Self-supervised super resolution of ultrasound images

Yukta Thapliyal

August 2021

Abstract

This paper proposes deep learning models for single image super resolution (SISR) of ultrasound (US) images. The proposed models are self-supervised and aim to perform blind SR. Self-supervision is a class of methods under unsupervised learning where any necessary supervision to the model is provided from input data only. Self-supervision allows the model to predict previously unseen images while avoiding the need for an explicit training phase. Blind SR in images refers to the task of performing super resolution on input images without any prior knowledge or assumption about the source of degradation (if present) and methods utilised in obtaining the input images. The proposed models combine 2D wavelet packet decomposition/transformation (WPD) with convolutional neural networks. WPD divides the input into different sub-bands for analysis. These sub-bands are synthesized through IWPD (Inverse WPD) to reconstruct the signal.

Based on extensive experimentation, the models give a fair performance on the different test cases. In ideal cases, where the source of the image is known, the benchmarking models used in this project perform better. One of the proposed models, which uses the simplest architecture, gives good performance in blind SR settings and is comparable in ideal cases. The second proposed model, which performs super resolution in two parts, gives a slightly lower performance than Model 1 according to evaluation metrics. However, visually, the predicted images by both models are very close.

Acknowledgement

First and foremost, I would like to express deepest gratitude and appreciation for my supervisor, Prof. Ioannis Psaromiligkos. He has been an excellent guide throughout this project and has helped me in finishing this work. He constantly provided me with valuable advice and suggestions that helped in this project. He has also taught me to better analyse experimental findings and work around encountered difficulties. Further, I have learnt to better present my work from him. I began working on my project during the quarantine and for the entire duration, he has been highly understanding, provided me with enough resources and held frequent meetings for discussions. I sincerely thank him for being extremely patient, kind and for also giving me latitude in the project.

I would also like to extend my gratitude towards my family: my father, my mother, and my sister. They have been truly loving and provided their unwavering support throughout my studies and beyond. I am grateful for their presence and encouragement.

Finally, I would like to thank all my friends who have been around. They have been a wonderful presence and helped me stay motivated to complete the project.

Contents

Chapter 1

Introduction

Diagnostic medical sonography, commonly known as ultrasound (US) is one of the most popular diagnostic imaging modalities. There exist other methods like X-ray, CT (Computed Tomography) scan, and MRI (Magnetic Resonance Imaging), but ultrasound is considered the safest amongst them. Compared to X-ray and CT scan, US is safe from exposing ionising radiation to patients and physicians. Though MRI is also protected from ionising radiation, its powerful magnets pose a risk to patients with metal implants. US also offers real-time patient examination. Ergo with immediate results, the doctor can even prescribe treatment to the patient in the same appointment. Other distinct advantages of ultrasound include being portable and low cost in comparison to the other three modalities. For these reasons, ultrasound is often considered the first choice in many diagnosis methods, especially in pregnancy.

The basic mechanism of ultrasound involves a transducer at its core. Within the transducer, a piezoelectric element produces sound pressure waves, typically in the range of 2-18 MHz. Lead zirconate titanate (PZT) crystal is the most popular choice for the piezoelectric element in medical ultrasound. The high-frequency ultrasound waves are emitted from a handheld probe and are focused on the region of interest (ROI) along the scan lines. The ROI is the organ or any other tissue(s) that needs to be scanned. These waves are allowed to pass through the skin, into the body and are constructively added at the focal point. Now as these sound waves are propagating within the body, they encounter different boundaries and a portion of these waves echo back into the transducer. As the reflections gather at the transducer, new sound waves are generated and emitted towards another focal point along the scan line. In this manner, as all the reflections along a single scan line are assessed, the ultrasound system jumps to a new scan line. This process continues until all the scan lines in an ROI are measured. At the surface of the skin, as shown in Fig 1.1, the intensity reflection coefficient α, which measures the ratio of reflected sound wave intensity to initial sound wave intensity is close to one. This concludes the fact that the majority of the ultrasound waves at the surface of the skin will be reflected back to probe. Therefore, in practice, a gel is applied on the skin surface that has similar acoustic impedance as skin. This is called impedance matching. After successful impedance matching on the skin surface, the sound waves encounter different tissues inside the body. These tissues have different impedance. For example, bones

being denser than fat will have a lower impedance than fat. Thus, reflection intensity will be different for different tissues or boundaries within the body.

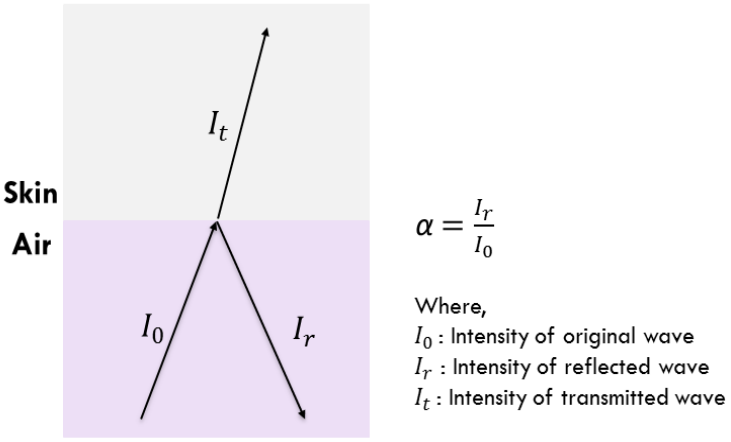


Figure 1.1: Ultrasound waves when probe comes in contact with skin. The acoustic impedance is given by $Z = \rho\, v$, where ρ is the density of medium through which the sound travels and v is the speed of the sound through the medium. We have, Z_{air}(Impedance of air) $= 1.2$ kg m $^{-3}$ x 343 m s^{-1} $= 0.0004$ x 10^6 kg m^{-2} s^{-1} and Z_{skin}(Impedance of skin) $=$ 1100 kg m^{-3} x 1537 m s^{-1} = 1.69 x 10^6 kg m^{-2} s^{-1}. Therefore $\alpha = (\frac{Z_{\text{skin}} - Z_{\text{air}}}{Z_{\text{skin}} + Z_{\text{air}}})^2 = 0.999$. It is to be noted that the ρ and v of air and skin here are representative values.

During this entire process, there are many sources of degradation that can seep into US images. One prominent source of degradation is speckle noise. Speckles are an inherent property of ultrasound that is caused by the scattering of waves. Scattering occurs when an ultrasound wave encounters objects having a size similar to their wavelengths. These scattered waves then interact depending on their phase and position and may be subjected to either constructive or destructive interference. Besides speckle noise, there is electronic noise, noise caused by de-focusing of beam, and more. Therefore, after scan conversion (transforming raw data to display form) US systems perform other processing steps like angle compounding, frame smoothing, boundary detection [1] to obtain quality US scans. Besides these steps, other clinical grade post-processing procedures are applied on the US scans to get final scans of improved quality before medical professionals use these scans.

1.1 Motivation and objective

Medical professionals often prefer to view post-processed US scans for diagnosis and unfortunately, these post-processed scans are an output of clinical grade scanners that use proprietary methods. Emulation of these techniques is difficult without access. Therefore, there has been research in recent times to find alternate methods for post-processing US images that have high enough quality that medical professionals could use. One such method is super resolution (SR). SR creates high resolution (HR) images from its low resolution (LR) counterpart.

Previously DL based methods have been successfully used for SR of natural images as discussed in Section 2.2.3. These methods mostly utilize and enhance convolutional neural network (CNN) as they are proficient in analyzing

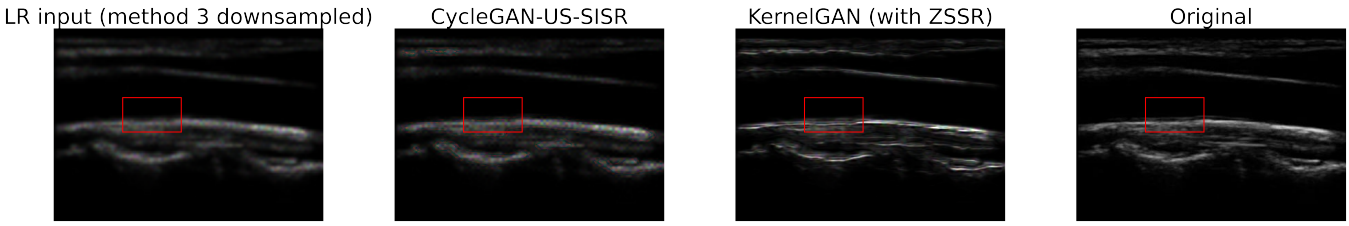
visual imagery. Amongst these models, general adversarial network (GAN) [2], discussed in Section 3.3, and its many variants are becoming increasingly popular in SR of natural images. GAN is an unsupervised model that is capable of generating synthetic images that resemble the data distribution of input images. Taking inspirations and viewing advantages of DL networks and CNNs, lately, there has been a surge in using them for SR of US images. Some of these methods and models are discussed in Section 2.2.3. For SR of US images, SR models designed for natural images cannot be directly used owing to the difference in characteristics of US images from natural images. US images inherently have speckles and other noise elements in them. Further, there is a wide availability of natural image datasets having images of different sizes and quality, while US image datasets are limited. It has been observed in this project that models having a simpler architecture and complexity work better for US images, in comparison to natural images where complexity does not degrade the performance. Further, US images require a separate data augmentation step, discussed in Section 4.3, prior to the training step that is not necessary for natural images. The objective of this project is to provide a DL-based model for SR of US images. The contribution of this project can be summarised as follows:

- Reviewing and summarising CNN-based works and methods used in natural image SR and US image SR.
- Propose self-supervised models for single image super resolution (SISR) of US images that combine methods from DL and wavelets from signal processing that perform well on a dataset having images created from 'non-ideal' or unknown methods.

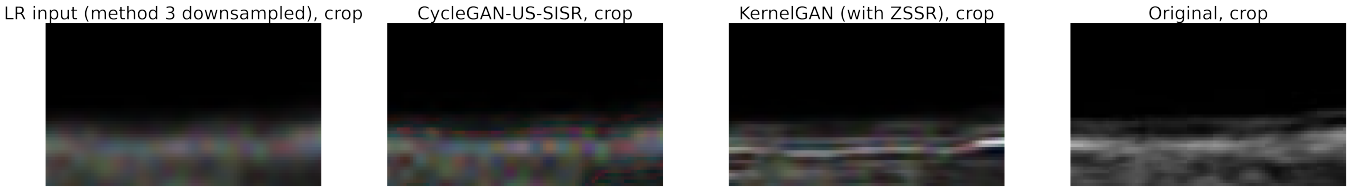
1.2 Challenges

The main challenge in SR of US is associated with the dataset. Firstly, due to privacy reasons, the variety and number of public datasets available are limited in comparison to other ML problems. Secondly, these public datasets sometimes do not disclose the complete information on acquisition methods and other properties of the US scans. Besides acquiring datasets, there is also no practical existence of $HR - LR$ tagged US scan images for training the models. Therefore, often when public datasets are used for model building, data augmentation methods are performed. The size of US scans can also sometimes cause problems in performing SR at large scales. Small size images do not hold enough information in them which could be used to learn a mapping to SR domain. For example, in upscaling a small US image by a factor of larger scales, such $\times 4$, other artificial artifacts and other cosmetic changes added into the SR image as can be seen in Fig.1.2b.

The other challenge is the lack of access to clinical-grade processors. Acquiring clinically post-processed US scans that depict the same scene as the predicted US scan (by the DL model) is difficult. Therefore, it is hard to determine if these predicted scans are up to par with the quality that medical professionals would consider. Most of the bench-marking in literature and also this project is done against other DL-based models and not against actual clinically post-processed scans. Deciding a loss function is also a difficult task in US SR. In absence of a reference HR



(a) A US LR input image is predicted using KernelGAN (with ZSSR) (defined in Section 3.2.5) and CyleGAN-US-SISR (defined in Section 3.3) and the original image. The Method 3 of donwsampling is defined in Section 4.5.2. It refers to $I_{LR} = ((I_{HR} \downarrow_{(lan3)} s) * (f_g) + ((I_{HR} \downarrow_{(lan3)} s) * f_g) \times n_r$ where the input image I_{LR} has been derived from original image, I_{HR} by first downsampling through a lanczos kernel ($lan3$), then passing through a gaussian kernel (f_g) and finally adding a speckle noise



(b) Here, a difference in original image and predicted image can be seen.

Figure 1.2: Noise and other artificial artifacts in predicted image for case of $X4$ upscaling.

image, it becomes difficult to decide on a loss criterion which optimises the model

1.3 Outline of the document

The rest of the document contains five chapters. Chapter 2 (Literature review) provides the literature review on DL-based SR of natural images and US images. Chapter 3 (Background) gives the description of the techniques and methods that are aligned with the literature review and are needed to understand the proposed model in the project. Chapter 4 (Methodology) gives the design and methods that form the proposed model and any prior assumptions. Chapter 5 (Experimental results and discussion) gives an analysis of experiments and discusses the findings. It also discusses the limitations of the project. Chapter 6 (Conclusion) concludes the project with a summary, implications of the project and gives possible future suggestions.

Chapter 2

Literature Review

This chapter begins with a brief introduction to the type of SR problem the project will be dealing with, i.e., single image super resolution (SISR). It then reviews the early DL models for natural image SISR to the current popular models. Lastly, it reviews SISR models and methods used in US images.

2.1 Single image super resolution of natural images

Super resolution (SR), in an imaging system, refers to the restoration technique of producing high resolution (HR) images or videos from its low resolution (LR) counterpart(s). For image SR, there exists multi-image super resolution (MISR) and single image super resolution (SISR). MISR creates the HR image from multiple LR images of the same scene and SISR obtains the HR image from a single LR image. Numerous approaches have been proposed in SISR and can be broadly classified into 4 categories [3]. These traditional methods include:

- Prediction-based methods which predict the image in HR space using a mathematical formula. Interpolation based methods, like bicubic, bilinear, wavelet-based interpolation, etc., are a popular example in this category,
- Edge based methods which focus on edge information for reconstructing HR images, like in [4],
- Statistical based methods, eg., [5], where certain image properties are used as priors for prediction of HR images from LR images; and lastly,
- Learning based methods which start with a pair of HR - LR images and attempt to find a mapping function that tries to learn a relation between LR space and HR space. For example, using weighted average as a learning function as in [6].

With the advent of neural networks (NNs) and their rising popularity in the past decade, deep learning (DL) has also ventured in SISR domain. These methods can be broadly classified into four categories: Traditional CNN

(Convolutinal Neural Network) based, non-traditional CNN based, GAN (General Adversarial Network) based and self-supervision.

2.1.1 Traditional CNN-based SISR

DL in SISR began with traditional CNN based models. These models have a simple architecture, mainly a conventional CNN with very few changes. In 2014, Dong et al. were the first ones to introduce this method. Their very popular network, SRCNN (Super Resolution Convolutional Neural Network) [3], obtained SISR by performing three non-linear operations of *i,)* patch extraction to extract a set of feature maps from LR image; *ii,)* non-linear mapping of these features to HR patch representations; and *iii,)* reconstruction of HR image corresponding to LR image. Based on SRCNN, FSRCNN (Faster SRCNN) [7] was published and as the name suggests, it is a faster SRCNN. FSRCNN architecture introduces a shrinkage layer after the feature extraction. The shrinkage layer reduces the number of feature maps. The feature maps are increased again to the previous count after the non-linear mapping layer and the final deconvolution layer obtains the HR image. Next, Kim et al. gave the first very deep CNN, known as VDSR (Very Deep Super Resolution) [8] which is based on popular VGG-Net [9]. VDSR takes advantage of an increased depth of 20 layers in the network and produces good result. Like SRCNN, VDSR also takes an interpolated (bicubic interpolation) LR image as input. However, unlike the above-mentioned models, to obtain the final HR image in VDSR, the residual of the deep network is added to the interpolated image. Therefore, the network learns residual errors between input interpolated LR image and output HR image.

2.1.2 Non-traditional CNN-based SISR

Non-traditional CNN here refers to networks that have CNN at core of their model. However, certain additions/modifications are made to conventional CNN for SISR. One typical modification is dropping the use of interpolated methods to upscale the LR image. Additionally, in most traditional CNN based SISR, the resolution of LR image is increased at the beginning, before or after first layer. This increases the load of the model for training. Models based on non-traditional CNN avoid carrying out the same practice. Some of the models in this category are discussed below.

Wavelet based

Wavelets [10] are specifically generated wave like oscillations that have a short life and zero mean. In wavelet analysis of images, low frequency and high frequency images, also called sub-bands, are created. Low frequency images are created by passing the image through a low pass filter (LPF)/blurring kernel/smoothing kernel. Similarly, an image is passed through a HPF to create sharper images where difference in intensity is more pronounced. A very popular application of wavelets is image compression. WDRN/WavResNet (Wavelet-based Deep residual Learning Network) [11] and MWCNN (Multi Level Wavelet CNN for Image Restoration) [12] use wavelet transformation for restoration tasks (denoising, super-resolution and artifacts removal).

Recursive layer based

Authors of VDSR also give DRCN (Deeply Recursive Convolutional Network) [13]. DRCN is similar to VDSR in depth, however, it introduces a recursive layer [14]. The input to the recursive layer is the feature maps produced after first stage of encoding.

Sub-pixel convolution network

In 2016, Shi et al. proposed ESPCN (Efficient Sub Pixel Convolutional Neural Network) [15] which enhances SRNN. ESPCN avoids taking an upsampled interpolated LR image as input to the network. It performs upsampling of the LR image within through sub-pixel convolution layer (discussed in Section 3.3). The sub-pixel convolution layer forms the last layer layer of the network, therefore as upsampling is happening towards the end of the network, the network saves computational load in comparison to the other approach of taking bicubic upsampled image.

Residual based

Though VDSR and DRCN use a skip connection to add input LR image to the residual of the network, large scale residual units were first exploited in SRResNet [16]. SRResNet is based on popular ResNet [17] and has replaced simple convolution layers with 16 residual blocks. For increasing the resolution of mapped SR image, two sub-pixel convolution layers are used after residual units. EDSR (Enhanced Deep Super-Resolution Network) [18] given by Lim et al. makes a number of modification on SRResNet. For example, removing Batch Normalisation within residual units of SRResNet. MSDR (Multi-Scale Deep Super-Resolution System) [18], also given by authors of EDSR, is multi-scale version of EDSR.

Miscellaneous

A mix of residual and recursive methods is used in DRRN (Deep Recursive Residual Network) [19]. DRRN uses stacked residual blocks instead of individual residual layers. These blocks contain recursive convolution within them, thereby effectively performing local residual learning in multiple paths.

LapSRN (Laplacian Pyramid Super Resolution Network) [20] given by Lai et al. has two parallel branches for feature extraction and image reconstruction. LapSRN gets its name from Laplacian Pyramid image processing technique [21] on which it is based. Further, instead of usual MSE loss function, LapSRN uses Charbonnier loss function [22]. Charbonnier loss is also called pseudo Huber loss or L1-L2 loss. Huber loss is preferred in models where sensitivity to outliers is needed to be relaxed. Directed by a hyperparameter δ, Huber loss can be either quadratic (MSE), where error is small, or absolute (MAE), when error is large. Huber loss is defined as

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2, & \text{if } \|y - f(x)\|_1 \leq \delta \\ \delta^2(\|y - f(x)\|_1 - \frac{1}{2}\delta), & \text{otherwise} \end{cases}$$

Pseudo Huber loss, on the other hand, combines both MSE and MAE in one equation and allows to set a specific penalty on outlying data. It is given as $L_\delta(y, f(x)) = \delta(-1 + \sqrt{1 + (\frac{y-f(x)}{\delta})^2})$. Ms-LapSRN (Multi-scale SRN) [23] is similar to LapSRN with parameter sharing across all pyramid levels and local residual learning.

SR+STN (Spatial Transformer Network)[24] given by Lu et al. is a relatively new and different approach to super resolution. It works by trying to find similar patches within a single image by KNN (K-Nearest Neighbour) [25]. These patches are then aligned by STN, which was introduced as an image classification method by DeepMind, Google. The knowledge accumulated by these aligned patches then form the HR image.

2.1.3 GAN-based SISR

GAN [2] (discussed in the next section) was given in 2014 as an unsupervised image and audio waveforms reconstruction model. Since then, GAN has been a source of inspiration for many different problems, such as style transfer of image or super resolution. GAN was first used in SISR of natural images in a model named SRGAN [26]. Though SRGAN outputs images with lower PSNR/SSIM in comparison to above methods, the perceptual quality of these images is higher, as reflected in its higher MOS (Mean Opinion Score). This is due to the use of perceptual based loss in its architecture. All of the above discussed methods use l_1 or l_2 or a combination of both as their loss function. These losses are solely distortion based and can give perceptually weird/poor results. It was after SRGAN that other models began to use perception-based losses in SISR. Similar to SRGAN is ESRGAN (Enhanced SRGAN)[27] which gives sharper SR reconstructed images.

2.1.4 Self-supervised learning (SSL)

SSL refers to unsupervised learning methods where the input data itself automatically generates some kind of supervisory signals to learn unobserved patterns or other properties in the data representation. There is no human annotated data provided to the model at any phase of its learning and thus the model learns on internal characteristics of the image. For example, one such method used in SISR is ZSSR [28]. In ZSSR, the model reconstructs a SR image without any prior training with external examples. The test image or examples generated from test image itself are input to an image-specific CNN, which learns the internal patch distribution and priors of the image to super resolve it. This type of learning can super resolve images encountered for the first time better than supervised learning based models.

The authors of ZSSR further give KernelGAN[29], which uses ZSSR and GAN for blind SR of natural images. Blind SR is a problem where the LR images are obtained by convolution of HR images with unknown kernels and optional addition of unknown noise. In real world, LR images are not synthetically obtained from a known kernel. Therefore, most state-of-the-art (SoTA) methods which train on LR images produced from known kernel, drop their performance when super resolving LR images that are generated from random/unknown kernels. KernelGAN finds a solution to this problem by predicting image specific downsampling kernel. The image-specific kernel is learned through a linear

FCN generator and the PatchGAN [30] discriminator. The discriminator tries to distinguish between patches from cropped real image and generated LR image. Once a suitable kernel is learned, image examples from the input image are fed as input to ZSSR for super resolution.

Paper	Year	Network	Outperforms based on evaluation metric
Dong et al. [3]	2014	SRCNN	-
Dong et al. [7]	2016	FSRCNN	SRCNN based on PSNR
Shi et al. [15]	2016	ESPCN	SRCNN based on PSNR
Kim et al. [8]	2016	VDSR	SRCNN based on PSNR/SSIM
Kim et al. [13]	2016	DRCN	SRCNN based on PSNR/SSIM
Shocher et al. [28]	2017	ZSSR	EDSR based on PSNR/SSIM
Bae et al. [11]	2017	WDRN/WavResNet	SCRNN, VDSR, DnCNN [31] based on PSNR/SSIM
Tao et al. [19]	2017	DRRN	SRCNN, FSRCNN, ESPCN, VDSR, DRCN, RED-Net [32] based on PSNR/SSIM
Ledig et al. [16]	2017	SRResNet	SRCNN, DRCN, ESPCN based on MOS of 26 raters, PSNR/SSIM
Ledig et al. [16]	2017	SRGAN	SRCNN, DRCN, ESPCN, SRResNet based on MOS of 26 raters
Kang et al.[18]	2017	EDSR & MDSR	SRCNN, VDSR, SRResNet based on PSNR/SSIM
Lai et al. [33]	2017	LapSRN	SRCNN, FSRCNN, VDSR, DRCN based on PSNR/SSIM/IFC
Liu et al. [12]	2017	MWCNN	IRCNN, VDSR, DnCNN, RED-Net, SRResNet, LapSRN, DRRN, MemNet [34], WavResNet based on PSNR/SSIR
Wang et al. [27]	2018	ESRGAN	performance similar to SRGAn based on PSNR/SSIM
Lai et al.[23]	2018	Ms-LapSRN	SRCNN, FSRCNN, VDSR, DRCN, DRRN, LapSRN based on PSNR/SSIM/IFC
Lu et al. [24]	2019	SR+STN	SRCNN, VDSR based on PSNR/SSIM
Bell-Kligler et al. [29]	2019	KernelGAN	blind SR algorithms form NTIRE' 2018 challenge [35]

Table 2.1: Comparison of different DL based SR networks. The comparison in the last column is based on information provided in the respective papers. It is to be noted that the performance assertion is based on different datasets in different papers. The evaluation metrics are PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index), IFC (Information Fidelity Criterion) and MOS (Mean Opinion Score).

