SINGLE IMAGE SUPER RESOLUTION WITH WIDE ACTIVATION FOR MOBILE DEVICES

Buddhika Kithmini Senevirathne

209379R

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

March 2022

SINGLE IMAGE SUPER RESOLUTION WITH WIDE ACTIVATION FOR MOBILE DEVICES

Buddhika Kithmini Senevirathne

209379R

Thesis submitted in partial fulfillment of the requirements
for the degree Master of Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

March 2022

DECLARATION

I declare that this is my own work and this dissertation does not incorporate, without acknowledgement, any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa, the non-exclusive right to reproduce and distribute my dissertation, in whole, or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Name of the candidate: Buddhika Piyum Kithmini Senevirathne (209379R)

Signature of the candidate: Date:

The above candidate has carried out research for the Masters Dissertation under my supervision.

Name of the supervisor: Dr. Thanuja Ambegoda

Signature of the supervisor: Date :

ABSTRACT

Single Image Super Resolution (SISR) revolves around the task of reconstructing a high-resolution image from a single low-resolution image. Numerous applications of SISR range from surveillance & security, medical imaging to photographic utilities. Although there are ample SISR solutions, especially those which are deployed as cloud services, there's a scarcity of effective on-device mobile SISR solutions. Even the existing solutions are mostly limited to high end mobile devices and most of the time limited by device architecture. An effective SISR solution which can run on any mobile device would be extremely helpful to the community in this context and can help gain a number of benefits in an edge-computing point of view, including storage and transfer optimization for image content. This research primarily focuses on creating such a solution, specifically focusing on usage of on-device Wide Attention Networks (WDSR) for SISR. In addition, a performance comparison will be done with other CNN based models.

Keywords: Single Image Super Resolution (SISR), CNN, WDSR, Tensorflow Lite

ACKNOWLEDGEMENT

I would like to express my profound gratitude to my supervisor, Dr. Thanuja Ambegoda for his continuous support and guidance throughout the project. Without his advice on consistent planning and organization and the supportive feedback provided on each step, this work would have not been realized.

I would also like to extend my thanks to Dr. Surangika Ranathunga for pointing me in the right direction at the inception of the project idea and to Prof. Sanath Jayasena for the invaluable passion for research he instilled in me during the undergraduate years. Last, but not least, I would like to thank the Department of CSE, UoM for providing me with an opportunity to pursue this research path.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DSP	Digital Signal Processing
EDSR	Enhanced Deep Super Resolution Network
ESPCN	Efficient Subpixel Convolutional Network
ESRGAN	Enhanced Super Resolution Generative Adversarial Network
GAN	Generative Adversarial Network
GPU	Graphic Processing Unit
LSC	Long Skip Connections
PSNR	Peak Signal to Noise Ratio
RCAN	Residual Channel Attention Network
RG	Residual Groups
SISR	Single Image Super Resolution
SR	Super Resolution
SRGAN	Super Resolution Generative Adversarial Network
SSC	Short Skip Connections
SSIM	Structural Similarity Index Measure

1. INTRODUCTION

1.1. Background

Image Super Resolution (SR) refers to the task of recreating a high-resolution output image from a provided low resolution input image. The original patented concept of super resolution [6] revolves around the idea of combining multiple low-resolution images into one high-resolution image, specifically for the purpose of digital zoom in cameras. However, with the advent of deep learning, a subfield of machine learning, the process of Single Image Super Resolution (SISR) has emerged, where the main focus is to achieve a higher resolution image from only a single image instead of a set of images. Applications of image super resolution are found across the fields ranging from surveillance and security [22], medical imaging [16, 18, 19] to recently, mobile device applications [12, 13].

Figure 1.1 – 4X upscaling of a bi-cubicc downsample image using EDSR, SRGAN, WDSR models.

Certain mobile phone manufacturers have recently launched intensive publicity and marketing campaigns about AI based super resolution in their high-end flagship mobile devices. It is declared that these devices are able to reach around 100X digital zoom capabilities along with the possibility of capturing fine details of remote objects, e.g.: - There are demonstrations of these mobile phones clearly capturing the details of the moon, including all of its visible craters, from the Earth. As mobile phones are limited in their capability of including infrastructure for optical zoom, a powerful super resolution based digital zoom can be extremely helpful. However,

super resolution being a feature of only the high-end mobile phones is a limitation which prevents the wide community making maximum use out of it.

1.2. Motivation

One of the initial objectives of this project was to create a mobile application, using an on-device machine learning based super resolution model, which could upscale images, captured via the device camera as well as those received from other sources; or functionally act as a super resolution zoom utility for any mobile device. Although there were solutions which were able to perform super resolution via connecting to cloud services, there was a gap for a solution which could do it as an on-device task. MobiSR by Lee et al. [12] had tried to provide a solution for this by introducing an on-device SR system which runs on diverse mobile device processors. This works by combining the CPU, GPU and the DSP. But as pointed by Liu et al [13], the solution is not successful due to the required multi-processor load balancing. However, Liu et al. [13] in their work SplitSR (published in January 2021), has been able to create a mobile application named ZoomSR which performs the task of super resolution which uses one neural network and one CPU. They present this as the ‘first-ever instance of on-device, deep learning-based SR’. This claim, however, is questionable as TensorFlow Lite has also provided their own implementation (in December 2020) of super resolution IoT and mobile devices. [21]

The basic difference between two implementations is, SplitSR uses a Deep Residual Channel Attention Network (RCAN) for super resolution while the TensorFlow lite implementation uses an implementation of Generative Adversarial Networks (GANs) known as the Enhanced Super-Resolution Generative Adversarial Network (ESRGAN). Apart from this, the latter supports GPU usage while the prior is designed only to utilize the CPU. There are strengths and weaknesses associated with each type of these models used and these will be considered in detail in the literature

review that follows. These implementations can be further optimized with respect to the aspects of parameter count, memory consumption, activations and runtime.

1.3. Objectives

Based upon the current work the research objectives of this project have been redefined as follows.

i. Create a super-resolution mobile application which can run with an on-device SR model on low to mid-range mobile devices.
ii. Experiment on the performance with respect to existing CNN based SR models including 'Wide Activation Networks - WDSR'.
iii. Optimize for better edge-device performance related to aspects of SR latency, memory consumption, improved device storage and better transfer amongst devices.

1.4. Thesis organization

Chapter 2 of this report presents a comprehensive literature review with the history and latest research related to the field of SISR and its application with mobile devices. Chapter 3 details the methodology which is used in this project. Figure 3.1 shows the high-level architecture diagram of the proposed solution. Figure 3.3 - 3.6 shows the results for training and validation. Figure 4.2 depicts a preview of the actual application.

2. LITERATURE REVIEW

2.1. SISR definition

Single Image Super Resolution (SISR) describes the activity of reconstructing or recovering an image of high-resolution (HR) from an image of low-resolution (LR) [3, 4, 6, 8, 11 – 14, 20, 23, 25, 28]. It is worthwhile to pay attention to the words 'recovering' and 'reconstructing' here, as the original application of SISR revolves around those. However, when viewed under a different light, applications of SISR extend beyond recovering lost information.

2.2. Degradation and recovery

SISR is defined in a context, where we try to recover information from a LR image, which in fact is a degraded version of a particular HR image. The SISR models utilize different mechanisms to handle this degradation. One of the most commonly used methods is to use a set of LR images degraded with bicubic downscaling or gaussian blurring from a set of HR images, as the training set [11, 28]. However, some researchers believe that synthetic degrading models such as bicubic degrading and gaussian degrading does not model the authentic degradation process happening in the real-world scenarios [3, 4]. Models which are designed to include parameters such as motion blur, Poison noise etc. [4] are shown to perform better when real-world SISR tasks are tested. Cai et al. in their work [3] discuss diverse degradation schemes and their effects on recovery in detail. In addition, they introduce a novel mechanism of obtaining LR, HR image sets using different focal lengths. Chen et al. [4] also complements it with their research, where they focus on alleviating the intrinsic tradeoff between resolution and field-of-view in realistic imaging systems.

2.3. SISR techniques and models

Modern researchers seek solutions for the SISR problem mainly via the application of deep neural networks. These solutions usually involve learning a mapping

between the LR images and their corresponding HR images [1, 3 - 5, 7, 8, 11, 13, 19, 20, 22 - 28]. Dong et al. is credited for pioneering in the field of SISR with his work [5] Super Resolution Convolutional Neural Network (SRCNN). Later on, there have been further improvements in SISR models including deep networks with residual learning [1], Laplacian pyramid structure [3], deep back projection [28], deep residual channel attention networks [26] and generative adversarial networks (GANs) [11, 14, 20, 24] which is the model that we are interested in this research.

2.3.1. CNN based approaches

Since, there are a multitude of implementations of SISR with CNN, we'll focus on a selected few out of them.

First, we will consider the implementation details of the SRCNN model developed by Dong et al. [5]. Bi-cubic upscaling is used in the preprocessing step of SRCNN to upscale the LR picture to the necessary size for the HR image. When the interpolated image is identified as Y , the researchers want to get an image F(Y) that is as near to the ground truth X as possible. As per the research, following 4 primary operations come next.

Patch extraction and representation – Extraction of overlapping patches from picture Y and representation of those patches as high-dimensional vectors. A collection of feature maps make up these vectors. The number of features is comparable to the vector's dimensionality.

Non-linear mapping – This operation involves mapping one high-dimensional vector onto another high-dimensional vector. Conceptually, each of the mapped vectors corresponds to an HR patch. These vectors are made up of a collection of feature maps.

