Chair of Digital Image Processing

Deep Learning for 3D Super-Resolution

Master Thesis

Author: Omar Hussein

1st Supervisor: Prof. Dr. Tomas Sauer

2nd Supervisor: Prof. Dr. Michael Granitzer

Statutory Declaration

This is to certify that :

1. The thesis comprises only my original work towards the Master's Degree.

2. Due acknowledgement has been made in the text to all other material used.

Omar Hussein

Passau, July 12, 2021

Acknowledgments

I would like to thank my wonderful family and friends for supporting and encouraging me while working on my master thesis. I would like to also thank Prof. Dr. Tomas Sauer for offering me the chance to work on such an interesting project and for providing useful feedback and guidance whenever it was required. My appreciation also goes to Thomas Lang and Ruben Fischer for their support and interesting discussions. A great deal of thanks to my friends Hazem Ghoraba and Marwa Hamrouni for their helpful opinion, encouragement and support.

In addition, I would like to thank the Chair of Digital Image Processing for providing the computing resources that were necessary for my work and the MGH-USC Human Connectome Project for providing access to their valuable data that was extremely useful for the project.

Abstract

3D voxel images created by CT and MRI scans are extremely useful in the medical and archaeological fields. However, various limitations of both technical and financial nature make it challenging to obtain images with high resolutions which encourages using super-resolution techniques to obtain high resolution (HR) images from their low resolution (LR) counterparts. Deep learning has been demonstrated to provide state of the art performance on the task of single image super-resolution by using convolutional neural networks to learn the mapping between the LR and HR domains. 3D networks have been shown to provide better performance on 3D image scans compared with 2D networks, however, they are harder to train due to their increased complexity and size. In this thesis, we propose a novel architecture for a 3D network (3DRDN) that utilizes both Residual and DenseNet connections to minimize the required size and complexity of the network for the task of super-resolution on 3D images. Furthermore, we integrate the network in two additional generative adversarial networks (Wasserstein GAN and Cycle-GAN) to obtain perceptually better upscaled images. To evaluate the performance of our models, we perform experiments using two 3D image datasets which demonstrate the excellent capabilities of our models on the task of 3D super-resolution.

Keywords: Super-Resolution, Deep Learning, 3D Convolutional Neural Network, Wasserstein GAN, Cycle-GAN, CT, MRI

Image Processing Terminology

Expression	Explanation
Resolution	The size (no. of pixels/voxels) of a 2D or 3D image
Upscaling	Increasing the resolution of an image
Downscaling	Decreasing the resolution of an image
HR Image	An image with original resolution that is considered ground truth
LR Image	An image with a smaller resolution than its original counterpart
Scale Factor	Resolution ratio of a HR to its LR counterpart
Image Fidelity	How similar a processed image is to the ground truth
Super-Resolution	Upscaling a LR image while maintaining its sharpness and fidelity

Contents

1 Introduction

1.1 Motivation

Magnetic Resonance Imaging (MRI) and Computed tomography (CT) scans are imaging techniques that utilize electromagnetic waves to create a visual representation of a scanned object and which are used in various fields such as the medical and archaeological ones.

They operate by having a device emit a series of radiation beams at the object from various directions and accordingly obtain multiple scans from different angles. This results in multiple 2D cross-section images of the object which when stacked on top of each other provide us with a 3D representation that is voxel based. The main difference between the two scanning techniques is the waves utilized with CT scans using X-rays and MRIs relying on radio waves and magnets fields. While they both create 3D images, MRI scans generally produce more detailed representations while CT scans on the other hand provide results faster and with less costs [1].

In the medical field, the scans are utilized as non-invasive techniques to obtain images of various human body parts such as brain, soft tissues and bones. They are very useful in diagnosing various medical ailments such as strokes, traumas and bone fracture. Another

important utility for them is in diagnosing cancer and identifying the location of tumors [1]. Figure 1.1 illustrates an MRI device and examples of MRI image output.

Figure 1.1: On the left [2] is an example of an MRI device used for scanning patients. The two images [3], [4] on the right are samples of MRI scanned images.

Their non-invasiveness makes them also equally useful in the archaeological field as various ancient historical objects, that need to be intricately examined, are easily susceptible to damage from even a simple touch in some extreme cases. The most scanned archaeological objects are usually mummies but various other artifacts such as clay tablets and statues are also examined. One of the most famous artifacts examined is an ancient mummy found in the Austrian alps called Otzi (The Ice Man). Thanks to CT scanning, the cause of death of Otzi was identified to be an arrowhead lodged in his shoulder [5].

Despite their great capabilities, there are some inherent limitations when using such scanning systems. First of all, the higher a resolution for an examined object is needed, the more scans need to be performed to get more accurate and detailed cross section images. This is however undesirable for medical purposes, since the scan entails releasing targeted doses of radiation at a patient. Although the doses are almost always small

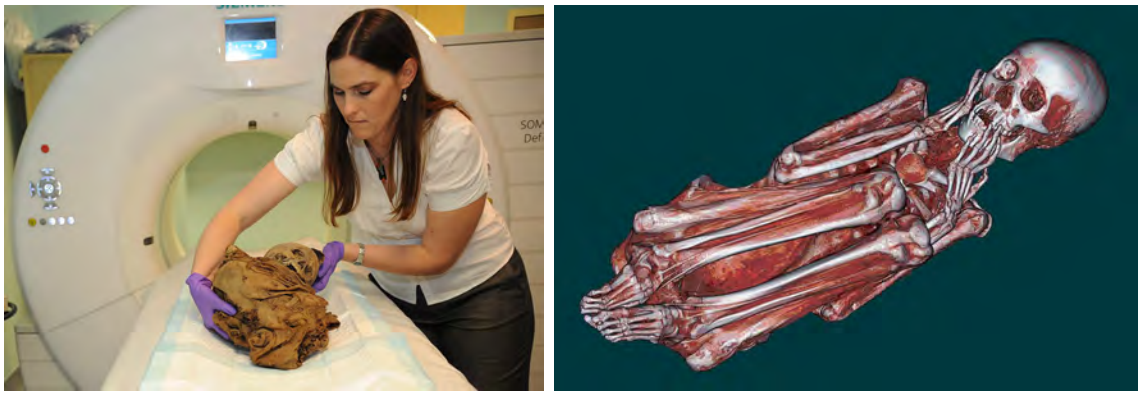


Figure 1.2: The left image [6] shows a Peruvian child mummy in preparation for being CT scanned and the right image [7] shows the 3D model result.

and unharmful, they still have to be minimized as a safety procedure even if it means sacrificing the resolution of the scanned images [8]. Furthermore, even in cases where the medical safety of the examined object is not a concern such as with archaeological artifacts, the cost of running the scanning device has to be taken into consideration and evidently a higher resolution scan will require higher costs especially for MRIs. Moreover, increasing the resolution in the Z-axis or Axial/Transverese plane, increases the noise of the resultant scan. Lastly, setting a high resolution scan will sacrifice the field of view of the scanning resulting in a smaller part being examined [9].

Due to the aforementioned reasons, in many cases only low resolution (LR) scans are available. Such LR scans are unfortunately not accurate enough in some use cases such as when doctors are trying to segment tumors to plan treatment or when archaeologists try to identify intricate details of fragile historical artifacts. Fortunately, the concept of obtaining high resolution (HR) images/scans from LR ones has been the target of various research and is one of the classical computer vision and image processing problems [10].

While there are various methods for performing super-resolution, the focus in our work will be on single image super resolution (SISR) where the upscaling is done on a single LR input as opposed to other super resolution methods that utilize multiple LR input

images to obtain a single HR output image.

Super-resolution problems are challenging because they are inherently ill-posed due to the the existence of multiple possible solutions as each HR image could be downgraded to multiple LR images. Prediction-based techniques that rely on using filters such as Bicubic interpolation or Lanczos resampling were among the first methods utilized to handle the problem. However while they are fast, they tend to produce overly smoothed results without sharp edges or details [11].

Sparse coding was considered to be the state of the art in terms of achieving single image super-resolution [10]. Its main focus is the extraction of both low and high resolution patches, learning the mapping between them and optimally aggregating them back into the new high/low resolution image form.

The authors of [10] have shown that a deep convolutional neural network is equivalent to the pipeline process of sparse coding. Their work has motivated countless others to utilize deep learning for performing super resolution. As a result, deep neural networks, which have gained popularity thanks to the increased computational capabilities offered by modern computing systems as well as increased available data for training, are now one of the modern techniques used for performing super resolution on images and have been the focus of various research to increase the quality of the upscaling process.

1.2 Contribution

The main focus of this thesis is the utilization of deep learning to perform super-resolution on 3D voxel models which are the typical representations used for CT and

MRI scans. To achieve our aim, we propose a novel network architecture which incorporates multiple state of the art deep learning methodology that is most beneficial for the task of super-resolution.

To begin with, unlike most research where a 2D architecture is utilized for 3D super resolution, our neural network model is 3D based which enhances the performance by extracting more information for learning the LR to HR mapping. Furthermore, our model incorporates both Residual and DenseNet connections which significantly eases gradient flow during backpropagation and tailors the model for super-resolution thanks to the input to output residual connection. Moreover, we integrate the model in a Wasserstein GAN that enhances its upscaling by incorporating perception-based adversarial loss to its objective loss function to improve the visual quality of its output. Lastly, our model is also integrated in a Cycle GAN architecture which helps in stabilizing the GAN training process and producing better visual results.

Our work is the only one, to the best of our knowledge, that integrates a 3D neural network in a Cycle GAN architecture for the task of super-resolution.

Another investigation, that we found interesting and useful to pursue and which is not solely related to the super resolution field but to the general image to image translation field, is the comparison of the performance of a model when trained in an unsupervised way with unpaired data (using Cycle GAN) versus when its trained in a supervised way using paired data. Our results demonstrate that supervised learning produces better results for the task of super-resolution.

In this thesis, we propose a novel architecture: **3D R**esidual and **D**enseNet **N**etwork (3DRDN) which we utilized in 3 architectures:

- **3DRDN**: In which we train it as a single convolutional neural network with Residual and DenseNet connections in a supervised way with the $L1$ loss.
- **3DRDN-WGAN**: In which we utilize it as a generator in a Wasserstein GAN (with gradient penalty) architecture which in addition to the $L1$ loss is trained with a perception based adversarial loss using the Wasserstein 1 distance.
- **3DRDN-CGAN**: In which we utilize 2 3DRDNs as 2 generators in a Cycle-GAN architecture and which both in addition to the $L1$ and adversarial loss are trained with cycle consistency and identity losses.

The following points summarize our contributions:

1. We propose a new novel architecture 3DRDN for performing super resolution on 3D images from CT and MRI scans. The model is concise and efficient without sacrificing the upscaling quality that it achieves.
2. We integrate the aforementioned 3DRDN with 2 different GAN architectures: WGAN-GP that adds the adversarial loss for human perception consideration and Cycle-GAN that adds a cycle consistency loss for better performance and stability during training.
3. We propose the concept of combining both Residual and DenseNet connections. The residual connection between simplifies the model's learning and the dense-net connections allow the reuse of features between layers further enhancing training and decreasing the model's size. Both connections also mitigate gradient flow problems leading to easier and faster training.
4. We utilize our network architectures and the SR task to determine whether its better to train a GAN model in a supervised or unsupervised fashion by comparing

the performance of our unsupervised Cycle-GAN version with other supervised versions of the model on the task of super resolution.

To demonstrate the capabilities of our models, we perform experiments on two different datasets (one made up of MRI scans and the other CT scans) and compare their performance with that of traditional interpolation methods. The results indicate that our models are superior and that they can be used to upscale both MRI and CT scans.

1.3 Outline

In chapter 2, we provide our literature review on the subject of single image super resolution with focus on research that inspired and guided our own work. We describe in intricate details the super-resolution problem, our models, architectures, loss functions and other important methodologies in chapter 3. Chapter 4 contains a description of the datasets that we used in our training and a description of the experiments that we performed as well as the preprocessing techniques that were used to obtain training data and the evaluation metrics that were utilized. In chapter 5, we present both the quantitative and visual results of our experiments and we provide our analysis on the results in chapter 6. Lastly, we provide our conclusions and future work recommendations in chapter 7.

2 Literature Review

In this chapter, we present our literature review where we discuss similar research work that is related to the field of super-resolution.

The authors of [12] produced a study that compares the most popular image quality assessment techniques; a subject that is of extreme importance for the super resolution field as it is through such assessments that we could evaluate the performance of any model that achieves upscaling. Initially, most image processing research utilized Mean Squared Error (MSE) or Mean Absolute Error (MAE) in addition to Peak Signal to Noise Ratio (PSNR) which all focused on pixel to pixel comparisons. However the popular work in [13] encouraged more research to rely on the structural similarity index measure (SSIM) that better evaluates human related perception loss. In addition, [14] proposed a technique that relies on information fidelity criterion (IFC) which is based on natural scene statistics. As of the time of writing this, academia is still undecided on whether it is error methods such as MAE/MSE/PSNR or human perception based methods such as SSIM that perform better image assessment. Most super resolution research utilizes both PSNR and SSIM in measuring the accuracy of any HR image obtained through super resolution.

Before the onset of deep learning as a popular method for single image super-resolution, most algorithms that were used focused on obtaining strong image prior information to

help in upscaling. The algorithms were categorized into 4 groups: prediction models, edge based methods, image statistical methods and patch (example) based methods with the latter achieving the greatest performance [15].

Example-based techniques produced sharper and more detailed results and one of its techniques sparse coding was considered to be the state of the art for super-resolution. Its process begins by extracting overlapping patches of images from the input low resolution image; they are then pre-processed and encoded using a low resolution dictionary. The low resolution dictionary is mapped into an equivalent high resolution dictionary which returns the equivalent high resolution of each patch which are then aggregated to construct the final high resolution image. The most important and challenging part is optimizing both the low and high resolution dictionaries that, through efficient mapping, allow up- or down-scaling the patches that make up an image.

2.1 2D Super Resolution

The work of [10] was significantly responsible for popularizing deep learning as a means of achieving super resolution. Prior to that, deep learning was mostly used for denoising as part of the post-processing of the image. However their work aimed at showing that deep learning can be used for the entire super resolution process.

They demonstrated that a Convolutional Neural Network (CNN) architecture constitutes three parts that are equivalent to the three operations performed in sparse coding which are patch extraction and representation, non-linear mapping and reconstruction. Their proposed model, aptly named Super Resolution Convolutional Neural Network (SRCNN), had a simple architecture of a not too deep CNN with 5 hidden layers as

properly training deeper networks was still extremely challenging around the time of their paper release.

Nevertheless, even that limited layered model was able to surpass all the state of the art super-resolution methods at the time. Moreover, since the model doesn't need any optimization after finishing training and is fully feed-forward, it can perform the super-resolution faster than the concurrent state of the art methods making it not only better but also more practical to use. Lastly, it could count on using more data to easily improve its performance which is an upstep from the normal sparse coding process that gets more complicated the more input data is used. Critical drawbacks of the model included is its incapability to perform different upscaling factors and requiring retraining from the beginning to achieve new super resolution factors.

The authors of [16] built their work on the foundation provided by the previous authors and managed to produce a new model architecture Fast Super Resolution by CNN (FSRCNN) that significantly improved on SRCNN and avoided many of its shortcomings. The model takes the low resolution image input as it is without relying on doing bicubic interpolation, to get the low resolution image to the desired size first, as a form of pre-processing which is the case with SRCNN. Instead, a transpose convolutional layer is appended at the end of the model to replace the bicubic interpolation which has two positive implications. One is that the computational complexity of upscaling and its effect on consecutive computations is delayed until the very last layer making the model overall less computationally complex. The other tremendously positive implication is that now only the last layer has to worry about upscaling meaning we need only fine tune it through retraining while freezing the other layers so that we can utilize the model for performing different upscaling factors as opposed to retraining the entire layers for SRCNN.

The model of [17] had 20 layers and was among the first ones to implement residual learning for the purpose of super resolution. Despite having a much deeper network than SRCNN, VDSR was able to achieve faster convergence thanks to residual learning as well as the utilization of a higher initial learning rate with gradient clipping during training. The authors managed to make the model capable of performing multi-scaling by using training datasets containing different scales which forced the model to learn super resolution with all of the different scaling factors represented in the datatset. Their results indicate that a single model can perform multi-scaling without even needing upscaling layers or fine-tuning such as the case with FSRCNN.

[11] was among first papers to utilize a Generative Adversarial Network (GAN) for super resolution. Their main aim was to differentiate their work by shifting away from the tradition of using mean squared reconstruction error as the objective function of the training. In addition to adversarial loss, they also utilized a perception based content loss for the generator's output. The content loss is determined using another network through which feature representations for the generator's output are obtained and compared with those of the expected output instead of comparing their equivalent pixels as in all previous work. The SRGAN model was the first of its kind capable of properly upscaling natural images with an upscaling factor of 4 and it had higher accuracy when compared with its contemporaries.

2.2 Super Resolution on 3D Voxel Images

[18] is one of the most recent publications in the field of super resolution for CT scans with deep learning. The authors proposed model named GAN-CIRCLE utilized a Cycle-GAN architecture for upscaling CT scans. In addition, to the Cycle-GAN loss components they

also added the joint sparsifying transform loss function which helped in maintaining the image sparsity while also minimizing noise or artifacts. However despite working with 3D images, their architecture was 2D based and their model operated individually on every slice from the CT scan. As a result, it ignored information from the axis perpendicular to the image slices.

In [19], the authors aimed at designing an architecture that uses deep learning with the main focus of performing 3D super resolution rather than just use deep learning to achieve 2D super resolution on slices of a 3D model before aggregating them back together. However, they applied upscaling on only two axes with the third one assumed to be already upscaled. The other significant aspect of the architecture is the utilization of dense-block layers. Another interesting change made was with the pre-processing done to obtain LR training data from the original HR data. Almost all of the related research obtained the LR data by applying Gaussian blurring on the HR data followed by shrinking the resulting output while the authors decided on instead applying Fast Fourier Transform (FFT) on the HR data followed by degrading it in the frequency domain before reverting back again using inverse FFT which produces LR data that more closely resembles real life LR data obtained through CT or MRI scans. The model developed, named Densely Connected Super Resolution Network (DCSRN), was able to outperform the FSRCNN and also took less than half the needed time to perform super resolution.

The work in [9] was also among the first to extend the usage of super-resolution with deep learning from 2D images to the 3D realm. The model proposed is called 3DSRCNN and is a neural network made up of 12 layers with a residual connection from the input to the last layer. Training such a deep network is quite difficult due to the increased likelihood of gradient explosion as well as slow convergence. In addition, 3D data is

significantly more challenging to work with as it requires more computational intensity and also takes up significantly more memory which could slow the training process in best case scenario or crash it in the worst. The residual connection of the model helps mitigate the explosive gradient problem and slow convergence and on top of that the authors also used an adjustable learning rate, gradient clipping and momentum stochastic gradient descent (SGD) to further optimize the training process. To handle the memory problem, the authors split the training data into smaller blocks to help facilitate the training process. However, their model relied only on MSE in its objective loss function and did not include any perception based loss components.

As can be seen from our literature view, as of this writing, we have not found any research work which designed a model for super-resolution that operates purely on 3D images using 3D convolutions while at the same time integrating a perception based loss (such as GAN adversarial loss) into its objective function. We designed our models to fit such requirements and determine how they affect the performance of the models on the task of super-resolution.

3 Methods

In this chapter, we first provide a theoretical explanation of the super resolution problem in section 3.1. In section 3.2, we provide an overview of convolutional layers. Sections 3.3 and 3.4 contain detailed descriptions of important design choices we made for our models to optimize them for the SISR task. In section 3.5, we explain various state of the art GAN architectures which we integrated with our models. Section 3.6 contains a description of two skip connection types that were integral in the design of our neural networks while in section 3.7, we showcase the architecture of all networks utilized in our models. Finally we provide a detailed description of the objective loss functions of our models in section 3.8.

3.1 Super Resolution Problem Statement

Given a LR image X, the task of super resolution aims to find the equivalent HR image Y that relates to X through the following equation:

$$X = f(Y) \tag{3.1}$$

where f is a continuous function that represents the downgrading of the image quality that results from CT/MRI scanning for only a relatively brief period of time.

Accordingly, the super resolution process requires finding g which is the inverse function of f and using it to obtain HR images:

$$Y = g(X) = f^{-1}(X) \tag{3.2}$$

Since no true inverse function g exists, due to the fact that multiple LR images could be obtained from the same HR source and vice versa, super resolution is considered an ill-posed inverse problem.

Other factors that further compound the problem of super resolution on LR CT/MRI scans, include the fact that such medical scans usually have more complex spatial variation and statistical properties than natural images and the fact that the scanning noise is introduced to data during the reconstruction process which results in unique noise as well as artifact patterns [18]. This all makes the task more challenging and limits the upscaling achieved using traditional super resolution methods.

Despite the presence of such challenges, modern super resolution methods can overcome them thanks to the fact that both X and Y share low dimensional information. In accordance with the sparse coding method which produces best results, a super resolution system has to extract patches from X and utilize a feature mapping to obtain equivalent patches that are used to reconstruct Y.

[10] has shown that CNNs naturally embed the sparse coding process and proven that deep learning is ideal for the super resolution task since the 3 sparse coding processes feature extraction, mapping and reconstruction are trained simultaneously. On top of that,

once trained, deep learning models can perform the computations for performing upscaling significantly faster than a traditional sparse coding process. Thanks to advancement in deep learning research, CNN based models achieve state of the art performance in the task of super resolution.

