Optimization Techniques of Deep Learning Models for Visual Quality Improvement

Lorenzo PALLONI [a], Leonardo GALTERI [b,1] and Marco BERTINI [a]
[a] *University of Florence*
[b] *Pegaso Telematic University*

Abstract. Video restoration is a widely studied task in the field of computer vision and image processing. The primary objective of video restoration is to improve the visual quality of degraded videos caused by various factors, such as noise, blur, compression artifacts, and other distortions. In this study, the integration of post-training quantization techniques was investigated to optimize deep learning models for super-resolution inference. The results indicate that reducing the precision of weights and activations in these models substantially decreases the computational complexity and memory requirements without compromising performance, rendering them more practical and cost-effective for real-world applications, where real-time inference is often required. When TensorRT was integrated with PyTorch, the efficiency of the model was further improved taking advantage of the INT8 computational capabilities of recent NVIDIA GPUs.

Keywords. video restoration, deep learning, neural networks, network quantization

1. Introduction

Video restoration is a widely studied task in the field of computer vision and image processing. The primary objective of video restoration is to improve the visual quality of degraded videos caused by various factors, such as noise, blur, compression artifacts, and other distortions. In recent years, deep learning techniques have gained significant attention in the field of video restoration due to their superior performance compared to traditional methods.

Traditional video restoration techniques involve the use of mathematical algorithms to reduce distortions [1]. Some of the commonly used techniques include median filtering, bilateral filtering, and wavelet-based denoising. Although these methods are effective in restoring some aspects of video quality, they have limited performance in handling complex distortions and restoring high-frequency details.

Deep learning-based video restoration methods have been shown to achieve state-of-the-art performance in the restoration of degraded videos. Using deep neural networks to learn the underlying mapping between degraded and ground-truth images, these methods allow the restoration of high-frequency details and complex video distortions.

[1] Corresponding Author: Leonardo Galteri, leonardo.galteri@unipegaso.it.

Super-resolution (SR), the process of increasing the spatial resolution of an image or video, is a critical task in video restoration. Deep learning-based super-resolution techniques typically use convolutional neural networks (CNNs) to learn the underlying mapping between low-resolution and high-resolution images.

The most commonly used approaches are single image super-resolution (SISR) and multiple-image super-resolution (MISR). In SISR, a CNN is trained to generate a high-resolution image from a single low-resolution image by minimizing a loss function that measures the difference between a generated image and its ground-truth counterpart. In MISR, a CNN is trained to generate a high-resolution image from multiple low-resolution images.

Deep learning models for video restoration, including super-resolution and artifact removal, have made significant progress in recent years. However, the computational complexity of these models remains a major hurdle for their deployment in real-world applications. One promising strategy for reducing the computational demands of deep learning models is the use of quantization techniques. Quantization refers to representing the weights and activations of a model with fewer bits, with the aim of minimizing the loss of accuracy. This approach can lead to substantial improvements in inference speed and memory usage, making it especially attractive for implementing video restoration models on resource-constrained devices.

In this paper we focus on examining the efficacy of quantization techniques to enhance inference speed and reduce memory usage of deep learning models applied to video restoration tasks, including artifact removal and SISR. Research encompasses an extensive review of the literature on quantization techniques for deep learning models and their application in the field of video restoration. In particular, we investigate the implementation and evaluation of post-training quantization using TensorRT [2], an NVIDIA SDK designed for high performance deep learning inference.

2. Related Works

This section reviews current literature on methods for addressing video restoration problems, and recent deep learning models for single image super-resolution. Furthermore, we review the state-of-the-art approaches for network quantization.

