Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Luis Alfonso Amezcua Aragón

Real-time neural network based video super-resolution as a service:

design and implementation of a real-time video super-resolution service using public cloud services

Master's Thesis
Espoo, January 5, 2022

Supervisor: Dr. Matti Siekkinen
Advisor: Dr. Matti Siekkinen

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author: Luis Alfonso Amezcua Aragón

Title:
Real-time neural network based video super-resolution as a service: design and implementation of a real-time video super-resolution service using public cloud services

Date:	January 5, 2022	**Pages:**	62
Major:	Computer Science	**Code:**	SCI3042
Supervisor:	Dr. Matti Siekkinen		
Advisor:	Dr. Matti Siekkinen		

Despite the advancements in video streaming, we still find limitations when there is the necessity to stream real-time video in a higher resolution (e.g., in super-resolution) through mobile devices with limited resources. This thesis work aims to give an option to address this challenge through a cloud service.

There were two main code components to create this service. The first component was aiortc (e.g., the WebRTC python version), the streaming protocol. The second component was the Efficient Sub-Pixel Convolutional Neural Network (ESPCN)-model, one of the outstanding methods to upscale video at the present time. These two code components were implemented in a virtual machine in the Microsoft Azure cloud environment with a customized configuration.

Qualitative as well as quantitative results of this work were obtained and analysed. To obtain the qualitative results two versions of the ESPCN-model were developed and for the quantitative outcomes three different configurations of HW/SW codecs and CPU/GPU utilisation were produced and analysed.

Besides finding and defining the code components mentioned before as optimal to create an efficient real-time video super-resolution service based on the cloud, another conclusion of this project is that sending or receiving information (frames) from the CPU to the GPU and vice-versa has a very big negative impact in the efficiency of the whole service. Hence, to limit this CPU-GPU interaction or to only use GPU (e.g., with the NVIDIA Virtual Processing Framework [VPF]) is critical for an efficient service. This issue can be avoided, as the quantitative results show, if a codec that only makes use of the GPU (e.g., a NVIDIA HW codec) is employed. Furthermore, the Azure cloud environment component, enables an efficient execution of the service in diverse mobile devices.

In future, the quality measure of the video super-resolution done by the ESPCN-model is suggested as a next step to do.

Keywords: super-resolution, video, streaming, ESPCN, aiortc, WebRTC, NVIDIA, Video Processing Framework

Language: English

Acknowledgements

I would first like to thank my supervisor Dr. Matti Siekkinen for all his help and advice with this Thesis. I would also like to thank Teemu Kämäräinen for his important support with the technical aspects of the project implementation.

I am also grateful to Aalto Scientific Computing for the valuable guidance in the development of the project implementation. I would like to offer my special thanks to Simo Tuomisto, Richard Darst and Marijn van Vliet.

Finally, I would like to extend my sincere thanks to Professor Dr. Kari Tammi for his trust and recognition on this work.

Espoo, January 5, 2022

Luis Alfonso Amezcua Aragón

Abbreviations and Acronyms

ACS	Adaptive Cropping Strategy
API	Application Programming Interface
BN	Batch Normalisation
CCA	Contrast-aware Channel Attention
CDN	Content Delivery Network
CNF	Context-wise Network Fusion
CNN	Convolutional Neural Network
ESPCN	Efficient Sub-Pixel Convolutional Neural Network
GAN	Generative Adversarial Network
GAP	Global Average Pooling
HR	High-Resolution
HSI	Hyper-Spectral Image
HVS	Human Visual System
IMDB	Information Multi-Distillation Block
IMDN	Information Multi-Distillation Network
IQA	Image Quality Assessment
ISP	Image Signal Processor
LR	Low-Resolution
MSE	Mean Squared Error
NIST	National Institute of Standards and Technology
ORTC	Object Real-Time Communication
OTT	Over-The-Top
P2P	Peer-to-Peer
PAN	PANchromatic image
PSNR	Peak Signal-Noise Ratio
QoS	Quality-of-Service
ResNet	Residual Neural Network
RTP	Real-Time Transport Protocol
RTSP	Real-Time Streaming Protocol
SISR	Single Image Super-Resolution

SR	Super-Resolution
SSIM	Structural Similarity Index Measure
SSL	Secure Socket Layer
TLS	Transport Layer Security
VGG	Visual Geometry Group
VM	Virtual Machine
VPC	Virtual Private Cloud
VPF	Video Processing Framework
VPN	Virtual Private Network
WebRTC	Web Real-Time Communication

Contents

Chapter 1

Introduction

The FIFA World Cup is one of the most popular sporting events in the world, and spectators were given the opportunity to watch games in 3D for the first time in summer 2010. But, more crucially, the games were viewed by many more people than they had ever been before. Fans were able to watch games in high-definition or near-HD quality on their laptops, cellphones, and other connected devices, including their televisions, thanks to advancements in video streaming technology and increased broadband Internet access usage.

However, sporting events are not the only sort of entertainment that attracts internet viewers. Many providers have been making their regular and premium content, such as news, programmes, shows, and movies, available on their websites for some time now. Despite the fact that many of these Web sites had regional limits, customers perceived an increase in the amount of material available to them. Some content providers and TV channels don't place limits on viewer geography, and as a result, they've gone from being a regional TV source to a worldwide one, resulting in an unexpected increase in ad income.

Against the assault of the Internet, the appeal of watching traditional broadcast TV content is dwindling by the day. Consumers may access Web content from a number of devices (e.g., mobile devices with limited resources) in a variety of locations connected through various sorts of access networks, not simply from a TV in the living room. Adaptive streaming systems that can handle the problems of this diversity as well as the scalability of content delivery to big audiences have sped-up this trend change, which has disrupted established business models and provided new income prospects [6]. If we add to this reality, the additional necessity that sometimes exists to stream video in a higher resolution in real-time, a new disruptor requires our attention.

1.1 Problem statement

This thesis work presents an option to attend the disruption that has just been mentioned through a real-time service that receives an input video stream at a certain resolution and streams out the same video in higher resolution. The super-resolution (SR) is done by a neural network.

A real-time service is a software program that must respond to external sources in a specific time interval. This kind of services must be highly available, otherwise it is unable to satisfy the required deadline. It is also critical that the response is reliable; under other conditions, it might produce the wrong action to be taken, producing also the lost of the deadline [12]. In order to meet these real-time requirements, a cloud virtual machine (VM) was utilised and the code that does the super-resolution was carefully customised to make the best use of the available VM hardware (e.g., an NVIDIA GPU). Moreover, a robust communication protocol was also chosen (e.g., Real-Time Transport Protocol [RTP]) to connect the VM, the server-side, with the client-side. A real-time service is actually one in which performance criteria are a crucial element of its specification, to the point that the service is judged to have failed if certain performance criteria are not satisfied. Each memory allocation and deallocation request takes a certain time to be completed, which can be harmful to performance if not properly handled. While inefficient memory management can degrade the performance of the service, smart memory management during run-time can result in considerable performance gains [14]. This memory management was carefully managed in the CPU to GPU and vice-versa information interchange for the service implementation.

The approach described in the previous paragraph aims to be the basis to answer three research questions in this work. First, what are the components of an efficient real-time video super-resolution service based on the cloud? Second, how can these components be combined to implement a service with a good performance? And third, which are the characteristics of these components?

1.2 Structure of the Thesis

The rest of this thesis is organized as follows. Chapter 2 begins with the definition of super-resolution and the explanation of its most relevant components such as frameworks, upsampling methods, network design and learning strategies. Then, some of the most recent state-of-the-art techniques in super-resolution research are discussed. Among the models that are presented, the

one that was used in this work, the Efficient Sub-Pixel Convolutional Neural Network (ESPCN)-model [19], is explained in detail. At the end of this chapter, the other two important elements of the real-time video super-resolution service are addressed, the cloud computing and the streaming protocol components. The cloud computing element allows the use of a huge capacity of processing resources by the service without global boundaries. Here the virtualization subsection is of special interest due to the fact that a VM is used in the implementation of the service. In the streaming protocols section, a differentiation between real-time and on-demand streaming protocols is made. Any of these two protocols could have been used for the real-time video super-resolution service though the focus in this work is on real-time protocol.

Chapter 3 in first place shows the architecture of the real-time video super-resolution service and in second place describes this service. In the description, the first code component aiortc which is an open-source Python library that implements Web Real-Time Communication (WebRTC) and ObjectReal-Time Communication (ORTC) is introduced [1] and the manner that this component interacts with the ESPCN-model, the second code component, is clarified. The tailored configuration of the Microsoft Azure cloud environment together with some security and network set-up is mentioned here too. At the end of this chapter the steps to execute the real-time video super-resolution service are provided.

Chapters 4 and 5 disclose the qualitative and quantitative results and analysis of this thesis work. To obtain the qualitative results two versions of the ESPCN-model were developed and for the quantitative results and its analysis three different configurations of SW/HW codecs and CPU/GPU utilisation were tested. In a first modified ESPCN-model version, all the original CPU-executable code functions that had a GPU (Tensor)-executable equivalent were accordingly substituted. In a second modified ESPCN-model version, an NVIDIA GPU-executable code framework, Video Processing Framework (VPF), was employed. These two ESPCN-model versions were combined with SW/HW codecs to create the three different configurations. Finally, in chapter 6 the conclusions of this project are summarized.

Chapter 2

Background

2.1 Super-resolution

This section discusses the work of [24]. In that paper, the authors begin mentioning that the term image SR refers to the process of recovering high-resolution (HR) images from low-resolution (LR) images. SR is a significant class of image processing methods in computer vision and image processing. While many of the practical uses of this technology are derived from the actual world, a significant number of practical applications stem from medical imaging, surveillance, and security, among other applications.

In recent times, deep learning models have been extensively investigated and frequently attain state-of-the-art performance via the fast growth of deep learning approaches. A range of deep learning techniques have been utilised to solve SR challenges, ranging from early Convolutional Neural Networks (CNN)-based techniques to more current potential SR techniques based on Generative Adversarial Networks (GANs). Various forms of network architectures, various kinds of loss functions, and different types of learning principles and tactics distinguish the family of SR algorithms employing deep learning methods.

The goal of image super-resolution is to retrieve the associated HR images from the LR images. The LR images are modelled as the output of a degradation, and most works directly model the degradation as a single down-sampling operation. The most frequently employed down-sampling technique is bi-cubic interpolation with anti-aliasing.

For image super-resolution, there are now a number of accessible datasets; each with its own set of image quantities, quality, resolution, and diversity. A selection of image datasets are contained in Table 2.1, which are often utilised by the SR community. In addition to these datasets, several others often

used for further vision tasks, including ImageNet, MS-COCO, VOC2012, and CelebA, are also utilised for SR. It is also common to combine various datasets for training.

Table 2.1: Super-resolution public image datasets. [24]

Dataset	Amount	Avg. Resolution	Avg. Pixels	Format	Category Keywords
BSDS300	300	(435.367)	154,401	JPG	animal, building, food, landscape, people, plant, etc.
BSDS500	500	(432,370)	154,401	JPG	animal, building, food, landscape, people, plant, etc.
DIV2K	1000	(1972,1437)	2,793,250	PNG	environment, flora, fauna, handmade object, people, scenery, etc.
General-100	100	(435,381)	181,108	BMP	animal, daily necessity, food, people, plant, texture, etc.
L20	20	(3843,2870)	11,577,492	PNG	animal, building, landscape, people, plant, etc.
Manga109	109	(826,1169)	966,011	PNG	manga volume
OutdoorScene	10624	(553,440)	249,593	PNG	animal, building, grass, mountain, plant, sky, water
PIRM	200	(617,482)	292,021	PNG	environments, flora, natural scenery, objects, people, etc.
Set5	5	(313,336)	113,491	PNG	baby, bird, butterfly, head, woman
Set14	14	(492,446)	230,203	PNG	humans, animals, insects, flowers, vegetables, comic, slides, etc.
T91	91	(264,204)	58,853	PNG	car, flower, fruit, human face, etc.
Urban100	100	(984,797)	774,14	PNG	architecture, city, structure, urban, etc.

Image quality is a term that relates to the visual characteristics of pictures and focusses on the perceptual evaluations of viewers. Subjective approaches based on human perception (i.e., the perceived realism of the image) and objective computational approaches are employed in image quality assessment (IQA).

One of the most frequent quality measurements for reconstruction of lossy transformation is the peak signal-noise ratio (PSNR). To determine PSNR, the greatest pixel value (denoted as L) and the mean squared error (MSE) between images are used. PSNR is the most often utilised assessment criterion for SR models due to the need for comparison in literature research and the absence of entirely reliable perceptual measures.

The structural similarity index (SSIM) is presented as a means of quantifying the structural similarity of images, given that the human visual system (HVS) is highly specialised for processing picture structures. Given that the SSIM measures reconstruction quality from the standpoint of the HVS, it better fits perceptual evaluation standards and is thus frequently employed.

Due to the fact that SR models may often assist with other visual tasks,

measuring reconstruction performance via other operations is another popular method in IQA. In particular, scientists input the original and rebuilt images into trained models by comparing the effects of the prediction performance to assess the reconstruction quality. In the assessment, object recognition, face recognition, face alignment and parsing, and other vision tasks are employed.

The YCbCr colour space, in addition to the generally known RGB colour space, is often used for SR. At the moment, there is no universally acknowledged standard for executing or assessing super-resolution for a particular space. Thus, it is important to bear in mind that the assessment findings might vary substantially when working (training or assessment) in dissimilar colour spaces or channels.

The following subsections (2.1.1 - 2.1.5) introduce the main concepts around supervised super-resolution and the subsection 2.1.6 presents an overview of unsupervised super-resolution and domain-specific applications according to the research from [24]. In that study, the authors mention that researchers have presented a number of super-resolution models using deep learning in recent years. The main area of these models is supervised SR, i.e., both LR and related HR images are trained. Basically, various configurations of a variety of components comprise these models. Some examples of these components include up-sampling techniques, model architectures, network architecture, and learning methodologies. These components are utilized by researchers to develop an integrated SR model for particular objectives.