2.2 Deep learning in super resolution of medical ultrasound

DL-based SR of natural images is a fairly new practice. Given that and the inherent property of ultrasound imaging that includes speckle noise, electronic noise upon reconstruction, blurring, and other artifacts; there is a lack of abundance in DL-based US-SR in literature, especially in post-processing techniques. Most of these methods use publicly available datasets to try to super resolve the US scans. Of course, one of the aspects of these methods is to try

to emulate clinical-grade processors, but the other important aspect is to diversify the areas in which ML could be used. Mainly these methods are learning-based, where they learn a non-linear LR-HR relationship to create US scans from LR space to HR space. Within these methods, GAN is observed to be favoured for reconstruction characteristics. Benchmarking of US SISR models discussed below is mostly done against SoTA methods from natural SISR as there is a lack of standardized methods in DL-based US SISR.

2.2.1 GAN-based US SISR

Since its invention in 2014, GANs have actively been employed in image processing. In US post-processing, initially, GANs were used for despeckling, like in DRNN [36]. While not exactly an SR model, DRNN aimed to enhance US image quality by removing speckle noise using GAN. It employs a ResNet-based generator and FCN discriminator. Compared to de-facto despeckling algorithms, DRNN achieved higher SSIM and PSNR without losing characteristic details. Vedula et al. [37] proposed another despeckling algorithm-based network to reconstruct ultrasound images. Their network reconstructs tissue images akin to "CT quality". While evaluation based on PSNR, their network outperforms a standard despeckling algorithm (TV despeckling algorithm). Other examples of such image enhancement using speckle reduction include GAN style network in [38] and [39].

One of the early uses of GAN for US SISR can be seen in [40], where Choi et al. employ SRGAN. Using 171 frame images acquired from 19 US videos, they modify the original SRGAN by reducing the number of residual layers and making changes in generator SRResNet. Tests on 665 blurred sub-sampled images were performed and their network performed better than SRResNet.

In [41], another GAN-based network using PatchGAN discriminator and a multi-scale encoder-decoder as the generator is used for US SISR. To overcome over-smoothing at final output, 4 types of losses are used. Trained on 5760 images from public datasets (CCA-US [1] and US-CASE [2]) and tested on 640 images, their network gives fair results on PSNR, SSIM, & IFC index over SRCNN and SRGAN. Their model provides a good trade-off between reconstruction accuracy and visual similarity to real ultrasound data.

In [42], CycleGAN [43] and ZSSR based network is proposed for SISR of US. The framework considers consistency between LR-HR pairs. The generator architecture is taken from the model used in [41], as some of the authors are the same. They use a multi-scale encoder-decoder for LR $\rightarrow$ HR transformation and a specifically formulated CNN for regression of HR $\rightarrow$ LR transformation. A cycle loss is introduced that administers consistency of LR $\rightarrow$ SR $\rightarrow$ LR and HR $\rightarrow$ LR $\rightarrow$ SR transformations. Based on public dataset evaluation, their network was able to achieve higher PSNR, SSIM & IFC over SRCNN, SRGAN, EDSR, SRFeat[44] and ZSSR[28] (with the exception of ZSSR having a higher PSNR in one of the two datasets). Another CycleGAN inspired network is MimickNet[45]. MimickNet takes DAS (Delay and Sum) beam data as input and outputs US images with quality comparable to DTCE^{TM} (Dynamic Tissue Contrast Enhanced), a trademarked clinical-grade post-processing algorithm. It is the first attempt, as per knowledge,

[1] http://splab.cz/en/download/databaze/ultrasound
[2] https://www.ultrasoundcases.info/

at mimicking clinical-grade post-processing US scan quality.

2.2.2 CNN-based US SISR

One of the simplest US SISR architectures is using traditional CNNs. CNNs are effective in capturing feature information from images. DECUSR (Deep Convolutional Network for Ultrasound Super Resolution)[46] is a 17 layer deep CNN. The simple architecture can be divided into 3 stages: feature extraction block, densely connected repeating blocks (RBs), and upsampling layers. The feature extraction block is a usual encoder, based on FCN [47]. Each RB embodies convolution and concatenation layers. The upsampling of LR input is carried out before stacked RBs with two layers; one directly upsamples the LR image and the other upsamples after the feature extraction block. DECUSR is compared to interpolation methods, SRCNN and EDSR on 8 different metrics including SSIM & PSNR and observes favourable results.

USSR (Unsupervised Super Resolution) [48] is another self-supervised network that came in the same year as ZSSR. Its technique is also similar to ZSSR. However, USSR uses dilated convolution in their image-specific CNN. The network, USSR (Unsupervised Super Resolution), is essentially unsupervised in the sense that they do not perform any prior training on any external data but only consider the test data. Like ZSSR, there is a formulation of a specific CNN to understand the relation of the LR-SR pair of test image before obtaining SR ultrasound. The whole model can be divided into three stages: data processing, image-specific CNN design, and USSR. Investigating the model on brachial plexus, cardiac, and brain ultrasound images, their model fared better than unsupervised SelfExSR[49] in terms of PSNR and SSIM.

2.2.3 U-Net based US SISR

U-Net [50] is an encoder-decoder-based FCN originally designed to perform biomedical image segmentation. U-Net has recently been employed alongside algorithms from traditional SR US imaging. A very good example of this is seen in paper [51] by van Sloun et al. Their study uses deep learning to perform traditional ultrasound localisation microscopy (ULM). ULM is an optic-based traditional ultrasound imaging technique to give quality scans. The network, named deepULM, is an end-to-end reconstruction model for obtaining high resolution high concentration contrast enhanced (CEUS) images. Although deepULM gives exceptional results in SR-US, it involves injection of contrasting agents (CAs) into the body for CEUS images. This can discourage some patients, especially pregnant women who undergo obstetric ultrasound to monitor the development of their fetus. Inspired by deepULM is DL-SRU (Deep Learning based Super Resolution Ultrasound) [52] which avoids the injection of CAs. DL-SRU is a U-Net like architecture and combines with a PTV (Particle Tracking velocity) [53] algorithm, a method to evaluate the trajectory of moving objects. Both these models are trained on synthetically produced data but evaluated on *in-silico* and *in-vivo* studies on animals and humans.

Paper	Year	Network	Underlying DL model
Lu et al.[48]	2018	USSR	Dilated CNN
Choi et al.[40]	2018	-	SRGAN
Huang et al.[45]	2018	MimickNet	CycleGAN
Mishra et al[36]	2018	DRNN	GAN
Dietrichson et al.[38]	2018	-	GAN
Zhang et al. [39]	2018	-	GAN
Sloun et al.[51]	2019	deepULM	U-Net
Liu et al. [41]	2020	-	PatchGAN + CNN
Liu et al. [42]	2020	-	CycleGAN + ZSSR
Temiz et al.[54]	2020	DECUSR	CNN
Park et al. [52]	2020	DL-SRU	U-Net

Table 2.2: Different papers based on deep learning ultrasound quality improvement. It should be noted that 4^{th} column only mentions the deep learning aspect, some of these papers have other algorithms that are integrated with DL algorithms to carry out operation mentioned in column.

Chapter 3

Background

This chapter gives a brief overview of the theoretical foundations of this project. It begins with a brief description of various components that make up an ML model. Thereafter, some state-of-the-art models used in SR of natural images are discussed. The chapter also lists some of the models that have been used in US SISR. Finally, a US SISR model is discussed in detail for a better understanding of the US SISR framework.

3.1 Components of a ML algorithm

There are four basic elements of a ML algorithm that form the crux of any ML task. These are: *i,)* ML model, *ii,)* cost function and optimization algorithm, *iii,)* dataset, and *iv,)* performance analysis.

3.1.1 Model

The ML model is at the core of a machine learning task. The goal of an ML model is to recognize patterns and learn other specific information about the data that is fed into it. These models are specific to the type of task that is to be performed. For example, a task of sentiment analysis may use models based on SVM (Support Vector Machine) [55]. Similarly, in US SISR, the task is to super resolve low resolution ultrasound images. This problem also falls under the broader and inter-disciplinary category of computer vision (CV). Computer vision is a field under artificial intelligence (AI) in which a computer/machine tries to gain an understanding of digital images akin to human visual images. For US SISR, DL is at the forefront of popular methods. DL refers to a family of methods, under the branch of ML, that use 'deep networks'. These deep networks, inspired by the biological neural network, use a large number of artificial nodes/neurons in multiple layers and form an artificial neural network (ANN). For image processing tasks, a class of sub-networks under DL called convolutional neural networks (CNN) are widely popular. The hierarchy of AI, CV, ML, ANN, and CNN can be understood from Fig. 3.1.

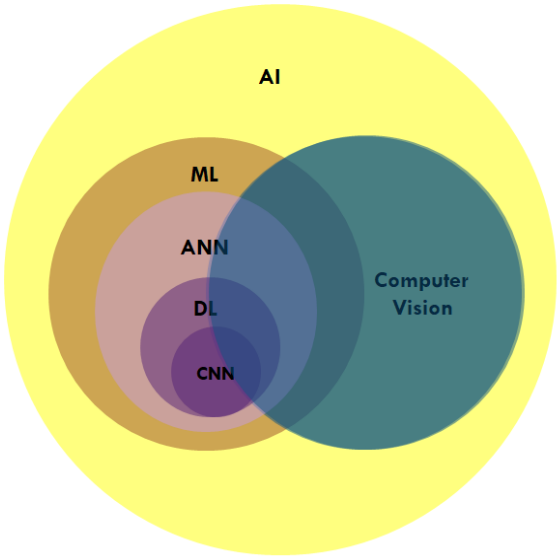


Figure 3.1: A relationship between different fields in Artificial intelligence

CNN

In a traditional feed-forward NN, each neuron in the hidden layer h is connected to every neuron in the previous $(h-1)^{th}$ layer and next $(h+1)^{th}$ layer. Therefore, in the case of image analysis, where an image has a large number of pixels, the learnable parameters, i.e., weights and biases, could reach up to millions for a single layer. This in turn significantly increases the training time and complexity. Convolutional neural networks, which are a regularised version of ANN are hence used for images. The fundamental principle behind CNN is to connect a patch of neurons, instead of all neurons, in the current hidden layer h to a patch of neurons in the next $h+1$ layer. The trainable parameters are shared across the patches for a particular hidden layer. Some fundamental concepts of a CNN are discussed below.

- **Traditional convolution and kernel**:

  Kernel/filter is an essential element of a CNN. A kernel is a matrix that is slid across patches of input image or output of a hidden layer to give a representation of reduced dimensionality (latent representation). The process is called convolution, which is the dot product of the kernel with the patches. Mathematically, the convolution operation in CNN is a cross-correlation operation which is defined as:

$$(I * k)[r, c] = \sum_j \sum_i k[j, i] I[r - j, c - i] \tag{3.1}$$

  where I is the input image, k is the kernel, and r and c denote the row and column index. The movement of the kernel over the image can be controlled by determining the stride hyperparameter, given by s. A stride of 1 means that the kernel will slide by 1 pixel at a time. Naturally, as the stride of the kernel increases, the output dimension of the particular hidden layer decreases. The operation can be seen in Fig. 3.2. In a

convolutional layer, there can be multiple kernels applied. Each kernel learns some particular feature from the image representation.

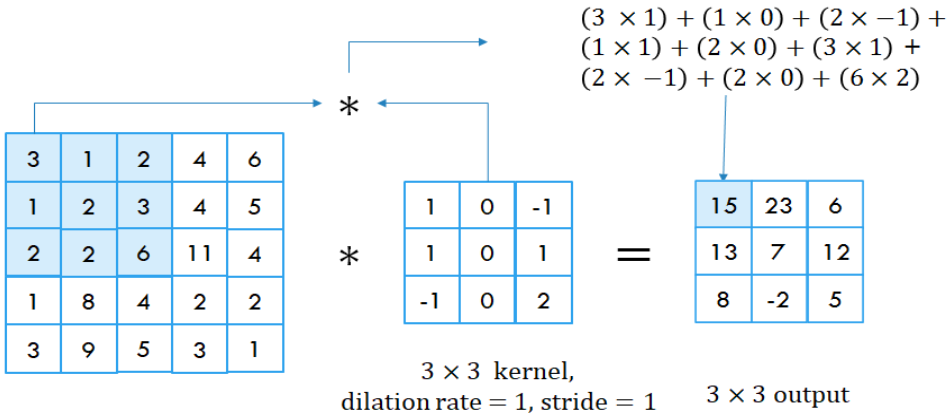


Figure 3.2: Traditional convolution operation

- **Zero padding**:

  A hyperparameter of the convolution layer is padding. Zero padding introduces empty pixels of value 0 to the borders of the image. A zero padding of p means that p empty pixels are introduced along the boundary of the image. Padding can be seen in Fig. 3.3. Padding can control the dimension reduction of the image. For example, consider a symmetric input of $n_i \times n_i$ pixels and a convolution layer with $k \times k$ kernel and p padding. Let the output dimension be $n_o \times n_o$. Then n_o can be calculated as:

$$n_o = 1 + \frac{n_i - k + 2p}{s} \tag{3.2}$$

- **Dilated convolution**:

  Dilated or atrous convolution [56] is a convolution operation with an 'inflated' kernel. The kernel is inflated by inserting holes in between kernel elements. Hyperparameter dilation rate, l, determines how much the kernel is inflated, in particular, $l - 1$ spaces are inserted between the kernel elements. Dilated convolution is given as:

$$(I *_l k) = \sum_j \sum_i k[j, i] I[r - lj, c - lj] \tag{3.3}$$

  where r and l are row and column indexes. For an input of $h_i \times w_i$, a dilation convolution will output $h_o \times w_o$

as following:

$$h_o = \lfloor 1 + \frac{h_i + 2p - l \times (k-1) - 1}{s} \rfloor \tag{3.4a}$$

$$w_o = \lfloor 1 + \frac{w_i + 2p - l \times (k-1) - 1}{s} \rfloor \tag{3.4b}$$

Dilated convolution operation can be seen in Fig. 3.3. A dilated convolution is advantageous in increasing the receptive field without increasing the size of the kernel. The receptive field measures the extent to which a neuron will capture the input data. As the kernels have holes in them, using dilated convolution in high resolution images can give good results. However, with lower resolution images, dilated convolution is not usually recommended.

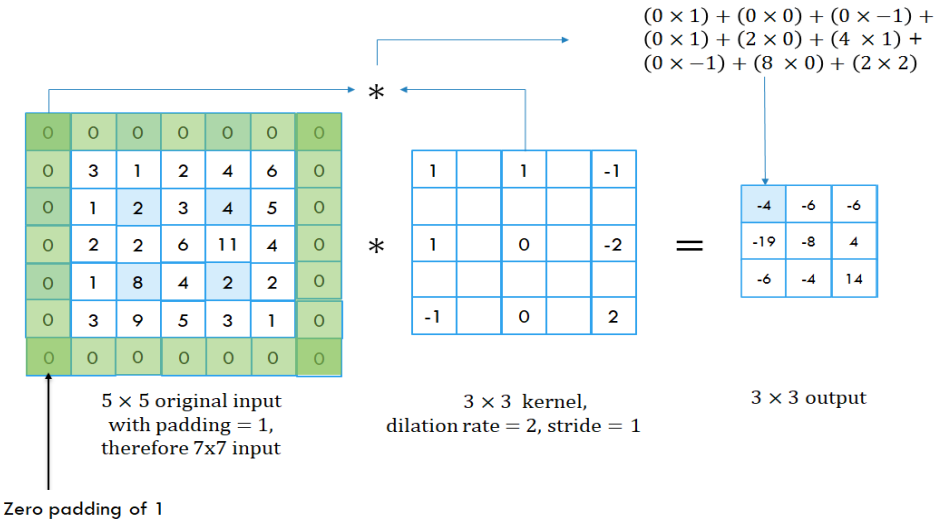


Figure 3.3: Dilated convolution operation

- **Transposed convolution**:

  Transposed convolution is an operation that upsamples the input feature map. Traditional convolution operation decreases the spatial representation of input as the number of layers increases. While this operation is very useful in capturing higher level features in tasks like image classification, the SR task is quite different. The output image needs to be of higher resolution than input. Therefore, transposed convolution is frequently used to upscale the latent representation of the image. Transposed convolution slides a learnable kernel over the image to upscale the image as in Fig. 3.4.

- **Sub-pixel convolution or pixel-shuffler**:

  Sub-pixel convolution is another upsampling technique. When upsampling images with transpose convolution, NNs often observe a checkered pattern within the images. This artifact is called checkerboard artifacts. The authors of ESPCN, where sub-pixel convolution was introduced, contest that sub-pixel convolution eliminates

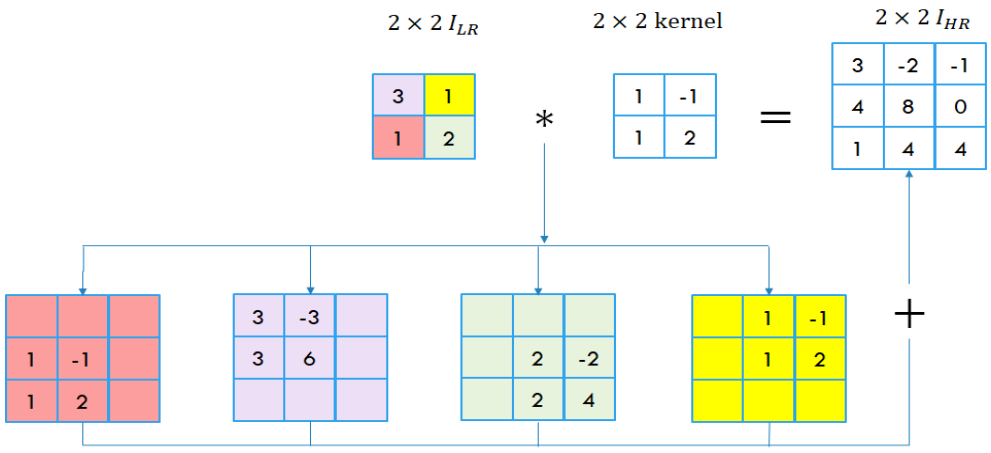


Figure 3.4: Transpose convolution operation

this kind of artifacts by periodically shuffling the pixels of feature maps produced by traditional convolution [15]. The operation is illustrated in Fig. 3.5. However, in practice, this is not always the case, as also observed in Fig. 5.7

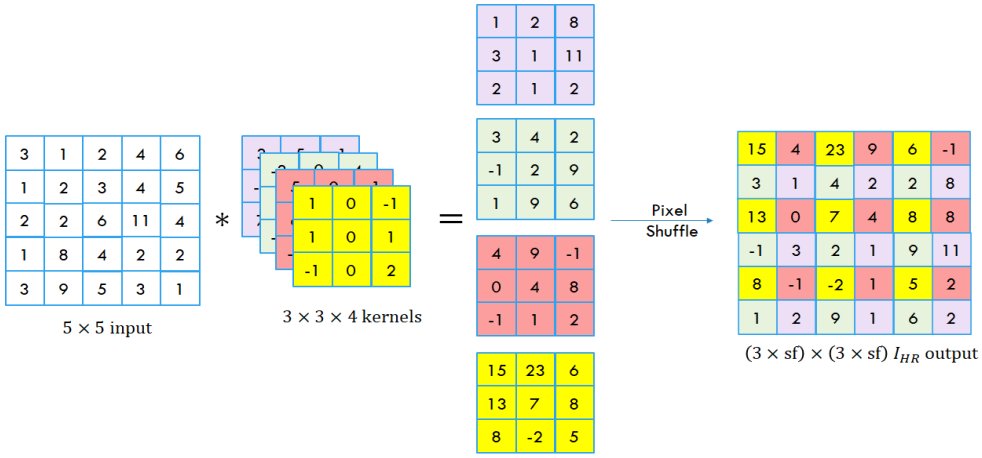


Figure 3.5: Sub-pixel convolution operation, sf here denotes the scaling factor.

- **Channels**:

  In NNs, the input is a vector, while in CNNs, the input is a multi-channelled image, e.g., an RGB image having 3 channels as per color. In the case of US, images are generally grayscale, therefore having one channel. During convolution, for each channel of the image, a separate kernel, often of the same dimension, is learned. While the channels of the original image, e.g., 3 channels in an RGB image, convey colour information, the channels of the convolutional layer convey some abstract aspect of information about the image, not necessarily colour. The number of channels in the output after convolution is defined during building of the model.

- **Pooling**:

  The pooling layer decreases the dimensions of the image by downsampling the feature maps. The reason for introducing the pooling layer is twofold. First, the feature maps record precise feature positions of the input

image. When the input image is augmented by operations like cropping and rotations, the feature map will show a different representation for the same image. Therefore, pooling creates a lower resolution image which is a summarised version or higher-level feature representation which ignores some of the redundant/finer details. Two types of pooling are commonly used in a CNN: max pooling and average pooling. Max pooling takes the maximum value from a patch on the feature map, while average pooling takes the average value of pixels. Max pooling and average pooling operation are depicted in Fig. 3.6. The second advantage of the pooling operation is the reduced computation load for the next layer due to the lower resolution of the image.

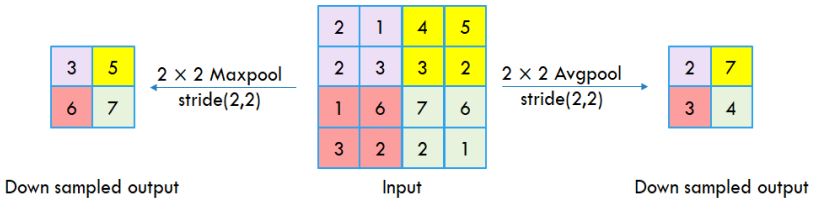


Figure 3.6: Pooling operation

- **Batch normalisation**:

  During NN training, input feature distribution for a layer varies with change in parameters of previous layers. Batch normalisation (BN) therefore normalises the distribution of its input for each batch of examples and outputs data that has mean close to zero and variance close to one. For a mini batch B having n examples, and each example of dimension $h \times w$ with c channels. Then the mean and variances are calculated for each channel across the batch and dimensions. Let b represent one example from B. Let b represent one example from B. Then the mean across a channel in the mini batch can be calculated as $\mu_c = \frac{1}{n \times h \times w} = \sum_{i=1}^{n} \sum_{j=1}^{h} \sum_{k=1}^{w} b_{ijk,c}$ and the variance across a channel c can be given as $\sigma_c^2 = \frac{1}{n \times h \times w} \sum_{i=1}^{n} \sum_{j=1}^{h} \sum_{k=1}^{w} (b_{ijk,c} - \mu_c)^2$. BN then rescales each example across the channel in a mini-batch to

$$\hat{b} = \frac{b - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} \tag{3.5}$$

  where ϵ is a small constant added for stability.

- **Activation function**:

  Activation or transfer functions in NNs add non-linearity to the network. An activation function takes input from neurons/nodes and decides how these neurons are to be 'activated'. The activation restricts the output of the activation function to a certain limit based on demand. This is necessary for deep networks as the weights in the layers can reach huge numbers which can cause computational issues. Some of the activation functions used in US SISR are given in Table 3.1.

Activation Function	Equation
Linear	$f(x) = x$
Logistic (a.k.a Sigmoid)	$f(x) = \frac{1}{1+e^{-x}}$
ReLU (Rectified Linear Unit)	$f(x) = 0, x \leq 0; f(x) = x, \quad x > 0$
Leaky ReLU	$f(x) = 0, \alpha x \leq 0; \quad f(x) = x, x > 0$
TanH	$-1 + \frac{2}{1+e^{-2x}}$

Table 3.1: Some of the popular activation functions used in ML

Residual blocks

As NNs go deeper, the problem of vanishing gradient is often encountered during training [17]. Residual blocks, that were introduced as part of ResNet [17] architecture help in mitigating this problem. The idea behind residual blocks is to introduce skip connection(s) so that the output of layers at the front end of the network is added directly to layers in the deeper part of the network, as can be seen in CycleGAN architecture, Fig. 3.10.

There is also the problem of accuracy saturation, commonly referred to as degradation in deep networks. As DL networks stack up more layers, the training accuracy of the model saturates before dropping rapidly. This is not an over-fitting case as deeper networks should at least increase the training accuracy. Therefore, an information loss is happening in the network as the network goes deeper. Residual blocks help in mitigating this problem by passing information from the shallow part of the network directly to deeper parts of the network.

