Semantically Correct High-resolution CT Image Interpolation and its Application

Jiawei Li

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master of Applied Science in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Electrical Engineering and Computer Science
University of Ottawa

Abstract

Image interpolation in the medical area is of vital importance as most 3D biomedical volume images are sampled where the distance between consecutive slices is significantly greater than the in-plane pixel size due to radiation dose or scanning time. Image interpolation creates a certain number of new slices between known slices in order to obtain an isotropic volume image. The results can be used for the higher quality of 2D and 3D visualization or reconstruction of human body structure.

Semantic interpolation on the manifold has been proved to be very useful for smoothing the interpolation process. Nevertheless, all previous methods focused on low-resolution image interpolation, and most of which work poorly on high-resolution images. Besides, the medical field puts a high threshold for the quality of interpolations, as they need to be semantic and realistic enough, and resemble real data with only small errors permitted.

Typically, people downsample the images into 32^2 and 64^2 for semantic interpolation, which does not meet the requirement for high-resolution in the medical field. Thus, we explore a novel way to generate semantically correct interpolations and maintain the resolution at the same time. Our method has been proved to generate realistic and high-resolution interpolations on the sizes of 526^2 and 512^2.

Our main contribution is, first, we propose a novel network, High Resolution Interpolation Network (HRINet)**,** aiming at producing semantically correct high-resolution CT image interpolations. Second, by combining the idea of ACAI and GANs, we propose a unique alternative supervision method by applying supervised and unsupervised training alternatively to raise the accuracy and fidelity of body structure in CT when interpolated while keeping high quality. Third, we introduce an extra Markovian discriminator as a texture or fine details regularizer to make our model generate results indistinguishable from real data. In addition, we explore other possibilities or tricks to further improve the performance of our model, including low-level feature maps mixing, and removing batch normalization layers within the autoencoder. Moreover, we compare the impacts of MSE based and perceptual based loss optimizing methods for high quality interpolation, and show the trade-off between the structural correctness and sharpness.

The interpolation experiments show significant improvement on both sizes of 256^2 and 512^2 images quantitatively and qualitatively. We find that interpolations produced by HRINet are sharper and more realistic compared with other existing methods such as AE and ACAI in terms of various metrics.

As an application of high-resolution interpolation, we have done 2D volume projection and 3D volume reconstruction from axial view CT data and their interpolations. We show the great enhancement of applying HRINet for both in sharpness and fidelity. Specifically, for 2D volume projection, we explore orthogonal projection and weighted projection respectively so as to show the improved effectiveness for visualizing internal and external human body structure.

Acknowledgement

First of all, I would like to express my sincere gratitude to my supervisor WonSook Lee for consistent support and guidance throughout my master research. She provides me such a precious opportunity to be a member of her research group which benefits me a lot in both study and life. I also appreciate her for offering me thoughtful comments and recommendations on this dissertation. From the selection of the subject to the completion of it, she has always given me patient guidance. Without her, I would not have progressed so well.

I want to give thanks for my colleagues in our lab who offer me great help and encouragement. It is my honor to spend time with all of you on studying and discussing. The weekly seminars and coffee break not only enhance my academic knowledge but also promote the relationship with each other. It truly has been an excellent time in this lab.

I am also thankful to the university and all its member's staff for all the considerate guidance. The university provides me with an excellent environment and advanced equipment for studying and doing research. The selfless help makes me feel the warmth of the campus.

I appreciate my classmates and friends, for their care and help for my study and life in the past two years, thank my friends for their support, and finally show my heartfelt thanks to my parents for their support and understanding.