2.1.1 Upsampling Methods

To successfully up-sample, it is important to know how to do it. Using CNNs to learn end-to-end up-sampling has increasingly become a trend despite the fact that there have been several conventional up-sampling approaches. Image interpolation, often known as image scaling, is a technique for resizing digital pictures that is frequently employed in image-related applications. The most conventional interpolation techniques are nearest-neighbour interpolation, bi-linear and bi-cubic interpolation, and Sinc and Lanczos resampling. However, the common side-effects of interpolation-based up-sampling algorithms include computational complexity, noise amplification, and blurring outcomes. As a result, the current tendency is to use learnable up-sampling layers (e.g., deep neural networks) instead of interpolation-based approaches.

Transposed convolution layers and sub-pixel layers are inserted into the SR field to overcome the drawbacks of interpolation-based approaches and learn up-sampling in an end-to-end way. The inverse of a normal convolution

is a transposed convolution layer, also known as a deconvolution layer. The intent of a transposed convolution layer is to forecast the probable input by utilizing features of the convolution output, as Figure 2.1 shows.

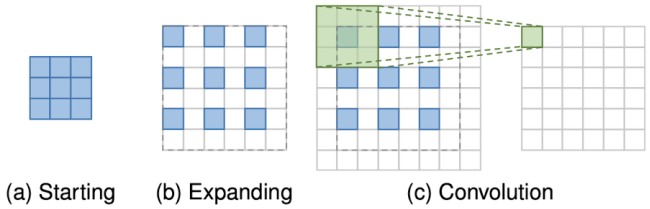

Figure 2.1: Transposed convolution layer. The input is represented by blue boxes, while the kernel and convolution result are represented by green boxes. [24]

Another end-to-end learnable up-sampling layer, the sub-pixel layer, achieves up-sampling by convolutionally creating a number of channels and then reshaping them, as shown in Figure 2.2. The sub-pixel layer has a larger receptive field, allowing it to supply contextual information in order to create more realistic details.

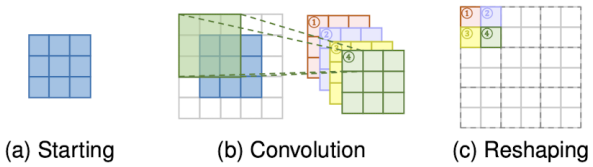

Figure 2.2: Sub-pixel layer. The input is represented by the blue boxes, while the boxes with various colours represent different convolution procedures and output feature maps. [24]

Finally, the meta-upscale module technique employs metalearning to solve the SR of variable scaling factors, as Figure 2.3 shows. This differs from the prior methods, which require a predefining of the scaling factors, i.e., training distinct up-sampling modules for various factors, which is not efficient and does not meet real-world demands.

The previously mentioned learning-based layers are now the most often utilised up-sampling techniques. These layers are often employed in the last up-sampling step, specially for the post up-sampling architecture, to rebuild

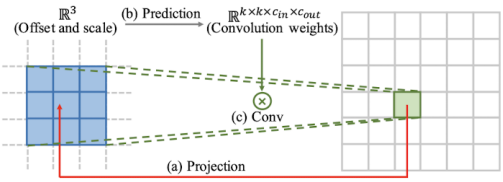

Figure 2.3: Meta upscale module. The projection patch is shown by blue boxes, while the convolution process with expected weights is represented by green boxes and lines. [24]

HR images using high-level representations derived in low-dimensional space. This achieves end-to-end SR while mitigating excessive operations in high-dimensional space.

2.1.2 Super-resolution Architectures

In a SR model architecture, the central challenge is to identify a means for accomplishing up-sampling (i.e., how to obtain HR output from LR input). While the topologies of current models vary significantly, they may be classified into four model architectures (as seen in Figure 2.4) based on the up-sampling techniques used and their position within the model.

A simple method to address this (e.g., the pre-upsampling SR architecture) is to employ classic up-sampling methods to create higher-resolution images, which are subsequently refined using deep neural networks. Generally, traditional approaches (e.g., bi-cubic interpolation) are applied to up-sample the LR images to rough HR images of the necessary size, after which deep CNNs are utilized to recreate high-quality details. This architecture has increasingly gained popularity, and the primary distinctions between similar models are in the posterior model design and learning procedures. Nevertheless, since the specified up-sampling often has undesirable side effects (e.g., noise amplification and blurring) and since the majority of tasks are conducted in high-dimensional space, the time and space costs are much greater than for other architectures.

It has been suggested by some researchers to replace predetermined up-sampling with end-to-end learnable layers that are incorporated at the final point of the models, with the goal of increasing computing efficiency and maximising the usage of deep learning technologies for higher resolution. Thus, the computation and spatial difficulty are greatly decreased since the feature extraction step, which has a high computational cost, happens only

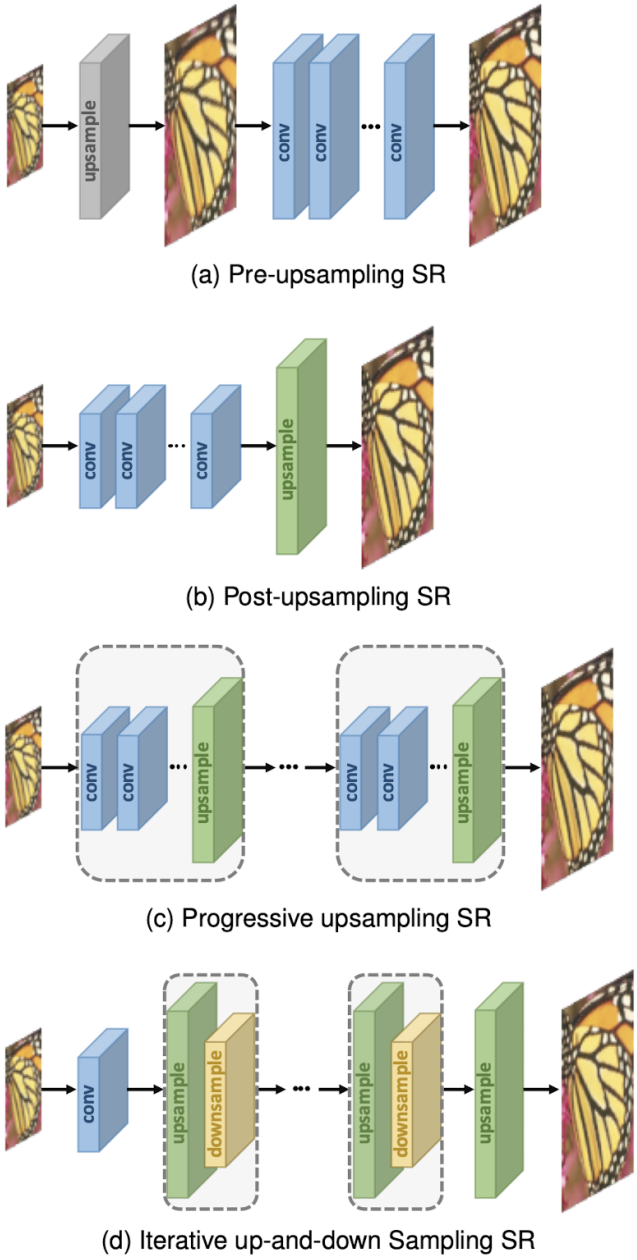

Figure 2.4: Deep learning-based super-resolution model architectures. The output size is represented by the cube size. Predefined up-sampling is represented by grey, whereas learnable up-sampling, down-sampling, and convolutional layers are represented by green, yellow, and blue, respectively. Stackable modules are the blocks contained by dashed boxes. [24]

in low-dimensional space and the resolution grows only at the end. This deep learning architecture has been a popular one and is now one of the most widespread architectures around.

Although the post-upsampling SR architecture significantly reduces computing costs, it still has certain drawbacks. On the one hand, due to the fact that the up-sampling is conducted in a single step, the learning complexity for high scaling factors is significantly increased. On the other hand, each scaling factor necessitates the training of a separate SR model, which is insufficient to handle the requirements for multi-scale SR. To address this issue, models based on a cascade of CNNs that gradually recreate higher-resolution images have been developed under a different architecture (e.g., the progressive upsampling SR architecture). By separating a complex job into easier tasks, the models in this architecture considerably minimise learning difficulties, particularly with large variables, and also manage multi-scale SR without incurring excessive spatial and temporal costs. Nevertheless, these models have several limitations, such as difficult model design for several stages and training stability, and further modelling guidelines and sophisticated training procedures are required.

To better represent the mutual reliance of LR-HR picture pairings, SR incorporates an efficient iterative process called back-projection. This SR architecture, known as iterative up-and-down sampling SR, attempts to iteratively perform back-projection refinement, which entails calculating the reconstruction error and then fusing it back to optimise the HR image intensity. However, the design parameters for the back-projection modules remain still uncertain.

2.1.3 Network Design

One of the most essential aspects of deep learning is the network design. To build the final networks in the super-resolution scene, researchers employ a variety of network design strategies on top of the SR architectures (See Figure 2.5).

SR models have made extensive use of residual learning as part of these strategies. These residual learning strategies are divided into two types: global and local residual learning. Instead of learning a difficult transformation from one full image to another, global residual learning merely involves learning a residual map to restore the missing high-frequency features. Due to the fact that the residuals in the majority of areas are near zero, the model complexity and learning difficulty are significantly decreased. Local residual learning is used to address the degradation issue caused by ever-increasing network depths, to decrease the training complexity, and to enhance the

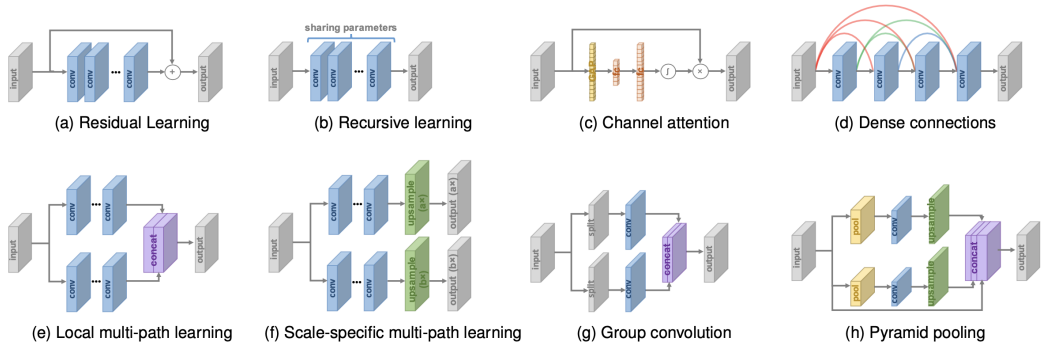

Figure 2.5: Network design strategies. [24]

learning ability. Both approaches utilize shortcut connections (e.g., typically scaled by a tiny constant) and element-wise addition. The distinction between these is that the former links the input and output pictures directly whilst the latter generally adds many shortcuts across layers of varying depths inside the network.

Recursive learning, which involves applying the same modules numerous times in a recursive way, was brought into the SR field to learn higher-level features without adding excessive parameters. Although recursive learning is capable of learning increasingly sophisticated representations without adding extra parameters, it cannot escape large computing costs. Multi-path learning is the process of sending information down many pathways; each of which performs a separate function, and then it fuses them back together to improve the modelling skills. Multi-path learning may be split into three categories: global, local, and scale-specific. Utilizing numerous pathways to extract the characteristics of various elements of the pictures is referred to as global multi-path learning. In their propagation, these routes might cross one another, considerably enhancing the learning capacity.

In the Dense Connections network design strategy, the feature maps of all previous layers are utilised as inputs for each layer in a dense block, and its own feature maps are utilised as inputs for all following layers. The dense connections do not only assist to decrease gradients, improve signal propagation, and promote reuse of features but also to decrease model size greatly by applying a low growth rate (i.e., the number of channels in dense blocks); after all the input characteristics, the channels are then combined. The Channel Attention network design strategy incorporates the Global Average Pooling (GAP) to flatten each input channel descriptor into a constant (i.e., a value that holds true for all channels) and passes these constants into two

dense layers to create channel-wise scaling factors.

Because convolution operations are the foundation of deep neural networks, it has been attempted by researchers to enhance convolution operations for improved performance or efficiency in the Advanced Convolution network design method. One attempt has been group convolution (See [24]). At the cost of a little performance loss, group convolution greatly decreases the number of parameters and operations. The Region-recursive Learning network design method is capable of adaptively customising an optimum search route for each picture based on its unique characteristics, completely exploiting the global intra-dependence of the pictures. While this technique performs somewhat better, the recursive technique necessitates a lengthy propagation channel, which significantly increases the computational cost and training complexity, particularly for super-resolving HR images.

2.1.4 Learning Strategies

Loss functions are utilised in the super-resolution discipline to quantify the reconstruction error and assist model improvement. Arguably,it was common to use the pixel-wise L2 loss in early periods; however later, it was identified that reconstruction effectiveness cannot be precisely measured. A number of loss functions (e.g., content loss, adverse loss) are thus employed to better measure reconstruction errors and to provide more realistic and better outcomes.

Pixel loss quantifies the difference between two images at the pixel level and is mostly comprised of L1 loss (i.e., the mean absolute error) and L2 loss (i.e., the mean square error). In reality, the L1 loss outperforms the L2 loss in terms of performance and convergence. Gradually, pixel loss became the most often utilized loss function. Nonetheless, pixel loss does not truly consider the visual qualities of a picture (e.g., the perceptual quality and textures); hence, the outputs often miss higher-frequency features and result in images that seem uninteresting and too smooth.

