School of Mathematics, Physics and Natural Science
Master of Science Degree in Computer Science

Thesis

OPTIMIZATION TECHNIQUES OF DEEP LEARNING MODELS FOR VISUAL QUALITY IMPROVEMENT

TECNICHE DI OTTIMIZZAZIONE DI RETI NEURALI PER IL MIGLIORAMENTO DELLA QUALITÀ VISUALE

LORENZO PALLONI

Supervisor: *Marco Bertini*
Co-supervisors: *Leonardo Galteri, Donatella Merlini*

Academic Year 2021-2022

CONTENTS

Robot: *What is my purpose?*
Rick: *You pass butter.*
Robot: *. . . Oh my God.*
Rick: *Yeah, welcome to the club, pal.*
— *Rick and Morty*

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 artefacts, 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 [8]. Some of the commonly used techniques include median filtering, bilateral filtering, and wavelet-based denoising. While these methods are effective in restoring some aspects of the 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 restoring degraded videos. By leveraging deep neural networks to learn the underlying mapping between the degraded and the ground truth images, these methods allow for the restoration of high-frequency details and complex distortions in videos.

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 artefact 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 involves representing the weights and activations of a model with fewer bits, aiming to minimize 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.

This thesis focuses on examining the efficacy of quantization techniques for enhancing the inference speed and reducing the memory usage of deep learning models applied to video restoration tasks, including artefact removal and SR. The 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, the thesis investigates the implementation and evaluation of post-training quantization using TensorRT [2], an NVIDIA SDK designed for high-performance deep learning inference.

The findings of this thesis are expected to contribute to the knowledge and development of more practical and efficient video restoration models. These models could potentially be deployed in a variety of real-world applications, such as mobile devices, embedded systems, and IoT devices.

2

BACKGROUND

In this chapter, the existing literature on methods to address the video restoration task is reviewed, with a focus on recent deep-learning models for single-image super-resolution that are closely related to this thesis.

Dong et al. [6] present the SRCNN model using convolutional neural networks for single-image super-resolution, while Kim et al. introduce VDSR [24] and DRCN [25] models with a residual learning framework [20]. Dong et al. [7] and Shi et al. [55] efficiently map low-resolution to high-resolution images using deconvolution and sub-pixel convolution layers, respectively. LatticeNet [34], LapSRN [26], MemNet [58], SRDenseNet [59], and RDN [70] are other lightweight and efficient SR models. Haris et al. [19] and DSRN [18] focus on iterative up-sampling and down-sampling and dual-state recurrent networks, respectively. MSRN [29] and RFA [32] exploit image features efficiently by using different blocks, while recently attention mechanisms have also been used to improve super-resolution image quality [4, 40, 44, 69].

In recent years, SR models based on a generative adversarial network (GAN) framework have also gained popularity. SRGAN [28] is a notable example, in which the generator network is trained to produce high-resolution images that are perceptually similar to the ground truth images, while the discriminator network is trained to distinguish between the generated and ground truth images. Since then, several GAN-based SR models have been proposed, including EnhancedNet [53], ESRGAN [63], SPSR [35], SRFBN [31], and SRFlow [33].

In [10], Galteri et al. use adversarial training to remove artefacts from lossy compression algorithms for images and videos, training models with larger batches and utilizing an ensemble of networks. They also address real-time artefact removal using MobileNetV2-inspired generators with depth-wise separable convolutions [12, 13]. Mameli et al. [39] apply the no-GAN approach for compression artefact removal and super-resolution, using pre-training and a few GAN training iterations. Kaneko

et al. introduce BNCR-GAN [22], an architecture that removes noise, compression artefacts, and blurring artefacts by combining three models and using adaptive consistency losses.

Pourreza et al. [49] propose a GAN-based quality enhancement method to address accuracy loss in action recognition tasks caused by video compression, using a frame-recurrent strategy. Yu et al. develop VR-GAN [67] for HEVC compression-generated artefact removal, operating at the inter-frame level and using flow estimation for coherence. In [62], He et al. enhance HEVC compressed videos using adversarial loss combined with L1 loss, employing a residual network generator.

In [9], 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. The decoding side uses a Relativistic GAN and a segmentation-aware loss.

Video super-resolution poses a greater challenge than single-image super-resolution because it requires accurate prediction of both content and motion. Motion information is crucial for reconstructing high-resolution frames from several low-resolution images. The most effective multi-image super-resolution methods include sliding-window techniques and recurrent approaches, both of which can potentially restore more high-resolution details in target frames as they utilize more visual information. However, these methods must consider motion content between frames to achieve top-quality super-resolution results.

Recurrent neural networks, widely used for various vision tasks like classification, detection, and segmentation, can process input of any length by sharing model weights over time. These recurrent models can account for long-range dependencies among pixels, making them well-suited for video super-resolution tasks. In recent years, many video super-resolution models based on recurrent neural networks have been developed, demonstrating their effectiveness in this field.

3

METRICS

To evaluate the performance of deep learning models for video restoration, various metrics have been proposed in the current literature, which can be broadly categorized into traditional and perceptual metrics.

3.1 TRADITIONAL METRICS

Traditional metrics are based on simple numerical comparisons between the generated and ground-truth images. Some commonly used traditional metrics include PSNR, MSE, SSIM [65], and MS-SSIM [66].

PSNR (Peak Signal-to-Noise Ratio) is a widely used metric that measures the ratio of the peak signal power to the noise power in an image. It is calculated as the logarithm of the ratio of the maximum possible pixel value to the mean squared error between the predicted and ground-truth images. However, PSNR has been criticized for not being a reliable measure of image quality, as it does not correlate well with human perception.

MSE (Mean Squared Error) measures the average squared difference between the predicted and ground-truth images, with lower values indicating better image quality. However, like PSNR, it has been found to correlate poorly with human perception [16, 64].

SSIM (Structural SIMilarity) is a more sophisticated metric that takes into account both structural information and pixel values in the image. It measures the similarity between the predicted and ground-truth images based on their luminance, contrast, and structure, and has been found to correlate better with human perception than PSNR and MSE.

MS-SSIM (Multi-Scale SSIM) is an improved version of the SSIM metric that takes into account the multi-scale nature of the human visual system.

3.2 PERCEPTUAL METRICS

Perceptual metrics aim to evaluate image quality based on human perception, measuring the visual similarity between predicted and ground-truth images, rather than simply their pixel-wise differences. Examples of deep learning-based perceptual metrics include FID [21], LPIPS [68], LPIPS-Comp [46], E-LPIPS [23], and DISTS [5].

FID (Fréchet Inception Distance) is a perceptual metric used to evaluate the similarity between two sets of images by measuring the distance between their feature representations obtained from a pre-trained neural network.

LPIPS (Learned Perceptual Image Patch Similarity) computes the similarity between two images based on their perceptual similarity at the patch level, using a deep neural network trained on human perceptual judgments.

LPIPS-Comp (LPIPS with Saliency Map Comparison) is an extension of LPIPS that incorporates saliency maps to enhance the metric sensitivity to the most salient regions in the image.

E-LPIPS (Ensembled LPIPS) is an improved version of the LPIPS metric that employs an ensemble of neural networks trained on different subsets of images to improve the stability and robustness of the metric.

DISTS (Deep Image Structure and Texture Similarity) is based on a Siamese neural network, which takes two input images, and extracts features from them. These features are then compared at multiple levels to compute a final similarity score between the two images.

Other perceptual metrics such as MOS (Mean Opinion Score), 2AFC (Two Alternative Forced Choice), and JND (Just Noticeable Difference) are all examples of perceptual metrics that are based on subjective human evaluations of image quality.

MOS involves asking human subjects to rate the quality of the predicted images on a scale from 1 to 5, with the MOS score calculated as the average of these ratings. 2AFC involves presenting two images to human subjects and asking them to choose the one that appears to be of higher quality, while JND involves asking subjects to identify the minimum perceptible difference between two images.

Overall, the choice of metric for evaluating super-resolution models depends on the specific application and the goals of the study. Traditional metrics such as PSNR, MSE, and SSIM are simple to compute and provide a good baseline for comparison. Perceptual metrics such as LPIPS and MOS provide a more accurate measure of human perception but are more

complex to compute and require additional resources. A combination of both traditional and perceptual metrics can provide a comprehensive evaluation of the performance of super-resolution models.

3.3 NO-REFERENCE METRICS

While Full-Reference Image Quality Assessment (FR-IQA) measures have been discussed thus far, it is worth noting that No-Reference IQA (NR-IQA) metrics are also commonly employed to assess processed images without their original counterparts. Examples of such NR-IQA metrics include BRISQUE [41], NIQE [42], PIQUE [61], and CONTRIQUE [38].