Reconstruction – The final high-resolution image is created by patch-wise aggregation of the aforementioned representations. It is expected to be as close to the actual X as possible.

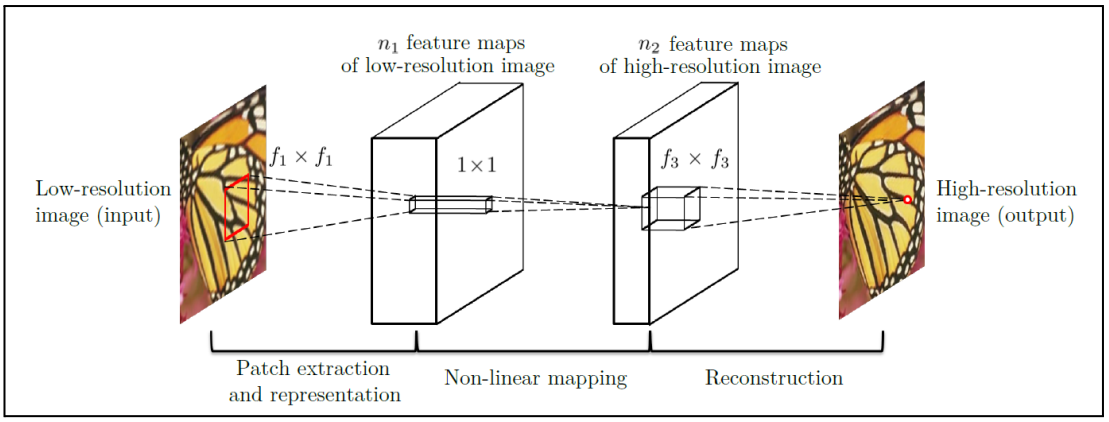


Figure 2.1 – SRCNN network architecture. Patch extraction, non-linear mapping and reconstruction layers work together to produce the SR output.

We will also consider the Residual Channel Attention Network (RCAN) proposed by Zhang et al. [26] since it is the method employed by SplitSR [13] which is amongst the closest existing works to our research.

RCAN consists of 4 major parts as well

Shallow feature extraction – One convolutional layer extracts the shallow feature from the input LR image

Residual in residual (RIR) deep feature extraction – Long skip connections and residual groups (RG) make up this component (LSC). Additionally, each residual group comprises of short skip connections(SSC) and residual channel attention blocks (RCAB) . With the help of this residual-in-residual structure, super-resolution CNNs with more than 400 layers can be trained with high performance.

Upscale module – Upscale module is capable of upscaling a feature vector. There are several possible choices for the upscale module including deconvolution layers, nearest neighbor up sampling + convolution and efficient sub-pixel convolutional network (ESPCN) [26]

Reconstruction – Reconstruction occurs via another convolutional layer.

Thereafter RCAN is optimized with a loss function such as L1, L2, perceptual and adversarial losses [11, 20, 26].

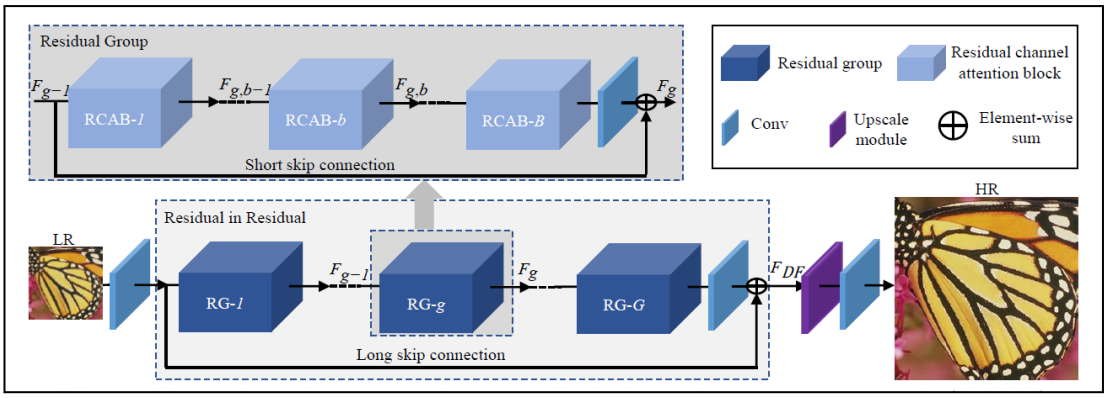


Figure 2.2 - RCAN network architecture. This consists of several residual groups which comprises a set of residual channel attention blocks and an upscale module.

The extremely lightweight quantization robust real-time super resolution network (XLSR) by Mustafa Ayazoglu [45] is one of the newest strategies that has been taken into consideration. The idea behind this was to build an extremely thin network based on the root modules for image classification described in [46]. In this instance, root modules have been expanded to address the SISR issue. Additionally, the network's last layer, a Clipped ReLU, has been employed to enable model uint8 quantization while providing a great balance between reconstruction quality and runtime.

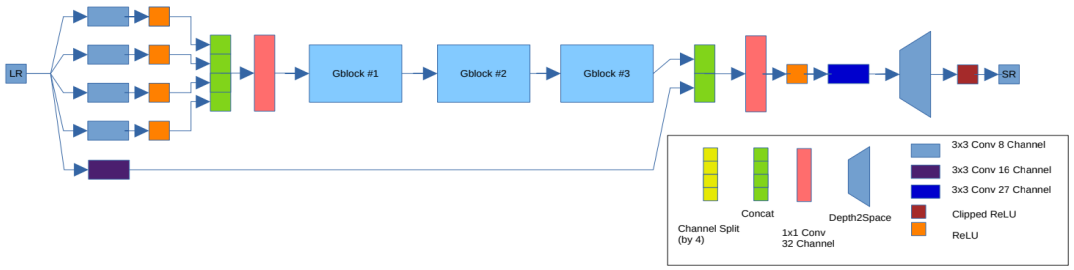


Figure 2.3 - XLSR network architecture

2.3.2. GAN based approaches

Recently, there's a lot of research happening on generative adversarial networks (GANs) for SISR [11, 20]. GAN based approaches may not produce significantly

better quantitative scores when compared with the non-GAN based CNN approaches with evaluation mechanisms such as peak signal to noise ratio (PSNR), structural similarity index (SSIM) etc. However, GANs are successful in producing images of higher perceptual quality which are more photo-realistic. GAN for super resolution (SRGANs) also consist of the two major components found in any GAN

Generator – Learn and generate data similar to real data. This can be the result of several iterative steps. The implementation of this is identified as an SRResNet, an SR CNN with residual blocks and skip connections. This type of network is used in many of the SR models to follow.

Discriminator – Identify the generated false data from the real data. Implementation consists of a normalizing layer, several discriminative blocks , and a ReLu activation layer.

Ledig et. Al [11] has made a significant improvement in the perceptual quality and photo realism of texture details in their SRGAN. They have made use of a customized perceptual loss function in order to achieve this.

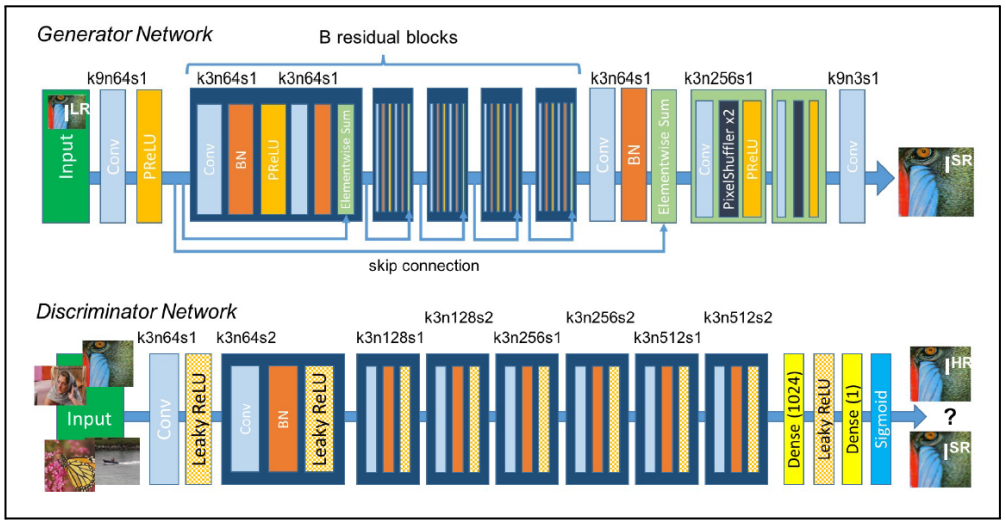


Figure 2.4 - GAN network architecture. This consists of two major parts, the generator network and the discriminator network..

2.3.3. Enhanced Deep Residual Network (EDSR)

Increasing the number of network parameters in a neural network leads to an increase in performance. In CNNs, the performance can be easily improved by aggregating several layers. This can also be done by adding more filters. However, it has been observed that addition of feature maps beyond a certain point makes the training process numerically unstable. B. Lim et al [43] introduces EDSR to overcome. Here they use a residual scaling with factor value 0.1. They also place a scaling layer after the last convolution layer of a residual block. These modules make the training highly stabilized in the EDSR model. The model architecture of EDSR is similar to SRResNe, with the exception of having ReLU activation layers outside the residual blocks.