Content loss uses a pre-trained image classification network to evaluate semantic differences between the images. Instead of pushing images to match pixels perfectly, the content loss promotes the output image to be perceptually close to the target image.

Due to their remarkable capacity to learn, GANs have gained increasing interest in recent years and have been applied to a variety of visual applications. The GAN is composed of a generator that generates results (e.g., text creation and image modification) and a discriminator that accepts as input the created results and instances sampled from the target distribution and determines if each input is from the target distribution. During the GAN

training, two stages are followed alternatively: a) fixing the generator and training the discriminator to discriminate more effectively and b) fixing the discriminator and training the generator to mislead the discriminator. With regard to super-resolution, adverse learning is easy to follow because the SR model must only be treated as a generator, and an additional discriminator must be defined in order to assess whether or not the image is created. Although the SR models with adverse loss and content loss produce a lower PSNR than those with pixel loss, the perceived quality advantages are considerable. The discriminator takes several difficult-to-learn latent patterns from genuine HR images and forces the produced HR images to comply, thereby assisting in the generation of more realistic images. Currently, however, the GAN training is not easy nor reliable.

While doing research, academics typically take into account various types of loss functions employing a weighted average in order to set overall constraints on the whole generation process. But even though there are a number of methods for measuring loss, their weights need empirical research, and identifying approaches to combine them fairly and effectively remains a challenge.

Batch normalisation (BN) has been suggested to decrease the internal covariate shift in deep CNNs in order to speed up and stabilise training. In the calibration of the intermediate feature distribution, as well as in the mitigation of vanishing gradients, the BN is able to apply larger learning rates and make use of less caution during initialisation. However, it is claimed that the BN eliminates the scale information included in each picture, thus omitting the range flexibility inherent in networks (See [24]).

Curriculum learning proposes starting with an easy job and progressively incrementing the complexity. Curriculum training is added to reduce learning difficulties. Curriculum learning significantly decreases the difficulty of training and significantly decreases the overall training time, particularly for large components.

Multi-supervision is the process of incorporating numerous supervision signals into a model to improve gradient propagation and prevent disappearing and bursting gradients. To build an HR picture, each recursive output of the unit is input into a reconstruction module, and the final prediction is constructed by combining all of the intermediate reconstructions. In reality, this multi-supervision strategy is commonly utilized by adding certain terms to the loss functions, and supervision signals are thus more successfully sent backwards, hence reducing the difficulty of training and improving the model training.

2.1.5 Other Improvements

Along with network design and learning strategies, there are other approaches that may be used to improve SR models. Firstly, context-wise network fusion (CNF) involves training discrete SR models with distinct architectures individually, feeding their predictions into individual convolutional layers, and eventually combining their outputs to obtain the final prediction result. In this CNF architecture, the final model generated by three lightweight SRCNNs is similar in performance and efficiency to state-of-the-art models, while utilizing significantly less computational resources.

Secondly, one of the most common methods for improving deep learning performance is data augmentation. Cropping, flipping, scaling, rotating, colour jittering, and other image augmentation techniques are important for image super-resolution.

Thirdly, multi-task learning is a term that refers to the process of enhancing the generalisation capacity by the use of domain-specific knowledge included in training signals for related tasks, such as object identification and semantic segmentation, head posture estimation, and face attribute inference. Due to the fact that different activities often concentrate on distinct elements of the data, the integration of related activities with SR models often enhances SR performance by offering more information and knowledge.

Fourthly, in Network Interpolation, relevant results are obtained with fewer artefacts by changing the interpolation weights without retraining networks. Finally, Self-ensemble, often known as improved prediction, is an inference approach often utilised by SR models.

More and more attention has been given to image super-resolution models based on deep learning in recent years, and they have reached the highest levels of performance. The majority of contemporary state-of-the-art SR models may be traced to a mixture of the strategies mentioned above. Besides SR accuracy, efficiency is another key factor. Some strategies might have a greater influence on efficiency, while others might have a lesser one.

2.1.6 Unsupervised Super-resolution and domain-specific applications

Research to date on super-resolution focuses mostly on supervised learning, that is, learning using matched LR-HR picture pairings. However, since collecting photographs of the same scene at multiple resolutions is challenging, the LR images in SR datasets are often created by conducting preset degradation on HR images. Thus, trained SR models really learn how to reverse the prescribed deterioration process. To determine the real-world

LR-HR mapping without the need for prior manual degradation, researchers are increasingly focusing on unsupervised SR, in which only unpaired LR-HR pictures are employed for training, since the resultant models are more likely to deal with SR difficulties in real-world circumstances.

Some domain-specific applications of image super-resolution are the following: first, depth maps are useful for numerous tasks, such as posture estimation and semantic segmentation because they record the depth (i.e., distance) between the viewpoint and objects in the scene. Nevertheless, due to cost and manufacturing restrictions, sensor-generated depth maps are often low-resolution and subject to degrading effects, such as noise, quantisation, and missing data. As a result, super-resolution is used to improve the spatial resolution of depth maps. Second, hyper-spectral images (HSIs) with hundreds of bands provide rich spectral characteristics and aid numerous visual tasks as compared to panchromatic images (PANs). Nevertheless, collecting high-quality HSIs is significantly more difficult than collecting PANs (i.e., RGB images with three bands) due to technology restrictions, and the resolution is significantly lower. As a result, super-resolution has been brought into this area, and researchers are increasingly utilising a combination of HR PANs and LR HSIs to forecast HR HSIs. Third, when it comes to generating LR training images for SR models, it is normal to downsample RGB photos manually (e.g., by bicubic downsampling). Real-world cameras, on the other hand, collect 12-bit or 14-bit raw pictures and then execute a number of processes (e.g., demosaicing, denoising, and compression) employing camera image signal processors (ISPs) to generate 8-bit RGB pictures. After running the pre-processing and enhancement algorithm, the RGB images have lost much of their original content and are markedly different from the original images captured by the camera. As a result, using the manually downsampled RGB picture directly for SR is unsatisfactory. To address this issue, academics are investigating means to utilise real-world images for SR. Fourth, multiple frames provide substantially more scene information for video super-resolution, and there is not only intra-frame spatial but also inter-frame temporal interdependence (e.g., motions, brightness, and colour changes). Thus, present research focusses on exploiting spatio-temporal interdependence more effectively, including explicit motion compensation (e.g., optical flow-based, learning-based) and recurrent methods, among others. Furthermore, others attempt to learn the motion compensation directly. For instance, the Video-ESPCN (VESPCN) method, covered later in subsection 2.2.4, learns motion compensation based on neighbouring frames utilising a trainable spatial transformer and feeds multiple frames into a spatio-temporal ESPCN for end-to-end prediction. In summary, super-resolution technology has a significant role to play in a wide variety of applications, particularly

when we are capable of handling large objects well but not small ones.

2.2 Super-resolution Research

Super-Resolution is a technique that has been studied for around 20 years. During this time, many techniques have been introduced to address this research area. Some of the latest super-resolution research is presented in this section.

First, the subsection 2.2.1 presents some relevant reasons behind making the deployment of this work in a cloud environment. Second, the subsections 2.2.2 - 2.2.4 describe three different state-of-the-art techniques for image and video super-resolution. One of these models, the ESPCN-model, was employed in the deployment of this work.

2.2.1 Latency and Throughput Characterisation of Convolutional Neural Networks for Mobile Computer Vision

The following subsection discusses the work of [13]. The machine learning arena is progressing continuously. In this regard, specialised hardware, runtimes optimised by vendors, and lightweight inference models are contributing to a continuously increasing inference performance. However, model conversion and portability are still a challenge due to immature tools and frameworks.

There are important latency-throughput trade-offs in CNNs for mobile computer vision, and its behaviour is quite complicated. One set of CNNs are MobileNets, which provide faster inference than state-of-the-art CNNs, such as visual geometry group (VGG) and residual neural network (ResNet). The primary uses of MobileNets are mobile and embedded computing.

Accuracy, inference latency, and system throughput are the most relevant metrics for a computer vision system. The end-to-end inference can be conducted on-device or remotely. In the former case, there is an extra latency due to the model loading and initialisation, and in the latter one, there exists an additional network latency. With reference to system throughput batching, as mentioned in subsection 2.1.4, can be used to improve the total processing time, but increases the latency.

In on-device object recognition and detection, the optimal batch size depends on the inference model and the computation platform, and it is generally calculated with real measurements. From a practical standpoint, the real

performance of a model on a certain hardware depends on many elements. In general, in remote object recognition, the inference is executed at a higher speed mainly when an application needs to employ various neural networks.

Even though the optimisation problem of a CNN inference model run-time performance is reduced using same initial parameters, such as the Graphics Processing Unit (GPU) architecture or the inference model version, its performance estimation is complex because of the different behaviour of computing hardware, optimised run-times, software libraries, and inference models. Furthermore, in an actual implementation, the uncertain essence of a system input further complicates the prediction. Machine learning frameworks are able to automatically optimise their configurations, but many parameters related to, for instance, the placement of a model operation in a Central Processing Unit (CPU) or in a GPU, still require a manual tuning. We faced this dilemma when implementing the SR video service of the present work. Memory limitations and the latency related to the model load and initialisation inhibit the use of only on-device deployments.

2.2.2 Lightweight Image Super-Resolution with Information Multi-distillation Network

The following subsection discusses the work of [15]. In this paper the authors argue that the purpose of single image super-resolution (SISR) is intrinsically ill-posed due to the fact that many HR images may be down-sampled to an identical LR picture. Several image SR approaches based on deep neural networks have been suggested by [15] to solve this challenge and have demonstrated promising results. To improve the quality of produced images, a deeper model would be useful.

The predominant tendency in present SR algorithms is towards increasing the number of convolution layers in order to enhance performance. As a consequence, the majority of them are forced to contend with a high number of model parameters, a large memory footprint, and poor training and testing velocities. These methods are still not suited to be employed with resource-constrained equipment. This limitation was addressed in the previous subsection. Numerous applications, such as video applications, edge devices, and smartphones, demand not just high performance but also fast execution speed. As a result, it is critical to develop a model that is both lightweight and efficient in order to achieve these requirements. This work aims to find a solution for this performance-speed combined goal.

To solve the aforementioned challenges, a lightweight information multi-distillation network (IMDN) was developed in order to better balance per-

formance and applicability. In this network, a complex information multi-distillation block (IMDB) was created. The suggested IMDB extracts granular characteristics that preserve partial information and treat additional characteristics at every stage (layer). To aggregate the numerous refined information acquired in all phases, a contrast-aware channel attention layer was created, geared primarily for the lower-level vision tasks. When it comes to image restoration, aspects that are more helpful (i.e. edges, corners, textures) are better used. The input image should be scaled to the intended size to handle the SR of any arbitrary scale factor utilising a single model. Moreover, a suggested adaptive cropping approach should thus be used to achieve image patch sizes suited to lightweight SR models with down-sampling layers. The graphical representation of this network is presented in Figure 2.6.

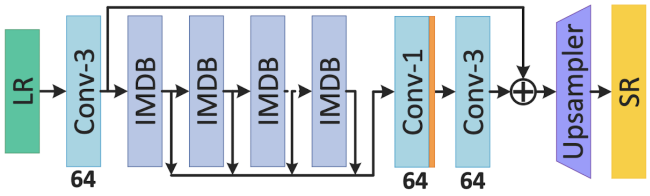

Figure 2.6: Architecture of the information multi-distillation network (IMDN). [15]

The following are the benefits of this method according to [15]. Firstly, it allows rapid and precise super-resolution of images. With a small number of parameters, competitive results are produced using the information multi-distillation block (IMDB) with a contrast-aware channel attention (CCA) layer. Secondly, the adaptive cropping strategy (ACS) enables network-based down-sampling processes (for example, a convolution layer with a stride of 2) to handle images of any size. By employing this approach, the cost of computing, memory, and downtime in the event of treating indefinite SR magnification may be greatly reduced. Thirdly, the experiments were conducted to determine the factors impacting real inference time, and it was discovered that the depth of the network is connected to the execution speed. The authors in [15] suggest that this may serve as a guide for designing a lightweight network. The model strikes an ideal balance between visual quality, quickness of inference, and memory occupation.

2.2.3 Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network

The following subsection discusses the work of [19]. The SR operation is essentially a one-to-many mapping from the LR to the HR space, with several possible solutions. Underlying all SR strategies is the presumption that much of the data has redundancy, so much that it can be derived from the lower frequencies. As a result, SR is an inference challenge.

Multiple approaches presuppose that various images of the same scene with varying viewpoints are usable as LR occurrences. These are classified as multi-image SR algorithms. Typically, these approaches necessitate computationally intensive image registration and fusion phases. SISR techniques are an additional class of methods. These techniques employ the tacit redundancy inherent in natural data in order to retrieve lost human resource data from a single LR occurrence. Typically, this manifests itself as local spatial correlations in images and extra temporal correlations in videos.

SISR techniques are used to recover an HR image from an individual LR input image. Recently, neural network-derived image representations have shown potential for SISR. One advantage of these approaches is that they can learn non-linear mappings, including LR and HR image patches, from the training on massive image databases, such as ImageNet. Other classifiers, such as random forests, have also effectively supported SISR.

The reliability of the algorithms, in particular their computing and memory costs, gains in significance with the growth of CNN. In comparison to previous manual modelling, the versatility of deep network design to learn non-linear connections has proved to be better in its reconstruction accuracy.

It is suggested in this technique to increment the resolution from LR to HR only at the termination point of the network, using a post-upsampling SR architecture as stated in subsection 2.1.2, and to super-resolve HR data from LR attribute maps. This removes the requirement for the majority of SR operations to be performed at the far larger HR resolution. To accomplish this, a more effective sub-pixel convolution layer for learning the super-resolution upscaling of images and videos is advised, as shown in Figure 2.7.