Table of Contents

Abstract ..... **ii**
Acknowledgement ..... **iii**
Table of Contents ..... **iv**
List of Figures ..... **vi**
List of Tables ..... **ix**
Chapter 1. Introduction **1**
1.1 Motivation 1
1.2 Overview of Our System 3
1.3 Thesis Organization 5
1.4 Contributions 6
1.4.1 Publications 6
Chapter 2. Literature Review **7**
2.1 Deep Learning and Related Knowledge 7
2.1.1 Application in Medical Field 10
2.2 Generative Adversarial Networks 11
2.2.1 Principle 11
2.2.2 GANs for Image Super-Resolution 13
2.2.3 WGAN, WAN-GP and other GANs Derivatives 14
2.3 Autoencoders 18
2.3.1 AE and VAE 18
2.4 Semantic Interpolation 20
2.4.1 Background Knowledge 20
2.4.2 Autoencoders and Interpolation 22
2.4.3 Adversarially Constrained Autoencoder Interpolation 22
2.5 Image Quality Assessment 23
2.5.1 Peak Signal-to-Noise Ratio 23
2.5.2 Structural Similarity Index 24
2.5.3 No-reference Ma 24
2.5.4 Natural Image Quality Evaluator 25
Chapter 3. Semantically Correct High-resolution Interpolation **26**

3.1 High-resolution Interpolation Network 26
3.1.1 Network Architecture 27
3.1.2 Low-level Feature Maps Mixing 28
3.1.3 PatchGAN Regularizer 29
3.2 Alternatively Supervising Training 31
3.2.1 Unsupervised Training Step (Step I) 31
3.2.2 Supervised Training Step (Step II) 32
3.2.3 Content Loss Function 35
3.3 Experiments 36
3.3.1 Dataset Description 36
3.3.2 Training Details and Parameters 37
Chapter 4. Evaluation 43
4.1 Large Gaps Interpolation 43
4.2 Adjacent Slices Interpolation 48
4.3 Feature Reconstruction and Representation Learning 50
4.4 Ablation Study 52
4.4.1 Effects of Main Components 52
4.4.2 Receptive Field Size 54
Chapter 5. Application 56
5.1 2D Volume Projection 56
5.1.1 Orthogonal Projection 56
5.1.2 Weighted Projection 58
5.2 3D Volume Reconstruction 60
Chapter 6. Conclusion 63
6.1 Conclusion 63
6.2 Future work 65
References 66
Publication by the Author 71

List of Figures

Figure 1-1: Example of overlapping and blurry problems. (a) The interpolated result is sharp but overlapped, close to interpolating in data space rather than latent space. (b) Interpolating in latent space, but the result is blurry and missing fine details. (c) "Appropriate" interpolations on the manifold. There are many possible paths like (1) (2) and (3) connect a pair of endpoints, but only (2) is considered semantically correct. 2

Figure 1-2: Overview of our system. 3

Figure 1-3: The application of our system. 4

Figure 2-1: The general architecture of GANs, from [17]. 12

Figure 2-2: The architecture of generator and discriminator in SRGAN, reprinted from [20]. 14

Figure 2-3: Value surfaces of WGAN critics trained to optimality on toy datasets using (top) weight clipping and (bottom) gradient penalty, reprinted from [33]. Critics trained with weight clipping fail to capture higher moments of the data distribution. The generator is held fixed at the real data plus Gaussian noise. 16

Figure 2-4: Gradient norms of deep WGAN critics during training on the Swiss Roll dataset either explode or vanish when using weight clipping, but not when using a gradient penalty (right), weight clipping (top) pushes weights towards two values (the extremes of the clipping range), unlike gradient penalty (bottom), reprinted from [33]. 16

Figure 2-5: Manipulating latent codes on MNIST, reprinted from [44]. (a) Varying c_1 on InfoGAN (Digit type) (b) Varying $c1$ on regular GAN (No clear meaning). (c) Varying c_2 from -2 to 2 on InfoGAN (Rotation). (d) Varying c_3 from -2 to 2 on InfoGAN (Width). ...17

Figure 2-6: The general architecture of autoencoders. 19

Figure 2-7: The principle of VAE, from [40]. Red solid lines denote the generative distribution $p_\theta(.)$ and blue dashed lines denote the distribution $q_\phi(z|x)$ to approximate the intractable posterior $p_\theta(z|x)$ by minimizing the KL divergence between them. 20