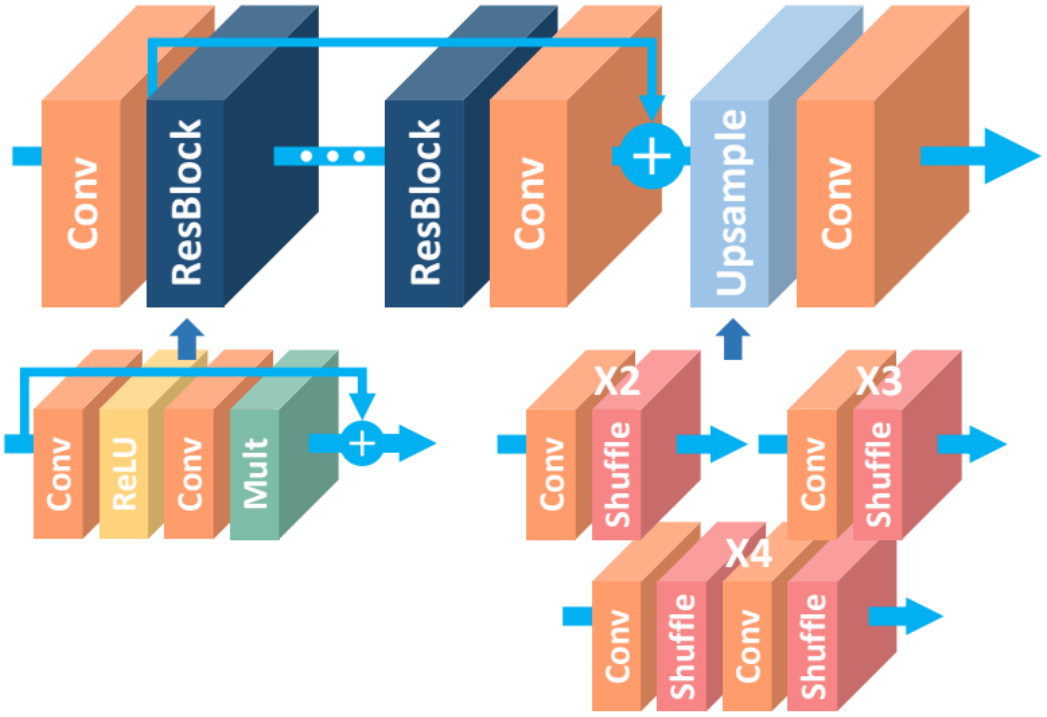


Figure 2.5 - EDSR architecture, which closely follows SRResNet, except the lack of ReLu activation outside residual blocks.

2.3.4. Wide activation based approach (WDSR)

WDSR is an improvement over EDSR. In WDSR, J. Yu et al. [40] offer an approach with a thinner identity mapping pathway and large activation layer in consideration of the fact that non linear ReLU layers hinder information flow towards the deeper

layers. Here, they demonstrate that it helps achieve appreciable increases for single picture super resolution without the need for new parameters and calculations and merely by extending features before ReLU activation in residual SR networks. Additionally, this outperforms SR networks with skip connections. J. Yu et al. suggest passing through more information and maintaining the networks' high degree of nonlinearity by extending features before ReLU. This method enables dense pixel value predictions by allowing low-level SR characteristics from earlier layers to pass through to the final layer with ease. The concept of wide activation centers on the search for effective means of feature expansion prior to ReLU. For real-time image SR application cases, simple parameter insertion is not ideal. WDSR-A is launched as we look for effective ways to expand features prior to ReLU. Prior to the activation layer in the residual blocks, WDSR-A has broader (2 x to 4 x) channels and a "slim identity mapping pathway." When the expansion ratio in this method exceeds 4, identity mapping pathway channels need to be slimmed down considerably, which is observed to significantly degrade accuracy. J. Yu et al. suggest a linear low-rank convolution that can convert a huge convolution kernel into two distinct low-rank convolution kernels to get around this problem. It is WDSR-B model. Without the use of any additional parameters or calculations, this SR network has a larger activation (6x to 9x). Additionally, this increases the super resolution accuracy. Additionally, WDSR employs weight normalization during training since it has been shown to improve super resolution accuracy in comparison to batch normalization or no normalization. Figure 2.3 shows how redundant convolution layers are taken out and the activation layer is improved. The up-sampling layer of the WDSR model is represented by the pixel shuffle layer in Figure 2.3.

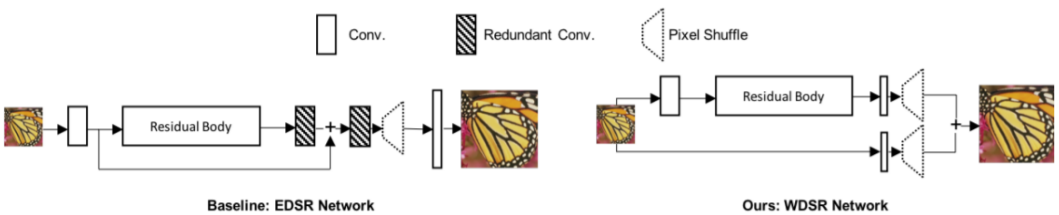


Figure 2.6 - WDSR removes redundant convolutional layers before the pixel shuffle, up-sampling layer.

2.4. Training datasets

For training, the majority of SISR models use LR and HR picture sets [11, 12, 13, 20]. DIV2K is a high-quality dataset with a 2K resolution that can be used for picture restoration jobs. Among the most often used datasets for training are Flick2K, another 2K image set collected from the website "Flickr," and the OutdoorSceneTraining (OST) dataset [20]. Researchers question the applicability of these data sets' simulated LR images in real-world situations because they use techniques like gaussian blurring and bi-cubic degradation. A number of researchers have also developed novel data collection methods to address this problem. A similar method is presented by Cai et al. [3], in which the focal length of the camera lens is adjusted to acquire LR, HR image pairs of the same scene.

2.5. Evaluation metrics

These evaluation metrics can also be identified as loss functions which quantify the information lost between the converted image and the ground truth. Most commonly used evaluation metrics for SISR are,

2.5.1. Mean Squared Error (MSE)

This is a pixel wise measurement taken amongst the converted image and the original HR image or the ground truth. The transformed image is more accurate to the original as the MSE decreases.

2.5.2. Peak Signal to Noise Ratio (PSNR)

The PSNR refers to the ratio between the maximum power of a signal to the power of corrupting noise. Having a greater PSNR is good. MSE and PSNR are connected. The PSNR increases as the MSE decreases.

2.5.3. Structural Similarity Index (SSIM)

Refers to a perceptual based model which defines image degradation in terms of perceived change in structural information. This considers contrast masking and

luminance masking terms where the latter represents the phenomenon where image distortion appears to be less visible in the bright regions and the former represents the phenomenon where distortions appear less visible in areas of the image with significant texture.

Out of these, MSE is the weakest measurement while SSIM is the strongest when it comes to assessing the perceptual quality. However, there are custom loss functions which better represent the perceptual quality rather than the quantitative indices which may not correctly represent the perceptual status at all. Few such examples are perceptual loss function and VGG loss. Ledig et al. in their work [11], presents the addition of adversarial loss and the content loss as the perceptual loss. In this context, content loss refers to a VGG loss based on the ReLU activation layers of a pretrained VGG network. Adversarial loss is taken as a logarithmic sum of certain probability measurements in the discriminator component.

However, most researchers tend to use PSNR and SSIM based measurement for evaluation since they have already become common criteria which can be used to compare performance among different SISR models.

These measurements along with scores on system resource usage and CPU, GPU execution times provide a comprehensive performance evaluation metrics for SISR.

$$MSE = \frac{1}{mn}\sum_{0}^{m-1}\sum_{0}^{n-1}\|f(i,j) - g(i,j)\|^2$$

$$PSNR = 20\log_{10}\left(\frac{MAX_f}{\sqrt{MSE}}\right)$$

f — original image represented as matrix data

g — degraded image represented as matrix data

m — number of pixel rows (height)

n — number of pixels columns (width)

i — index of the particular row

j — index of the particular column

Max(f) — HR image's maximum signal value

2.6. Use cases

The primary application of SISR is to recover an HR image from LR image as per the definition itself. However, there are a multitude of ways where SISR can extend beyond image information recovery. Since deep neural networks and especially GANs are able to generate new information, these are also said to be hallucinating, meaning they are creating content which was not there earlier, on their own.

Apart from recovering information, SISR can enhance an image at its maximum resolution as captured by an imaging device, thus providing photo-realistic digital zoom functionality. In this research we are focused on an on-device solution rather than a cloud deployed service. We are considering the scenario of running a SISR machine learning model on an edge device as edge computing is anticipated to be the future of computing.

On-device SISR can be very useful for a person with their static photos. A user can capture a photo of a document and can be confident that it can be viewed later since a properly trained SISR can make the text more readable [13]. Another benefit would be when a user stores images on a device, this system can reduce the storage requirements for the images thus providing a novel way of compression. Extending from that a more important feature would be, while sending images from one device to another, on-device SISR can reduce the requirements for data transmission. The transmitted images can then later be up-sampled on the destination device. This can also be highly useful in the field of IoT, where sensor networks need to send image data and internet connectivity is sub optimal. 'NVIDIA Maxine' [17] has a similar

mechanism in place for video conferencing where they have reduced the data transmission requirement by 90% using super resolution.

Another classic use case of SISR is its applícation in surveillance and security. Advances in SISR have enabled the development of surveillance systems with superior zoom capabilities [22]. With ubiquitous computing, an on-device SISR model could further facilitate surveillance and security systems, improving the security of the system itself by reducing the number of external connections.