BRISQUE (Blind/Referenceless Image Spatial Quality Evaluator) is an NR-IQA metric that relies on a spatial Natural Scene Statistics (NSS) model [52] of locally normalized luminance coefficients in the spatial domain, as well as the model for pairwise products of these coefficients. BRISQUE predicts the image quality score by using a support vector regression model trained on an image database with corresponding Differential Mean Opinion Score (DMOS) values. DMOS is the difference between the perfect quality score and the MOS. Thus, a higher MOS value and a lower DMOS value indicate better quality. The database contains images with known distortion such as compression artefacts, blurring, and noise, and it contains pristine versions of the distorted images.

NIQE (Natural Image Quality Evaluator) computes the distance between the NSS-based features calculated from a given image to the features obtained from an image database used to train the model. The features are modelled as multidimensional Gaussian distributions.

PIQUE (Perception-based Image QUality Evaluator) uses an opinion-unaware methodology that does not require any training data. It extracts local features to predict quality and estimates quality only from perceptually significant spatial regions. PIQUE can generate a fine-grained block-level distortion map and it has low computational complexity despite working at the block-level.

CONTRIQUE (CONTRastive Image QUality Evaluator) obtains image quality representations in a self-supervised manner. It tries to predict distortion type and degree as an auxiliary task to learn features from an unlabeled image dataset containing a mixture of synthetic and realistic distortions. A convolutional neural network is trained using a contrastive pairwise objective to solve the auxiliary problem. The network weights are frozen during evaluation and a linear regressor maps the learned representations to quality scores in a No-Reference setting.

Datasets like LIVE [54], TID2008 [48], CSIQ [27], and TID2013 [47] are considered FR-IQA datasets since they need a reference image to assess the image quality. In contrast, AVA [43] and LIVE In the Wild [14] are NR-IQA datasets, where image quality is evaluated independently without the necessity of a reference image.

3.4 VIDEO QUALITY METRICS

Some image quality metrics, such as PSNR or SSIM, can be used to assess the quality of a video, by considering one frame at a time. The quality measure for each frame can be recorded and averaged over time to determine the overall quality of the video. However, this approach does not take into account certain types of degradation that occur over time, such as moving artefacts caused by packet loss and its concealment. A video quality metric that takes into account the temporal aspects of quality degradation, like VMAF [30], may provide more accurate predictions of how humans perceive quality.

VMAF (Video Multimethod Assessment Fusion) is a perceptual video quality metric developed by Netflix to evaluate the visual quality of compressed videos. VMAF combines multiple elementary quality metrics, aggregates them through pooling strategies, and uses a machine learning model to estimate human-perceived video quality. It involves four main steps:

1. Feature extraction: VMAF extracts low-level features from both the original and compressed videos, focusing on aspects that affect human-perceived video quality. Primary features include VIF (shared visual information), SSIM, ADM (local luminance and contrast deviations), and Motion (which affects compression artefact visibility).

2. Feature pooling: VMAF segments video frames into non-overlapping blocks and calculates elementary metrics (VIF, SSIM, ADM) for each block. It then aggregates these scores across all blocks in a frame using a pooling strategy (mean or harmonic mean), resulting in a per-frame score for each metric.

3. Support Vector Regression model: VMAF employs a support vector regression model to combine per-frame elementary metrics into a single quality score. The model is trained on a dataset of videos annotated with subjective quality scores, reflecting human opinions.

4. Final VMAF score: The support vector regression model outputs a per-frame VMAF score representing perceived video quality. These per-frame scores are typically pooled (using the mean) to compute a single VMAF score for the entire video.

4

ARCHITECTURES

In this chapter, an overview of the UNet and SRUNet model architectures used in the experiments is provided. Additionally, the employed training setup is outlined, which consists of a Generative Adversarial Network (GAN) framework that incorporates LPIPS and SSIM as the generator loss.

4.1 UNET ARCHITECTURE

The UNet architecture was introduced by Ronneberger et al. [51] in 2015 for biomedical image segmentation tasks.

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

The UNet model is composed of an encoder and a decoder, with skip connections between them. The encoder takes the input 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 input image x and returns a feature map $\mathbf{f}$:

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

The decoder takes the feature map $\mathbf{f}$ as input and applies a series of convolutional layers to upsample the image while preserving the details. The decoder can be represented as a function g_ϕ that takes the feature map $\mathbf{f}$ and returns the generated high-resolution output image $\hat{y}$:

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

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

$$z = x + W_2\sigma(W_1 x + b_1) + b_2$$

where W_1 and W_2 are weight matrices, b_1 and b_2 are bias vectors, and σ is a non-linear activation function.

Figure 1 shows a high-level representation of the UNet architecture.

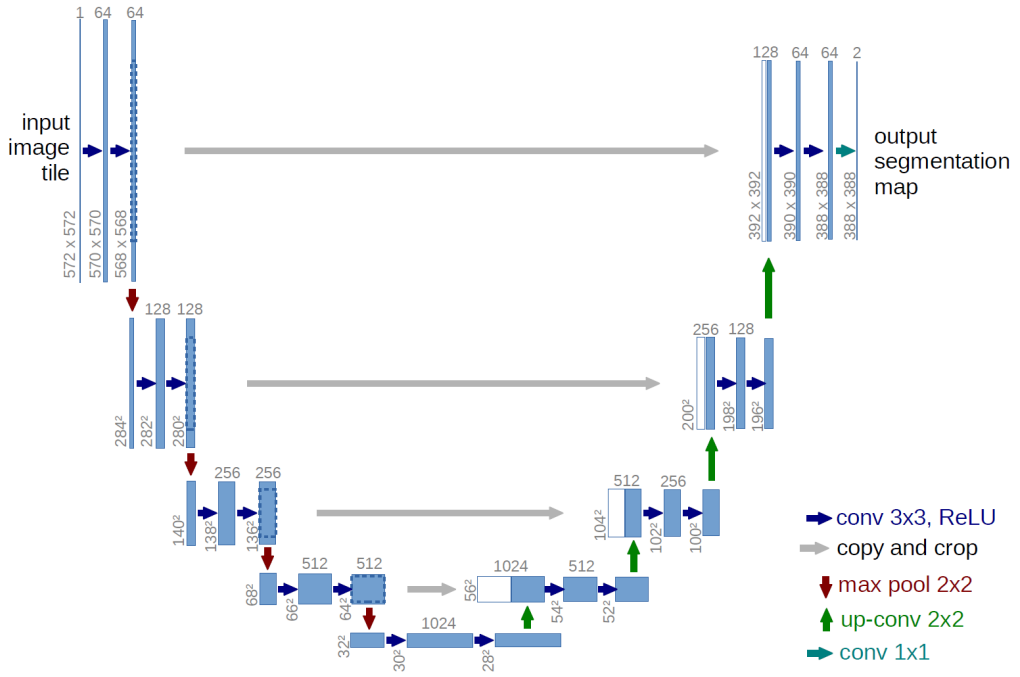

Figure 1: UNet architecture. The name is inspired by its shape like a U. It has a contracting path on the left side, a fully-connected layer in the middle, and an expansive path on the right side. The contracting path consists of repeated blocks of convolutional layers followed by ReLU activation and ending with max-pooling. The output is passed through a fully-connected layer and then to the expansive path. The expansive path has the same number of blocks as the contracting path but up-samples instead of max-pooling. The input to each block is the output of the previous block concatenated with the corresponding block from the contracting path.

4.2 SRUNET ARCHITECTURE

SRUNet is an adaptation of the UNet architecture for super-resolution and compression artefact removal, proposed in [60] 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 scaled up with a linear interpolation function and the output of the second to the last layer scaled up 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 up-sample 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 but at double resolution. Alternatives, such as bilinear upsample with convolution, the transposed convolution, or even the reshuffling to a higher dimension with the same depth, would add an exaggerated overhead since they would work in the high-resolution space.

Modelling the problem as producing a residual on the top of the up-sampled 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 from the up-sampled image. Furthermore, faster convergence of the training process is ensured.

Figure 2 shows a high-level representation of the SRUNet architecture.