Figure 2-8: Examples of interpolations for synthetic lines dataset, reprinted from [44]. (a) Random samples from the dataset. (b) A perfect interpolation from $11/14\pi$ to 0 (ground truth). (c) Interpolating in data space rather than latent space. (d) An interpolation but with abruptly changes from one image to the other. (e) A smooth interpolation which takes a longer path from the start to end point than necessary. (f) An interpolation which takes the correct path but where intermediate points are not realistic. 21

Figure 2-9: Adversarially Constrained Autoencoder Interpolation (ACAI), reprinted from [44]. A critic network tries to predict the interpolation coefficient α corresponding to an interpolated datapoint. The autoencoder is trained to fool the critic into outputting $\alpha = 0$. 22

Figure 2-10: Main steps of no-reference metric Ma [52]. 24

Figure 3-1: The overview of our model. An encoder is trained to produce low-level feature maps which are interpolated with interpolation coefficient α, and the decoder tries to disentangle the latent representation and mapping it into data space. D_1 is the discriminator tries to predict α, and D_2 is trained to distinguish whether the input is real or not. The overall autoencoder can be regarded as Generator, trained to making the D_1 output $\alpha = 0$ consistently. 27

Figure 3-2: Network architecture of the autoencoder part. "EN" or "DE" stands for encoder or decoder block. The low-level feature maps mixing is done beyond the autoencoder part, and α is interpolation coefficient. Bottleneck is where the numbers of channels getting decreased dramatically. 29

Figure 3-3: The principle of PatchGAN. Each patch is mapped into a single prediction of the output vector of D_2. 30

Figure 3-4: The comparison of PixelGAN, PatchGAN and ImageGAN. 30

Figure 3-5: The principle optimization objective in Supervised training step (Step II). For a pair of inputs with large gap, the content loss $\mathcal{L}_{\text{cont}}$ is calculated between their interpolations and corresponding ground truth. 34

Figure 3-6: Samples from CT slices dataset. 37

Figure 3-7: (a) Interpolations without clear meaning generated by unrelated input images (the four corners encircled in red dash lines), shown in the form of 2D-grid interpolation. (b) Random samples based on a reference distribution in latent space. 38

Figure 3-8: Low-level feature maps for a specific input image (left, size of 256^2) with size of 16^2, 8^2 and 4^2 before mixing. All features are normalized to the range of [0, 1]. 39

Figure 3-9: Examples from different sizes of PatchGAN receptive field. The receptive field applied for training increases from left to right. The best results are when the patch size equals to 70×70. 41

Figure 4-1: Examples of interpolating on a large gap where ground truth is available. The leftmost and rightmost columns are known slices as inputs, and the gap between them is five consecutive slices. Results are produced by (a) AE, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual, (f) Ground truth. 45

Figure 4-2: Examples of interpolating on a large gap where ground truth is available. The leftmost and rightmost columns are known slices as inputs, and the gap between them is five consecutive slices. Results are produced by (a) AE, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual, (f) Ground truth. 46

Figure 4-3: Examples of interpolating on a large gap where ground truth is available. The leftmost and rightmost columns are known slices as inputs, and the gap between them is five consecutive slices. Results are produced by (a) AE, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual, (f) Ground truth. 47

Figure 4-4: Qualitative results and locally enlarged view. The upper left and lower left are ground truth of interpolations, other results are intercepted from a series of continuous interpolations. The best result is highlighted in bold and second best is underlined. (a)

Ground truth, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual. 48

Figure 4-5: Examples of interpolating between adjacent slices. The left most column and right most column are input images, the middle ones are interpolations. Results are produced by (a) AE, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual 49

Figure 4-6: Examples of interpolating between adjacent slices. The left most column and right most column are input images, the middle ones are interpolations. Results are produced by (a) AE, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual 50