2.1. Video Restoration

Dong et al. [3] present the SRCNN model using convolutional neural networks for single image super-resolution, while Kim et al. introduce VDSR [4] and DRCN [5] models with a residual learning framework [6]. Dong et al. [7] and Shi et al. [8] efficiently map low-resolution to high-resolution images using deconvolution and sub-pixel convolution layers, respectively. LatticeNet [9], LapSRN [10], MemNet [11], SRDenseNet [12], and RDN [13] are other lightweight and efficient SR models. Haris et al. [14] and DSRN [15] focus on iterative upsampling and downsampling and dual-state recurrent networks, respectively. MSRN [16] and RFA [17] exploit image features efficiently using different blocks, while recent attention mechanisms have also been used to improve super-resolution quality [18,19,20,21].

In recent years, SR models based on a generative adversarial network (GAN) framework have also gained popularity. SRGAN [22] is a notable example in which the gener-

ator network is trained to produce high-resolution images that are perceptually similar to ground truth images, while the discriminator network is trained to distinguish between generated and ground truth images. Since then, several GAN-based SR models have been proposed, including EnhancedNet [23], ESRGAN [24], SPSR [25], SRFBN [26], and SRFlow [27].

In [28], Galteri et al. use adversarial training to remove artifacts from lossy compression algorithms for images and videos, training models with larger batches and using an ensemble of networks. They also address real-time artifact removal using MobileNetV2-inspired generators with depth-wise separable convolutions [29,30]. Mameli et al. [31] apply the no-GAN approach for compression artifact removal and super-resolution, using pre-training and a few iterations of the adversarial training. Kaneko et al. introduce BNCR-GAN [32], an architecture that removes noise, compression artifacts, and blurring artifacts by combining the three models and using adaptive consistency losses.

Pourreza et al. [33] propose a GAN-based quality enhancement method to address the accuracy loss in action recognition tasks caused by video compression, using a frame-recurrent strategy. Yu et al. developed VR-GAN [34] for HEVC compression-generated artifact removal, operating at the inter-frame level and using flow estimation for coherence. In [35], He et al. improve HEVC compressed videos using adversarial loss combined with L1 loss, employing a residual network generator.

In [36], Galteri et al. present a full pipeline architecture featuring semantic deep encoding and decoding, with a semantic mask for each frame to allocate more bits to semantically interesting content.

More recently, several different approaches based on visual transformers, such as [37], have been proposed, which have shown excellent performance in the reconstruction task, but are more time consuming than the other state-of-the-art methods.

2.2. Quantizazion

Quantization is a technique that reduces computational complexity and memory usage of deep learning models while maintaining their accuracy and performance. This is important for deploying these models on devices with limited resources. The technique represents model parameters and activations using fewer bits than the standard 32-bit floating-point format.

There are different methods for quantizing neural networks, such as uniform quantization and non-uniform quantization. Uniform quantization divides weights and activations into equally spaced levels, while non-uniform quantization assigns more bits to important data and fewer bits to less important data. However, implementing non-uniform quantization on common computational hardware can be challenging, so uniform quantization is the current standard method due to its simplicity and efficient mapping to hardware. [38].

Another way in which quantization methods can be divided is in fixed-point, integer, and hybrid quantization. Fixed-point quantization represents numbers with a fixed number of bits, typically 8 or 16, which reduces the memory footprint of the model. Integer quantization represents weights and activations as integers instead of floating-point values, which reduces the model's memory footprint even further. Hybrid quantization combines fixed-point and floating-point representations for different layers of the model to achieve a balance between accuracy and memory requirements.

It is often necessary to fine-tune the parameters in a neural network after quantization. There are two ways to accomplish this: one approach involves retraining the model, which is known as Quantization-Aware Training (QAT), while the other approach is to modify the parameters without retraining, a method commonly referred to as Post-Training Quantization (PTQ).

The technique called Quantization Aware Training (QAT) retrains a neural network with quantized parameters to minimize the errors introduced by quantization. During QAT, forward and backward passes are executed in floating point, and the model parameters are quantized after each gradient update using projected gradient descent. Performing quantization after weight update in floating-point precision is crucial for accuracy. Using backpropagation in floating point avoids errors due to zero-gradient or gradients with high errors when accumulating gradients in quantized precision. To address the non-differentiable quantization operator issue, the Straight Through Estimator (STE) is used, which approximates the quantization operator well, except for ultra-low-precision quantization like binary quantization.