In our work, we utilize an advanced deep learning CNN architecture to perform single image super resolution. The network is composed of multiple non-linear concatenated blocks that learn high frequency details and which are combined with a residual module to produce the final output.

3.2 Convolutional Layers

Convolutional Neural Networks (CNNs) are one of the most powerful and widely utilized network architectures in the computer vision field and convolutional layers are the fundamental building blocks that make up their architectures.

In traditional fully connected networks (FCNs) each node in a layer is connected to all of the nodes in both the previous and next layers. Combining the nodes with activation functions that introduce non-linearity (such as RELU) is what makes the entire network capable of approximating complex non-linear functions through training. Such a network will utilize all of the features from the input and learn to identify patterns in them which is greatly useful in various machine learning tasks such as classification and regression.

This approach however, is extremely difficult to utilize on large images as the input layer will need to have nodes for each pixel of the image and it is non-feasible to build a neural network with such a large input layer.

The CNN solution is to utilize kernel convolutions where a small matrix of numbers (kernel or filter) is passed over the entire image and used to transform it based on the values inside the filter. Such filters can therefore be used to detect local patterns in the image rather search the entirety of the image for patterns. Key to the success of convolutional layers is this translation invariance property as it significantly reduces the number of required parameters in a network since a collection of trained filters can be used to determine if certain patterns appear at any location in the image rather than only be trained to identify the patterns at specific locations.

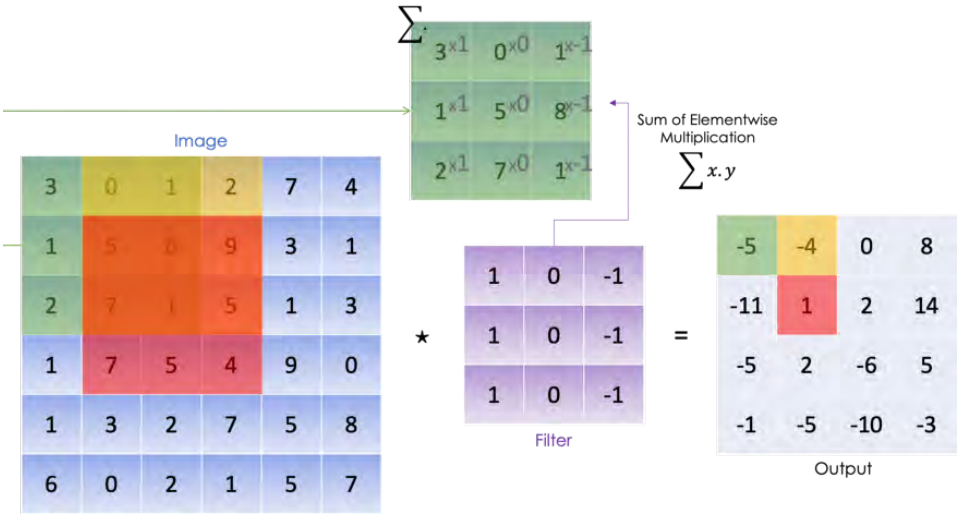


Figure 3.1: An illustration [20] that demonstrates how a filter creates an output feature map from its input.

The following equation demonstrates how the filter interacts with input (image or feature maps from the previous layer) to create the feature maps of the next layer:

$$O[m,n] = (I * F)[m,n] = \sum_{j=0}^{J} \sum_{k=0}^{K} F[j,k] * I[fj, fk] \tag{3.3}$$

where O, F and I represent the output feature map, filter (or kernel) and input image (or input feature map) respectively. m and n are the dimensions of the output feature map

while J and K are the dimensions of the filter. fj and fk are the values from the input which coincide with the filter as it passes over the input. Essentially the filter passes over the entire input and the dot products of the filter's values with their equivalent ones from the input are added to form a value that is part of the output feature map which is also a matrix. Figure 3.1 shows an example of the convolutional computing process.

3.3 3D versus 2D models

Plenty of research on the task of CT/MRI super resolution actually utilizes 2D models for 3D super resolution by slicing the 3D voxel image into multiple 2D ones which are then processed individually before being combined back into a 3D image output such as the case with [18].

Such 2D models however ignore structural information in the axis perpendicular to the slices as a source of prior which limits their capabilities. Furthermore, some information will be harder to acquire from only a 2D image and might be completely lost; for instance if a blood vessel runs perpendicular to the 2D image slices, it would appear as a small dot in all of them and the model would have to decide when processing each 2D slice whether the dot is noise that should be eliminated or whether it is part of the image that has to be upscaled.

Utilizing a fully 3D model with 3D convolutions would mitigate that problem as the entirety of the image is used at the same time as a source of information when processing it. [19] and [21] have performed experiments where a 3D model was compared to a 2D one using the same training conditions and their results clearly indicate the superiority of the 3D models.

Using 3D models however, brings up a new set of challenges especially with regards to training. First of all, 3D data takes up much more space in memory than 2D data which puts a limit on the batch size used during training. Moreover, a 3D model would have significantly more parameters than a 2D one as a result of having 3 dimensional filters for its convolutional layers. Besides resulting in more computational as well as time costs, this also means that more data is needed to properly train the 3D model to an acceptable performance when compared to a 2D one. Furthermore, having a huge number of parameters could force the model to overfit the training data and fail to generalize well when handling unseen data.

In our work we utilize 3D models and to handle the memory issue we use, as training data, 3D patches from the CT/MRI scans rather than utilize the entire voxel image. The data is split into uniform blocks of voxels that consume significantly smaller space in the memory. Another nice additional benefit gained is that we also end up with substantially more data for training the model. This is especially so when we also combine this with data augmentation methods for the 3D patches to help minimize overfitting. To minimize the number of model parameters we utilize multiple levels of DenseNet connections in our model which force it to efficiently utilize learned features to the utmost and require significantly smaller number of filters for each convolutional layer.

Some research work such as [22] performs 3D super resolution but eases the task by doing the upscaling for only two of the three axes with the third axis assumed to have the correct HR resolution. Our work however, does the upscaling on 3D models for the entire three axes.

3.4 Super Resolution Implementation Design

In this section, we discuss two possible high level design choices for a network architecture to achieve super resolution and explain the reasons for our choice of one of them. Naturally a LR image will have lower dimensions than its equivalent HR version and there are multiple ways to deal with that when performing upscaling using deep learning.

The first design choice would be to utilize the LR image with its default small dimensions as input to the deep neural network. Such a network would have to, therefore, incorporate interpolation during learning to increase the dimensions of the output image. Fortunately transpose convolutional layers (sometimes incorrectly termed deconvolutional layers) are capable of doing just that and can be considered to be the 'upscaling' layers in a neural network.

The advantages of using such a system is that all computations before the upscaling layers will be done in the LR domain with its smaller dimensions which will be significantly faster. The time and computational costs could be even further minimized by utilizing valid padding which will further shrink the resultant feature maps of each convolutional layers. Ideally the upscaling layers would be placed near the end to minimize the computations that are done in the HR domain with its bigger upscaled dimensions. [11] and [16] are examples of research work that utilized this concept in the design of their super-resolution models.

There are however, downsides in using such an architecture to perform super resolution. First of all, the trained network will only be capable of performing upscaling with a specific scale factor which is the ratio of the network output dimensions to its input dimensions. Furthermore, interpolation will have to be learned which will increase the

training time and result in the model having more parameters which increases the likelihood of the model overfitting if not handled appropriately such as by using more training data .

On the other hand, another design approach would be to introduce interpolation (such as bicubic) into the pre-processing stage of the LR input data. The interpolation will make the LR and HR images have the same dimensions although naturally the LR images will have less clarity and sharpness since its information will be spread over more pixels (or voxels in the 3D case). [22] and [9] are examples of research work that utilized this architectural design.

Although the network will have to operate from the beginning with the larger dimensions of the HR domain, it will be capable of performing different scaling factors for the super resolution process as any LR input could first be upscaled from any dimension using interpolation before being used as input to the model. Not only could the same architecture be used for different upscaling factors, but as the research of [16] has shown, even the same model can be used to perform super resolution with different scaling factors if it's training data had a diverse distribution of different scaling ratios between the LR and HR images.

In our architecture, we decided to go with the second design approach as it is very beneficial to have a model capable of performing upscaling with different scaling factors when dealing with data from CT/MRI scans that might not necessarily have uniform LR dimensions.

More importantly is the fact that the second design also simplifies the task of integrating the Cycle-GAN architecture with our models which requires four networks: two (generator plus discriminator) for upscaling and two others for downscaling. With the first

design, the two generators will have to have different architectures as one will have to have upscaling layers while the other will have to have downscaling layers. The two discriminators will also have input layers with different sizes which makes comparing their output not as reliable. With the second design however, both of the two generators and two discriminators can share the same architecture which simplifies designing and training them and also minimizes the hyperparameters of the entire architecture.

In addition, since our data and models are 3D, we already have plenty of incentive to split the input into patches before using them for training which means that our model does not operate in the full HR domain with its larger dimensions but with the smaller dimensions of the split patches. Therefore, the major disadvantage of using the second design approach in which time and computational costs are increased is somewhat mitigated. Since we are dealing with input that has patch dimensions rather than HR dimensions, we also do not employ valid padding as this could cause too much loss of information during forward propagation and therefore we employ instead same padding to maintain the same dimensions of the patches from the input till the output layers.

3.5 Generative Adversarial Networks

As previously mentioned in section 3.1, performing super resolution using deep learning requires training a network to approximate a function g that achieves upscaling on its input. A core concept of deep learning is utilizing an objective loss function to guide the training of the network until it is sufficiently capable of performing the desired task.

The simplest and most common loss function for guiding a computer vision task such as super resolution would be either the Mean Squared Error (MSE) or the Mean Absolute Error (MAE) and this was the case with most of the initial SISR research. Networks that

are trained using such objective functions produce results that have high peak signal to noise ratios (PSNR) which is a widely used metric for evaluation in computer vision tasks. However, these results are sometimes unfortunately not very satisfying from a human perception point of view.

The problem arises from the fact that PSNR and by extension MAE and MSE are incapable of capturing high texture perceptual differences between images since they only operate on a pixel to pixel or in our case voxel to voxel basis. Therefore a network might achieve a very small error and yet produce an upscaled image that is not very sharp or lacking in some details.

The reason is due to the ill-posed nature of the super resolution problem which we discussed in section 3.1; namely that a single LR image might be the downscaled result of multiple HR images. This under-constrained nature of the problem with its multiple solutions leads the MAE or MSE loss functions to simply averaging over all the possible HR solutions and thus obtaining a result that is overly smoothed and lacking in high texture information details as pointed out in the work of [11].

It is thus imperative to integrate into the objective loss function a metric that can better represent the perceptual quality of the images in order to be obtain more appealing results from a human perception point of view. One such loss component could be obtained by utilizing Generative Adversarial Networks (GANs).

3.5.1 Standard GAN

GANs are one of the most popular generative neural network models that are used to achieve state of the art performance in many deep learning applications such as computer vision. The main premise is that 2 networks, a generator and a discriminator, are trained

simultaneously in an adversarial manner. The generator tries to produce realistic output such as well upscaled images in our case while the discriminator tries to identify whether its input was obtained from the generator or whether it was genuine true (ground truth) data. This min-max game ideally forces the generator to produce data (images) that are visually similar to its training data which will allow it to fool the discriminator.

This training process when handled properly will in ideal situations result in a network (the generator) capable of producing images that are more natural looking and with high perceptual quality that is often lacking when just utilizing MAE or MSE for training. Essentially rather than average over all of the possible HR image solutions, it will force the network to produce output that is as close as possible to the true HR image in the image search space thanks to the adversarial training.

The following equation demonstrates the min-max game that the generator G and discriminator D play to enhance both of their respective capabilities:

$$\min_{G} \max_{D} E[\log(D(I^{HR}))] + E[\log(1 - D(G(I^{LR})))] \tag{3.4}$$

where I^{HR} and I^{LR} are respectively high resolution and low resolution images from a training set and E is the expected value over all data instances. $D(I^{HR})$ represents the discriminator's probability estimation that real data input is real and $D(G(I^{LR}))$ represents its probability estimation that the output of the generator data is real.

By maximizing the two components in the equation, the discriminator tries to enhance its performance in correctly identifying HR images from images produced by the generator. The generator on the other hand improves its performance by increasing the probability that the discriminator will think that its output was a true HR image which, if the discriminator has decent capabilities, means that its images are perceptually getting

better.

Training such a model is challenging however, as there are certain states that could be reached by the networks during training that lead the generator to producing undesirable output. One of the most common problems is mode collapse where the generator learns a specific way that is always capable of fooling the discriminator without having to produce a realistic or visually appealing output as intended. Another common problem is that the discriminator converges significantly faster than the generator without giving it a chance to learn how to deceive it which leads to the generator not learning anything. The reason is that a perfect discriminator will make the loss function produce zero output which makes it impossible to train the generator as there will be no loss gradient to update its weights. Therefore, a careful balance has to be maintained while training both of them so that they both get a chance at improving their capabilities.

One interesting way to look at the entire GAN training process, is that images whether they are of low or high resolutions come from a certain probability distribution and that the discriminator is learning how to differentiate between them while the generator is learning how to produce images that are from a probability distribution that is as close as possible to that of the ground truth or real HR images.

3.5.2 Comparing probability distributions

One of the classical methods to encourage a model to produce data with a desired probability distribution is by using the maximum likelihood function demonstrated in the following equation:

$$\max_{\theta \in R^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)}) \tag{3.5}$$

where P_θ is a parametric family of probability densities from which we find the parameters (modified by the model) that maximize its likelihood on data $x^{(i)}$ which comes from the desired probability distribution. This is equivalent to minimizing the Kullback-Leibler divergence $KL(P_r||P_m)$ between the probability distributions of real data P_r and the model output P_m.

The following equation demonstrates the KL divergence:

$$KL(P||Q) = \int_x P(x) \log \frac{P(x)}{Q(x)} dx \tag{3.6}$$

The KL divergence as a metric suffers from various problems that limit its capability when comparing any probability distributions. First of all, as a divergence measurement it is not symmetric i.e. $KL(P_r||P_g) \neq KL(P_g||P_r)$. Moreover, a probability density might not even exist which is the case with distributions with low dimensional manifold making training impossible. Furthermore, even if we found a good probability density, it might be computationally expensive to sample from if it is highly dimensional.

More importantly however, as can be seen from the KL equation, if there is any x where $P(x) > 0$ and $Q(x) = 0$ then the entire KL divergence will reach a value of infinity. This is the case at points x where there is no overlap between the distribution P and Q. This is a significantly huge problem since it is not uncommon to deal with probability distributions that have low dimensional manifold which means that most likely the two distributions will have negligible overlap or in other words there will be plenty of x values where $Q(x) = 0$. At such values the KL divergence will be infinite. Variational Auto-Encoders utilize the KL divergence and as a result suffer from its limitations.

A better metric than KL for comparing probability distributions is the Jensen-Shannon JS divergence:

$$JS(P,Q) = \frac{1}{2}KL(P||P_m) + \frac{1}{2}KL(Q||P_m) \tag{3.7}$$

where P_m represents a mixture distribution made from a combination of P and Q.

Unlike KL, the JS divergence is symmetric and additionally it does not explode to infinity if there is no overlap between the compared probability distributions but will instead produce a diversion with the value $\log 2$.

It however, still suffers from some problems such as the fact that its divergence measurements are not continuous and more importantly, although it produces a divergence of $\log 2$ rather than infinity when there is no overlap, it is still not a useful value for training any model as it is not differentiable meaning the gradient used for training any model using the backpropagation method will be zero.

The original GAN which utilizes the JS divergence can overcome the divergence's limitations by introducing random noise into its output probability distribution which will make the distribution defined everywhere and ensure that there will always be overlap between it and the real data distribution. Such noise however, is completely undesirable as it degrades the quality of the generated output and makes any resultant images from the generator blurry.

3.5.3 Wasserstein GAN with gradient penalty

The research work of [23] aimed specifically at handling all of the aforementioned problems when using either KL or JS for measuring the divergence between two probability distributions.

Fundamentally rather than training the generator to learn a probability density function from which a sample is obtained for divergence measurement, they proposed instead training the generator to learn a function that transforms an existing probability distribution (either uniform or normal) into the desired real data probability distribution.

Another important key point is the method they proposed for comparing the probability distributions which is the Earth Mover (EM) or Wasserstein distance:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y)\sim\gamma}[||x - y||] \tag{3.8}$$

where $\Pi(P_r, P_g)$ represents the set of all joint distributions $\gamma(x, y)$ with real data distribution P_r and generator output distribution P_g as marginals.

The easiest way to conceptualize the EM distance is to think of the first probability distribution as a collection of specific masses at certain points/locations with the target probability distribution being the same collection of masses but at different points/locations. In such a scenario, moving a specific mass m across a distance d would require effort which would cost $m * d$. Accordingly, the EM or Wasserstein distance would be the cost of the minimum effort required to move masses across distances so that the first probability distribution is transformed into the target distribution.

From the equation, $\Pi(P_r, P_g)$ represents all the possible mass transport plans that could transform P_r to P_g with γ representing the masses and $x - y$ representing the distances across which the masses will be moved. The infimum or the smallest effort of all these possible transport plans would then be the EM or Wasserstein distance between the probability distributions P_r and P_g.

A nice example that demonstrates the superiority of measuring the Wasserstein distance as opposed to the KL or JS divergences was introduced by the authors which uses the

three metrics to measure the distance/divergence between two disjoint probability distributions P_0 and P_θ (shown in figure 3.2) defined over R^2. Given a uniform distribution $U[0, 1]$ from which we sample y, if we assume $P_0 = (0, y)$ and $P_\theta = (\theta, y)$ then we have two distributions that have the same height with a distance θ between them (across the x axis).

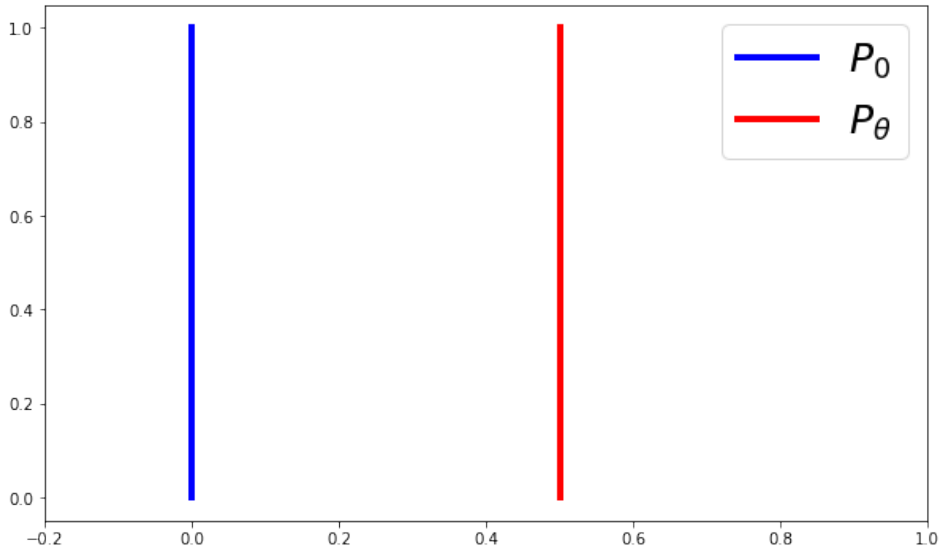


Figure 3.2: A plot showing the two probability distributions (P_0 and P_θ) that do not intersect (i.e. are disjoint) when $\theta \neq 0$.

The following results are obtained from measuring the distance between them using either KL or JS:

$$KL(P_0||P_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

$$JS(P_0, P_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

As can be seen, the KL divergence is zero when the distributions are identical however it

is infinity when there is no overlap between them. JS is slightly better by not reaching infinity when there is no overlap however $\log 2$ is not useful for guiding a network training since it is not differentiable and thus effectively is just as useless in identifying a distance between distributions.

The following simple equation illustrates the EM or Wasserstein distance between the two distributions:

$$W(P_0, P_\theta) = |\theta| \tag{3.9}$$

As demonstrated by the equation, EM is capable of properly measuring the distance between the distributions and its measurement is a function of θ which is variable meaning that it is differentiable and can allow gradient descent.

It is clear that EM is more useful for training a generator network to transform a probability distribution into another desired one however, one major problem faced when using it is the intractability of computing the infimum of all possible efforts or transport plans required to transform the probability distribution. The authors overcame this problem by transforming the EM distance measurement using the Kantorovich-Rubinstein duality which transforms the original equation into the following:

$$W(P_r, P_g) = \sup_{||f||_L \leq 1} E_{x \sim P_r}[f(x)] - E_{x \sim P_g}[f(x)] \tag{3.10}$$

The transformed equation shifts to finding the supremum among all the 1-Lipschitz functions $f : X \to R$. A neural network (in our case the discriminator) can be trained to approximate a 1-Lipschitz function $f_{w w \in W}$ (w represents all its possible weights) to measure the EM or Wasserstein distance:

$$W(P_r, P_g) = \max_{w \in W} E_{x \sim P_r}[f_w(x)] - E_{z \sim P_r(z)}[f_w(g_\theta(z))] \tag{3.11}$$

where g represents the generator and z is its input which could be noise or structured input such as LR images in our case. The discriminator tries to maximize this approximation function which means its maximizing the difference between real data distribution x and generator output distribution $g_\theta(z)$ while the generator will train to minimize the function which will shrink the distance between the two distributions and since it can only affect the second component the result will be that its output distribution will shift closer to that of the real data. Thanks to the fact that the Wasserstein distance is continuous, no matter what state the two networks reach there will always be feedback to help the generator to produce better output.