3.1.2 Cost function and optimization algorithm

A cost function gives a mathematical relationship between the learned output of the model and the expected output from the model. Most ML models seek to learn parameters that either minimises or maximises the cost function. The parameters are learned/updated through backpropagation [57]. If a cost function is being minimized, it is also interchangeably referred to as loss function or error function. For example, a very simple cost function in the SR problem could be the pixel difference between the predicted image and the ground truth image as following:

$$J(\theta) = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I_{HR}(i,j) - M_\theta(I_{LR}(i,j))]^2 \tag{3.6}$$

where I_{HR} is the true image and I_{LR} is the input to the Model M having parameter θ (or a set of parameters defined by θ). An efficient optimization algorithm here helps to find optimum arguments for the cost function that minimises (or maximises) it, i.e.,

$$\theta^* = \arg\min J(\theta) \tag{3.7}$$

Gradient descent

One of the simplest and earliest optimisation algorithms used in ML is gradient descent. It is a first-order optimisation algorithm, i.e., use first-order derivative or gradient to find local minima (or maxima) of a differentiable cost function.

The algorithm of gradient descent is as follows:

1. Initialize θ. This is a hyperparameter and can be defined during model building. For example, initialising weights from a uniform distribution of predefined range, or initialising bias to zero.
2. Choose a learning rate α. This is also a hyperparameter.
3. Repeat until convergence (reach local minima):

$$\theta_{t+1} = \theta_t - \alpha \, \nabla J(\theta_t) \tag{3.8}$$

where t denotes a successive gradient step. Here, if function J is differentiable in the neighbourhood of θ and α is small enough, then $J(\theta_{t+1}) \leq J(\theta_t)$. Gradient descent updates the model parameters through backpropagation after all the data samples have been passed through the model once, i.e., one epoch. Gradient descent can also be extended to mini-batch gradient descent and stochastic gradient descent (SGD). In mini-batch gradient descent, the parameters are updated after each mini-batch of samples passes through the model. In SGD, the parameters are updated after each data sample.

Adam algorithm

Adam algorithm (derived from adaptive moment estimation) is an extension of SGD (with momentum) and is a very popular optimisation algorithm. It is also a first-order optimisation algorithm and combines adaptive estimates of lower order moments (first and second). Adam algorithm works well with sparse gradients [58] and can adjust the learning rate during training. The algorithm can be briefly summarised as:

1. Initialise θ. This can be any random initialisation.
2. Choose learning rate α, exponential decay rates for momentum $\beta_1, \ \beta_2$
3. Initialise: first moment vector $m_o \ = \ 0$, second moment vector $v_o \ = \ 0$
4. Repeat until convergence:

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) \nabla J(\theta_t) \tag{3.9a}$$

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2) (\nabla J(\theta_t))^2 \tag{3.9b}$$

$$\hat{m}_{t+1} = \frac{m_{t+1}}{(1 - \beta_1^{t+1})} \tag{3.9c}$$

$$\hat{v}_{t+1} = \frac{m_{t+1}}{(1 - \beta_2^{t+1})} \tag{3.9d}$$

$$\theta_{t+1} = \theta_t - \alpha \times \frac{\hat{m}_{t+1}}{\hat{v}_{t+1} + \epsilon} \tag{3.9e}$$

where t is succesive time step and ϵ is a small constant to provide numeric stability.

3.1.3 Dataset

It is widely accepted that a huge number of samples are required to accurately train DL models. Per contra, there is a lack of large data sets in the medical US due to patient privacy and permission overheads. Further considering the case of SR in US SISR, there are no HR-LR pairs in existence originally. To deal with these issues, data augmentation and data generation is carried out before training the model.

Common data augmentation methods include creating mirror reflection, i.e., vertically or horizontally flipping the image, and rotating images by different angles. Another step is the generation of HR parent - LR child pairs. An LR child is a lower resolution counterpart of an HR parent. Thus, each HR parent - LR child pair serves as a train and target sample in absence of actual HR-LR pairs. One of the common methods to create LR child images is downsampling (using pooling methods) a US image and adding an iid (independent and identically distributed) noise to the image. Noise and other degrading effects are added to the LR child to add variability in the dataset. Simply augmenting the dataset with rotation, mirror flips and downsampling still does not add much 'variety' to the dataset. The number of examples that are distinct in the dataset still remains small. With noise and degrading effects, there is more opportunity for the model to learn from a richer dataset. This also helps in avoiding over-fitting while training the model. Over-fitting occurs in the case of training a large NN with a small dataset (which is often the case in US SISR), the NN can memorise the training dataset instead of learning a mapping from input to output.

Therefore, if given an HR image, I_{HR}, the LR counterpart, I_{LR}, can be given as:

$$I_{LR} = d(q(I_{HR})) + n \tag{3.10}$$

where q is a downsampling operation, d is represents some other degrading operation(s) and n is noise.

Besides creating LR images, there can be other types of synthetic data generation, such as in [51], where synthetic CEUS (Contrast Enhanced US) images are created with microbubbles. The data generation or modification step is particular to the SR model used.

3.1.4 Performance analysis

After the model has been trained on the dataset, there needs to be some performance measure that evaluates the competence of the ML model. This performance measure usually varies with the ML task. For example, in classification tasks, accuracy (number of correct classifications out of total) can be a performance measure. For image comparisons,

PSNR (Peak Signal to Noise ratio, discussed in 4.5.2) and SSIM (Structural Similarity Index, discussed in 4.5.2) are the standards. Besides these, one of the recent and popular metric is VGG loss.

Losses like MSE capture pixel-wise differences with restricted capability in calculating perceptual differences [59]. Therefore, a different kind of loss, VGG loss, which measures the perceptual similarity between the predicted image (from the model) and the true image/ground truth image is becoming popular in SISR models. VGG loss is derived from the popular 19 layers image classification model VGG19 [9].

If the last 3 fully connected (FC) layers of a pre-trained VGG network are disregarded, the remaining network can be used to extract feature maps of an image. Therefore, to calculate the VGG loss between the predicted image and the ground truth image, feature maps from the pre-trained VGG layers are first extracted for the two images. The final loss is then obtained by calculating the Euclidean distance between the two extracted feature maps. For example, in a SISR model that generates SR image, $\hat{I}_{HR}$, and the original HR image is I_{HR}, the VGG loss can be calculated as:

$$L_{C/j,i} = \frac{1}{W_{j,i}H_{j,i}} \sum_{x=1}^{W_{j,i}} \sum_{y=1}^{H_{j,i}} (\phi_{j,i}(I_{HR})_{x,y} - \phi_{j,i}(\text{Model}(I_{LR}))_{x,y})^2 \tag{3.11}$$

where $\phi_{j,i}(\cdot)$ represents extracted feature map of an image after the i^{th} convolution layer of j^{th} convolution block in the VGG-19 network. Therefore, $\phi_{j,i}(I_{HR})$ and $\phi_{j,i}(\text{Model}(I_{LR}))$ calculate the feature maps of the original HR image, I_{HR} and reconstructed SR image, $\hat{I}_{HR}$, respectively. $W_{j,i}$ and $H_{j,i}$ are the width and height of feature map after the i^{th} convolution layer of j^{th} block. If j and i are lower numbers, say $j = 2$, $i = 2$, then the output feature maps retain finer details (like colour, texture and exact shape) better than when j and i are a higher number, for e.g., 4 and 3 respectively. Therefore, the number of VGG layers to consider for loss consideration depends on the SR problem. Usually, for images of lower dimensions, a low i and j value is preferred.

3.2 DL models used in SISR

In this section, a summary of some of the prominent underlying DL algorithms is mentioned in Table 2.1 and Table 2.2. Throughout the models, I_{LR} denotes a low-resolution image, I_{HR} denotes its high resolution counterpart and $\hat{I}_{HR}$ denotes the reconstructed SR image.

3.2.1 U-Net

U-Net [50] is a 23-layer biomedical image segmentation model. U-Net is based on a fully convolutional network (FCN) [47] and can be trained with a limited number of examples and still give good results. The training examples are heavily augmented to form a bigger training set.

The U-Net architecture, shown in Fig. 3.7, consists of a contracting path and an almost symmetric expansive path. The contracting path tries to capture features while the extracting path tries to localise, i.e., assign a class label to each pixel. The contracting path is made of repeated blocks that perform a sequence of convolution $\rightarrow$ ReLU $\rightarrow$

max pooling operations. After each block, the number of channels is doubled. Following the contracting path, the expansive path performs a sequence of upsampling → concatenation (with cropped image from contracting path) → convolution → ReLU. The concatenation step, called "crop & copy", helps in delivering localisation information from the contracting path to the expansive path. U-Net does not use padding while convolution, and to avoid the output image being smaller than the input, it uses an overlap tile strategy while training [50], where the output is predicted in parts from the input.

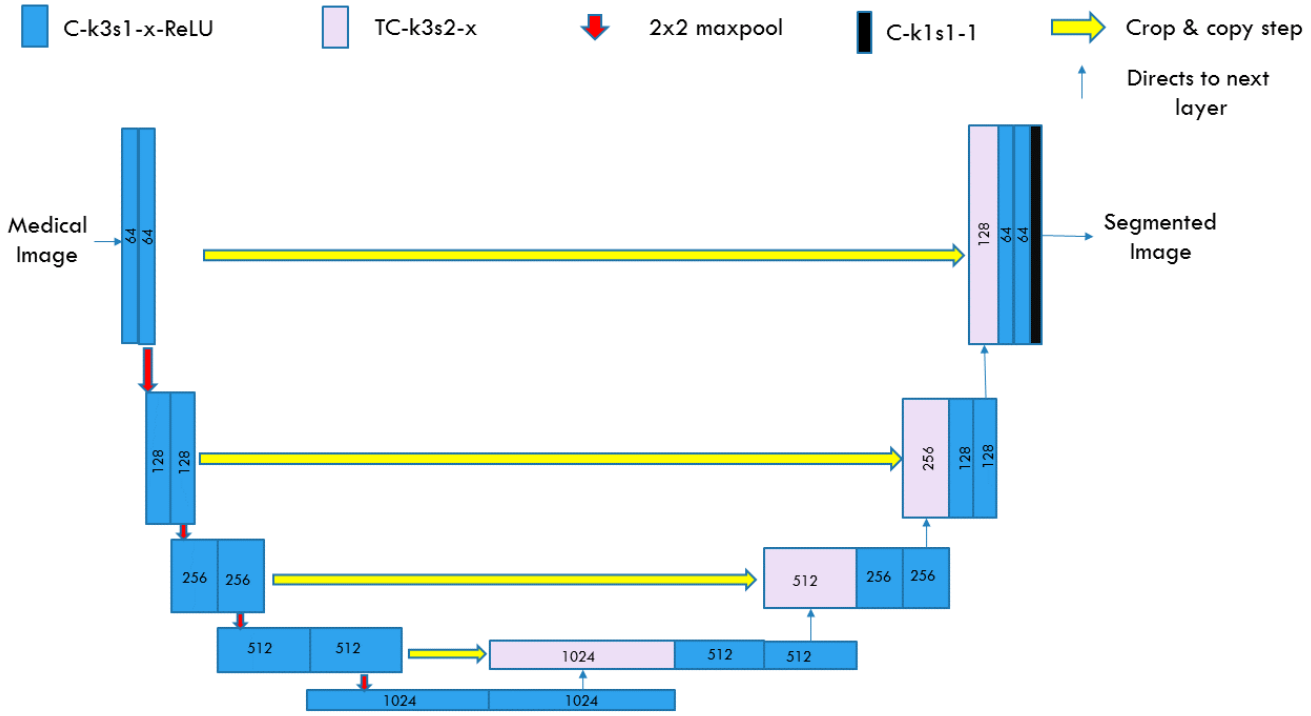


Figure 3.7: U-Net architecture; where a C-k3s1-x-ReLU denotes a convolution block of convolution layer of 3×3 kernel and stride 1 with output channels equal to x followed by ReLU, and TC-k3s2-x denotes a transpose convolution layer with 3×3 kernel and stride 2 with x output channels. The output channels are written inside the convolution layers. Image reproduced from [50]

3.2.2 GAN

Invented by Ian Goodfellow and his colleagues in 2014, the GAN [2] is a famous DL algorithm that generates high dimensional data, images, and audio waveforms, based on real samples. For example, a popular application of GAN is generating realistic human faces from samples of existing human faces. GANs have a supervised component in their training process, however, as a whole, they fall under the category of the unsupervised algorithm. In the present day, GAN has been applied in image processing, computer vision, natural language processing (NLP), and music.

GANs are DL generative models. The main task of GAN is to discern patterns and information in given data distribution, i.e., real distribution, $p_{data}(X)$ well enough to create new synthetic data samples that look like they have been derived from the $p_{data}(X)$. The basic architecture of a GAN, as depicted in Fig. 3.8, involves two networks:

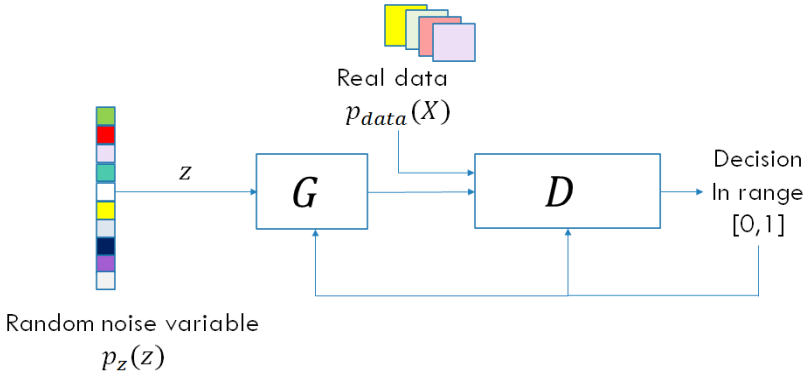


Figure 3.8: GAN; G represents a generator which takes in a random noise sample $z \sim p_z(z)$ and generates output $G(z)$. The discriminator D takes either the generated instance $G(z)$ or instance from real distribution $x \sim p_{data}(x)$ as input.

a generator (G) and a discriminator (D). Both these networks are independent of each other but are trained simultaneously.

Given input samples from a random noise prior distribution $p_z(z)$, the task for the generator is to learn the real data distribution and generate synthetic data that matches the distribution of real data $p_{data}(X)$. Meanwhile, the discriminator takes in samples from either the real distribution or from the generator and tries to classify the input samples as real (from real distribution) or fake (from generator). The output to the discriminator is a scalar which is a value in the range [0,1]. A value closer to one means that, with a very high probability, the input to the discriminator was from real distribution and vice versa. Both the discriminator and the generator then get updated after each round (a decision from the discriminator on an input sample). Initially, when a GAN model starts training, the discriminator is 'stronger' and is able to discern real from fake with a very high probability. However, with each update, the generator gradually gets better, and eventually, the discriminator is only able to make a random guess on the samples provided to it, i.e., with probability $\frac{1}{2}$ that a sample is real or fake.

In this fashion, the framework is analogous to a minimax game. The minimax game in GAN is played by two players: D and G, where G's loss is D's gain and vice versa. Therefore, the value function $V(G, D)$ can be given as:

$$V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.12}$$

where $G(z)$ is generator's output given input random noise sample $z \sim p_z(z)$, $D(x)$ is the discriminator's estimate of probability of real data x being a real instance and $D(G(z))$ is the discriminator's estimate of probability of $(G(z))$ being a real instance. $\mathbb{E}_{x \sim p_{data}(x)}$ is the expected value over real data instances and $\mathbb{E}_{z \sim p_{data}(z)}$ is the expected value over all random noise samples. The discriminator in $V(D, G)$ tries to maximise $\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$, while the generator tries to minimise $\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$.

In the original paper, multi-layer perceptrons are used in both, generator and discriminator. However, the generator is free to use any type of NN. On the other hand, the discriminator is preferred to have fully connected layers leading

to a classifier.

3.2.3 PatchGAN

PatchGAN, illustrated in Fig. 3.9, applies GAN to image-to-image translation, i.e., to convert an image from one domain to another. For example, turning a black & white photograph into a colour one. Instead of the usual discriminator which outputs a scalar value, PatchGAN outputs a 2D array, also called D-map. While the typical GAN looks at the entire picture and then decides if it is real or fake, PatchGAN compares patches within pictures and outputs a matrix. Since the PatchGAN discriminator looks into local textural information independently and then reaches a decision, it has become a standard discriminator in many GAN variants.

The discriminator of PatchGAN is what sets it different from a traditional GAN network. Therefore, PatchGAN can use any type of generator network. In the original paper, two types of generator architectures were tested: an encoder-decoder architecture and a U-Net like architecture with some alterations. The discriminator of a PatchGAN is a simple CNN consisting of a sequence of convolution $\rightarrow$ BatchNorm (except after the first convolution) $\rightarrow$ LeakyReLU blocks.

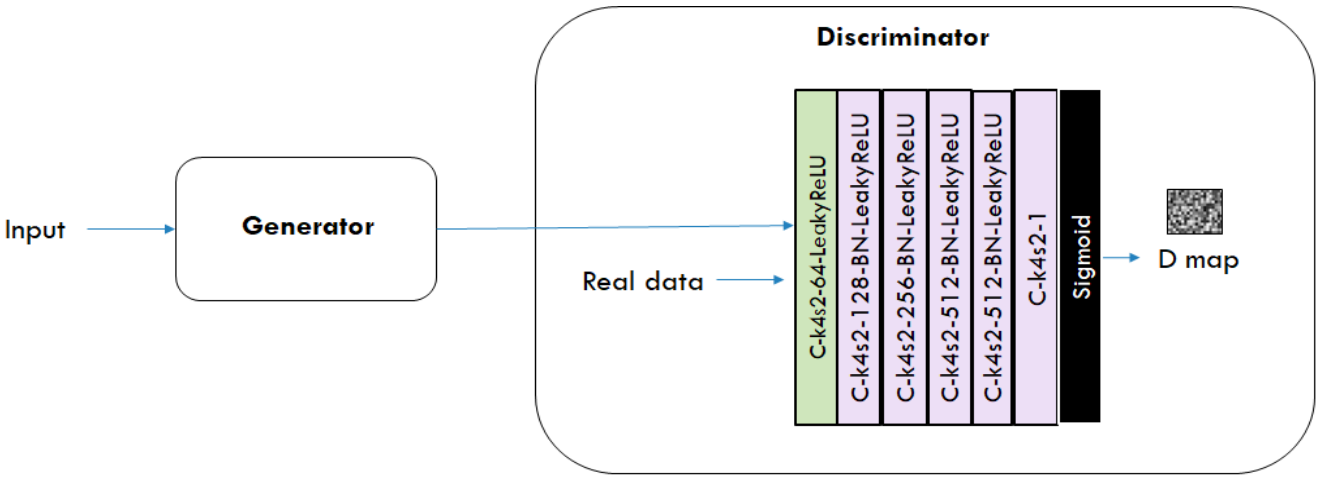


Figure 3.9: PatchGAN architecture; where a C-k4s2-128-BN-LeakyReLU denoted a convolution block of convolution with 4×4 kernel and stride 2 with output channels equal to 128 followed by batch normalisation followed by ReLU; reproduced from [30]

3.2.4 CycleGAN

CycleGAN, another variant of GAN, is also used for image-to-image translation. For example, as in the original paper [43], to change a photograph to Van Gogh style painting. Original CycleGAN used unpaired images, i.e., the photograph and Van Gogh painting come from their respective groups/domains. For a photograph in its domain that depicts a particular scene, there is no Van Gogh painting in the other domain that depicts the same scene and vice-versa.

Let there be a source domain S having n images of a particular style, such that $\{I_i^S\}_{i=1}^n \in S$, where I_i^S is the i^{th} image in domain S. Similarly, let the target domain have m images of another style, such that $\{I_i^T\}_{i=1}^m \in T$. The data distribution of the domains is $I^S \sim p_{data}(I^S)$ and $I^T \sim p_{data}(I^T)$. The task of CycleGAN is to learn a mapping such that images from domain S resemble the style of the images from domain T. The cycle term in CycleGAN then comes from its two mappings: $S \rightarrow T$ and $T \rightarrow S$. Images from domain S are translated to replicate images of style T and then these replicated images are translated back to images of style S. As two types of translation take place in the model, naturally this task involves two GANs. Generator G_1 is responsible for learning $S \rightarrow T$ mapping such that $G_1(I^S) \approx I^T$ and generator G_2 is responsible for learning $T \rightarrow S$ mapping such that $G_2(I^T) \approx I^S$. Discriminator D_1 discriminates between $G_1(I^S)$ and I^T and discriminator D_2 discriminates between $G_2(I^T)$ and I^S. The loss between the discriminator and the generator, called the adversarial loss, for the two GANs is defined as:

$$\min_{G_1} \max_{D_1} L_{GAN}(G_1, D_1, S, T) = \mathbb{E}_{I^T \sim p_{data}(I^T)}[\log D_1(I^T)] + \mathbb{E}_{I^S \sim p_{data}(I^S)}[\log\big(1 - D_1(G_1(I^S))\big)], \tag{3.13a}$$

$$\min_{G_2} \max_{D_2} L_{GAN}(G_2, D_2, T, S) = \mathbb{E}_{I^S \sim p_{data}(I^S)}[\log D_2(I^S)] + \mathbb{E}_{I^T \sim p_{data}(I^T)}[\log\big(1 - D_2(G_2(I^T))\big)] \tag{3.13b}$$

where $L_{GAN}(G_1, D_1, S, T)$ is the adversarial loss for the GAN that maps $S \rightarrow T$ and $L_{GAN}(G_2, D_2, S, T)$ is the adversarial loss for the GAN that maps $T \rightarrow S$. To check the consistency of $S \rightarrow T \rightarrow S$ mapping, a consistency loss called the cycle consistency loss is introduced in CycleGAN. This loss tries to ward off discrepancies in the G_1 and G_2 mappings by calculating differences as:

$$L_{cyc}(G_1, G_2) = \mathbb{E}_{I^S \sim p_{data}(I^S)}[\|G_2(G_1(I^S)) - I^S\|_1] + \mathbb{E}_{T \sim p_{data}(I^T)}[\|G_1(G_2(I^T)) - I^T\|_1] \tag{3.14}$$

The eventual goal of CycleGAN is to find optimal mapping G_1 and G_2 such that

$$G_1{}^*, G_2{}^* = \arg \min_{G_1, G_2} \max_{D_1, D_2} L(G_1, G_2, D_X, D_Y) \tag{3.15}$$

where,

$$L(G_1, G_2, D_1, D_2) = L_{GAN}(G_1, D_1, S, T) \; + \; L_{GAN}(G_2, D_2, T, S) + \; \lambda \, L_{cyc}(G_1, G_2) \tag{3.16}$$

The hyperparameter λ controls the relative importance of the objectives. The architecture of CycleGAN can be seen in Fig. 3.10.

The generator comprises three components: encoder, transformer and decoder. The encoder extracts features from the image, then the transformer made up of residual blocks takes care of actual translation, and decoder performs transpose-convolution. For the discriminator network, authors use PatchGAN.

3.2.5 Self-supervised learning (SSL)

SSL or unsupervised learning methods, as briefly discussed in Section 2.1.4, are a class of unsupervised methods where the training phase of the model is solely based on a test image and examples generated from the test image.

ZSSR ("Zero-Shot" Super Resolution)

ZSSR, as discussed in Section 2.1.4, is a self-supervised learning model. Given an I_{LR} image, ZSSR predicts $\hat{I}_{HR}$ image based on examples generated from the single input I_{LR} itself. The motivation behind the formulation of ZSSR is the existence of recurring small image patches within natural images [60]. Thus, when the $\hat{I}_{HR}$ is formed after observing samples from I_{LR}, it is able to observe internally hidden recurring information within the image better than supervised methods [28]. This also allows ZSSR to perform SISR of images that it encounters for the first time, unlike supervised methods. In supervised SISR methods, the model is trained on a general collection of images and tested on similar images. However, if images having internal characteristics varying from this general collection of images is tested on the previously trained model, the model will have difficulty in predicting the output SR image. The overview of ZSSR, as seen in Fig. 3.11, is as follows:

1. **Data augmentation and generation**: Provided input I_{LR}, the image is downsampled and randomly augmented (Random crop of fixed size 128×128 + random rotation + random affine transformation + random mirror inversion) to create HR parent. The HR parent is then again downsampled to create an LR child. This way one HR-LR pair is generated.

2. **Training**: Train the synthetically generated HR-LR pair on a small CNN network. At every iteration, a new HR-LR pair is fed to the network. Therefore, the model progressively learns its parameters through reconstruction error which can be defined as:

$$\text{Reconstruction error} = \|\text{HR parent} - \text{Model(LR child)}\|_1 \tag{3.17}$$

3. **Predict**: Take prediction of original I_{LR} on the trained CNN network.