4.3 TRAINING SETUP

To train both the UNet and SRUNet models, a Generative Adversarial Network (GAN) framework was employed. Introduced in [17], 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 true 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

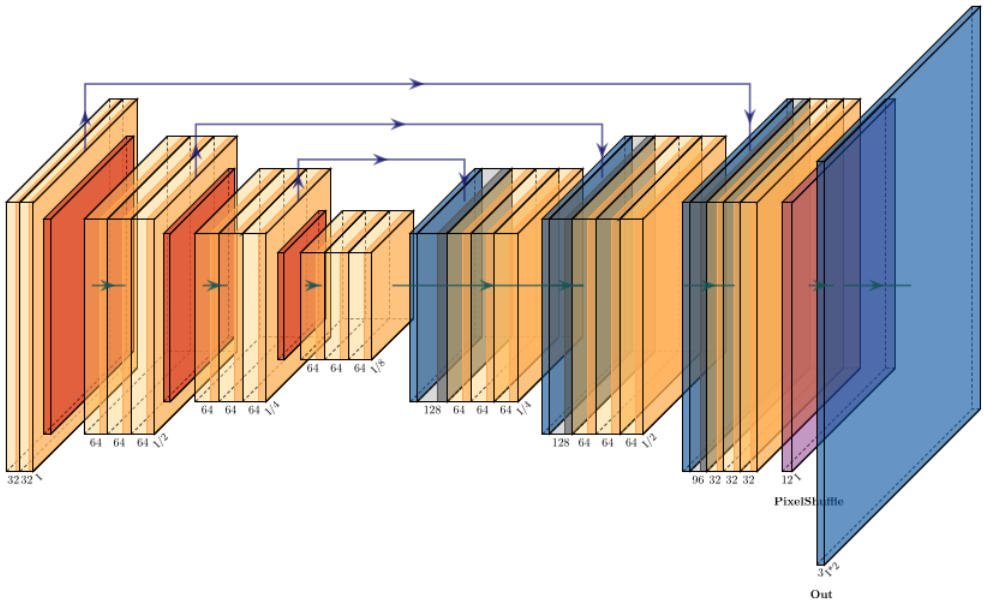

Figure 2: SRUNet architecture. It differs from a classical UNet architecture mainly by the decrease in the number of filters in each convolutional layer, and the use of a residual layer as the final layer.

competition between the two networks leads to the generator improving over time and producing images with more quality.

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

In this setup, LPIPS and SSIM were used as the 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. See chapter 3, for further details on perceptual and non-perceptual metrics.

The generator loss can be represented as follows:

$$\mathcal{L}_G = \underbrace{-\log(D(\hat{y}))}_{\text{adversarial loss}} + \underbrace{w_{\text{LPIPS}} \cdot \text{LPIPS}(\hat{y}, y) + w_{\text{SSIM}} \cdot (1 - \text{SSIM}(\hat{y}, y))}_{\text{content loss}}$$

where $\hat{y}$ is the generated high-resolution image, y is the ground truth high-resolution image, $D(\hat{y})$ is the probability score outputted by 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 the one used in [28], where the authors followed the architectural guidelines described in [50] and incorporated LeakyReLU [37] activation with α set to 0.2 (see fig. 3) while avoiding max-pooling in the network.

The discriminator network structure is illustrated in fig. 4. It includes eight convolutional layers, with a 3×3 filter kernel size that increases by a factor of 2, ranging from 64 to 512 kernels, similar to the VGG network [56]. Strided convolutions are employed to decrease the image resolution each time the number of features doubles. The final output of 512 feature maps is fed into two dense layers, followed by a sigmoid activation function to obtain a classification probability for the input sample.

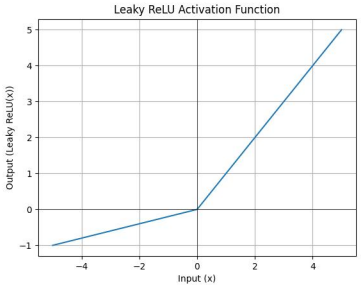

Figure 3: LeakyReLU ($\alpha := 0.2$).

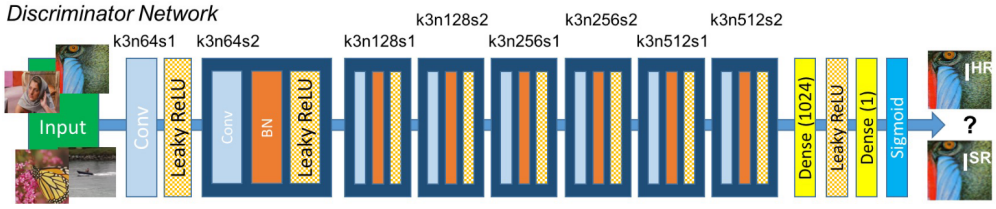

Figure 4: The architecture of the discriminator network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

During the model training process, the Adam optimizer was utilized with a learning rate of 1×10^{-4}. The model was trained for 191,268 iterations, with 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 sole data augmentation strategy applied was the 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 [36], 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 content types, such as natural scenes, man-made objects, and cityscapes, with some examples shown in fig. 5.

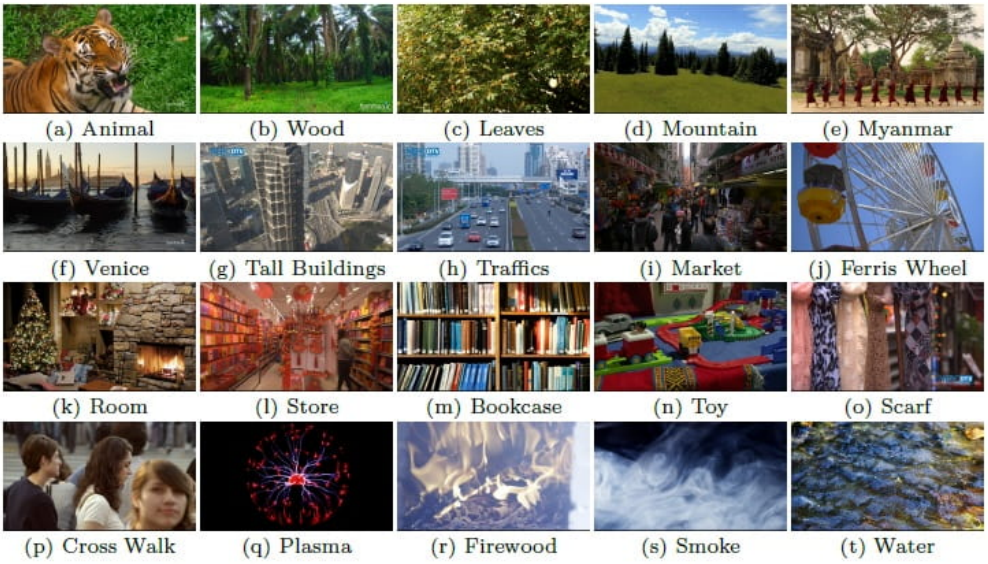

Figure 5: Some frame examples from the BVI-DVC dataset.

While the original resolution was 2160p, the dataset authors created down-scaled versions at 1080p, 540p, and 270p, yielding 800 sequences and a total of 51,200 frames. The original BVI-DVC dataset served as the ground truth for the training dataset. Input data was 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 aimed to mitigate the mode-collapse issue described in [11]. 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 down-scaled version of high-quality videos, was crucial for applying super-resolution to compressed videos and performing both artefact reduction and super-resolution. Training the model solely for super-resolution

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

5

OPTIMIZATIONS

This chapter will explore various methods to enhance deep learning models for super-resolution and visual artefact removal. The quality of the resulting output can be significantly influenced by multiple factors, such as the model's architecture, the loss function chosen, the evaluation metric(s), the way the model weights are updated during training, and other relevant hyperparameters.

The loss function is a crucial component of these models, as it measures the difference between the generated output and the ground-truth output. Depending on the task at hand, different loss functions can be utilized, such as the widely used mean squared error (MSE) and mean absolute error (MAE), as well as perceptual loss - discussed in chapter 3 - which considers high-level features of the image such as FID, LPIPS, DISTS, among others available in the literature.

The network architecture is also an important factor that determines the complexity of the model and its capacity to capture intricate patterns in the input data. For example, the UNet and SRUNet architectures used in this work are both based on CNNs with an encoder-decoder structure, use residual layers, skip connections, interpolation and pixel shuffle modules. Further details on these model architectures can be found in chapter 4.

To prevent overfitting and improve generalization ability, regularization techniques can be employed. These techniques include dropout and data augmentation. Dropout randomly drops out neurons during training to prevent overfitting, while data augmentation artificially expands the training dataset by applying transformations such as rotation, cropping, and flipping. Some of these techniques, among others were used in this research, as described in section 4.3

5.1 QUANTIZATION

Another approach to optimise deep learning models, and one of the primary focuses of this thesis, is quantization. Quantization techniques are widely studied for their ability to decrease the computational complexity and memory usage of deep learning models while maintaining their accuracy and performance. This is particularly significant for enabling the deployment of deep learning models on devices with limited resources, such as mobile phones and embedded systems. Quantization is a technique that reduces the memory footprint and computation requirements of deep learning models by representing their parameters and activations with 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 involves dividing the dynamic range of the weights and activations into a fixed number of equally spaced levels. Non-uniform quantization, on the other hand, assigns more bits to important weights and activations and fewer bits to less important ones allowing for more precise capture of signal information. However, implementing non-uniform quantization schemes on common computational hardware, such as GPUs and CPUs, can be challenging. Therefore, the current standard method for quantization is uniform quantization due to its simplicity and efficient mapping to hardware [15].