Although QAT is effective, its main drawback is the computational cost of retraining the neural network model, which may take several hundred epochs to recover accuracy, particularly in low-bit precision quantization scenarios. Post-Training Quantization (PTQ) is an alternative approach that establishes the quantization parameters for both weights and activations without performing any retraining on the neural network model. Consequently, PTQ is a rapid technique for quantizing neural network models, but often results in lower accuracy compared to QAT. The investment in retraining may be worthwhile for models expected to be deployed for a long time and where both efficiency and accuracy are critical, but this may not be the case for models with a short lifetime.

3. Methodology

In this section, we provide a detailed overview of the UNet and SRUNet architectures used in our experiments. Additionally, the training setup used is outlined, which consists of a Generative Adversarial Network (GAN) framework that incorporates LPIPS and SSIM as generator loss. Finally, we show the methods we use to quantize the network.

3.1. Architectures

The UNet architecture was introduced by Ronneberger et al. [39] in 2015 for biomedical image segmentation tasks.

Let x be a low resolution input image with spatial dimensions of $W \times H \times C$, where W and H represent the width and height of the image and C is the number of channels. The objective of a super-resolution approach is to generate a high resolution output image y with spatial dimensions of $W' \times H' \times C$, where W' and H' are larger than W and H, respectively.

The UNet model is composed by an encoder and a decoder, with skip connections between them. The encoder takes the image x as input and applies a series of convolutional layers to reduce the dimensionality of the image and capture important features. The encoder can be represented as a function f_θ that takes the image x as input and returns a feature map $\mathbf{f}$:

$$\mathbf{f} = f_\theta(x)$$

On the other hand, the decoder takes the feature map $\mathbf{f}$ as input and applies a series of convolutional layers to upscale the feature maps and generate the output image while preserving the details. The decoder can be represented as a function g_ϕ that takes the last encoder feature map $\mathbf{f}$ and returns the generated high resolution output image $\hat{y}$:

$$\hat{y} = g_\phi(\mathbf{f})$$

Skip connections are used to connect the corresponding encoder and decoder layers. Specifically, the feature maps of the encoder are concatenated with the feature maps of the corresponding decoder layers to help the model capture fine-grained details and ensure that the output image is more accurate.

SRUNet is an adaptation of the UNet architecture for super-resolution and compression artifact removal, proposed in [40] by Vaccaro et al. The main two modifications are the decrease in the number of filters in each convolutional layer and the use of a residual layer as the final layer.

The final residual layer computes the difference between the input image x upscaled with a linear interpolation function and the output of the second to the last layer upscaled with the pixel shuffle function. In this way, the model is set to learn the difference between a low-resolution image and its high-resolution version.

Pixel-shuffle (also known as sub-pixel convolutional layer) is the fastest upsample layer available: it comprises a depth-compression of the output tensor into 12-channels via convolution operation, and then these features are reshuffled into an RGB image at double resolution. Alternatives, such as bilinear upsample with convolution, transposed convolution, or even reshuffling to a higher dimension with the same depth, may add an additional overhead since they work in the high-resolution space.

Modeling the problem as producing a residual at the top of the upsampled image is particularly convenient. This forces the model to focus on the high-frequency patterns, sharpening edges or increasing texture details, since the low-frequency patterns are still in the upsampled image. In addition, a faster convergence of the training process is typically observed.

3.2. Adversarial Training

To train both the UNet and SRUNet models, we employed a Generative Adversarial Network (GAN) framework. Introduced in [41], the GAN framework consists of two models, a generator and a discriminator.

In the context of super-resolution, the generator network takes a low-resolution image as input and generates a high-resolution image, while the discriminator tries to distinguish between the generated high-resolution image and the ground truth high-resolution image.

