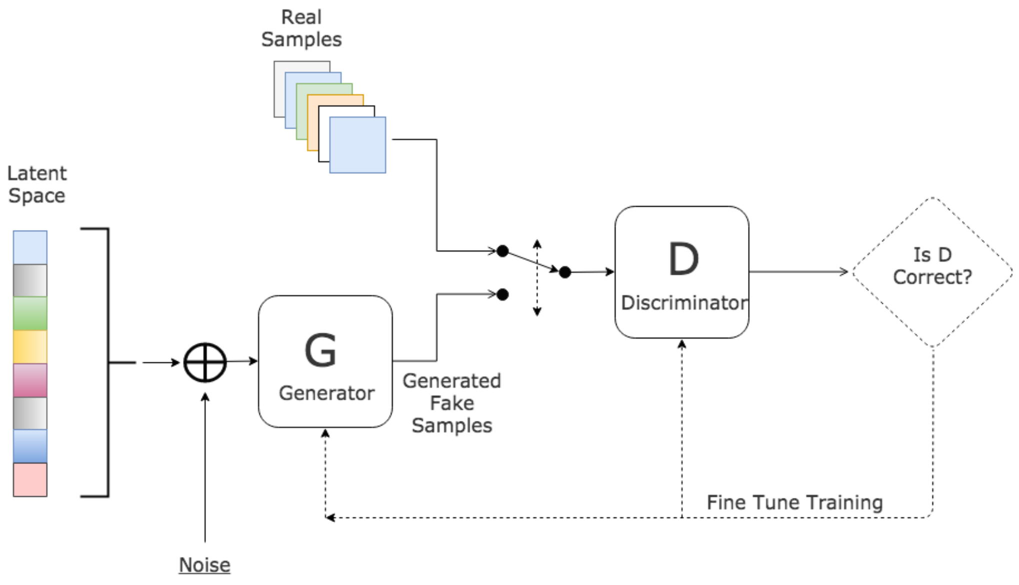


Figure 2.4: An overview of GAN architecture. Taken from [19].

2.2 Super-resolution and Upsampling

The task of increasing the resolution of an image can be split into 2 subcategories: image upsampling and super-resolution [29]. While both tasks aim to increase the resolution of an image, super-resolution also aims to improve the quality of the image, by removing noise and other visual imperfections that are present in the original image. Image upsampling is a simple process, that can be done using various interpolation methods, super-resolution is a more complex process, that often uses deep neural networks to estimate the missing pixels in the image.

2.2.1 Upsampling Methods

Interpolation-based methods are the simplest way to increase the resolution of an image. These methods work by estimating the missing pixels in the image. These methods are simple and computationally efficient, but they are not able to remove visual imperfections in the original image, instead, they are more likely to cause new ones. Paper by Parsania *et al.* [43] gives a good overview of these methods. In the following section, the most common methods are described in more detail.

- **Nearest neighbor Interpolation**: Using values of the nearest pixel to estimate the missing pixels, this method is the simplest, but it can cause heavy visual imperfections such as blocky or jagged edges.

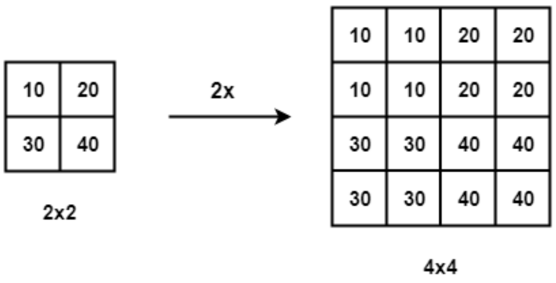


Figure 2.5: Example of nearest neighbor interpolation. Taken from [2].

- **Bilinear Interpolation**: Computing the weighted average of the four nearest pixels, this method calculates the missing pixel. The result is a smoother image, but there can still be visual imperfections, such as blurring.

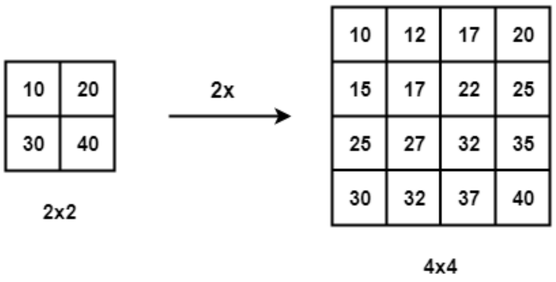


Figure 2.6: Example of bilinear interpolation. Taken from [3].

- **Bicubic Interpolation**: This method uses a cubic polynomial to estimate the missing pixels. It works similarly to bilinear interpolation, but it uses a bigger neighborhood to estimate the missing pixels, sixteen pixels to be exact. This results in a smoother and sharper image.

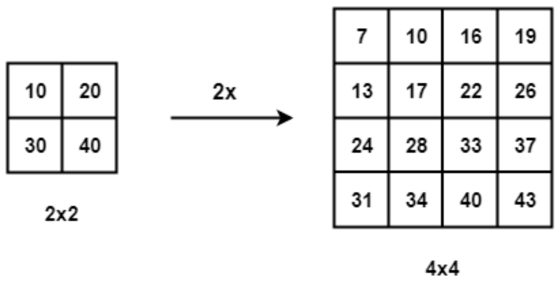


Figure 2.7: Example of bicubic interpolation. Taken from [4].

2.2.2 Super-resolution Methods

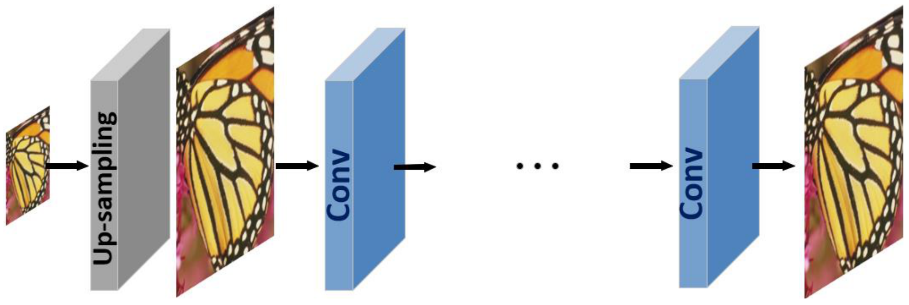
To increase the resolution of an image, together with improving its quality, neural networks are used almost exclusively. Using neural networks to estimate the missing pixels is a more complex and computationally expensive approach, but it can produce significantly better results. These methods range from simple U-Net-like architectures to more complex methods involving GANs or Transformers. Newer approaches are also using the attention mechanism, which helps the network to focus on the important parts of the image. These methods often produce higher-quality results than the traditional interpolation-based methods, one of their biggest advantages is their ability to improve images in more ways, than just increasing the resolution.

We can divide these methods into 3 categories by in what order they process the features: pre-upsampling, post-upsampling and progressive upsampling [53]. The difference between these methods can be seen in Figure 2.8.

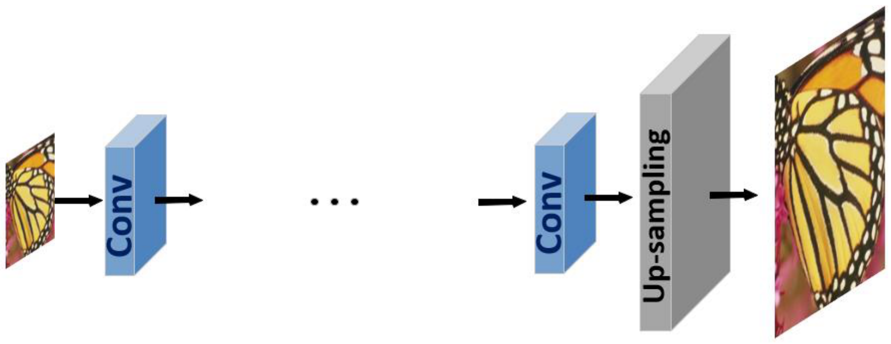
- **Pre-upsampling** This method is based on upsampling the image before it is passed through a network, refining details. The upsampling is done using traditional methods, such as interpolation-based methods, described in Section 2.2.1.

- **Post-upsampling** This method uses a neural network to obtain features from the low-resolution image and then uses these features to generate the high-resolution image. This method is more computationally efficient, because of the lower spatial dimensions of the image being processed by the neural networks. Most commonly, traditional methods are used to upsample the features, but using neural networks is also possible, such as Pixel Shuffle layer [46] or Transposed Convolutions [56].

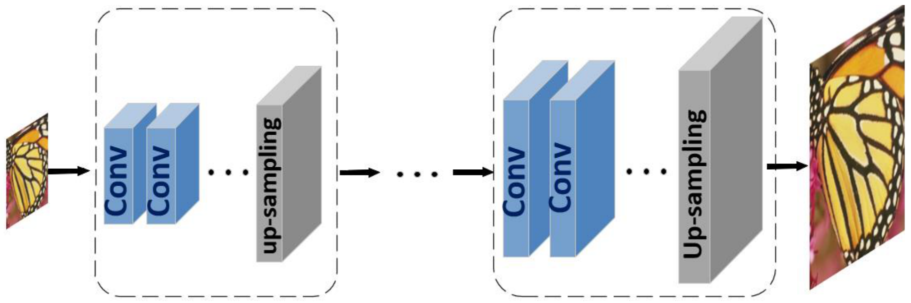
- **Progressive upsampling** For problems, where big scaling factors are required, progressive upsampling methods are used. These methods use multiple neural network sections, each scaling the image by a smaller factor. This approach is computationally expensive, but it can produce significantly better results, especially when the scaling factor is high.



(a) Pre-upsampling. Taken from [16].



(b) Post-upsampling. Taken from [16].



(c) Progressive upsampling. Taken from [16].

Figure 2.8: Different methods of deep learning architectures for image upsampling.

Image super-resolution is traditionally strongly supervised, meaning that the network is trained on pairs of low and high-resolution images. However, a common problem is not having the high-resolution images available. Therefore, multiple different weakly-supervised methods have been proposed. While these methods show promising results, they are usually not as performant as the fully-supervised methods.

- **Self-supervised learning** The biggest advantage of this method is that it does not require any high-resolution data at all. The network is trained only on low-resolution samples. For example, paper *Self-Supervised Super-Resolution for Multi-Exposure Push-Frame Satellites* by Nguyen*et.al* [39] uses 1 of the frames from the satellite as the ground truth and the other as the input.
- **Semi-supervised learning** This method uses a small number of samples consisting of high and low-resolution pairs and a large number of samples consisting only of low-resolution images. The network is trained on the labeled data. Then different methods are used to utilize the unlabeled data, such as consistency regularization as in paper *Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results* [49] or pseudo-labeling, which was popularized by paper *Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks* [28].
- **Unsupervised learning** This method does not require any labeled data. The network is trained on the low-resolution images only. It is more commonly used in image restoration tasks, where the reference image is not available. A popular architecture for this task is CycleGAN proposed in paper [59]. The main idea behind this method is to use generator and discriminator networks. More about this method can be found in Section 2.1.2.

2.3 Metrics

Determining the quality of an image is a crucial, yet difficult task, due to the subjective nature of human perception. While technical parameters, such as dynamic range, noise ratio, etc., can be used to measure the quality of an image, they do not always correlate with the perceived quality of the image. However, there exists a plethora of metrics that can be used to measure image quality.

The connection to human perception is fundamental because the ultimate goal of many image-processing tasks is to produce images that are pleasing or useful to human viewers. Thus, metrics often strive to mimic the way humans evaluate visual quality. This is challenging because human perception is highly complex and subjective, influenced by a multitude of factors including context, prior experience, and even individual differences in vision. We can split these metrics into 3 categories: full-reference metrics, no-reference metrics and reduced-reference metrics [5]. As stated in the previous section, the most common way to evaluate the quality of an image is to compare it to a reference image, therefore full-reference metrics are usually used to optimize the neural networks. However, to assess the perceived quality of the images, no-reference and reduced-reference metrics can be used.

Mean Squared Error (MSE) is a very simple, yet widely used full-reference metric. It is calculated as:

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} [I(i,j) - K(i,j)]^2 \,, \tag{2.1}$$

where I is the original image, K is the predicted image and M and N are the dimensions of the image, i and j are the indices in the image. Therefore this can be interpreted as the average squared difference between each pixel of the original and predicted image. This method is heavily influenced by the outliers, which can cause the metric to be very high, even if the image is visually very similar to the original.

Mean Absolute Error (MAE) can be a better choice in case outliers are not considered a problem. It is pretty similar to the mean squared error, but instead of squaring the difference, the absolute value is used. This effectively makes the outliers have a lesser impact on the result. Is calculated as follows:

$$MAE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} |I(i,j) - K(i,j)| \,. \tag{2.2}$$

Peak Signal-to-Noise Ratio (PSNR) [21] is a widely used metric for measuring the quality of an image. It quantifies the accuracy of the reconstructed image by assessing the level of distortion or noise introduced during image processing. It is expressed in decibels providing a logarithmic scale for the comparison of the original and reconstructed images. Higher values of PSNR indicate better quality of the reconstructed image, where the threshold for good values varies from task to task. The formula for PSNR is as follows:

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right), \tag{2.3}$$

where MAX_I is the maximum possible pixel value of the image calculated as $2^b - 1$, where b is the number of bits per pixel. As mentioned before, MSE represents the mean squared error metric. However, while PSNR is widely used, it is not a perfect metric.