It is worth pointing out that the discriminator in this case is no longer identifying whether an input is real or produced by the generator. Instead it is now trained to measure the distance between the real data probability distribution and the generator's output distribution which is why it is more accurate to call it the 'critic'. This new generative setup is what the authors termed Wasserstein GAN (**WGAN**).

Another important point is that the above equation only applies if f_w is 1-Lipschitz which the authors achieved by using weight clipping for the discriminator/critic after every update. This is the biggest weakness of WGANs as clipping the weights properly is difficult to achieve without jeopardizing the Lipschitz constraint and even if done properly could lead to slow convergence.

Fortunately, another research work [24] produced a modified version of WGAN that rather than use weight clipping to guarantee the 1-Lipschitz constraint, utilizes instead a gradient penalty for the critic. Since a differentiable function is only 1-Lipschitz if it has gradients with norm at most 1 everywhere, the authors proposed to penalize the critic if its gradient norm moves away from the target norm of value 1. This new model

is widely known as WGAN with gradient penalty (**WGAN-GP**) and overcomes the clipping problems of the normal WGAN.

In our work we utilize a WGAN-GP setup for our second model (3DRDN-WGAN) since it is the state of the art architecture for generative networks to produce output that matches real data the most while mitigating the challenges normally encountered when training a normal GAN architecture such as mode collapse and having to carefully balance the capabilities of either the generator or the discriminator with respect to the other.

3.5.4 Cycle GAN

One of the latest developments in the research field of GANs is the work of [25] which focuses on 'image to image translation' a popular computer vision task where the aim is to transform an image from a specific source domain to a different target domain. Examples of such tasks include style transfer where the aim is transforming a realistic photo into a painted image with the style of a specific painter such as Van Gogh, object transfiguration where for example the task is transforming pictures of horses into ones where the horses are replaced with zebras or season transfer which could switch the season displayed on an image from summer to winter or vice versa.

The architecture developed by the authors called **Cycle-GAN** takes a different approach to solving its computer vision task as the authors were aiming to handle a specific problem namely the lack of paired data for image to image translation tasks. For instance in order to utilize most GANs for learning the mapping required to transform a realistic photo to one painted by Monet, we would require training data composed of pairs of images with one being in the realistic domain and the other being the same image but

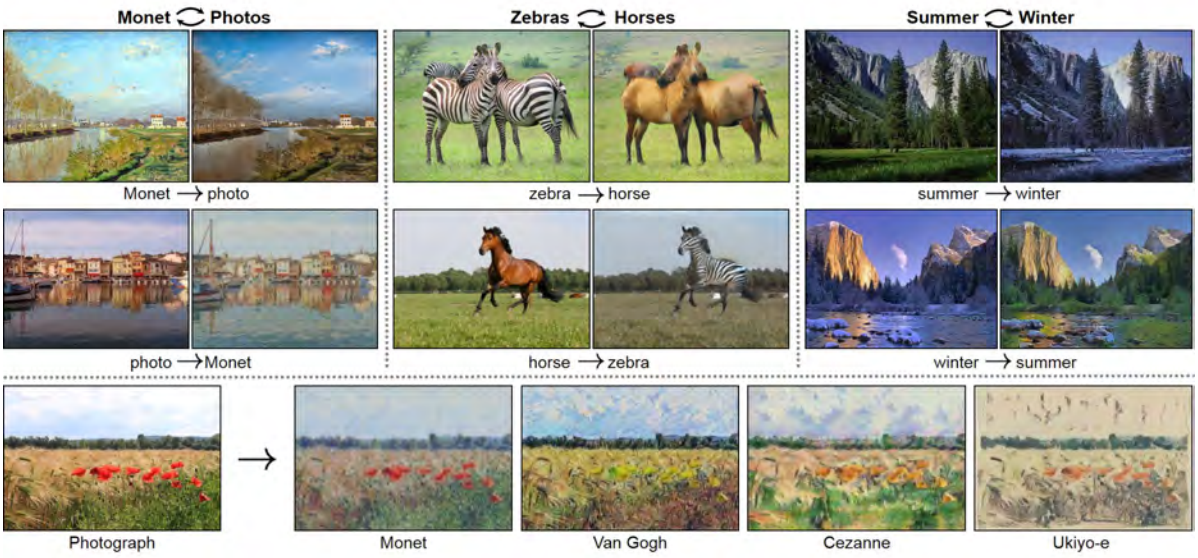


Figure 3.3: A demonstration [25] of some of the possible image to image translations performed using Cycle GANs.

in Monet's painting style. Naturally large quantities of such data would usually be hard to acquire and in many cases almost impossible.

Therefore the authors aimed at designing Cycle-GAN such that it can utilize unpaired data which is more easily obtainable. To solve the previously mentioned example the model would simply need a collection of Monet's paintings and unrelated realistic photos to learn the mapping required to transform an image from one of the two domains to the other one.

In order to achieve that, the model would have to utilize an algorithm that learns an underlying connecting relationship between the two domains. Since it will have to be learnt without paired data, rather than learning the mapping from a specific image from a domain to its counterpart in the other domain, it tries instead to learn the mapping from the probability distribution of an entire set to that of the target set. A standard GAN uses adversarial loss to guarantee that the output of the generator matches the distribution of the target domain thanks to feedback from the discriminator. However

this is not enough to obtain visually acceptable results due to the under-constrained nature of the problem as infinitely many mappings can lead to the same output probability distribution and most of them do not lead to outputs that are paired to their inputs in a meaningful way.

Key to the success of the Cycle-GAN is its integration of the cycle consistency concept in its training. Cycle consistency can be easily explained through a linguistic example: If we have language translation system that can for example translate sentences between English and German then the system is cycle consistent if it translates an English sentence to German and after translating the resulting German sentence back to English it produces the same initial English sentence. The authors demonstrated that utilizing a cycle consistency based loss helps in regularizing the mappings and providing useful constrains when learning them.

Intuitively cycle consistency can only be achieved by learning two mappings that are inverse of each other and linking them during the training to obtain 2 models with each capable of performing transformations from one of the two domains to the other. Therefore Cycle-GAN employs two generators for learning each of the two mappings and in addition to training each of them in an adversarial manner using a discriminator, they are both linked using the cycle consistency loss in their objective loss function.

To learn an image to image translation task between two domains X and Y using Cycle-GAN, two generators G and F are trained to learn the mappings $G : X \rightarrow Y$ and $F : Y \rightarrow X$ respectively. In addition two discriminators D_Y and D_X are used to adversarially train G and F respectively with D_Y forcing G to produces output that is as close as possible to the probability distribution of domain Y and likewise for D_X with F with regards to domain X.

In this scenario, cycle consistency is composed of two components: (a) Forward cycle consistency that makes the generators transform an image from domain Y to X and then transform the result back to domain Y and forces them to produce a result that is as close to the initial image as possible $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. The other component is (b) Backward cycle consistency which is essentially the same concept but in reverse $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$.

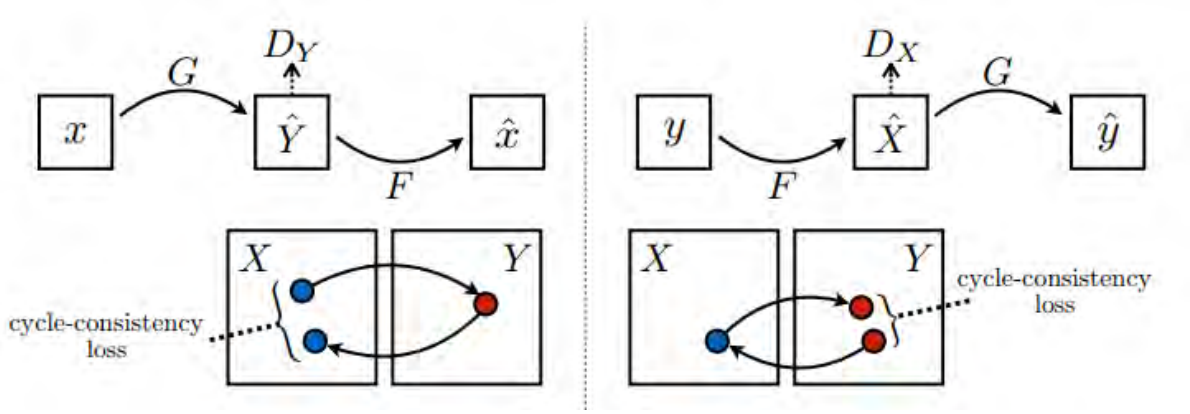


Figure 3.4: An illustration [25] of the cycle consistency loss framework with the left part showing the forward cycle consistency loss and the right part showing the backward version.

Both components are added to the objective loss function during training through the cycle consistency loss function:

$$L_{CYC} = E_{y \sim P_Y}[||G(F(y)) - y||] + E_{x \sim P_X}[||F(G(x)) - x||] \tag{3.12}$$

where P_Y and P_X are the probability distributions of Y and X respectively.

Another important component of Cycle-GAN objective function is the identity loss component which forces each of the generators to not modify parts of their input that are already part of the target domain meaning that $G(x) = x$ and $F(y) = y$. The following

demonstrates the identity loss function:

$$L_{IDT} = E_{x \sim P_X}[||F(x) - x||] + E_{y \sim P_Y}[||G(y) - y||] \tag{3.13}$$

In our work we utilize the Cycle-GAN architecture for our third model (3DRDN-CGAN) since super resolution could be viewed as an image to image translation problem with the HR space being assumed as the Y domain and the LR space as the X domain. However, we modify the objective loss function by replacing the normal adversarial loss used in the original Cycle-GAN with the WGAN-GP adversarial version in our implementation. We also combine it with a supervised loss component since we have paired data for training.

We did however train a version of the model (**3DRDN-UCGAN**) without supervised learning and with data that is unpaired to compare the performance of unsupervised learning models with supervised ones in our experiments. The unsupervised version of the model is more in line with the original Cycle-GAN implementation as it is trained with collections of LR and HR images that are unrelated with each other leading to it solving a more complex image to image translation problem where we transform any image between 2 (HR and LR) domains.

3.6 Skip Connections

CNNs have established themselves as the dominant neural network models in the field of computer vision due to their superior performance. Although their concept was developed over 20 years ago, it is only since a relatively recent period that it became feasible to train deep versions of them thanks to improvement in computing technology [26].

However despite the hardware improvement, challenging problems emerged when training substantially deep CNNs such as the vanishing gradient problem where gradient information gets squashed to zero during the computations before reaching the beginning of the network during gradient back-propagation. Another problem with deep CNNs is performance degradation where the networks perform badly despite being deeper and not overfitting.

Plenty of research work has been done to help mitigate such problems and one result from them is the concept of skip connections which aims at shortening the paths between layers in a way that eases the information/gradient flow. We utilize two types of skip connections which are the DenseNet and Residual connections.

3.6.1 DenseNet Connections

The authors of [26] proposed the Dense Convolutional Network (DenseNet) where they introduced their suggested method of implementing skip connections. In their architecture, every layer in a dense block receives input from all the preceding layers and projects its input into every consecutive layer. The connections are further distinct from other skip connections in that the output features of the layers are combined through concatenation rather than summation. As a result, given a block with L as the number of layers, a densenet version of the block would have $\frac{L(L+1)}{2}$ connections as opposed to a traditional architecture which would have just L.

The following equation shows the output of the l^{th} layer in a DenseNet block:

$$x_l = H_l([x_0, x_1, ..., x_{l-1}]) \tag{3.14}$$

where H_l represents the non-linear transformation done by a layer and $[x_0, x_1, ..., x_{l-1}]$ represents the concatenation of the output from all previous layers.

Despite having more connections, DenseNets require less parameters than traditional networks due to the effect of the dense connections. The reason is that networks in general pass information through the layers with each one ideally extracting useful features and then passing them on. In traditional networks with normal sequential layers, features might need to be relearned or fail to get passed on at each layer. DenseNets however directly pass the extracted features from each layer to all the next layers meaning that no features will have to be relearned. This solid preserving of learned features allows the DenseNet to have convoluted layers with significantly smaller number of needed filters to extract features when compared with the convoluted layers of a normal network.

In addition to needing less filters and parameters, DenseNet connections allow easier flow of information which makes training the network easier as every layer has a direct connection to the last layer from which the gradient back propagates during training. Furthermore, the authors observed that the connections also produce a regularizing effect that helps in minimizing overfitting.

An important hyperparameter that emerges from the DenseNet architecture is the **growth rate** k which represents the number of filters that each layer adds to the concatenated output at the end of the Dense block. Given l layers with k growth rate in a Dense block, the number of output feature maps of the block will be $k_0 + k * l$ where k_0 is the number of feature maps of the input to the block which is also concatenated to the final output. Thanks to the re-usability of feature maps due to the layer wise dense connections, state of the art results could be achieved with a relatively small k when compared with other non-DenseNet based connections [26]. Another important

hyper parameter is the number of units or layers in each Dense block and naturally the number of dense blocks utilized in the network.

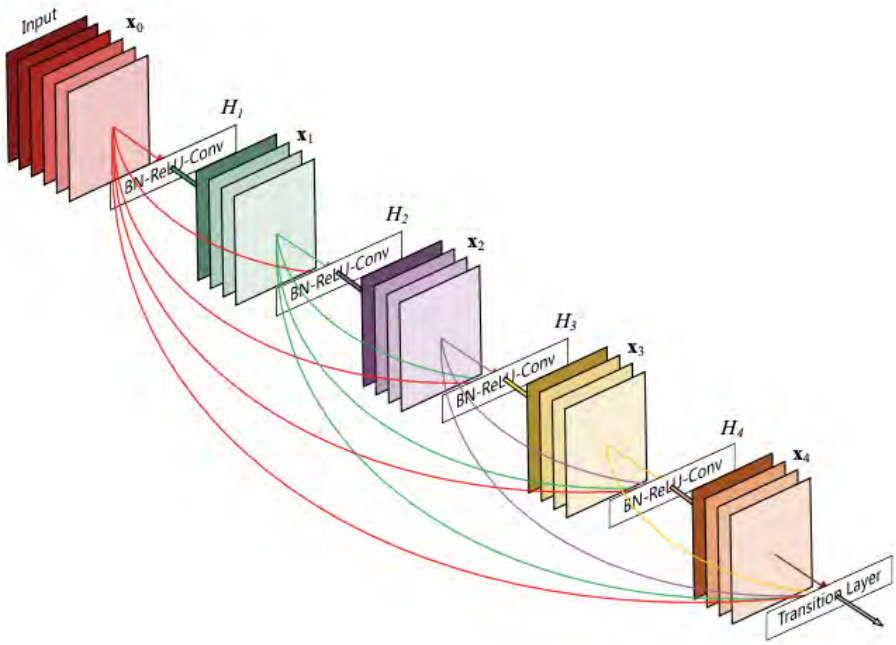


Figure 3.5: An illustration [26] of a dense block example with 5 layers or units.

We performed multiple experiments to determine the most suitable growth rate, number of dense blocks and number of dense units per block for our models. In addition as per the author's suggestion, we also use compression and bottleneck layers (achieved by 1x1 convolutions) between the dense blocks to help control the growth of the concatenated output feature maps which could overwhelm the memory during training.

3.6.2 Residual Connection

Another skip connection that we utilized is the residual one which was popularized by [27] in their widely known network ResNet. Developed also for the purpose of training deeper networks, ResNets were among the first of their kind to be properly trained with

more than 100 layers with state of the art performance on various computer vision tasks such as image classification. The key to ResNet success was the utilization of identity mapping for their shortcut/skip connections which neither added more parameters nor increased the computational complexity.

The following equation shows the output of the l^{th} layer in a ResNet block:

$$x_l = H_l(x_{l-1}) + x_{l-1} \tag{3.15}$$

During training, layers are normally modified through their parameters to approximate a desired function $H_l()$, however the authors hypothesize that it is better to have them instead approximate the residual function $H_l() + r$ where the second value is obtained through identity mapping. Their argument is that in some cases the identity mapping itself will be part of the required function in which case it would be extremely difficult to approximate it using just sequential non-linear layers. In cases where the identity mapping path is more optimal for approximating the function, the training will force the network to ignore the non-linear layers that cause the degradation problem by driving their weights towards zero resulting in overall better performance.

In our network design, we utilize a residual connection from the input to the output which we feel is an intuitive addition to any network designed for performing super resolution as the input and output images of the process are largely similar. This means that our model will be trained to predict the residual which has smaller values and thus smaller features will need to be extracted from the input allowing us to utilize a smaller and more concise model which is faster and also easier to train.

3.7 Network Architecture

In this section we describe the architecture of each of the 3 models implemented and explain our design choices.

3.7.1 3DRDN

The basic 3DRDN model is made up of a single deep 3D convolutional network. DenseNet connections which are described in details in section 3.6.1 are the integral building blocks that make up the model.

The input of the model is made up of 3D image patches with the dimensions 40x40x40 instead of the entire images. We explain in section 4.2 the various ways in which this facilitates the training process.

First the input passes through a 3D convolutional layer which has $2 \cdot k$ as the number of filters where k is the growth rate hyper-parameter of DenseNets. The output of this layer is then propagated into a series of b DenseNet blocks with each being made up of u Dense units. A Dense unit is made up of a batch normalization layer followed by a Leaky RELU based activation layer before a convolutional layer which has k as the number of filters. Figure 3.7 demonstrates the inner architecture of a dense block with u as 4.

Inspired by [22], we also utilize a compressor layer (essentially a 1x1x1 convolutional layer) before each DenseNet block to force each block to only have $2 \cdot k$ feature maps as its input. This also means that each DenseNet block produces an output that has $2 \cdot k + u \cdot k$ feature maps since there are u units in each block. Moreover, we also insert a bottleneck layer (also essentially a 1x1x1 convolutional layer) with $4 \cdot k$ filters before

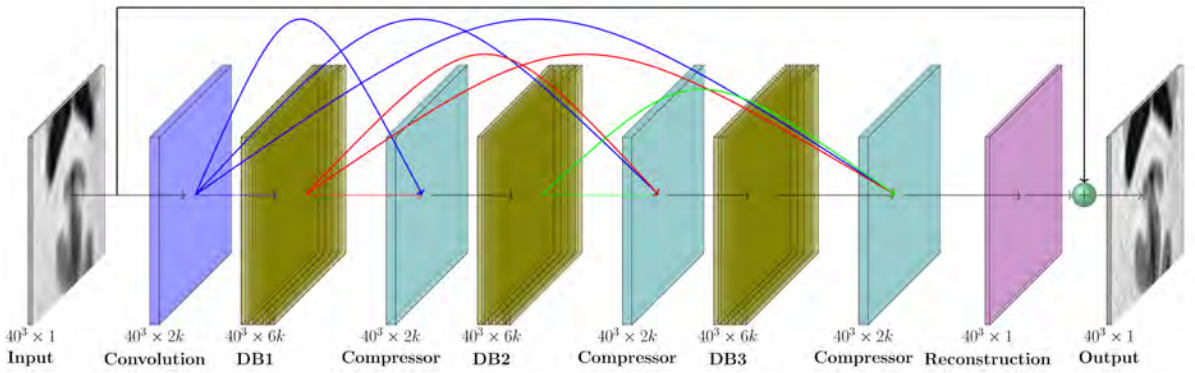


Figure 3.6: Architecture of the 3DRDN model with 3 dense blocks (b). In addition to the layer dense connections between the first convolutional layer and each dense block, we also utilize a residual connection from the input image to the output of the reconstruction layer. Figure 3.7 illustrates the architecture of a dense block.

any dense unit that gets an input with more than $4 \cdot k$ feature maps as recommended by the authors of the DenseNet paper [26].

Furthermore, the output of all of the dense blocks are also connected in a DenseNet fashion with each other as well as with the output of the first convolutional layer. Since we have b Dense blocks with each having u dense units, the final concatenated dense output output will have $2 \cdot k + b \cdot u \cdot k$ feature maps. Unlike [22], we insert this output into another compression layer with $2 \cdot k$ filters, which significantly improved our model's performance, followed by a reconstruction layer that is made up of a 1x1x1 convolutional layer which produces a single feature map representing the residual output of the model. This residual output is then summed with the 3D image patch input using a residual connection to produce the final output of the model. Figure 3.6 illustrates a high-level overview of the architecture.

For all of the convolutional layers across the entire model, with the obvious exception of compressor, bottleneck and reconstruction layers, we utilize 3x3x3 as the size of the filter (kernel) and utilize same padding for all of the convolutional layers to minimize loss of information through the continuous shrinkage of the feature maps size.