The two networks are trained in an adversarial manner: the generator tries to fool the discriminator by generating high-resolution images as similar as possible to the ground-truth images, while the discriminator tries to correctly classify the generated images as

true or generated. This competition between the two networks leads to the generator improving over time and producing images of higher quality.

The generator loss is composed by two parts: an adversarial loss and a content loss. The generator is encouraged to generate high-resolution images that belong as much as possible to the distribution of the ground-truth images by adversarial loss and to generate high-resolution images that are as close as possible to the actual ground-truth images by content loss.

In this setup, LPIPS and SSIM were used as content loss. LPIPS is a perceptual similarity metric that measures the distance between two images in terms of their perceptual features. SSIM is a structural similarity metric that measures the similarity between two images in terms of their structure, luminance, and contrast.

Generator loss can be represented as follows:

$$\mathscr{L}_G = -\log(D(\hat{y})) + w_{LPIPS} \cdot LPIPS(\hat{y}, y) + w_{SSIM} \cdot (1 - SSIM(\hat{y}, y)) \tag{1}$$

where $\hat{y}$ is the generated high-resolution image, y is the ground truth, $D(\hat{y})$ is the probability score of the discriminator for the generated image, w_{LPIPS} and w_{SSIM} are hyperparameters that control the relative importance of the LPIPS and SSIM losses.

The design chosen for the discriminator network is the same as that used in [22], where the authors followed the architectural guidelines described in [42] and incorporated LeakyReLU [43] activation with α set to 0.2 while avoiding max-pooling in the network.

3.3. Network quantization

TensorRT is an NVIDIA inference optimization tool that can accelerate deep learning models on NVIDIA GPUs. In recent years, there has been increasing interest in integrating TensorRT with PyTorch [44], one of the most popular deep learning frameworks that allows users to easily develop and train deep learning models, to take advantage of the performance benefits of TensorRT during inference.

There are several ways to integrate TensorRT with PyTorch. One approach is to use the ONNX [45] format, which is an open standard for representing deep learning models. PyTorch models can be converted to the ONNX format using the `torch.onnx.export` function and the resulting ONNX file can then be optimized for inference using TensorRT. The optimized model can be loaded back into PyTorch using the `torch.onnx.import` function, allowing users to continue working with the model in PyTorch.

Another approach is to use Torch-TensorRT [46]. Torch-TensorRT is a compiler that is designed for PyTorch, TorchScript, and FX, with the aim of optimizing them for NVIDIA GPUs through the use of NVIDIA's TensorRT.

TorchScript is a language and runtime environment for PyTorch that allows users to save and execute PyTorch models in production without the full PyTorch framework. FX is an experimental toolkit for building high-performance machine learning models in PyTorch, enabling easy experimentation and deployment of models on various hardware platforms using pure Python syntax.

Unlike PyTorch's Just-In-Time (JIT) compiler, Torch-TensorRT is an Ahead-of-Time (AOT) compiler. This means that before deploying your TorchScript code, it is required to go through an explicit compile step to convert a standard TorchScript or FX

program into a module targeting a TensorRT engine. Torch-TensorRT works as a PyTorch extension, and the compiled modules can seamlessly integrate into the JIT runtime. Once compiled, the optimized graph should feel no different from running a TorchScript module. Additionally, access to TensorRT's suite of configurations at compile time is provided, allowing one to specify the operating precision (FP32/FP16/INT8) and other settings for the module. The Torch-TensorRT Python API offers a simple and user-friendly approach to using PyTorch data loaders in conjunction with TensorRT calibrators. By specifying the desired configuration, the DataLoaderCalibrator class can be employed to establish a TensorRT calibrator.

In the experiments we conducted throughout this work, Post-Training Quantization was utilized as the chosen quantization method, employing the Torch-TensorRT framework. This approach allowed for the quantization of UNet and SRUNet without retraining, resulting in a more efficient and expedient process.