Healthcare is another domain, where SISR is having a significant impact on [16, 18, 19]. In the developing parts of the world where low-end to mid-range mobile devices are more common, this can aid in multiple ways. Plug and play medical equipment such as compact ultrasound probes facilitate medical imagery where the mobile device functions as the communication and computation center[13]. Furthermore, researchers develop utilities to augment the functionalities of the built-in mobile phone cameras, so that they can be used for medical tests such as identification of refractive errors and cataracts. Mobile apps that can identify traumatic brain injuries based on pupil dilation can be sighted as another important application of SISR [13]. An enhanced on-device SISR could help the growth of all these areas.

2.7. Mobile-device SISR

There are a multitude of applications which provide SISR on mobile devices by connecting to deployed cloud services. However, in this research we are focused on solutions which can perform SISR as an on-device functionality. A complete flow of ML based SISR including the training phases is not feasible on a mobile platform. The most practical way of running an on-device ML application is with transfer learning where pretrained models can be converted and used for the desired task. There has been a lot of research into running ML based solutions on mobile devices. “MobileNets” developed by Google [7, 15] provides a set of efficient models which can be used for different image processing tasks. In addition to this, DeepSense by

Huynh et al. [8] and DeepX by Lane et al. [10] provide platforms which facilitate efficient execution of neural networks on mobile devices.

When we consider the on-device SISR capability, Liu et al.'s [13] ZoomSR is amongst the first applications to make it to the list. MobiSR developed by Lee et al. [12] has researched on deploying an on-device application prior to this. This system was designed to run on diverse mobile device processors, combining the CPU, GPU and the DSP. But as pointed out by Liu et al [13], there are problems implementing that on a mobile device due to the required load balancing on multiple processors. ZoomSR uses an RCAN architecture with one neural network for SISR. It runs on a TVM using one CPU. However, even ZoomSR only acts as a prototype application which upscales an image from the device gallery and does not provide any other functionality. These details about ZoomSR are based only on the details provided in their research paper. The application was not made publicly available at the time of writing this report.

Apart from those solutions, "TensorFlow Lite" team has created their own solution [21] which can perform the SISR task. This approach makes use of an ESRGAN developed with Keras for SISR. The ESRGAN model is first converted to the on-device format with TensorFlow Lite converter. Then we can run it on a mobile device with the TensorFlow lite inference. However, the default solution they have provided has some limitations as well. The solution is limited to a number of pre-selected images. In addition to that their default solution has image sizes set to a predefined 50x50 prior to model conversion. Hence, it does not support dynamic resizing and resizing and resizing a different image requires retraining of the model. However, these challenges can be overcome and mitigated with the functionality available in the TensorFlow system itself.

SR learning models usually tend to be more resource-intensive than other discriminative models [12], as each layer in these networks needs to maintain or upscale the spatial dimensions of their feature maps. Although low-mid level mobile

phones are limited in the resources corresponding to processing power and memory, they usually have cameras capable of producing high resolution images. Therefore very deep networks seem to be infeasible for the mobile environment. As a result, we are made to consider a solution which provides better run-time latency and if not avoidable, provides sub-optimal PSNR, MSE and SSIM values.

Based on current research on mobile on-device SR [12, 13] I tested on the possibility of using different frameworks which allow on-device inference of neural networks such as **Apache TVM, NCNN, Tensorflow Lite**. Apache TVM and Tensorflow Lite employ model compression techniques to create optimized models which would run on mobile hardware.

These frameworks support optimizing the models for mobile environments via the operation of **quantization** and **pruning** [35]. In quantization, the precision of the numbers used to represent a model's parameters are reduced, for example, from default float32 to float 16. The result is a smaller model size which performs the computations faster. Pruning is the process of removing parameters of a model in a way that it would have minor impacts on its predictions. The advantage of a pruned model lies in the fact that they can be compressed more effectively. However, functionally they could occupy the same size on the disk and have the same run-time latency.

Apache TVM is machine learning inference and compiler framework which can transform the computation graph minimizing the memory utilization, optimizing the data layout and fusing computation patterns [37]. In addition to graph optimization, TVM also supports optimizations via quantization and pruning.

NCNN is another high-performance neural network inference computing framework which is optimized for mobile platforms. NCCN runs independently and contains no third party dependencies. [38]

Tensorflow Lite also facilitates neural network inference for mobile platforms. Tensorflow Lite provides the optimization via quantization and pruning as mentioned above. In addition, Tensorflow Lite supports optimization via clustering. In clustering, the weights of each layer in a model are grouped into a predefined number of clusters, and the centroid values for the weights of each individual cluster is shared. This aids in reducing the complexity by reducing the number of unique weight values in a model. [39]

Study	**Outcome Measures**	**Relevant Findings**
[3] J. Cai et al - "Towards real-world single image super-resolution: A new benchmark and a new model" - ICCV -, 2019.	- PSNR, SSIM	- Degradation kernels differing to bi-cubic degradation. - Important to train ours using a similar dataset
[11] C. Ledig *et al* - "Photo-realistic single image super-resolution using a generative adversarial network" - *CVPR*, 2017.	- Perceptual loss function - PSNR, SSIM measures - Mean Opinion Score	- SRGAN. GAN capable of super resolution and produces better quality than previous CNN based work. - Perceptual loss function which includes VGG based content loss, MSE content loss and adversarial loss - Mean Opinion Score which proves better perceptual quality may directly map with PSNR and SSIM figures.
[12] R. Lee et al - "MobiSR: Efficient on-device super-resolution through heterogeneous mobile processors" - *MobiCom '19* - 2019.	- PSNR drops - Total Variation (TV) thresholds	- RCAN based model for super resolution - Heterogenous mobile processors used for running the models. CPU, GPU, DSP combined together - Quality of Result (QoR) vs latency tradeoff - SR model compression
[13] X. Liu et al - "SplitSR: An End-to-End Approach to Super-Resolution on Mobile Devices" - 2021.	- PSNR, SSIM - Latency focused, accuracy focused	- RCAN based model for super resolution - Hybrid architecture, standard convolutional blocks lightweight residual blocks - ZoomSR mobile application which can perform on device SR

		- Tiling, Packing, Vectorization, Rolling operations - TVM virtualization to run the on-device models
[20] X. Wang *et al* - "ESRGAN: Enhanced super-resolution generative adversarial networks" - in *Lecture Notes in Computer Science* - 2019	- Perceptual loss - Qualitative results	- Improved Generator from SRGAN where basic blocks are replaced with original residual basic in basic blocks - All Batch Normalization layers are removed from generator - Relativistic Average Discriminator replaces normal discriminator.
[27] Y. Zhang et al - "Residual Dense Network for Image Super-Resolution" - *IEEE/CVF CVPR*, 2018	- PSNR, SSIM	- RDN for image super resolution - Notebooks and Code related to 'idealo' implementation
[40] J. Yu et al - "Wide activation for efficient and accurate image super-resolution" - 2018.	- PSNR, SSIM	- WDSR for image super resolution - WDSR-8-B Model extracted from here

Table 2.1 - Related research work (Filtered)

3. METHODOLOGY

The expected outcome of this research is to create an on-device SISR application experimenting with the CNN approaches and optimize the

edge-computing attributes. In the 'Mobile Device SISR' section of the literature review details of the existing solution were discussed.

The outcome of this research stands out from the rest as there are no applications yet, which can act as an externally installable SISR based super resolution utility for mobile devices. Existing solutions mentioned above [13] are limited in their functionality and also inefficient in producing an output.

3.1. Implementation

3.1.1. Mobile Framework

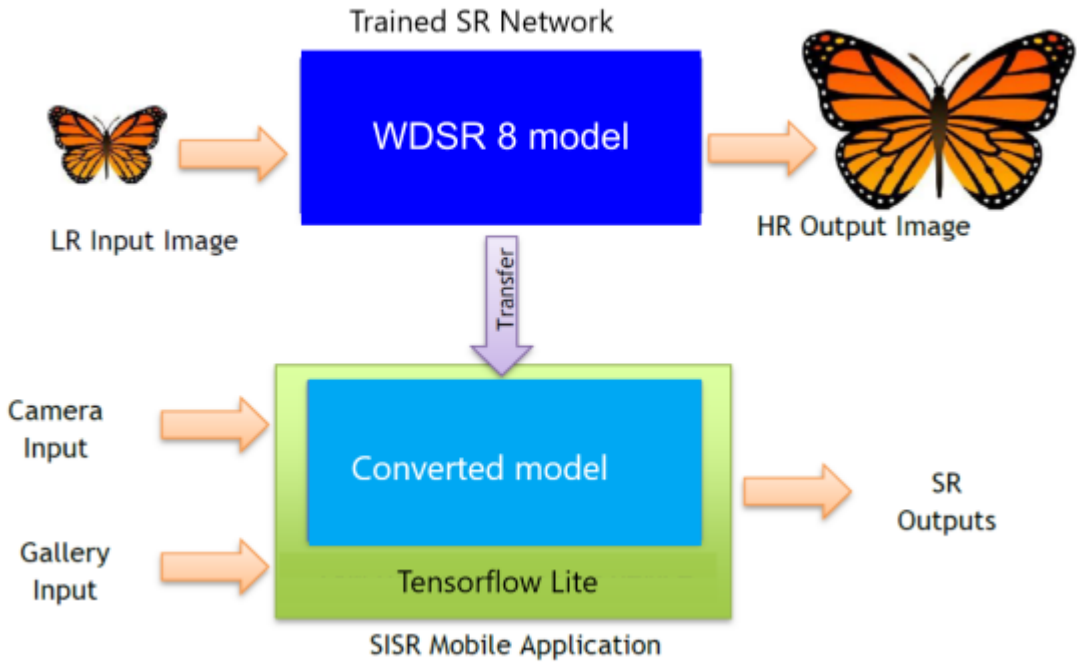


Figure 3.1 - Proposed overall architecture for the mobile on-device SISR application.