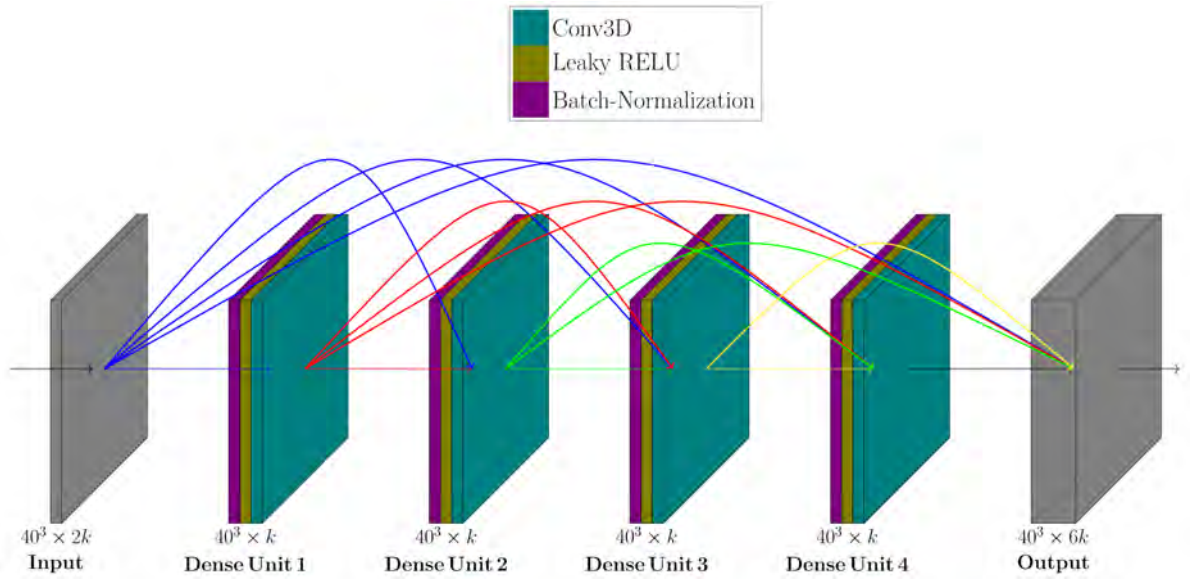


Figure 3.7: Detailed visualization of a dense block from the 3DRDN model with 4 dense units (u). As illustrated, the output of each layer (including the input layer) is concatenated into the input of all the layers following its next layer.

We performed multiple experiments to identify the optimum values for the DenseNet hyperparameters of the model which are k the growth rate, b the number of dense blocks and u the number of dense units per dense block. Designing a deeper and wider (with more filters) model almost always guarantees superior performance, however this comes at a cost of having significantly more computation time which could limit the utilization of such a model for real life applications such as super resolution and furthermore it would require more data to be properly trained. Accordingly, we aimed at striking a balance between designing the model to be deep enough to successfully perform super resolution while also being lightweight enough to realistically be usable in upscaling large 3D models. We noticed early during the experiments that 12 is the lowest value for k with which we can effectively perform super resolution and since increasing this hyperparameter contributed heavily to slowing down training, we ended up simply choosing 12 in our final design. b and u were more flexible to work with and after experimenting we selected the values of 3 and 4 for them respectively.

To induce non-linearity in the model, an activation function was needed and accordingly we also performed experiments to identify the most suitable one among the most popular functions: RELU, ELU, Swish and Leaky RELU. We achieved good results with both RELU and Leaky RELU but ended up choosing the latter since it doesn't suffer from the 'dead RELU' problem where a node that receives negative input gets stuck outputting zero with little chance of recovery since the function gradient at zero is also zero. Leaky RELU however, overcomes this problem by adding a small positive gradient to negative inputs which provides a better chance for recovery. The following equation demonstrates the Leaky RELU function:

$$\mathrm{f(x)} \equiv \begin{cases} x, & \text{if } x > 0 \\ \alpha * x, & \text{if } x \leq 0 \end{cases}$$

where α is the constant variable that induces the gradient for negative input values. We compared our model's performance with different values and in the end we utilized a value of 0.2 for α in our implementation.

Lastly we also performed experiments to determine the most suitable initializer for the weights of the model. We performed experiments with the following initialization techniques: HeUniform, HeNormal, XavierUniform and XavierNormal. Since we obtained faster convergence with HeUniform, we ended up selecting it for our model. The bias parameters of the model, on the other hand, were simply initialized with zero values.

3.7.2 3DRDN-WGAN

The 3DRDN-WGAN is made up of 2 networks that implement a WGAN-GP model: a generator which has the architecture of 3DRDN and a critic which is another network

trained to measure the Wasserstein distance between the real data probability distribution and the generator's output distribution.

The critic that we implemented was inspired mainly by the discriminator from the work [11] with 3 important modifications. First of all, we modified our version so that it can handle 3D images rather than 2D ones by replacing its 2D convolutions with 3D ones. The second modification is that we removed, from its final layer, the sigmoid cross entropy activation function since our version is a WGAN-GP rather than a normal GAN. Thus our critic network outputs a scalar estimation of the Wasserstein distance instead of a binary output that determines whether its input is true data or from the generator as the case with normal GANs. The third and last modification is replacing its batch normalization layers with layer normalization layers as recommended by the authors of WGAN-GP [24] in order to not mess up the gradient penalty design. Unlike the generator, the critic does not have any skip connections and is a normal convolutional feed forward network.

The input of the critic are also 3D image patches with dimensions 40x40x40 which go through a convolutional layer with $k = 64$ filters followed by a Leaky RELU layer. As suggested by [11], we do not utilize valid padding to compress the feature space but utilize instead strides. We implement a sequence of blocks that are made up of a convolutional layer followed by layer normalization and then by a Leaky RELU layer. For each block we alternate between utilizing a stride of 1 and stride of 2 for its convolutional layer until we shrink the feature images to a size of 3x3. Furthermore, every 2 blocks we double the filters utilized for the convolutional layers until we reach 512 filters.

The 512 feature images are flattened and then followed by a dense layer of size 1024 which is followed by another Leaky RELU layer before a final dense layer that produces

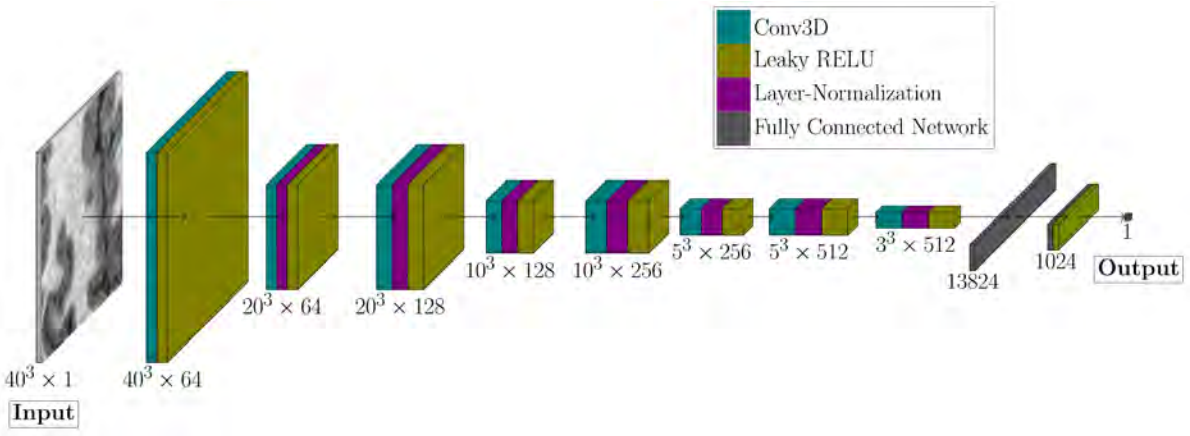


Figure 3.8: Architecture of the critic of the 3DRDN-WGAN model. The convolutions from the hidden layers alternate between doubling the number of input filters and decreasing the size of the feature map by using a value of 2 for strides.

a single non-activated output which represents the Wasserstein distance that the network will be trained to measure. Figure 3.8 shows an overview of the architecture of the critic.

Just like with the generator, for all of the convolutional layers of the critic we utilize filters with the size 3x3 and all of its Leaky RELU layers use a value of 0.2 for alpha. Its weights are also initialized using HeUniform while the biases are zero initialized.

3.7.3 3DRDN-CGAN

Inline with the Cycle-GAN architecture, the 3DRDN-CGAN model is composed of two generators and two critics. Fortunately in our design the HR and LR images have the same resolution thanks to the fact that interpolation is implemented in the data pre-processing step as described in section 3.4.

This allows us to utilize the same architecture for the generator G which performs upscaling as well as the generator F which performs downscaling and the same applies for the two critics as well. The two generators utilize the architecture of the 3DRDN

model while the two critics use the same architecture of the critic from the 3DRDN-WGAN model.

3.8 Objective Loss Function

To force our networks to learn the non-linear mapping between the HR and LR domain we utilize training data from which we can input LR 3D image patches to a model and compare its output $\hat{y}$ with ground truth y which is the original HR version of the 3D image patch. We utilize an objective loss function L to calculate the error of the output of the model which is minimized by modifying the parameters θ (weights and biases) of the model using the Adam optimizer and backpropagation. The following equation summarises the overall training process:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i}^{N} L(\hat{y}_i, y_i) \tag{3.16}$$

where N is the number of LR & HR image pairs used in the training.

Each model has its objective function with each one incorporating the previous model's function and adding new components to it.

3.8.1 3DRDN

We begin with the simplest model 3DRDN which has the following objective loss function:

$$L_{3DRDN} = L_{SUP}(I^{HR}_{x,y,z}, I^{SR}_{x,y,z}) \tag{3.17}$$

where L_{SUP} represents the supervised loss component, $I^{HR}_{x,y,z}$ represents the HR 3D image patches which are the ground truth and $I^{SR}_{x,y,z}$ represents the upscaled patches which are the output of the model.

L_{SUP} forces the model to produce 3D images which are as close as possible to their HR counterparts. We utilize the MAE or L_1 loss for the supervised training as it performed better in our experiments compared with the MSE or L_2 loss:

$$L_{SUP}(I^{HR}_{x,y,z}, I^{SR}_{x,y,z}) = E[|I^{HR}_{x,y,z} - I^{SR}_{x,y,z}|] \tag{3.18}$$

where $E()$ denotes the expectation operator or the mean value for each the 3 dimensions.

3.8.2 3DRDN-WGAN

The next model is the 3DRDN-WGAN which integrates the previous basic model in a GAN architecture where a generator and a critic are trained together in an adversarial manner to simultaneously enhance their performance. The following equation demonstrates the relationship between the critic and the generator:

$$\min_G \max_D L_{WGAN}(I^{HR}_{x,y,z}, I^{SR}_{x,y,z}) = -E[D(I^{HR}_{x,y,z})] + E[D(I^{SR}_{x,y,z})] + \lambda E[(||\Delta D(I^{\tilde{HR}}_{x,y,z})||_2 - 1)^2] \tag{3.19}$$

The first two components are for the Wasserstein distance estimation; the estimation is based on the function shown in equation 3.11 which is an approximation of the Kantorovich-Rubinstein duality transformation of Wasserstein/EM distance. The signs

are reversed by multiplying the approximation function by -1 which however does not change the behavior of either the critic or the generator.

The critic tries to maximize its output score on input from the generator while minimizing its score on real data input. That way it learns to identify the probability distribution of the two possible input data and measure the distance between them in an unsupervised learning way. The generator on the other hand tries to minimize the output score of the critic on its input leading to the generator's output distribution moving closer to that of the real data which will make it more visually similar. In machine learning terms, we can say that the critic learns to find the embedding or latent representation for its input image patches, separate them into 2 clusters and maximize the distance between them while the generator tries to minimize it.

The third component is an addition that allows training the modified WGAN (WGAN-GP) without having to resort to gradient clipping which was a requirement in the original WGAN. The component is added instead to penalize deviation of the gradient norm of the critic from one which forces the Lipschitz constraint as we discussed in section 3.5.3. $I^{\tilde{HR}}_{x,y,z}$ is a uniform sampling from pairs of $I^{HR}_{x,y,z}$ and $I^{SR}_{x,y,z}$ and λ is a regularization parameter.

Only the second term is affected by the generator (since $I^{SR}_{x,y,z}$ is its output) which is why its the only part incorporated into its objective loss function while the critic incorporates all three.

The following is the objective loss function of the generator:

$$L_{3DRDN-WGAN} = L_{SUP}(I^{HR}_{x,y,z}, I^{SR}_{x,y,z}) + \lambda_1 L_{ADV}(I^{SR}_{x,y,z}) \tag{3.20}$$

where λ_1 is a weighting parameter for which we use the value 0.01 based on experimentation. $L_{ADV}(I^{SR}_{x,y,z})$ is the adversarial loss:

$$L_{ADV}(I^{SR}_{x,y,z}) = -E[D(I^{SR}_{x,y,z})] \tag{3.21}$$

The following is the objective loss function of the discriminator:

$$L_{Discriminator} = E[D(I^{SR}_{x,y,z})] - E[D(I^{HR}_{x,y,z})] + \lambda E[(||\Delta D(I^{\tilde{H}R}_{x,y,z})||_2 - 1)^2] \tag{3.22}$$

We utilize a value of 10 for λ which is the convention in most of the WGAN-GP implementations.

3.8.3 3DRDN-CGAN

The final model is the 3DRDN-CGAN which integrates two of the previous 3DRDN-WGAN model in an architecture where one pair is trained to perform super resolution and the other pair is trained to perform the inverse which is downscaling.

The following equation demonstrates the objective loss function of the generator G which is responsible for super resolution:

$$L_G = L_{SUP}(I^{HR}_{x,y,z}, I^{SR}_{x,y,z}) + \lambda_1 L_{ADV}(I^{SR}_{x,y,z}) + \lambda_2 L_{CYC}(I^{HR}_{x,y,z}, I^{LR}_{x,y,z}) + \lambda_3 L_{IDT}(I^{HR}_{x,y,z}) \tag{3.23}$$

with the two new additions being the cycle consistency loss L_{CYC} and the identity loss L_{IDT} which are the integral components of the CycleGAN architecture which we described in section 3.5.4. Each λ is a hyperparameter responsible for the weighting of its accompanying loss component. The variable $I^{LR}_{x,y,z}$ represents the LR 3D image

patches that are used to train the other generator F in its supervised loss function and which are also utilized in the cycle consistency loss of the generator G.

The other generator F, which performs downscaling, has a mostly similar objective loss function but with different inputs for the components:

$$L_F = L_{SUP}(I^{LR}_{x,y,z}, I^{DS}_{x,y,z}) + \lambda_1 L_{ADV}(I^{DS}_{x,y,z}) + \lambda_2 L_{CYC}(I^{HR}_{x,y,z}, I^{LR}_{x,y,z}) + \lambda_3 L_{IDT}(I^{LR}_{x,y,z}) \tag{3.24}$$

where $I^{DS}_{x,y,z}$ represents the output images of the generator F which are the downscaled output of their original HR versions.

After performing various experiments we ended up utilizing the values 0.01, 0.01 and 0.005 for λ_1, λ_2 and λ_3 respectively in the loss function of both generators.

The following equation demonstrates the cycle consistency loss L_{CYC} which links the generator G, responsible for SR, with the generator F which is responsible for downscaling:

$$L_{CYC} = E[||G(F(I^{HR}_{x,y,z})) - I^{HR}_{x,y,z}||] + E[||F(G(I^{LR}_{x,y,z})) - I^{LR}_{x,y,z}||] \tag{3.25}$$

The loss encourages the resultant value of $G(F(I^{HR}_{x,y,z}))$ to be as close as possible to $I^{HR}_{x,y,z}$ and likewise $F(G(I^{LR}_{x,y,z}))$ to $I^{LR}_{x,y,z}$ which embeds forward cycle consistency and backward cycle consistency respectively for each part of the equation. This forces the two generators to use the same latent space for feature mapping which helps prevent degeneracy during their learning by adding more constraints.

The other integral loss component of a CycleGAN system is the identity loss L_{IDT}:

$$L_{IDT} = E[||F(I^{LR}_{x,y,z}) - I^{LR}_{x,y,z}||] + E[||G(I^{HR}_{x,y,z}) - I^{HR}_{x,y,z}||] \tag{3.26}$$

which forces the generator G to not modify any input which is already in the HR domain and likewise for the generator F with regards to the LR domain. While its an integral component in the original Cycle-GAN design especially for object transfiguration tasks, we feel that it is not integral for super resolution as we expect all parts of the LR image to be upscaled rather than just a specific part. For that reason we decreased its contribution to the objective function by giving its weighting parameter a very small value.

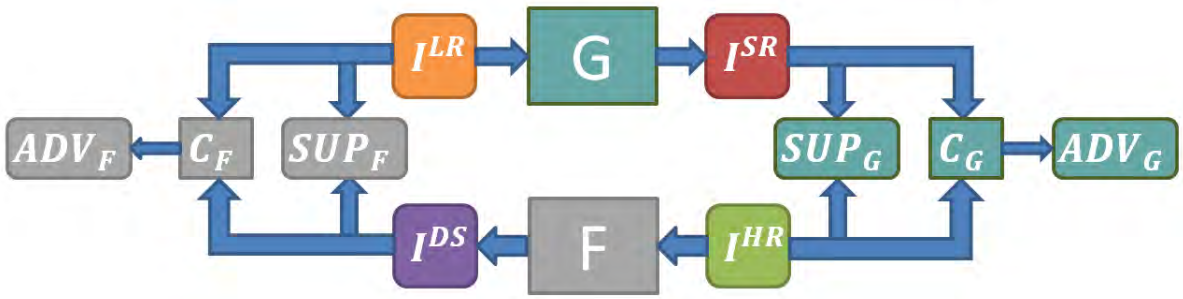
The objective loss functions of the two critics are the same as that of the critic from the 3DRDN-WGAN model. Figure 3.9 provides a visual representation of the training framework of the 3DRDN-CGAN model.

Lastly we show the objective function of a modified version (**3DRDN-UCGAN**) of the third model that relies completely on unsupervised training.

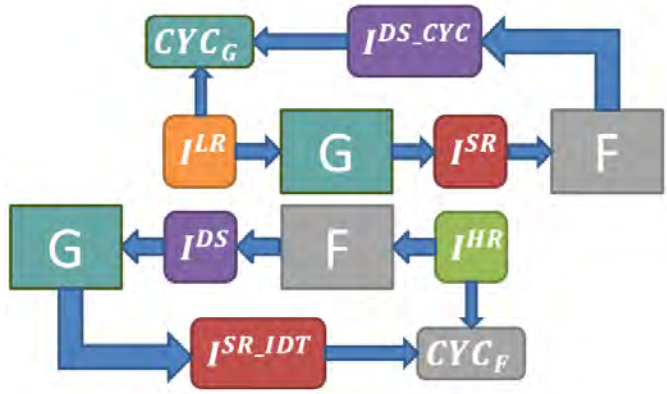
The following is the objective loss function of its generator G:

$$L_G = \lambda_1 L_{ADV}(I^{SR}_{x,y,z}) + \lambda_2 L_{CYC}(I^{HR}_{x,y,z}, I^{LR}_{x,y,z}) + \lambda_3 L_{IDT}(I^{HR}_{x,y,z}) \tag{3.27}$$

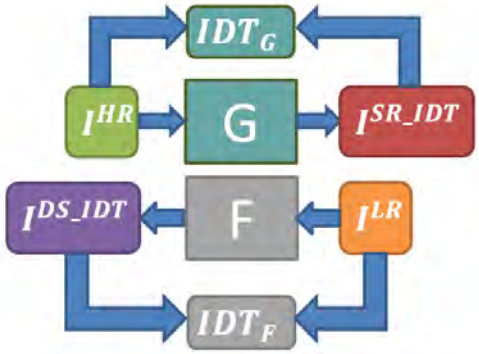
As can be seen, it does not have the supervised loss component and can therefore be trained without paired data which is also the case for generator F. For the 3DRDN-UCGAN model We utilize the values 0.1, 0.1 and 0.05 for λ_1, λ_2 and λ_3 respectively in the loss function of both generators.



(a) Supervised and Adversarial Loss



(b) Cycle Consistency Loss



(c) Identity Loss

Figure 3.9: Overview of the framework for the training of the 3DRDN-CGAN model. I^{HR}, I^{LR}, I^{SR} and I^{DS} are respectively the high-resolution, low-resolution, upscaled and downscaled images. G, F are generator models and C_G and C_F are their respective critic models. SUP, ADV, CYC and IDT are respectively the supervised, adversarial, cycle consistency and identity losses.

4 Experiments

In this section, we describe in detail the experiments that we performed. First we introduce the datasets that were used for training our models and evaluating their performance. Then we describe the pre-processing that was done to obtain downscaled LR images from the original HR ones as well as the pre-processing performed to fit all input imagess into the architecture of our models. We then explain the settings that were utilized for training the models in terms of their hyperparameters and other training details. Lastly, we introduce the metrics that we utilized for evaluating the results.

4.1 Datasets

Since our models operate exclusively on 3D images, we required data with such dimensions to train them and for that we obtained two datasets.

The first dataset is from the MGH-USC Human Connectome Project which aggregates a collection of publicly available diffusion and anatomical neuroimaging data that is collected from various projects and studies from all over the neuroimaging community. Our data was particularly provided by the ABIDE project from which we obtained MRI scans of 1087 subjects and all of the scans were 3D T1W images of exclusively human

brains. The scans do not have a uniform resolution but the average value of their dimensions is approximately 256x261x184 voxels for each scan. We refer to this dataset as the **ABIDE dataset**.