Structural Similarity Index (SSIM) [21] is an advanced metric used for measuring the similarity between 2 images. It considers image degradation as a change in structural information. The idea behind structural similarity is that the pixels in the image are not independent, but they are related to their neighboring pixels. SSIM is calculated using three terms: luminance, contrast and structure. Calculating SSIM is done by applying the following formula to multiple windows of the image:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{2.4}$$

where x and y are the predicted and original windows, μ_x and μ_y are the average intensity values of the windows, σ_x^2 and σ_y^2 are the variances of the windows, σ_{xy} is the covariance of the windows, C_1 and C_2 are constants, that are used to stabilize the division in case of a small denominator, calculated as:

$$C_1 = (K_1L)^2, C_2 = (K_2L)^2, \tag{2.5}$$

where K_1 and K_2 are usually set to 0.01 and 0.03 respectively, and L is the dynamic range of the pixel values, calculated as MAX_I in peak signal-to-noise ratio.

Perceptual Loss [23] is a different method of obtaining the error metric, using a pre-trained neural network to do so. This method is called perceptual loss. The idea behind this method is to use a neural network trained on a large dataset to extract features from 2 images being compared on different levels. These features are then compared using a predefined function, which can be simple, such as Euclidean distance. This method is more complex and aims to minimize the difference between the feature maps extracted from the original and the predicted image. This helps to also include the human perception of the image, as the neural networks learned to mimic the behavior of the human cortex.

No Reference Image Quality Evaluation Metrics While all the previously mentioned metrics require a reference image to compare the predictions to, in some tasks, such as image restoration, the reference image is not available. Therefore, multiple unsupervised image quality evaluation metrics are used.

These metrics are mostly based on the idea, that a good-quality image should conform to some statistical properties. Some of the most common metrics are: Naturalness Image Quality Evaluator (NIQE) [36], Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) [37] or Perceptual Sharpness Index (PSI) [13]. These metrics are not a part of this work, as a comparison to the reference image is possible. Therefore the metrics mentioned in the previous sections are used.

2.3.1 Metrics in Video Quality Enhancement

All of the previously mentioned metrics can be used only to measure the quality of a single frame. To use these metrics on a video as a whole, the average of those metrics is used the most often. However, this can lead to some problems, such as flickering or motion blur caused by the instability of the predictions. Therefore, metrics that take the temporal aspect of the video into account can be used [41].

Chapter 3

Video Super-resolution and State-of-the-art Architectures

Video super-resolution is a process of increasing the resolution and quality of a video. This task has become increasingly popular in recent years with the rise of deep learning, allowing far better results than the traditional methods. Video super-resolution shows promising results in many different areas, such as medical imaging, gaming, video surveillance, etc.

With the rise of deep learning, many different architectures and approaches have been proposed. Most of them are based on the same principles as the image super-resolution while taking the temporal aspect of the video into account [24]. Some are modeling this using 3D convolutional layers [25, 57], another commonly used approach is to use a combination of 2D convolutional layers and recurrent layers [32, 30]. In recent years, using attention mechanism to attend to different frames also became widely used [8].

This chapter focuses on the most recent and state-of-the-art methods used for video quality enhancement. It covers the most common neural network architectures, as well as the most common datasets used for training and evaluation, while also briefly covering optical flow, which is commonly used in video quality enhancement.

3.1 Single to Multi-Frame Techniques

There are different ways to produce a high-quality video from a low-quality one. They differ in many aspects, number of input and output frames of one neural network forward pass, being one.

Frame-by-Frame The simplest way to improve the quality of a video is to improve the quality of frames one by one, neural network is used to predict a single frame from a single frame. This approach is simple and computationally efficient, however, it does not take neighboring frames into account, which leads to temporal information being lost. This approach is the same as image quality enhancement but is applied to each frame of the video sequentially. This approach is not used very commonly, as it usually leads to visual imperfections, such as flickering. More about methods used for single-image super-resolution can be found in paper [54]. These methods can then be applied to each frame of the video sequentially.

N-Frame to 1-Frame This method uses multiple input frames to produce a single high-quality frame. This approach is more complex and computationally expensive because of more input data. However, this approach takes into account the temporal information from neighboring frames, which usually leads to better results, with visual imperfections, such as motion blur, flickering, etc., being reduced. A deeper insight into how the information from the neighboring frames is used can be found in paper *Rethinking Alignment in Video Super-Resolution Transformers* [45]. This is also the most common approach used in state-of-the-art methods [30, 8].

3.2 Restoration Computation

Another way to categorize the methods is by the way they compute the output frames, some methods use parallel approaches, where each frame, a group of frames respectively are processed independently, while others use recurrent approaches, where the information from the previously output high-quality frames is used to predict the next one. Each of these methods has its own set of advantages and disadvantages.

Recurrent approaches This method tries to use previously predicted frames to predict the next one. Models using this method are usually small and computationally efficient because they share parameters across frames, however, they lack parallelizability due to their recursive nature. Another big disadvantage is that they lack long-range modeling capabilities, and noise can accumulate over time.

Parallel appraches On the other hand, parallel approaches are computationally very expensive. They process each frame, set of frames respectively, independently, which leads to a large number of parameters, therefore the models are usually very large. However, they are highly parallelizable, which leads to faster training and inference. Their ability to model temporal dependencies comes from processing multiple frames and using features from neighboring ones. A well-known method using this approach is EDVR [52], which is described in more detail in Section 3.3.2 or Video Super-Resolution Transformer [8].

Combination of parallel and recurrent approaches Each of those methods has its own advantages and disadvantages and there are new ways discovered pretty often. Some recent methods, such as RVRT [32] and VRT [30] are using a combination of parallel and recurrent approaches, where neighboring frames are processed in parallel within the globally recurrent architecture. This approach is computationally efficient, while still being able to use information from long-range temporal dependencies.

3.3 Model Architectures

There are many different neural network architectures used for video quality enhancement, ranging from simple U-Net-based architectures, to generative adversarial networks (GANs) and vision transformers. This section covers the most common and best-performing architectures.

3.3.1 U-Net

U-Net is a simple and effective architecture, first proposed in 2015 by Ronnenberger *et.al* [44]. Its main purpose was a segmentation of medical images. However, it proved to be a very versatile architecture, that can be used for various image-related tasks. U-Net consists of an encoder, that compresses the input image into a lower-dimensional space, and a decoder, that generates the output image from this representation. Residual connections are used to connect the encoder and decoder, which enables the network to preserve original information.

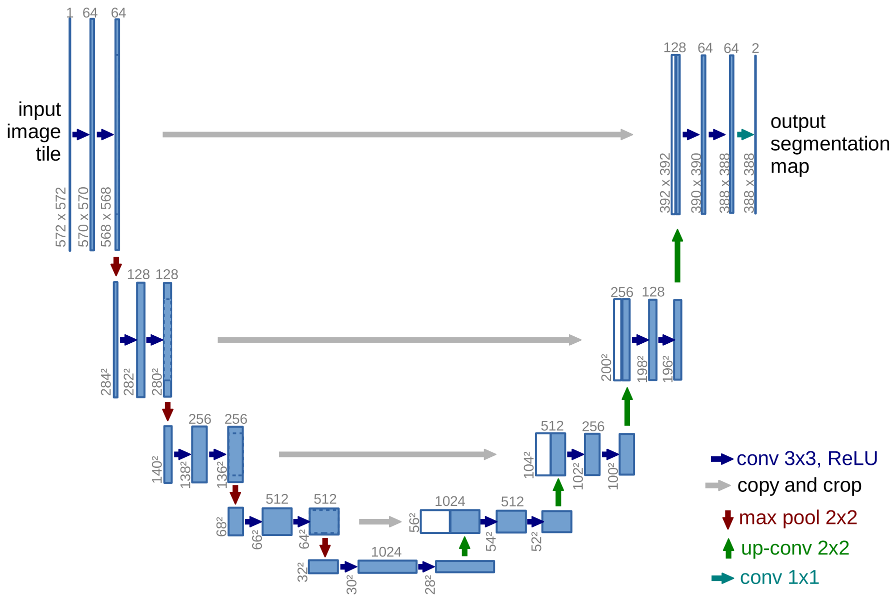


Figure 3.1: U-Net architecture. Taken from [44]

Encoder consists originally of simple blocks, where each is composed of three convolutional layers with the rectified linear unit (ReLU) [1] activation function and batch normalization [22]. After each block, max pooling is used to reduce the spatial dimensions of the image. The output from before the max pooling is saved, and then used as input to the corresponding layer of the decoder where it is concatenated with the output of the lower decoder layer.

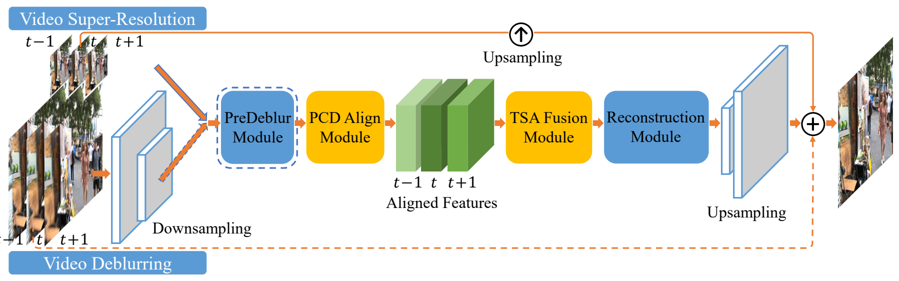
Decoder Similarly to the encoder, consists of blocks with the same components as in the encoder, however, they reduce the channel dimensionality of the image, instead of increasing it. After each block, upsampling is used to increase the spatial dimensions of the image. Up-convolutional layers are used for that.

Over time, many different variants of U-Net have been proposed, each with its quirks. Currently, U-Net-based architectures are widely used in different tasks in image processing, because of the ability of the encoder and decoder to produce high-quality features, which can then be used further. The following part of the neural network, producing the final output is called **head**. The architecture of the head can vary, depending on the task, such as linear layers for classification, convolutional layers for segmentation, etc.

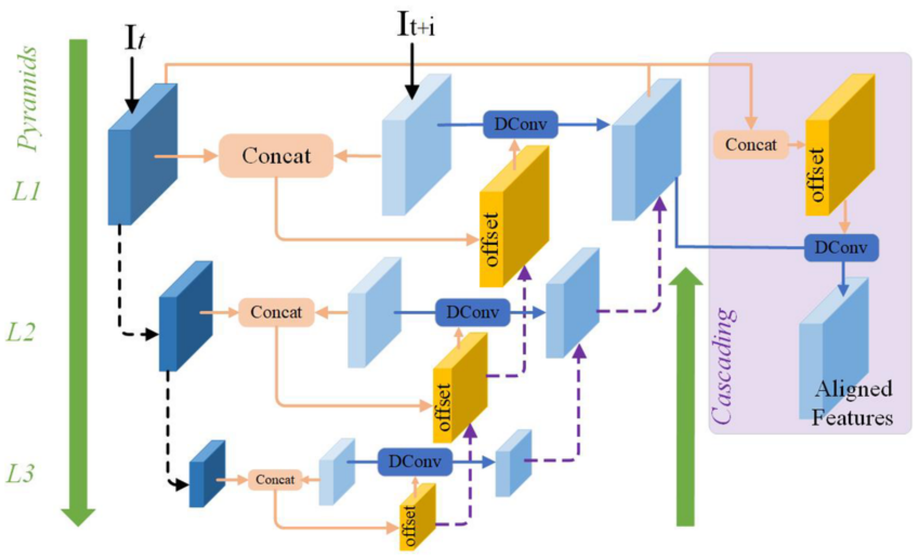
3.3.2 EDVR

Another well-performing architecture is *Video Restoration with Enhanced Deformable Convolutional Networks* (EDVR), proposed by Wang *et.al* [52]. This architecture is specifically designed for video quality enhancement. It uses a multi-frame approach, where multiple frames are used to produce a single high-quality frame. The architecture is versatile and can be used for various tasks such as super-resolution, denoising, deblurring, etc.

The architecture consists of four main parts: feature extraction, feature alignment, feature fusion and lastly reconstruction. Many novel and unusual methods are used, such as deformable convolutions and attention mechanism.



(a) Overview of EDVR architecture. Taken from [52].



(b) Feature alignment module. Taken from [52].

Figure 3.2: EDVR architecture and feature alignment module.

Feature extraction As shown in Figure 3.2a, input to the neural network are N frames, where the center frame is the frame that is being predicted. The frames are passed through a feature extraction network, that is the same for each of the frames. It consists of a series of convolutional layers, that extract features and reduce the spatial dimensionality of the frames. The output of this layer are feature maps, that are then used in the next part of the network.