I tested with the inference frameworks **Apache TVM, NCNN, Tensorflow Lite** (mentioned in the 'Mobile Device SISR' section of the literature review) for the mobile on-device SISR. I did an analysis using converted pretrained models SRGAN, EDSR and WDSR models to 4x upscale a set of 100, 50 x 50 images on a low end Android mobile device with 'Mediatek' MT6739 : Quad core CPU 1.5

GHz: GE8100 570MHz and 1 GB RAM specifications. In the models used the upscaling time varied image wise and the average time was considered.

	Frameworks		
Model	**TVM**	**NCNN**	**Tensorflow Lite**
SRGAN	3,432	3,501	**2,860**
EDSR	3,105	3,170	**2,591**
WDSR(B-32)	3,048	3,109	**2,502**
WDSR(B-8)	3,012	3,071	**2,498**

*Table 3.1 - Average run time in ms to upscale a set of 100, 50*50 low res images. Default settings were allowed in the frameworks without any additional specific optimizations.*

Perceptual quality appeared to be similar across the applications. Given the fact that the same model was used across the platform, unless the platforms were to heavily alter the functionality of the model, that was the expected outcome. As perceptual quality appeared within a similar range, it was decided to use Tensorflow Lite for the mobile on-device SISR application considering the latency.

3.1.2. SR Models

Mainly three models were considered and tested out on the Mobile platform

SRGAN model

As stated earlier, 'Generative Adversarial Network' based SR model performs with the aid of two main subnetworks. Generator network, which can learn and generate data similar to real data. Discriminator network, which learns to identify the generated fake data from the real data. While SRGAN models are capable of producing high perceptual quality, training and prediction on them are relatively expensive.

The implementation of the SRGAN model with Tensorflow 2.0, develops on the prior work done by M. Krasser [44]. Generator consists of an SRResNet network block and the discriminator consists of a normalization layer, several discriminative

blocks, a dense layer and ReLu followed by another dense layer with sigmoid activation.

EDSR

As stated earlier, EDSR closely follows the architecture of SRResNet with the exception of no ReLu activations layers outside of the residual blocks. The implementation consists of a normalization layer, several residual blocks(which consists of 2D convolutional kernels and a scaling layer), up-sampling layer, 2D convolution kernel and a normalizing layer.

WDSR

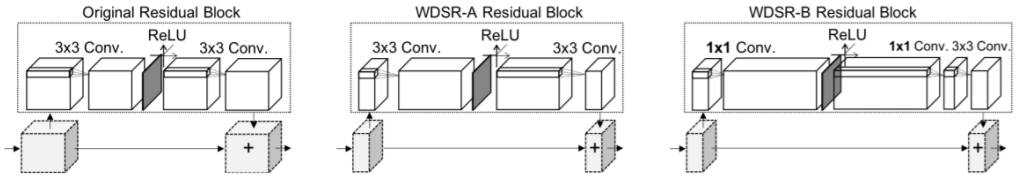


Figure 3.2 - Left - regular residual block type: Right - WDSR-B model with wider activation and linear low-rank convolution: Middle - WDSR-A model with wide activation

J. Yu et al. has improved upon EDSR to produce WDSR. WDSR has two variants namely WDSR-A and WDSR-B. As discussed earlier, the WDSR slims down the identity mapping pathway along with wider channels. This has been shown to increase the accuracy without the need of any additional parameters. In addition to this, WDSR employs weight normalization, which further increases the accuracy with batch normalization or no normalization.

The WDSR-B model was used in the mobile SISR application. The reason for selecting this was it is fairly lightweight compared with other models. The Tensorflow implementation of WDSR consists of a normalization layer, 2D convolution kernel with weight normalization, residual block (which consists of several layers of weight normalized 2d convolution kernels), repeating layers of weight normalized 2D convolution kernels and pixel shuffle layers followed by a final denormalizing layer.

3.1.3. Training

Training of the models were mainly done with DIV2K,. The default train, test, validation splits of the DIV2K dataset consists of 800 training images, 100 testing images and 100 validating images. 4X bi-cubic downsampling was used to obtain a set of downscaled images. Later on a set of image transformations including random flip, random crop, random rotate were used for the purpose of data augmentation as the training data set was limited to only 800 images. The specifications of the computer on which the training was carried out is as follows.

Specifications

- Processor - Intel Core i7-1165G7 @ 2.80GHz 2.80 GHz
- RAM - 8.00 GB
- GPU - NVIDIA MX 330
- System type - 64-bit operating system, x64-based processor

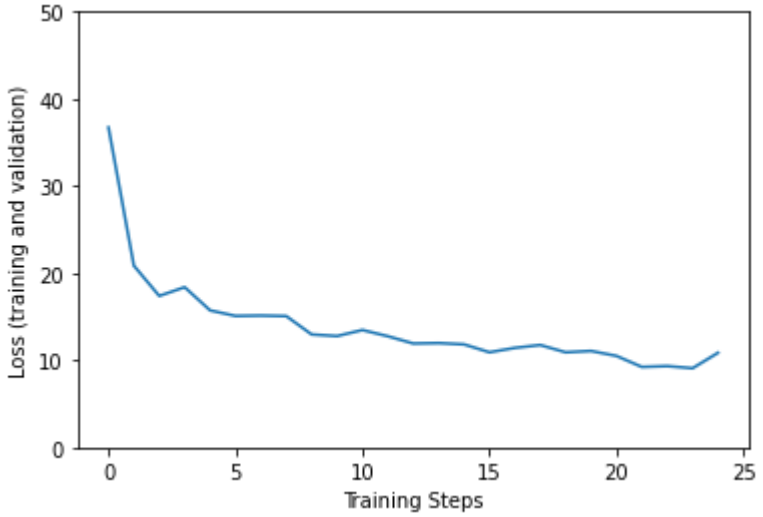
Average training times for 10 epochs on CPU (800 images in the training dataset, 16 batch size.

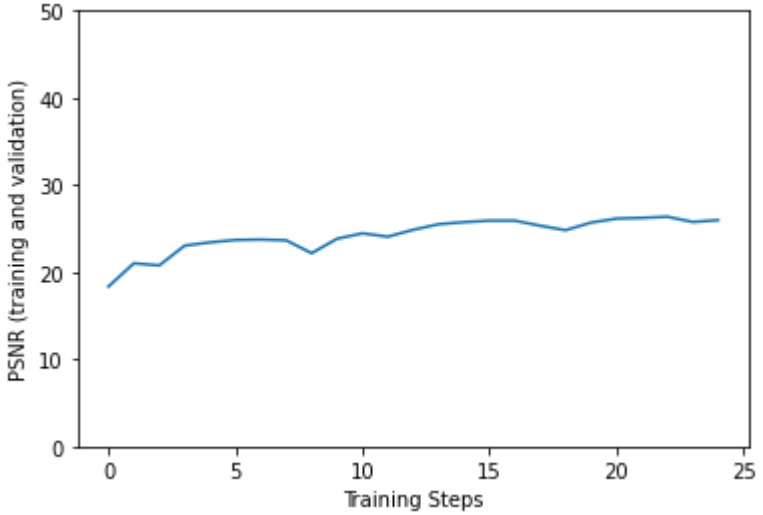
- EDSR - 1 hrs
- WDSR - 1 hrs
- SRGAN - 2 hrs

Loss, PSNR & SSIM

The variation of loss, PSNR and SSIM across during training is discussed below. For Figure 3.3 - 3.6 examples, training is done for 10 epochs (500 iterations) and the validations are done every 20 steps . Dataset is prepared from DIV2K with bicubic downsampling. As the training set is limited to 800 images data augmentation is done with random crops, flips and rotations.