The other dataset that we utilized is composed of a single CT scan of a Peruvian mummy which was provided by the Chair of Digital Image Processing from Passau University. We refer to this dataset as the **Mummy dataset**. Aside from being a CT scan rather than MRI, two other significant aspects differentiate this dataset from the ABIDE one. Firstly, the scan covers the entire body of the mummy and even includes non-human items such as corn meaning that it has more visual variety despite being a single scan. The other difference is the substantially larger size of the scan as it has a dimension of 2304x1896x1896 voxels which makes it require extra pre-processing before being utilized in training our models. We crop large quantities of voxels from the scan that contain zero values to avoid having them in our training samples. Additionally, we also chop it into 24 smaller scans so that we can later split its data between training, validation and testing sets.

4.2 Preprocessing

Before our models can process any of the images from the datasets, we first have to perform some preprocessing on them which allows our models to operate on them and that also provides us with HR and LR paired data for training.

Firstly, to have paired data for training we use the HR (ground truth) scans to obtain LR counterpart versions of them. We follow the convention of most superresolution research work in trying to simulate the CT/MRI scanning process by first adding Gaussian noise

with a standard deviation of 0.25 to the HR images and then lowering their 3 dimensional resolution by the desired upscaling factor using bicubic interpolation to get the LR counterparts. Before inserting the LR images to the models, we use again bicubic interpolation to increase their resolution back to the HR version size before processing them using our models as explained in section 3.4.

We then normalize all of the images to a range between 0 to 1 so that our models can more easily operate on them and converge faster while training.

Rather than operate on an entire scan, our models are designed to only process a 3D image patch of size 40x40x40 which provides two benefits. The first is that all of the 3D images from our datasets (or any CT/MRI dataset for that matter) are large enough that performing computations on them using normal computing resources is practically impossible due to the limited memory resources available.

This problem can easily be circumnavigated by simply splitting any of the images into patches with the size 40x40x40 which are significantly easier to process and perform computations on. The other benefit is that splitting an image into patches provides more data for training and helps in minimizing overfitting while training our models. One minor disadvantage that should be pointed is that this minimizes the information that a model has access to when upscaling as rather than working on the entire image it only operates on patches of it one at a time.

However, as our experiments indicate, even with such limited information, the models can still produce perceptually good looking upscaled versions of each patch which can then be aggregated together to obtain the upscaled version of the entire image. The size of 40x40x40 was chosen for the patches as it was small enough to easily insert multiple

instances of it into the GPU memory for computations while also being large enough to contain enough information for the models to perform upscaling with acceptable results.

For only the training data, after extracting the patches we additionally perform data augmentation on them by randomly flipping the image around any of the 3 axes. Furthermore, during training at each epoch we extract the patch from a random location from each image which acts as another form of data augmentation that helps in minimizing overfitting. Figure 4.1 summarises all of the preprocessing performed in our experiments.

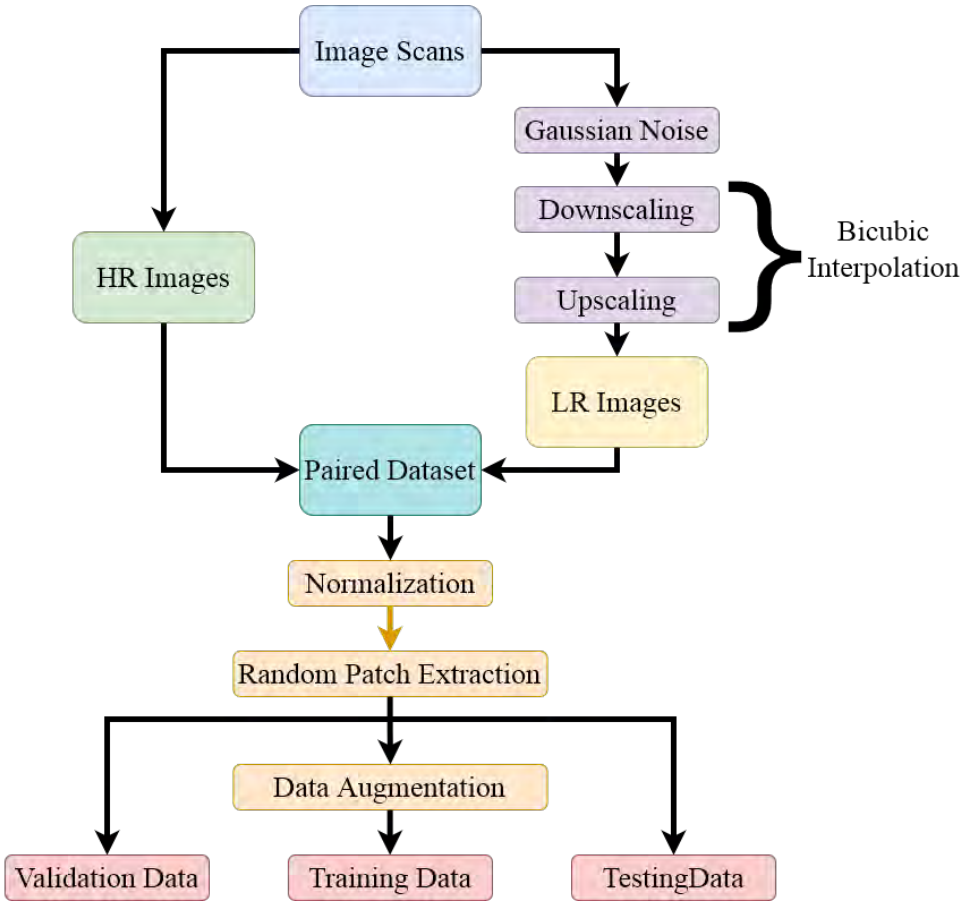


Figure 4.1: An illustration of the preprocessing steps performed before training.

4.3 Training Settings

For training, we implemented a dataset pipeline using tensorflow that extracts the images from the disk and performs all of the preprocessing steps while utilizing prefetching and caching to speed up the training process.

The 1087 images from the ABIDE dataset were split into 760 for training, 163 for validation and 164 for testing. The Mummy dataset, which has only 24 images, was split into 16, 4 and 4 for training, validation and testing respectively. The models rely only on the training data for optimizing their parameters with the validation data being used only for evaluation after every epoch to checkout for overfitting while the testing data is only used for evaluation at the end after the training is complete.

When training the 3DRDN model we utilized a batch size of 8 3D images patches and trained the model for 400 epochs with a learning rate of 1×10^{-4} which took around 14 hours of training time.

For the 3DRDN-WGAN model, we first fully train a 3DRDN model and carry its weights into the generator of the 3DRDN-WGAN model. Then we only train the critic for 30 epochs without updating the generator to help the critic catch up to the generator in terms of performance capabilities. Afterwards, we start training both of them for 300 epochs however, as recommended by the WGAN designers [23], for each iteration of training the generator, we train the critic for 3 iterations to make sure the critic can more accurately estimate the Wasserstein distance before the generator attempts to minimize it. We use a learning rate of 5×10^{-6} and a batch size of 6 with the training taking around 3 days and 6 hours to complete.

For the 3DRDN-CGAN (as well as 3DRDN-UCGAN) model, we likewise train the two

generators fully first before transferring their weights. Afterwards, we also only train the two critics for 30 epochs before training all 4 networks with the same cycle of 3 critic training iterations followed by 1 for the generators for 300 epochs. The learning rate utilized was also 5×10^{-6} while the batch size was only 2 to avoid encountering out of memory (OOM) errors. Training either of the 3DRDN-CGAN or 3DRDN-UCGAN models took the longest time which was approximately 10 days with our computing resources.

All the of the training was done using the Adam optimizer with values of 0.9 and 0.999 for β_1 and β_2 respectively for minimizing our objective loss functions. The models were developed using the Tensorflow version 2 library and the training was done using an Nvidia Geforce RTX 3090 GPU with a RAM memory of 24 Gigabytes.

4.4 Evaluation

In our work we utilize, in addition to MAE, two popular techniques for performing image quality assessment in accordance with most contemporary research. We describe the techniques, which are the Peak Signal to Noise Ratio (PSNR) and the Structural Similarity Index Measure (SSIM), in this section:

4.4.1 PSNR

PSNR is an image quality mathematical measurement that operates on voxel to voxel (or pixel) basis, allowing us to estimate how close each voxel of the upscaled image is to the original image. It represents the ratio of the maximum possible power of a signal

to the power of noise that corrupts its representation or fidelity. The following is its equation:

$$PSNR = 10 \cdot \log_{10} \frac{s^2}{MSE} \tag{4.1}$$

where s is the maximum value obtained with the bit representation of the image and MSE is the mean squared error between the voxels.

4.4.2 SSIM

SSIM is designed to compare aspects of the images that have the most effect on human visual perception. It achieves that by comparing the luminance, contrast and structure of the original image with the upscaled one in a combinatorial manner. For it to be calculated, we need to first obtain the mean, standard deviation and covariance values from the two compared images x and y by using an 11x11 filter kernel:

$$\mu_x = \frac{1}{T}\sum_{i=1}^{T} x_i \quad \mu_y = \frac{1}{T}\sum_{i=1}^{T} y_i$$

$$\sigma_x^2 = \frac{1}{T-1}\sum_{i=1}^{T}(x_i - \overline{x})^2 \quad \sigma_y^2 = \frac{1}{T-1}\sum_{i=1}^{T}(y_i - \overline{y})^2$$

$$\sigma_{xy}^2 = \frac{1}{T-1}\sum_{i=1}^{T}(x_i - \overline{x})(y_i - \overline{y})$$

SSIM is then calculated using the following equation:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4.2}$$

where $c1$ and $c2$ are two variables added to stabilize the division based on the dynamic range of the voxel values.

5 Results

In this section, we show the results of various experiments that we performed to demonstrate the capabilities of our models both quantitatively in terms of evaluation metric scores as well as qualitatively by showcasing visual results of their super-resolution output.

As baseline, we utilize the following upscaling methods: Nearest Neighbour, Bilinear, Bicubic interpolation and Lanczos resampling methods. For quantitative evaluation, we utilize the 3 metrics PSNR, SSIM and MAE to measure how similar is the upscaling output of each method (with LR images as input) to the HR (ground truth) images. We also demonstrate a sample of the visual super-resolution output of all interpolation methods and our models on the same input to allow comparing the quality of their super-resolution. We should point out that all of the scans are 3D and that the 2D images illustrated in this thesis are 2D slices from the 3D scans; we also demonstrate 3D snippets of upscaled scans of all of the methods and models.

To make the comparison fair, we only operate on the unseen testing data from both datasets. Therefore, the experiments on the ABIDE dataset were performed on the 164 testing MRI scans while the experiments on the Mummy dataset were performed on the 4 testing CT scans.

5.1 ABIDE Dataset

Table 5.1 demonstrates the performance of each interpolation method as well as our models on the ABIDE datatset for super-resolution with an upscaling factor of 2. As can be seen, the 3DRDN model obtains the best performance in terms of all 3 evaluation metrics with the 3DRDN-CGAN model obtaining the second best performance.

Quantitative Evaluation on the ABIDE Dataset (x2 Scaling Factor)			
Method	PSNR †	SSIM †	MAE ‡
Nearest Neighbour	22.876 ± 2.567	0.767 ± 0.121	0.069 ± 0.020
Bicubic	19.487 ± 2.015	0.752 ± 0.138	0.106 ± 0.024
Bilinear	19.780 ± 2.648	0.764 ± 0.109	0.103 ± 0.031
Lanczos	18.515 ± 1.939	0.731 ± 0.143	0.119 ± 0.025
3DRDN	$\mathbf{31.781 \pm 3.069}$	$\mathbf{0.887 \pm 0.058}$	$\mathbf{0.021 \pm 0.019}$
3DRDN-WGAN	30.703 ± 2.872	0.850 ± 0.060	0.024 ± 0.022
3DRDN-UCGAN	28.982 ± 3.499	0.848 ± 0.091	0.032 ± 0.026
3DRDN-CGAN	$\underline{31.033 \pm 2.977}$	$\underline{0.872 \pm 0.055}$	$\underline{0.023 \pm 0.021}$

Table 5.1: Quantitative evaluation on x2 upscaling with the testing data from the ABIDE dataset. † indicates that the higher the score the better while ‡ indicates that the lower the score the better. **Bold** indicates the best value while underline indicates the second best value.

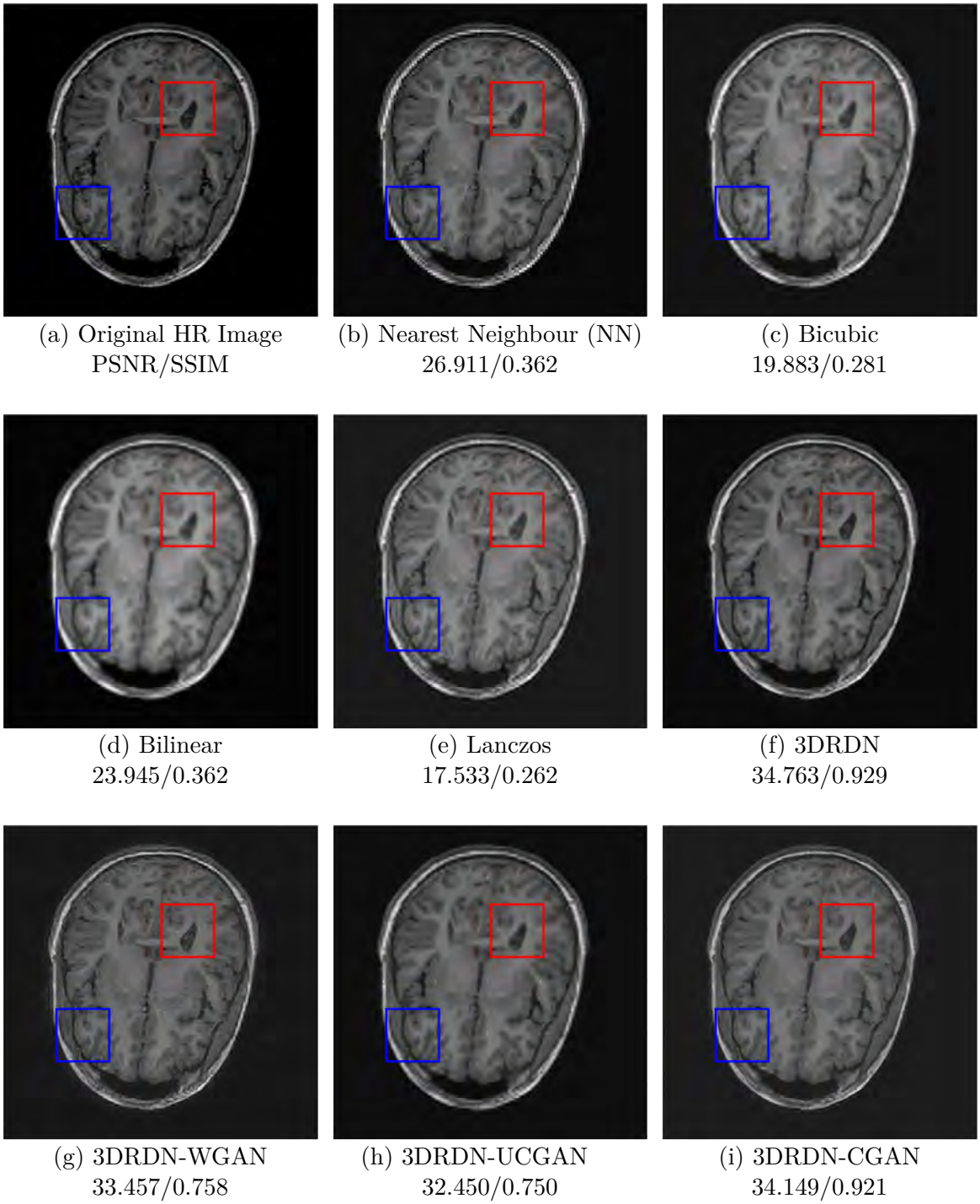
Figure 5.1 demonstrates the x2 upscaling performed by each method on an image scan from the ABIDE dataset. For further easier visual comparison, in figure 5.2, two patches are extracted from the same position from all of the images to allow visually comparing their minute details as well. We also showcase 3D comparison of snippets of x2 upscaled images in figure 5.3. Furthermore, additional visual upscaled sample results from the ABIDE dataset are added in appendix A.1.

In addition we performed another experiment to evaluate the performance of our models on super-resolution with a scaling factor of 4. Table 5.2 demonstrates the metric results of each interpolation method and our models on upscaling testing data from the ABIDE dataset with factor of 4. Once again, the 3DRDN model obtains the best metric scores with this time the 3DRDN-WGAN model coming second.

Quantitative Evaluation on the ABIDE Dataset (x4 Scaling Factor)			
Method	PSNR †	SSIM †	MAE ‡
Nearest Neighbour	20.059 ± 2.670	0.529 ± 0.133	0.091 ± 0.029
Bicubic	18.359 ± 2.275	0.606 ± 0.140	0.118 ± 0.031
Bilinear	17.256 ± 2.931	0.599 ± 0.127	0.138 ± 0.046
Lanczos	15.913 ± 1.658	0.544 ± 0.139	0.157 ± 0.031
3DRDN	$\mathbf{30.687 \pm 6.312}$	$\mathbf{0.763 \pm 0.090}$	$\mathbf{0.025 \pm 0.030}$
3DRDN-WGAN	$\underline{29.575 \pm 7.204}$	$\underline{0.712 \pm 0.128}$	$\underline{0.028 \pm 0.037}$
3DRDN-UCGAN	27.988 ± 4.597	0.542 ± 0.250	0.034 ± 0.034
3DRDN-CGAN	29.299 ± 5.918	0.673 ± 0.115	0.029 ± 0.035

Table 5.2: Quantitative evaluation on x4 upscaling with the testing data from the ABIDE dataset. † indicates that the higher the score the better while ‡ indicates that the lower the score the better. **Bold** indicates the best value while underline indicates the second best value.

We also showcase visual results of the upscaling achieved by all methods and our models with a factor of 4 in figure 5.4 while figure 5.5 also shows more detailed results by illustrating patches from the x4 upscaled images.



(a) Original HR Image
PSNR/SSIM

(b) Nearest Neighbour (NN)
26.911/0.362

(c) Bicubic
19.883/0.281

(d) Bilinear
23.945/0.362

(e) Lanczos
17.533/0.262

(f) 3DRDN
34.763/0.929

(g) 3DRDN-WGAN
33.457/0.758

(h) 3DRDN-UCGAN
32.450/0.750

(i) 3DRDN-CGAN
34.149/0.921

Figure 5.1: Visual comparison of the (x2) upscaling performed on an entire MRI scan from the ABIDE dataset using the baseline interpolation methods and our models. Below each upscaled image are its metric scores that compare how close it is to the original HR image. The red and blue patches are further highlighted in figure 5.2 for more detailed comparison.