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 re-training the model, which is known as Quantization-Aware Training (QAT), while the other approach is to modify the parameters without re-training, a method commonly referred to as Post-Training Quantization (PTQ).

QAT involves re-training a neural network model with quantized parameters to address the perturbation introduced by quantization, which

may shift the model away from its convergence point in floating point precision. During QAT, forward and backward passes are performed in floating point on the quantized model, with the model parameters being quantized after each gradient update using projected gradient descent. It is crucial to perform this projection after the weight update in floating point precision to maintain accuracy. Back-propagation in floating point is also essential to avoid zero-gradient or gradients with high errors, which can occur when accumulating gradients in quantized precision, particularly in low-precision scenarios. The non-differentiable quantization operator is approximated using the Straight Through Estimator (STE) to address this issue. Despite being a coarse approximation, STE typically works well in practice except for ultra-low-precision quantization such as binary quantization.

Although QAT has been demonstrated to be effective, its main drawback is the computational cost of re-training the neural network model. This re-training may take several hundred epochs to recover accuracy, particularly in low-bit precision quantization scenarios. If a quantized model is expected to be deployed for a long time and both efficiency and accuracy are critical, then the investment in re-training is likely worthwhile. However, this may not be the case for all models, particularly those with a short lifetime (e.g., models that require online learning or frequent updates to the training set due to a test distribution that shifts over time).

An alternative approach that avoids QAT overhead of re-training the model is Post-Training Quantization. During PTQ, the quantization parameters for both weights and activations are established without performing any re-training on the neural network model. Consequently, PTQ is a rapid technique for quantizing neural network models. However, this approach often results in lower accuracy compared to QAT.

5.2 TENSORRT TO SPEED UP INFERENCE

TensorRT is an inference optimization tool developed by NVIDIA that can accelerate deep learning models on NVIDIA GPUs. In recent years, there has been increasing interest in integrating TensorRT with PyTorch [45], 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 [1] 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 [3]. 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 scripting language and runtime environment for PyTorch that allows you to serialize and save PyTorch models and run them in production without needing the full PyTorch framework. It enables the execution of PyTorch models efficiently on a wide range of devices.

FX is a new, experimental toolkit for building high-performance machine learning models in PyTorch. It allows developers to compose models using pure Python syntax and provides tools for optimizing and deploying those models on various hardware platforms. FX is designed to enable fast experimentation and easy production deployment of PyTorch models.

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, you must 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 compiled modules can integrate seamlessly into the JIT runtime. Once compiled, the optimized graph should feel no different from running a TorchScript module. Additionally, you have access to TensorRT's suite of configurations at compile time, allowing you to specify operating precision (FP32/FP16/INT8) and other settings for your module.

The Torch-TensorRT Python API offers a simple and user-friendly approach to utilizing PyTorch data-loaders in conjunction with TensorRT calibrators. By specifying the desired configuration, the DataLoaderCalibrator class can be employed to establish a TensorRT calibrator.

Several studies have investigated the use of TensorRT to improve the inference performance of deep learning models in various domains. For example, Stäcker et al. [57] applied TensorRT-based post-training quantization to representative object detection networks, such as RetinaNet for image-based 2D object detection and PointPillars for LiDAR-based 3D object detection. The authors focused on deploying these networks on an edge AI platform, examining the conversion process from the

PyTorch training environment to the deployment environment, and comparing the performance of TensorRT and TorchScript. Their experiments demonstrated slight advantages of TensorRT for convolutional layers and TorchScript for fully connected layers. Furthermore, they analyzed the trade-off between runtime and performance when optimizing the deployment setup, finding that quantization significantly reduced runtime while having a minimal impact on detection performance.

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

5.3 DATA-LOADER TO SPEED UP TRAINING

In this section is presented a novel and custom data-loading pipeline implemented for training super-resolution models. Its primary goal is to reduce computational bottlenecks arising from four interrelated factors: GPU RAM capacity, GPU batch processing time, CPU image fetching time, and image size. The models were trained on an Ubuntu 20.04 server using a single GPU Titan Xp with 12GB RAM. The BVI-DVC dataset is stored on an external HDD, which cannot be transferred to an SSD due to its size.

Initially, a naive PyTorch data loader was implemented to load one image at a time, extracting only one patch from each loaded image. However, this approach led to a significant bottleneck for the GPU as the CPU's processing time exceeded that of the GPU's forward/backward pass. A more efficient solution would be to store patches from each frame (or a subset of frames) in SSDs near the CPU, allowing parallel processing to prevent GPU bottlenecks.

To overcome these challenges, the custom data-loading pipeline comprises two main components: the BufferGenerator and the BatchGenerator. The BufferGenerator serves buffered chunks of items from a list and applies a function to each item before it enters the buffer. The BatchGenerator generates batches of patches taken from prefetched frames, with both image selection and cropping taking place randomly.

The pipeline also provides different behaviours for the training, validation, and testing stages. In the training stage, the data is shuffled, while in the validation and testing stages, it remains in its original order. The custom data-loading pipeline ensures efficient training of super-

resolution models without causing GPU bottlenecks or excessive memory consumption.

6

EXPERIMENTS

In this chapter, various experiments and their corresponding results will be showcased, highlighting the inference speed and image quality assessment. 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.

6.1 QUANTITATIVE RESULTS

Tables 1 and 2 present the performance of UNet and SRUNet models, as well as their TensorRT-optimized versions (FP32, FP16, INT8), on perceptual and traditional metrics over 60 test frames.

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 have similar performance in LPIPS and DISTS, 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 to 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 2, both 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 3 shows VMAF scores (see section 3.4) for both UNet and SRUNet models, and their optimized versions using a test video of 2-minute length. The harmonic mean is shown because if there were outliers (i.e., frames for which the score is lower than average by more than a certain

	LPIPS ↓	DISTS ↓	BRISQUE ↓
UNet	**0.2897 ± 0.0138**	**0.1222 ± 0.0067**	31.1372 ± 1.1690
UNet-FP32	**0.2897 ± 0.0138**	**0.1222 ± 0.0067**	31.1373 ± 1.1684
UNet-FP16	0.2898 ± 0.0138	0.1223 ± 0.0067	31.1383 ± 1.1691
UNet-INT8	0.3041 ± 0.0137	0.1283 ± 0.0066	**29.6849 ± 1.0138**
SRUNet	0.3111 ± 0.0151	**0.1717 ± 0.0047**	27.3738 ± 3.5705
SRUNet-FP32	0.3111 ± 0.0151	**0.1717 ± 0.0047**	27.3736 ± 3.5697
SRUNet-FP16	0.3111 ± 0.0151	**0.1717 ± 0.0047**	27.3790 ± 3.5739
SRUNet-INT8	**0.3068 ± 0.0137**	0.1722 ± 0.0044	**26.1546 ± 3.2273**

Table 1: Evaluations on perceptual metrics on 60 test frames (mean ± standard deviation).

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.

Execution times for both UNet and SRUNet, along with their TensorRT-optimized versions (FP32, FP16, INT8), are shown in table 4. Both UNet and SRUNet models demonstrate reduced evaluation times across all precisions. The INT8-optimized version exhibit the fastest evaluation times for both models, 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. For a visual representation, see fig. 8.

The INT8-optimized UNet and SRUNet models achieve 2.38X and 2.26X speedup compared to their original implementations, respectively. Furthermore, memory consumption was reduced up 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 versions.

	SSIM ↑	MS-SSIM ↑	PSNR ↑
UNet	**0.8952 ± 0.0084**	**0.8517 ± 0.0067**	**21.6506 ± 0.1269**
UNet-FP32	**0.8952 ± 0.0084**	**0.8517 ± 0.0067**	**21.6506 ± 0.1269**
UNet-FP16	**0.8952 ± 0.0084**	**0.8517 ± 0.0067**	**21.6506 ± 0.1269**
UNet-INT8	0.8941 ± 0.0084	0.8508 ± 0.0067	21.6388 ± 0.1286
SRUNet	**0.8894 ± 0.0084**	**0.8457 ± 0.0062**	21.3670 ± 0.1248
SRUNet-FP32	**0.8894 ± 0.0084**	**0.8457 ± 0.0062**	21.3670 ± 0.1248
SRUNet-FP16	**0.8894 ± 0.0084**	**0.8457 ± 0.0062**	**21.3671 ± 0.1248**
SRUNet-INT8	0.8882 ± 0.0084	0.8442 ± 0.0062	21.3320 ± 0.1224

Table 2: Evaluations on traditional metrics on 60 test frames (mean ± standard deviation).