EDSR 8





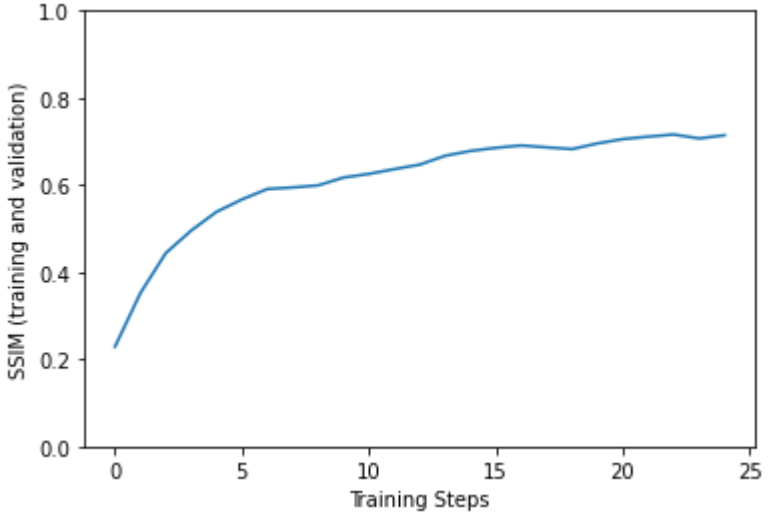
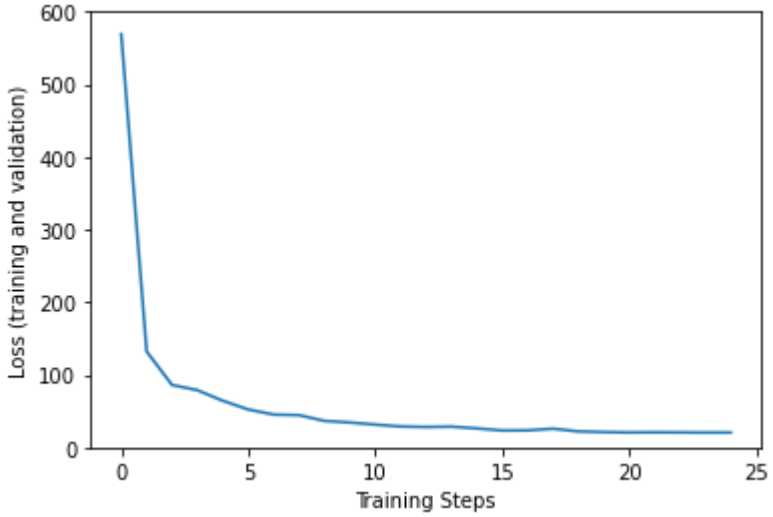
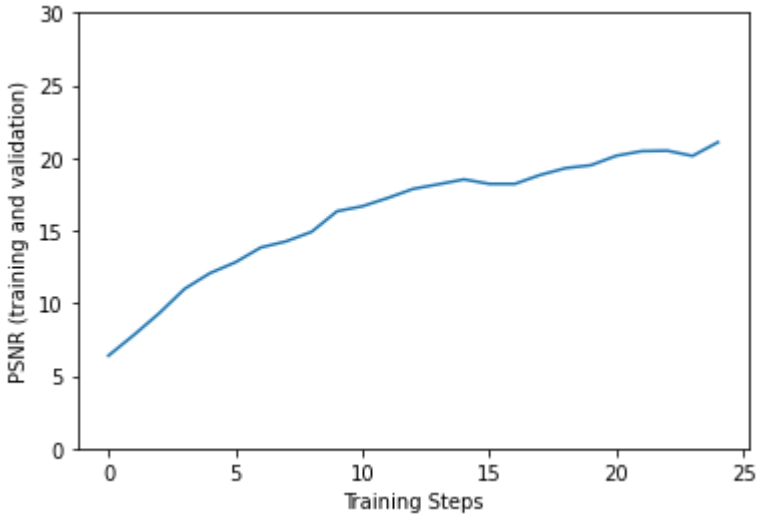


Figure 3.3 - EDSR 8 loss, PSNR & SSIM

EDSR 32





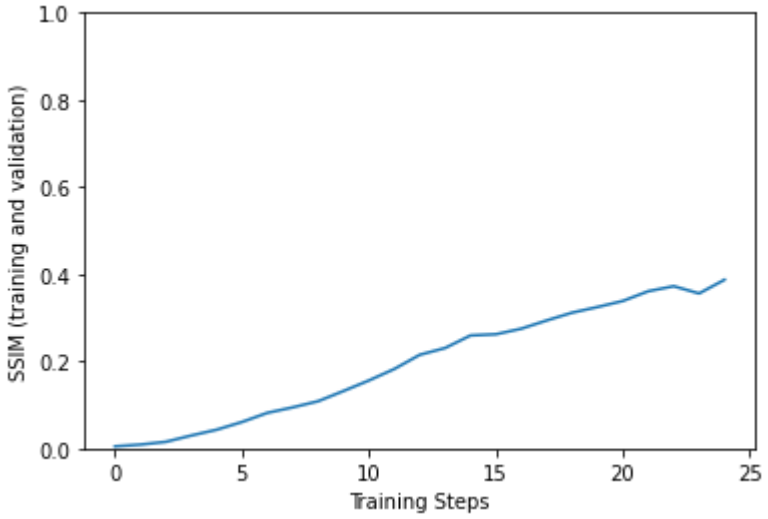
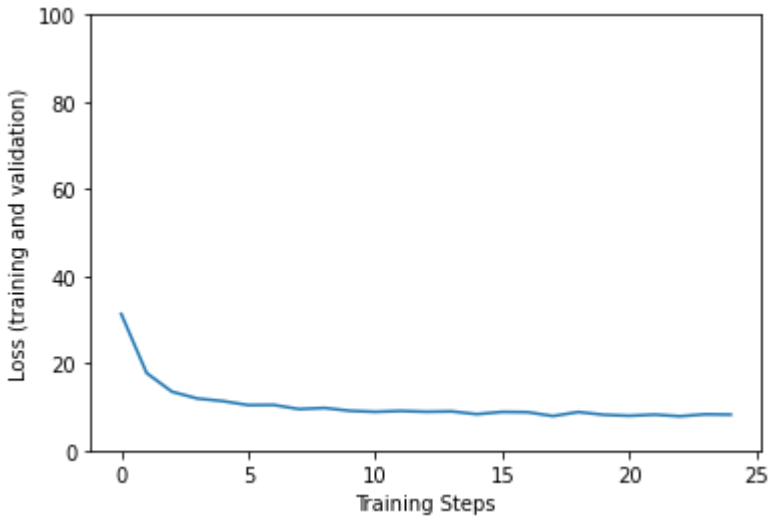
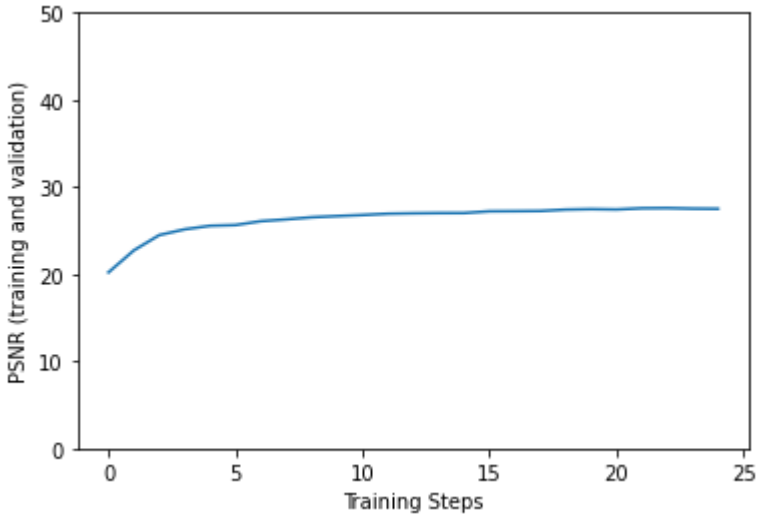


Figure 3.4 - EDSR 32 loss, PSNR & SSIM

WDSR 8





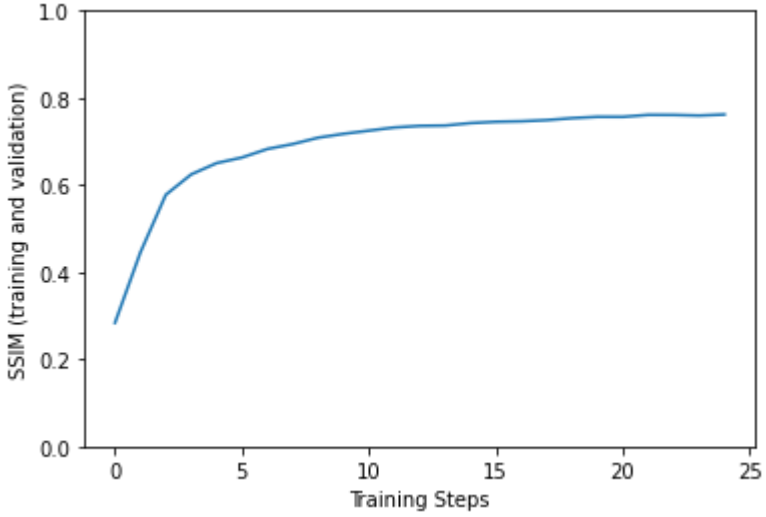
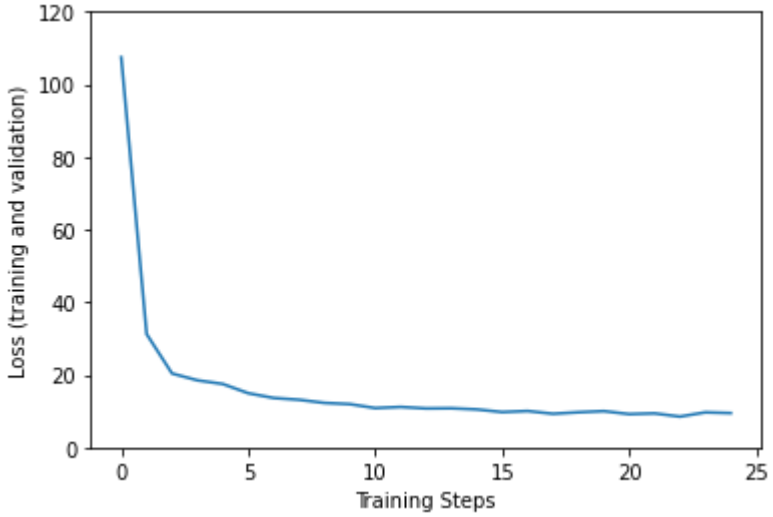
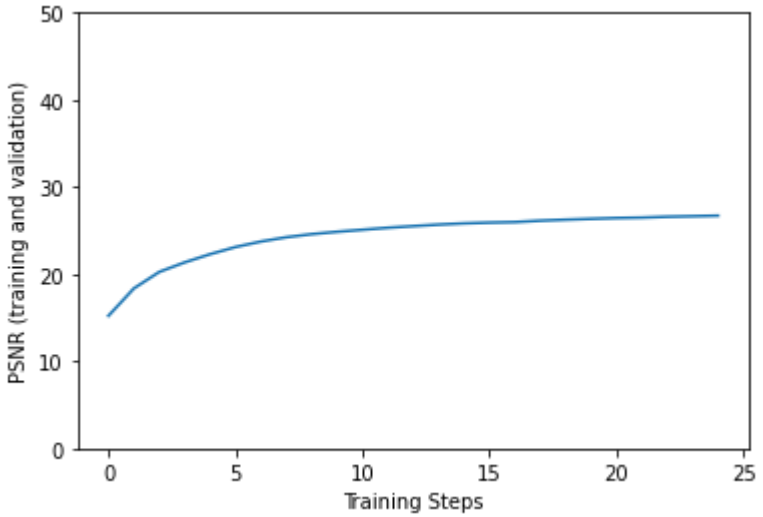