4. Experiments

In this section, various experiments and their corresponding results will be showcased, highlighting the evaluation of inference speed and image quality. Basic PyTorch implementations of UNet and SRUNet will be used, as well as their compiled counterparts with TensorRT, allowing for different precisions of weights/activations, namely, FP32, FP16, and INT8.

4.1. Training Setup

During the model training process, the Adam optimizer was used with a learning rate of 1×10^{-4}. The model was trained for 191,268 iterations, each batch consisting of 14 image patches randomly sampled from the training set. A patch size of 96×96 was used, with one random crop per image. The only data augmentation strategy applied was horizontal reflection, avoiding warping or rotation transforms to maintain consistency with real H.265 encoded frames.

Model training required approximately 72 hours on a single NVIDIA Titan Xp. The proposed model was trained using the BVI-DVC dataset [47], designed for deep video compression tasks. This dataset includes 200 frame sequences truncated at the 64^{th} frame, with frame rates ranging from 25 to 120. The sequences feature various types of content, such as natural scenes, man-made objects, and cityscapes.

Although the original resolution was 2160p, the authors of the dataset created downscaled versions at 1080p, 540p, and 270p, generating 800 sequences and a total of 51,200 frames. The original BVI-DVC dataset served as the ground truth for the training dataset. The input data were generated by compressing each sequence using the H.265 codec with a Constant Rate Factor (CRF) of 23 while reducing the resolution to one-quarter of the original size. Fixing the CRF aims to mitigate the mode collapse issue described in [48]. However, it should be noted that a fixed CRF does not guarantee uniform frame quality; for example, frames with motion might exhibit lower quality than stationary ones.

Training the models on compressed videos, rather than only the downscaled version of high-quality videos, was crucial for applying super-resolution to compressed videos and performing both artifact reduction and super-resolution. Training the model solely

Model	LPIPS ↓	DISTS ↓	BRISQUE ↓	SSIM ↑	MS-SSIM ↑	PSNR ↑
UNet	**0.2897**	**0.1222**	31.1372	**0.8952**	**0.8517**	**21.6506**
UNet-FP32	**0.2897**	**0.1222**	31.1373	**0.8952**	**0.8517**	**21.6506**
UNet-FP16	0.2898	0.1223	31.1383	**0.8952**	**0.8517**	**21.6506**
UNet-INT8	0.3041	0.1283	**29.6849**	0.8941	0.8508	21.6388
SRUNet	0.3111	**0.1717**	27.3738	**0.8894**	**0.8457**	21.3670
SRUNet-FP32	0.3111	**0.1717**	27.3736	**0.8894**	**0.8457**	21.3670
SRUNet-FP16	0.3111	**0.1717**	27.3790	**0.8894**	**0.8457**	**21.3671**
SRUNet-INT8	**0.3068**	0.1722	**26.1546**	0.8882	0.8442	21.3320

Table 1. Evaluation of perception and traditional metrics averaged on 60 test frames.

for super-resolution could result in the model failing to detect features to super-resolve or even enlarging compression artifacts and reducing the overall quality.

4.2. Quantitative Results

The perceptual metrics used for evaluation are LPIPS, DISTS, and BRISQUE. Lower values of these metrics indicate better image quality. Table 1 shows that the UNet model and its optimized versions perform similarly according to the LPIPS and DISTS metrics, with the INT8 version showing a slight decrease in these metrics. Both UNet and SRUNet models slightly improve in terms of the BRISQUE score when optimized with INT8.

SSIM, MS-SSIM, and PSNR are the non-perceptual metrics used for evaluation. For these metrics, higher values indicate better image quality. In Table 1, both the UNet and SRUNet models show similar performance across all precisions in SSIM and MS-SSIM. The UNet model has a slightly higher PSNR value than the SRUNet model, and the INT8-optimized versions of both models have marginally lower PSNR values compared to their FP32 and FP16 counterparts.