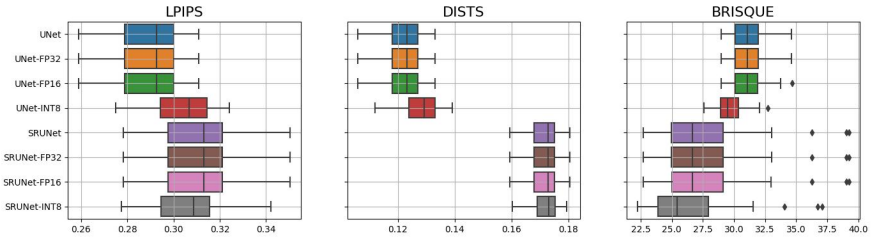

Figure 6: Box-plots of perceptual metrics on 60 test frames (the lower, the better).

	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 3: VMAF scores on a 120-second-test video.

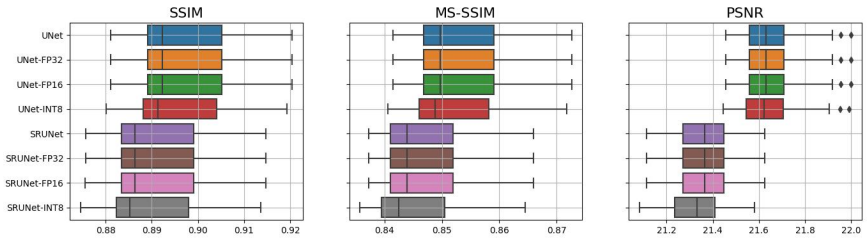

Figure 7: Box-plots of traditional metrics on 60 test frames (the higher, the better).

	times [s] ↓	speedup ↑
UNet	0.0348 ± 0.0004	
UNet-FP32	0.0279 ± 0.0004	1.247X
UNet-FP16	0.0279 ± 0.0004	1.247X
UNet-INT8	**0.0146 ± 0.0006**	**2.38X**
SRUNet	0.0123 ± 0.0001	
SRUNet-FP32	0.0087 ± 0.0005	1.41X
SRUNet-FP16	0.0087 ± 0.0004	1.41X
SRUNet-INT8	**0.0054 ± 0.0006**	**2.27X**

Table 4: Evaluation times over 300 runs (mean ± standard deviation).

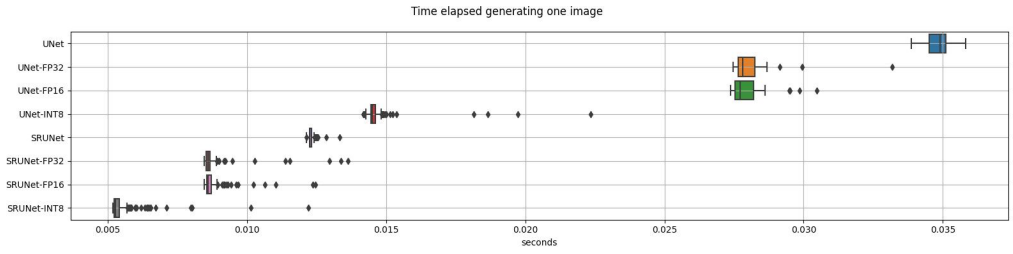

Figure 8: Average times elapsed in seconds for generating one image using different versions of UNet and SRUNet implementations.

6.2 QUALITATIVE RESULTS

This section aims to showcase the consistent performance of non-optimized and TensorRT-optimized UNet and SRUNet models in restoring distorted frames. Evaluations are displayed in Figures 9 to 14, 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 forementioned figures are 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.

Figure 9: Qualitative results evaluating UNet, UNet-FP32, UNet-FP16, and UNet-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).

Figure 10: 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).

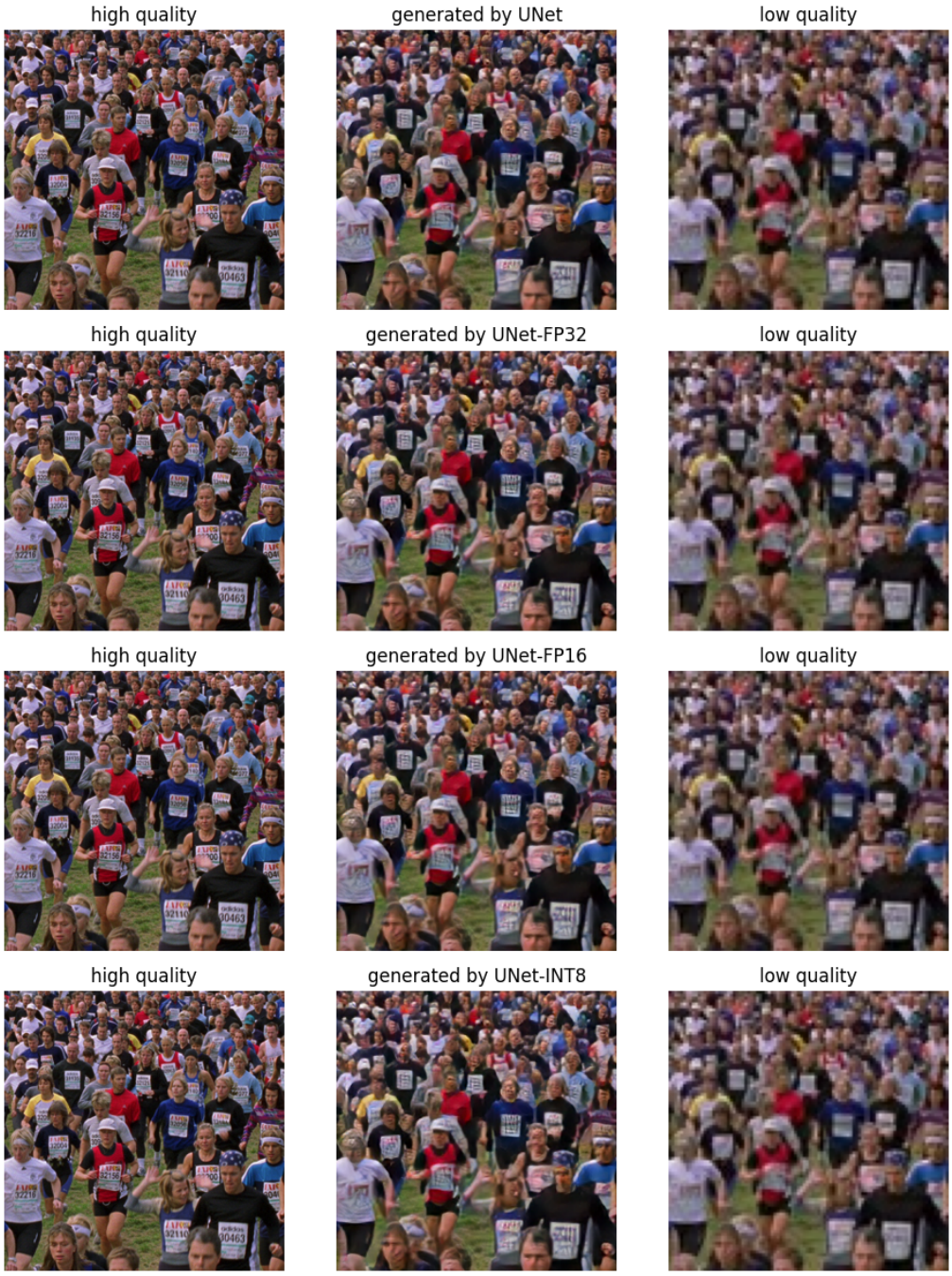

Figure 11: Qualitative results evaluating UNet, UNet-FP32, UNet-FP16, and UNet-INT8 models on a compressed and down-scaled 96×96 patch of a test video frame producing a 384×384 patch displayed in the middle column, along with the ground truth (left column), and the resized version using bicubic interpolation (right column).