Figure 3.5 - WDSR 8 loss, PSNR & SSIM

WDSR 32





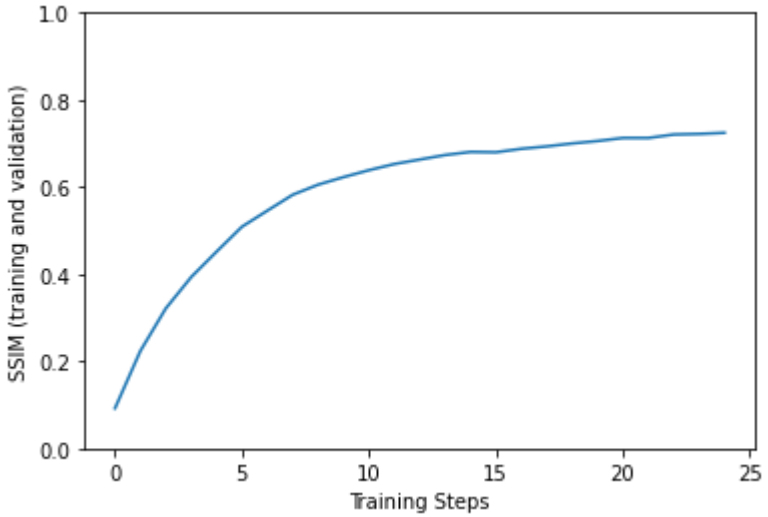


Figure 3.6 - WDSR 32 loss, PSNR & SSIM

3.1.4. Tensorflow Lite Conversion

The Tensorflow Lite converter tool converts a Tensorflow model into an optimized Flatbuffer Tensorflow Lite model. There are two variations of the tool, a Python API and a command line tool. I have used the Python API for the conversion of the models used in this project.

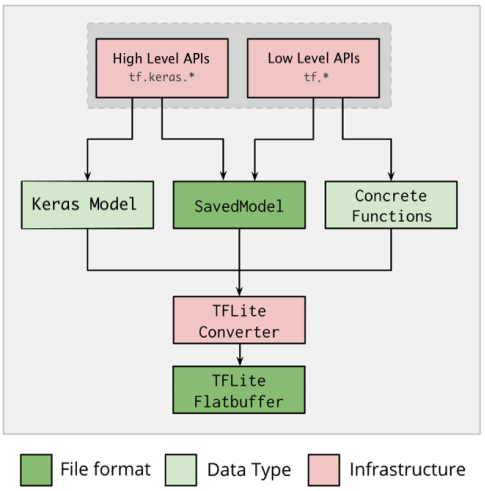


Figure 3.7 - Tensorflow light conversion flow

Mainly for the usage in game development and other performance-sensitive applications, Google first developed the effective cross-platform serialization library known as FlatBuffers. C, C++, C#, Java, Kotlin, JavaScript, PHP, Go, Python, Rust, and Swift are just a few of the languages it supports.

3.1.5. Tensorflow Lite Inference

The steps below are what TensorFlow Lite inference typically entails:

1. Setting up a model

   The execution graph of the model is contained in the.tflite model, which must be loaded into memory.

2. Data transformation

In most cases, the model's raw input data does not come in the format that the model anticipates. For instance, might have to adjust the size or format of an image to make it work with the model.

3. Running inference

   In this phase, the model will be executed using the TensorFlow Lite API. This entails a few stages, including creating the interpreter and allocating tensors.

4. Interpreting output

   When obtaining the model inference results, the tensors must be interpreted in a way that makes sense and is applicable to your application.

For the majority of popular mobile and embedded platforms, including Android, iOS, and Linux, TensorFlow inference APIs are available in a variety of programming languages.

Android platform

Java or C++ APIs can be used to carry out TensorFlow Lite inference on Android. Java APIs can be directly used in your Android Activity classes, and they offer ease. Although the C++ APIs are faster and more flexible, developing JNI wrappers are necessary to transfer data between the Java and C++ levels.

I have opted for C++ approach for performance reasons

3.1.6. Tensorflow Lite Quantization

Post training quantization was tried out on the converted model. This was expected to minimize model size while also enhancing CPU and hardware accelerator latency with little loss in model fidelity. Although the file size for WDSR-B-8 reduced from 5.8MB to 1.5MB the latency did not improve as expected.

4. TEST RESULTS

Given below is a sample of the upscaling results obtained by using the models SRGAN, EDSR and WDSR with a 4X upscaling factor (Models trained with crop transformation).

Figure 4.1 - 4X upscaling with SRGAN, EDSR and WDSR

MSE, PSNR, SSIM were measured for the converted models, SRGAN, WDSR-B, WDSR-B-8. We used the same low-end android mobile device with basic 'Mediatek'

MT6739 : Quad core CPU 1.5 GHz: GPU GE8100 570MHz and 1 GB RAM device which was used for preliminary testing. (Training with crop transformation only)

	PSNR	**SSIM**
Bicubic	20.8965	0.6897
SRGAN	18.5048	0.5674
EDSR	21.3751	0.7441
WDSR-A-8	21.4090	0.7452
WDSR-A-32	21.4079	0.7478
WDSR-B-8	21.4179	0.7479
WDSR-B-32	**21.4185**	**0.7485**

Table 4.1 - PSNR, SSIM values of different models

Out of the given models WDSR-B-32 (with 32 residual) layers showed the best performance in terms of MSE, PSNR and SSIM. However, WDSR-A-32 and WDSR-B-8 were also competing closely with equal MSE values and close PSNR, SSIM values.

However, as per the analysis in Table 3.1, since the run times for WDSR-B-8 seems to be better compared with WDSR-B-32, prior was selected for the model despite the slightly lower accuracy.

After introducing the random flip, random rotate, transformations in addition to the random crop, the WDSR-B models were trained again for 10 epochs to obtain the improved results in Table 4.2.

	PSNR	**SSIM**
WDSR-B-8	**27.4232**	**0.7609**
WDSR-B-32	26.1637	0.7206

Table 4.2 - PSNR & SSIM readings of improved models

The model PSNR, SSIM depends on the data set and method of training as well. Although the WDSR-B-8 appeared to be slightly behind in the earlier case, after retraining appeared to be in the lead. The increase in performance of WDSR-B-8 could be attributed to faster training due to shallower depth with lesser number of residual blocks.