Figure 4-7: Feature reconstruction results from the autoencoder part of each model. 51

Figure 4-8: Random samples based on a reference distribution in latent space. 52

Figure 4-9: Interpolating on non-relevant inputs to show how the model learns to interpolate "semantically" step by step with the increasing number of training epochs. At the beginning, AE only trained with D_1. Later, we added the regularizer D_2. 53

Figure 4-10: Histogram interaction with ground truth. Receptive field sizes of (a) 1×1 (PixelGAN) (b) 16×16 (c) 70×70 (d) 142×142 (e) 256×256 (ImageGAN). 55

Figure 5-1: Examples of 2D orthogonal projection reconstruction from any view. The Euler angle (a) $\theta = 0$ (coronal view); (b) $\theta = \pi/4$; (c) $\theta = \pi/2$ (sagittal view); (d) $\theta = 3\pi/4$. 57

Figure 5-2: Comparison of different approaches dealing with gaps before orthogonal projection. (a) Repeat the previous line of pixels value (b) Bilinear interpolation (c) Not filling the gaps (f) HRINet interpolation. 58

Figure 5-3: Illustration of how we create the sagittal view orthogonal projection and weighted projection from axial view images. Red arrows stand for the direction we calculate weighted sum, which projecting whole images into one-pixel-width lines (yellow dash lines). (a) weighted projection, where the weight w obeys weights distribution $\mathcal{W}(k, n)$. (b) Orthogonal projection, where each pixel has identical weight. 59

Figure 5-4: Examples of weighted projection results. The last row shows the distribution of weights according to their relative positions in axial view, all of them are in the range of [0, 1]. (a) $\mathrm{w} \propto \frac{1}{\mathrm{k}}$; (b) $w \propto k$; (c) $\mathrm{w} \in \mathcal{N}\left(\frac{\mathrm{n}}{4}, \left(\frac{\mathrm{n}}{11}\right)^2\right)$; (d) $\mathrm{w} \in \mathcal{N}\left(\frac{\mathrm{n}}{2}, \left(\frac{\mathrm{n}}{11}\right)^2\right)$, where w is weight, k stands for relative position and $k \in [0, n]$, n is the width of the axial view image. 60

Figure 5-5: The inner organs and tissues model (first row) and overall structural model (second row) created from original axial view slices and their interpolations. 61

Figure 5-6: The illustration of improvements on 3D volume reconstruction. (a) Modeling on original dataset. (b) Modeling with high quality interpolations. 62

List of Tables

Table 3-1: Description of the dataset. The rows with shadow are test sets. 36

Table 3-2: The contrast of feature map sizes be passed through for interpolation between ACAI and HIRNet. 39

Table 3-3: Examples of feature maps with various sizes for a specific input image in Figure 3-8, the 2^{th}, 4^{th} and 8^{th} channels are visualized after activation function and are normalized to the range of [0, 1]. 40

Table 4-1: Different metrices for evaluating the performance of our model. The best result is highlighted in bold and second-best is underlined. (PSNR/SSIM and Ma et al. are the higher the better, RMSE is the lower the better).....44

Table 4-2: No-reference metrices for evaluating the performance of our model. The best result is highlighted in bold and second best is underlined. (NIQE the lower the better, Ma et al. [52] the higher the better).....49

Table 4-3: Different metrices for evaluating the performance of feature reconstruction. Tests are done for 256^2 inputs. The best result is highlighted in bold and second best is underlined. (PSNR and SSIM are the higher the better, L1 and L2 loss are the lower the better.) 51

Table 4-4: Ablation study for content loss, regularizes and training method. (PSNR/SSIM and Ma et al. [52] the higher the better).....54