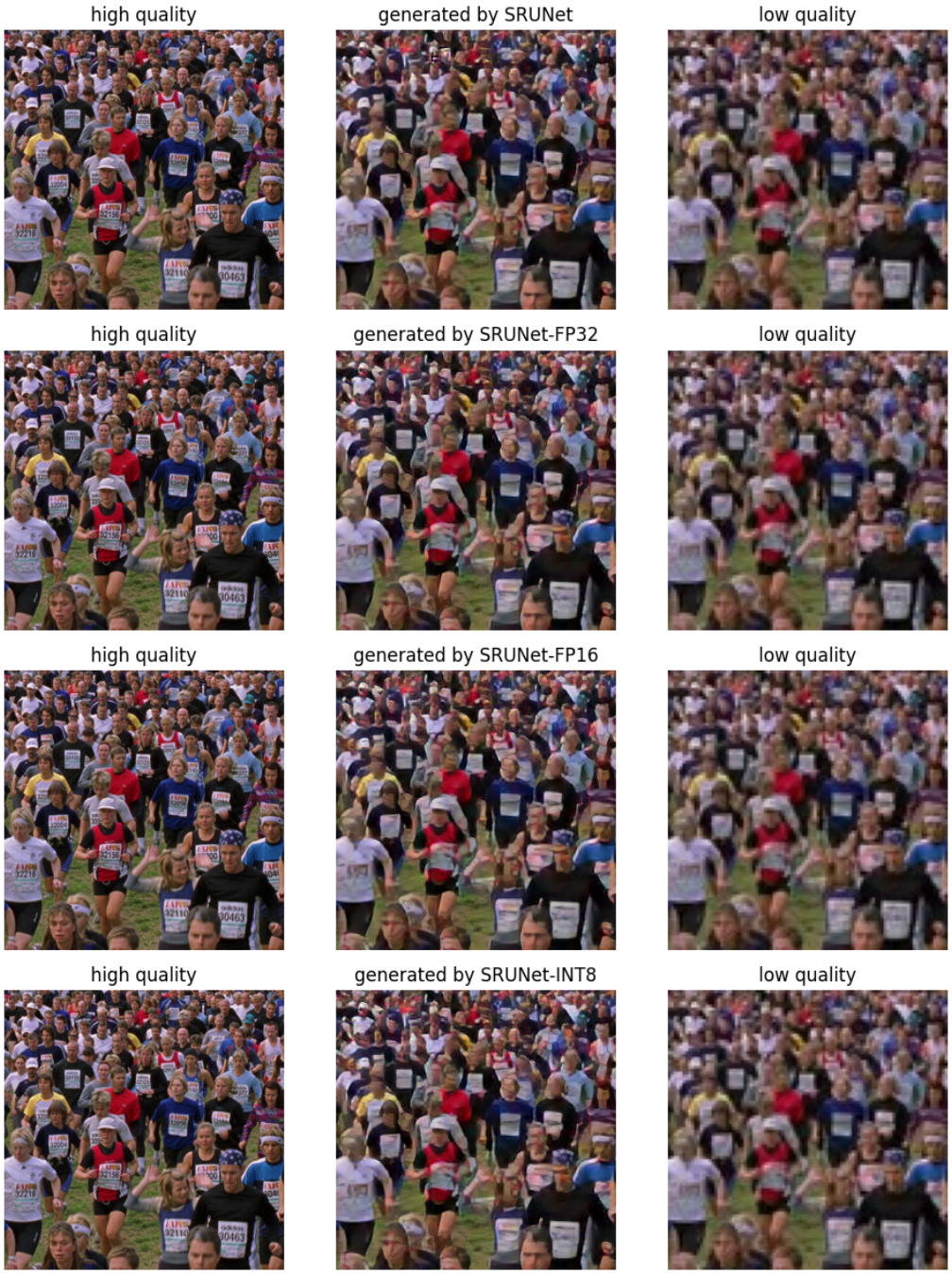

Figure 12: Qualitative results evaluating SRUNet, SRUNet-FP32, SRUNet-FP16, and SRUNet-INT8 models on a compressed and down-scaled 96×96 patch of a test video frame producing a 384×384 patch displayed in the middle column, along with the ground truth (left column), and the resized version using bicubic interpolation (right column).

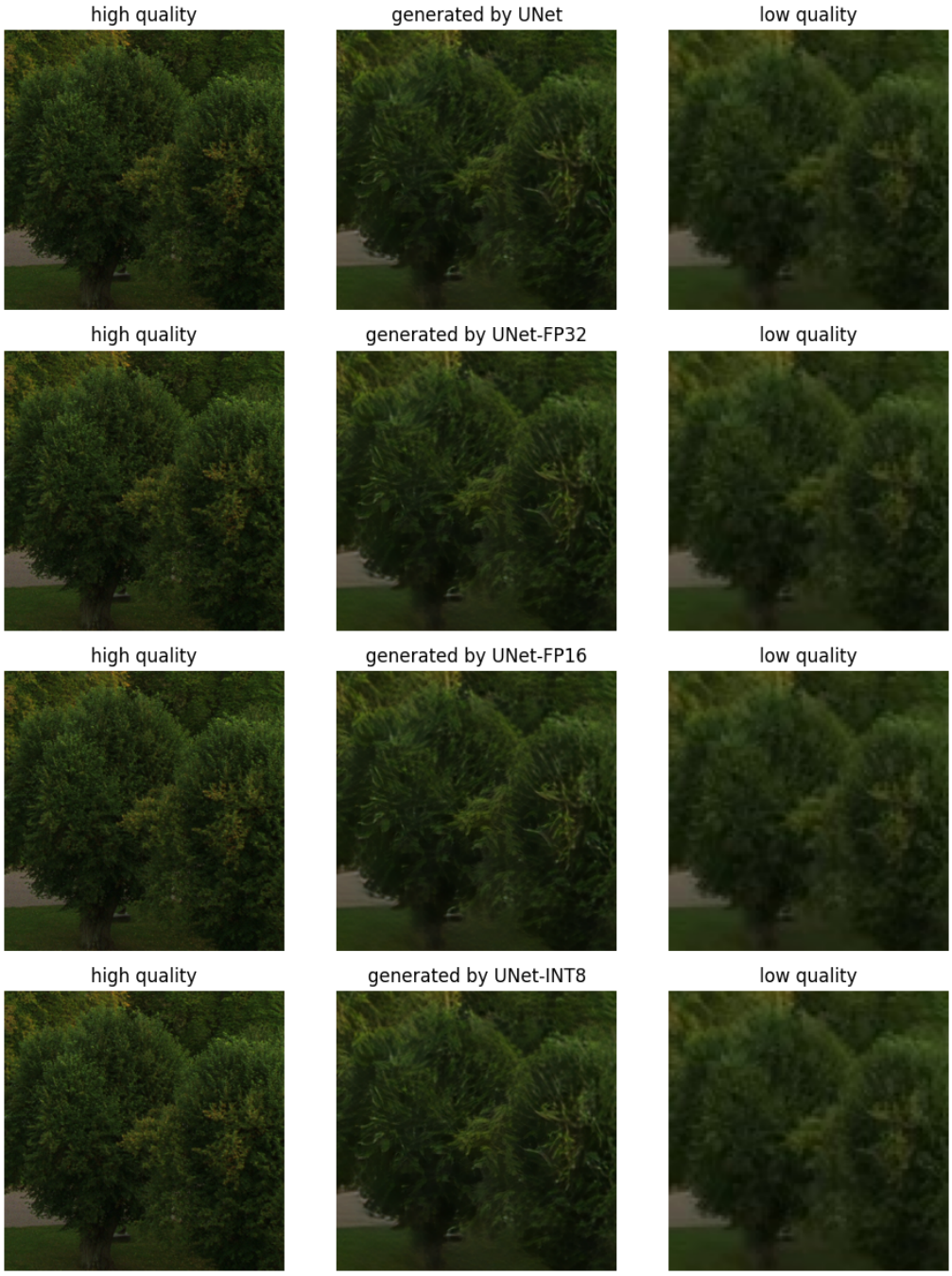

Figure 13: Qualitative results evaluating UNet, UNet-FP32, UNet-FP16, and UNet-INT8 models on a compressed and down-scaled 96×96 patch of a test video frame producing a 384×384 patch displayed in the middle column, along with the ground truth (left column), and the resized version using bicubic interpolation (right column).