Given tha the upscaling is performed by the final tier of the network, the network receives each LR picture directly, and features are extracted utilizing nonlinear convolutions in LR space. For high definition (HD) videos, the lower resolution and smaller filter are both adequate to process the data while maintaining enough computational and memory complexity to enable super-resolution to be achieved in real time. Since no specific interpolation

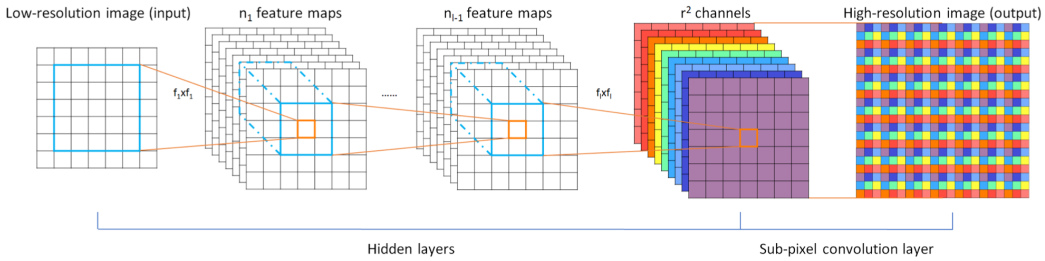

Figure 2.7: The efficient sub-pixel convolutional neural network (ESPCN), contains two convolution layers for extracting feature maps and a sub-pixel convolution layer for aggregating the feature maps from LR space and constructing the SR picture in a single step. [19]

filter is used, the network knows the processing employed for SR implicitly. In comparison to a single fixed filter upscaling at the first layer, the network will learn a stronger and more dynamic LR to HR mapping.

The performance of the network is a significant benefit. This makes it an excellent candidate for SR video, which enables frame-by-frame super-resolution of images. The SR model is faster by a factor of ten times compared with the quickest methods previously released. In comparison to the SRCNN 9-5-5 ImageNet model, the number of convolutions required to super-resolve a single image is $r \times r$ times lower, and the total number of model parameters is 2.5 times smaller. Thus, the super-resolution overall difficulty of the operation is $2.5 \times r \times r$ times smaller.

In conclusion, the function extraction stages are suggested to be performed in the LR space rather than the HR space. For this, a novel sub-pixel convolution layer capable of super-resolving LR data into HR space at a very low computational expense is proposed. This CNN model is the first that is able to create real-time SR HD videos on a single GPU.

2.2.4 Real-Time Video Super-Resolution with Spatio-Temporal Networks and Motion Compensation

The following subsection discusses the work of [8]. That study is an evolution of the ESPCN-model presented in the previous subsection but now focused in videos. In order to transition from images to videos, the authors consider time as another data dimension (which has a strong degree of correlation).

The development of video SR methods have mostly taken place in a similar fashion to the development of image SR methods. Similarly, approaches

to dictionary learning have been adjusted from images to videos. Example-based patch recurrence is another method, which considers that patches in a single picture or video follow multi-scale connections.

Motion estimation and compensation is an important tool for revealing temporal associations. As a result, video SR methods that directly model motion across frames are quite popular. Motion compensation may also be achieved concurrently with the SR job as part of the broader modelling of the down-scaling mechanism. Moreover, recurrent bidirectional networks have been used to investigate joint motion compensation for SR with neural networks.

Due to the absence of appropriate solutions for High Definition (HD) video SR, prior solutions were not capable of successfully using temporal correlations when running in real-time. Although the ESPCN-model makes effective employment of sub-pixel convolution, its naive extension to videos, handling frames separately, struggles to take advantage of inter-frame redundancy and does not implement a temporally compatible outcome. Inferring parameters for a spatial mapping between two images is possible with spatial transformer networks. To achieve a quick and precise video SR algorithm, the efficiency of sub-pixel convolution is merged with the success of spatio-temporal networks and motion compensation in this method. Furthermore, a spatial-transformers-based motion compensation scheme is built, which is coupled with spatial and time models to achieve a rather effective, end-to-end solution for video SR with motion compensation. A high-level diagram of the suggested method is depicted in Figure 2.8.

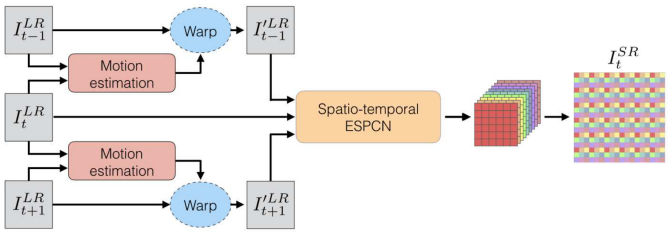

Figure 2.8: Suggested design for video SR. [8]

The starting point of this method is the real-time image SR ESPCN-model. The other modules that are part of the proposed design for video SR are the following: sub-pixel convolution SR, spatio-temporal networks (i.e. early fusion, slow fusion, and 3D convolutions), and spatial transformer motion compensation.

The performance benefits of sub-pixel convolutions are combined with

temporal fusion techniques in [8] to show real-time spatio-temporal SR models. In comparison to independent single frame processing, the employed spatio-temporal models in [8] improve reconstruction accuracy and temporal consistency while also reducing computing complexity. The examined models in [8] are expanded to provide a motion compensation system focused on spatial transformer networks that is both effective and trainable for video SR.

2.3 Cloud Computing

Cloud computing is a relevant element of the video super-resolution service because it makes possible the provisioning of large amounts of computing power that, for instance, are not available in mobile devices to up-scale in real-time the input video stream. Likewise, cloud computing is a borderless and global tool, thus the service can be accessed from any place [4].

Cloud computing was made possible by the fast advancement of processing and storage technology, as well as the success of the Internet. In this approach, resources (such as CPU and storage) are made available to users as general utilities, which can be leased and released over the Internet on-demand.The conventional service provider position in cloud computing is split into two: infrastructure providers that administer cloud platforms and lease resources depending on use, and service providers who rent resources from one or more infrastructure providers to serve end users. The following subsections (2.3.1 - 2.3.3) discuss related topics based on the work of [25].

2.3.1 Overview

The National Institute of Standards and Technology (NIST) supplies a comprehensive definition of cloud computing that encompasses all critical elements of the technology: "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [as quoted in [25]].

Rather than a new technology cloud computing is an operational model that integrates a number of already existing technologies to do business differently. The following technologies are often compared to cloud computing. Grid Computing is a distributed computing approach in which networked resources are coordinated to accomplish a shared computational goal. Utility Computing is an approach for delivering on-demand resources and billing

consumers based on use instead of a fixed fee. Virtualization is a technology for abstracting physical hardware specifics and providing high-level applications with virtualized resources. A VM is a term used to describe a virtualized server. Virtualization technology is the underlying component of cloud computing. Autonomic Computing seeks to create computer systems that can manage themselves, i.e., respond to internal and external observations without the need for human involvement. Virtualization technology is utilised in cloud computing to accomplish the aim of offering computer resources as a utility. Certain elements of it are similar to those of grid computing and autonomous computing, but it differs from them in other features.

2.3.2 Architecture

As shown in Figure 2.9, the architecture of a cloud computing environment is split into four layers: the hardware/data centre layer, the infrastructure layer, the platform layer, and the application layer.

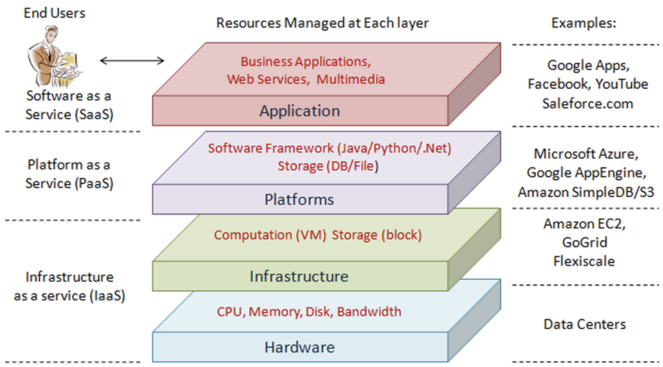

Figure 2.9: Cloud computing architecture. [25]

The hardware layer includes physical servers, switches, routers, and power supplies and is in charge of managing the physical resources of the cloud. In reality, data centres usually implement the hardware layer. Using virtualization technologies, such as VMware, the infrastructure/virtualization layer provides a pool of storage and computational resources. The platform layer is comprised of operating systems and application frameworks. Its goal is to reduce the amount of time and effort required for delivering apps directly into VM containers. The application layer contains the real cloud applications that may benefit from the automatic-scaling functionality in order to gain higher performance, greater availability, and lower operating costs.

Cloud computing has a more modular design. This is comparable to how the OSI model for network protocols is designed. The architectural flexibility of cloud computing enables it to accommodate a broad variety of application needs while lowering administration and maintenance costs.

Clouds provide services that may be classified into three types. Firstly, Infrastructure as a Service (IaaS) manages the supply of infrastructure resources on demand, typically in the form of VMs. Secondly, Platform as a Service (PaaS) addresses the provision of platform-layer resources, such as operating system support and frameworks for software development. Thirdly, Software as a Service (SaaS) relates to the delivery of on-demand applications via the Internet. These three types represent the cloud computing business model, which is depicted in Figure 2.10.

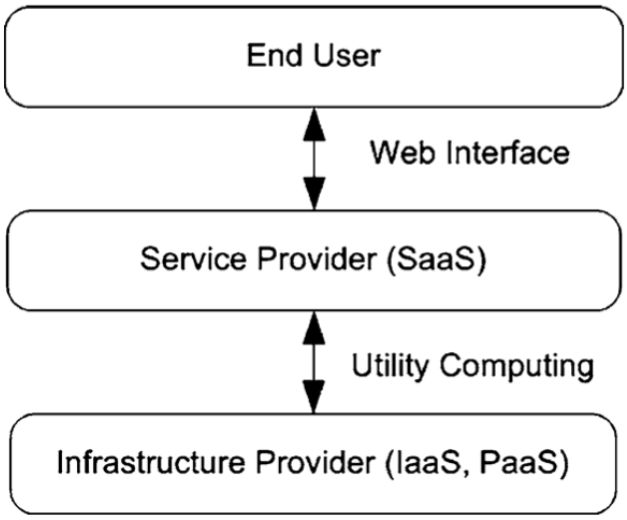

Figure 2.10: Business model of cloud computing. [25]

According to the objective, when moving an application to the cloud environment, there are distinct kinds of clouds to consider, each with its own set of advantages and disadvantages. A Public Cloud is a cloud in which service providers make their resources available to the broader public as a service. A Private Cloud / Internal Cloud is intended for usage only by a particular company. A Hybrid Cloud is a mix of public and private cloud models in which part of the service infrastructure is hosted in private clouds and the rest is hosted in public clouds. A Virtual Private Cloud (VPC) is a platform built on top of public clouds. A VPC makes use of virtual private

network (VPN) technology, which enables service providers to create their own topology and security configurations, such as firewall rules.

2.3.3 Characteristics

The following are some of the key differences between cloud computing and conventional service computing. Multi-tenancy is a co-location of services owned by different providers in the same data centre. The service providers and the infrastructure provider share responsibility for the performance and administration of these services. In shared resource pooling, the infrastructure provider makes available a pool of computing resources that may be dynamically allocated to various resource users. Geo-distribution and ubiquitous network access refers to clouds which are widely accessible through the Internet and utilise it as a service delivery network. As a result, cloud services may be accessed from any device with an Internet connection. Furthermore, many of the clouds today are made up of data centres spread throughout the world. Service oriented is a type of cloud computing which is based on a service-oriented operating model. As a result, it puts a high value on service management. In dynamic resource provisioning, computing resources may be acquired and released on an adaptive form. In this provisioning, service providers may obtain resources depending on current demand via dynamic resource provisioning. In self-organizing, service providers have the ability to control their resource usage based on their own requirements. In utility-based pricing, cloud computing pricing is based on a pay-per-use basis. By charging consumers on a per-use basis, utility-based pricing reduces service running costs.

2.3.4 Virtualization

This subsection presents the definition of a VM as stated by [20]. When a device or resource, such as a server, storage equipment, network or operating system, is virtualized, the virtualization framework separates the resource into one or more execution environments, and the virtualization framework creates a virtual instance of the device or resource. A VM is a virtual instance of a physical host. Virtualization refers to the process of deploying and managing virtual instances. A hypervisor or virtual machine monitor is the software or firmware that builds a virtual machine on the host hardware. The supply of consolidated VM instances according to the needs of computing clients is an essential component of virtualization in cloud computing. Multiple VMs may be hosted on the same physical host, allowing dynamic multiplexing of computing and communication resources and increasing the

resource usage of the physical infrastructure and its scalability. In cloud data center settings, VM consolidation is a common practice.

Creating a virtual hard drive image, configuring virtualized resources, installing the operating system, and initialising application services are all steps in the VM creation process. After being generated from scratch or copied from templates, each VM is immediately paused. Each paused VM is a static instance that takes up disc space; during the VM initialization phase, it may be changed by installing new programmes or altering the related settings. A VM is a virtual instance that must be scheduled to allocate computing resources on a physical host. When a real CPU is assigned to a virtual machine, it enters the operating state. On a local host, a VM may be halted or suspended, or it can be transferred to another host. The state transition diagram of the VM lifespan is shown in Figure 2.11.