Table 4-5: Histogram interaction scores with ground truth. 54

Chapter 1. Introduction

1.1 Motivation

Since most 3D biomedical volume images are sampled at locations where the distance between consecutive slices is significantly larger than in-plane pixel size [70][71][72]. Especially, the number of scans during in-vivo CT data scanning is limited for the purpose of avoiding the risks of radiation on patients [73] as well as reducing the examining time, which leads to spatial missing between adjacent CT slices. The missing parts, also called gaps, are sometimes neglectable when the distance of the gap is relatively large. In some cases, the gap distance can be up to 1/50 or 1/40 of actual CT image size. Those large gaps between slices might lead to huge bias during the process of 3D reconstruction of bone geometry and tissue features [74]. In 3D reconstruction or 2D projection reconstruction, to eliminate those gaps and get better subsequent results, what people usually do is, for each CT slices, stacking the same pixels cubic with the same pixel value repeatedly in the vertical direction, which could cause discontinuities in the silhouette and texture of reconstructed 3D human structure. The results modeled by this way have uneven and unsmooth surfaces and huge bias.

Image interpolation is used to create a certain number of new slices between known slices to obtain an isotropic volume image. And the created slices must keep the semantic consistency and smoothness with the adjacent known slices, which is extremely hard to maintain in traditional image processing approaches because we can hardly access the underlying data manifold and therefore cannot create realistic interpolations.

Deeping learning provided solutions to these problems by training a very deep neural network to disentangle the abstract latent representation among images with the same distribution. Another benefit is that the manifold learning in deep learning parses the complex nonlinear correlation between data points and their embeddings so that make it possible to generate realistic data points by resampling in the latent space.

The autoencoder [39] has been proved to be efficient for dimensionality reduction generating latent representation of data points that obey the same distribution [40][41][45]. The highly abstract and entangled latent code captures the semantic representation of images on the manifold, and linearly interpolation between latent space reflects the semantically smooth interpolation on data space[40][44]. The ACAI model [44] shows the potential of making the autoencoder learn the way of latent embedding correctly under adversarial constraints.

However, interpolating between high-resolution images is always a very challenging task. Because the autoencoder based models are all have a common defection: The autoencoder is lossy way for image compression, although it can correctly capture the latent representation, and keeping

rough shape and structure when reconstructed by decoder, fine details like texture are reluctant to be recovered, especially observable in the case of highly dimensionality reduction [40][42]. Besides, the size and dimensionality of the latent code are very sensitive for interpolation. Inappropriate size of latent code might lead to overlapping or blurry problems [44][46] for interpolated images as illustrated in **Figure 1-1**. Concretely, the overlapping problem would happen when the dimensions of latent code is too much relative to image size. And the blurry case is usually caused by the too few dimensions for feature representation. Previously, work [44] [40][42] has been done on generating low-resolution interpolations to avoid the overlapping and blurry problems. But nowadays the medical field put high demands on high-resolution and realistic interpolations which makes the problem become more obvious.