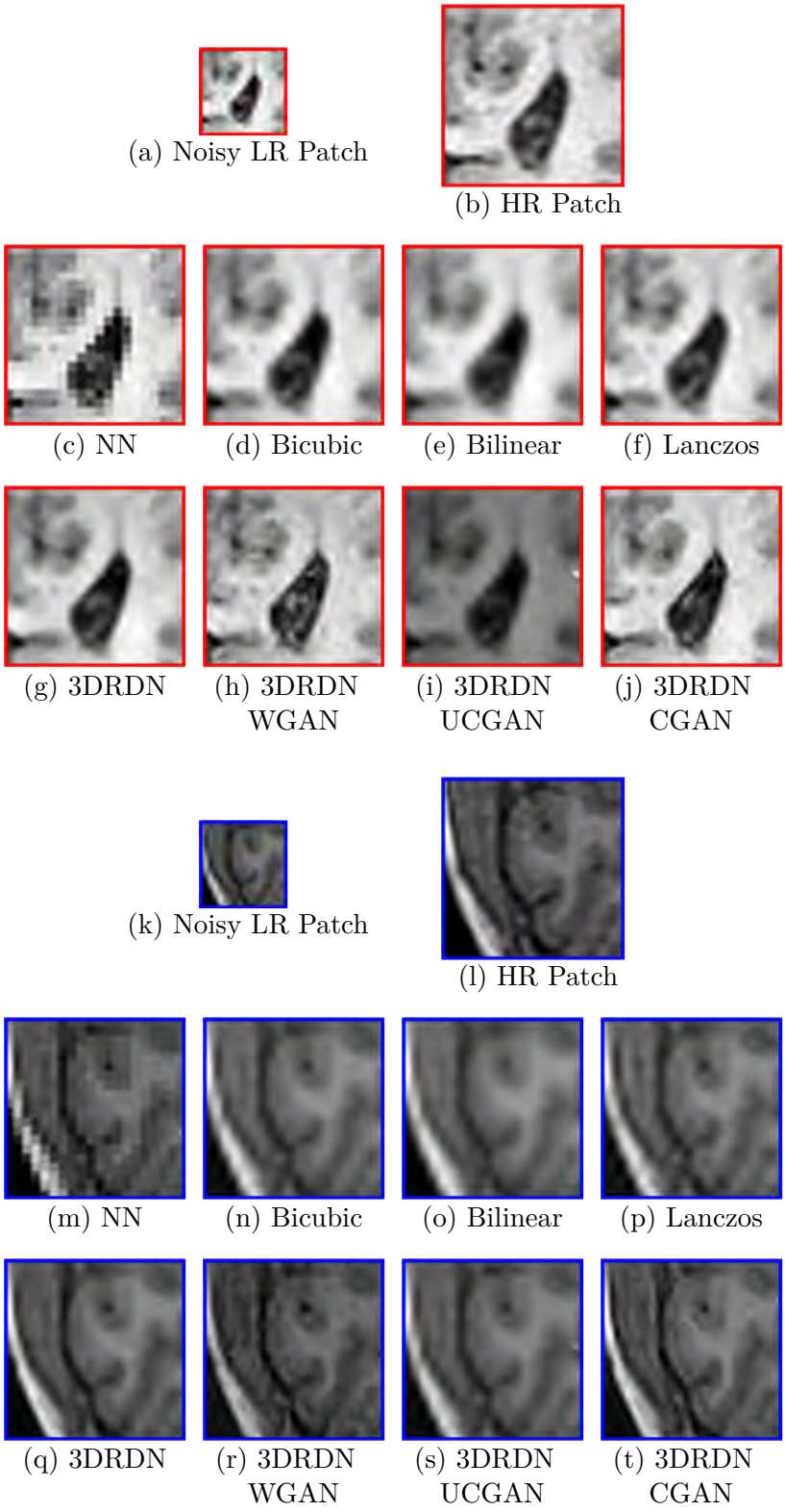
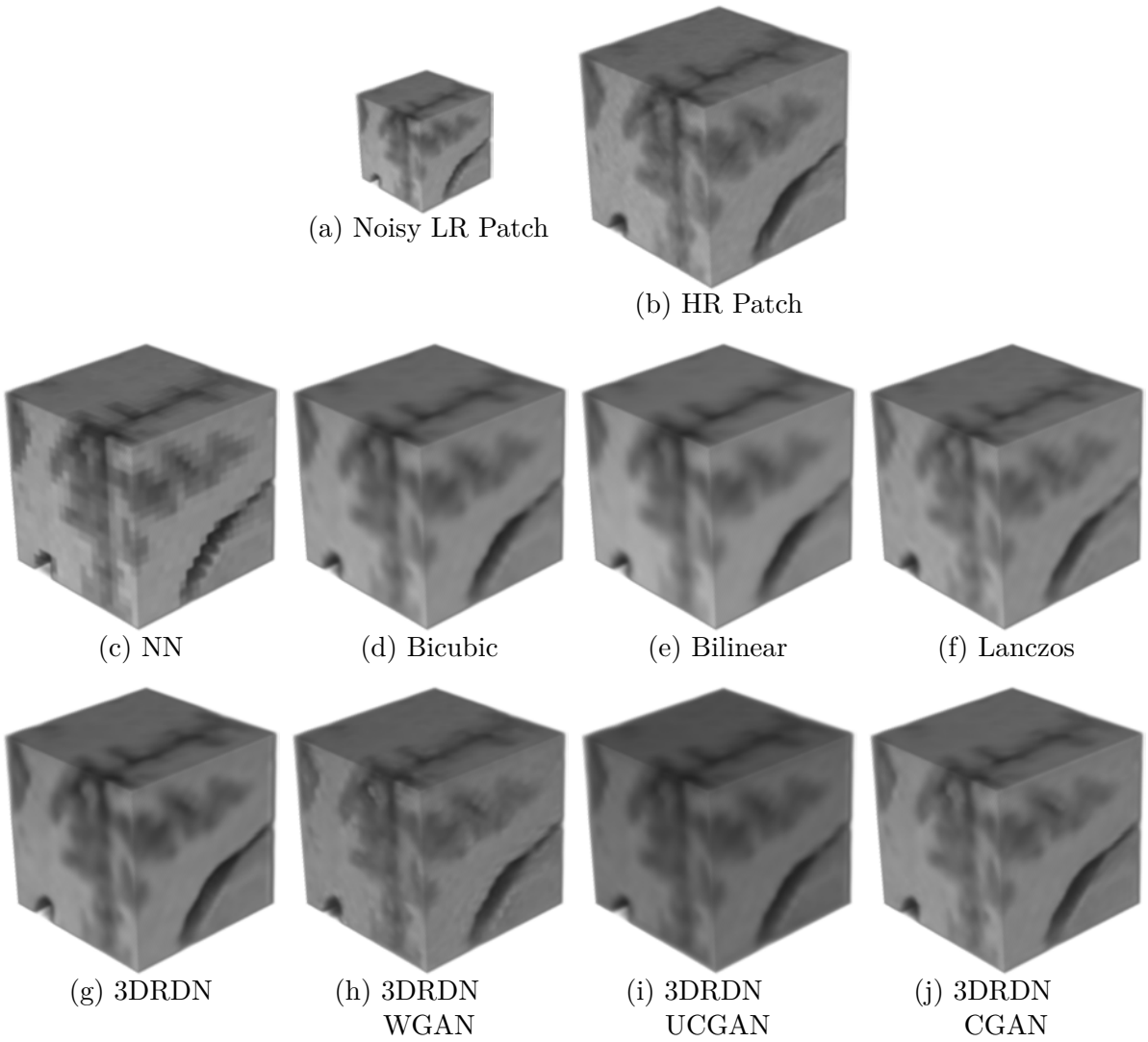


Figure 5.2: Detailed visual comparison of the red and blue patches from the x2 upscaled MRI scans in figure 5.1.

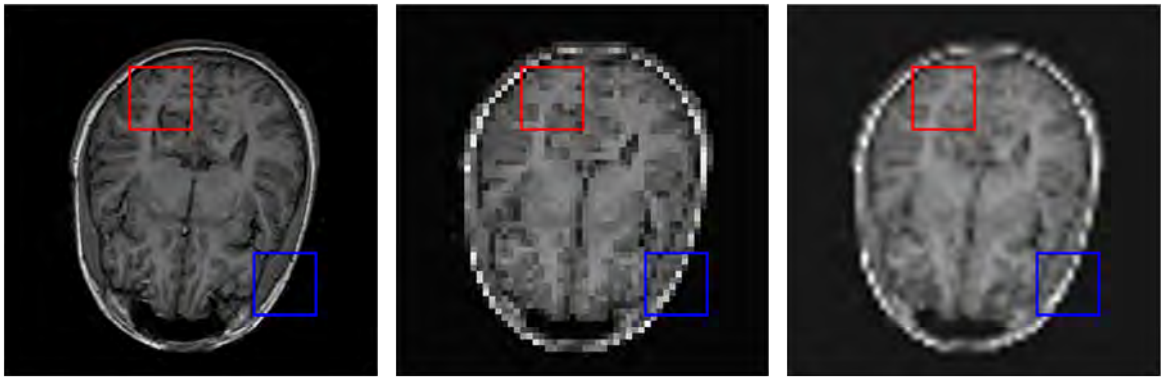


(a) Noisy LR Patch (b) HR Patch

(c) NN (d) Bicubic (e) Bilinear (f) Lanczos

(g) 3DRDN (h) 3DRDN WGAN (i) 3DRDN UCGAN (j) 3DRDN CGAN

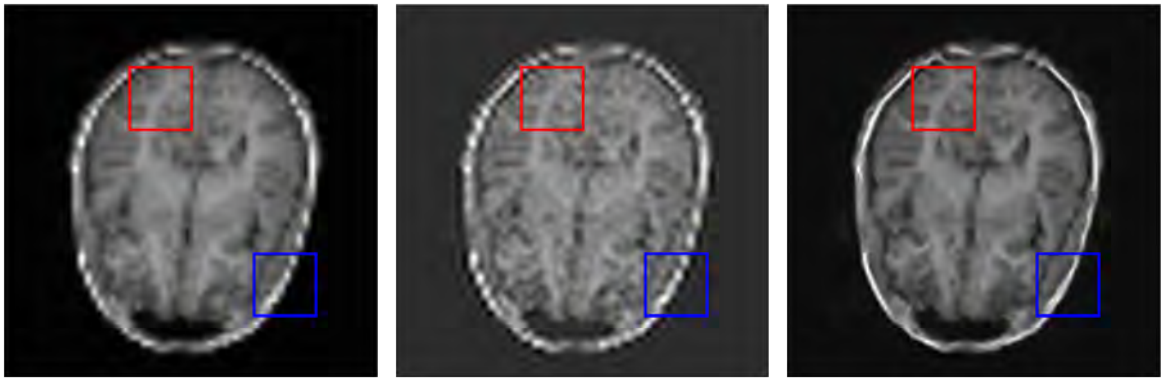
Figure 5.3: 3D visual representation of x2 upscaled patches from an MRI scan from the ABIDE dataset.



(a) Original HR Image
PSNR/SSIM

(b) Nearest Neighbour (NN)
17.824/0.156

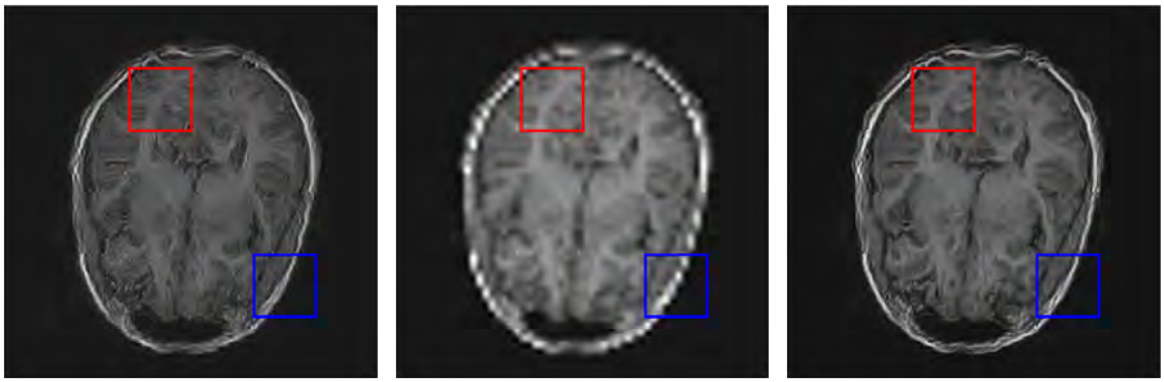
(c) Bicubic
17.438/0.186



(d) Bilinear
17.035/0.182

(e) Lanczos
14.026/0.156

(f) 3DRDN
26.380/0.746



(g) 3DRDN-WGAN
24.742/0.775

(h) 3DRDN-UCGAN
23.686/0.625

(i) 3DRDN-CGAN
25.654/0.710

Figure 5.4: Visual comparison of the (x4) upscaling performed on an entire MRI scan from the ABIDE dataset using the baseline interpolation methods and our models. Below each upscaled image are its metric scores that compare how close it is to the original HR image. The red and blue patches are further highlighted in figure 5.5 for more detailed comparison.

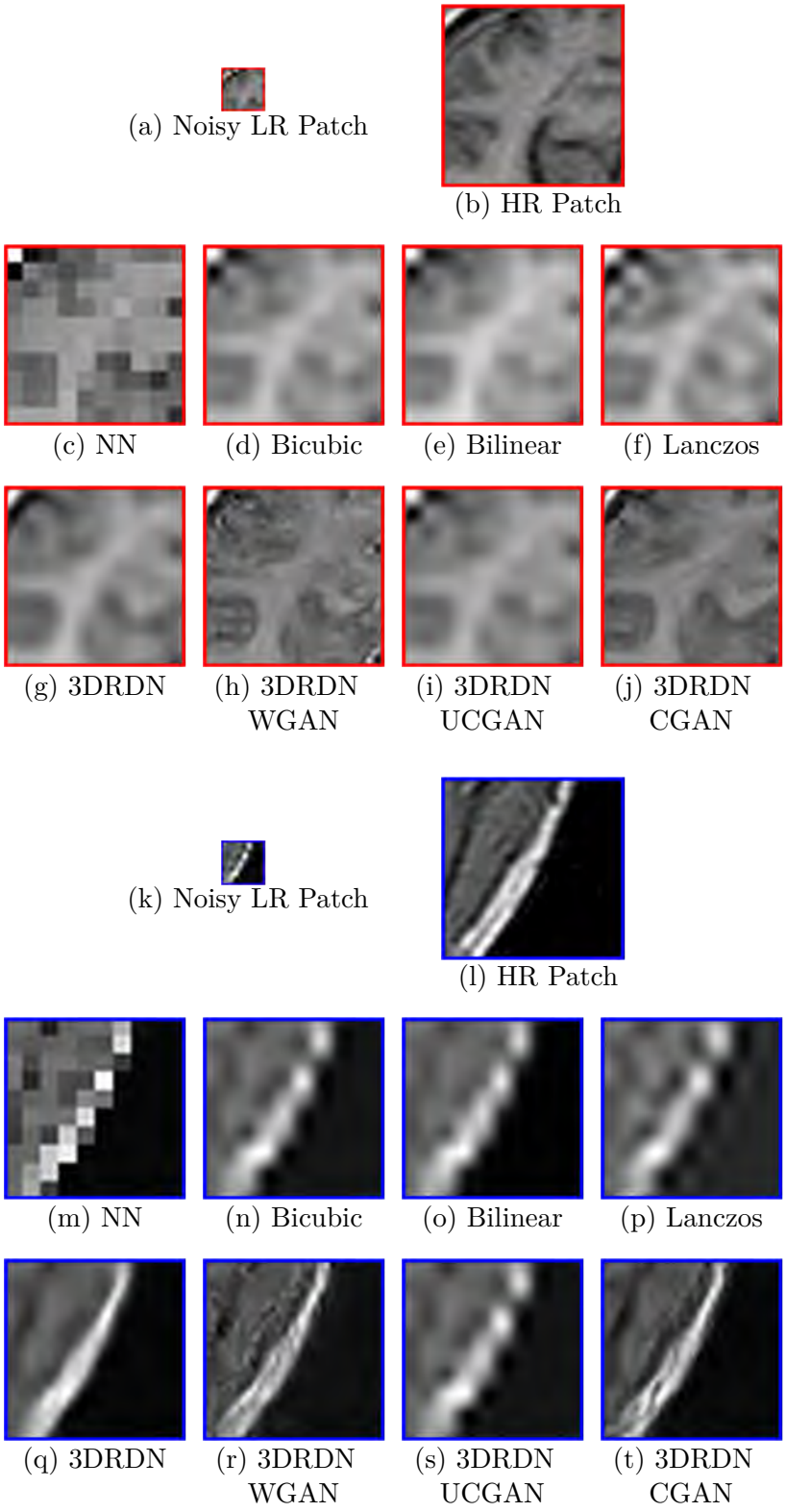


Figure 5.5: Detailed visual comparison of the red and blue patches from the x4 upscaled MRI scans in figure 5.4.

5.2 Mummy Dataset

For the mummy dataset, table 5.3 shows the evaluation of the x2 upscaling performed on its testing dataset by each method and model. As opposed to the previous dataset, it is the 3DRDN-CGAN model that performs best with the 3DRDN model coming second.

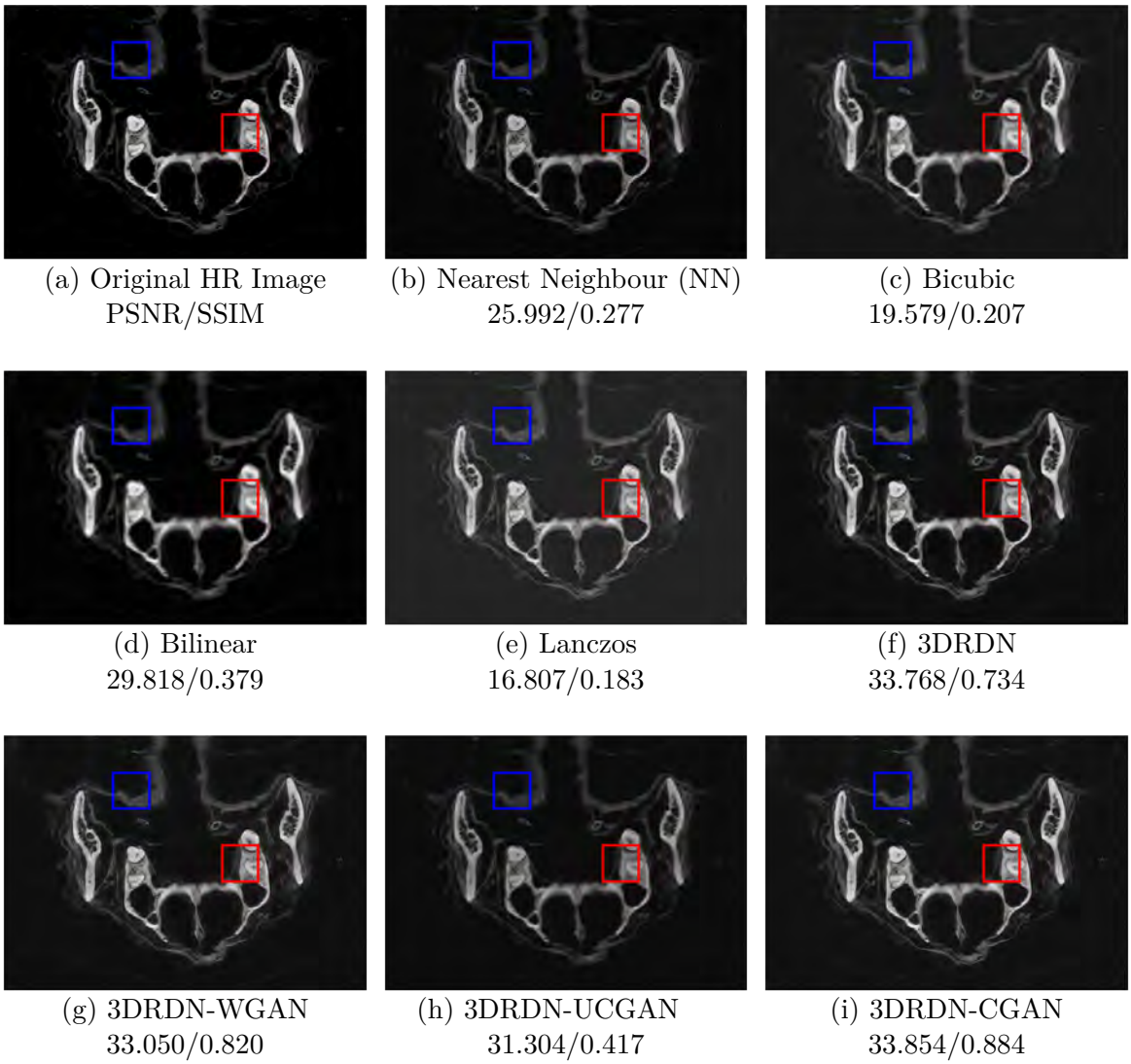
Quantitative Evaluation on the Mummy Dataset (x2 Scaling Factor)			
Method	PSNR †	SSIM †	MAE ‡
Nearest Neighbour	23.268 ± 3.744	0.160 ± 0.226	0.073 ± 0.030
Bicubic	19.517 ± 2.796	0.300 ± 0.291	0.110 ± 0.033
Bilinear	27.227 ± 6.035	0.323 ± 0.220	0.053 ± 0.041
Lanczos	15.347 ± 1.95	0.178 ± 0.216	0.174 ± 0.034
3DRDN	$\underline{40.505 \pm 5.669}$	$\underline{0.784 \pm 0.123}$	$\underline{0.008 \pm 0.004}$
3DRDN-WGAN	36.587 ± 4.597	0.743 ± 0.075	0.011 ± 0.005
3DRDN-UCGAN	37.415 ± 4.755	0.570 ± 0.262	0.013 ± 0.007
3DRDN-CGAN	$\mathbf{40.809 \pm 3.388}$	$\mathbf{0.891 \pm 0.049}$	$\mathbf{0.006 \pm 0.002}$

Table 5.3: Quantitative evaluation on x2 upscaling with the testing data from the Mummy dataset. † indicates that the higher the score the better while ‡ indicates that the lower the score the better. **Bold** indicates the best value while underline indicates the second best value.

In a similar manner, we demonstrate the visual results of the different upscaling methods on a CT scan sample from the Mummy dataset in figure 5.6. Two patches are also extracted from the upscaled images and shown in figure 5.7 for a more detailed comparison. Moreover, we showcase 3D comparison of snippets of x2 super-resolved images in figure 5.8. We also demonstrate more upscaled sample results from the Mummy dataset in appendix A.2.

In one of our experiments, we trained the basic 3DRDN model only on the ABIDE dataset and then evaluated its performance on the Mummy dataset. Our model was able to generalize nicely on the entirety of the completely unseen Mummy dataset achieving

a score of 33.649, 0.859 and 0.0172 for the PSNR, SSIM and MAE metrics respectively which surpassed all of the baseline interpolation methods.



(a) Original HR Image
PSNR/SSIM

(b) Nearest Neighbour (NN)
25.992/0.277

(c) Bicubic
19.579/0.207

(d) Bilinear
29.818/0.379

(e) Lanczos
16.807/0.183

(f) 3DRDN
33.768/0.734

(g) 3DRDN-WGAN
33.050/0.820

(h) 3DRDN-UCGAN
31.304/0.417

(i) 3DRDN-CGAN
33.854/0.884

Figure 5.6: Visual comparison of the (x2) upscaling performed on part of the CT scan from the Mummy dataset using the baseline interpolation methods and our models. Below each upscaled image are its metric scores that compare how close it is to the original HR image. The red and blue patches are further highlighted in figure 5.2 for more detailed comparison.