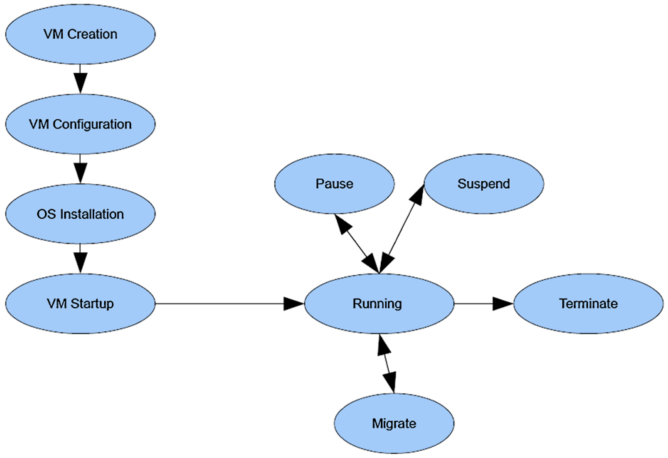

Figure 2.11: Diagram of virtual machine state transitions. [20]

Template-based VMs, process forking, and VM forking are all methods for deploying virtual machines. To decrease startup latency, service providers use VM templates. The state of a running application is not stored in a template-based image.

2.4 Streaming Protocols

This section on streaming protocols is based on the research of [17] and [6].

Internet video, also known as over-the-top (OTT) services, can be classified into three distinct categories: user-generated content created primarily

by amateurs (such as the content served by YouTube), professionally generated content created by studios and networks to promote their commercial offerings and programming (such as the content served by ABC.com), and direct movie sales to consumers over the Internet (also referred to as electronic sell-through, or EST). Netflix and Apple TV are included in the last category.

Cable and IPTV services are distributed over managed networks because they utilise multicast transmission and require certain quality-of-service (QoS) characteristics. Traditional streaming technologies like Microsoft Windows Media and Apple QuickTime, as well as adaptive streaming technologies like Microsoft Smooth Streaming and Apple HTTP Live Streaming, all run on essentially unmanaged networks. These streaming protocols transmit video to viewers over a unicast connection (from a host or content delivery network [CDN]) using either a proprietary streaming mechanism built on top of an existing transport protocol, usually TCP and rarely UDP, or the standard HTTP protocol over TCP.

Due to its ease, progressive download, which employs HTTP over TCP, has been a popular method for viewing online information in the past. The playout can begin as soon as enough relevant data has been obtained and buffered in a progressive download. Progressive download, on the other hand, lacks the flexibility and rich features of streaming. If there are various options with varying resolutions for the same material, the viewer must pick the most appropriate version before the download begins. If the selected version does not have enough band width, the viewer may encounter frequent freezes and re-buffering. Adaptive streaming overcomes these disadvantages while maintaining the ease of progressive download.

Progressive downloads are used with streaming to provide adaptive streaming. Progressive download, on the one hand, occurs when an adaptive streaming client sends HTTP request messages to an HTTP server to retrieve specific segments of material, and then displays the media as the content is being delivered. These segments, on the other hand, are brief, allowing the client to download only what is required and to employ trick modes more effectively, creating the appearance that the client is streaming. More crucially, short-duration chunks (for example, MPEG4 file fragments) are available at several bitrates, corresponding to various resolutions and quality levels, allowing the client to move between them at each request. Since adaptive streaming makes use of HTTP, it is able to take use of the ubiquitous connection that HTTP provides.

Video streaming is the process of transmitting information, more specifically, a video from a source to one or more destinations. Information may be shared between nodes in a network in a variety of ways. Typically, the kind of

information shared by the system dictates the communication method used. In general, the media streaming protocol is a method for sending real-time media across a network that takes into consideration the packet structure and algorithms. When streaming, there are two types of communication protocols: push-based and pull-based protocols.

The complexity of the server architecture is one of the key distinctions between push-based (real-time) and pull-based (on-demand) streaming. On the one hand, bitrate control is generally a client duty in pull-based streaming, which makes server implementation much easier. Pull-based streaming may also run on top of HTTP. As a result, a standard Web server may provide media content via pull-based streaming with minimum modifications.

On the other hand, push-based streaming requires a dedicated server that supports Real-Time Streaming Protocol (RTSP) or a comparable purpose-built protocol that includes built-in algorithms for bitrate control, re-transmission, and content caching. As a result, pull-based streaming may be more cost effective than push-based streaming.

Regardless of the reduced server expenses, pull-based streaming is typically less efficient than push-based streaming in terms of overhead related to the underlying transport protocol. RTP, the push-based protocol, has a lower transmission overhead than HTTP over TCP. Furthermore, because RTP often operates on top of UDP, RTP lacks the re-transmission dynamics of TCP and congestion management features.

Client buffering is supported by both push- and pull-based streaming protocols, both at the start of a session as well as following trick-mode switches such as fast forward to play. Since the client can start with a lower-bitrate stream in adaptive streaming techniques, the initial client buffering period can be significantly smaller than in non-adaptive streaming methods. This provides for a quicker starting time and improved responsiveness. However, multicast support is one of the major advantages of push-based streaming. Multicast allows servers to deliver a packet just once to a group of clients that are waiting for it.

The use of content caching can improve the efficiency of network routing and packet delivery. Pull-based adaptive streaming, where each fragment may be stored in the network individually, is most likely the most effective use of caching.

On the one hand, when a server and a client establish a connection in push-based streaming, the server begins to stream packets to the client until the client terminates or interrupts the session. On the other hand, the media client is the active entity that requests content from the media server in pull-based streaming. Thus, whether the server is inactive or blocked for that client, the answer of the server is determined by the requests of the client.

The distinctions between push-based streaming and pull-based streaming (here, pull-based streaming exclusively refers to streaming over HTTP) are summarised in Table 2.2. This data was derived from an evaluation of the transmission protocols for the H.265 encoder. According to [6] the push-based protocols are more efficient than pull-based protocols in terms of overhead.

Table 2.2: Comparison between push-based and pull-based streaming protocols. [17]

Characteristic	Push-based	Pull-based
Source	Broadcasters and servers like Windows media, QuickTime, RealNetworks Helix Cisco CDS/DCM	Web servers such as LAMP, RealNetworks Helix, Microsoft BS, Cisco CDS
Protocols	RTSP, RTP, UDP	HTTP
Bandwidth usage	Likely more efficient	Likely less efficient
Video monitoring	RTCP (RTP transport)	Currently propietary
Multicast support	Yes	No

As a result of the advantage in push-based protocols, firstly RTSP, which is based on RTP, is examined in subsection 2.4.1. RTSP is used to play and pause, for instance, a video transfer over RTP. Secondly, the WebRTC project, which is managed by Google and is based on RTP, is analysed in subsection 2.4.2. This latter system enables real-time communications with a high degree of quality, low latency, and minimal bandwidth usage through certain Application Programming Interfaces (APIs). WebRTC is a standard for media streaming from/to browsers. This was the protocol used in the implementation of this work because the service is called from a web browser. Thirdly, the On-Demand hypertext transfer protocol (HTTP), which could have also been utilised in the implementation of this work, is explained in subsection 2.4.3.

2.4.1 Real-Time Streaming Protocol

RTSP is a non-connection focused application layer protocol that utilises an identifier-based session. RTSP often utilizes the UDP protocol to exchange video and audio data, whereas TCP is utilised for control (i.e., TCP is used only when necessary). The RTSP protocol has a syntax that is similar to that of the HTTP protocol and offers three operations. RTSP is a session-based application layer protocol that does not need a connection. To exchange

video and audio data, RTSP typically utilises the UDP protocol, whereas the control protocol is TCP. The RTSP protocol offers three operations and has a syntax similar to that of the HTTP protocol. Firstly, in the retrieval of media from media server operation, the client might request a presentation description through HTTP or any other approach. The presentation description includes the multicast addresses and ports that will be utilised for the continuous media if the presentation is multicast. However, for security considerations, the client specifies the destination if the presentation is being transmitted exclusively to the client through unicast. Secondly, in the invitation of a media server to a conference operation, a media server might be "invited" to join an existing conference, either to playback media within the presentation or to record all or a subset of the media included inside the presentation. This conference functionality is particularly helpful for delivering educational software. During the conference, several participants may take turns "pressing the remote control buttons". Thirdly, when adding media to an existing presentation operation, especially live presentations, it is beneficial for the server to notify the client when new media becomes available.

The syntax of an RTSP URL is largely similar to that of an HTTP URL, with the exception that RTSP uses the scheme rtsp:// rather than http:// as the HTTP protocol does. However, it also includes additional request methods such as DESCRIBE, SETUP, PLAY, PAUSE, and TEARDOWN. The DESCRIBE method is utilised to obtain information about the presentation or object specified by the RTSP URL. In response to this request, the server returns a description of the requested resource. This answer corresponds to the RTSP startup phase and includes a list of the multimedia streams that are required. The SETUP method, on the other hand, is employed to determine the manner in which the stream can be delivered. The URL of the multimedia stream is contained in the request, as well as a transport specification that typically incorporates a port for receiving RTP data (i.e., video and audio) and another port for receiving RTCP data (i.e., metadata). The server replies by verifying the chosen parameters for the stream and filling in the remaining fields, such as the server-selected ports. Consequently, each stream must be setup prior to submitting a PLAY request. The PLAY request initiates data stream shipping by a server component through the ports specified using the SETUP request method. Additionally, the PAUSE method briefly suspends one or all streams in order to restart them later with a PLAY request. Finally, the TEARDOWN request method terminates data shipping, freeing all resources. In this respect, it is worth noting that a TCP connection is first made between the client and the server, which is initiated by the client and is usually established on the well-known TCP port 554.

The precise request order of the RSTP protocol is depicted in Figure 2.12.

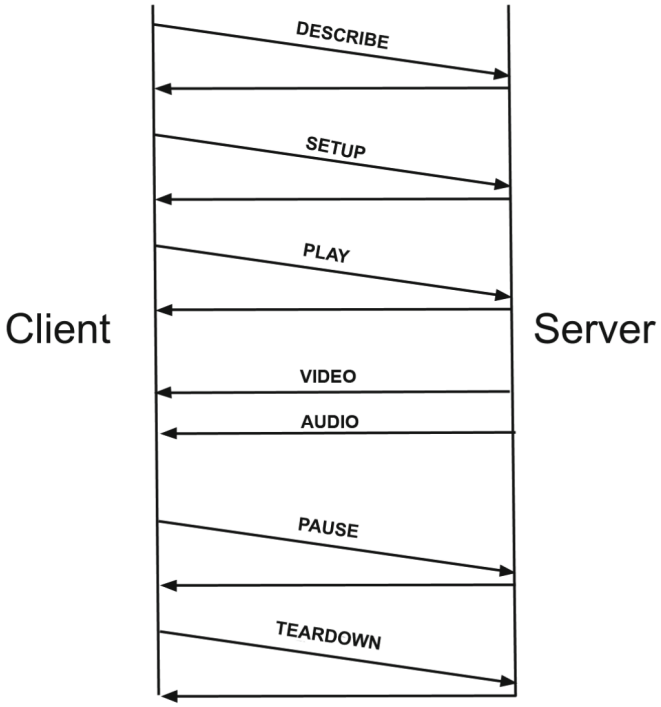

Figure 2.12: RTSP request order. [17]

2.4.2 Web Real-Time Communications Protocol

WebRTC is a World Wide Web Consortium (W3C) set of APIs that enables browser applications to make calls, video chats, and utilise peer-to-peer (P2P) files without the need for a plugin. The initial WebRTC implementation was developed by Google and was made open source. This implementation has been worked on by several organisations, such as the Internet Engineering Task Force (IETF) to standardise protocols and the World Wide Web Consortium (W3C) to standardise browser APIs.

There are four main APIs in WebRTC. The getUserMedia API enables the extraction of video and audio streams from devices (i.e., microphone or camera). This API call also enables us to capture a snapshot of our screen or share it with other users. The RTCPeerConnection API enables the audio/video stream to be configured. This includes a variety of various activities such as signal processing, codec execution, bandwidth administration, and streaming security. This API enables the implementation of

these various activities without the need for programmer involvement. The RTCDataChannel API enables connected users to share video or audio data. RTCDataChannel is a bidirectional data channel that enables any data type to be sent between peers. RTCDataChannel does this by using Websockets, which enable bidirectional communication between the client and server and enable the utilisation of either a slower and more reliable TCP connection or a quicker but less trustworthy UDP connection. The geoStats API provides a means of obtaining various statistics about a WebRTC session. The request order for the WebRTC protocol, illustrated in Figure 2.13, is employed to initiate a call with the API as explained previously.

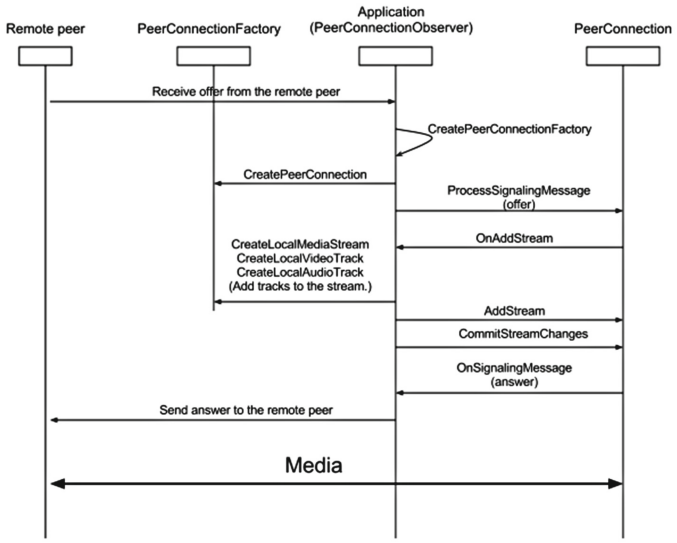

Figure 2.13: WebRTC request order. [17]

[17] describes the implementation of a system based on each of the two previously mentioned protocols while taking into consideration the same scheme and circumstances. The system was built on the usage of Android devices. For the sake of comparing the streaming establishment and sending packages with each protocol, various simulations were conducted. For both communication establishment and package sending, substantial time savings were observed while using the WebRTC protocol as opposed to the RTSP protocol in the simulations. As mentioned earlier, due to these significant advantages, the WebRTC protocol was utilised in the implementation of the present work. The time advantages of WebRTC over RTSP can be observed in Figure 2.14.