KernelGAN

KernelGAN, also given by authors of ZSSR, is a blind SR method. KernelGAN uses the PatchGAN with a simple generator as shown in Fig. 3.12. In the Blind SR problem (discussed in Section 2.1.4), the I_{LR} images provided to the model have been created by downsampling by an unknown kernel. Most of the SR methods mentioned in Section 2.2.3 perform super resolution on images that have been downsampled by a known kernel, commonly bicubic interpolation. This is the 'ideal' case. However, in the case of 'non-ideal' scenario, images have been obtained by downsampling by an unknown kernel. Therefore, when these models that have been trained on 'ideal' cases fail

to predict 'non-ideal' LR images with the same accuracy as they depict in 'ideal' case [29]. Therefore, the goal of KernelGAN is to find an image-specific downsampling kernel for the LR image. This downsampling kernel is able to create LR images from the input image that retains the distribution of patches across the input image. A KernelGAN is trained on HR-LR synthetically generated pairs until the generator of the GAN becomes proficient in creating image patches that can fool the discriminator. The objective function for KernelGAN is to find:

$$G^*(I_{LR}) = \arg\min_G \max_D [\mathbb{E}_{x\sim \text{patches}(I_{LR})}[\|D(x) - 1\|_1 \ + \ \|D(G(x))\|] + R] \tag{3.18}$$

where R is a regularisation term on the generator and is defined as a sum of four losses [29].$\mathbb{E}_{x\sim \text{patches}(I_{LR})}$ is the expected value over all patches from original input I_{LR}. Once the generator converges to G^*, the layers of the generator are simply convolved together to output a 2D-matrix as an image specific downsampling kernel.

3.2.6 Multi-level Wavelet-CNN (MWCNN)

MWCNN is a supervised learning method of restoring images. The task of image restoration models is to clean/restore noisy or otherwise degraded images that have been provided to them. The architecture of MWCNN is U-Net based, except it replaces max-pooling with 2 level wavelet packet decomposition (WPD) and transposed convolution with inverse wavelet packet decomposition (IWPD). Its architecture can be seen in Fig. 3.13.

Wavelet Packet decomposition (WPD)

WPD is an extension of the wavelet transform. Level 2 WPD is a signal processing method where the signal/frequency is discretely sampled by wavelets into a set of four sub-bands. For reconstruction of the image, an inverse WPD operation is performed on the sub-bands. A wavelet in 2D WPD is a simple filter that is convolved with the image to create a sub-band image. This sub-band image is then downsampled. The operation can be summarised as $(f * I) \downarrow 2$, where f is the wavelet filter and I is the image. There are two ways in which one can perform 2D DWT on an image. The first method, as depicted in Fig. 3.14 is to first convolve with a low pass filter (f_L) and high pass filter (f_H) along the rows of the image. Next, perform a downsampling operation on the two sub-bands and thereafter again perform wavelet transformation using f_L and f_H along the columns of each of the previously obtained sub-band images. The other method, which is used here, is to directly convolve the image I with four filter: f_{LL}, f_{LH}, f_{HL} and f_{HH}.

The first sub-band is I_1, is derived by convolution of the image with f_{LL}, and is called the average coefficient or average sub-band. This sub-band contains all the low-frequency details of the image. Low-frequency regions in an image refer to parts in the image where the change in pixel value is gradual. For example, if a patch inside the boundary of a solid blue color circle is considered, the change in adjacent pixel values will be gradual. The next three sub-bands are: I_2, I_3, and I_4 and are called the Horizontal sub-band, vertical sub-band, and diagonal sub-bands respectively. All these three sub-bands capture high-frequency details of the image. High-frequency details within an

image refer to parts where the pixel value changes rapidly. For example, at the boundary of the blue-colored circle, the pixel values will change rapidly. The four filters used here are Haar 2D filters/wavelets and are defined as:

$$f_{LL} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},\ f_{LH} = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix},\ f_{HL} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix},\ f_{HH} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.19}$$

Given image I, the four sub-bands after 2D haar wavelet transformation can also be summarised as:

$$I_1(i,j) = I(2i-1,2j-1) + I(2i,2j-1) + I(2j-1,2j) + I(2i,2j) \tag{3.20a}$$

$$I_2(i,j) = -I(2i-1,2j-1) + I(2i,2j-1) - I(2j-1,2j) + I(2i,2j) \tag{3.20b}$$

$$I_3(i,j) = -I(2i-1,2j-1) - I(2i,2j-1) + I(2j-1,2j) + I(2i,2j) \tag{3.20c}$$

$$I_4(i,j) = I(2i-1,2j-1) - I(2i,2j-1) - I(2j-1,2j) + I(2i,2j) \tag{3.20d}$$

where (i,j) gives the row and column number within the sub-bands. This forms one downsampling operation. The bi-orthogonal property of WPD allows for sub-sampling without significant information loss. Therefore, in image denoising and compression models, WPD can provide a better downsampling method than usual pooling operations [12].

At the contracting path of the MWCNN network, 2D IWPD is carried out. 2D IWPD is the opposite of WPD and can be seen in Fig. 3.14.

3.3 US SISR example

One of the models from Table 2.2 is discussed here. For convenience, the model [42] is referred to as CycleGAN-US-SISR in what follows. CycleGAN-US-SISR is a self-supervised model that takes inspiration from CycleGAN and ZSSR to perform SISR of US images. The architecture of the model is shown in Fig. 3.15.

The similarity between ZSSR and CycleGAN-US-SISR is in their method of data augmentation, learning rate update method and the self-ensemble method [18] used to predict $\hat{I}_{HR}$ images. Besides these similarities, the model and losses in ZSSR and CycleGAN-US-SISR are completely different. On the other hand, both CycleGAN and CycleGAN-US-SISR use a cycle consistency to check the $S(LR) \rightarrow T(\hat{HR}) \rightarrow S(LR))$ consistency and an adversarial loss between the generators and discriminators. However, CycleGAN uses two generators and two discriminators while CycleGAN-US-SISR uses four generators and two discriminators. The other difference between both the models is in losses. CycleGAN-US-SISR introduces a pixel loss and VGG loss in addition to adversarial loss and cycle loss. The derivation of cycle consistency loss is also slightly different in CycleGAN-US-SISR.

The general method of CycleGAN-US-SISR begins with the data augmentation step which takes in an I_{LR} image and augments it to create a pair of $I_{HR\ parent} - I_{LR\ child}$ parent child (similar to ZSSR). Let the data distribu-

tion of $I_{HR\ parent} - I_{LR\ child}$ parent child pair be defined as $I_{HR\ parent} \sim p_{\text{ HR data}}(I_{HR\ parent})$ and $I_{LR\ child} \sim p_{\text{ LR data}}(I_{LR\ child})$. The model performs SISR of US images in two complete 'cycles'. Two generators and a discriminator are employed to complete one such complete cycle. In cycle 1, a mapping from LR domain $\rightarrow$ HR domain $\rightarrow$ LR domain is carried out. Cycle 2 carries out a mapping from LR domain $\rightarrow$ HR domain $\rightarrow$ LR domain . Generator G_1 which carries out the mapping of LR domain $\rightarrow$ HR domain is a multi-scale deep network and generator G_2 which maps HR domain $\rightarrow$ LR domain is a single scale deep network. The two discriminators used have architecture similar to one used in PatchGAN. D_1 discriminates between $G_1(I_{LR\ child})$ and $I_{HR\ parent}$, while D_2 discriminates between $G_2(I_{HR\ parent})$ and $I_{LR\ child}$. The four losses: pixel loss (L_{pixel}), perception/VGG loss (L_{percep}), adversarial loss (L_{adv}) and cycle loss L_{cyc} are defined as:

$$L_{pixel} = \frac{1}{n}\sum_{i=1}^{n}[\|G_1(I_{LR\ child}) - I_{HR\ parent}\|_1 + \|G_2(I_{HR\ parent}) - I_{LR\ child}\|_1] \tag{3.21a}$$

$$L_{percep} = \frac{1}{n}\sum_{i=1}^{n}[\|\phi(G_1(I_{LR\ child})) - \phi(I_{HR\ parent})\|_2 + \|\phi(G_2(I_{HR\ parent})) - \phi(I_{LR\ child})\|_2] \tag{3.21b}$$

$$L_{adv} = \frac{1}{n}\sum_{i=1}^{n}[-\log(D_1(G_1(I_{LR\ child}))) - \log(D_2(G_2(I_{HR\ parent})))] \tag{3.21c}$$

$$L_{cyc} = \frac{1}{n}\sum_{i=1}^{n}[\|G_2(G_1(I_{LR\ child})) - I_{LR\ child}\|_1 + \|G_1(G_2(I_{HR\ parent})) - I_{HR\ parent}\|] \tag{3.21d}$$

where n is the number of $I_{HR\ parent} - I_{LR\ child}$ pairs. The final loss, L_{total}, that optimises the model is a aggregation of all the above losses and is given as:

$$L_{total} = \alpha L_{pixel} + \beta L_{percep} + \gamma L_{adv} + \eta L_{cyc} \tag{3.22}$$

where α, β, γ and η are coefficients that control the weight of losses in the total loss.

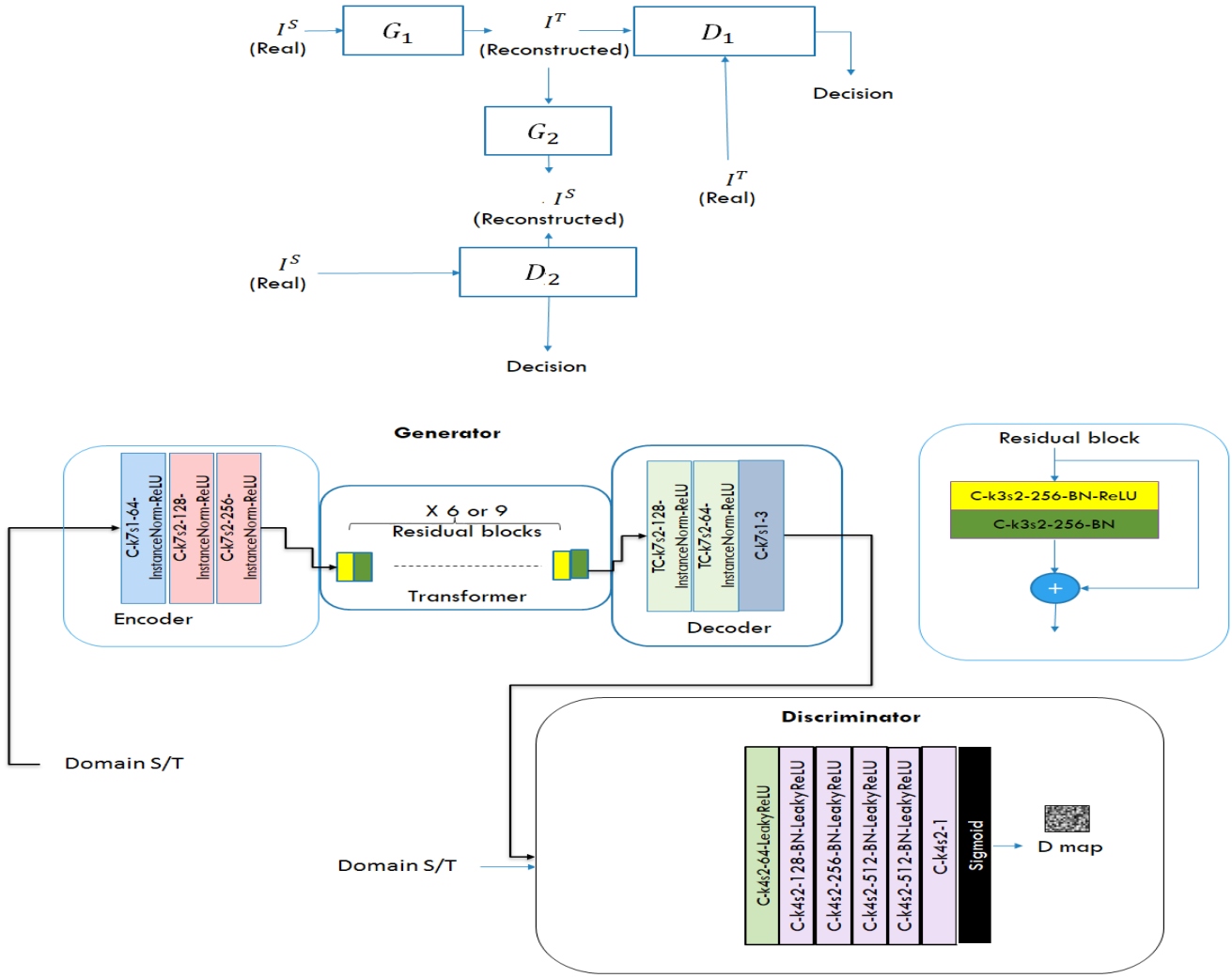


Figure 3.10: CycleGAN architecture; a C-k(k')-s(s')-x-BN-ReLU block consists of a k'×k' convolutional layer having stride s' and output channels x, followed by batch normalisation and ReLU. A C-k7s1-x-InstanceNorm-ReLU layer means a 7×7 convolutional layer with stride 1 and x output channels, followed by Instance normalisation and followed by ReLU activation. Instance normalisation differs from batch normalisation in normalising across each sample (instance) and each channel as opposed to across each channel in a mini batch. The mean is given as $\mu_{nc} = \frac{1}{h \times w} \sum_{i=1}^{h} \sum_{j=1}^{w} b_{nc,ij}$ and the variance is given $\sigma_{nc}^2 = \frac{1}{n \times h \times w} = \sum_{i=1}^{h} \sum_{j=1}^{w} (b_{nc,ij} - \mu_n c)^2$, where $h \times w$ are the dimension of a instance having c channels in a batch of n instances. The instance then is rescaled to $\hat{b} = \frac{b - \mu_{nc}}{\sqrt{\sigma_{nc}^2 - \epsilon}}$ where ϵ is added for numerical stability; reproduced from [43]

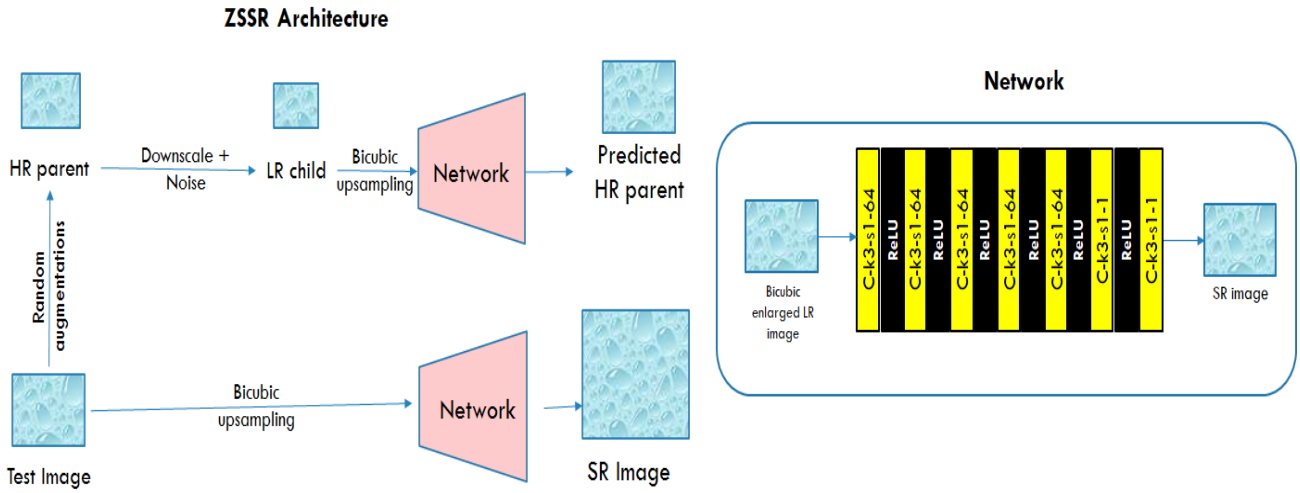


Figure 3.11: ZSSR architecture; a C-k3-s1-64 convolution layer means a 3×3 convolution layer with stride 1 and 64 output filters. Reproduced from [28]

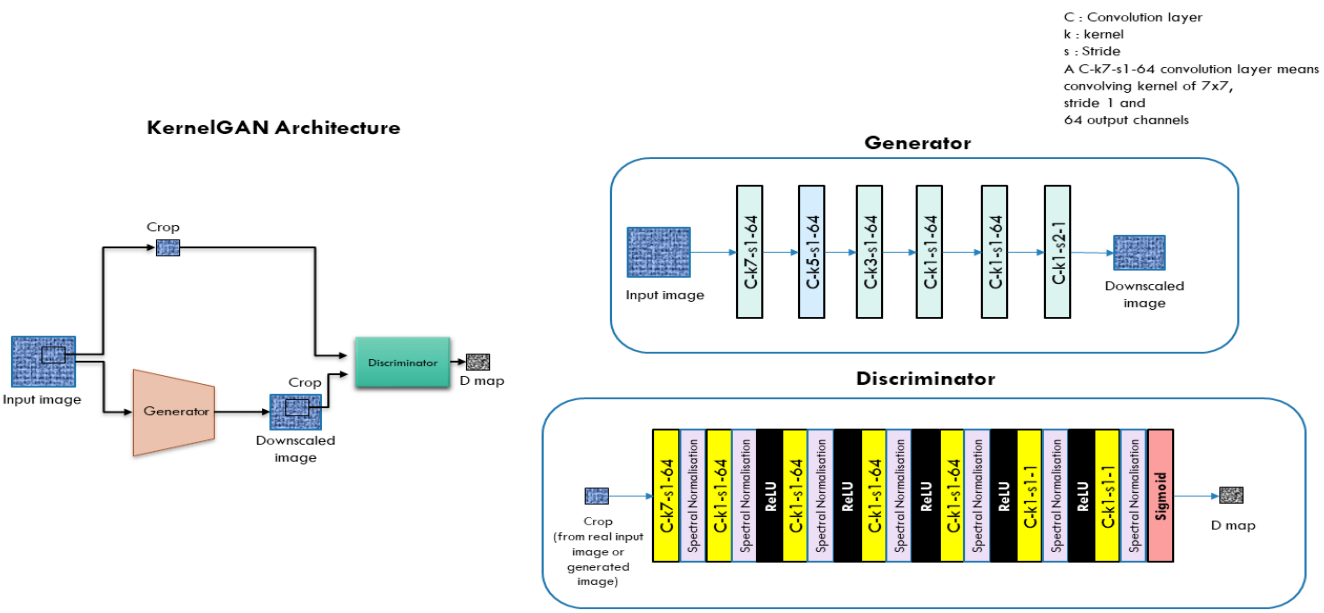


Figure 3.12: kernelGAN architecture; reproduced from [29]

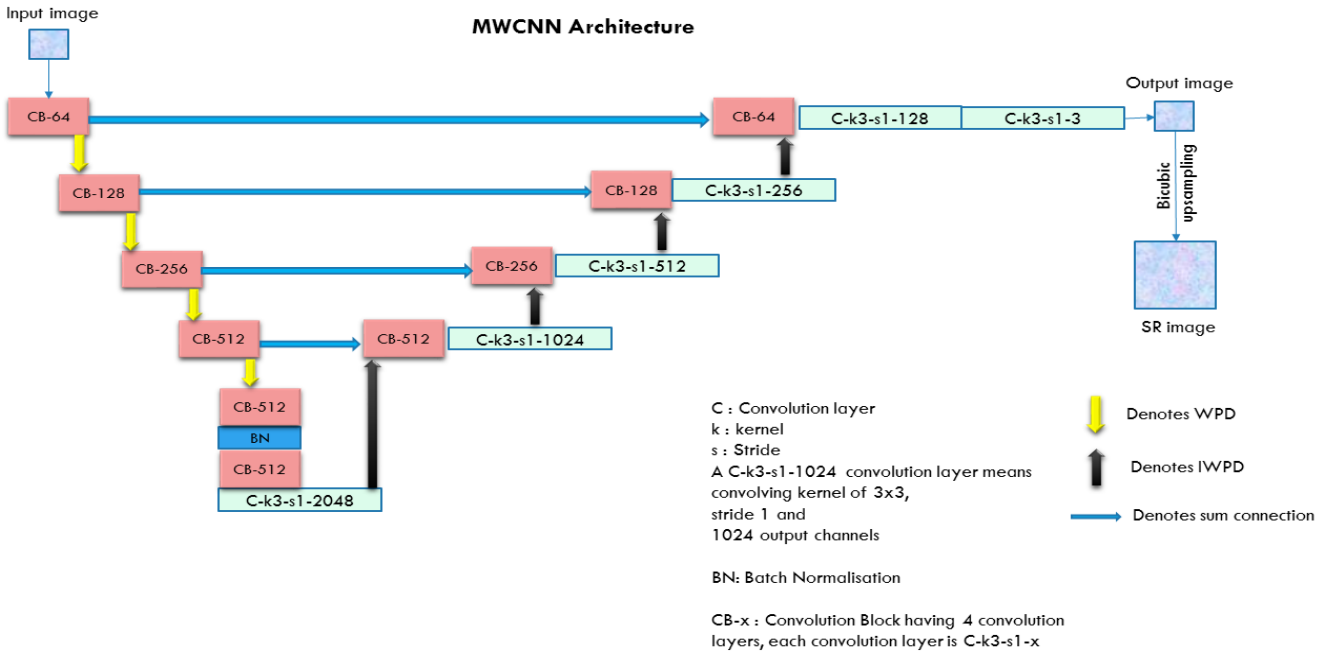


Figure 3.13: MWCNN architecture

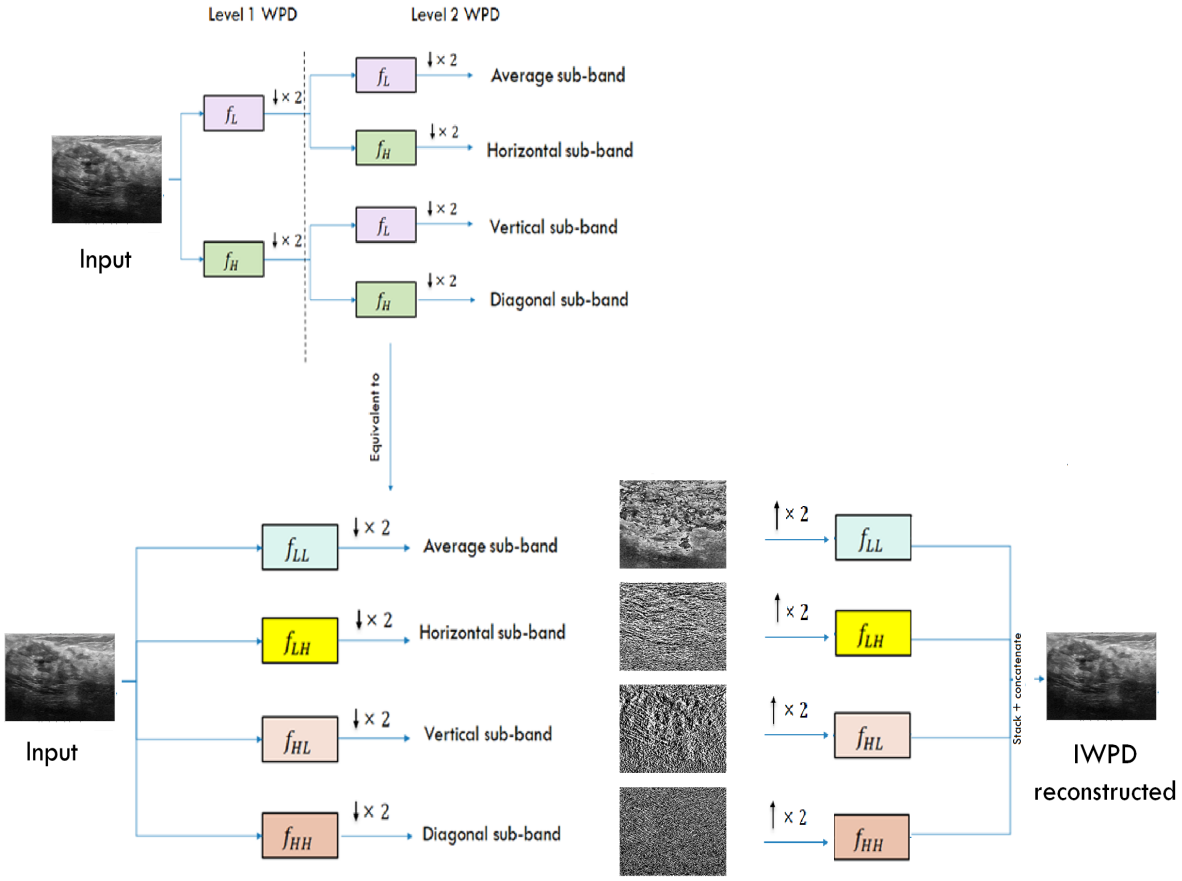


Figure 3.14: A complete level 2 wavelet packet transformation.