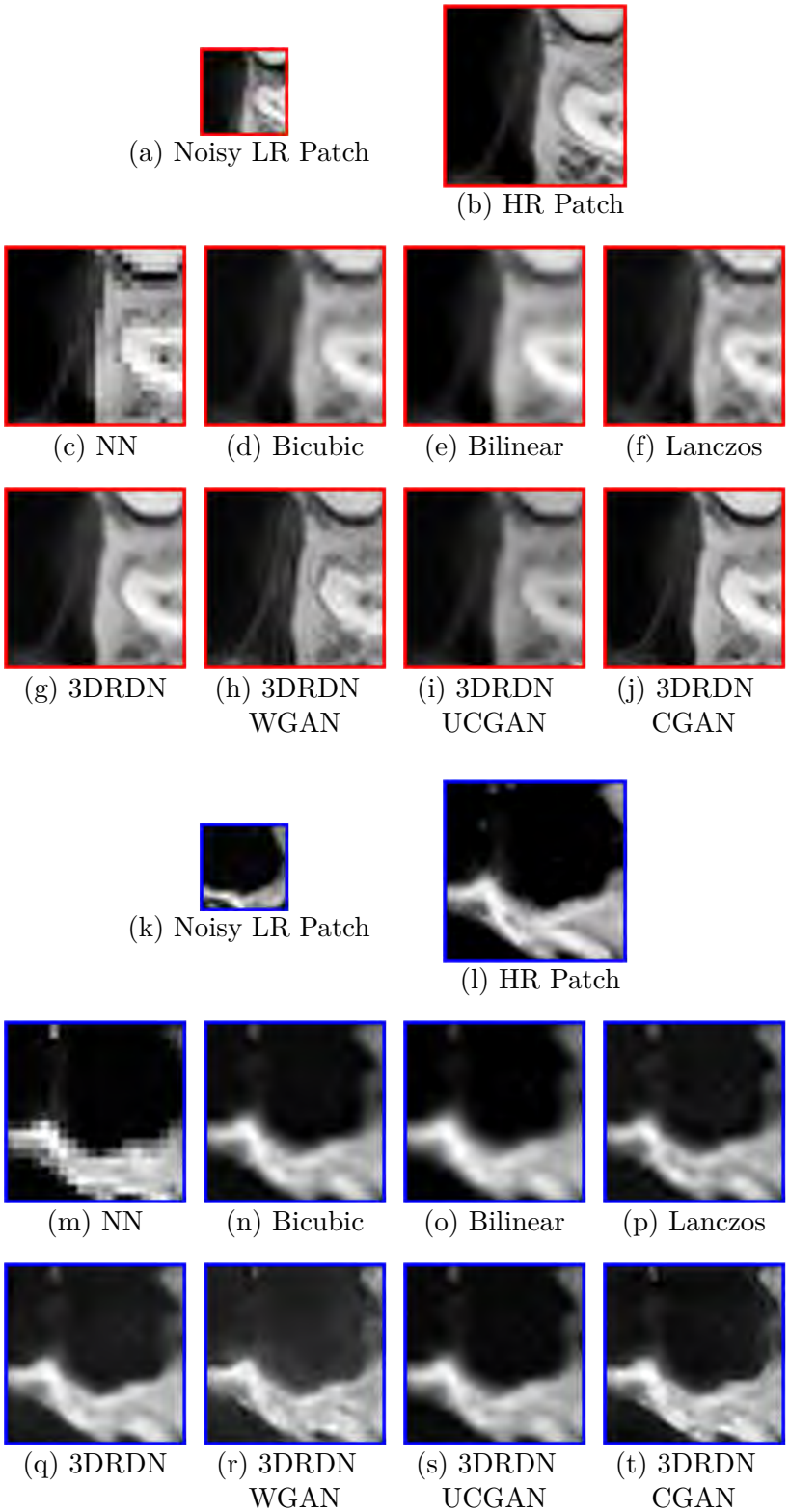
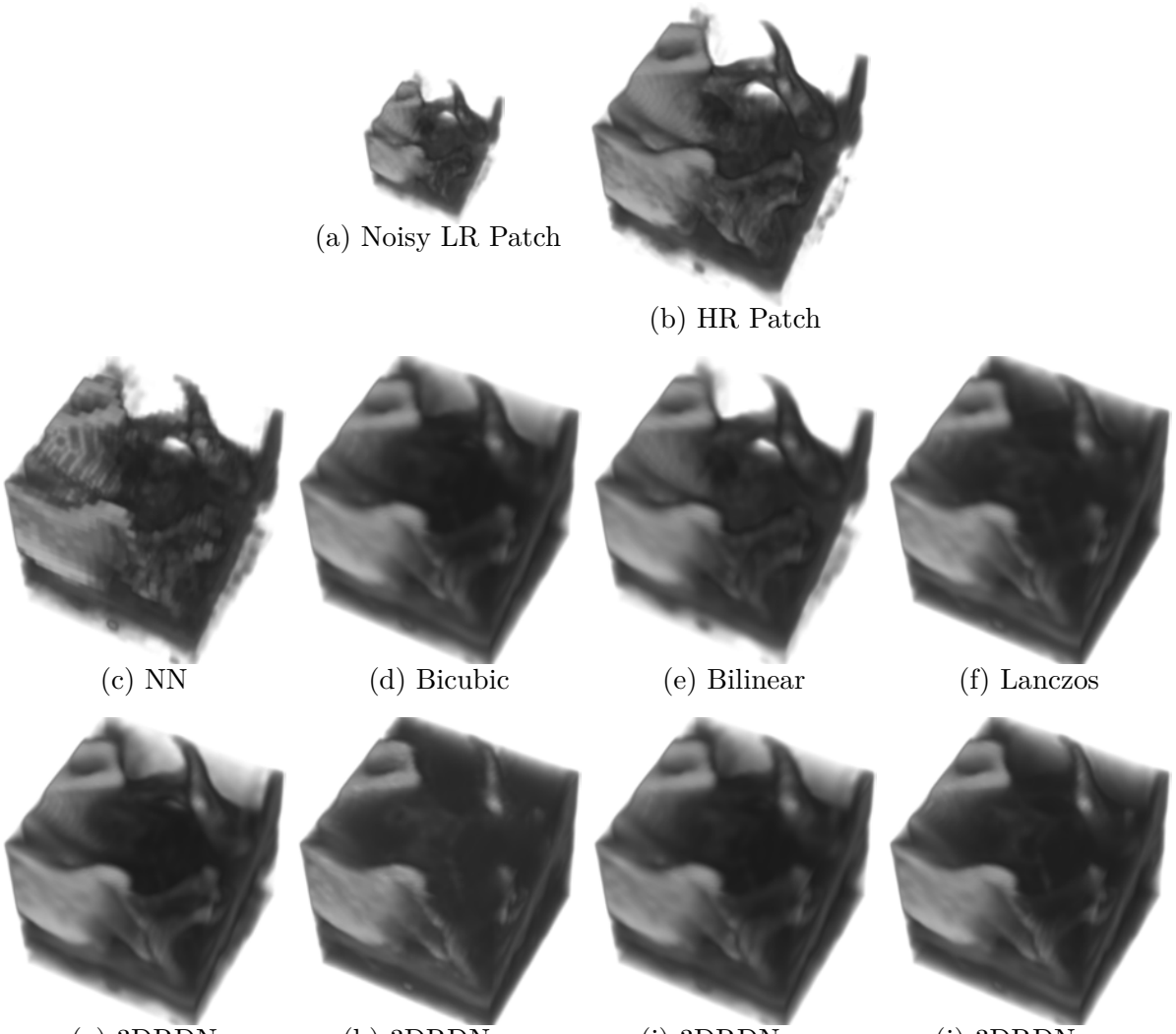


Figure 5.7: Detailed visual comparison of the red and blue patches from the x2 upscaled CT scans in figure 5.1.



(a) Noisy LR Patch (b) HR Patch

(c) NN (d) Bicubic (e) Bilinear (f) Lanczos

(g) 3DRDN (h) 3DRDN WGAN (i) 3DRDN UCGAN (j) 3DRDN CGAN

Figure 5.8: 3D visual representation of x2 upscaled patches from a CT scan from the Mummy dataset.

6 Discussion

The experiments prove that our models are significantly superior to all of the baseline interpolation methods in terms of quantitative evaluations as well as the visual quality of the upscaling achieved by them. As can be seen from the comparison figures, the upscaled output of our models is sharper and better at removing noise thanks to the better mapping they learned during training.

The first thing that our results indicate is that the evaluation metrics MAE, PSNR and even SSIM do not necessarily correlate with better visual output as they sometimes fail to convey human perception similarity. This is shown in various examples such as from figure 5.4 where the 3DRDN model obtained a higher score than 3DRDN-CGAN while outputting an image with noticeably less fidelity than the output of the 3DRDN-CGAN model.

6.1 Dataset Comparison

In our experiments, we aimed at testing our models on diverse data and therefore we selected two sets with one containing MRI scans (ABIDE) and the other containing CT scans (Mummy). The evaluation metrics (from tables 5.1 and 5.3) indicate that our

models perform significantly better on the Mummy dataset compared to the ABIDE set. However, the visual output on samples from the Mummy dataset is not observably better (closer to ground truth) in our subjective opinion. One reason that could explain the discrepancy is that the Mummy dataset contains substantially more empty (zero) voxels when compared to the ABIDE set and such empty voxels are easier to upscale meaning its easier to obtain high metric scores on scans with more of them. In our opinion, our models visually perform better on the ABIDE dataset and this was expected since there is less training data from the Mummy dataset which is made up of only 24 scans as opposed to 1087 scans for the ABIDE dataset.

6.2 Model Assessments

3DRDN is the simplest and easiest to train model which achieved the best performance in terms of evaluation scores on the ABIDE dataset. Its visual output, however, is not necessarily closer to the original HR image when compared to that of the WGAN or CGAN models. While it is fairly capable of sharpening edges, in some cases it is not able to accurately replicate the patterns of some areas such as tissue bodies and produces instead a more smoothed area when compared to the original HR images and the output of other GAN based models. This is due to its sole reliance on the MAE loss which results in the perceived smoothness. We believe however, that this is also what allows it to remove noise better than the other models which could explain why it achieved the best scores on the ABIDE dataset.

The 3DRDN-WGAN model on the other hand, while having lower metric scores, produces output that is visually more similar to the ground truth than 3DRDN due to its integration of the (perception-based) Wasserstein adversarial loss which helps it produce

images that mimic the probability distribution of the trained dataset. Its output looks more realistic and closer to that of the HR images when compared to the output of the basic 3DRDN model and it does not suffer from the over-smoothness problem of the 3DRDN model as it manages to reproduce tissue body patterns that are extremely similar to those of the original HR images. The downside is that it fails to properly suppress noise when compared to the 3DRDN model and as a result some artifacts sometimes appear in its upscaled output.

The 3DRDN-CGAN is the most complex of all our models, which integrates supervised, adversarial losses from the two previous models and also adds the cycle-consistency and identity loss functions to its objective loss function. This allowed it to perform significantly better than 3DRDN-WGAN as its optimization during adversarial training was further simplified thanks to the constraints from the two new loss components. Aside from having excellent metric evaluations, its visual output is also very similar to the ground truth like that of the 3DRDN-WGAN but with less noise and artifacts. Although its output is slightly more over-smoothed than 3DRDN-WGAN, it is not to the same extent as with the 3DRDN model. Overall in our opinion, 3DRDN-CGAN produces the best visual results.

3DRDN-UCGAN which relied on unpaired data and unsupervised learning performed considerably worse than the rest of the models both in terms of metric evaluations and very noticeably in terms of visual output quality. This can be attributed to its lack of any supervised loss component as it instead tries to learn the mapping between the overall probability distribution of the entire LR training set and the distribution of the entire HR set instead of between two paired images. Since this task is much harder, the model was harder to train and optimize its parameters and therefore performed poorly compared to the others.

6.3 Model Generalizability

The experiments also indicate that our models can generalize very well on unseen data which we guaranteed by using validation data during training to be constantly on the lookout in case overfitting occurs. Not only do the models generalize well on unseen data from the same dataset from which it was trained but the experiments show that even training on one dataset is enough for the models to also perform very well on a different dataset that has substantial differences in various aspects such as resolution, structure and quality.

This was demonstrated in one of the experiments where only the ABIDE dataset was used for training and it allowed the 3DRDN model to also perform almost just as well on the Mummy dataset. We attribute this to our utilization of very powerful data augmentation methods thanks to the 3D nature of the data which allowed us to perform 3D rotations and the fact that we operate on randomly extracted patches rather than the entire image during training which also counts as a form of data augmentation.

6.4 Upscaling with different factors

The experiments also prove that our architecture can be used without any modification for training the models on different upscaling factors. In one experiment, we trained the models for an upscaling factor of 4 on the ABIDE dataset. The results from table from table 5.2 indicate that the performance of our models substantially surpasses that of the baseline methods. From our models, 3DRDN-WGAN and 3DRDN-CGAN performed visually the best.

However, while the visual x4 upscaled output results of our models are also better than those of the baseline methods, the fidelity of the output of our models are not as good as with their output when upscaling by a factor of 2. This is natural as the LR images for training the models on x4 upscaling are extremely deteriorated during the preprocessing stages which makes it significantly more challenging for our models to upscale them back to be similar to the original HR images. We did not train our models on x4 upscaling for the Mummy dataset as it does not have enough data to properly optimize our models for such a challenging task.

6.5 Supervised versus Unsupervised Learning

Another interesting experiment that we were able to perform is that of the comparison between training using supervised versus unsupervised learning. In our setup, the 3DRDN, 3DRDN-WGAN, 3DRDN-CGAN represent supervised learning that requires paired data during training while 3DRDN-UCGAN represents unsupervised learning that can be trained using unpaired data. The results indicate that for super-resolution, supervised learning models perform substantially better on the super-resolution task. Therefore, since obtaining paired training data for the super-resolution task is not unfeasible, we do not recommend using the Cycle-GAN unsupervised method for super-resolution tasks.

6.6 Best Model for Super-Resolution

Lastly, we would like to provide our opinion on which of the models we believe to be the best for super resolution. The experiment results provide no clear cut superior model

as the 3DRDN model has the best metric evaluations on the ABIDE dataset and also visually minimizes noise the best while the 3DRDN-CGAN model has the best scores on the Mummy dataset and produces output that is most similar to the ground truth with minimum over-smoothness.

Accordingly, we believe that the medical or archaeological purpose of the scans, on which super-resolution is performed, must be taken into account when deciding which model is more suited. For example, if analyzing the patterns on the tissue bodies in the scan is a critical part of its medical analysis then the 3DRDN-CGAN or 3DRDN-WGAN models are more suited due to their accurate replication of the patterns of such tissues. On the other hand, these patterns might in some cases be noise in the image whose removal will ease its analysis and in such cases the 3DRDN model will produce the better results as it can remove noise much better. In our opinion, such decisions are best taken by domain experts such as doctors or radiologists.

7 Conclusion

In this thesis, we demonstrated our work in which we designed, developed and tested novel neural network models for the task of superresolution on 3D image scans.

Our basic model is unique in that it utilizes both Residual and DenseNet connections in its 3D based architecture. In our second model, we integrate it in a Wasserstein GAN architecture to obtain output that is visually more similar to the ground truth. In our third model, we further integrate it in a Cycle GAN architecture for visually similar output with even less noise. The source code of our work implementation is released at https://github.com/omagdy/3DRDN-CycleGAN.

To evaluate the capabilities of our models, we utilized two different datasets: one composed of MRI scans and the other CT scans and used the datatasets for training our models and comparing their performance with traditional super-resolution methods such as Bicubic and Bilinear interpolation. Both the quantitative scores and visual comparisons demonstrate the superiority of our models in the task of super-resolution on both MRI and CT scan data with the basic 3DRDN model obtaining the highest metric scores for the ABIDE dataset and the 3DRDN-CGAN model obtaining the best scores for the Mummy dataset.

This shows that our models can be utilized by radiologists, doctors and other medical workers for enhancing image scans in terms of both upscaling and denoising.

7.1 Future Work Recommendations

Despite the excellent performance of our models, there are various ways in which their performance could be further enhanced.

7.1.1 Data

First of all, using more CT scan data for training would greatly enhance the generalizability of our models on upscaling 3D image scans created by CT devices. Moreover, it would be even better to increase the variety of both MRI and CT data utilized for training by obtaining images from different scanning machines, with different spatial resolutions and different scanned body parts.

7.1.2 Preprocessing

In our work, to obtain LR image scans, we performed downscaling on HR images after adding Gaussian noise to them to simulate the noise and artifacts that can appear during CT/MRI scanning in real life. As suggested by [28], this might be too strict as in some cases our models might have to upscale LR images that do not have noise. Thus, it is preferable to modify the distribution of the Gaussian noise added throughout the training data such that it resembles a normal distribution. [19] suggested an even superior idea for MRI data which is performing their downscaling in the frequency domain by using

FFT to obtain decently looking LR images that better represent those obtained while MRI scanning.

This is especially important for upscaling 3D images with a factor of 4 as our experiments have shown that our current preprocessing methods deteriorate the images too much when obtaining LR versions of them for training.

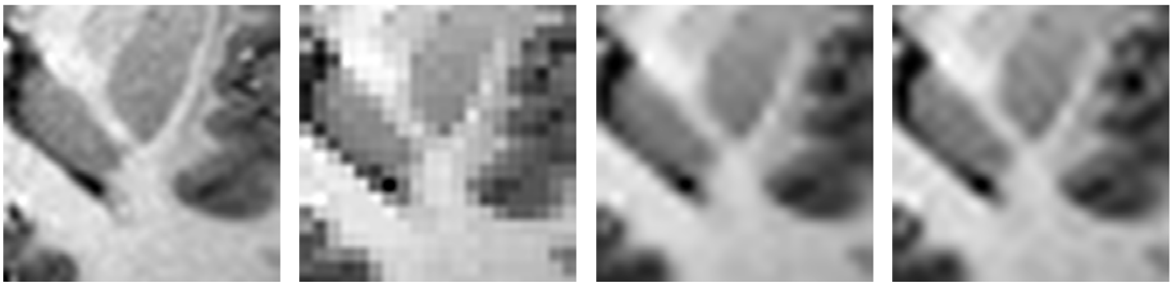
7.1.3 Objective Loss Function

Another enhancement, that has the potential to greatly improve the performance of our models, is to replace the MAE loss component of their objective loss functions with the more perception based loss of the Frechet Inception Distance (FID) which will make the supervised component of the objective function better at forcing the model to produce output that is visually even closer to the ground truth.

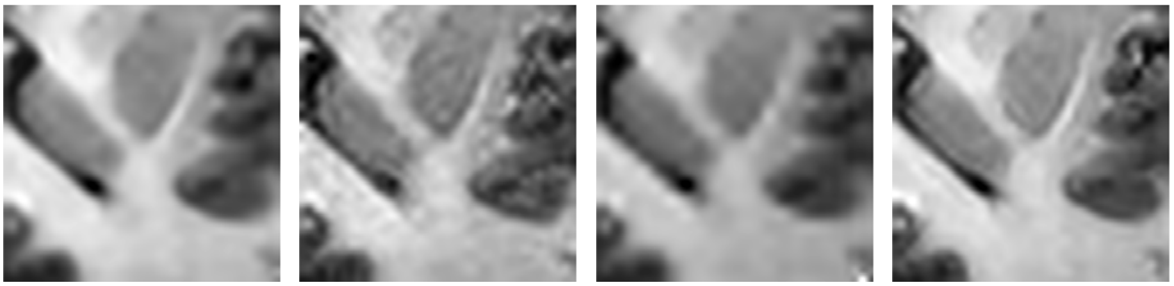
Lastly, even more experiments need to be performed with our models as they have an extensive list of hyper-parameters that need to be carefully explored to reach their optimum capabilities on the task of super-resolution. Also for x4 upscaling, more experiments are needed with deeper versions of our models that have even more dense units, dense blocks or growth rate k so that they can perform better even it makes them more challenging to train or require more time for upscaling during operation time on the 3D images.