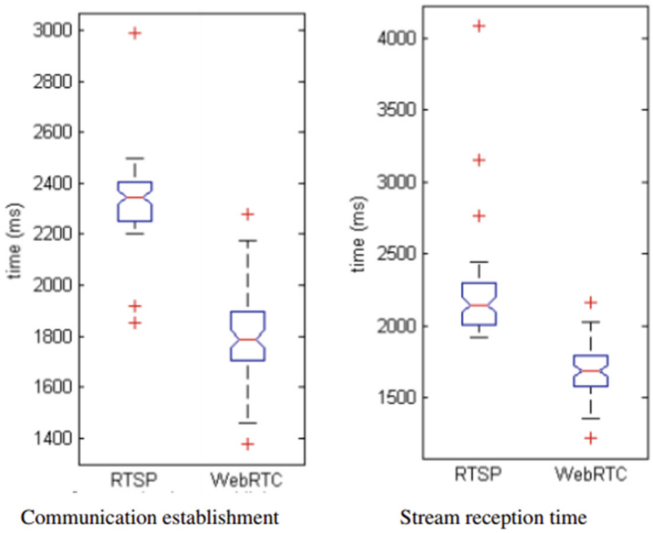

Figure 2.14: Times comparative between RTSP and WebRTC. [17]

2.4.3 On-Demand Streaming Protocol

A pull-based streaming protocol determines the bitrate at which the client receives the content depending on the connection speed of the client and the existing network capacity. HTTP is the principal download protocol on the Internet and is a widely used protocol for pull-based media delivery.

On IP networks today, progressive download is one of the most extensively utilised pull-based media streaming strategies. The media client sends an HTTP request to the server and begins extracting content from it as rapidly as feasible via progressive download. When the client reaches a certain minimum buffer level, it begins playing the media while downloading content from the server in the background. The client buffer stays at a suitable level to allow continuous playing as long as the download rate does not fall below the playback rate. Though, if network conditions deteriorate, download speed may fall behind playback speed, potentially resulting in a buffer underflow.

Similarly to push-based streaming protocols, pull-based streaming protocols employ bitrate modification to avoid buffer overflow. The example implementation shown in Figure 2.15 divides the media material into short-duration media segments (also known as fragments), each of which is encoded at a different bitrate and may be decoded separately.

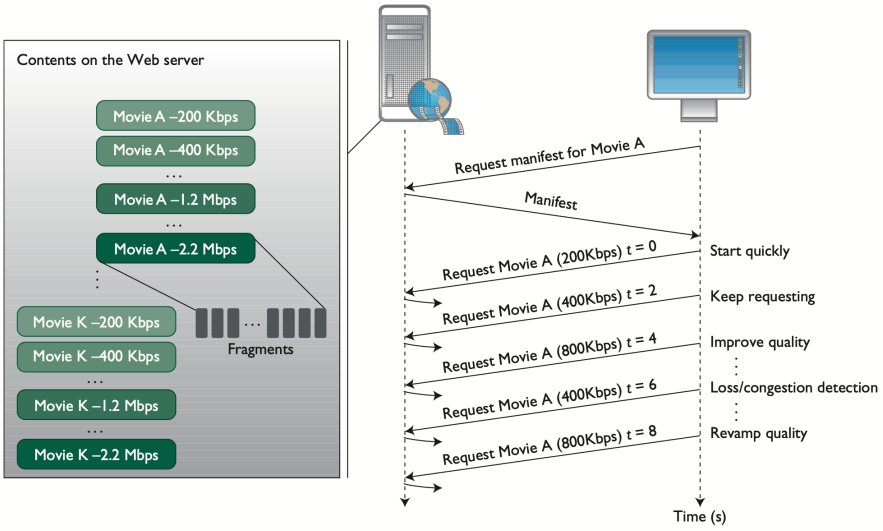

Figure 2.15: An example of how a client responds to network circumstances while using pull-based adaptive streaming. [6]

Chapter 3

Implementation

3.1 Architecture

The architecture used for the real-time video super-resolution service was manly the architecture of a Web application [5]. A high-level overview of the individual pieces that comprise this service architecture is shown in Figure 3.1.

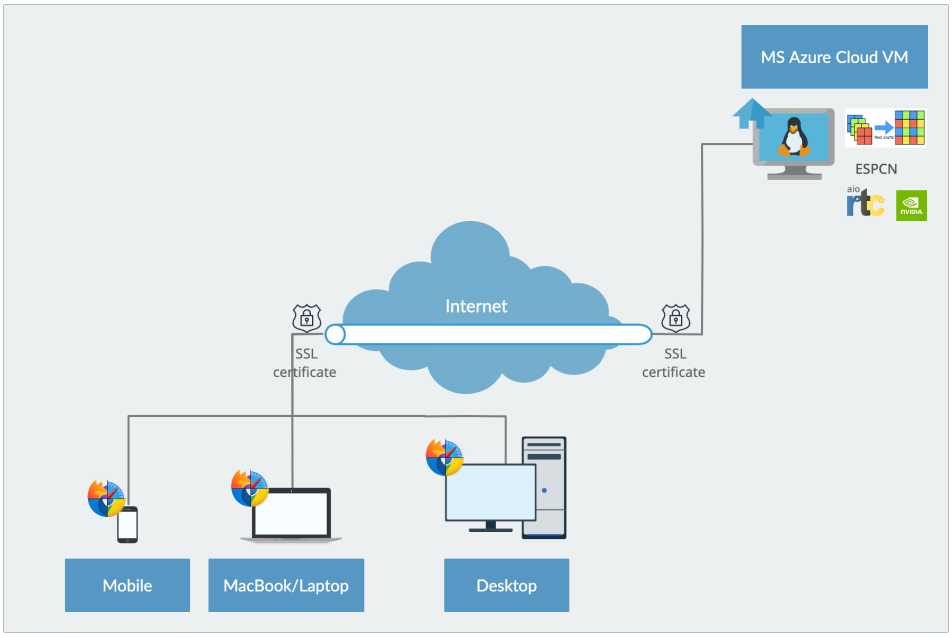

Figure 3.1: Real-time video super-resolution service architecture.

On the client-side, in any kind of device (e.g., Mobile, MacBook/Laptop or Desktop) we have a browser. On the server-side, we have the ESPCN-model and WebRTC (e.g., aiortc) which uses RTP as communication protocol. The server is a VM in Microsoft Azure with an NVIDIA GPU.

At a high level, the process that the real-time video super-resolution service follows to work is very simple. Once the service is started in the server, the user types the service URL in the address bar of the web browser, selects the desired configuration parameters and clicks the start button. Then the camera of the device begins to send the video streaming to the server where it is up-scaled to the factor that was selected in the configuration parameters and send back the up-scaled video streaming to the device. The next section explains this work process with further details.

3.2 Service description

The real-time video super-resolution service developed in this work has two main code components: aiortc and an implementation of the ESPCN-model proposed by [19]. These two components were implemented in a VM on the Microsoft Azure cloud environment [1]. In the following subsections, the relevant implementation details of the service code components and cloud environment are presented.

3.2.1 Code components

The first code component is aiortc, which is an open-source Python library that implements WebRTC and ORTC. It is based on the standard asynchronous I/O framework of Python, asyncio. The aiortc implementation is straightforward and easy to understand. Consequently, it is an excellent starting place for learning about and using WebRTC. Additionally, it is simple to develop novel products by employing the huge module library of the Python ecosystem [1]. WebRTC, hence aiortc, allows low-latency video, audio, and arbitrary data streams to be sent and received over the network by web servers and clients, including web browsers [22].

The aiortc implementation utilised in this work is based in the server example provided in the aiortc github site [2]. This example consists of three files (e.g., server.py, index.html and client.js), which had to be customised for the particular requirements of the real-time video super-resolution service that was developed.

[1]The code is available at https://github.com/lamezkua/Real-Time-Video-Super-Resolution-Service

In the server.py file, the code related to the second code component, the ESPCN-model, was included. This code replaced the server example code related to different transformations that were aimed to be applied to the video stream.

The ESPCN-model implementation was developed with the code available in [10] as a starting point. This code was modified in order to have an acceptable performance under the demanding operating requirements for a real-time video super-resolution service. In a first modified ESPCN-model version, in general, all the original CPU-executable code functions that had a GPU (Tensor)-executable equivalent were accordingly substituted. In a second modified ESPCN-model version, the NVIDIA VPF GPU-executable code framework was employed [3]. For posterior reference in this document, the first modified ESPCN-model version is called "SWCodec&CPU/GPU code" configuration if a software codec was used and "HWCodec&CPU/GPU code" configuration if a hardware codec was used. The second modified ESPCN-model version is called "HWCodec&GPU code" configuration. In both versions, the original available pre-trained weights were used. Table 3.1 presents a summary of the two modified ESPCN-model versions.

Table 3.1: Code components of the ESPCN-model versions.

	First version		Second version
	SWCodec&CPU/GPU code	HWCodec&CPU/GPU code	HW-Codec&GPU code
aiortc	x	x	x
ESPCN-model	x	x	x
GPU (Tensor)-executable code	x	x	
NVIDIA GPU-executable code (VPF)			x
software codec	x		
hardware codec		x	x

In regard to the code environment, the packages installation for the first ESPCN-model version combined directly the requirements of the aiortc and the ESPCN-model. For the second ESPCN-model version, it was necessary to first install manually the VPF requirements according to the instructions defined in [23] and then on top of these framework requirements, the packages required for aiortc and the ESPCN-model were sequentially installed.

3.2.2 Cloud environment

As mentioned in the beginning of this section, the code components that were described in the previous subsection were implemented on the Microsoft Azure cloud environment. There was no other reason to choose this cloud

environment other than the availability; there existed access to Azure with no cost for this project.

The Azure product that was selected to host the real-time video super-resolution service was a Virtual Machine [21]. Cloud-based Azure Virtual Machines offer on-demand computing resources that are billed depending on consumption. In a broader sense, a virtual machine functions similarly to a server: it is a computer inside a computer that provides users the same experience as they would on the host operating system. Virtual machines, in general, are isolated from the rest of the system, which means that the programme running inside a virtual machine cannot escape or interfere with the underlying server itself. Each virtual machine is equipped with its own virtual hardware, which may include CPUs, RAM, hard drives, network connections, and other components. The main components of the utilised VM in this project are listed in Table 3.2.

Table 3.2: Azure Virtual Machine components.

Size	Operating System	vCPUs	Memory (GB)	GPU	GPU Card	Disk Size (GB)
Standard_NC6s_v3	Linux (Ubuntu 18.04)	6	112	1	NVIDIA Tesla V100-PCIE-16GB	1024

Moreover, as a means to allow access to the service from the external clients, the inbound port rules that are shown in Figure 3.2 were defined. As well, source, destination, port, and protocol were specified for every rule.

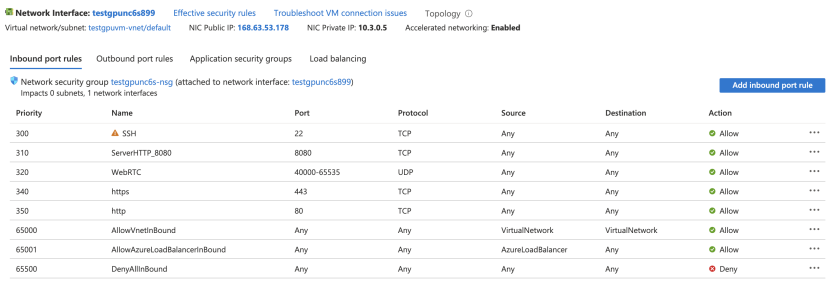

Figure 3.2: Service inbound port rules.

Additionally, Certbot was used to obtain and set up a free digital certificate from Let's Encrypt, a collaboration between the Electronic Frontier Foundation, Mozilla, and a number of other sponsors. Accordingly, a public key and a private key are utilised by a digital certificate or public key (formerly known as an SSL certificate) to enable secure communication between a client programme (e.g., a web browser or an email client) and a server over

an encrypted secure socket layer (SSL) or transport layer security (TLS) connection. The certificate identifies the server and encrypts the first step of connection (secure key exchange). The certificate contains information about the key, the identification of the server, and the digital signature of the certificate issuer. If the programme that begins the contact trusts the issuer and the signature is legitimate, the key may be employed to securely interact with the server designated by the certificate [9].

Finally, as long as it is simpler to remember a website name instead of memorising a numerical IP address, a domain name was defined for the real-time video super-resolution service (e.g., video-sr.cs.aalto.fi). The domain name was registered through the domain name service of Aalto University [11].

3.2.3 Service access

The relevant steps that must be followed up to access the real-time video super-resolution service include the following.

On the server side,

1. Load the corresponding code environment and
2. Start the server application (e.g., server.py) including the certificate information.

Then, on the client side,

1. Open a browser window and provide the following Uniform Resource Locator (URL) and port to start the client side (https://video-sr.cs.aalto.fi:8080/) and
2. Select from the displayed user interface (UI) the desired parameters to start the service.

An example of the UI with the running real-time video super-resolution service can be seen in Figure 3.3.