Feature alignment is the next part of the network, used to align the features of the neighboring frames, that are misaligned due to the temporal nature of the video. EDVR proposes a novel approach: using deformable convolutions for this task.

Input to this layer are the feature maps from the previous part, which are processed pairwise, each frame together with the center (reference) one, as shown in Figure 3.2b. Alignment is done by Pyramid, cascading and deformable alignment. The output of this layer are aligned feature maps, that are then used in the next part of the network. This part is crucial for the task, as it aligns the features of the neighboring frames. Multiple important concepts and ideas are used in this part:

- **Pyramid** Concept commonly used in optical flow, using strided convolutions to down-sample the feature maps, by a factor of 2 to be exact, effectively obtaining a pyramid of feature maps in different spatial resolutions. Feature maps for each pair of frames are then concatenated, passed through a series of convolutional layers and used as offsets to the deformable convolutions. The neighboring frame is then aligned using deformable convolutions with these offsets.

- **Cascading** Another commonly used concept in optical flow extraction. Used here to refine the alignment of the frames. If we denote that the pyramid has L levels and lower levels have smaller spatial resolution, then cascading is done by upsampling and concatenating the offsets from level $l+1$ to the concatenated features at level l, this effectively incorporates the information from lower resolution levels to refine the alignment of the higher resolution levels.

- **Deformable Convolutions** Dai *et al.* [10] first proposed the deformable convolution layer in 2017, its purpose is to allow convolution kernels to use information from different parts of the image, rather than just the fixed grid. This is achieved by adding offsets to the convolutional kernels, that are learned during the training process. This allows the network to focus on the important parts of the image, such as edges, textures, etc. Original convolutional layers use a fixed grid, where the kernel is applied to each location p on the grid. This can be calculated as:

  $$y(p_0) = \sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n), \tag{3.1}$$

  where R denotes the grid of locations of the convolutional kernel, w are the weights of the kernel and x is the input feature map.

  As stated, deformable convolutions propose to add offsets to the grid, those offsets effectively change the grid to an irregular grid of locations where the kernel is applied as shown in Figure 3.3.

$$y(p_0) = \sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n + \Delta p_n), \tag{3.2}$$

where Δp_n are the offsets. These offsets are usually learned, they are typically fractional, to work around this, bilinear interpolation is used to calculate the value at the location of the kernel.

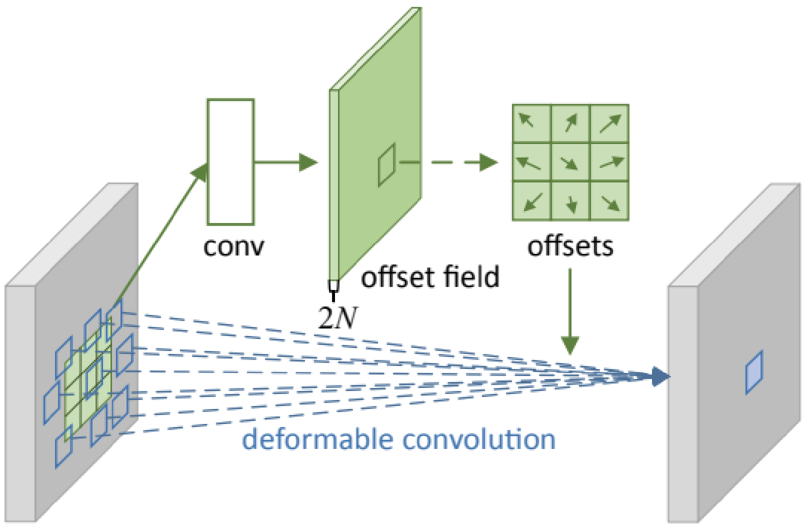


Figure 3.3: Example of deformable convolution. Taken from [10].

Feature fusion in video quality enhancement is more difficult than in a strict image context. For video, both temporal and spatial relations must be accounted for. EDVR proposes a method called TSA (Temporal and Spatial Attention) mechanism. It is based on the idea, that the more similar the neighboring frame is to the reference one, the more attention should be paid to it. This is done by calculating the similarity between the frames in an embedding space, these temporal attention maps are then multiplied with the feature maps pixel-wise as such:

$$\tilde{F}_{t+i}^{\alpha} = F_{t+i}^{\alpha} \circ h(F_{t+i}^{\alpha}, F_t^{\alpha}), \tag{3.3}$$

where F_{t+i}^{α} is the feature map of the i-th frame, $h(F_{t+i}^{\alpha}, F_t^{\alpha})$ is the similarity distance between reference and neighboring feature maps. This effectively weights the feature maps, so the network can focus on the important frames.

After this, spatial attention is used to focus on the important parts of each frame. Using convolutional layers, spatial masks are then calculated from the fused feature maps, these masks are then multiplied pixel-wise with the original fused features.

Reconstruction is the last part of the network, where the final high-quality frame is produced. To produce the final high-quality frame, the fused feature maps are passed through a series of residual blocks, as described in the following paragraph, that further refine the features. This is highly modular and can be replaced by any other architecture, depending on the task and nature of the data.

Upsampling is then used to decrease the channel dimensionality, while increasing the spatial dimensionality of the feature maps. Worth noting is that in the case of other video processing tasks, where inputs already have high resolution, this step is not skipped, but rather at the beginning of the neural network, the original images are downsampled using strided convolutions.

Residual Blocks were first proposed by He *et al.* [17] in 2015. A type of neural network architecture whose primary purpose is to help with the training of very deep neural networks. Due to the nature of backpropagation, the gradients tend to vanish in the case of very deep networks.

The proposed solution is to add a clear path for the information to flow forward, therefore the information has a clear path in backpropagation as well. This is achieved by adding a skip connection, that bypasses the convolutional layers. The output of the convolutional layers is then added to the input of the block, this is called residual connection. The formula for the residual block is as follows:

$$y = F(x, \{W_i\}) + x, \tag{3.4}$$

where $F(x, \{W_i\})$ is the mapping to be learned, x denotes input to the block, and y is the output. The operation F + x is performed by a shortcut connection and element-wise addition.

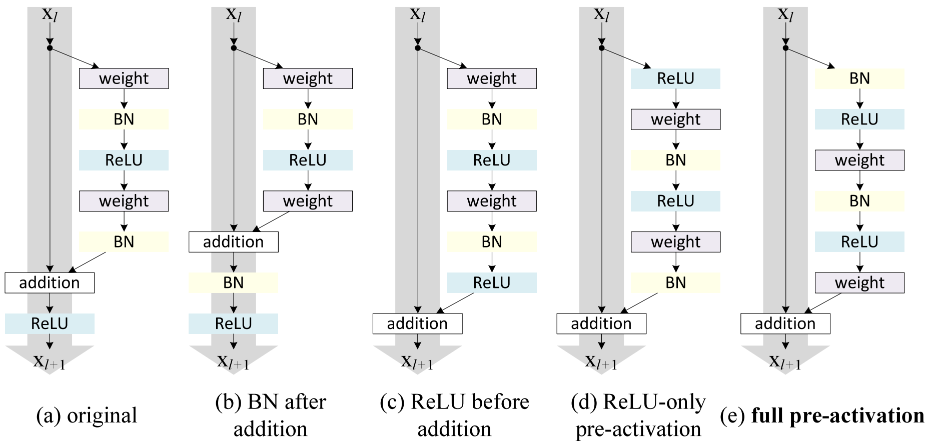


Figure 3.4: Examples of different residual blocks. Taken from [18].

This architecture is widely used, and the learnable mappings can have various forms as visible in Figure 3.4. One of the most common ones is **Bottleneck residual block**, where the mapping consists of 1×1 convolutional layer, followed by 3×3 convolutional layer, and another 1×1 convolutional layer. This is used to reduce the number of parameters, while still being able to learn complex mappings. Since the original formula requires the mapping to preserve spatial dimensions, the convolutional layers use a stride of 1 and a padding of 1 while maintaining the number of channels.

3.3.3 RVRT

An architecture proposed by Liang *et al.* [32] is currently one of the state-of-the-art architectures used in image and video quality enhancement. It's a leading architecture in multiple benchmarks, such as Vimeo90K [55]. As stated in Section 3.2, this architecture uses a combination of parallel and recurrent approaches. Authors propose a solution that, given video features $F^i \in \mathbb{R}^{T \times H \times W \times C}$, where T is the number of frames, features are then split into $\frac{T}{N}$ groups of features $F^i \in \mathbb{R}^{\frac{T}{N} \times N \times H \times W \times C}$, where N is the number of frames in a single clip. Each of these clips is then processed in parallel while using information from previous frames and previous layers in a globally recurrent manner.

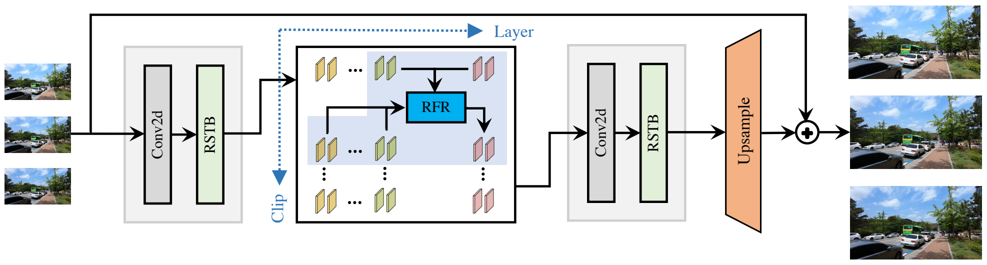
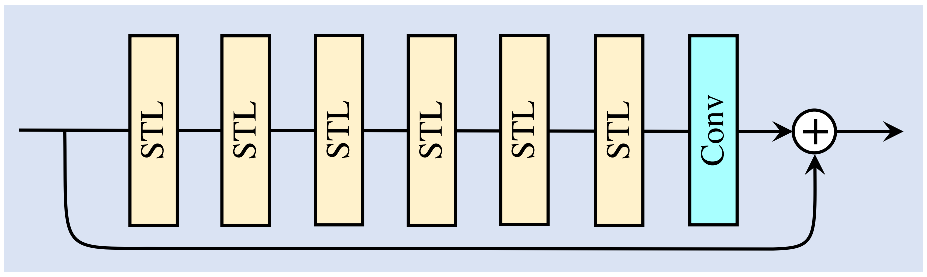


Figure 3.5: Overview of RVRT architecture. Taken from [32].

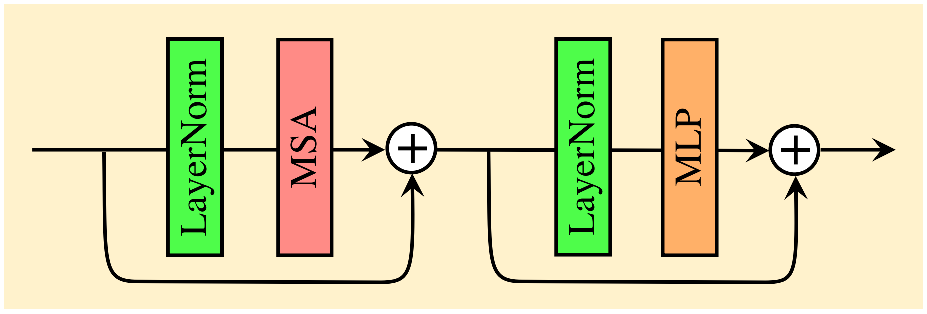
As in most other architectures, architecture consists of three main parts: feature extraction, feature refinement and high-quality frame reconstruction. However, authors use novel methods in multiple places in the neural network, such as Residual Swin Transformer Blocks [34] or Guided Deformable Attention mechanism, which is used to align the features from different clips.

Feature extraction is done by a sequence of layers, where firstly features are extracted by a simple convolutional layer, and then those features are passed through a series of Residual Swin Transformer Blocks (RSTB), further described in the following paragraph. In case of tasks different than super-resolution, the input features are downsampled before the RSTB layers, to reduce the computational cost.

Residual Swin Transformer Blocks are a modification of the Vision transformers, as proposed in [11] have become a standard architecture used in image processing tasks, however, due to the $O(n^2)$ complexity of the attention mechanism [51], they are very expensive computationally, which makes them not-so suitable for video and image processing. To address this issue, Liu *et al.* proposed a version called Swin Transformer [34]. This method works by splitting the image of input shape $H \times W \times C$ into non-overlapping patches of shape $\frac{HW}{M^2} \times M^2 \times C$, where $M \times M$ is the size of the patch and $\frac{HW}{M^2}$ is the total number of patches. This effectively reduces the computational complexity of the attention mechanism. This is computed in parallel multiple times, following [51] and results are concatenated. However, if the partitioning of the image is fixed for different layers, the network can not connect different parts of the image. Because of that, shifted windows are used in alternation with regular windows. Windows are shifted by $([\frac{M}{2}], [\frac{M}{2}])$ pixels before partitioning. This allows the attention mechanism to attend to the whole image. Residual Swin Transformer Blocks [31] are then just successive Swin Transformer Blocks followed by a convolution layer, whose output is then added element-wise to the input of the block, as shown in Figure 3.6a.