A Additional Visual Results

A.1 ABIDE Dataset

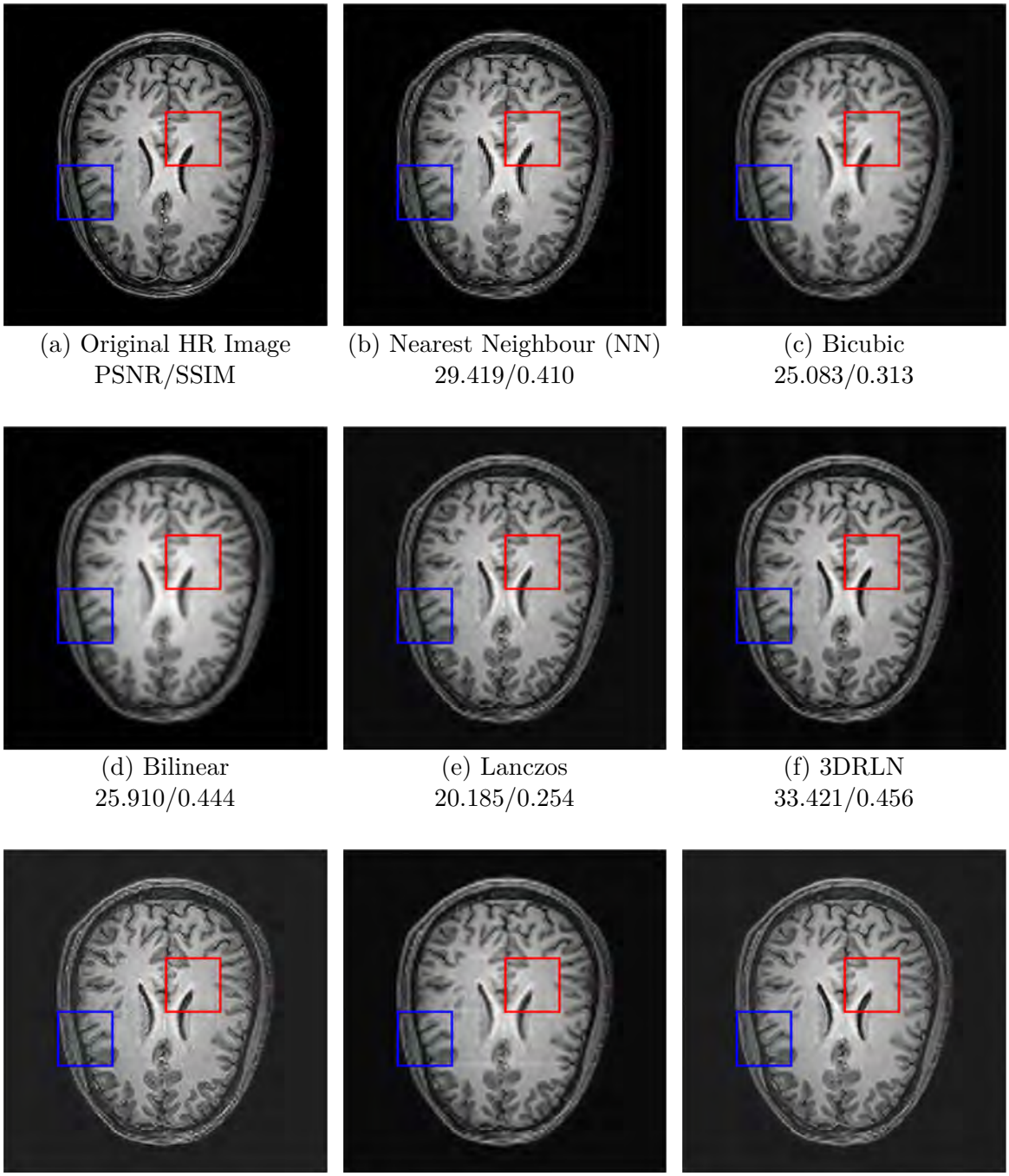


(a) HR Image PSNR/SSIM | (b) Nearest Neighbour 25.835/0.850 | (c) Bicubic 22.912/0.870 | (d) Lanczos 22.808/0.863



(e) 3DRDN 32.150/0.932 | (f) 3DRDN-WGAN 31.696/0.905 | (g) 3DRDN-UCGAN 31.779/0.916 | (h) 3DRDN-CGAN 31.939/0.922

Figure A.1: Visual comparison of patches of an MRI image, from the ABIDE dataset, that is upscaled with a factor of 2 using the baseline interpolation methods and our models. Below each upscaled patch are its metric scores that compare how close it is to the original HR image patch.



(a) Original HR Image
PSNR/SSIM

(b) Nearest Neighbour (NN)
29.419/0.410

(c) Bicubic
25.083/0.313

(d) Bilinear
25.910/0.444

(e) Lanczos
20.185/0.254

(f) 3DRLN
33.421/0.456

(g) 3DRLN-WGAN
34.111/0.653

(h) 3DRLN-UCGAN
29.347/0.305

(i) 3DRLN-CGAN
28.172/0.280

Figure A.2: Visual comparison of the (x2) upscaling performed on an entire MRI scan from the ABIDE dataset using the baseline interpolation methods and our models. Below each upscaled image are its metric scores that compare how close it is to the original HR image. The red and blue patches are further highlighted in figure A.3 for a more detailed comparison.

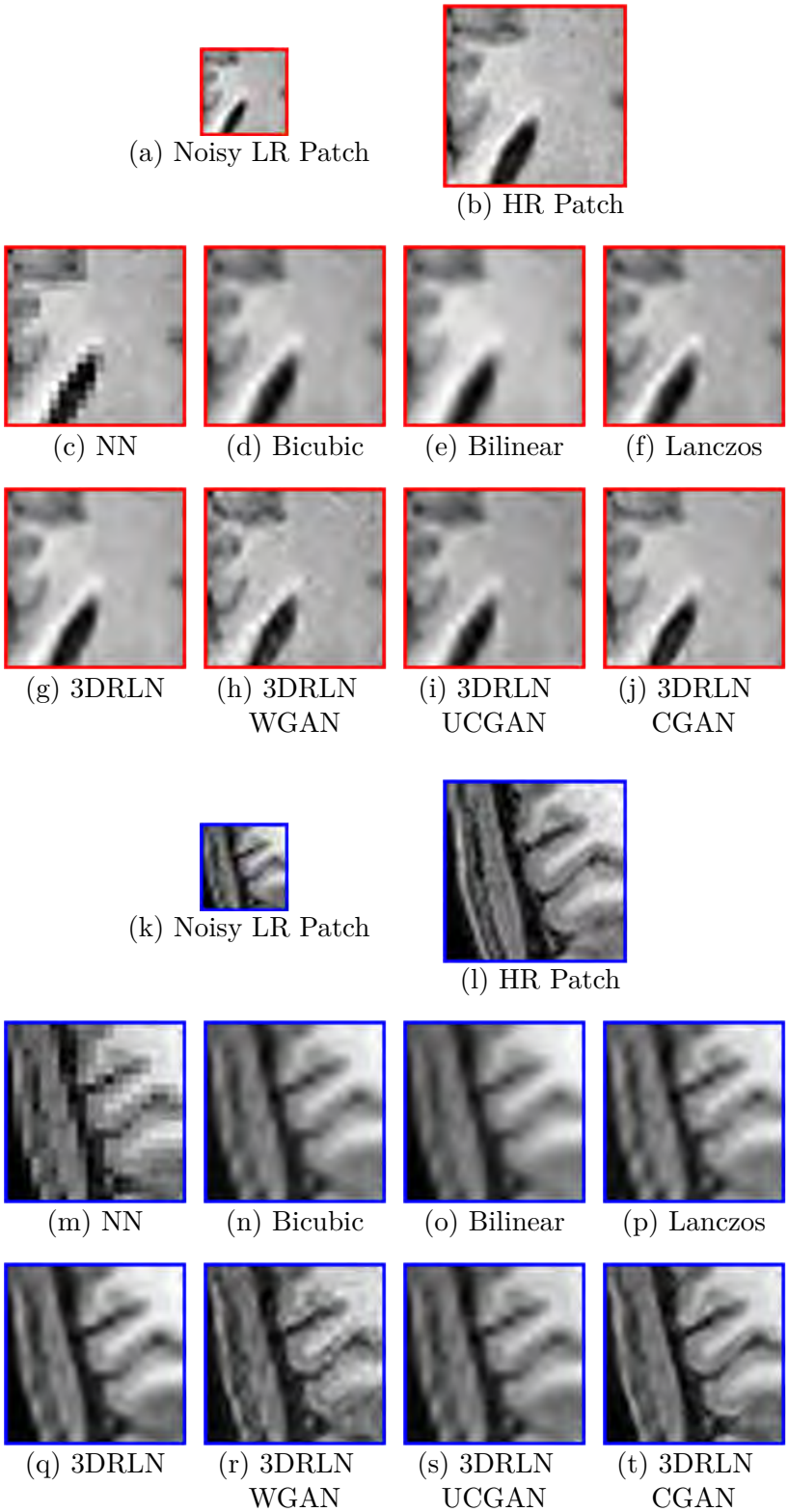
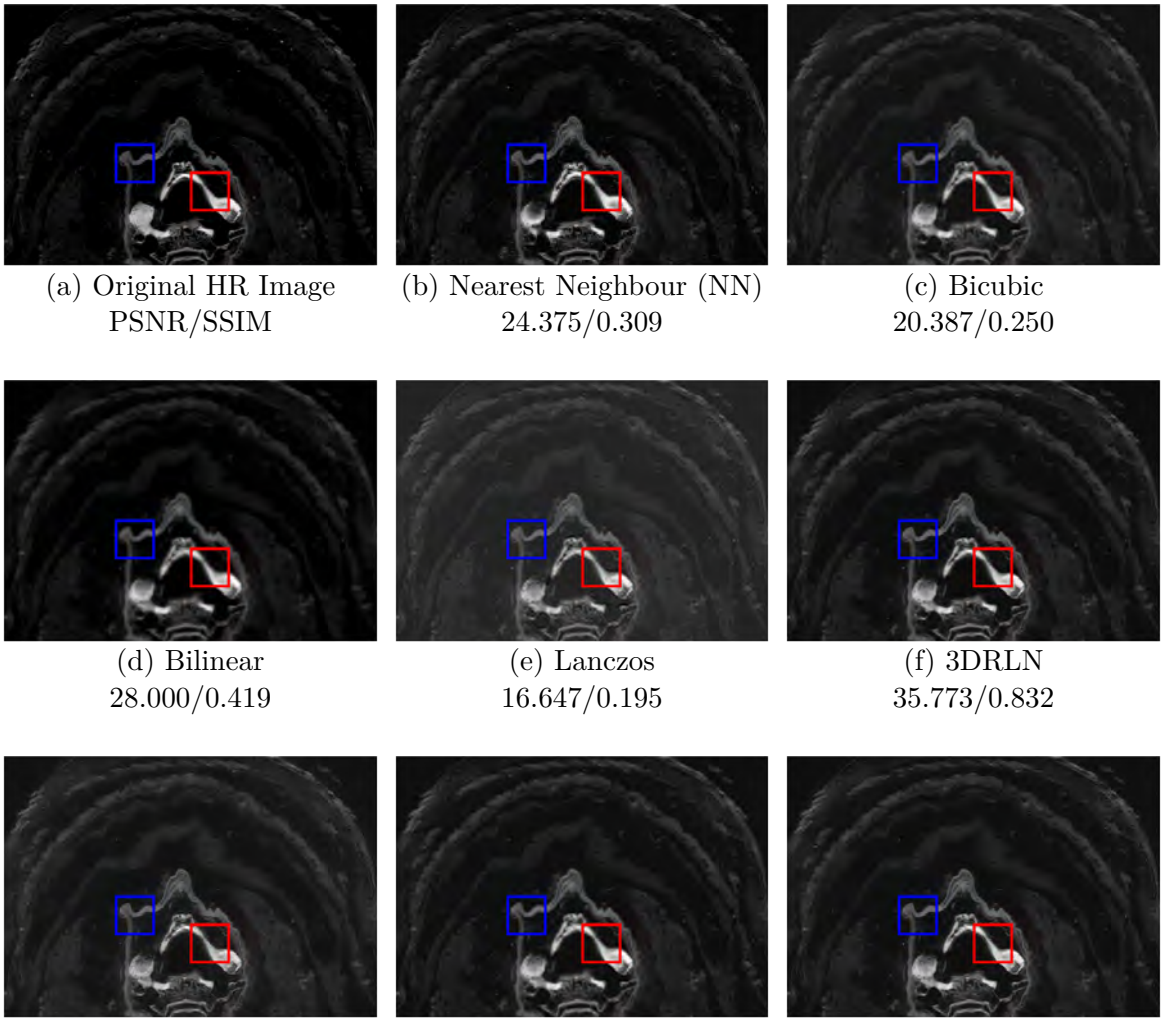


Figure A.3: Detailed visual comparison of the red and blue patches from the x2 upscaled MRI scans in figure A.2

A.2 Mummy Dataset



(a) Original HR Image
PSNR/SSIM

(b) Nearest Neighbour (NN)
24.375/0.309

(c) Bicubic
20.387/0.250

(d) Bilinear
28.000/0.419

(e) Lanczos
16.647/0.195

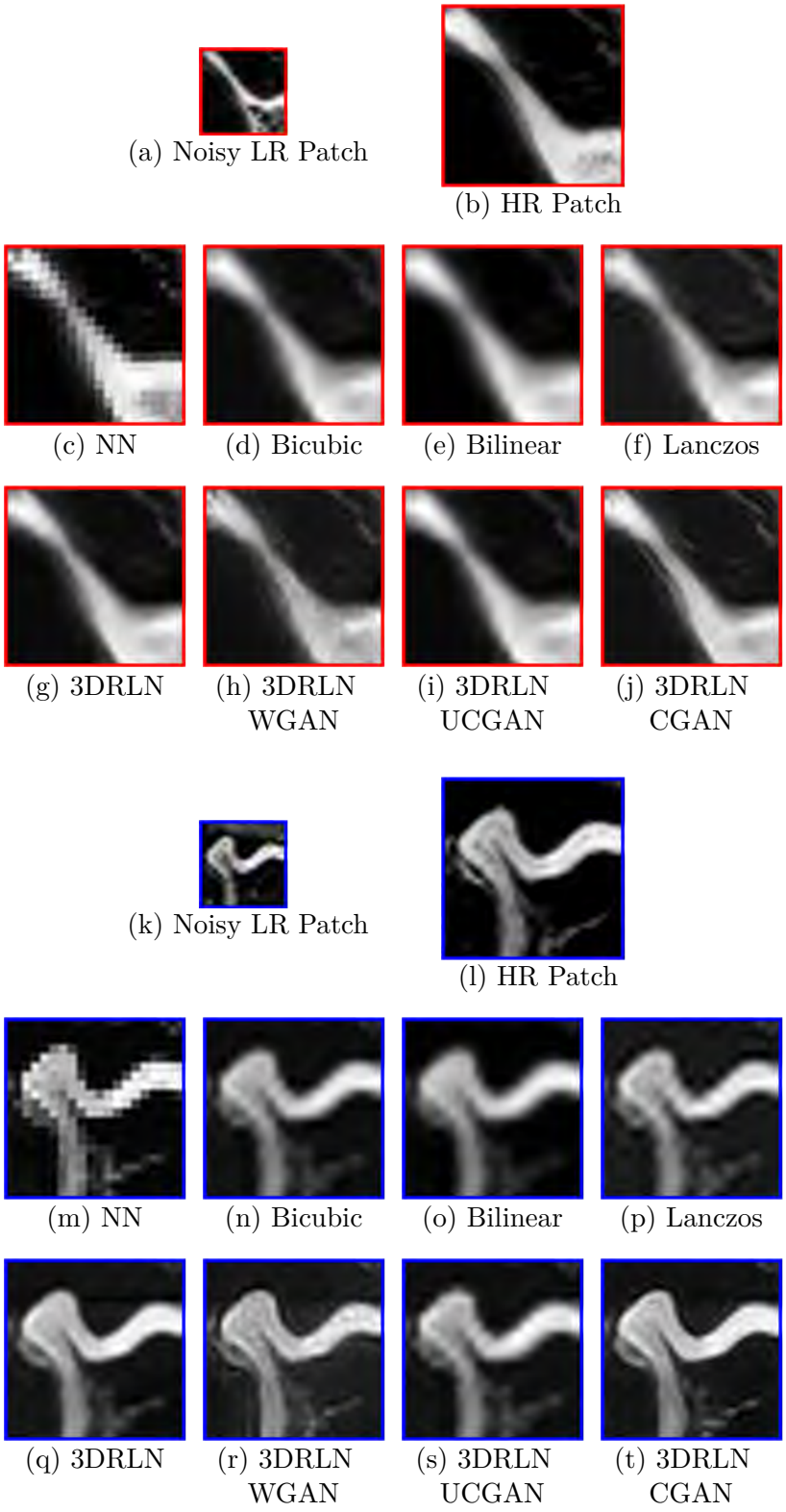
(f) 3DRLN
35.773/0.832

(g) 3DRLN-WGAN
33.971/0.773

(h) 3DRLN-UCGAN
33.901/0.694

(i) 3DRLN-CGAN
35.573/0.872

Figure A.4: Visual comparison of the (x2) upscaling performed on part of the CT scan from the Mummy dataset using the baseline interpolation methods and our models. Below each upscaled image are its metric scores that compare how close it is to the original HR image. The red and blue patches are further highlighted in figure A.5 for a more detailed comparison.



(a) Noisy LR Patch (b) HR Patch

(c) NN (d) Bicubic (e) Bilinear (f) Lanczos

(g) 3DRLN (h) 3DRLN WGAN (i) 3DRLN UCGAN (j) 3DRLN CGAN

(k) Noisy LR Patch (l) HR Patch

(m) NN (n) Bicubic (o) Bilinear (p) Lanczos

(q) 3DRLN (r) 3DRLN WGAN (s) 3DRLN UCGAN (t) 3DRLN CGAN

Figure A.5: Detailed visual comparison of the red and blue patches from the x2 upscaled CT scans in figure A.4

List of Figures

List of Tables

Bibliography

[1] Charles Patrick Davis. *CT Scan vs. MRI Differences between Machines, Costs, Uses*. 2019. URL: https://www.medicinenet.com/ct_scan_vs_mri/article.htm.

[2] Thomas Angus, Imperial College London. *fMRI*. 2019. URL: https://commons.wikimedia.org/wiki/File:190603_Functional_magnetic_resonance_imaging_at_the_Imperial_Centre_for_Psychedelic_Research.jpg.

[3] Nevit Dilmen. *NPH MRI 032*. 2005. URL: https://commons.wikimedia.org/wiki/File:NPH_MRI_032.png.

[4] Ranveig. *MRI head side*. 2005. URL: https://commons.wikimedia.org/wiki/File:MRI_head_side.jpg.

[5] Stephen Hughes. "CT scanning in archaeology." In: *Computed Tomography-Special Applications/Ed. L. Saba. InTech Europe* (2011), pp. 57–70.

[6] U.S. Navy photo by Mass Communication Specialist 3rd Class Samantha A. Lewis. *Peruvian child mummy CT scanning*. 2011. URL: https://commons.wikimedia.org/wiki/File:US_Navy_110427-N-2531L-135_Tori_Randall,_Ph.D._prepares_a_550-year_old_Peruvian_child_mummy_for_a_CT_scan.jpg.

[7] U.S. Navy photo. *Peruvian child mummy CT scan*. 2005. URL: https://commons.wikimedia.org/wiki/File:US_Navy_110427-N-YY999-001_A_CT_scan_of_a_Peruvian_mummy_taken_at_Naval_Medical_Center_San_Diego_provides_details_of_the_muscular_and_skeletal_stru.jpg.

[8] Yvette Brazier. *How does a CT or CAT scan work?* 2018. URL: https://www.medicalnewstoday.com/articles/153201.

[9] Yukai Wang et al. "CT-image of rock samples super resolution using 3D convolutional neural network." In: *Computers & Geosciences* 133 (2019), p. 104314.

[10] Chao Dong et al. "Image super-resolution using deep convolutional networks." In: *IEEE transactions on pattern analysis and machine intelligence* 38.2 (2015), pp. 295–307.

[11] Christian Ledig et al. "Photo-realistic single image super-resolution using a generative adversarial network." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4681–4690.

[12] Umme Sara, Morium Akter, and Mohammad Shorif Uddin. "Image quality assessment through FSIM, SSIM, MSE and PSNR—a comparative study." In: *Journal of Computer and Communications* 7.3 (2019), pp. 8–18.

[13] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity." In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[14] Hamid R Sheikh, Alan C Bovik, and Gustavo De Veciana. "An information fidelity criterion for image quality assessment using natural scene statistics." In: *IEEE Transactions on image processing* 14.12 (2005), pp. 2117–2128.

[15] Chih-Yuan Yang, Chao Ma, and Ming-Hsuan Yang. "Single-image super-resolution: A benchmark." In: *European conference on computer vision*. Springer. 2014, pp. 372–386.

[16] Chao Dong, Chen Change Loy, and Xiaoou Tang. "Accelerating the super-resolution convolutional neural network." In: *European conference on computer vision*. Springer. 2016, pp. 391–407.

[17] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. "Accurate image super-resolution using very deep convolutional networks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1646–1654.

[18] Chenyu You et al. "CT Super-resolution GAN Constrained by the Identical, Residual, and Cycle Learning Ensemble (GAN-CIRCLE)." In: *IEEE Transactions on Medical Imaging* PP (June 2019), pp. 1–1. DOI: 10.1109/TMI.2019.2922960.

[19] Yuhua Chen et al. "Brain MRI super resolution using 3D deep densely connected neural networks." In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE. 2018, pp. 739–742.

[20] Ismail Mebsout. *Convolutional Computation Example*. 2020. URL: https://miro.medium.com/max/875/1*CKFWAxDyPKjJBzzQTKVqYw.png.

[21] Chi-Hieu Pham et al. "Brain MRI super-resolution using deep 3D convolutional networks." In: *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*. IEEE. 2017, pp. 197–200.

[22] Yuhua Chen et al. "Efficient and accurate MRI super-resolution using a generative adversarial network and 3D multi-level densely connected network." In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 91–99.

[23] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.

[24] Ishaan Gulrajani et al. "Improved training of wasserstein gans." In: *arXiv preprint arXiv:1704.00028* (2017).

[25] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

[26] Gao Huang et al. "Densely connected convolutional networks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[27] Kaiming He et al. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[28] Mariana-Iuliana Georgescu, Radu Tudor Ionescu, and Nicolae Verga. "Convolutional Neural Networks with Intermediate Loss for 3D Super-Resolution of CT and MRI Scans." In: *IEEE Access* 8 (2020), pp. 49112–49124.