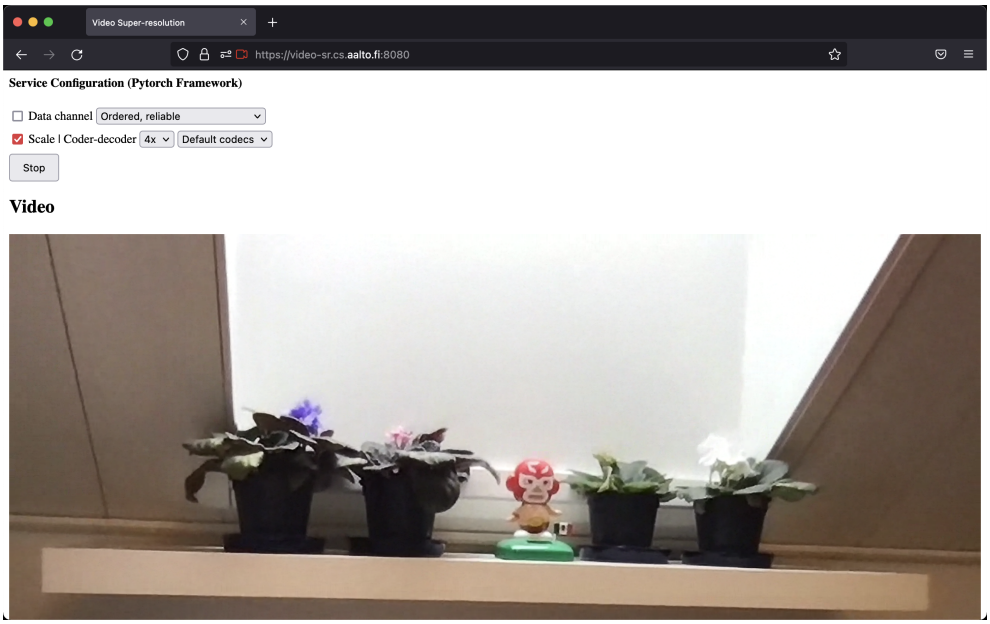

Figure 3.3: Real-time video super-resolution service.

Chapter 4

Evaluation

The code component that was evaluated to measure the performance of the real-time video super-resolution service was the ESPCN-model. In order to do this evaluation, the two implemented versions with its three configurations defined in the previous chapter were tested. Firstly, the "SWCodec&CPU/GPU code" configuration with a software codec was evaluated, secondly the "HW-Codec&CPU/GPU code" configuration with a hardware codec was assessed and thirdly the "HWCodec&GPU code" configuration with a hardware codec was tested.

For the "SWCodec&CPU/GPU code" configuration, the software h264/ libx264 decoder/encoder were used. For the "HWCodec&CPU/GPU code" configuration the hardware h264_cuvid/h264_nvenc decoder/encoder were used and for the "HWCodec&GPU code" configuration the hardware h264 codec was used.

Additionally, the two modified ESPCN-model code versions were tested in a "stand-alone" environment, that is, they were evaluated independently of the aiortc component and the input was a video file to facilitate the evaluation process. The file employed in all the benchmarks was the open-source mp4 video file file_example_MP4_480_1_5MG.mp4 [1].

Three evaluation criteria were considered for the benchmark of the two implemented versions with its three configurations. First processing time, second CPU use and third GPU utilisation.

[1]This mp4 file can be downloaded from https://www.jdownloads.org/downloads/summary/2-uncategorised/724-file-example-mp4-480-1-5mg.html

4.1 Processing time evaluation

For the processing/execution time evaluation, python methods and NVIDIA events were utilized. Moreover, three different frame sizes were evaluated (e.g., 480x270 base size, 512x288 up-scaled size and 224x126 down-scaled size). The results of the corresponding evaluations are shown in Figures 4.1, 4.2 and 4.3.

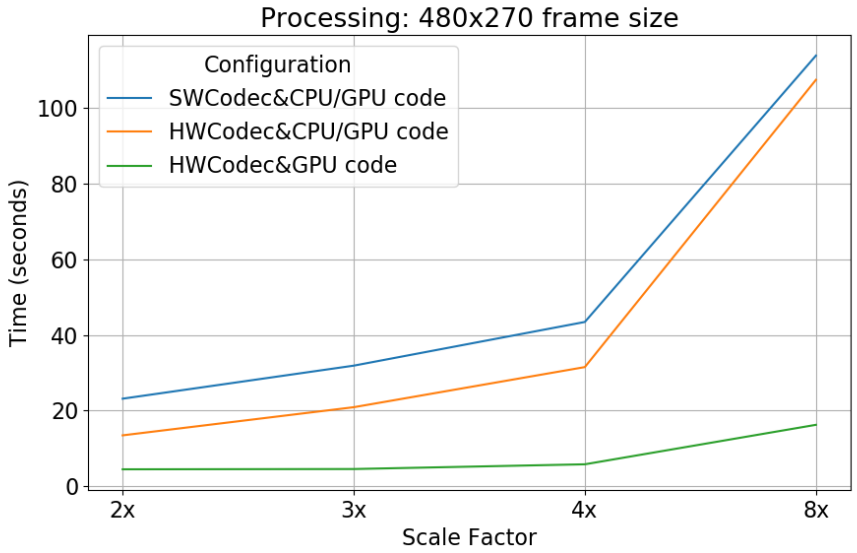

Figure 4.1: First evaluation criterion.

4.2 CPU and GPU utilization evaluation

For the CPU and GPU utilization evaluation, the Scalene python library was used [7]. Scalene is a python module that provides high-performance CPU, GPU, and memory profiling. When compared to other profilers, it is orders of magnitude quicker and provides significantly more comprehensive information. Scalene is an efficient application. It doesn't use instruments or tracing tools of python. Instead, it takes samples. Its overhead is normally between 10% and 20%. (and often less). Scalene does profiling at the line level and per function, pointing out which functions and lines of code are taking up the most time in the program [18].

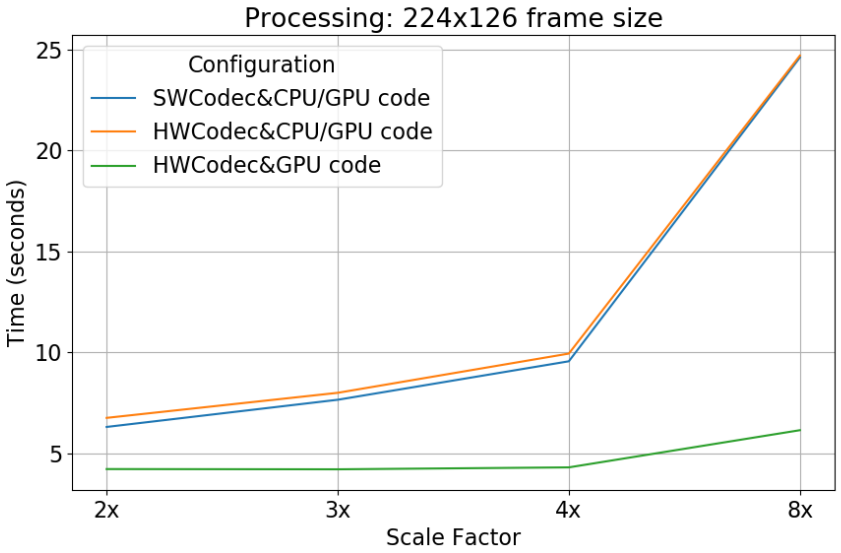

Figure 4.2: First evaluation criterion.

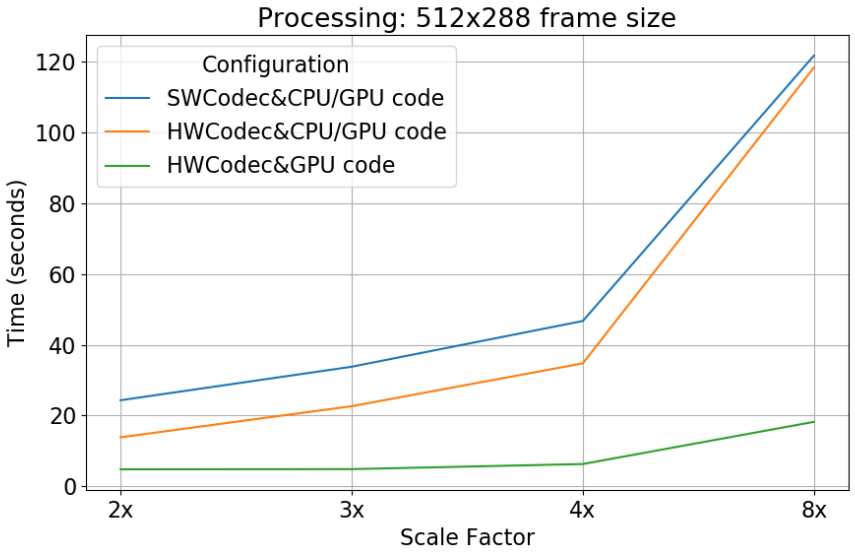

Figure 4.3: First evaluation criterion.

In the CPU and GPU utilization evaluations, the frame base size 480x270 was only evaluated. Furthermore, in the CPU use evaluation, the three dif-

ferent but complementary measures that scalene provides (e.g., python, native and system CPU use) were considered to better understand the whole behaviour of the three configurations. The python CPU usage measure specifies the amount of time that the program spent executing python code. The native CPU utilization gives the amount of time that the program spent executing non-python code (e.g., libraries written in C/C++). The system CPU use shows the amount of time that the program spent in the system (e.g., I/O). Figures 4.4, 4.5 and 4.6 show the three CPU benchmarks and Figure 4.7 presents the GPU evaluation for each configuration.

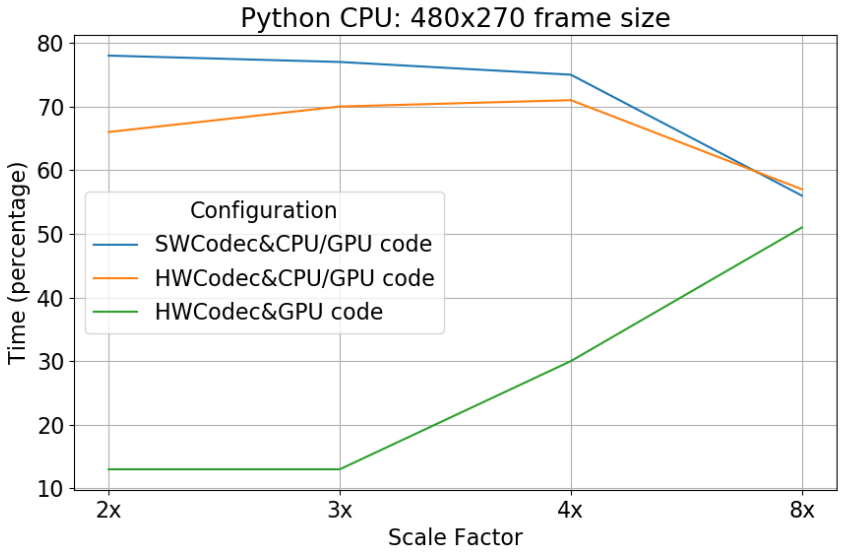

Figure 4.4: Second evaluation criterion.

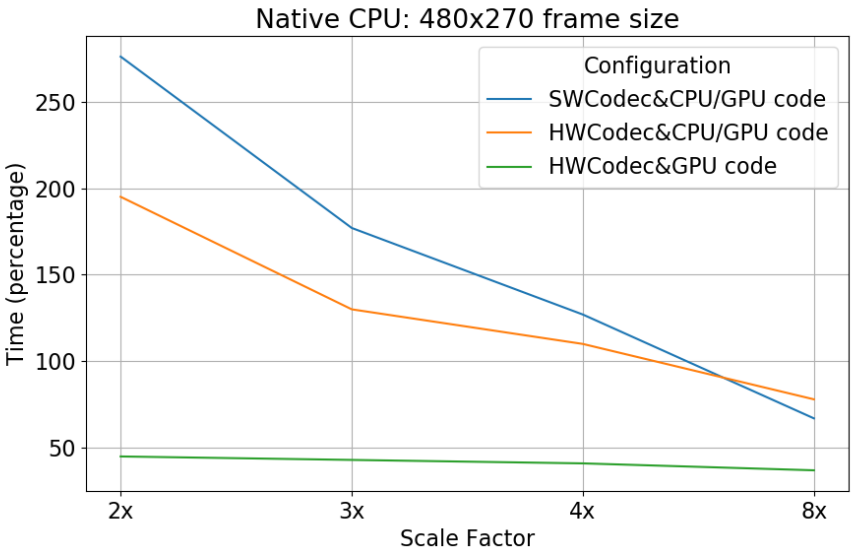

Figure 4.5: Second evaluation criterion.

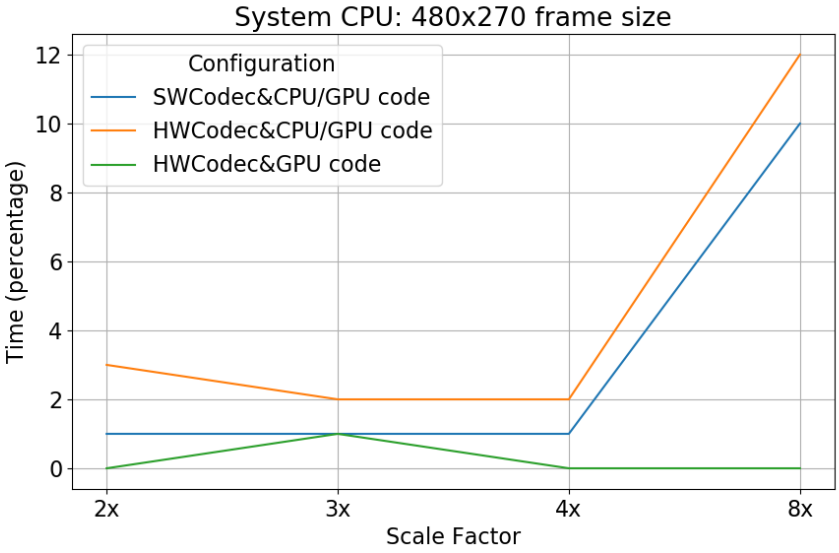

Figure 4.6: Second evaluation criterion.