(a) Residual Swin Transformer Block. Taken from [31].



(b) Swin Transformer Layer. Taken from [31].

Figure 3.6: Architecture of the Residual Swin Transformer Block and its components.

Feature Refinement follows the feature extraction. The video features are split into clips $F \in \mathbb{R}^{\frac{T}{N} \times N \times H \times W \times C}$, where N is the number of frames in 1 clip, T is the original number of frames of the video, and C, H, W are the number of channels, height and width of the feature maps. Each of these clips is then processed in parallel by a sequence of recurrent feature refinement blocks in a globally recurrent framework, utilizing the information from the previous clips, as shown in Figure 3.7. This is done by aligning the features from the previous clip using the Guided Deformable Attention (GDA). Afterwards, the aligned features of the previous clip, together with features output by the previous refinement layers are used to predict the output features of the current clip and current layer. This can be generalized as such:

$$\begin{aligned}\hat{F}^i_{t-1} &= GDA(\hat{F}^i_{t-1}, O_{t-1\to t}, \hat{F}^{i-1}_{t-1}, F^i_t)\\ F^i_t &= RFR(F^0_t, F^1_t, \ldots, F^{i-1}_t, \hat{F}^i_{t-1}),\end{aligned} \tag{3.5}$$

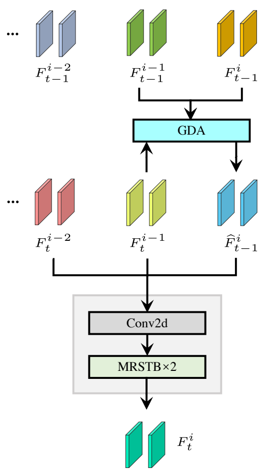


Figure 3.7: An overview of the feature refinement module. Taken from [32]

where F^0_t, F^1_t, ..., F^{i-1}_t are the features from the feature extraction, previous layers respectively, RFR is the recurrent feature refinement module, that consists of multiple convolution layers together with RSTB blocks and GDA is the guided deformable attention mechanism. The RSTB blocks are modified, to use 3D attention maps, so they can attend to the temporal and spatial features of the features.

3.4 Optical Flow

Optical flow is a pattern of motion of objects in a scene caused by the relative motion between the observer and the scene [20]. Optical flow is represented by a two-dimensional vector field, where each vector is a displacement vector showing the movement of points from one frame to another. Optical flow is computed from a sequence of images. It is calculated by various methods, from simple methods, such as Lucas-Kanade method [35], to more complex methods, such as deep neural networks. In this work, optical flow is used to align the features of the neighboring frames inside a neural network. To extract it, a neural network called Recurrent All-Pairs Field Transforms for Optical Flow (RAFT) [50] is used. This network is state-of-the-art in optical flow extraction and is used in many different tasks, such as video quality enhancement, video object segmentation, etc.

3.4.1 RAFT

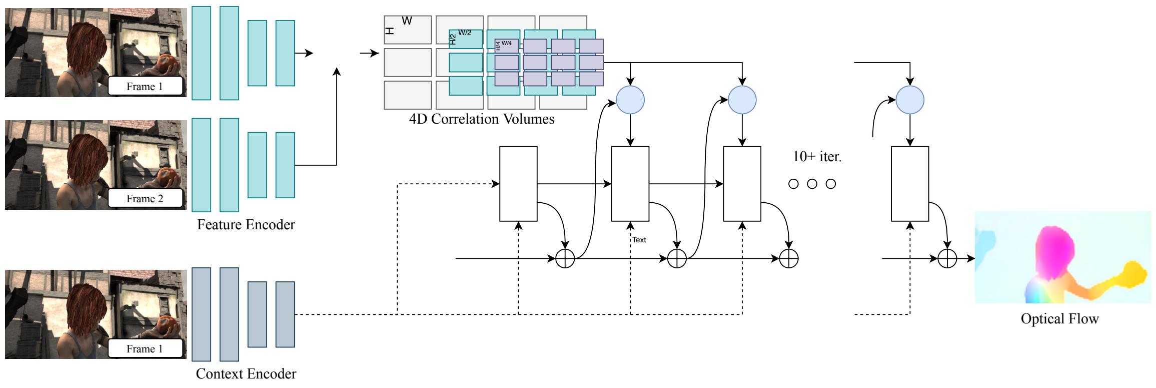


Figure 3.8: Architeture of the RAFT model. Taken from [50].

RAFT is a neural network architecture, first proposed in 2020 by Teed *et al.* [50]. Input to the neural network are 2 frames: F_1 and F_2. The output is a dense displacement field (f_1, f_2) mapping each pixel (u, v) in F_1, to its corresponding location $(u', v') = (u+f^1(u), v+f^2(v))$ in F_2.Raft architecture consists of 3 main parts:

Feture Encoder is a part of the network, responsible for extracting features from both frames using a series of convolutional layers arranged in residual blocks. Feature extractor shares weights across both of the frames. Additionally to the feature extractor, a context network is also used. This network shares architecture, but not weights with the feature extractor. It is used to extract extra features only from the first frame.

Correlation layer is calculated from the features extracted in the previous step. This is done by calculating the dot product between all pairs of feature vectors. The result is a tensor $C \in \mathbb{R}^{H \times W \times H \times W}$, where H and W are the height and width of the feature maps. Then a correlation pyramid is constructed, this is done by pooling the correlation volume across the last 2 dimensions. This effectively gives information about both small and large displacements.

Iterative updates of optical flows is done by estimating a sequence of optical flows $\{f_1, \ldots, f_N\}$. In each iteration, displacement update Δf is predicted. This is then added to the previous one as such:

$$f_{n+1} = f_n + \Delta f. \tag{3.6}$$

Then updating of the optical flow is done recursively. This is done by sampling the correlation map with the current flow estimate and then using this correlation, together with current optical flow prediction and latent hidden state to predict a new hidden state. That is then further processed to generate the displacement update Δf that is then added to the current optical flow prediction. This process is repeated N times to get the final optical flow estimate. The final estimate is in a lower resolution than the image, so it is upsampled with a combination of convolutions and a weighted combination of neighboring pixels.

3.5 Datasets

This section covers the most common datasets used for training neural networks for video quality enhancement and the types of data that are present. Gathering data is a significant part of training a neural network. In the case of video quality enhancement, multiple difficulties arise, such as the big differences between different visual imperfections and their primary causes. To give an example, video taken in low light conditions will be mostly noisy, while video taken from a moving vehicle will be heavily blurred. It is very difficult to train a neural network that handles all these cases similarly. Therefore, it is crucial to choose the right dataset for the task.

Table 3.1: Comparison of Video Super-Resolution Datasets

Dataset	Number of Samples	Types of Videos	Lengths
REDS [38]	300 clips	Diverse scenes, dynamic	100 frames per clip
Vimeo90K [55]	89,800 clips	Various everyday scenes	7 frames per clip
Vid4	4 clips	Classic test sequences	40-100 frames per clip

REDS [38] stands for Realistic and Dynamic Scenes dataset. This dataset was first proposed in the NTIRE19 challenge and has since become one of the most widely used datasets for video quality enhancement. As the name suggests, it contains mostly dynamic scenes, such as moving vehicles, people, etc. It contains images with different issues, such as blur, compression, low resolution, etc. The blur is caused by merging subsequent frames of the original 120 frames-per-second video. The dataset contains 300 video sequences in total, where each sequence contains 100 frames in the resolution of 720×1280 and 180×320 for the downscaled images.

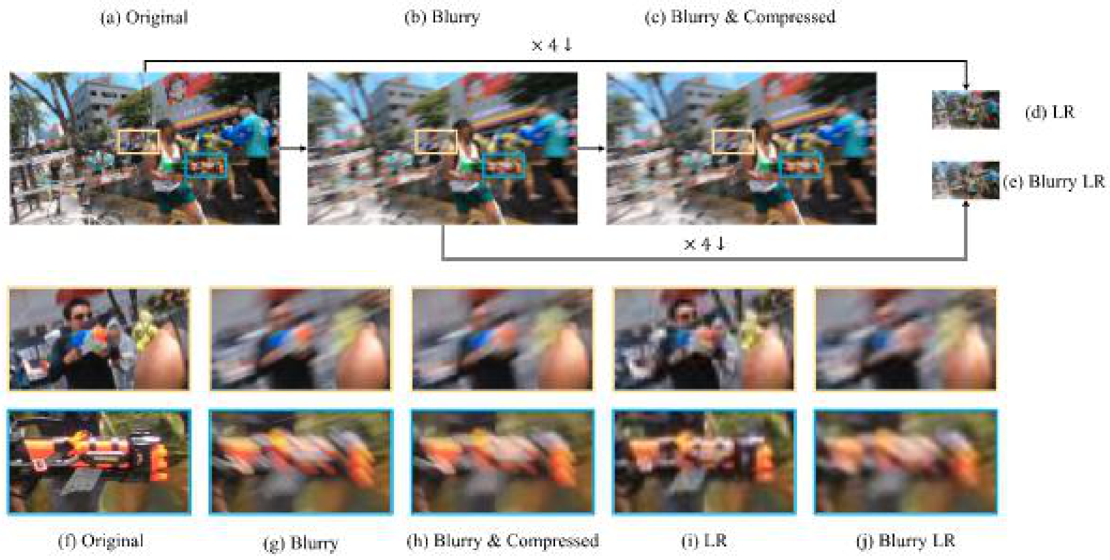


Figure 3.9: Example of frames from REDS dataset. Taken from [38].

Vimeo90K [55] is a dataset first proposed by Xue *et al.* in paper *Video Enhancement with Task-Oriented Flow*. This dataset is one of the largest datasets used for video quality enhancement, containing 89,800 video clips in total downloaded from Vimeo. The dataset contains videos with various issues, such as low camera resolution, motion blur, compression artifacts, etc. The primary purposes of this dataset are video super-resolution, video denoising and frame interpolation.

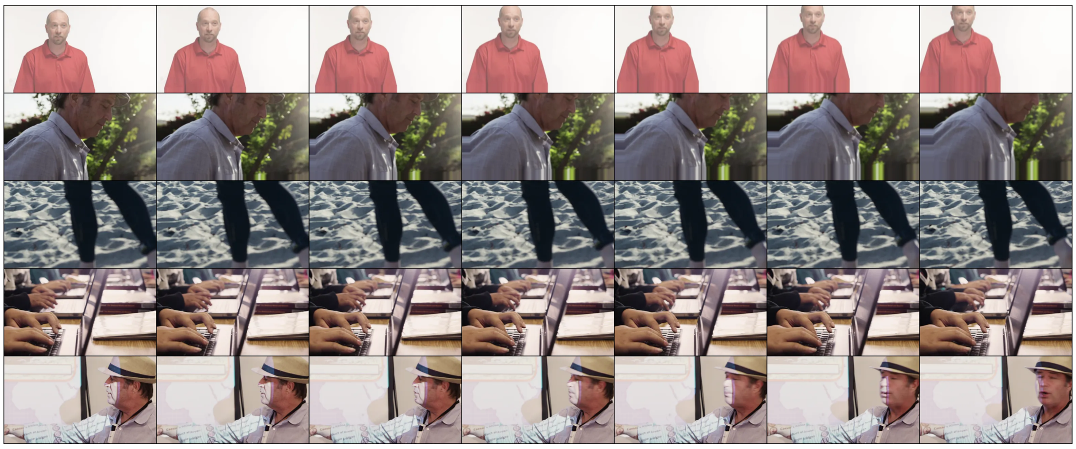


Figure 3.10: Example of frames from Vimeo90K dataset. Image taken from [40].

3.5.1 Vid4

is a small dataset consisting of only 4 videos, each with 40-100 frames. Consists of 4 sequences: `Walk`, `City`, `Foliage`, and `Calendar`. These sequences are well-established in the literature for assessing the performance of video super-resolution methods. It is generally not used for training, but rather for testing and evaluation of the neural network.