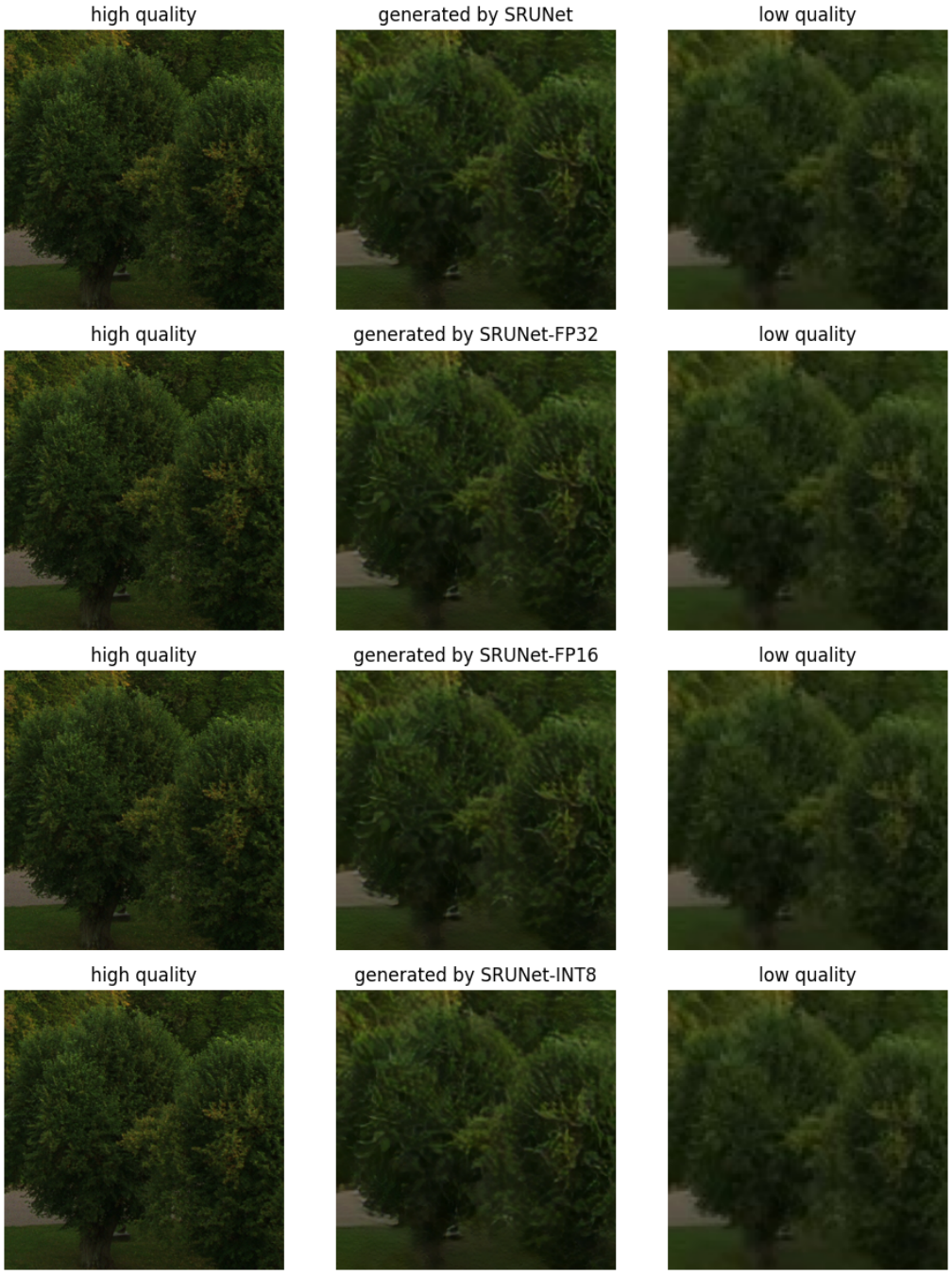

Figure 14: Qualitative results evaluating SRUNet, SRUNet-FP32, SRUNet-FP16, and SRUNet-INT8 models on a compressed and down-scaled 96×96 patch of a test video frame producing a 384×384 patch displayed in the middle column, along with the ground truth (left column), and the resized version using bicubic interpolation (right column).

7

CONCLUSIONS

In this study, the integration of post-training quantization techniques was investigated to optimize deep learning models for super-resolution during inference. The results indicate that reducing the precision of weights and activations in both the UNet and SRUNet models substantially decreases 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, a powerful inference optimization tool developed by NVIDIA, also played a key role in achieving these results. By integrating TensorRT with PyTorch, the efficiency of the model was further improved taking advantage of the INT8 computational capabilities of recent NVIDIA GPUs.

The outcomes of this thesis show promising results, and there are still areas for future research. For example, 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 greater detail.

It is hoped that these findings will stimulate additional advancements and applications for future research in optimizing deep learning models for video quality improvement.

8

ACKNOWLEDGEMENTS

The completion of this thesis marks the end of my Master's degree in Computer Science and represents a significant milestone in my life journey. I would like to take this opportunity to express my deepest gratitude to everyone who has supported and impacted me along the way. I apologize in advance to anyone I may inadvertently forget.

I would like to thank Professors Leonardo Grilli, Carla Rampichini, Emanuela Dreassi, Anna Gottard, Fabrizio Cipollini, Donatella Merlini, and Cecilia Verri among others from DISIA, as well as Professors Fabio Shoen, Marco Bertini, and Paolo Frasconi among others from DINFO. They not only taught me "maths stuff" but also contributed to my personal growth and development in various ways.

I am particularly grateful to my supervisor Marco Bertini and co-supervisor Leonardo Galteri for their invaluable guidance and the engaging conversations we shared during our weekly catch-ups while working on this thesis.

I appreciate the guidance and support of Marco Carrettoni and Luigi Di Lillo during my time as a Data Scientist at Swiss Re, and the people at Henesis who became not only coworkers but also friends: Riccardo Fava, Luca Baldini, Manuel Carzaniga, Davide Vurro, and Konstantin Koshmak. Special thanks to Luca Ascari, Maria Josè Vallejo, Freddy Symons, Stefano Lai, and Andrea Valenti for their direct and indirect help with my thesis.

I would also like to thank Edoardo Baroncelli for making me believe I could succeed in Maths, and Sara Agostini, my psychotherapist, who helped me cope with my demons. Special thanks to the Sicari and Benci-Bonini families for hosting me and making me feel at home, Mattia Pasqui for his unique perspective on life, Greta Cavaciocchi for keeping me away from bad habits, and Lucio Martelli and David Goggins for inspiring me.

Michele De Vita and Emilio Cecchini provided invaluable assistance during the initial stages of my Master's degree and became two of my closest friends.

Throughout my journey, I have been incredibly fortunate to meet individuals who have come to feel like family, touching my life with their warmth and support. With heartfelt affection, I extend my deepest gratitude to Matteo Lo Nigro, Simone Bonfanti, and Bruno Naselli from my early years in Quarrata; Giovanni Poli, Mirko Galardi, Tommaso Lenzi, Lorenzo Focardi, Shadi Moini, Noemi Benci, and Agnese Parca (aka "The Paranormals") who accompanied me through my undergraduate years; Emma Fani, Sole Urciullo, and Gabriele Montrone, who made my time spent at The Student Hotel truly memorable; and Marta Biondi and Lisa Modena (aka "GinJandPatate"), my current roommates in Parma who continue to brighten my days.

I would like to thank my family, including my father Massimo Palloni, with whom I had a complex and impactful relationship; my siblings Alessandro, Silvia, and Martina Palloni; my grandmother Gabriella Becagli; my uncles Gabriele and Gionata Finocchi; and my aunts Elena Finocchi and Ilaria Ciottoli. My mother, Beatrice Finocchi, has been my most significant support, and I am eternally grateful for her love and assistance.

Lastly, I want to thank Giovanni Barbarani, the most brilliant person I know and the closest person to me. Your influence on my life choices is immeasurable.

BIBLIOGRAPHY

[1] J. Bai, F. Lu, K. Zhang, et al. Open neural network exchange (onnx). `https://github.com/onnx/onnx`, 2019. (Cited on page 25.)

[2] N. Corporation. Nvidia tensorrt, 2016. Software available from `https://developer.nvidia.com/tensorrt`. (Cited on page 6.)

[3] N. Corporation. Torch-tensorrt, 2021. Software available from `https://github.com/NVIDIA/Torch-TensorRT`. (Cited on page 26.)

[4] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11065–11074, 2019. (Cited on page 7.)

[5] K. Ding, K. Ma, S. Wang, and E. P. Simoncelli. Image quality assessment: Unifying structure and texture similarity. *arXiv preprint arXiv:2004.07728*, 2020. (Cited on page 10.)

[6] C. Dong, C. C. Loy, K. He, and X. Tang. 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*, pages 184–199. Springer, 2014. (Cited on page 7.)

[7] C. Dong, C. C. Loy, and X. Tang. 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*, pages 391–407. Springer, 2016. (Cited on page 7.)

[8] L. Fan, F. Zhang, H. Fan, and C. Zhang. Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art*, 2(1):1–12, 2019. (Cited on page 5.)

[9] L. Galteri, M. Bertini, L. Seidenari, T. Uricchio, and A. Del Bimbo. Increasing video perceptual quality with gans and semantic coding. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 862–870, 2020. (Cited on page 8.)

[10] L. Galteri, L. Seidenari, M. Bertini, and A. Del Bimbo. Deep generative adversarial compression artifact removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4826–4835, 2017. (Cited on page 7.)

[11] L. Galteri, L. Seidenari, M. Bertini, and A. Del Bimbo. Deep universal generative adversarial compression artifact removal. *IEEE Transactions on Multimedia*, 21(8):2131–2145, 2019. (Cited on page 20.)