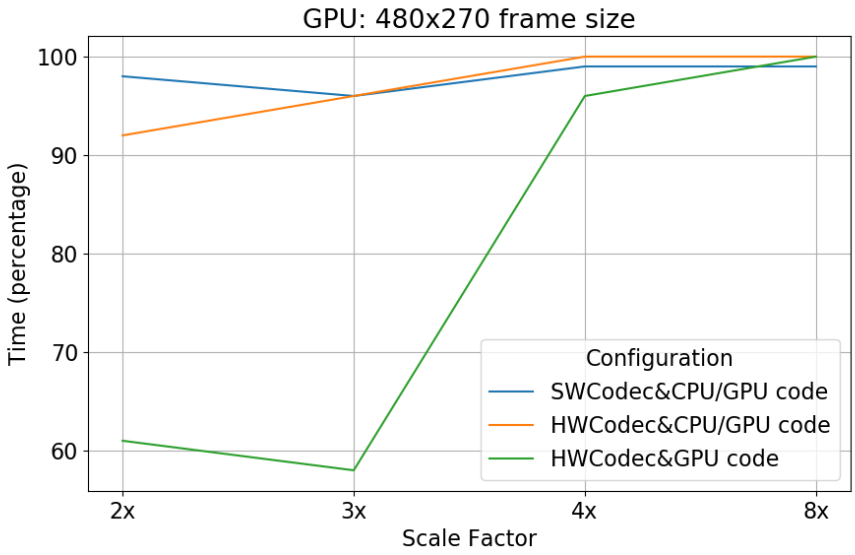

Figure 4.7: Third evaluation criterion.

Chapter 5

Discussion

Qualitative as well as quantitative results were obtained with the implementation of the present work. Qualitative results are related to the implementation exposed in chapter 3. Quantitative results are connected with the experiments explained in chapter 4.

5.1 Qualitative results

In relation to the qualitative results, as I already mentioned, two code versions were generated. Originally, the second modified ESPCN-model version (e.g., the VPF version) was created with the hypothesis that it was going to be much faster than the first version because it consisted of a pure NVIDIA GPU-executable code implementation. From a qualitative perspective, we can confirm that it was not. In both versions, for all scale factors but for the 8x factor, practically there was not delay in the video super-resolution rendering. For the 8x scale factor, there was a delay of approximately 2 seconds. However, the second version helped us instead to realize that the first code version performance was more efficient than thought. Nevertheless, in future work it would be desirable to verify if this behaviour is kept when more than one client is connected to the video super-resolution service.

5.2 Quantitative results

In reference to the quantitative results, as it was also discussed, three criteria were evaluated. These evaluation elements were processing time, CPU use and GPU utilization.

5.2.1 Processing time evaluation

In the evaluation of the first criterion (e.g., execution time), for the "SWCodec &CPU/GPU code" and "HWCodec&CPU/GPU code" configurations it was persistent in the three different frame sizes that for the scale factors 2x-3x and 3x-4x the processing time increase was uniform (the slope of the corresponding lines joining the 2x-3x and 3x-4x processing time values was practically the same). For the "HWCodec&GPU code" configuration the processing time was practically the same (the slope of the corresponding lines joining the 2x-3x and 3x-4x processing time values was practically zero). For the scale factor 4x-8x, the processing time increase was of approximately 1.5x, 2.5x and 3x for the down-scaled, basic and up-scaled frame sizes. The reason of this relevant 4x-8x processing time increase is explained in the analysis of the second criterion (e.g., system CPU use).

Here is important to mention that, due to efficiency and performance, the implementation of the code versions was focused in maximizing the use of the GPU and minimizing the utilization of the CPU. Though, the results were not in all the configurations as good as desired.

5.2.2 CPU and GPU utilization evaluation

In the evaluation of the second criterion (e.g., CPU use), firstly for the python CPU use, that is, the code where we are able to make improvements, we can see a consistent important time percentage CPU use for the scale factors 2x-3x and 3x-4x and a drop of 10-15% for the scale factor 4x-8x in the "SWCodec&CPU/GPU code" and "HWCodec&CPU/GPU code" configurations. This time percentage CPU use drop should be related to the corresponding increase of the system CPU use for this same 4x-8x scale factor. In the "HWCodec&GPU code" configuration, the time percentage CPU use for the scale factor 2x-3x was the same and for the scale factors 3x-4x and 4x-8x the time percentage CPU use increased in a consistent manner.

Secondly, the native CPU use (e.g., used by C/C++ libraries) decreased as the scale factors increased for the three configurations. This behaviour is most likely related to the increasing frame size, that is, once the native code is in memory, the CPU focuses in the execution of other instructions related to the up-scaling of the bigger frame size thus the native CPU percentage time use decreases.

Thirdly, in the system CPU utilization, we can clearly appreciate that for the "HWCodec&GPU code" configuration there is practically zero system CPU utilization, that is, zero I/O or zero swapping. For the "SWCodec &CPU/GPU code" and "HWCodec&CPU/GPU code" configurations we no-

tice that there is some system CPU utilization for the scale factors 2x-3x and 3x-4x but for the 4x-8x scale factor the system CPU utilization is increased in a relevant manner (e.g., an increment of 6-10 times). As long as system CPU utilization or I/O or swapping is a very time consuming operation, this behaviour explains an important element in the much better performance of the "HWCodec&GPU code" configuration when compared to the other two configurations.

In the evaluation of the third criterion (e.g., GPU use), we observe that for the "SWCodec&CPU/GPU code" and "HWCodec&CPU/GPU code" configurations the GPU time percentage use was always high, that is, between 90-100 % for the different scale factors. For the "HWCodec&GPU code" configuration this GPU time percentage use was only in this use range for the 4x and 8x scale factors and for the 2x and 3x scale factors the GPU percentage utilization was around half the GPU percentage use for the 4x and 8x factors. This behaviour together with the one described in the python CPU use criterion where we have similar results, show us the advantage of using the "HWCodec&GPU code" configuration versus the other two configurations when scaling video in applications with the 2x and 3x factors and provide evidence to answer two of the research questions (e.g., How can these components be combined to implement a service with a good performance? and Which are the characteristics of these components?).

Then, if we remember the main research question of this work (e.g., what are the components of an efficient real-time video super-resolution service based on the cloud?), we can now claim that one very important element of an efficient real-time video super-resolution service is a codec with an outstanding performnce, that is , a codec such as the one used in the "HWCodec&GPU code" configuration.

5.3 Additional considerations

Firstly, the default codecs of aiortc were used without change. This was a limitation for a quantitative testing of the service in real-time (e.g., not reading from a video file). A substitution of these codecs by the ones used in the evaluation chapter could be tried in a future. This would imply to modify the aortic code which might not be a trivial task. In this manner, a proper evaluation in real-time of the proposed service in this work could be achieved.

Secondly, another weakness of this work was its purely theoretical scope. The potential applications of the developed service are many. It would have been more meaningful to associate this service with a practical use-case (e.g.,

a surveillance application). To look for this practical usage-scenario combined with an improved new technique for SR would also be a remaining interesting future task.

Lastly, the evaluation of the service in real-time was qualitative. To create the use-case suggested before would also help to clearly define performance criteria against which real-time performance could be measured [14]. Furthermore, due to the fact that performance uncertainties of the existing cloud models, which usually do not give any real-time assurances and display non-deterministic performance because of shared compute and network resources, have inhibited the adoption of cloud technologies for time-sensitive and crucial services, this future version of the service proposed in this work could be implemented in a network compute fabric [16] which would provide the platform of a real-time cloud [12].

Chapter 6

Conclusions

The components defined in this work to create an efficient real-time video super-resolution service based on the cloud are aiortc (e.g., open-source Python library that implements Web Real-Time Communication) and an implementation of the ESPCN-model combined together into a Microsoft Azure cloud environment with an NVIDIA GPU.

The first code component (e.g., aiortc), allows that low-latency video is sent and received over the network by web-servers and clients, including web-browsers. In the second code component (e.g., ESPCN-model), the computational efficiency is improved and the computation and spatial complexity are considerably reduced.

The original ESPCN-model implementation was modified to implement a service with a good performance. The first modified ESPCN-model version (e.g., where all the original CPU-executable code functions that had a GPU (Tensor)-executable equivalent were substituted) practically had the same efficiency as the second modified ESPCN-model version (e.g., where the NVIDIA VPF GPU-executable code framework was used), which represents the most efficient currently available model implementation. Hence, the implementation of the ESPCN-model component can be considered as efficient from a qualitative/practical point of view.

In a real-time service like the one implemented in this project, every code line that sends or receives information (frames) from the CPU to the GPU and vice-versa (e.g., I/O) highly counts to decrease the efficiency of the whole service thus should be avoided as much as possible. Moreover, the performance benchmark that was carried-out having as input video files, from a quantitative perspective, shows that if the default codec that aiortc employs would have been configured as a HW codec (e.g., the NVIDIA one), in the real-time service whose qualitative result has just been mentioned in the paragraph before, the efficiency would have significantly increased in

its implementation. Thus, we can also conclude that the efficiency /performance of the codec plays an outstanding relevant role in the overall service configuration.

Additionally, the Azure cloud environment component, with its robust characteristics (e.g., with an NVIDIA GPU) and configured as a server with the help of the aiortc component, enables an efficient execution of the service in diverse client devices with limited resources.

In future, besides the further actions suggested in chapter 5 (e.g., Discussion), a next step for the present research would be to measure the quality of the video super-resolution done by the method selected in this work (e.g., the ESPCN-model) and compare it with other state-of-the-art models to corroborate how good the generated video is.

Bibliography

[1] aiortc — aiortc documentation. https://aiortc.readthedocs.io/en/latest/index.html. Accessed: 2021-10-30.

[2] examples/server at main - aiortc/aiortc.

[3] Arzumanyan, R., and Arzumanyan, V. A. P. b. VPF: Hardware-accelerated video processing framework in python. https://developer.nvidia.com/blog/vpf-hardware-accelerated-video-processing-framework-in-python/, Dec. 2019. Accessed: 2021-10-31.

[4] Avram, M. G. Advantages and challenges of adopting cloud computing from an enterprise perspective. *Procedia technol. 12* (2014), 529–534.

[5] Banga, S., Jack, M., stef, Barakat, M., Wissam, and Pepple, D. What is web application architecture? components, models, and types. https://hackr.io/blog/web-application-architecture-definition-models-types-and-more. Accessed: 2021-12-28.

[6] Begen, A., Akgul, T., and Baugher, M. Watching video over the web: Part 1: Streaming protocols. *IEEE Internet Comput. 15*, 2 (2011), 54–63.

[7] Berger, E. D. Scalene: Scripting-language aware profiling for python, 2020.

[8] Caballero, J., Ledig, C., Aitken, A., Acosta, A., Totz, J., Wang, Z., and Shi, W. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), IEEE.

[9] What is a certificate? — certbot 1.21.0.dev0 documentation. https://certbot.eff.org/docs/what.html. Accessed: 2021-10-31.

[10] Changyu, L. ESPCN-PyTorch: A PyTorch implementation of ESPCN based on CVPR 2016 paper Real-Time single image and video Super-Resolution using an efficient Sub-Pixel convolutional neural network.

[11] Domain name system DNS. `https://www.aalto.fi/en/services/domain-name-system-dns`. Accessed: 2021-10-31.

[12] Eker, J., Angelsmark, O., and Sefidcon, A. `https://www.ericsson.com/en/blog/2020/11/what-is-real-time-cloud`. Accessed: 2021-12-29.

[13] Hanhirova, J., Kämäräinen, T., Seppälä, S., Siekkinen, M., Hirvisalo, V., and Ylä-Jääski, A. Latency and throughput characterization of convolutional neural networks for mobile computer vision. In *Proceedings of the 9th ACM Multimedia Systems Conference* (New York, NY, USA, 2018), ACM.

[14] Hillary, N. `https://www.nxp.com/docs/en/white-paper/CWPERFORMWP.pdf`. Accessed: 2021-12-29.

[15] Hui, Z., Gao, X., Yang, Y., and Wang, X. Lightweight image super-resolution with information multi-distillation network. In *Proceedings of the 27th ACM International Conference on Multimedia* (New York, NY, USA, 2019), ACM.

[16] `https://www.ericsson.com/en/edge-computing/network-compute-fabric`. Accessed: 2021-12-29.

[17] Santos-González, I., Rivero-García, A., González-Barroso, T., Molina-Gil, J., and Caballero-Gil, P. Real-time streaming: A comparative study between RTSP and WebRTC. In *Ubiquitous Computing and Ambient Intelligence*. Springer International Publishing, Cham, 2016, pp. 313–325.

[18] scalene: Scalene: a high-performance, high-precision CPU, GPU, and memory profiler for python.

[19] Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), IEEE.

[20] Shiraz, M., Abolfazli, S., Sanaei, Z., and Gani, A. A study on virtual machine deployment for application outsourcing in mobile cloud computing. *J. Supercomput. 63*, 3 (2013), 946–964.

[21] Smith, J., and Nair, R. *Virtual machines: Versatile platforms for systems and processes*. Morgan Kaufmann, Oxford, England, 2005.

[22] (Tsuchiya), Y. T. Python WebRTC basics with aiortc. `https://dev.to/whitphx/python-webrtc-basics-with-aiortc-48id`, Feb. 2021. Accessed: 2021-10-30.

[23] VideoProcessingFramework wiki.

[24] Wang, Z., Chen, J., and Hoi, S. C. H. Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[25] Zhang, Q., Cheng, L., and Boutaba, R. Cloud computing: state-of-the-art and research challenges. *J. Internet Serv. Appl. 1*, 1 (2010), 7–18.