Table 2 shows the VMAF scores for both the UNet and SRUNet models and their optimized versions using a 2-minute test video. The harmonic mean is also shown because if there were outliers (i.e., frames for which the score is lower than the average by more than a certain number of standard deviations), the score would decrease substantially more than the traditional average.

In general, the differences in performance between the plain models and their optimized versions are relatively small for both perceptual and traditional metrics.

The execution times for both UNet and SRUNet, together with their TensorRT-optimized versions (FP32, FP16, INT8), are shown in Table 3. Both the UNet and SRUNet models demonstrate reduced inference times for all precisions. The INT8-optimized version exhibits the fastest evaluation times for both models as expected, followed by FP16 and FP32 versions. Moreover, the SRUNet model consistently outperforms the UNet model for all precisions, and the relatively small standard deviations indicate consistent performance over the 300 runs.

The INT8-optimized UNet and SRUNet models achieve a 2.38X and 2.26X speed-up compared to their original implementations, respectively. Furthermore, memory consumption was reduced to 63.3% for the UNet (from 30MB to 11MB) and up to 53.8% (from 2.6MB to 1.2MB) for the SRUNet using the INT8 version.

Model	VMAF (mean) ↑	VMAF (harmonic mean) ↑
UNet	47.71	47.20
UNet-FP32	**47.72**	**47.21**
UNet-FP16	47.71	47.20
UNet-INT8	47.47	46.96
SRUNet	**47.20**	**46.65**
SRUNet-FP32	47.19	46.64
SRUNet-FP16	47.19	**46.65**
SRUNet-INT8	47.18	46.64

Table 2. VMAF scores on a 120 second test video.

Model	times [s] ↓	speedup ↑
UNet	0.0348	-
UNet-FP32	0.0279	1.247X
UNet-FP16	0.02794	1.247X
UNet-INT8	**0.0146**	**2.38X**
SRUNet	0.0123	-
SRUNet-FP32	0.0087	1.41X
SRUNet-FP16	0.0087	1.41X
SRUNet-INT8	**0.0054**	**2.27X**

Table 3. Evaluation time averaged on 300 runs.

4.3. Qualitative Results

This section aims to showcase the consistent performance of non-optimized and TensorRT-optimized UNet and SRUNet models in restoring distorted frames. The evaluations are shown in Figure 1, using three 384×384 patches taken from the first frame of three distinct 1920×1080 test videos, each representing a different environment. Each patch is compressed using H.265 and its resolution is reduced by a scaling factor of 4 resulting in a 96×96 patch. Each of the aforementioned figures is organized into three columns: ground truth (left), restored image using a super-resolution model (middle), and resized image using bicubic interpolation (right). The rows in these figures represent model variations, with only the middle column varying across different models.

5. Conclusions

In this paper, we investigate the integration of post-training quantization techniques to optimize deep learning models for super-resolution inference. The results indicate that reducing the precision of weights and activations in both the UNet and SRUNet models substantially decreases the computational complexity (up to 2.38X speedup) and memory requirements (up to 63.3% of size reduction) without compromising performance, rendering them more practical and cost-effective for real-world applications, where real-time inference is often required.

The use of TensorRT also played a key role in achieving these results. When TensorRT was integrated with PyTorch, the efficiency of the model was further improved taking advantage of the INT8 computational capabilities of recent NVIDIA GPUs.

Figure 1. Qualitative results evaluating SRUNet, SRUNet-FP32, SRUNet-FP16, and SRUNet-INT8 models on a compressed and down-scaled 96×96 patch of a test video frame producing a 384×384 patch displayed in the middle column, along with the ground truth (left column), and the resized version using bicubic interpolation (right column).

The results of this study show promising results and there are still areas for future research. Other optimization techniques such as pruning, weight sharing, or model distillation could be explored to further reduce the computational complexity and memory requirements of deep learning models. In addition, the impact of different quantization parameters and techniques could be investigated in more detail.