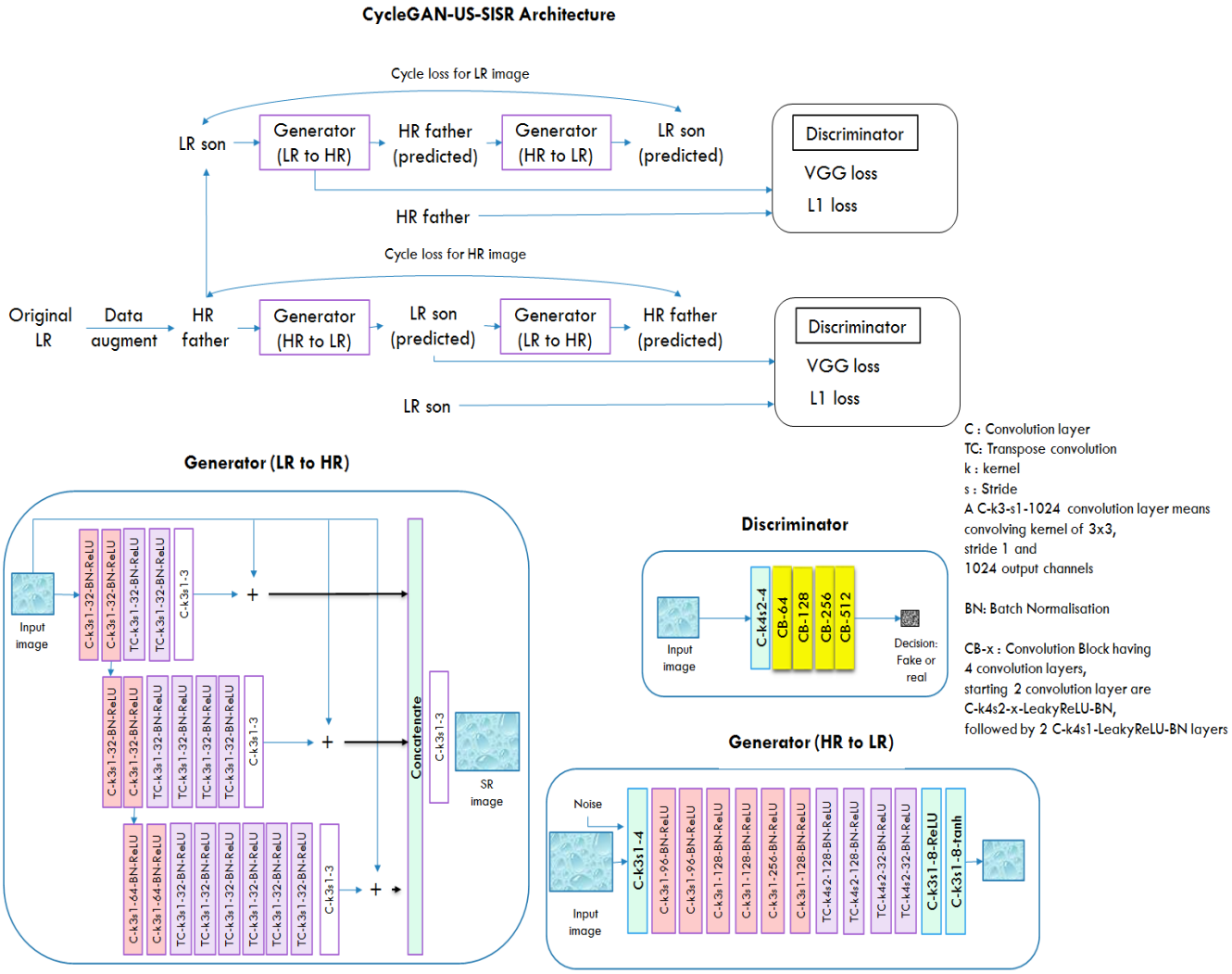


Figure 3.15: CycleGAN-US-SISR architecture, reproduced from [42]

Chapter 4

Methodology

This chapter describes the dataset and dataset augmentation methods used for the simulations. Secondly, metrics used for evaluations are briefly discussed. The chapter concludes with experimental setup of models used for simulations.

4.1 Overview of proposed approach

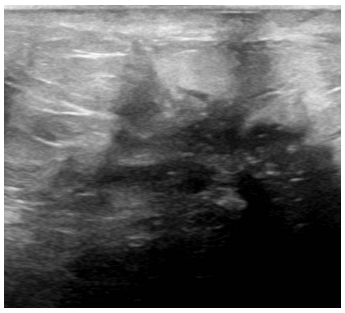
The study design of this project developed gradually as the literature review was conducted. Firstly, all the models mentioned in the literature review of US SISR take input (LR images) with the assumption of 'ideal' downsampling. This means that the test set comprised LR images that were obtained by downsampling of HR images through a known kernel or method which was usually bicubic interpolation. One of the test sets in this project has images from this 'ideal-downsampling' of HR images. Further, to measure the performance in an unexpected case, existing networks and proposed models were tested with two other test sets having images from 'non-ideal downsampling' methods. An unexpected case is when there is no information regarding the means by which the LR images were formed (from the HR images). The 'non-ideal downsampling' methods refer to methods other than bicubic downsampling.

Another issue was the lack of established benchmark models in the US SISR case. As per the literature review conducted, currently, there does not exist any network that has been widely accepted as the standard for US SISR. Therefore, three state-of-the-art models from natural image super resolution were tested beside a US SISR method. All four of these models were used as a benchmark as well as inspiration for models developed in this project.

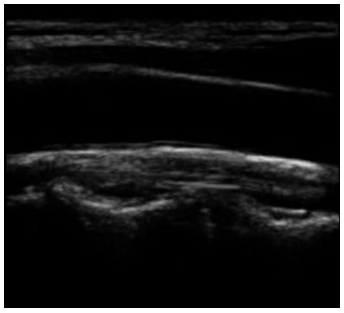
4.2 Dataset

Two publicly available datasets have been used in this project: breast ultrasound image dataset from Kaggle [1], and CCA-US [2]. The Kaggle dataset includes breast ultrasound images of 600 women between the ages of 25 - 75 years old.

[1] https://www.kaggle.com/aryashah2k/breast-ultrasound-images-dataset
[2] http://splab.cz/en/download/databaze/ultrasound



(a) A sample US image from kaggle dataset



(b) A sample US image from CCA-US dataset

Figure 4.1: Sample US images

The dataset was formed for classification, detection and segmentation tasks of breast cancer. Therefore, the original dataset consists of 780 images of three categories: normal, malignant and benign breast cancer. An average image is of size 500×500 pixels. No other information including the method and machine used for scans acquisition has been shared. The CCA-US dataset is provided by the signal processing laboratory of Brno University of Technology. It contains 84 B-mode ultrasound images of common carotid artery (CCA) acquired from 10 volunteers of mean age 27.5 ± 3.5 years and weight 76.5 ± 9.7kg. Sonix OP ultrasound scanner was used with linear array transducers of 10 MHz and 14MHz frequencies. The average resolution of images is 390×330 pixels.

4.3 Dataset Augmentation

The dataset augmentation step consists of cropping, downsizing and introducing noise. From US-CCA, 83 images were selected. The image that was not selected had small dimensions, which caused problems in the $\times 4$ upscaling case in KernelGAN (one of the benchmark models). Therefore, to keep a consistency of the test sets across all models, it was omitted. In the Kaggle dataset, 547 suitable images were selected. The images that were omitted either had written text in the middle of the images or huge black boundaries. These image characteristics could obstruct the learning process of LR-HR mapping and therefore were excluded. Therefore, a final dataset of 630 ultrasound images was formed.

The images were then cropped to dimensions (width(w) $\times$ height(h)) divisible by 4. Since upscaling is performed by a factor of $\times 2$ and $\times 4$, this step helps in keeping output image size (images that are super resolved by the models) consistent with original image size (dataset images). For example, consider a test image I_T of 310×162 size. If I_T is downsampled by $\times 1/2$ factor through bicubic interpolation, the dimensions are reduced to 155×81. If this image is again downsampled by $\times 1/2$ factor, further reduced the size of image is 78×40. Let's call this 78×40 image as I_{T_4}. Now, gradually performing SR on I_{T_4} results in sizes $78 \times 40 \ \rightarrow \ 156 \times 80 \ \rightarrow \ 312 \times 160$. Therefore, a discrepancy is possible if image dimensions are not divisible by SR upscaling factor.

The next step involved downsizing the 630 dataset images to form LR-HR pairs. Three methods were used to get the LR images. In what follows, the original images are denoted as I_{HR} while the low-resolution images are denoted

as I_{LR}. The three methods are as follows:

1. Method 1 ('ideal' downsampling):

   In the first method, simple bicubic interpolation (bic) [61] was used to downsize the images by a downscaling factor sf, i.e,

   $$I_{LR} = I_{HR} \downarrow_{(bic)} sf \tag{4.1}$$

2. Method 2 ('non-ideal' downsampling):

   The second method uses lanczos kernel [62] with radius 3 (lan3) to downsize the images. The images were then passed through an anisotropic diffusion filter (f_{ad}) [63] (the default MATLAB anisotropic filter is used) which smooths the images. Lastly, speckle noise was added to the images. The resulting image is of the form: $I_{\text{Noisy image}} = I_{\text{Original image}} + (n_r \times I_{\text{Original image}})$. The noise n_r is a uniformly distributed random noise having dimensions similar to the original image. Each element of the noise matrix is generated independently from a uniform distribution between interval $(-\frac{\sqrt{3}}{10}, \frac{\sqrt{3}}{10})$. The LR image in this method is given as:

   $$I_{LR} = ((I_{HR} \downarrow_{(lan3)} sf) * f_{ad}) + ((I_{HR} \downarrow_{(lan3)} sf) * f_{ad}) \times n_r \tag{4.2}$$

   where $*$ denotes convolution operation.

3. Method 3 ('non-ideal' downsampling):

   The third method is similar to the second method, except it uses a 5×5 Gaussian blurring kernel (k_g) [64] drawn from a Gaussian function having mean 0 and standard deviation 0.6. The LR image therefore can be given as:

   $$I_{LR} = ((I_{HR} \downarrow_{(lan3)} sf) * (f_g) + ((I_{HR} \downarrow_{(lan3)} sf) * f_g) \times n_r \tag{4.3}$$

4.4 Evaluation Metrics

For evaluation of test results, 3 metrics are used: PSNR (Peak Signal to Noise Ratio), SSIM (Structural Similarity), Alex Loss. In what follows, I_{ref} refers to the original/reference image, and I_{der} refers to the derived or predicted image.

1. PSNR :

   PSNR is a conventional pixel-based method used for comparison of I_{der} with I_{ref}. PSNR compares individual pixels in both images. The pixel-wise difference is calculated as:

   $$MSE = \frac{1}{w \times h} \sum_{i=0}^{w-1} \sum_{i=0}^{w-1} [I_{ref} - I_{der}]^2 \tag{4.4a}$$

   $$PSNR = 20 \log \left(\frac{\text{Maximum pixel value}}{\sqrt{MSE}} \right) \tag{4.4b}$$

where w is the width of the image and h is the height of the image.

2. SSIM :

   SSIM is often used to compare I_{der} with I_{ref} based on perceived texture similarity of images. SSIM compares the luminance, contrast, and structure between the two images to give a similarity measure. SSIM values range between $[-1, 1]$ or $[0, 1]$, where the value of 1 indicates high structural similarity and a value of 0/-1 indicates low structural similarity. The consolidated formula for traditional SSIM is:

$$SSIM(I_{ref}, I_{der}) = \frac{(2\ \mu(I_{ref})\ \mu(I_{der}) + C_1)\ (2\ \sigma(I_{ref}I_{der}) + C_2)}{(\mu(I_{ref})^2 + \mu(I_{der})^2 + C_1)\ (\sigma(I_{ref})^2\ \sigma(I_{der})^2) + C_2} \tag{4.5}$$

   where $\mu(\cdot)$ calculates the average over all pixel values of input image, $\sigma(\cdot)$ calculates the standard deviation of all pixel values of input image, $\sigma_{I_{ref}I_{der}}$ calculates the covariance of I_{ref} and I_{der}. Also, C_1 and C_2 are constants to provide stability in case of denominator close to 0. They are given by $C_1 = K_1 \times R$ and $C_2 = K_2 \times R$. Finally, R is the range of data, usually calculated as $2^{\text{Number of bits per pixel}} - 1$. K_1 and K_2 are algorithm parameters and are small constants.

   However, the above equation is not practical as it calculates the SSIM once for the entire image. Therefore, the Mean Structural Similarity Index (MSSIM) is often preferred. MSSIM differs from SSIM in convolving a kernel (typically Gaussian) with I_{ref} and I_{der} image to make a patch by patch comparison. The kernel slides patch-by-patch over the images and calculates the local SSIM, i.e., for each patch. For m total local SSIM calculated, the MSSIM is:

$$MSSIM(I_{ref}, I_{der}) = \frac{1}{m} \sum_{i=1}^{m} SSIM(I_{ref,i}\ , I_{der,i}) \tag{4.6}$$

   In practice, MSSIM is often referred to as SSIM when used. In this project as well, MSSIM with a range of [-1,1] is used and referred to as SSIM .

3. Alex Loss:

   Besides PSNR and SSIM, VGG loss has also become a standard for comparing the perceptual similarity of images in machine learning models. VGG loss has been discussed in 3.1.4. Alex loss is quite similar to VGG loss, except it uses the shallower AlexNet [65] network for comparison of the feature maps of two images. Here, Alex loss is used in place of VGG loss as it takes less memory and according to [66], Alex loss processes visual information similar to a human brain. It is defined as:

$$Alex\ Loss = \frac{1}{W_j H_j} \sum_{x=1}^{W_j} \sum_{y=1}^{H_j} (\phi_j(I_{HR})_{x,y} - \phi_j(\text{Model}(I_{LR}))_{x,y})^2 \tag{4.7}$$

   where $\phi_j(\cdot)$ represents extracted feature map of an image after the j^{th} convolution layer in the network. There-

fore, $\phi_j(I_{HR})$ and $\phi_j(\text{Model}(I_{LR}))$ calculate the feature maps of the original HR image, I_{HR} and reconstructed SR image, $\hat{I}_{HR}$, respectively. W_j and H_j are the width and height of feature map after the j^{th} convolution layer. Here j is 5, meaning the first five convolution layers of AlexNet have been used for deriving the loss.

4.5 SR networks

4.5.1 Experimental setup

From the literature review, four models were tested: ZSSR[28] (Fig. 3.11), KernelGAN[29] (Fig. 3.12), MWCNN[12] (Fig. 3.13) and CycleGAN-US-SISR model[42] (Fig. 3.15) (this is not the official model name, but has been used here for convenience). MWCNN is the only supervised model here, all other models are self-supervised. Supervised models, while good, are restricted to specific training data. Therefore, the focus of this project was to propose models that use self-supervision.

The test set consisted of 40 images picked randomly from the Kaggle dataset (33 images) and US-CCA (7 images). With 3 methods of creating LR images and two upscaling factors, each model was tested 6 times. For training of MWCNN, 590 images were randomly split into training set and evaluation set in the ratio of $5:1$. For ZSSR[3], KernelGAN[4] and CycleGAN-ZSSR[5], original publicly available codes are used with no modifications. ZSSR is run on Tensorflow, while KernelGAN and CycleGAN-ZSSR use Pytorch. For MWCNN, the official code of the model is in MATLAB. Therefore, to avoid longer run-times in MATLAB, an unofficial TensorFlow adaption[6] of the model was used.

4.5.2 Proposed Architecture

Two model architectures are proposed here which will be referred to as Model 1 (M_1) and Model 2 (M_2) in the following text. A similarity between both these models is the use of wavelets in their structure and the use of C-k3s1-64 convolutional layer (3x3 convolution with stride 1 and 64 output channels). The proposed models take inspiration from MWCNN and ZSSR. ZSSR was used for its ability to use internal characteristics of images during training and therefore give good results on images previously 'unseen'. Wavelets were used for their ability to capture frequency and local information [67]. WPD downsampling divides the image into four sub-bands where each sub-band captures average, horizontal and vertical, and diagonal details of the image. Within the network, the CNN filters individually learn patterns on these sub-bands. A common step in all these models is the formation of HR-LR pairs. This method is similar to the one used in ZSSR. The input image, $I_{LR\ original}$ is passed through an augmentation step which performs random shearing, scaling, cropping, rotations, and mirror reflections and creates output $I_{HR\ parent}$. This HR parent is then further downsampled to form LR child, $I_{LR\ child}$. A random normal noise of mean $= 0$ and standard deviation 30

[3]https://github.com/assafshocher/ZSSR
[4]https://github.com/sefibk/KernelGAN
[5]https://github.com/hengliusky/UltraSound_SSSR
[6]https://github.com/chintan1995/Image-Denoising-using-Deep-Learning/blob/main/Models/MWCNN_256x256.ipynb

is added to the LR child. The HR parent and LR child form an HR-LR pair for training.

Model 1

Model 1 (M_1) uses a 14 layer architecture and can be seen in Fig. 4.2. The proposed model starts training on a single test image by generating HR-LR pairs at each epoch. The image-specific network is then trained on 1500 such HR-LR pairs (1500 epochs). Let the HR parent be $I_{HR\ parent}$ and the LR child is denoted by $I_{LR\ child}$. $I_{LR\ child}$ is fed into the model and the loss is evaluated between predicted SR, $M_1(I_{LR\ child})$ and $I_{HR\ parent}$ as follows:

$$Loss = \|I_{HR\ parent} - M_1(I_{LR\ child})\|_1 + [\lambda \times (1 - \frac{1}{1 + VGG_{loss}})] \tag{4.8}$$

where λ is an adjustment constant that scales the term $(1 - \frac{1}{1+VGG_{loss}})$ to a maximum value λ and minimum zero. While training, $\lambda = 10$ was chosen as it gave the best results empirically. The VGG_{loss} is calculated after second convolution layer of block 1. The term $\|I_{HR\ parent} - M_1(I_{LR\ child})\|_1$ calculates the regression loss between the input and predicted image, while $\lambda \times (1 - \frac{1}{1+VGG_{loss}})$ calculates the perceptual quality. As mentioned earlier, lower blocks of VGG-19 network help in capturing the finer details and for the same reason, block 1 has been used here. L1 loss is used because of its relative relaxation in outlier cases, as opposed to L2 loss, in calculating difference between image pixels.

Gradual SR technique, is used for higher upscaling numbers, e.g., $\times 4$ upscaling. In gradual SR, for performing SR on an input image of dimensions $h \times w$ for SR factor $\times 4$ and gradual scale factor 2, the input image will be first upscaled to $2h \times 2w$. This image $2h \times 2w$ is now treated as the input image in the network and will be upscaled to $4h \times 4w$. The model uses 3 x 3 convolution layer and 64 output filters followed by ReLU activation (abbreviated as C-k3s1-64-ReLU) throughout, except two layers: the last layer which is C-k3s1-3-ReLU (3×3 convolution layer with 3 output filters followed by Leaky ReLU) and the 9^{th} layer which is C-k3s1-256-Leaky ReLU. The input image is first passed through four C-3ks1-64 - Leaky ReLU layers. 2D WPD is then performed on the feature maps. Again a $4 \times$ C-3ks1-64 -ReLU block forms the network followed by the 9^{th} layer. After the 9^{th} layer, there is an IWPD step. Following the IWPD upsampling step is a $4 \times$ C-3ks1-64 -ReLU block. There is a skip connection from the 4^{th} layer to the 10^{th} layer. The output layer (C-k3s1-3) also adds a skip connection from the input (identity mapping) before forming the final SR image. The proposed model, while having similarities to MWCNN and ZSSR, is different in the following ways:

- Model depth: MWCNN used here is a 46 layer deep model and ZSSR is a 7 layer model. While the proposed model is a 14 layer model. The model is 14 layered as the intention was to keep it simple as well as introduce wavelets. A 14 layer model, with 4 layers training on the input image, the next 5 layers training on the sub-band images of input, and the next 5 layers training on the IWPD reconstructed image, can achieve this.

- Weight Initialization: Both MWCNN and ZSSR draw out initial weights from Gaussian distributions. While

Model 1's weights are initialised from an orthogonal matrix obtained from QR decomposition of a matrix of random numbers drawn from a Gaussian distribution. This orthogonal initialisation was observed to provide better performance in comparison to initialisation from Gaussian distributions in Model 1. Hu et al. in [68] also prove that orthogonal initialisation provides faster convergence in deep networks as opposed to initialisation from standard Gaussian initialization with iid weights.

- Downsampling: MWCNN performs downsampling (by using WPD) and upsampling (by using IWPD) four times in their network. ZSSR uses no downsampling operation in its network. In the proposed model, downsampling and upsampling are performed once by using WPD and IWPD.

- Loss: MWCNN uses an L2 loss and ZSSR uses an L1 loss. While the proposed model uses a combination of L1 loss and VGG loss as a combination of both can capture pixel-wise difference along with perceptual similarity. Simply using regression-based losses (L1 and L2 loss) updates the model parameters based only on pixel differences which can be stringent. VGG loss compares the images (train and target images) based on higher-level differences, like the style of images, as it is based on feature maps.

- Prediction: ZSSR also uses a self-ensemble method like in [18] for prediction while M_1 does not use this. In self-assemble, the prediction is taken on 8 augmented versions of input (4 rotations + their mirror reflections). The final SR image is the median of those 8 predictions (mean in [18]). It was empirically found that there was not any significant difference when self-assemble was used in M_1, therefore, a simple criterion was preferred.

Model 2

Model 2 takes further inspiration from KernelGAN, except it uses a simpler loss made of binary cross-entropy. Similar to KernelGAN, the objective of the generator is to learn a downsampling kernel specific to the image and the goal of the discriminator is to differentiate between patches from the real data, i.e., $I_{HR\ parent}$ and generated data, i.e., $G(I_{LR\ child})$. The objective of this GAN is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim \text{patches}(I_{HR\ parent})}[\log(D(x))] + \mathbb{E}_{y\sim \text{patches}(I_{LR\ child})}[\log(1-D(y))] \tag{4.9}$$

where $\mathbb{E}_{x\sim \text{patches}(I_{HR\ parent})}$ is the expectation of x being a patch from $I_{HR\ parent}$ and $\mathbb{E}_{y\sim \text{patches}(I_{LR\ child})}$ is the expectation of y being a patch from $I_{LR\ child}$. Similar to GAN, the goal of the generator is to minimise the value function while the goal of the discriminator is maximise its function.

The original objective of the generator which is to minimise $\log(1-D(G(y)))$ causes GAN to be slow in the beginning as the discriminator's job is easy in this phase [2]. This causes slow convergence at the beginning of GAN training. Therefore, in practice, GAN's are trained with the generator's objective as maximising $\log(D(G(y)))$. The objective of generator changes from minimising its discriminator's loss on its generated instances (in $\log(1-D(G(y)))$

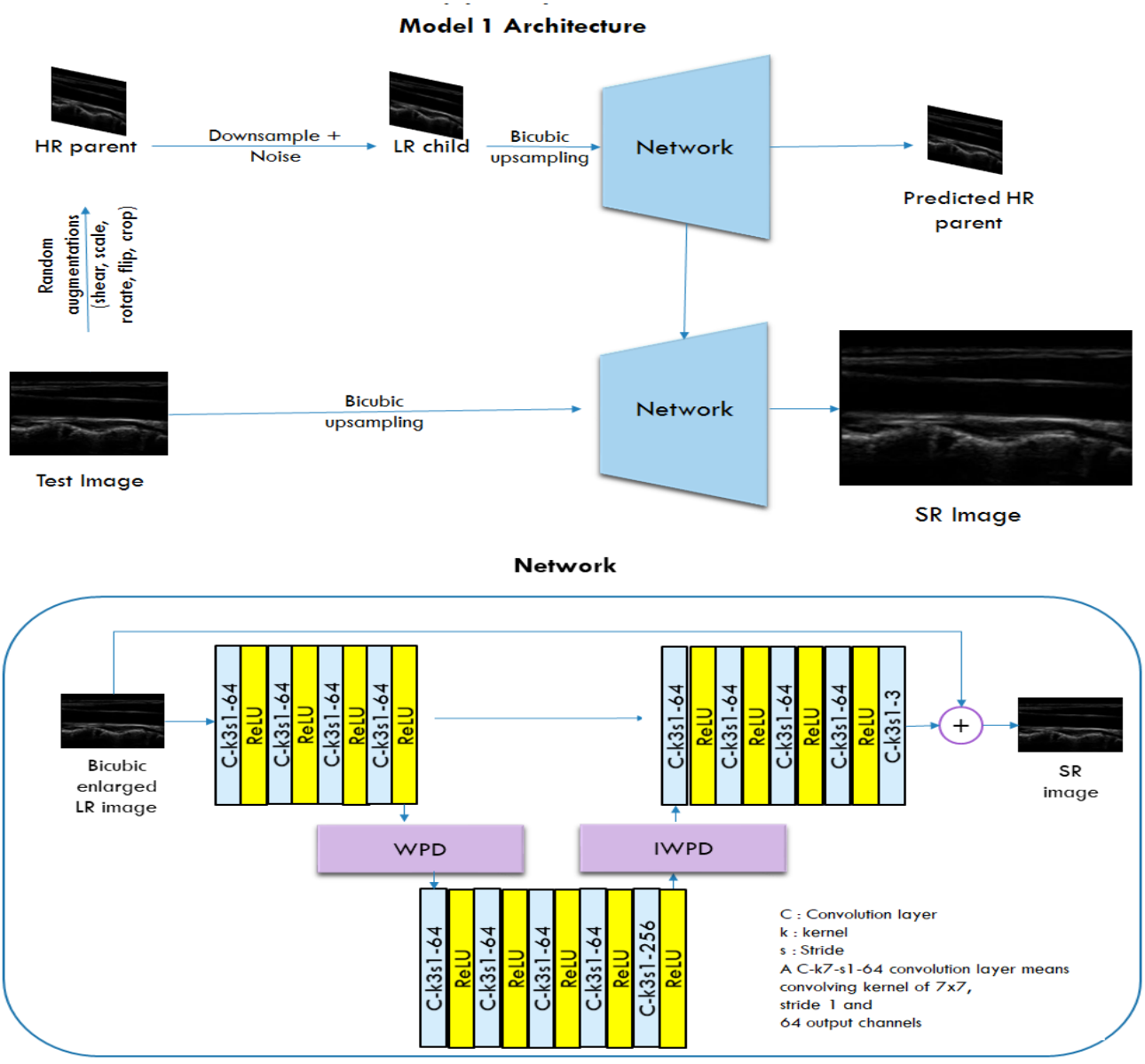


Figure 4.2: Model 1 architecture; WPD and IWPD represent level 2 wavelet packet decomposition and Inverse wavelet packet decomposition respectively.