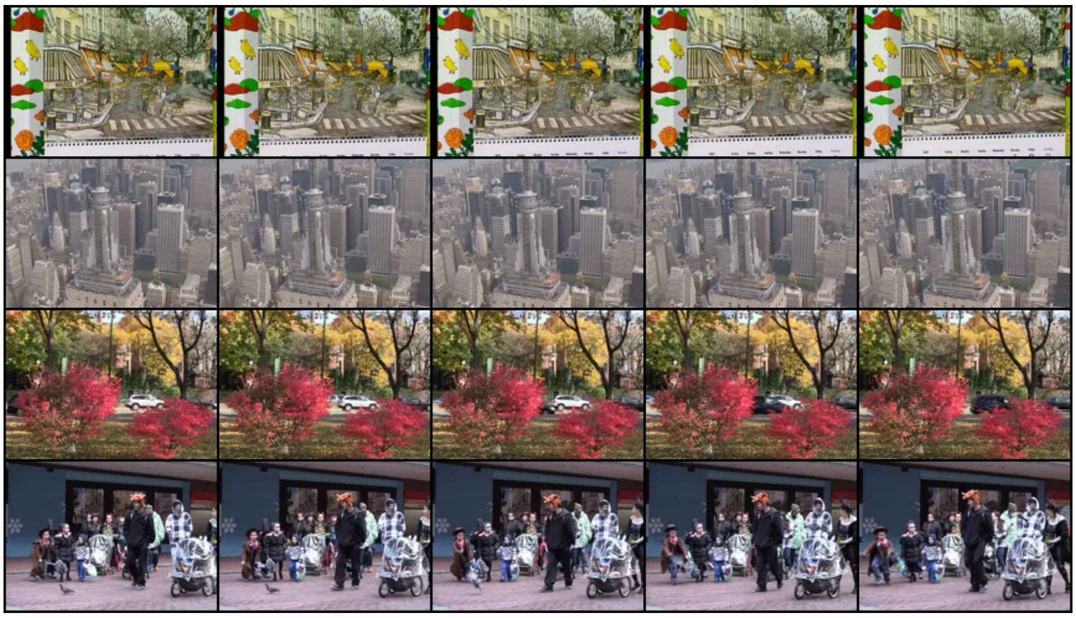


Figure 3.11: Example frames from each sequence of Vid4 dataset. Image taken from [40].

Chapter 4

Problem Definition and Proposed Solution

Video super-resolution is the task of increasing the resolution and overall quality of a video. The proposed solution establishes a neural network architecture that uses deformable convolutions and explicitly calculated optical flow. The proposed architecture is small and fast, while still being able to produce high-quality videos. This can be used in various areas, for example in the automotive industry or medical imaging.

This work is focused on a creation of a novel module, that uses explicitly calculated optical flow as an offset to deformable convolutions. This module is then used in a U-Net-like architecture, which was proven to work well in video super-resolution tasks. Different configurations of the module are proposed.

This chapter first defines the problem and draws the motivation for the work. Then, details of the work are presented, such as the dataset used, the neural network architecture and training details.

4.1 Problem Definition and Motivation

Video super-resolution can be interpreted as a task of increasing the resolution and quality of a sequence of frames. Different from image super-resolution, this task can also incorporate features from the neighboring frames. Most of the state-of-the-art methods use a feature alignment module to align those features towards the reference one, and then extract them using classical methods, such as convolutional layers. However, many of these methods are computationally expensive and difficult, even impossible to use in real-time scenarios.

This work is focused on the creation of a novel module, that can be used in different architectures as a feature alignment module. The goal is to prove that using this module can lead to better results, outperforming the models without this module.

Given that this module requires explicitly calculated optical flow, it requires a dataset, in which motion is present and visible. Therefore, the REDS [38] dataset is used. This dataset while still being quite large, contains videos from mostly real-life scenarios, such as moving vehicles, people, etc.

4.2 Dataset Preparation and Choice

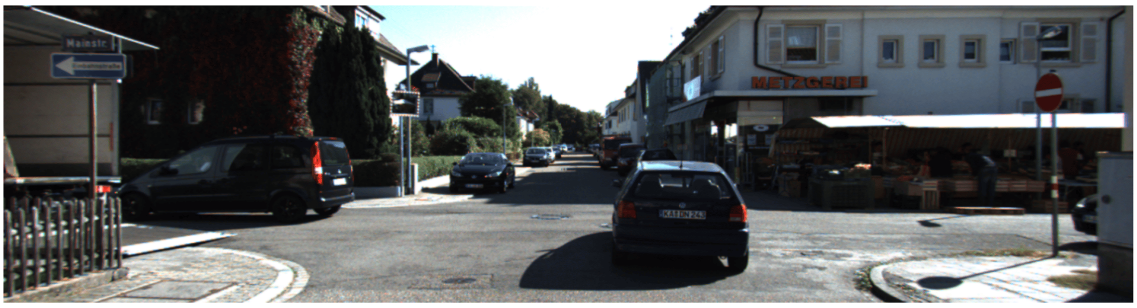
As stated, the REDS dataset is used to train and validate the network. This dataset doesn't contain an explicitly calculated optical flow, therefore a state-of-the-art model is used to predict that. Model RAFT [50] was chosen for this task. More about this architecture is written in Section 3.4.1. This model is chosen because of its high performance and speed, while still being able to produce close to state-of-the-art results. For each triplet of input frames, two optical flow maps are extracted, from the reference frame to the previous and the next frame. Two image pairs are therefore passed through the RAFT model (T_i^l, T_{i-1}^l) and (T_i^l, T_{i+1}^l), where T_i^l is the reference frame, and T_{i-1}^l and T_{i+1}^l are the previous, next frame respectively. These are then used to align the features of the neighboring frames towards the reference one.

To extract optical flow, pre-trained weights for RAFT were used, as provided by the authors of the original paper. Weights from RAFT trained on KITTI [33] dataset were used, as this dataset contains data from dynamic scenes, similar to the REDS dataset.

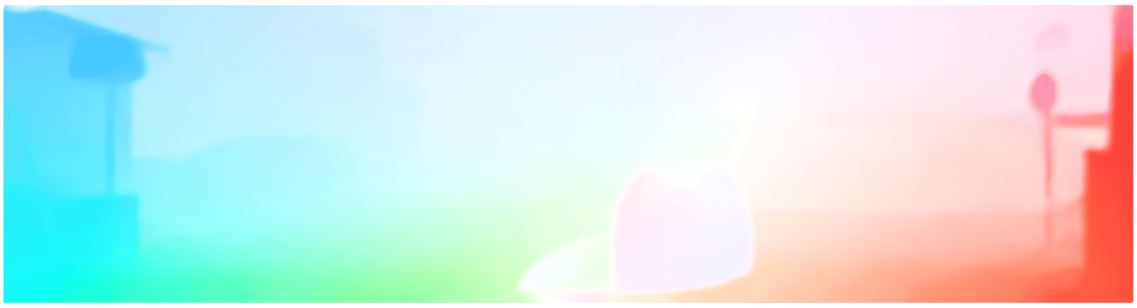
The task of video super-resolution also introduces a new hyperparameter to choose, that is the number of frames used to predict the high-quality frame. Most methods use an odd number of frames and predict the middle frame. This thesis tests the module on 3 frames, given reference frame T_i^l is the low-quality frame in timestep i, and T_i^h is the corresponding high-quality frame, the module is using frames T_i^l, T_{i-1}^l and T_{i+1}^l to predict T_i^h. Therefore, the optical flow mapping frame T_i^l to T_{i-1}^l, T_{i+1}^l respectively, has to be calculated.

Another decision to be made is whether to use extra augmentations to enlarge the dataset. Creating augmentations that represent visual imperfections caused by video in real-life scenarios is a difficult problem on its own. Because of that, the proposed architecture being small, and the dataset being large enough, no augmentations are used. For different architectures used with this module, this can be changed.

(**a**) Frame at time t.



(**b**) Frame at time $t+1$.



(**c**) Resulting optical flow.

Figure 4.1: Sample images from KITTI [33] dataset and resulting optical flow. Taken from [42].

4.3 Neural Network Architecture

The proposed module only aligns the features between the frames. There are still other tasks to be done, such as feature extraction, feature refinement and upscaling. For this, a combination of already existing methods is used. To extract features, U-Net-like (2.1.2) architecture is used, as it is a simple and effective architecture proven to work in various computer vision-related tasks. Also, there are existing implementations of this architecture, with different backbones and heads, such as Segmentation Models Pytorch, which is further discussed in Section 5.1.

To further refine features, a sequence of bottleneck residual layers (3.3.2) is used. This is done in such a way that this module can be easily replaced by any other architecture, depending on the task and nature of the data. In the end, upscaling is done by a series of pixel-shuffle [46] layers. This is a simple and effective way to increase the resolution of the image. Again, changing this part of the architecture is possible, depending on the task and nature of the data. The whole architecture can be seen in Figure 4.2.

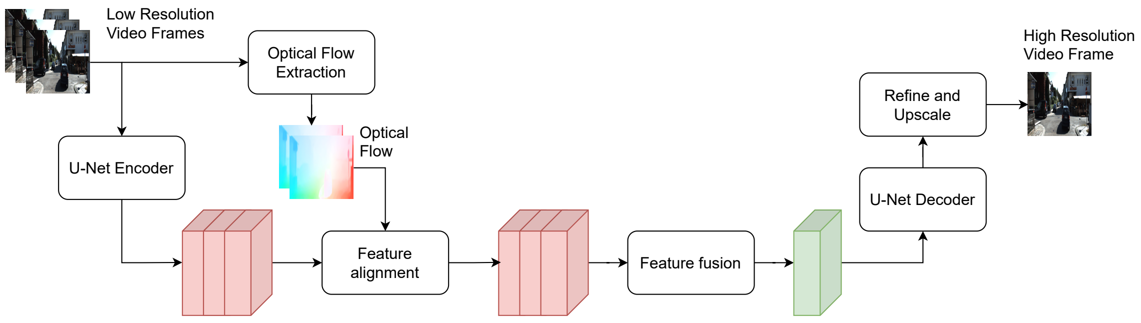


Figure 4.2: Schema of the proposed solution. Details about the concrete modules used are discussed further in this chapter.

4.3.1 Feature Extraction

Firstly, multi-resolution features from frames T^l_{i-1}, T^l_i and T^l_{i+1} are extracted using a backbone of the U-Net. Weights of the backbone are shared across all frames, which is done by reshaping mini-batch of inputs $X \in \mathbb{R}^{B \times T \times H \times W \times C}$ to $\mathbb{R}^{B*T \times H \times W \times C}$, where B is the batch size, T is the number of frames, H and W are the height and width of the image, and C is the number of channels. This is then passed through the backbone, and reshaped back to $\mathbb{R}^{B \times T \times H \times W \times C}$.

4.3.2 Feature Alignment

The feature alignment module is the main part of this work. Given mini-batches of features (F_{i-1}, F_i, F_{i+1}), where F_i is the reference frame, and F_{i-1} and F_{i+1} are the neighboring frames, the module aligns the features from the neighboring frames to the reference frame. Outputs of each resolution level of the U-Net backbone are used, this is done to align features with different spatial resolutions. Also, this enables processing basic features in the first levels, and more complex features in the higher levels.

As the outputs of the U-Net backbone grow in the number of channels and their spatial resolution decreases, matching optical flow maps are downsampled in a bilinear manner to match the spatial dimensions of the features. With decreasing spatial dimensions, the optical flow maps are also divided by 2 element-wise, to adapt the magnitude to the decreasing spatial dimensions. Optical flow for each pair of frames, (F_{i-1}, F_i) and (F_{i+1}, F_i) is then reshaped and expanded to the shape of:

$$deformable\_groups * kernel\_width * kernel\_height * 2 \ \times H \times W, \tag{4.1}$$

where deformable groups are a hyperparameter of the network, kernel width and height is the size of the convolutional kernel and H and W are the height and width of the feature maps. Then, fixed relative offsets of the convolutional kernel $((-1,-1),(-1,0),\dots,(1,1))$ are added across the first dimension, to get the final offsets. These offsets are then used as input to the deformable convolution layer. This effectively aligns the features from the neighboring frames to the reference frame, on each level of the backbone. Experiments were also conducted using learnable mapping, such as convolutional layers to reshape and calculate the offsets given the raw optical flow. More about this is discussed in the Chapter 6. Since pair (F_i, F_i) maps the features from the reference frame to itself, the resulting optical flow would be a zero tensor. Therefore, a series of default convolutional layers is used, only to match the output shape of the reference frame to the ones of the neighboring frames. The schema of the feature alignment module can be seen in Figure 4.3.

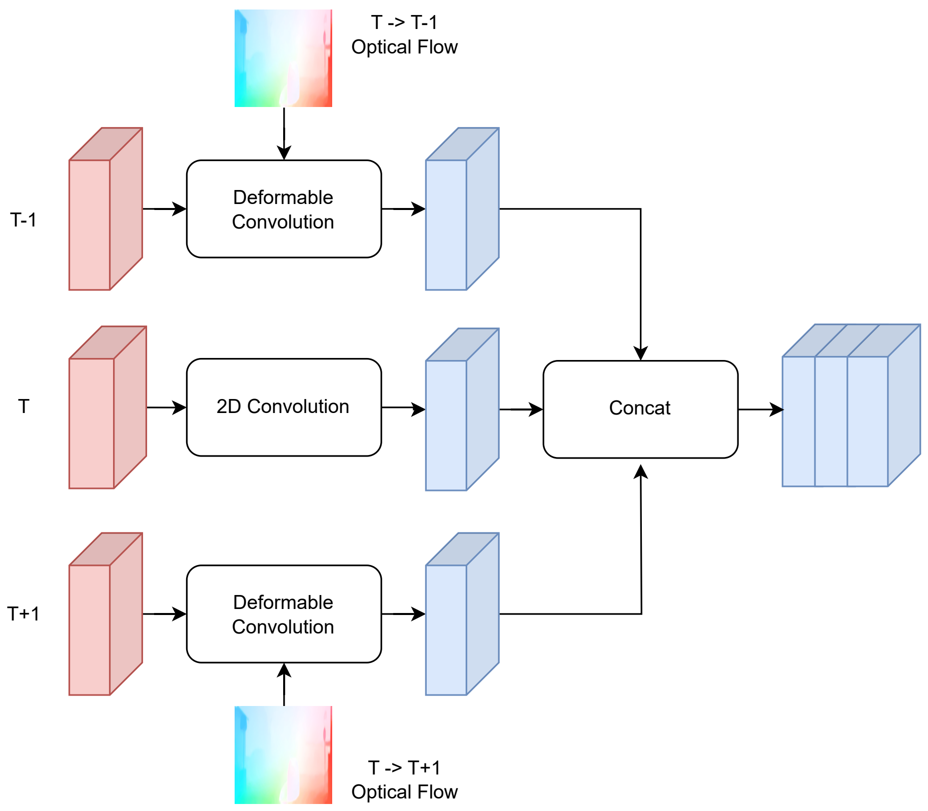


Figure 4.3: Schema of the feature alignment module.