[12] L. Galteri, L. Seidenari, M. Bertini, and A. Del Bimbo. 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*, pages 183–195. Springer, 2019. (Cited on page 7.)

[13] L. Galteri, L. Seidenari, M. Bertini, T. Uricchio, and A. Del Bimbo. Fast video quality enhancement using gans. In *Proceedings of the 27th ACM international conference on multimedia*, pages 1065–1067, 2019. (Cited on page 7.)

[14] D. Ghadiyaram and A. C. Bovik. Massive online crowdsourced study of subjective and objective picture quality. *IEEE Transactions on Image Processing*, 25(1):372–387, 2015. (Cited on page 12.)

[15] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021. (Cited on page 24.)

[16] B. Girod. What's wrong with mean-squared error? *Digital images and human vision*, pages 207–220, 1993. (Cited on page 9.)

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. (Cited on page 17.)

[18] W. Han, S. Chang, D. Liu, M. Yu, M. Witbrock, and T. S. Huang. Image super-resolution via dual-state recurrent networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1654–1663, 2018. (Cited on page 7.)

[19] M. Haris, G. Shakhnarovich, and N. Ukita. Deep back-projection networks for super-resolution. In *Proceedings of the IEEE conference on*

computer vision and pattern recognition, pages 1664–1673, 2018. (Cited on page 7.)

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. (Cited on page 7.)

[21] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. (Cited on page 10.)

[22] T. Kaneko and T. Harada. Blur, noise, and compression robust generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13579–13589, 2021. (Cited on page 8.)

[23] M. Kettunen, E. Härkönen, and J. Lehtinen. E-lpips: robust perceptual image similarity via random transformation ensembles. *arXiv preprint arXiv:1906.03973*, 2019. (Cited on page 10.)

[24] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016. (Cited on page 7.)

[25] J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1637–1645, 2016. (Cited on page 7.)

[26] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 624–632, 2017. (Cited on page 7.)

[27] E. C. Larson and D. M. Chandler. Most apparent distortion: full-reference image quality assessment and the role of strategy. *Journal of electronic imaging*, 19(1):011006–011006, 2010. (Cited on page 12.)

[28] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, 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, pages 4681–4690, 2017. (Cited on pages 7 and 19.)

[29] J. Li, F. Fang, K. Mei, and G. Zhang. Multi-scale residual network for image super-resolution. In *Proceedings of the European conference on computer vision (ECCV)*, pages 517–532, 2018. (Cited on page 7.)

[30] Z. Li, C. Bampis, J. Novak, A. Aaron, K. Swanson, A. Moorthy, and J. Cock. Vmaf: The journey continues. *Netflix Technology Blog*, 25:1, 2018. (Cited on page 12.)

[31] Z. Li, J. Yang, Z. Liu, X. Yang, G. Jeon, and W. Wu. Feedback network for image super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3867–3876, 2019. (Cited on page 7.)

[32] J. Liu, W. Zhang, Y. Tang, J. Tang, and G. Wu. Residual feature aggregation network for image super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2359–2368, 2020. (Cited on page 7.)

[33] A. Lugmayr, M. Danelljan, L. Van Gool, and R. Timofte. 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*, pages 715–732. Springer, 2020. (Cited on page 7.)

[34] X. Luo, Y. Xie, Y. Zhang, Y. Qu, C. Li, and Y. Fu. 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*, pages 272–289. Springer, 2020. (Cited on page 7.)

[35] C. Ma, Y. Rao, Y. Cheng, C. Chen, J. Lu, and J. Zhou. Structure-preserving super resolution with gradient guidance. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7769–7778, 2020. (Cited on page 7.)

[36] D. Ma, F. Zhang, and D. R. Bull. Bvi-dvc: A training database for deep video compression. *IEEE Transactions on Multimedia*, 24:3847–3858, 2021. (Cited on page 20.)

[37] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, Georgia, USA, 2013. (Cited on page 19.)

[38] P. C. Madhusudana, N. Birkbeck, Y. Wang, B. Adsumilli, and A. C. Bovik. Image quality assessment using contrastive learning. *IEEE Transactions on Image Processing*, 31:4149–4161, 2022. (Cited on page 11.)

[39] F. Mameli, M. Bertini, L. Galteri, and A. Del Bimbo. Image and video restoration and compression artefact removal using a nogan approach. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4539–4541, 2020. (Cited on page 7.)

[40] Y. Mei, Y. Fan, Y. Zhou, L. Huang, T. S. Huang, and H. Shi. 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*, pages 5690–5699, 2020. (Cited on page 7.)

[41] A. Mittal, A. K. Moorthy, and A. C. Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on image processing*, 21(12):4695–4708, 2012. (Cited on page 11.)

[42] A. Mittal, R. Soundararajan, and A. C. Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012. (Cited on page 11.)

[43] N. Murray, L. Marchesotti, and F. Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2408–2415. IEEE, 2012. (Cited on page 12.)

[44] B. Niu, W. Wen, W. Ren, X. Zhang, L. Yang, S. Wang, K. Zhang, X. Cao, and H. Shen. 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*, pages 191–207. Springer, 2020. (Cited on page 7.)

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner,

L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. (Cited on page 25.)

[46] Y. Patel, S. Appalaraju, and R. Manmatha. Saliency driven perceptual image compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 227–236, 2021. (Cited on page 10.)

[47] N. Ponomarenko, O. Ieremeiev, V. Lukin, K. Egiazarian, L. Jin, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, et al. Color image database tid2013: Peculiarities and preliminary results. In *European workshop on visual information processing (EUVIP)*, pages 106–111. IEEE, 2013. (Cited on page 12.)

[48] N. Ponomarenko, V. Lukin, A. Zelensky, K. Egiazarian, M. Carli, and F. Battisti. Tid2008-a database for evaluation of full-reference visual quality assessment metrics. *Advances of modern radioelectronics*, 10(4):30–45, 2009. (Cited on page 12.)

[49] R. Pourreza, A. Ghodrati, and A. Habibian. Recognizing compressed videos: Challenges and promises. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. (Cited on page 8.)

[50] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. (Cited on page 19.)

[51] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. (Cited on page 15.)

[52] D. L. Ruderman. The statistics of natural images. *Network: computation in neural systems*, 5(4):517, 1994. (Cited on page 11.)

[53] M. S. Sajjadi, B. Scholkopf, and M. Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *Proceedings of the IEEE international conference on computer vision*, pages 4491–4500, 2017. (Cited on page 7.)

[54] H. R. Sheikh, M. F. Sabir, and A. C. Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Transactions on image processing*, 15(11):3440–3451, 2006. (Cited on page 12.)

[55] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. 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*, pages 1874–1883, 2016. (Cited on page 7.)

[56] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. (Cited on page 19.)

[57] L. Stäcker, J. Fei, P. Heidenreich, F. Bonarens, J. Rambach, D. Stricker, and C. Stiller. Deployment of deep neural networks for object detection on edge ai devices with runtime optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1015–1022, 2021. (Cited on page 26.)

[58] Y. Tai, J. Yang, X. Liu, and C. Xu. Memnet: A persistent memory network for image restoration. In *Proceedings of the IEEE international conference on computer vision*, pages 4539–4547, 2017. (Cited on page 7.)

[59] T. Tong, G. Li, X. Liu, and Q. Gao. Image super-resolution using dense skip connections. In *Proceedings of the IEEE international conference on computer vision*, pages 4799–4807, 2017. (Cited on page 7.)

[60] F. Vaccaro, M. Bertini, T. Uricchio, and A. Del Bimbo. Fast video visual quality and resolution improvement using sr-unet. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1221–1229, 2021. (Cited on page 17.)

[61] N. Venkatanath, D. Praneeth, M. C. Bh, S. S. Channappayya, and S. S. Medasani. Blind image quality evaluation using perception based features. In *2015 twenty first national conference on communications (NCC)*, pages 1–6. IEEE, 2015. (Cited on page 11.)

[62] T. Wang, J. He, S. Xiong, P. Karn, and X. He. Visual perception enhancement for hevc compressed video using a generative adversarial network. In *2020 International Conference on UK-China Emerging Technologies (UCET)*, pages 1–4. IEEE, 2020. (Cited on page 8.)

[63] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018. (Cited on page 7.)

[64] Z. Wang and A. C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009. (Cited on page 9.)

[65] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. (Cited on page 9.)

[66] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. (Cited on page 9.)

[67] S. Yu, B. Chen, Y. Xu, W. Chen, Z. Chen, and T. Zhao. Hevc compression artifact reduction with generative adversarial networks. In *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6. IEEE, 2019. (Cited on page 8.)

[68] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. (Cited on page 10.)

[69] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 286–301, 2018. (Cited on page 7.)

[70] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2472–2481, 2018. (Cited on page 7.)