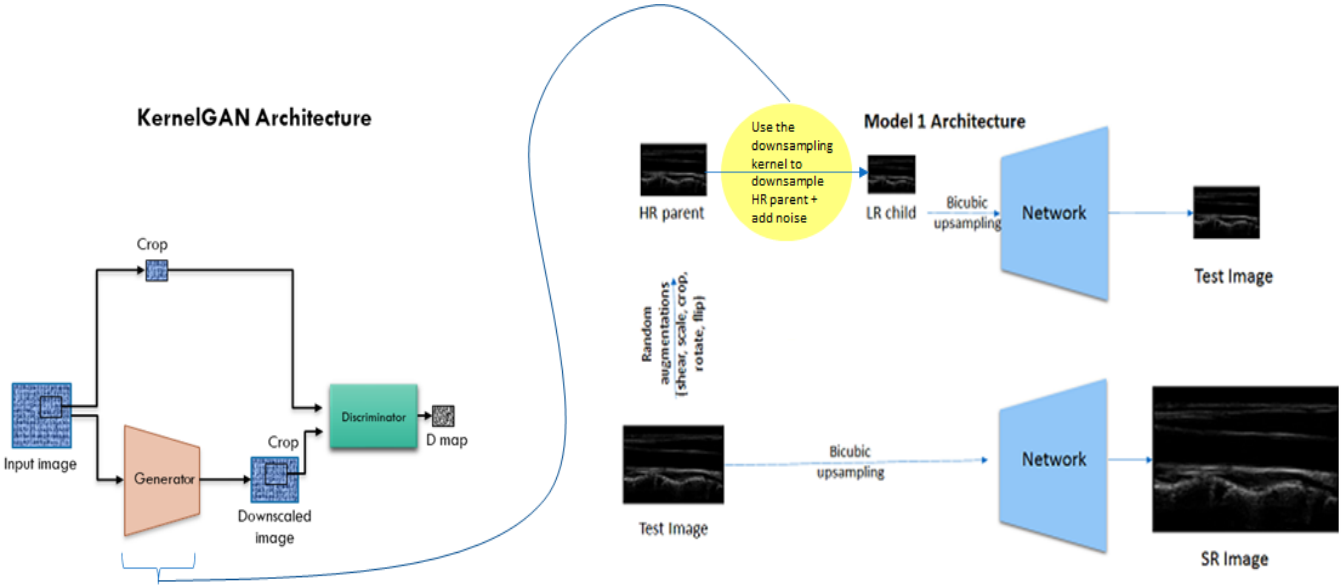


Figure 4.3: Model 2 architecture

to maximising discriminator's output on its generated instances (in $\log D(G(y))$). This means that if the generator can create patches that the discriminator classifies as real, then the generator ha succeeded in its task. The generator loss is then updated as:

$$G_{loss} = -\log(D(G(\text{patches from } I_{LR\ child}))) \tag{4.10}$$

while the loss of the discriminator is updated as:

$$D_{loss} = -\log(\text{patches from } I_{HR\ parent}) + \log(\text{patches from } I_{HR\ child}) \tag{4.11}$$

After the generator has trained on HR-LR patches for 3000 epochs, the weights in the convolution kernel are convolved to create a 2D matrix which is used as a downsampling kernel as in [29]. The image-specific downsampling kernel downsamples the $I_{HR\ parent}$ to create $I_{LR\ child}$ at every epoch. Thus, we avoid the use of bicubic interpolation to create $I_{LR\ child}$. The architecture of Model 2 can be seen in Fig. 4.3.

Chapter 5

Experimental Results and discussion

This chapter reports the experimental results for the models mentioned in the previous chapter, which are ZSSR, KernelGAN, MWCNN, CycleGAN-US-SISR, proposed models 1 and 2. Secondly, a discussion on results from these models on test sets comprised of images from different methods of downsampling is discussed.

5.1 Experimental Results

The experimental results are reported for three methods of obtaining LR and two cases of upscaling the US images. Therefore, the six cases for each model are i.) Method 1, $\times 2$ upscaling, ii.) Method 2, $\times 2$ upscaling, iii.) Method 3, $\times 2$ upscaling, iv.) Method1, $\times 4$ upscaling, v.) Method 2, $\times 4$ upscaling, and v.) Method 3, $\times 4$ upscaling. Based on evaluation with PSNR, SSIM and Alex Loss, the results are reported in Table 5.1, Table 5.2, Table 3 5.3, and Table 5.4. Overall, based on PSNR, SSIM, and Alex Loss, the proposed self-supervised Model 1 performs better in half of the cases of 'non-ideal' downsampling, i.e., method 2 and method 3, and has a comparable performance in the remaining half of 'non-ideal' downsampling cases. Model 2 also produces results similar to Model 1. For the case of 'ideal downsampling', ZSSR and CycleGAN-US-SISR give the best performances, followed by the proposed model 1 and the rest of the models. One downside of the proposed models is unnecessary blurring around the edges. This could be due to the information loss in repeated upsampling and downsampling operations on the input images during the training phase of the models.

The run times of different models are reported in Table 5.5. The runtime of ZSSR is the shortest while the proposed models and KernelGAN (with ZSSR) take the longest runtimes per image.

5.2 Observation and discussion

Downsampling in the case of upscaling by $\times 2$ generally produces better results than the case of $\times 4$ upscaling for all the models. This is due to input images of dimensions approximately 195×165 and 250×250 in dataset 1 and dataset

2 respectively for $\times 2$ scaling. While for $\times 4$ case, dataset 1 has images of size approximately 97×82 and dataset 2 has images of size 125×125. Adding to the size factor, the models (except MWCNN) do not predict using any external reference image.

CCA-US dataset (7 images): $\times 2$ upscaling				
Model	**Method 1**	**Method 2**	**Method 3**	**Evaluation metric**
ZSSR [28]	**45.3611** $\pm$ 2.1210	34.0592 $\pm$ 1.2132	35.5616 $\pm$ 1.8275	PSNR
	0.9893 $\pm$ 0.0000	0.9188 $\pm$ 0.0001	0.9566 $\pm$ 0.0000	SSIM
	0.0338 $\pm$ 0.0000	0.1522 $\pm$ 0.0003	0.1045 $\pm$ 0.0002	Alex Loss
KernelGAN [29] + ZSSR [28]	35.8389 $\pm$ 2.5026	34.3130 $\pm$ 1.8194	37.7225 $\pm$ 2.2358	PSNR
	0.9570 $\pm$ 0.0001	0.9341 $\pm$ 0.0020	0.9657 $\pm$ 0.0000	SSIM
	0.0647 $\pm$ 0.0001	0.1556 $\pm$ 0.0049	**0.0657** $\pm$ 0.0001	Alex Loss
MWCNN [12]	29.1289 $\pm$ 2.5026	31.2508 $\pm$ 2.1547	31.3971 $\pm$ 5.8312	PSNR
	0.9570 $\pm$ 0.0001	0.8561 $\pm$ 0.0020	0.8054 $\pm$ 0.0059	SSIM
	0.1856 $\pm$ 0.0002	0.1556 $\pm$ 0.0049	0.1463 $\pm$ 0.0037	Alex Loss
CycleGAN-US-SISR [42]	44.9243 $\pm$ 2.3342	34.2508 $\pm$ 1.1446	35.7947 $\pm$ 1.8226	PSNR
	0.9886 $\pm$ 0.0000	0.9184 $\pm$ 0.0001	0.9558 $\pm$ 0.0001	SSIM
	0.0289 $\pm$n 0.0001	0.1469 $\pm$ 0.0003	0.0995 $\pm$ 0.0000	Alex Loss
Model1	42.6696 $\pm$ 2.2986	**36.9016** $\pm$ 0.9783	**38.9662** $\pm$ 2.3373	PSNR
	0.9823 $\pm$ 0.0000	**0.9397** $\pm$ 0.0001	**0.9703** $\pm$ 0.0000	SSIM
	0.0781 $\pm$ 0.0001	**0.1105**$\pm$ 0.0002	0.0884$\pm$ 0.0002	Alex Loss
Model2	40.8408 $\pm$ 1.2525	36.3347 $\pm$ 1.0041	37.4261 $\pm$ 2.1760	PSNR
	0.9360 $\pm$ 0.0004	0.9228 $\pm$ 0.0003	0.8949 $\pm$ 0.0005	SSIM
	0.1301 $\pm$ 0.0010	0.1582 $\pm$ 0.0030	0.1657 $\pm$ 0.0014	Alex Loss

Table 5.1: Experimental result for CCA-US dataset, $\times 2$ upscaling. Method 1, 2 ,3 refer to the downsampling methods discussed in 4.3 through which input to the models I_{LR} is formed.

5.2.1 Benchmark models

ZSSR and CycleGAN-US-SISR

Among the benchmarking models, CycleGAN-US-SISR and ZSSR are two networks that give a comparable performance in all the test cases. CycleGAN-US-SISR is using a combination of four losses, while ZSSR uses a simpler L1 loss. With regards to architecture, ZSSR has a simpler architecture with 7 convolution layers and ReLU activations. On the other hand, CycleGAN-US-SISR has a very complex architecture with 4 generators and 2 discriminators. ZSSR is a model designed for SISR of natural images while CycleGAN-US-SISR is a model designed specifically for SISR of US images. All these factors led to the assumption that for SISR of US images, a simpler architecture would be a good choice. Another point to note on why a simpler architecture may work on US images is the inherent nature of these images. US images, as opposed to most natural images, are simple grayscale images. Meaning each image only stores information regarding the intensity of light. On the other hand, a natural image has a larger number of high-frequency regions and colour/pixel value changes in its structure. When performing SISR on natural images, the model must learn on a picture that has more information in all its channels than a US image. Therefore, using an architecture with a lesser number of convolutional layers and a lesser number of output channels in each convolutional layer seems to work well for US SISR. Further, in the case of blind SR and self-supervision, designing a complex loss function

Kaggle dataset (33 images): $\times 2$ upscaling				
Model	**Method 1**	**Method 2**	**Method 3**	**Evaluation metric**
ZSSR [28]	**41.4393** $\pm$ 5.3378	25.6254 $\pm$ 0.0056	26.0542 $\pm$3.4005	PSNR
	0.9788 $\pm$ 0.0000	0.5838 $\pm$ 0.0006	0.6542 $\pm$ 0.0062	SSIM
	0.0243 $\pm$ 0.0000	0.1374 $\pm$ 0.0006	0.1125 $\pm$ 0.0001	Alex Loss
KernelGAN [29]	34.0756 $\pm$ 9.3306	27.7581 $\pm$ 4.1716	28.4265 $\pm$ 4.7513	PSNR
	0.9272 $\pm$ 0.0008	0.6886 $\pm$ 0.0049	0.7611 $\pm$ 0.0047	SSIM
	0.0497 $\pm$ 0.0004	0.0911 $\pm$ 0.0005	0.0824 $\pm$ 0.0002	Alex Loss
MWCNN [12]	31.4682 $\pm$ 18.0504	29.9679 $\pm$ 11.1570	30.7567 $\pm$ 13.5010	PSNR
	0.9109 $\pm$ 0.0008	**0.8254** $\pm$ 0.0011	**0.8810** $\pm$ 0.0006	SSIM
	0.0558 $\pm$ 0.0006	0.0796 $\pm$ 0.0008	0.0543 $\pm$ 0.0003	Alex Loss
CycleGAN-US-SISR [42]	40.9962 $\pm$4.8578	25.4931 $\pm$ 3.0656	25.8789 $\pm$ 3.5966	PSNR
	0.9773 $\pm$ 0.0000	0.5763 $\pm$ 0.0061	0.6439 $\pm$ 0.0068	SSIM
	0.0208 $\pm$ 0.0004	0.1398 $\pm$ 0.0005	0.1062 $\pm$ 0.0001	Alex Loss
Model1	38.5234 $\pm$ 3.9060	**31.3743** $\pm$ 1.5939	**32.9226** $\pm$ 2.8843	PSNR
	0.9563 $\pm$ 0.0002	0.7911 $\pm$ 0.0011	0.8727 $\pm$ 0.0008	SSIM
	0.0460 $\pm$ 0.0003	**0.0687** $\pm$ 0.0007	**0.0536** $\pm$ 0.0003	Alex Loss
Model2	33.3628 $\pm$ 7.8995	30.0942 $\pm$ 1.8006	30.9415 $\pm$ 1.7527	PSNR
	0.8784 $\pm$ 0.0058	0.7495 $\pm$ 0.0034	0.8115 $\pm$ 0.0032	SSIM
	0.0821 $\pm$ 0.0015	0.1045 $\pm$ 0.0036	0.0778 $\pm$ 0.0015	Alex Loss

Table 5.2: Experimental result for the Kaggle dataset, $\times 2$ upscaling. Method 1, 2 ,3 refer to the downsampling methods discussed in 4.3 through which input to the models I_{LR} is formed.

seems redundant because of two reasons. First, there is no information regarding the source and method of obtaining I_{LR} images, i.e., downsampling kernel and type of noise. Secondly, in the case of self-supervision, there is also no reference/target image that could give us an accurate loss. Owing to these factors, the models proposed in this project use simple losses and simple architectures.

MWCNN

MWCNN is the only supervised model in the benchmarking models. Despite having an explicit training phase, it has an average performance. However, it does show consistency in all test cases, as there are no unexpected cosmetic changes in the predicted images. One of the limitations of the model lies in its training phase. To train the model in mini-batches, each image in the dataset needs to be interpolated to a fixed size. When interpolating an already interpolated input image (I_{LR}), there is a further loss of information in the dataset. However, MWCNN was not originally designed for SISR. MWCNN is a restoration model to produce clean images from corrupt/noisy images. With its use of wavelet downsampling and upsampling in lieu of traditional pooling operations, MWCNN can remove noise from images and fairly retains the structure well. For this reason, the wavelet downsampling and upsampling techniques were used in the proposed models.

KernelGAN

A completely different model from the other benchmark models is KernelGAN. It was designed for learning the downsampling kernel for the blind SR problem. The model was previously used in blind SR of natural images and has

CCA-US dataset (7 images): $\times 4$ upscaling				
Model	**Method 1**	**Method 2**	**Method 3**	**Evaluation metric**
ZSSR [28]	35.6793 $\pm$ 2.4680	31.2926 $\pm$ 0.8435	**32.2217** $\pm$ 1.7551	PSNR
	0.9467 $\pm$ 0.0001	**0.8506** $\pm$ 0.0004	**0.8979** $\pm$ 0.0003	SSIM
	0.0808 $\pm$ 0.0002	**0.1869** $\pm$ 0.0006	**0.1621** $\pm$ 0.0003	Alex Loss
KernelGAN [29]	26.1786 $\pm$ 2.3820	26.0638$\pm$ 2.2.3605	29.1976 $\pm$3.4894	PSNR
	0.8321 $\pm$ 0.0010	0.8199 $\pm$ 0.0009	0.8694 $\pm$ 0.0006	SSIM
	0.2074 $\pm$ 0.0009	0.2379 $\pm$ 0.0004	0.1678 $\pm$ 0.0005	Alex Loss
MWCNN [12]	28.2801 $\pm$ 7.4760	26.9909 $\pm$ 1.9565	26.5557 $\pm$ 5.8312	PSNR
	0.7838 $\pm$ 0.0090	0.5567 $\pm$ 0.0042	0.7227 $\pm$ 0.0058	SSIM
	0.1896 $\pm$ 0.0059	0.2372 $\pm$ 0.0020	0.2249 $\pm$ 0.0057	Alex Loss
CycleGAN-US-SISR [42]	**36.5823** $\pm$ 2.8956	31.3216$\pm$ 0.9152	31.9597 $\pm$ 2.1805	PSNR
	0.9503 $\pm$ 0.0001	0.8468 $\pm$ 0.0004	0.8943 $\pm$ 0.0004	SSIM
	0.0975 $\pm$0.0002	0.1913 $\pm$ 0.0010	0.1685 $\pm$ 0.0004	Alex Loss
Model1	33.8589 $\pm$ 2.3945	**32.0137** $\pm$ 1.4641	31.7447 $\pm$ 2.0199	PSNR
	0.8827 $\pm$ 0.0001	0.8425 $\pm$ 0.0003	0.8593 $\pm$ 0.0003	SSIM
	0.2556 $\pm$ 0.0011	0.2769 $\pm$ 0.0016	0.2736 $\pm$ 0.0011	Alex Loss
Model2	33.4878 $\pm$ 1.8064	31.7291 $\pm$ 1.0909	31.1756 $\pm$ 0.6893	PSNR
	0.8795 $\pm$ 0.0008	0.8402 $\pm$ 0.0010	0.8353 $\pm$ 0.0006	SSIM
	0.2313 $\pm$ 0.0014	0.2361 $\pm$ 0.0009	0.2755 $\pm$ 0.0009	Alex Loss

Table 5.3: Experimental result for CCA-US dataset, $\times 4$ upscaling. Method 1, 2 ,3 refer to the downsampling methods discussed in 4.3 through which input to the models I_{LR} is formed.

given exceptional performance in NTIRE 2020 challenge [69]. Using KernelGAN to learn the downsampling kernel of input I_{LR} images and then using the obtained kernel in ZSSR has given good performance on US images. Based on performance metrics (PSNR, SSIM, Alex Loss), KernelGAN + ZSSR may not give the best performance, however, visually images derived from KernelGAN are able to retain the structure well in the majority of the cases. Therefore, in proposed model 2, a KernelGAN with simpler loss is designed to learn the downsampling kernel of US images and then tested with architecture from proposed model 1.

5.2.2 Performance based on downsampling method

Ideal downsampling

For this case, CycleGAN-US-SISR and ZSSR outperform other models as the model architectures already assume that the input I_{LR} images are a result of bicubic interpolation. Model 1 also gives a good performance. For $\times 4$ case, the gap between CycleGAN-US-SISR and ZSSR and proposed Model 1 is lower than in the $\times 2$ case. While images from CycleGAN-US-SISR and ZSSR are near-perfect in comparison to the original, proposed Model 1 images have an over smoothing effect near the edges. KernelGAN also produces good images, however, the produced images are perceived to have higher contrast and sharper edges than original images, especially in the $\times 4$ case. Images produced by MWCNN are also good visually. Lastly, the proposed Model 2 gives similar performance to Model 1. A side by side comparison of images predicted from these models can be seen in Fig. 5.1 for $\times 2$ scaling and Fig. 5.2 $\times 4$ scaling. Therefore, for ideal cases, a simpler model is perfectly capable of producing SR images without indulging in complicated losses.

Kaggle dataset (33 images): $\times 4$ upscaling				
Model	**Method 1**	**Method 2**	**Method 3**	**Evaluation metric**
ZSSR [28]	**33.2300** $\pm$ 5.2050	24.8055 $\pm$ 2.5436	25.1098 $\pm$ 3.0107	PSNR
	0.8809 $\pm$ 0.0007	0.5298 $\pm$ 0.0049	0.5817 $\pm$ 0.0054	SSIM
	0.0677 $\pm$ 0.0010	0.2001 $\pm$ 0.0005	0.1958 $\pm$ 0.0002	Alex Loss
KernelGAN [29]	25.9043 $\pm$ 10.3544	24.1170$\pm$ 2.5436	25.5240 $\pm$4.6654	PSNR
	0.7110 $\pm$ 0.0080	0.5528 $\pm$ 0.0067	0.6358 $\pm$ 0.0048	SSIM
	0.0912 $\pm$ 0.0008	0.1431 $\pm$ 0.0008	0.1310 $\pm$ 0.0008	Alex Loss
MWCNN [12]	29.2626 $\pm$ 14.4879	27.1243 $\pm$ 8.3364	26.5236 $\pm$ 10.6172	PSNR
	0.8482 $\pm$ 0.0011	**0.7089**$\pm$ 0.0018	0.7277 $\pm$ 0.0058	SSIM
	0.0791 $\pm$ 0.0009	**0.1204** $\pm$ 0.0013	0.1216 $\pm$ 0.0012	Alex Loss
CycleGAN-US-SISR [42]	32.9072 $\pm$ 4.7455	24.7451$\pm$ 2.5508	25.0184 $\pm$ 3.0784	PSNR
	0.8744 $\pm$ 0.0007	0.5257$\pm$ 0.0050	0.5759 $\pm$ 0.0058	SSIM
	0.0713 $\pm$0.0011	0.1997 $\pm$ 0.0005	0.1904 $\pm$ 0.0003	Alex Loss
Model1	31.4986 $\pm$ 3.9234	**28.7633** $\pm$ 2.1375	**29.0328** $\pm$ 3.3374	PSNR
	0.8171 $\pm$ 0.0013	0.7046 $\pm$ 0.0017	**0.7439** $\pm$ 0.0020	SSIM
	0.1082 $\pm$ 0.0010	0.1206 $\pm$ 0.0020	**0.1109** $\pm$ 0.0014	Alex Loss
Model2	30.8769 $\pm$ 3.3308	28.5800 $\pm$ 2.4044	28.6963 $\pm$ 3.1207	PSNR
	0.7863 $\pm$ 0.0015	0.6984 $\pm$ 0.0017	0.7201 $\pm$ 0.0019	SSIM
	0.1265 $\pm$ 0.0025	0.1226 $\pm$ 0.0027	0.1417 $\pm$ 0.0037	Alex Loss

Table 5.4: Experimental result for Kaggle dataset, $\times 4$ upscaling. Method 1, 2 ,3 refer to the downsampling methods discussed in 4.3 through which input to the models I_{LR} is formed.