4.3.3 Feature Fusion

Technically, not a part of the feature alignment module, but still crucial for the task. Feature fusion is done by a series of convolutional layers, that combine the aligned features from both neighboring frames with the reference frame. This is done by first concatenating the features along the channel dimension and then passing them through a sequence of convolutional layers, reducing their channel dimensionality. In this work, the number of channels is reduced by a factor of 3, to match the number of features from a single frame, to be then passed further through the decoder part of the architecture. As this module is used in each level of the U-Net architecture, the fused features are also processed further by the decoder part of the network, where they're combined with the features from the lower levels.

Experiments were done with different ways of fusing the features, such as using fused features of the lower spatial resolution as correction to the higher resolution features, etc. More about this is discussed in the Chapter 6.

4.3.4 Feature Refinement

The output of the U-Net decoder is effectively a single mini-batch of features $F_i^h \in \mathbb{R}^{B \times C \times H \times W}$, where B is the batch size, C is the number of channels, and H and W are the height and width of the feature maps. Because of the network still being relatively small, these features are then passed through a series of bottleneck residual blocks 3.3.2, that further refine the features, enhancing the quality of the final output. This part of the network is highly modular, and can be easily replaced by any other architecture, depending on the task and nature of the data.

4.3.5 Upscaling

To increase the resolution of the image, a series of pixel-shuffle [46] layers in combination with convolutional layers is used. This is a simple and effective way to increase the resolution of the image, while still being able to produce high-quality results. This part of the network is also highly modular. The number of pixel-shuffle layers can be changed depending on the desired scaling factor.

4.4 Training Details

To train the neural network, it is crucial to choose the right loss function and optimizer. For the loss function, Mean Squared Error is used. This is a simple and effective loss function, that is widely used in image processing tasks. It is calculated as the average squared difference between each pixel of the original and predicted image. To optimize the neural network, Adam [26] optimizer is used. This is a widely used optimizer, that is known for its fast convergence and good generalization capabilities. It is currently one of the most widely used optimizers and is considered a standard in the field of deep learning.

To evaluate the performance of the neural network, Peak Signal-to-Noise Ratio [21] and Structural Similarity Index [21] are used. These are widely used metrics for measuring the quality of an image. The higher the value of the PSNR and SSIM, the better the quality of the image.

Chapter 5

Implementation Details

In this chapter, the implementation details, such as used technologies, programming languages, patterns and reasoning behind those choices are discussed. The architecture of the program, neural network and modules is presented.

5.1 Technologies Used

As is the standard in the field of deep learning, Python programming language is used. The reason for this is the wide range of libraries, tools and frameworks available for this language. Python is also widely used in the scientific community due to its simplicity and high-level nature, abstracting away the low-level details such as memory management, etc.

Pytorch

To develop the neural network, Pytorch[1] is used. It offers an object-oriented API, that is easy to use and develop in. Pytorch is also widely used in the scientific community and is considered one of the best deep-learning frameworks available. Another benefit is the ability to use graphics card acceleration, which speeds up the training process significantly due to its highly parallel nature. This is available on many different graphic card backends, such as CUDA or METAL

Pytorch Lightning

To further abstract the training, Pytorch Lightning[2] is used. This is a high-level interface for Pytorch, that abstracts away the low-level details of training, such as setting up the training loop or logging. The main reason behind choosing this framework was its ability to easily scale the training to multiple GPUs, even multiple nodes, which is crucial for training large neural networks.

[1] https://pytorch.org/
[2] https://lightning.ai/docs/pytorch/stable/

Segmentation Models Pytorch

To use the U-Net architecture, a library called Segmentation Models Pytorch[3] was used. This library offers an API for many different architectures, such as U-Net which was used in this work. These architectures are easily configurable, with different backbones, provided by Pytorch Image Models (TIMM)[4], heads and number of layers. This library also offers pre-trained weights for many different architectures, which can be used to speed up the training process.

Weights and Biases

Weights and Biases, or W&B[5] is a tool used for logging, visualization and experiment tracking. It provides integration with Pytorch Lightning, which makes logging and tracking of the model hyperparameters very easy. It also offers a wide range of visualizations, such as visualizing model gradients throughout the training process, which can be helpful in debugging the neural network.

Other libraries were used for smaller tasks, such as:

1. **MMCV**[6] – for an implementation of the deformable convolution layer.
2. **Matplotlib**[7] – for visualization of the data, metrics and results.

5.2 Architecture of the Program

The codebase is divided into 2 main folders: `src` and `scripts`, where `src` contains the main parts of the source code, such as neural network architecture, loss functions, training loop, etc. and `scripts` contains scripts for data preparation and evaluation. Inside the `configs` folder, multiple JSON files are located, that contain the configuration options for the neural network, such as the number of layers, number of channels, etc. These are then loaded into the program and proper classes are instantiated using factory methods.

Src folder contains the main codebase of the program, such as classes required by Pytorch Lightning or all the different datasets and neural network architectures. Those are located inside a `arch` subfolder, where each module is split into a separate file. Code for `SRUnetDecoder` and `SuperResolutionUnet` is adapted and edited from Segmentation Models Pytorch, however rest of the source code is written from scratch.

The main training script is located in the `main.py` file, where the neural network is trained, and the results are logged to Weights and Biases, or Tensorboard[8] respectively. This script is also responsible for loading the configuration, setting up the training loop, saving model checkpoints, etc.

[3] https://smp.readthedocs.io/en/latest/
[4] https://huggingface.co/docs/hub/timm
[5] https://wandb.ai/site
[6] https://mmcv.readthedocs.io/en/latest/
[7] https://matplotlib.org/
[8] https://www.tensorflow.org/tensorboard

Scripts folder contains scripts for data preparation and evaluation. To prepare the training and validation dataset, the RAFT model is used, source code was adapted and edited from the official implementation of the model[9]. The script to extract optical flow using this model `prepare_dataset.py` is located inside the `RAFT` subfolder, it was created based on the existing evaluation scripts from the official RAFT repository. This script is responsible for loading the dataset, passing it through the model and saving the optical flow maps in a proper format.

Then, scripts for preparing a single video, and evaluating the neural network are present in this folder. These scripts are responsible for loading the video provided by the user, saving it in a proper format and then upscaling it to the desired resolution.

[9]https://github.com/princeton-vl/RAFT

Chapter 6

Experiments

In this chapter, the details of the experiments and training setup are presented. Different architectures of the feature alignment module are tested, together with different configurations of the neural network. The results are then compared to the baseline models, such as single-frame U-Net and bilinear interpolation.

6.1 Training Setup

Different configurations of the neural network are tested and trained. Multiple graphics cards of type Nvidia A5500, and A5000 respectively were used for training. The experiments were conducted for a maximum of 1000 epochs, but the training was stopped early if the validation metrics weren't improving.

To speed up the training process, the optical flows were pre-calculated and saved to disk. This was done to avoid recalculating the optical flow for each epoch, as this is a time-consuming process. The optical flow was calculated using the RAFT model, as described in Section 3.4.1. The dataset used for training and validation was REDS, containing 300 video clips in total, each consisting of 100 frames in the resolution of 180×320 and ground truth high-quality frames in the resolution of 720×1280, however, the images were padded to the resolution that is processable by the U-Net architecture. Models were trained on different batch sizes, differing from 4 to 24, based on the configuration of the neural network. `ModelCheckpoint` [1] provided by Pytorch Lightning was used to save the model weights after each epoch, if the validation metrics improved. Adam [26] optimizer was used, together with a learning rate ranging from $1e-4$ to $3e-4$, depending on the configuration of the neural network. The learning rate was adapted during the training process using a learning rate scheduler [2].

6.2 Single Frame vs Multi-Frame

The first experiment was conducted, whether processing multiple frames at once and fusing the aligned features would show better results than single-frame methods, such as a simple U-Net architecture with a single frame as input or bilinear interpolation. To adapt the U-Net architecture for super-resolution, an upsampling module consisting of a series of pixel-shuffle layers, together with convolutional layers was added to the end. To adapt the

[1] https://lightning.ai/docs/pytorch/stable/api/lightning.pytorch.callbacks.ModelCheckpoint.html
[2] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.CosineAnnealingLR.html

architecture for processing multiple features, a feature alignment and fusion module were added instead of classical U-Net skip connections. How this module works, is shown in Figures 4.2 and 4.3. A comparison of the results is shown in Figure 6.1, where zoomed-in parts of multiple images are shown. The results show that using multiple frames and fusing the aligned features can improve the quality of the images significantly, mostly in details such as sharp edges. The results are also compared to bilinear interpolation, which is surpassed significantly by both single and multi-frame methods.

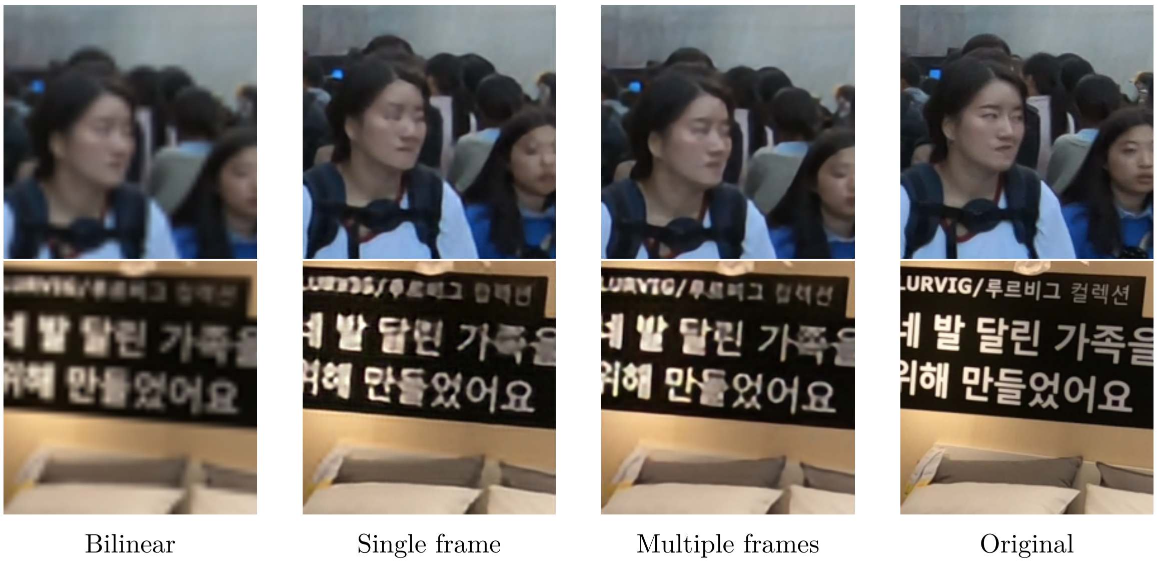


Figure 6.1: Comparison of different methods.

6.3 Different Configurations of U-Net Architecture

The second experiment was conducted, whether using a different backbone with a different number of levels would prove beneficial. Pytorch Image Models provide implementations of different backbones, so ResNet [17] and Vgg [47] were compared, where Vgg19 was used with 4 levels, compared to ResNet34 with 5 levels. The results in Table 6.1 show that using Vgg19 with 4 levels gives slightly better results, this could be due to the fact of higher output channels of the decoder (32 compared to 16 in ResNet34). Therefore, for future experiments with the proposed module architecture, as described in Section 6.4, Vgg19 with 4 levels was used.