References

[1] Fan L, Zhang F, Fan H, Zhang C. Brief review of image denoising techniques. Visual Computing for Industry, Biomedicine, and Art. 2019;2(1):1-12.

[2] Corporation N. NVIDIA TensorRT; 2016. Software available from `https://developer.nvidia.com/tensorrt`.

[3] Dong C, Loy CC, He K, Tang X. Learning a deep convolutional network for image super-resolution. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13. Springer; 2014. p. 184-99.

[4] Kim J, Lee JK, Lee KM. Accurate image super-resolution using very deep convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 1646-54.

[5] Kim J, Lee JK, Lee KM. Deeply-recursive convolutional network for image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 1637-45.
[6] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 770-8.
[7] Dong C, Loy CC, Tang X. Accelerating the super-resolution convolutional neural network. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14. Springer; 2016. p. 391-407.
[8] Shi W, Caballero J, Huszár F, Totz J, Aitken AP, Bishop R, et al. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 1874-83.
[9] Luo X, Xie Y, Zhang Y, Qu Y, Li C, Fu Y. Latticenet: Towards lightweight image super-resolution with lattice block. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16. Springer; 2020. p. 272-89.
[10] Lai WS, Huang JB, Ahuja N, Yang MH. Deep laplacian pyramid networks for fast and accurate super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 624-32.
[11] Tai Y, Yang J, Liu X, Xu C. Memnet: A persistent memory network for image restoration. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 4539-47.
[12] Tong T, Li G, Liu X, Gao Q. Image super-resolution using dense skip connections. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 4799-807.
[13] Zhang Y, Tian Y, Kong Y, Zhong B, Fu Y. Residual dense network for image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2018. p. 2472-81.
[14] Haris M, Shakhnarovich G, Ukita N. Deep back-projection networks for super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2018. p. 1664-73.
[15] Han W, Chang S, Liu D, Yu M, Witbrock M, Huang TS. Image super-resolution via dual-state recurrent networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2018. p. 1654-63.
[16] Li J, Fang F, Mei K, Zhang G. Multi-scale residual network for image super-resolution. In: Proceedings of the European conference on computer vision (ECCV); 2018. p. 517-32.
[17] Liu J, Zhang W, Tang Y, Tang J, Wu G. Residual feature aggregation network for image super-resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2020. p. 2359-68.
[18] Dai T, Cai J, Zhang Y, Xia ST, Zhang L. Second-order attention network for single image super-resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2019. p. 11065-74.
[19] Mei Y, Fan Y, Zhou Y, Huang L, Huang TS, Shi H. Image super-resolution with cross-scale non-local attention and exhaustive self-exemplars mining. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2020. p. 5690-9.
[20] Niu B, Wen W, Ren W, Zhang X, Yang L, Wang S, et al. Single image super-resolution via a holistic attention network. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16. Springer; 2020. p. 191-207.
[21] Zhang Y, Li K, Li K, Wang L, Zhong B, Fu Y. Image super-resolution using very deep residual channel attention networks. In: Proceedings of the European conference on computer vision (ECCV); 2018. p. 286-301.
[22] Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, et al. Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 4681-90.
[23] Sajjadi MS, Scholkopf B, Hirsch M. Enhancenet: Single image super-resolution through automated texture synthesis. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 4491-500.
[24] Wang X, Yu K, Wu S, Gu J, Liu Y, Dong C, et al. Esrgan: Enhanced super-resolution generative adversarial networks. In: Proceedings of the European conference on computer vision (ECCV) workshops; 2018. p. 0-0.
[25] Ma C, Rao Y, Cheng Y, Chen C, Lu J, Zhou J. Structure-preserving super resolution with gradient guidance. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2020. p. 7769-78.