Model	**Runtime**
ZSSR	X2, X4 : $\sim$ 33 seconds per image
KernelGAN (with ZSSR)	X2, X4 : $\sim$ 300 seconds per image
CycleGAN-US-SISR	X2, X4 : $\sim$ 72 seconds per image
MWCNN	X2, X4: Train time : $\sim$ 2 hours 42 minutes Test time : $\sim$ 0.9 seconds per image
Model 1	X2 : $\sim$ 90 seconds per image X4 : $\sim$ 142 seconds per image
Model 2	X2 : $\sim$ 300 seconds per image X4 : $\sim$ 450 seconds per image

Table 5.5: Running times of models on NVIDIA Tesla K80 GPU

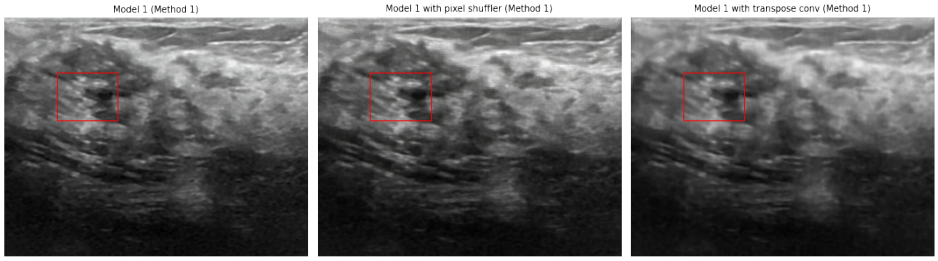
Non-ideal downsampling

For the non-ideal downsampling methods, proposed model 1 gives a better performance in half of the cases. In the remaining cases, its performance is second-best in almost all cases. Proposed model 2 also gives similar performance. In both models 1 and 2, some images have blurring around the edges. The next best performance, perceptually, is given by KernelGAN. For the $\times 2$ case, KernelGAN is very good at retaining the structure and producing an SR image that matches the original image. However, in cases of $\times 4$ upscaling, it sometimes makes cosmetics changes around the edges. This is because the input image has unexpected noises and effects which the kernelGAN kernel learns as characteristics of the image and while super-resolving, these characteristics are enhanced. Proposed model 2 is able to avoid this to an extent because of the sub-bands in its SR network. Similarly, MWCNN gives an average performance but has smoother edges. Meanwhile, CycleGAN-US-SISR and ZSSR which gave the best performances in ideal downsampling case, predict SR images with noise in them. This is mainly due to their losses which are centered

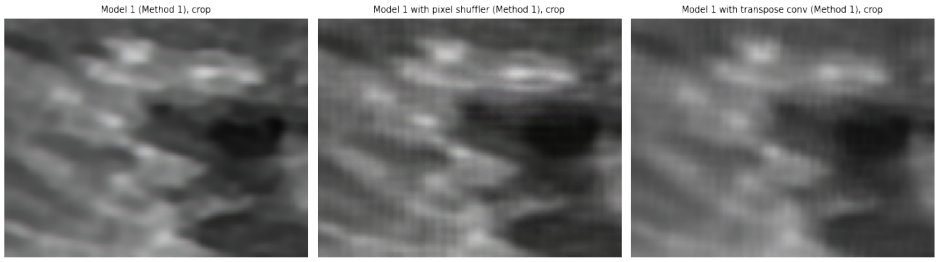
around bicubic up-sampled I_{LR} images. A side-by-side comparison of images predicted from these models can be seen in Fig. 5.5, Fig. 5.6 for method 2 and Fig. 5.3, Fig 5.4 for method 3. Therefore, in non-ideal cases of SR, besides the CNN network, other image processing steps are crucial.

5.2.3 Discussion on Proposed Models

Both the proposed models were designed while keeping blind SR and non-ideal downsampling cases of US SISR in mind. Proposed Model 1, which is the simplest model in terms of architecture and loss further gives a good performance. It can efficiently perform SISR while maintaining the similarity to original images in most cases. One of the main reasons for blurring is the repeated bicubic interpolation in the model. First, the LR image is obtained by downsampling the HR image, then again it is upsampled before input to the network. The model also does not use any pre-processing or post-processing on the images to detect edges or remove noise. To avoid repeated bicubic interpolation, transposed convolution and pixel shuffler were also tried within Model 1. The model variant with pixel shuffler and transposed convolution avoided bicubic upsampling of $I_{LR\ child}$ before network by performing upsampling within the DL network. The results based on performance metrics were slightly lower than Model 1 and Model 2 (as tested for $\times 2$ case). However, there were pronounced checkerboard effects in the resulting images. This can be seen in Fig 5.7.



(a) Model-wise predicted images



(b) Crops from the above predicted images

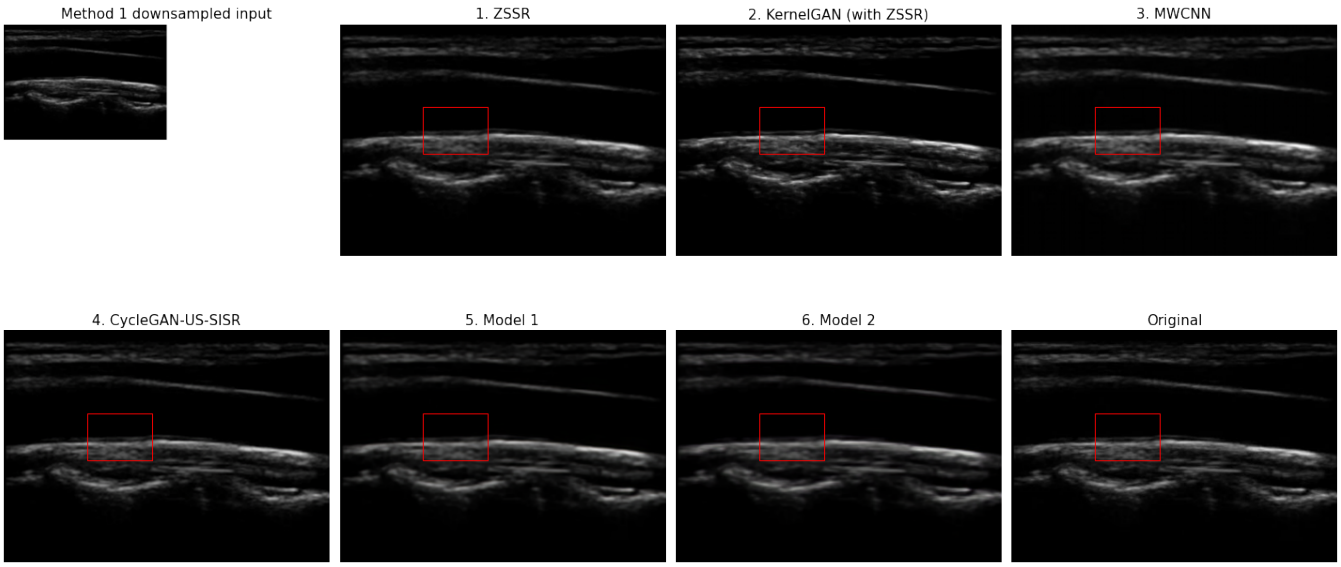
Figure 5.7: A predicted image on different models for the case of 'non-ideal' case (method 2) input on dataset 1, $\times 4$ upscaling. Here checkerboard effects can be observed in case of pixel shuffle and transpose convolution.

KernelGAN inspired Model 2 also gives a comparable performance to Model 1. This further confirms the assumption that for US images, a simpler model may perform better than an overly complex model. Further, the data augmentation step which forms the HR parent and LR child while training the model is crucial. It was observed that results were

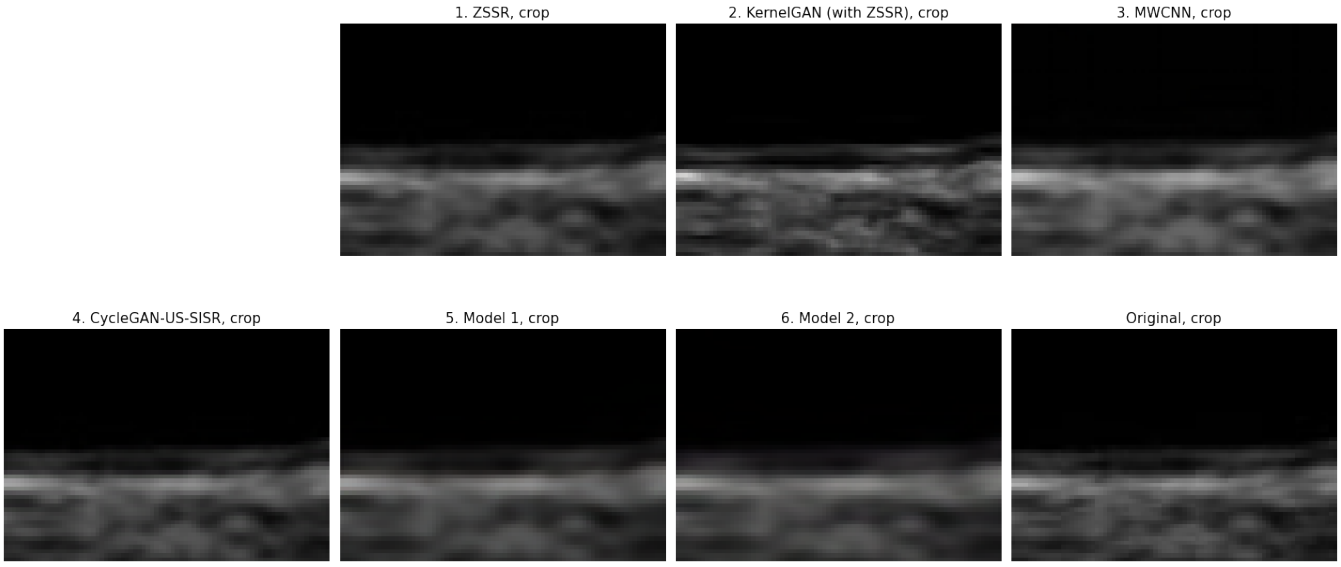
enhanced in Model 1 when random shearing and scaling were introduced in addition to random cropping, flipping, and rotation.

Model	Code Link if online code used	Model hyperparamteres
ZSSR (self-supervised model)	Official TensorFlow code	**Initialization**: Weights: From normal distribution of mean = 0 and std = 0.01 Bias: 0 **Loss**: L1 loss **Max epoch**: 3000 **Optimizer**: ADAM **Learning Rate (LR)**: Initial LR: 1e-3 Minimum LR: 9e-9 **Monitor Quantity**: Linear fit of reconstruction error every 60 iteration The LR drops by a factor of 1.5 if std of reconstruction error is greater than slope of monitor quantity
KernelGAN (self-supervised model)	Official PyTorch code	**Initialization**: Weights: Xavier Normal with gain = 0.1 Bias: 0 **Generator Loss**: Combination of 6 losses **Discriminator Loss**: L2 loss **Max epoch**: 3000 **Optimizer**: ADAM with beta 1 = 0.5, beta 2 = 0.999 LR: 2e-4
MWCNN (supervised model)	Unofficial TensorFlow code	**Initialization**: Weights: Xavier Uniform Bias: 0 **Loss**: L2 loss **Epoch**: 100 **Optimizer**: ADAM **LR**: Initial LR: 1e-4 Minimum LR: 1e-6 **Monitor quantity**: Validation loss The LR drops by a factor 0.70 if there is no improvement(change of 1e-4) in monitor quantity for 3 epochs
CycleGAN-US SISR (self-supervised model)	Official PyTorch code	**Initialization**: Weight: From normal distribution of mean = 0 and std = 0.02 Bias: 0 **Loss**: total loss = l1 loss + VGG loss + adversarial loss + Cycle loss **Max epoch**: 3000 **Optimizer**: ADAM **LR**: Initial LR: 1e-3 Minimum LR: 9e-6 **Monitor Quantity**: Linear fit of reconstruction error \cite{} every 60 iteration. The LR drops by a factor of 1.5 if std of reconstruction error is greater than slope of monitor quantity
Model 1 (self-supervised model)	-	**Initialization**: Weight: Orthogonal Bias: 0 **Loss**: L1 loss + VGG Loss **Max Epoch**: 1500 **Optimizer**: ADAM with beta 1 = 0.9, beta 2 = 0.999 **LR**: Initial LR: 1e-3 Minimum LR: 1e-8 **Monitor Quantity**: Loss between HR-LR pair The LR drops by a factor of 2 if there in no improvement(change of 1e-3) in 100 iterations
Model 2 (self-supervised model)	-	**Initialization**: Weight: Xavier uniform Bias: 0 **Generator Loss**: Cross-entropy loss **Discriminator Loss**: Cross-entropy loss **Epoch**: 3000 **Optimizer**: ADAM with beta 1 = 0.5, beta 2 = 0.999 **LR**: 2e-4

Table 5.6: Model Hyperparameters; The hyperparameter for KernelGAN and Model 2 are for the GAN network only. KernelGAN uses ZSSR for image prediction while Model 2 uses Model 1 for prediction.

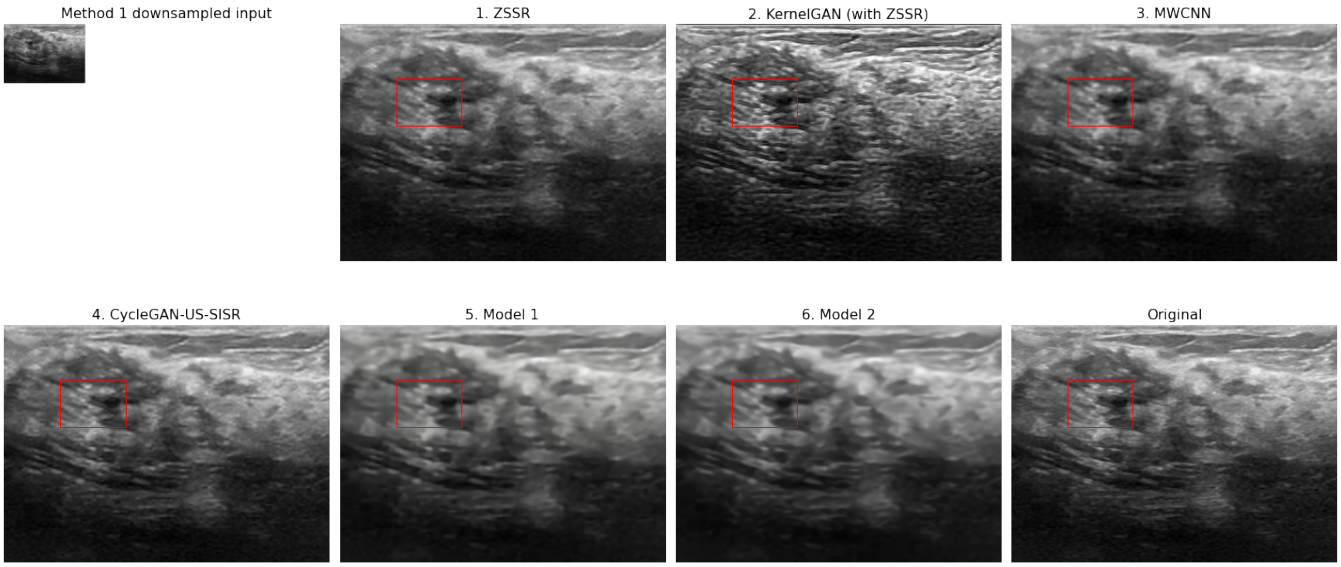


(a) Model-wise predicted images

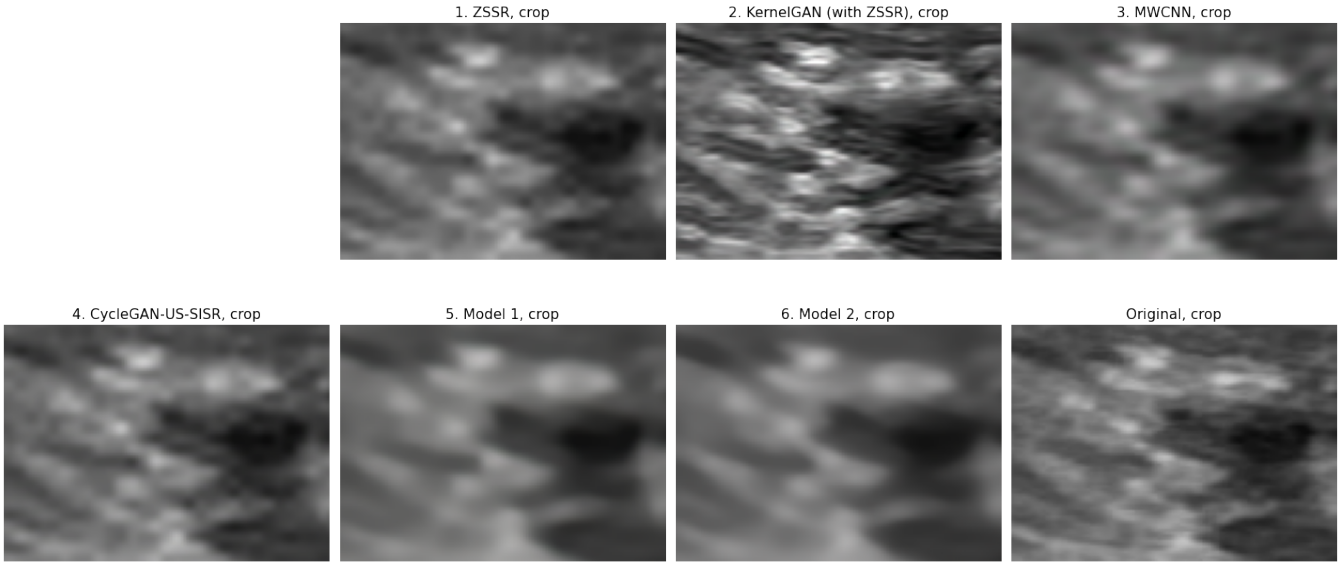


(b) Crops from the above predicted images

Figure 5.1: A predicted image on different models for the case of 'ideal' case (method 1) input on CCA-US dataset, $\times 2$ upscaling

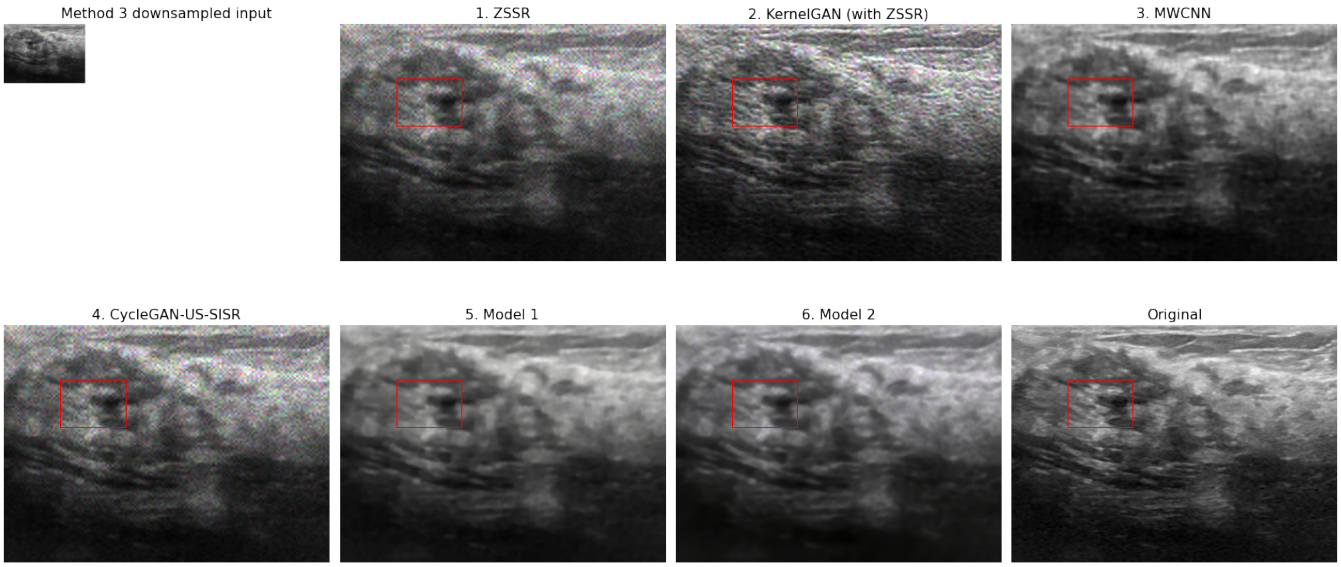


(a) Model-wise predicted images

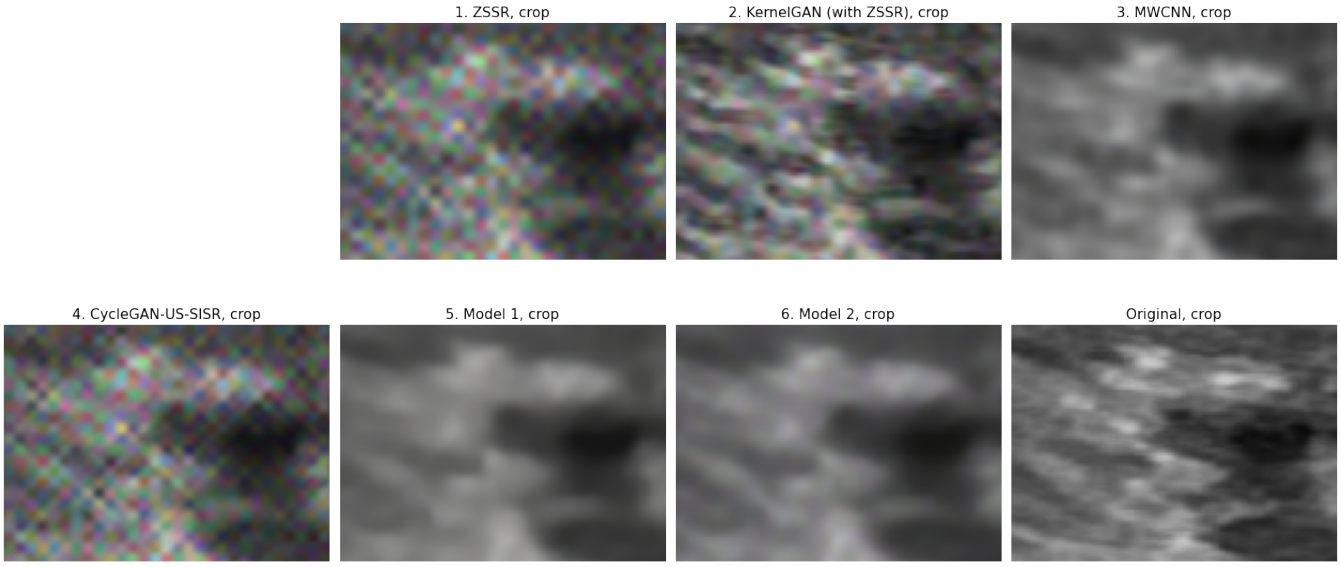


(b) Crops from the above predicted images

Figure 5.2: A predicted image on different models for the case of 'ideal' case (method 1) input on the Kaggle dataset, $\times 4$ upscaling.

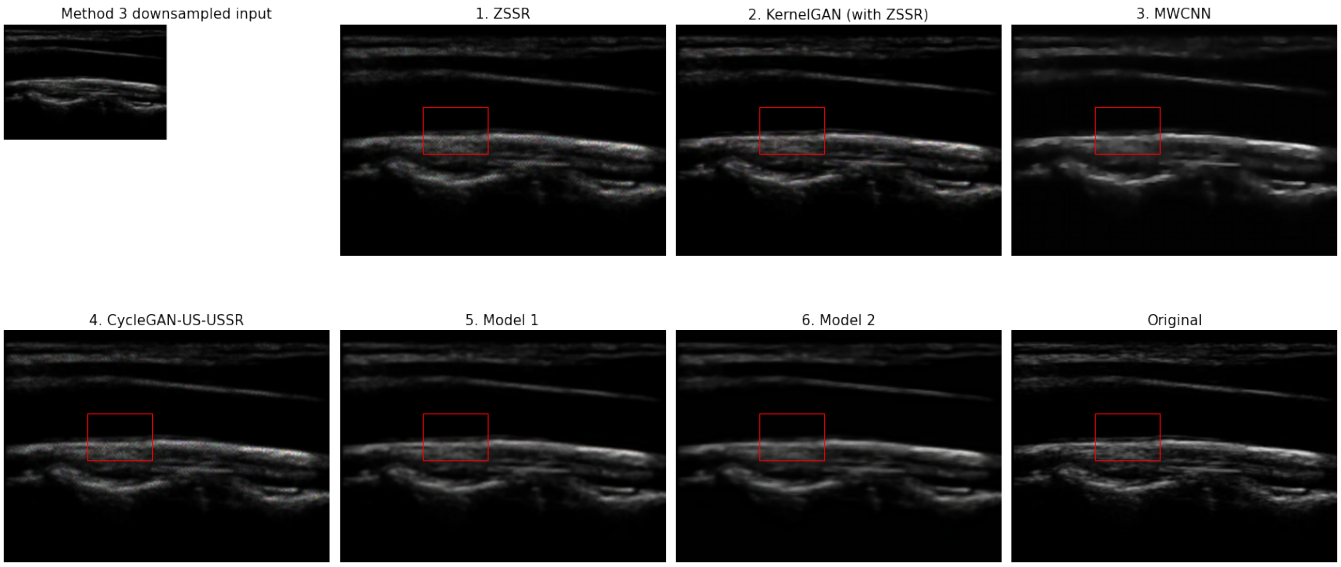


(a) Model-wise predicted images

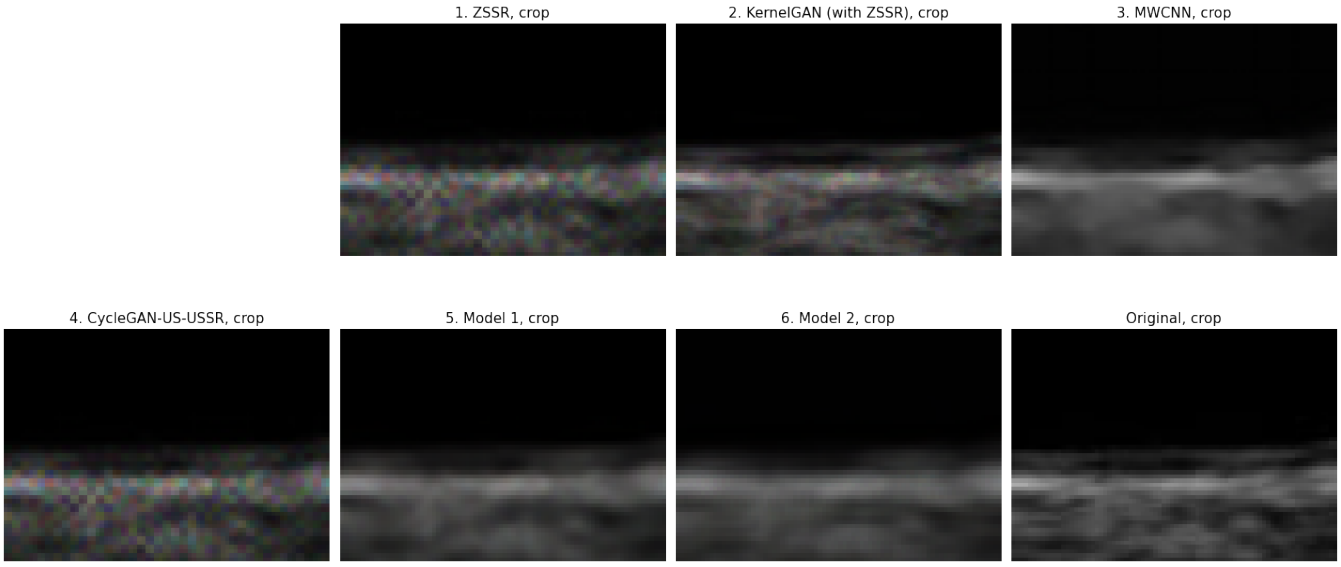


(b) Crops from the above predicted images

Figure 5.3: A predicted image on different models for the case of 'non-ideal' case (method 3) input on the Kaggle dataset, $\times 4$ upscaling.