Table 6.1: Performance Metrics for ResNet34 and Vgg19 backbones

Model	**Validation**		**Training**	
	SSIM	**PSNR (dB)**	**SSIM**	**PSNR (dB)**
ResNet34	0.790	27.70	0.824	27.41
Vgg19	0.800	28.02	0.845	28.33

6.4 Proposed Module Architecture Experiments

6.4.1 Cascading of Lower-Level Features

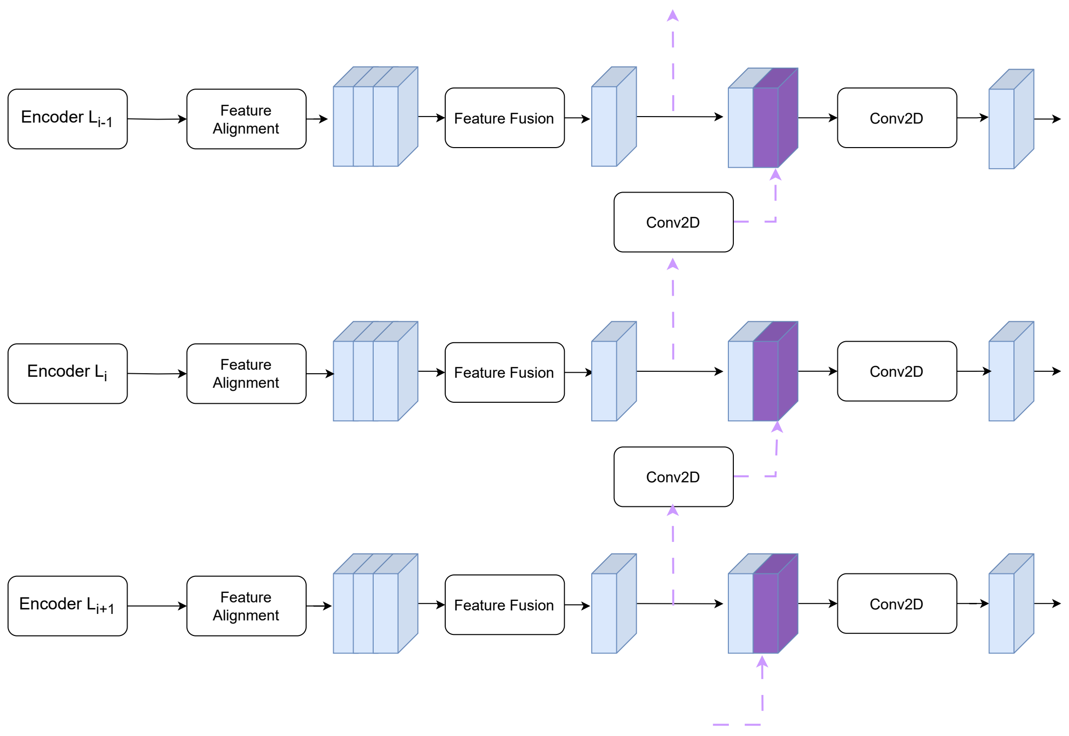


Figure 6.2: Schema of the cascading of lower-level features. Lower-level features (purple dashed lines) get concatenated with current-level fused features, further processed and passed into the decoder.

This section describes the experiments conducted with the proposed feature alignment module. The first proposed test was to use a cascading of lower-level features from the feature-alignment module, to concatenate them with current-level fused features, further process them and pass them through the decoder. Schema of this approach can be seen in Figure 6.2. The results in Table 6.2 show that this approach very slightly improves the performance of the network, while also showing the training metrics being very similar to the validation ones, meaning further training could prove beneficial.

Table 6.2: Performance Metrics for Cascading of lower-level features

Model	**Validation**		**Training**	
	SSIM	**PSNR (dB)**	**SSIM**	**PSNR (dB)**
Vgg19	0.800	28.02	0.845	28.33
Vgg19 + cascading	0.810	28.05	0.828	27.91

6.4.2 Learnable Mapping of Optical Flow

All of the previous tests used raw optical flow, reshaped and expanded to the shape fitting the deformable convolution layer, this process is described in Section 4.3.2. The next experiment used a single convolutional layer to preprocess the raw optical flow, this effectively creates a learnable mapping of the raw optical flow to the offsets used in the deformable convolution layer. In Table 6.3, final results and comparison can be seen, proving that learnable mapping of the optical flow improves the results significantly, both in terms of PSNR and SSIM metrics. A comparison to bilinear interpolation and single-frame U-Net can be seen as well. Further visual results can be seen in Figure 6.3.

Table 6.3: Performance Metrics for Final Evaluation of the Models

Model	**Validation**		**Training**	
	SSIM	**PSNR (dB)**	**SSIM**	**PSNR (dB)**
Bilinear	0.700	25.70	0.710	25.50
Single-frame U-Net	0.790	27.40	0.815	27.15
ResNet34	0.790	27.70	0.824	27.41
Vgg19 + cascading	0.810	28.05	0.828	27.91
Vgg19+cascading+learnable OF	0.818	28.28	0.892	30.17

6.5 Summary of Experiments

The first experiment proved that using multiple frames and fusing them, shows better results than a single-frame method, also surpassing bilinear interpolation. Further experiments showed that experimenting with different U-Net backbones and leaving the channel dimensionality of the fused features as high as possible can improve the performance significantly. To further refine the high-level features on the higher levels of the network, cascading of lower-level features can be used, which slightly improves the performance and makes the details of the images sharper. Afterward, preprocessing the raw optical flow by a single convolution layer, creating a learnable mapping, improves the performance significantly, visual results together with the performance metrics are shown in Figure 6.3 and Table 6.3 respectively.

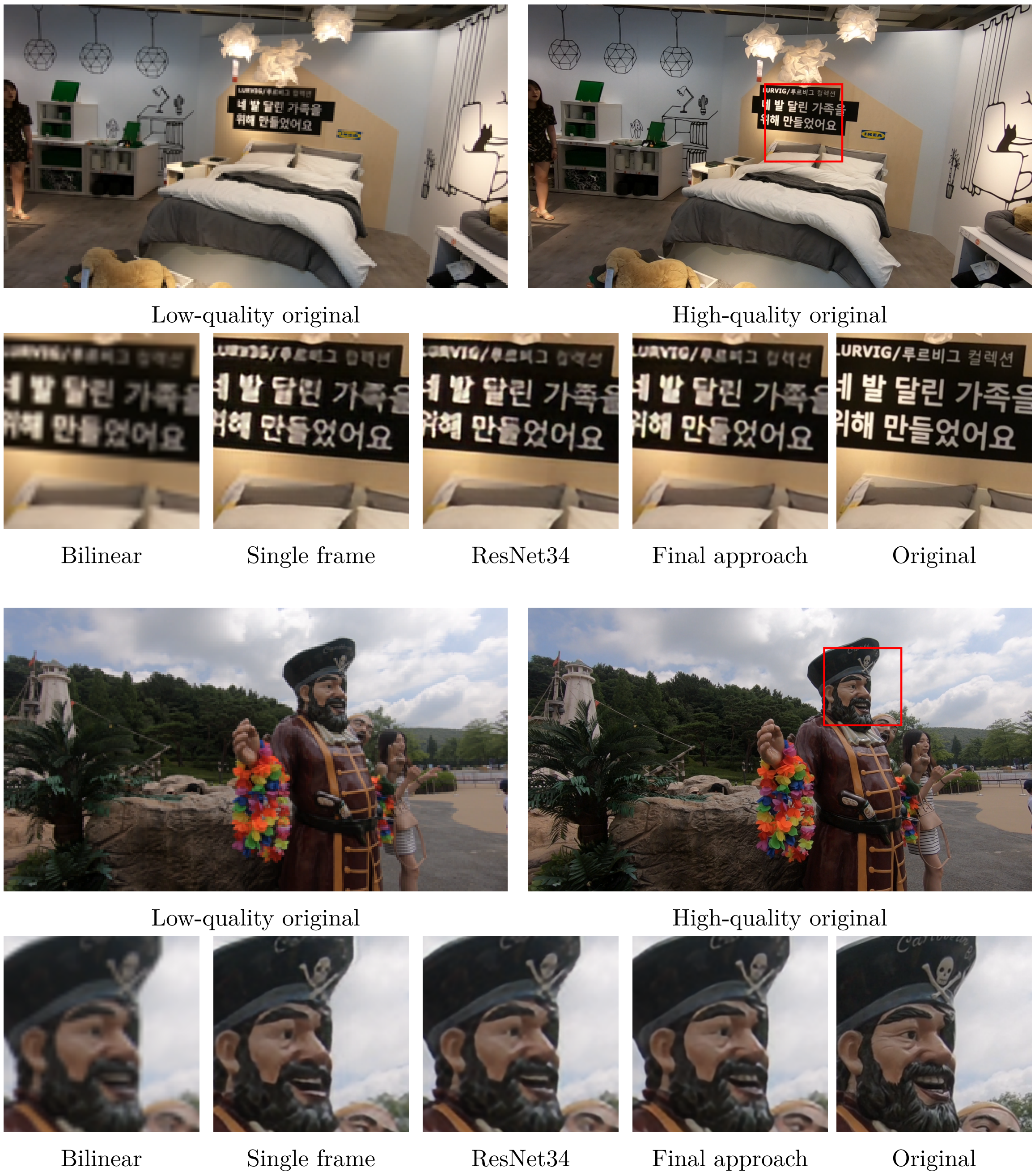


Figure 6.3: Visual results of different methods.

Chapter 7

Conclusion

The purpose of this work was to study existing methods of video super-resolution, conduct experiments and propose a novel module that can be used in different architectures, to improve the feature alignment between the frames, which is a common challenge in video super-resolution.

To complete this task, studying current methods was necessary to learn the problems in video super-resolution and their existing solutions. Then finding a suitable dataset was crucial, as the dataset is the foundation of the neural network training. REDS was chosen, as it contains videos from real-life scenarios, such as moving vehicles, people, etc. This dataset was then used to train the neural network and to evaluate the performance of the proposed module. Lastly, the neural network architecture for the experiments with the proposed module had to be chosen and implemented. U-Net was chosen, as it is a simple and effective architecture that is proven to work in various computer vision-related tasks. Then, multiple experiments were conducted, using different hyperparameter combinations, different configurations and different feature alignment module architectures.

As shown in Chapter 6, the proposed module showed promising results when plugged into U-Net architecture, with experiments showing that using Vgg19 [47] encoder gives slightly better results than using ResNet34 [17] encoder. Also as shown in the table 6.2 better results are seen when using cascading of lower-level features inside the feature alignment module. In table 6.3, a bigger improvement can be seen with the preprocessing of the raw optical flow by a single convolutional layer.

While the experiments showed results that significantly surpassed single-frame U-Net architecture and bilinear interpolation, multiple improvements could be still done. For example, experimenting with the feature fusion module could prove very beneficial, as using a single convolutional layer, as done in this work, might not be enough to fuse features from multiple frames properly. Implementing this with a temporal attention module could prove beneficial for the overall results. Experiments with different loss functions, such as learnable loss functions, like Adversarial or Perceptual loss. Another improvement could be made in the feature upscaling module, where changing it for a more complex architecture could show further improvements. Also, experiments with the proposed feature alignment module in different types of neural networks, such as GANs may show interesting results.

In the end, this work proposes a new module for feature alignment of multiple frames in a task of video super-resolution, with the implementation of easily configurable training and inference of different models.

Bibliography

[1] Agarap, A. F. *Deep Learning using Rectified Linear Units (ReLU)*. arXiv, 2018. DOI: 10.48550/ARXIV.1803.08375. Available at: https://arxiv.org/abs/1803.08375.

[2] AI Learner. *Image Processing: Bicubic Interpolation* [online]. 2018. Accessed: [2024-05-01]. Available at: https://theailearner.com/2018/12/29/image-processing-bicubic-interpolation/.

[3] AI Learner. *Image Processing: Bilinear Interpolation* [online]. 2018. Accessed: [2024-05-01]. Available at: https://theailearner.com/2018/12/29/image-processing-bilinear-interpolation/.

[4] AI Learner. *Image Processing: Nearest Neighbour Interpolation* [online]. 2018. Accessed: [2024-05-01]. Available at: https://theailearner.com/2018/12/29/image-processing-nearest-neighbour-interpolation/.

[5] Atidel, L., Larabi, C., Beghdadi, A., Viennet, E. and Bouridane, A. Knowledge-based Taxonomic Scheme for Full-Reference Objective Image Quality Measurement Models. *Electronic Imaging*. january 2017, vol. 2017, p. 64–78. DOI: 10.2352/ISSN.2470-1173.2017.12.IQSP-228.

[6] Bank, D., Koenigstein, N. and Giryes, R. *Autoencoders*. arXiv, 2020. DOI: 10.48550/ARXIV.2003.05991. Available at: https://arxiv.org/abs/2003.05991.

[7] Buades, A., Coll, B. and Morel, J.-M. Non-Local Means Denoising. *Image Processing On Line*. 2011, vol. 1, p. 208–212. https://doi.org/10.5201/ipol.2011.bcm_nlm.

[8] Cao, J., Li, Y., Zhang, K. and Van Gool, L. *Video Super-Resolution Transformer*. arXiv, 2021. DOI: 10.48550/ARXIV.2106.06847. Available at: https://arxiv.org/abs/2106.06847.

[9] Chatterjee, P. and Milanfar, P. Is Denoising Dead? *IEEE Transactions on Image Processing*. Institute of Electrical and Electronics Engineers (IEEE). april 2010, vol. 19, no. 4, p. 895–911. DOI: 10.1109/tip.2009.2037087. ISSN 1941-0042. Available at: http://dx.doi.org/10.1109/TIP.2009.2037087.