[26] Li Z, Yang J, Liu Z, Yang X, Jeon G, Wu W. Feedback network for image super-resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2019. p. 3867-76.
[27] Lugmayr A, Danelljan M, Van Gool L, Timofte R. Srflow: Learning the super-resolution space with normalizing flow. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16. Springer; 2020. p. 715-32.
[28] Galteri L, Seidenari L, Bertini M, Del Bimbo A. Deep generative adversarial compression artifact removal. In: Proceedings of the IEEE International Conference on Computer Vision; 2017. p. 4826-35.
[29] Galteri L, Seidenari L, Bertini M, Del Bimbo A. Towards real-time image enhancement GANs. In: Computer Analysis of Images and Patterns: 18th International Conference, CAIP 2019, Salerno, Italy, September 3–5, 2019, Proceedings, Part I 18. Springer; 2019. p. 183-95.
[30] Galteri L, Seidenari L, Bertini M, Uricchio T, Del Bimbo A. Fast video quality enhancement using gans. In: Proceedings of the 27th ACM international conference on multimedia; 2019. p. 1065-7.
[31] Mameli F, Bertini M, Galteri L, Del Bimbo A. Image and video restoration and compression artefact removal using a NoGAN approach. In: Proceedings of the 28th ACM International Conference on Multimedia; 2020. p. 4539-41.
[32] Kaneko T, Harada T. Blur, noise, and compression robust generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2021. p. 13579-89.
[33] Pourreza R, Ghodrati A, Habibian A. Recognizing compressed videos: Challenges and promises. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops; 2019. p. 0-0.
[34] Yu S, Chen B, Xu Y, Chen W, Chen Z, Zhao T. Hevc compression artifact reduction with generative adversarial networks. In: 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP). IEEE; 2019. p. 1-6.
[35] Wang T, He J, Xiong S, Karn P, He X. Visual perception enhancement for HEVC compressed video using a generative adversarial network. In: 2020 International Conference on UK-China Emerging Technologies (UCET). IEEE; 2020. p. 1-4.
[36] Galteri L, Bertini M, Seidenari L, Uricchio T, Del Bimbo A. Increasing video perceptual quality with gans and semantic coding. In: Proceedings of the 28th ACM International Conference on Multimedia; 2020. p. 862-70.
[37] Liang J, Cao J, Fan Y, Zhang K, Ranjan R, Li Y, et al. Vrt: A video restoration transformer. arXiv preprint arXiv:220112288. 2022.
[38] Gholami A, Kim S, Dong Z, Yao Z, Mahoney MW, Keutzer K. A survey of quantization methods for efficient neural network inference. arXiv preprint arXiv:210313630. 2021.
[39] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. Springer; 2015. p. 234-41.
[40] Vaccaro F, Bertini M, Uricchio T, Del Bimbo A. Fast video visual quality and resolution improvement using SR-UNet. In: Proceedings of the 29th ACM International Conference on Multimedia; 2021. p. 1221-9.
[41] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial networks. Communications of the ACM. 2020;63(11):139-44.
[42] Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:151106434. 2015.
[43] Maas AL, Hannun AY, Ng AY, et al. Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml. vol. 30. Atlanta, Georgia, USA; 2013. p. 3.
[44] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in Neural Information Processing Systems 32. Curran Associates, Inc.; 2019. p. 8024-35. Available from: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.
[45] Bai J, Lu F, Zhang K, et al.. Open Neural Network Exchange (ONNX). GitHub; 2019. `https://github.com/onnx/onnx`.
[46] Corporation N. Torch-TensorRT; 2021. Software available from `https://github.com/NVIDIA/Torch-TensorRT`.
[47] Ma D, Zhang F, Bull DR. BVI-DVC: A training database for deep video compression. IEEE Transactions on Multimedia. 2021;24:3847-58.
[48] Galteri L, Seidenari L, Bertini M, Del Bimbo A. Deep universal generative adversarial compression artifact removal. IEEE Transactions on Multimedia. 2019;21(8):2131-45.