4.1. Mobile Application

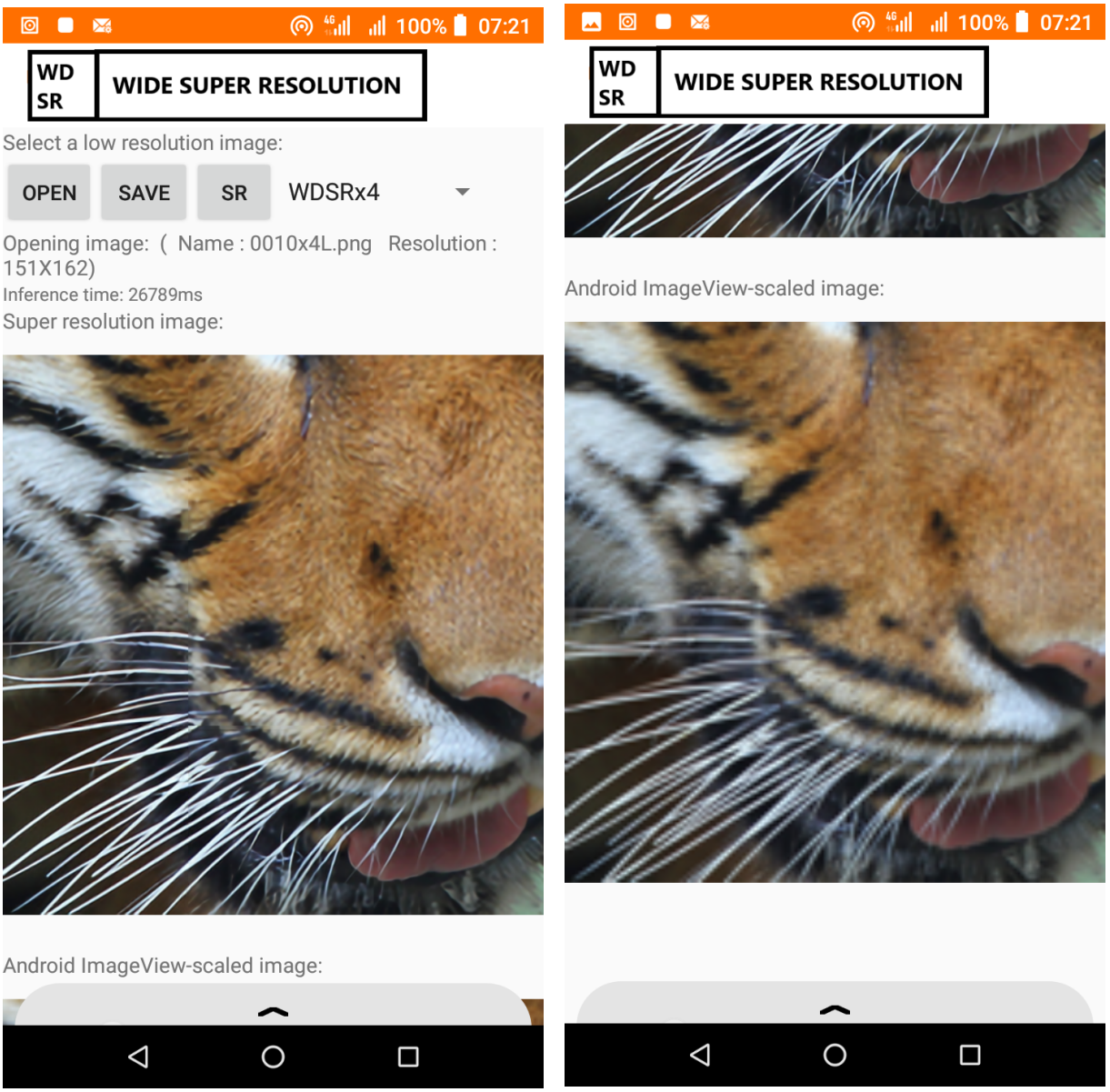


Figure 4.2 - Mobile application preview

5. CONCLUSION & FUTURE WORK

5.1. Conclusion

In this project, I have trained a modified version of the existing WDSR model (WDSR-B-8), and used that with Tensorflow Lite. I have been able to produce a mobile application which can convert low resolution images into high resolution images on-device with acceptable quality and latency. It can also easily be switched to use EDSR and SR-GAN models for upscaling from the interface.

5.2. Future work

- The performance of the WDSR-B-8 model could be improved further to provide a near real time super resolution experience. (User scoring similar to XLSR [45])
- This could be extended for on device video super resolution
- An analysis needs to be done with more devices with focus on how the device architecture affects the model performance.

REFERENCES

[1] N. Ahn, B. Kang, and K.-A. Sohn, “Fast, accurate, and lightweight super-resolution with cascading residual network,” in *Computer Vision – ECCV 2018*, Cham: Springer International Publishing, 2018, pp. 256–272.

[2] O. Alsing, “Mobile Object Detection using TensorFlow Lite and Transfer Learning,” KTH Royal Institute of Technology, Stockholm, Sweden, 2018.

[3] J. Cai, H. Zeng, H. Yong, Z. Cao, and L. Zhang, “Toward real-world single image super-resolution: A new benchmark and a new model,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[4] C. Chen, Z. Xiong, X. Tian, Z.-J. Zha, and F. Wu, “Camera Lens Super-Resolution,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in *Computer Vision – ECCV 2014*, Cham: Springer International Publishing, 2014, pp. 184–199.

[6] N. Georgis, F. Carpio, and P. J. Hwang, “Super-resolution digital zoom,” 8587696, 2013.

[7] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” 2017.

[8] L. N. Huynh, R. K. Balan, and Y. Lee, “DeepSense: A GPU-based deep convolutional neural network framework on commodity mobile devices,” in *Proceedings of the 2016 Workshop on Wearable Systems and Applications - WearSys ’16*, 2016.

[9] A. Ignatov *et al.*, “PIRM Challenge on Perceptual Image Enhancement on Smartphones: Report,” in *Lecture Notes in Computer Science*, Cham: Springer International Publishing, 2019, pp. 315–333.

[10] N. D. Lane *et al.*, “DeepX: A software accelerator for low-power deep learning inference on mobile devices,” in *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2016.

[11] C. Ledig *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[12] R. Lee, S. I. Venieris, L. Dudziak, S. Bhattacharya, and N. D. Lane, “MobiSR: Efficient on-device super-resolution through heterogeneous mobile processors,” in *The 25th Annual International Conference on Mobile Computing and Networking - MobiCom ’19*, 2019.

[13] X. Liu, Y. Li, J. Fromm, Y. Wang, Z. Jiang, A. Mariakakis, S. Patel, “SplitSR: An End-to-End Approach to Super-Resolution on Mobile Devices,” 2021.

[14] H. Ren, A. Kheradmand, M. El-Khamy, S. Wang, D. Bai, and J. Lee, “Real-World Super-Resolution using Generative Adversarial Networks,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

[15] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[16] E. Sert, F. Özyurt, and A. Doğantekin, “A new approach for brain tumor diagnosis system: Single image super resolution based maximum fuzzy entropy segmentation and convolutional neural network,” *Med. Hypotheses*, vol. 133, no. 109413, p. 109413, 2019.

[17] S. Sharma , “AI Can See Clearly Now: GANs Take the Jitters Out of Video Calls,” *https://blogs.nvidia.com/blog/2020/10/05/gan-video-conferencing-maxine/*, 05-Oct-2020.

[18] W. Shi *et al.*, “Cardiac image super-resolution with global correspondence using multi-atlas patchmatch,” *Med. Image Comput. Comput. Assist. Interv.*, vol. 16, no. Pt 3, pp. 9–16, 2013.

[19] Dinh-Hoan Trinh, M. Luong, F. Dibos, J.-M. Rocchisani, Canh-Duong Pham, and T. Q. Nguyen, “Novel example-based method for super-resolution and denoising of medical images,” *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1882–1895, 2014.

[20] X. Wang *et al.*, “ESRGAN: Enhanced super-resolution generative adversarial networks,” in *Lecture Notes in Computer Science*, Cham: Springer International Publishing, 2019, pp. 63–79.

[21] W. Wei, “How to generate super resolution images using TensorFlow Lite on Android,” *https://blog.tensorflow.org/*, 18-Dec-2020. .

[22] W. W. Z. Wilman and P. C. Yuen, “Very low resolution face recognition problem,” in *2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, 2010.

[23] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao, “Deep learning for single image super-resolution: A brief review,” *IEEE Trans. Multimedia*, vol. 21, no. 12, pp. 3106–3121, 2019.

[24] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, “Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018.

[25] K. Zhang *et al.*, “AIM 2020 challenge on efficient super-resolution: Methods and results,” in *Computer Vision – ECCV 2020 Workshops*, Cham: Springer International Publishing, 2020, pp. 5–40.

[26] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, “Image super-resolution using very deep residual channel attention networks,” in *Computer Vision – ECCV 2018*, Cham: Springer International Publishing, 2018, pp. 294–310.

[27] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, “Residual Dense Network for Image Super-Resolution,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[28] F. Zhu and Q. Zhao, “Efficient single image super-resolution via hybrid residual feature learning with compact back-projection network,” in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019.

[29] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, “SwinIR: Image Restoration Using Swin Transformer,” arXiv [eess.IV], 2021

[30] W. Sun and Z. Chen, “Learned image downscaling for upscaling using content adaptive resampler,” arXiv [cs.CV], 2019

[31] K. Zhang, L. Van Gool, and R. Timofte, “Deep unfolding network for image super-resolution,” arXiv [eess.IV], 2020.

[32] A. Mittal, R. Soundararajan, and A. C. Bovik, “Making a ‘completely blind’ image quality analyzer,” IEEE Signal Process. Lett., vol. 20, no. 3, pp. 209–212, 2013

[33] C. Ma, C.-Y. Yang, X. Yang, and M.-H. Yang, “Learning a no-reference quality metric for single-image super-resolution,” arXiv [cs.CV], 2016.

[34] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” arXiv [cs.CV], 2015.

[35] T.-J. Yang, Y.-H. Chen, and V. Sze, “Designing energy-efficient convolutional neural networks using energy-aware pruning,” arXiv [cs.CV], 2016

[36] X. Wang, L. Xie, C. Dong, and Y. Shan, “Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data,” arXiv [eess.IV], 2021

[37] “TVM: An end to end IR stack for deploying deep learning workloads on hardware platforms,” Apache.org, 17-Aug-2017. [Online]. Available: https://tvm.apache.org/2017/08/17/tvm-release-announcement. [Accessed: 22-Feb-2022].

[38] Ncnn: Ncnn is a high-performance neural network inference framework optimized for the mobile platform [Online]. Available: https://github.com/Tencent/ncnn. [Accessed: 22-Feb-2022].

[39] “TensorFlow lite,” TensorFlow. [Online]. Available: https://www.tensorflow.org/lite/guide. [Accessed: 22-Feb-2022].

[40] J. Yu et al., “Wide activation for efficient and accurate image super-resolution,” arXiv [cs.CV], 2018.

[41] Y. Zhang et al., “Cascade wide activation multi-scale networks for single image super-resolution,” in 2019 6th International Conference on Systems and Informatics (ICSAI), 2019, pp. 1222–1227.

[42] C. Wang, Z. Li, and J. Shi, “Lightweight image super-resolution with adaptive weighted learning network,” arXiv [cs.CV], 2019

[43] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, “Enhanced deep residual networks for single image super-resolution,” arXiv [cs.CV], 2017.

[44] M. Krasser, super-resolution: Tensorflow 2.x based implementation of EDSR, WDSR and SRGAN for single image super-resolution.

[45] M. Ayazoglu, “Extremely lightweight quantization robust real-time single-Image Super Resolution for Mobile devices,” arXiv [cs.CV], 2021.

[46] Y. Ioannou, D. Robertson, R. Cipolla, and A. Criminisi, “Deep roots: Improving CNN efficiency with hierarchical filter groups,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.