[10] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G. et al. *Deformable Convolutional Networks*. arXiv, 2017. DOI: 10.48550/ARXIV.1703.06211. Available at: https://arxiv.org/abs/1703.06211.

[11] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X. et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*.

arXiv, 2020. DOI: 10.48550/ARXIV.2010.11929. Available at: https://arxiv.org/abs/2010.11929.

[12] Elad, M., Kawar, B. and Vaksman, G. *Image Denoising: The Deep Learning Revolution and Beyond – A Survey Paper –*. arXiv, 2023. DOI: 10.48550/ARXIV.2301.03362. Available at: https://arxiv.org/abs/2301.03362.

[13] Feichtenhofer, C., Fassold, H. and Schallauer, P. A Perceptual Image Sharpness Metric Based on Local Edge Gradient Analysis. *IEEE Signal Processing Letters*. 2013, vol. 20, no. 4, p. 379–382. DOI: 10.1109/LSP.2013.2248711.

[14] georgeseif94. *Bilateral Filter* [online]. 7. May 2017. Accessed: 2024-05-06. Available at: https://ailearningcentre.wordpress.com/2017/05/07/bilateral-filter/.

[15] Goodfellow, I. J., Pouget Abadie, J., Mirza, M., Xu, B., Warde Farley, D. et al. *Generative Adversarial Networks*. arXiv, 2014. DOI: 10.48550/ARXIV.1406.2661. Available at: https://arxiv.org/abs/1406.2661.

[16] Hajian, A. and Aramvith, S. Fusion Objective Function on Progressive Super-Resolution Network. *Journal of Sensor and Actuator Networks*. march 2023, vol. 12, p. 26. DOI: 10.3390/jsan12020026.

[17] He, K., Zhang, X., Ren, S. and Sun, J. *Deep Residual Learning for Image Recognition*. arXiv, 2015. DOI: 10.48550/ARXIV.1512.03385. Available at: https://arxiv.org/abs/1512.03385.

[18] He, K., Zhang, X., Ren, S. and Sun, J. *Identity Mappings in Deep Residual Networks*. arXiv, 2016. DOI: 10.48550/ARXIV.1603.05027. Available at: https://arxiv.org/abs/1603.05027.

[19] Hitawala, S. *Comparative Study on Generative Adversarial Networks*. arXiv, 2018. DOI: 10.48550/ARXIV.1801.04271. Available at: https://arxiv.org/abs/1801.04271.

[20] Horn, B. K. and Schunck, B. G. Determining optical flow. *Artificial Intelligence*. 1981, vol. 17, no. 1, p. 185–203. DOI: https://doi.org/10.1016/0004-3702(81)90024-2. ISSN 0004-3702. Available at: https://www.sciencedirect.com/science/article/pii/0004370281900242.

[21] Horé, A. and Ziou, D. Image Quality Metrics: PSNR vs. SSIM. In: *2010 20th International Conference on Pattern Recognition*. 2010, p. 2366–2369. DOI: 10.1109/ICPR.2010.579.

[22] Ioffe, S. and Szegedy, C. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv, 2015. DOI: 10.48550/ARXIV.1502.03167. Available at: https://arxiv.org/abs/1502.03167.

[23] Johnson, J., Alahi, A. and Fei Fei, L. *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*. arXiv, 2016. DOI: 10.48550/ARXIV.1603.08155. Available at: https://arxiv.org/abs/1603.08155.

[24] Kappeler, A., Yoo, S., Dai, Q. and Katsaggelos, A. K. Video Super-Resolution With Convolutional Neural Networks. *IEEE Transactions on Computational Imaging*. 2016, vol. 2, no. 2, p. 109–122. DOI: 10.1109/TCI.2016.2532323.

[25] Kim, S. Y., Lim, J., Na, T. and Kim, M. *3DSRnet: Video Super-resolution using 3D Convolutional Neural Networks*. arXiv, 2018. DOI: 10.48550/ARXIV.1812.09079. Available at: https://arxiv.org/abs/1812.09079.

[26] Kingma, D. P. and Ba, J. *Adam: A Method for Stochastic Optimization*. arXiv, 2014. DOI: 10.48550/ARXIV.1412.6980. Available at: https://arxiv.org/abs/1412.6980.

[27] Kramer, M. A. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*. 1991, vol. 37, no. 2, p. 233–243. DOI: https://doi.org/10.1002/aic.690370209. Available at: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209.

[28] Lee, D.-H. Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*. july 2013.

[29] Lepcha, D. C., Goyal, B., Dogra, A. and Goyal, V. Image super-resolution: A comprehensive review, recent trends, challenges and applications. *Information Fusion*. 2023, vol. 91, p. 230–260. DOI: https://doi.org/10.1016/j.inffus.2022.10.007. ISSN 1566-2535. Available at: https://www.sciencedirect.com/science/article/pii/S1566253522001762.

[30] Liang, J., Cao, J., Fan, Y., Zhang, K., Ranjan, R. et al. *VRT: A Video Restoration Transformer*. arXiv, 2022. DOI: 10.48550/ARXIV.2201.12288. Available at: https://arxiv.org/abs/2201.12288.

[31] Liang, J., Cao, J., Sun, G., Zhang, K., Van Gool, L. et al. *SwinIR: Image Restoration Using Swin Transformer*. arXiv, 2021. DOI: 10.48550/ARXIV.2108.10257. Available at: https://arxiv.org/abs/2108.10257.

[32] Liang, J., Fan, Y., Xiang, X., Ranjan, R., Ilg, E. et al. *Recurrent Video Restoration Transformer with Guided Deformable Attention*. arXiv, 2022. DOI: 10.48550/ARXIV.2206.02146. Available at: https://arxiv.org/abs/2206.02146.

[33] Liao, Y., Xie, J. and Geiger, A. *KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D*. arXiv, 2021. DOI: 10.48550/ARXIV.2109.13410. Available at: https://arxiv.org/abs/2109.13410.

[34] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y. et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. arXiv, 2021. DOI: 10.48550/ARXIV.2103.14030. Available at: https://arxiv.org/abs/2103.14030.

[35] Lucas, B. D. and Kanade, T. An iterative image registration technique with an application to stereo vision. In: *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, p. 674–679. IJCAI'81. Available at: http://dl.acm.org/citation.cfm?id=1623264.1623280.

[36] Mittal, A., Soundararajan, R. and Bovik, A. C. Making a "Completely Blind" Image Quality Analyzer. *IEEE Signal Processing Letters*. Institute of Electrical and Electronics Engineers (IEEE). march 2013, vol. 20, no. 3, p. 209–212. DOI:

10.1109/lsp.2012.2227726. ISSN 1558-2361. Available at: http://dx.doi.org/10.1109/LSP.2012.2227726.

[37] Mittal, A., Moorthy, A. K. and Bovik, A. C. Blind/Referenceless Image Spatial Quality Evaluator. In: *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*. 2011, p. 723–727. DOI: 10.1109/ACSSC.2011.6190099.

[38] Nah, S., Baik, S., Hong, S., Moon, G., Son, S. et al. NTIRE 2019 Challenge on Video Deblurring and Super-Resolution: Dataset and Study. In: *CVPR Workshops*. June 2019.

[39] Nguyen, N. L., Anger, J., Davy, A., Arias, P. and Facciolo, G. *Self-Supervised Super-Resolution for Multi-Exposure Push-Frame Satellites*. arXiv, 2022. DOI: 10.48550/ARXIV.2205.02031. Available at: https://arxiv.org/abs/2205.02031.

[40] OpenMMLab. *Awesome Datasets for Super-Resolution: Introduction and Pre-processing* [online]. 2021. Accessed: [2024-05-01]. Available at: https://openmmlab.medium.com/awesome-datasets-for-super-resolution-introduction-and-pre-processing-55f8501f8b18.

[41] Pahalawatta, P. V. and Tourapis, A. M. Motion estimated temporal consistency metrics for objective video quality assessment. In: *2009 International Workshop on Quality of Multimedia Experience*. 2009, p. 174–179. DOI: 10.1109/QOMEX.2009.5246955.

[42] Pandey, T., Pena, D., Byrne, J. and Moloney, D. Leveraging Deep Learning for Visual Odometry Using Optical Flow. *Sensors*. 2021, vol. 21, no. 4. DOI: 10.3390/s21041313. ISSN 1424-8220. Available at: https://www.mdpi.com/1424-8220/21/4/1313.

[43] Parsania, P. and Virparia, P. A Comparative Analysis of Image Interpolation Algorithms. *IJARCCE*. january 2016, vol. 5, p. 29–34. DOI: 10.17148/IJARCCE.2016.5107.

[44] Ronneberger, O., Fischer, P. and Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. arXiv, 2015. DOI: 10.48550/ARXIV.1505.04597. Available at: https://arxiv.org/abs/1505.04597.

[45] Shi, S., Gu, J., Xie, L., Wang, X., Yang, Y. et al. *Rethinking Alignment in Video Super-Resolution Transformers*. arXiv, 2022. DOI: 10.48550/ARXIV.2207.08494. Available at: https://arxiv.org/abs/2207.08494.

[46] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P. et al. *Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network*. arXiv, 2016. DOI: 10.48550/ARXIV.1609.05158. Available at: https://arxiv.org/abs/1609.05158.

[47] Simonyan, K. and Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv, 2014. DOI: 10.48550/ARXIV.1409.1556. Available at: https://arxiv.org/abs/1409.1556.

[48] Su, J., Xu, B. and Yin, H. A Survey of Deep Learning Approaches to Image Restoration. *Neurocomputing*. february 2022, vol. 487. DOI: 10.1016/j.neucom.2022.02.046.

[49] Tarvainen, A. and Valpola, H. *Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results.* arXiv, 2017. DOI: 10.48550/ARXIV.1703.01780. Available at: https://arxiv.org/abs/1703.01780.

[50] Teed, Z. and Deng, J. *RAFT: Recurrent All-Pairs Field Transforms for Optical Flow.* arXiv, 2020. DOI: 10.48550/ARXIV.2003.12039. Available at: https://arxiv.org/abs/2003.12039.

[51] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. *Attention Is All You Need.* arXiv, 2017. DOI: 10.48550/ARXIV.1706.03762. Available at: https://arxiv.org/abs/1706.03762.

[52] Wang, X., Chan, K. C. K., Yu, K., Dong, C. and Loy, C. C. *EDVR: Video Restoration with Enhanced Deformable Convolutional Networks.* arXiv, 2019. DOI: 10.48550/ARXIV.1905.02716. Available at: https://arxiv.org/abs/1905.02716.

[53] Wang, Z., Chen, J. and Hoi, S. C. H. *Deep Learning for Image Super-resolution: A Survey.* arXiv, 2019. DOI: 10.48550/ARXIV.1902.06068. Available at: https://arxiv.org/abs/1902.06068.

[54] Wang, Z., Chen, J. and Hoi, S. C. H. *Deep Learning for Image Super-resolution: A Survey.* arXiv, 2019. DOI: 10.48550/ARXIV.1902.06068. Available at: https://arxiv.org/abs/1902.06068.

[55] Xue, T., Chen, B., Wu, J., Wei, D. and Freeman, W. T. Video Enhancement with Task-Oriented Flow. *International Journal of Computer Vision (IJCV).* Springer. 2019, vol. 127, no. 8, p. 1106–1125.

[56] Ye, C., Evanusa, M., He, H., Mitrokhin, A., Goldstein, T. et al. *Network Deconvolution.* arXiv, 2019. DOI: 10.48550/ARXIV.1905.11926. Available at: https://arxiv.org/abs/1905.11926.

[57] Ying, X., Wang, L., Wang, Y., Sheng, W., An, W. et al. Deformable 3D Convolution for Video Super-Resolution. *IEEE Signal Processing Letters.* Institute of Electrical and Electronics Engineers (IEEE). 2020, vol. 27, p. 1500–1504. DOI: 10.1109/lsp.2020.3013518. ISSN 1558-2361. Available at: http://dx.doi.org/10.1109/LSP.2020.3013518.

[58] Zhang, K., Zuo, W., Chen, Y., Meng, D. and Zhang, L. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing.* Institute of Electrical and Electronics Engineers (IEEE). july 2017, vol. 26, no. 7, p. 3142–3155. DOI: 10.1109/tip.2017.2662206. ISSN 1941-0042. Available at: http://dx.doi.org/10.1109/TIP.2017.2662206.

[59] Zhu, J.-Y., Park, T., Isola, P. and Efros, A. A. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.* arXiv, 2017. DOI: 10.48550/ARXIV.1703.10593. Available at: https://arxiv.org/abs/1703.10593.

Appendix A

Structure of the Attached Medium

src/	Folder with source files for training and network architectures.
scripts/	Folder with scripts for evaluation, data preparation and adapted RAFT source code.
Weights/	Folder with pre-trained weights.
tex/	Folder with LaTeX source files.
data/	Folder with sample data for training and evaluation.
configs/	Folder with example config files for training.
README.md	README file with instructions.
requirements.txt	Python libraries requirements.
poster.pdf	Poster file.
video.mp4	Short video with the results.
thesis.pdf	Thesis.
thesis-print.pdf	Thesis for print.