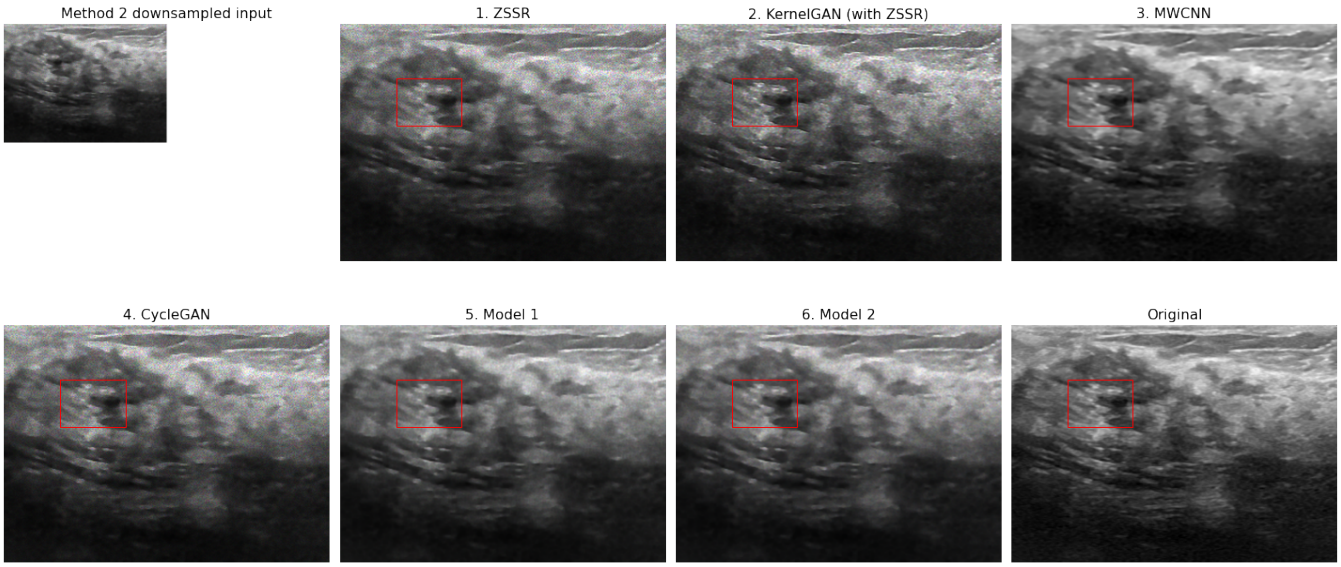


(a) Model-wise predicted images

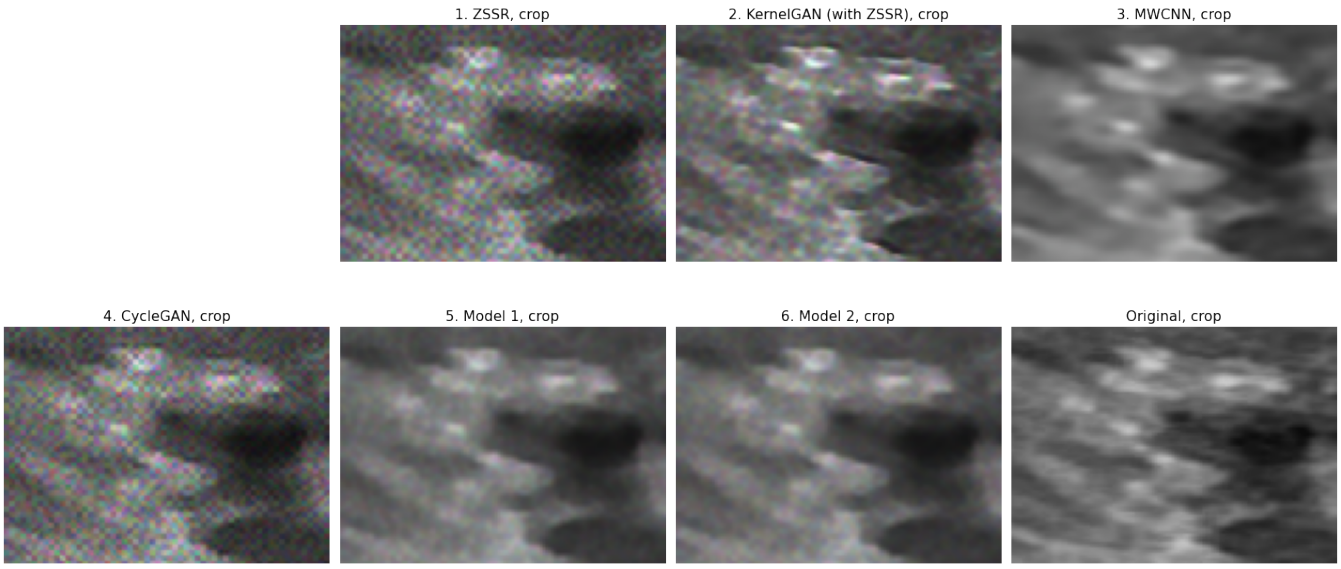


(b) Crops from the above predicted images

Figure 5.4: A predicted image on different models for the case of 'non-ideal' case (method 3) input on CCA-US dataset, $\times 2$ upscaling.

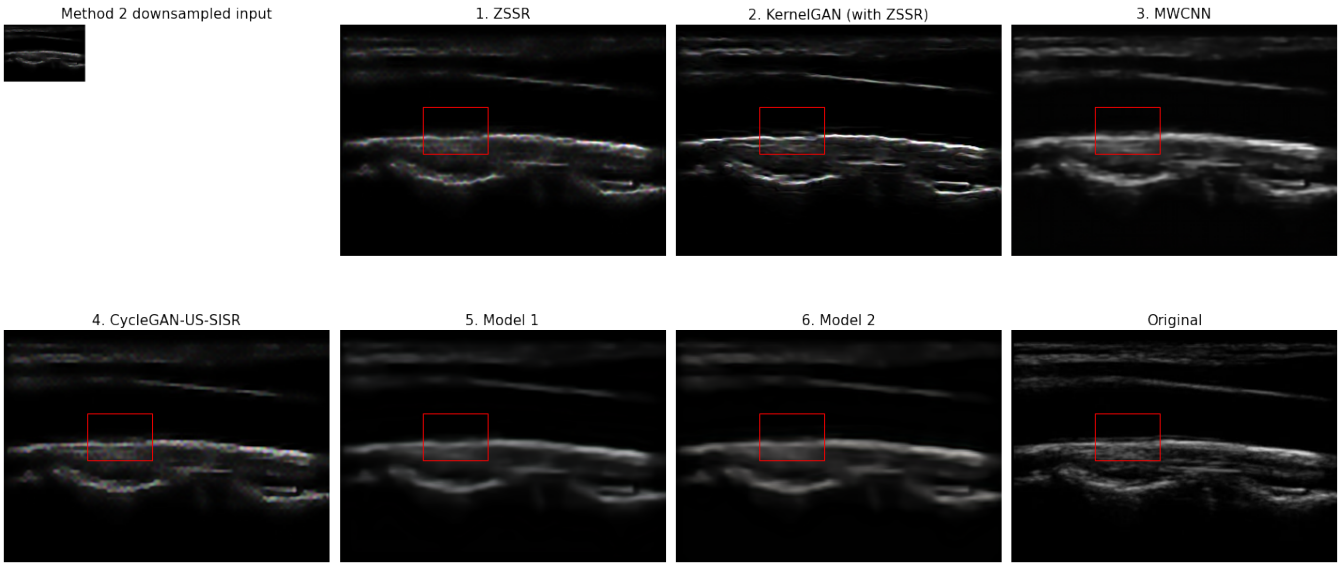


(a) Model-wise predicted images

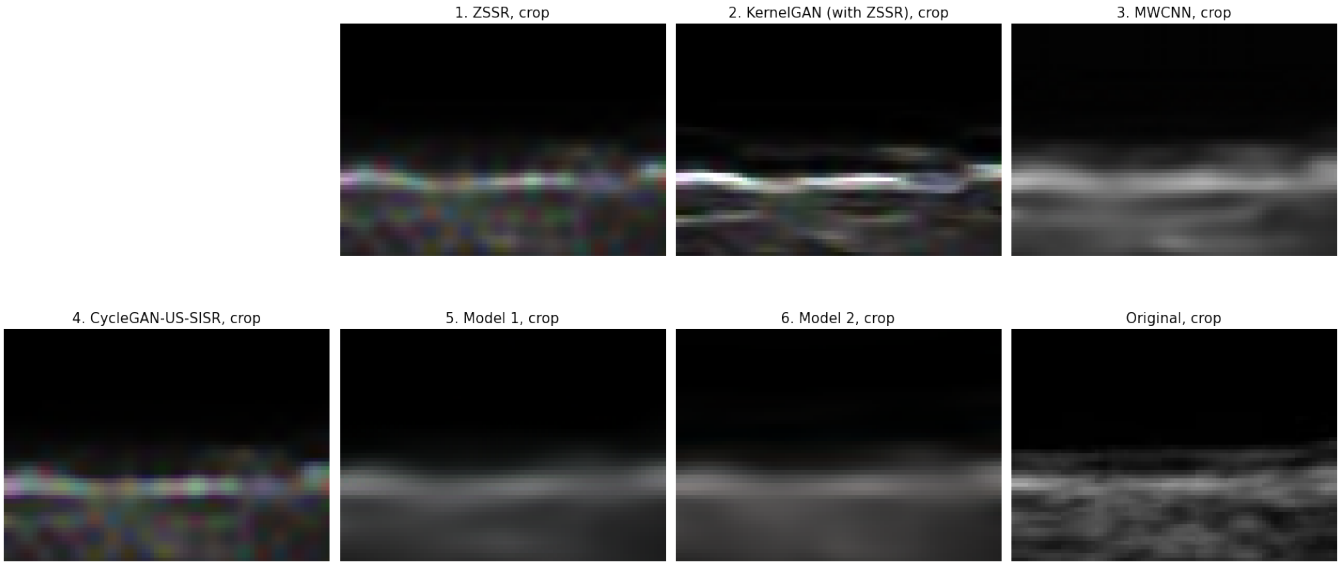


(b) Crops from the above predicted images

Figure 5.5: A predicted image on different models for the case of 'non-ideal' case (method 2) input on the Kaggle dataset, $\times 2$ upscaling.



(a) Model-wise predicted images



(b) Crops from the above predicted images

Figure 5.6: A predicted image on different models for the case of 'non-ideal' case (method 2) input on dataset 1, $\times 4$ upscaling.

Chapter 6

Conclusion

This report proposes two models for single image super-resolution of medical ultrasound images. The proposed models are self-supervised and do not need any extensive external training and test datasets. The proposed models use training methods like those used in ZSSR and KernelGAN and combine these methods with wavelet upsampling and downsampling like used in MWCNN. Proposed Model 1 combines ZSSR and wavelets, while Model 2 performs SR in two parts. It first uses a KernelGAN with simpler loss to obtain a downsampling kernel. Thereafter, this downsampling kernel is used in Model 1 which performs the SISR of US images. The use of wavelets seems to have a positive effect in restoring the images, but there is also certain loss around high-frequency regions, like edges, in the image.

The models are benchmarked against three natural images and one US image SISR model. Based on quantitative evaluation metrics (PSNR, SSIM, and Alex Loss), the proposed models give good performance in the case of blind SR test cases. The US images from the models do not contain excess noise and retain the original structure of the image. However, visually there are blurred edges in the images. In the case of performing SR on ideal LR images, the proposed models give worse results than benchmarking models. However, the results of the proposed Model 1 are still comparable to the benchmarking models. Based on experimental results, a simple model with a simple loss function, like Model 1, would be well suited for US SISR.

6.1 Future work

One of the limitations of both models is the bicubic upsampling of LR images before feeding them into the model for super-resolution. There could be merit in exploring alternate options to bicubic upsampling, or exploring other interpolation techniques that give better results in upsampling cases. Further, in this project, all the wavelet transformation has been performed with Haar wavelet which is one of the simplest wavelets in the wavelet family. Testing the models on different wavelets to possibly find a wavelet that is better suited for US images could improve the performance. Additionally, Model 2 which learned a downsampling kernel on the input images was a method

from blind SR of natural image and did not significantly enhance the performance of the model. However, exploring other blind SR techniques from natural image models and then catering them to the US SISR problem still seems like a viable step. One such model is FSSR [70] which according to [71] predicts better natural images than KernelGAN (with ZSSR).

Besides these changes in the proposed models, exploring methods specific to the general US image processing pipeline can be studied and included in the DL models. Specifically, edge detection is recommended to retain sharper edges in the images.

Further, as seen in the resulting images, a complete blind SR of US images may not be a very good idea if performing self-supervised training. There needs to be an image pre-processing step based on certain assumptions, like the presence of noise, that pre-processes the input image before forming the HR parent.

References

[1] M. Ali, D. Magee, and U. Dasgupta, "Signal processing overview of ultrasound systems for medical imaging," Texas Instruments, Tech. Rep. SPRAB12, Nov. 2008.

[2] I. J. Goodfellow, J. P. Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, ser. NIPS'14, vol. 2. Cambridge, MA, USA: MIT Press, 2014, pp. 2672–2680.

[3] D. Chao, L. Chen, H. Aiming, and T. Xiaoou, "Learning a deep convolutional network for image super-resolution," in *Computer Vision – ECCV 2014*. Cham: Springer International Publishing, 2014, pp. 184–199.

[4] R. Fattal, "Image upsampling via imposed edge statistics," *international Association for Computing Machinery's Special Interest Group on Computer Graphics and Interactive Techniques (ACM SIGGRAPH) 2007 papers*, vol. 26, no. 3, pp. 95–es, Jul 2007.

[5] H. Zhang, J. Yang, Y. Zhang, and T. Huang, "Non-local kernel regression for image and video restoration," in *Computer Vision – ECCV 2010*. Berlin, Heidelberg: Springer, 2010, pp. 566–579.

[6] H. Chang, D.Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*, vol. 1, Jan. 2004, pp. I–275.

[7] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," *Computing Research Repository (CoRR)*, vol. abs/1608.00367, 2016.

[8] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1646–1654.

[9] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, 2015, pp. 730–734.

[10] H. Y. Feng, J. P. Wang, Y. C. Li, and J. Chen, "Wavelet theory and application summarizing," in *Information Computing and Applications*. Berlin, Heidelberg: Springer, 2011, pp. 337–343.

[11] W. Bae, J. Yoo, and J. C. Ye, "Beyond deep residual learning for image restoration: Persistent homology-guided manifold simplification," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE Computer Society, 2017, pp. 1141–1149.

[12] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-CNN for image restoration," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 886–88 609.

[13] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1637–1645.

[14] S. Richard, P. Alex, W. Jean, C. Jason, M. D. Christopher, N. Andrew, and P. Christopher, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642.

[15] W. Shiand J. Caballeroand F. Huszárand J. Totzand Andrew P. Aitkenand R. Bishopand D. Rueckert and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," *Computing Research Repository (CoRR)*, vol. abs/1609.05158, 2016.

[16] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 105–114.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[18] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jul 2017, pp. 1132–1140.

[19] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2790–2798.

[20] W. Lai, J. Huang, N. Ahuja, and M. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," *Computer Research Repository (CoRR)*, vol. abs/1704.03915, 2017.

[21] E. Adelson, C. Anderson, J. Bergen, P. Burt, and J. Ogden, "Pyramid methods in image processing," *RCA Eng.*, vol. 29, Nov. 1983.

[22] P. Charbonnier, L. B. Feraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *Proceedings of 1st International Conference on Image Processing*, vol. 2, 1994, pp. 168–172.

[23] W. Lai, J. Huang, N. Ahuja, and M. Yang, "Fast and accurate image super-resolution with deep laplacian pyramid networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2599–2613, 2019.

[24] J. Lu, W. Hu, and Y. Sun, "A deep learning method for image super-resolution based on geometric similarity," *Signal Processing: Image Communication*, vol. 70, pp. 210–219, 2019.

[25] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, R. Meersman, Z. Tari, and D. C. Schmidt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 986–996.

[26] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 105–114.

[27] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Computer Vision – ECCV 2018 Workshops*. Cham: Springer International Publishing, 2019, pp. 63–79.

[28] A. Shocher, N. Cohen, and M. Irani, ""Zero-shot" super-resolution using deep internal learning," *Computer Research Repository (CoRR)*, vol. abs/1712.06087, 2017.

[29] S. B. Kligler, A. Shocher, and M. Irani, "Blind super-resolution kernel estimation using an internal-GAN," in *The Conference and Workshop on Neural Information Processing Systems (NeurIPS)*, Dec. 2019.

[30] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *Computer Research Repository (CoRR)*, vol. abs/1611.07004, 2016.

[31] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[32] X. J. Mao, C. Shen, and Y. B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, pp. 2810—2818.

[33] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4809–4817.

[34] Y. Tai, J. Yang, X. Liu, and C. Xu, "MemNet: A persistent memory network for image restoration," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4549–4557.

[35] R. Timofte et al., "NTIRE 2018 challenge on single image super-resolution: Methods and results," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 965–96 511.

[36] D. Mishra, S. Chaudhury, M. Sarkar, and A. S. Soin, "Ultrasound image enhancement using structure oriented adversarial network," *IEEE Signal Processing Letters*, vol. 25, no. 9, pp. 1349–1353, 2018.

[37] S. Vedula, O. Senouf, A. Bronstein, O. Michailovich, and M. Zibulevsky, "Towards ct-quality ultrasound imaging using deep learning," *ArXiv*, vol. abs/1710.06304, 2017.

[38] F. Dietrichson, E. Smistad, A. Ostvik, and L. Lovstakken, "Ultrasound speckle reduction using generative adversial networks," in *2018 IEEE International Ultrasonics Symposium (IUS)*, Dec. 2018, pp. 1–4.

[39] X. Zhang, J. Li, Q. He, H. Zhang, and J. Luo, "High-quality reconstruction of plane-wave imaging using generative adversarial network," in *2018 IEEE International Ultrasonics Symposium (IUS)*, Oct. 2018, pp. 1–4.

[40] W. Choi, M. Kim, J. HakLee, J. Kim, and J. BeomRa, "Deep CNN-based ultrasound super-resolution for high-speed high-resolution B-mode imaging," in *2018 IEEE International Ultrasonics Symposium (IUS)*, 2018, pp. 1–4.

[41] J. Liu, H. Liu, X. Zheng, and J. Han, "Exploring multi-scale deep encoder-decoder and PatchGAN for perceptual ultrasound image super-resolution," in *Neural Computing for Advanced Applications*. Singapore: Springer Singapore, 2020, pp. 47–59.

[42] H. Liu, J. Liu, T. Tao, S. Hou, and J. Han, "Perception consistency ultrasound image super-resolution via self-supervised CycleGAN," *Neural Computing and Applications*, Jan. 2021.

[43] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2242–2251.

[44] S. J. Park, H. Son, S. Cho, K. S. Hong, and S. Lee, "SRFeat: Single image super-resolution with feature discrimination," in *Computer Vision – ECCV 2018*. Cham: Springer International Publishing, 2018, pp. 455–471.

[45] O. Huang, W. Long, N. Bottenus, M. Lerendegui, G. E. Trahey, S. Farsiu, and M. L. Palmeri, "MimickNet, mimicking clinical image post- processing under black-box constraints," *IEEE Transactions on Medical Imaging*, vol. 39, no. 6, pp. 2277–2286, 2020.

[46] H. Temiz and H. S. Bilge, "Super resolution of B-mode ultrasound images with deep learning," *IEEE Access*, vol. 8, pp. 78 808–78 820, 2020.

[47] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640 –651, Apr. 2017.

[48] J. Lu and W. Liu, "Unsupervised super-resolution framework for medical ultrasound images using dilated convolutional neural networks," in *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, 2018, pp. 739–744.

[49] J. B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5197–5206.

[50] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, 2015, pp. 234–241.

[51] R. J. G. van Sloun, O. Solomon, M. Bruce, Z. Z. Khaing, Y. C. Eldar, and M. Mischi, "Deep learning for super-resolution vascular ultrasound imaging," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 1055–1059.

[52] J.H. Park, W. Choi, G. Y Yoon, and S. J Lee, "Deep learning-based super-resolution ultrasound speckle tracking velocimetry," *Ultrasound in Medicine & Biology*, vol. 46, no. 3, pp. 598–609, 2020.

[53] S. Baek and S. Lee, "A new two-frame particle tracking algorithm using match probability," *Experiments in Fluids*, vol. 22, pp. 23–32, Nov. 1996.

[54] H. Temiz and H. S. Bilge, "Super resolution of B-mode ultrasound images with deep learning," *IEEE Access*, vol. 8, pp. 78 808–78 820, 2020.

[55] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[56] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *4th International Conference on Learning Representations, ICLR*, May 2016.

[57] Y. Lecun, "A theoretical framework for back-propagation," in *Proceedings of the 1988 Connectionist Models Summer School*. Morgan Kaufmann, 1988, pp. 21–28.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[59] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[60] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 977–984.

[61] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[62] S. Fadnavis, "Image interpolation techniques in digital image processing: An overview," *Int. Journal of Engineering Research and Applications*, vol. 4, no. 1, pp. 70–73, Oct. 2014.

[63] J. Weickert, *Anisotropic Diffusion in Image Processing*. Stuttgart, Germany: B. G. Teubner, 1998.

[64] The University of Auckland, New Zealand, "Gaussian filtering," https://arxiv.org/abs/2107.03055, accessed: 2021-08-26.

[65] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, pp. 1097–1105.

[66] D. L. K. Yamins and J. J. DiCarlo, "Using goal-driven deep learning models to understand sensory cortex," *Nature Neuroscience*, vol. 19, no. 3, pp. 356–365, March 2016.

[67] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Transactions on Information Theory*, vol. 36, no. 5, pp. 961–1005, 1990.

[68] W. Hu, L. Xiao, and J. Pennington, "Provable benefit of orthogonal initialization in optimizing deep linear networks," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[69] A. Lugmayr et al., "NTIRE 2020 challenge on real-world image super-resolution: Methods and results," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 2058–2076.

[70] M. Fritsche, S. Gu, and R. Timofte, "Frequency separation for real-world super-resolution," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 3599–3608.

[71] A. Liu, Y. Liu, J. Gu, Y. Qiao, and C. Dong, "Blind image super-resolution: A survey and beyond," https://arxiv.org/abs/2107.03055, Jul 2021, accessed: 2021-07-25.