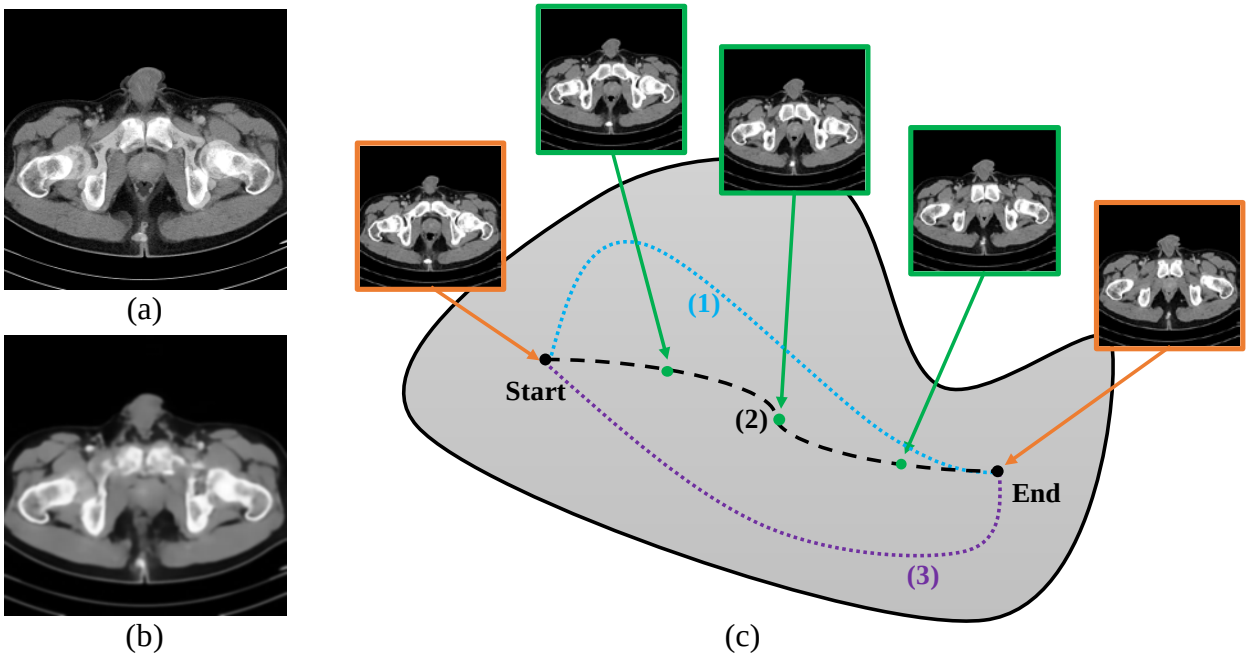


Figure 1-1: Example of overlapping and blurry problems. (a) The interpolated result is sharp but overlapped, close to interpolating in data space rather than latent space. (b) Interpolating in latent space, but the result is blurry and missing fine details. (c) "Appropriate" interpolations on the manifold. There are many possible paths like (1) (2) and (3) connect a pair of endpoints, but only (2) is considered semantically correct.

Thus, we proposed High-Resolution Interpolation Network (HRINet) aiming at producing realistic high-resolution CT image interpolations between known CT slices, and we also proposed a novel training method, alternatively supervising training that combine the idea of supervised and unsupervised training to improve the quality and accuracy of interpolations while keeping semantic smoothness. The results can also be used for 2D projected visualization and 3D body structural reconstruction. For instance, we can visualize the CT data in any given view including sagittal, coronal and axial views regardless in which view the scanning is done.

1.2 Overview of Our System

Figure 1-2 shows the overview of our system for high-resolution image interpolation. The input is a series of axial view CT slices with spatial continuity that scanned from the same patients. For each pair of adjacent slices, the space between slices is a constant, usually from 1*mm* to 7*mm* varies from machine to machine. Our purpose is, creating several high-resolution and realistic interpolations between these known slices to fill in every gap. The content of interpolated results should smoothly morph from one slice to the adjacent slice while keeping the "semantic similarity".

The generator is a vital part of the process, where the main components contain encoder, decoder and feature fusion mechanism. The encoder encodes the input images and generates latent code as the compressed feature representations, the process is known as "embedding". Feature fusion mechanism takes the latent code and accomplishes a kind of feature fusion in a way in latent space and generates the mixed latent code that combines features from two input data points. The mixed latent code is then decoded by decoder, mapping the latent representation to data space, more specifically, as a form of RGB or grayscale image as output, depending on the vector space of input.

The discrimination mechanism contains several regularizers that giving complicated feedback to the generator according to the overall quality, texture sharpness, structural correctness and semantic smoothness extend of produced interpolations. The discrimination mechanism supervises the generator and forces it to generate realistic results that merging the characteristics above. Meanwhile, some parameters of regularizers are updatable so that the discriminating ability is also improved as the training of the generator. Theoretically, the discrimination mechanism would make the generator generate indistinguishable interpolations from the dataset in the end, we consider that our model reaches the final state.

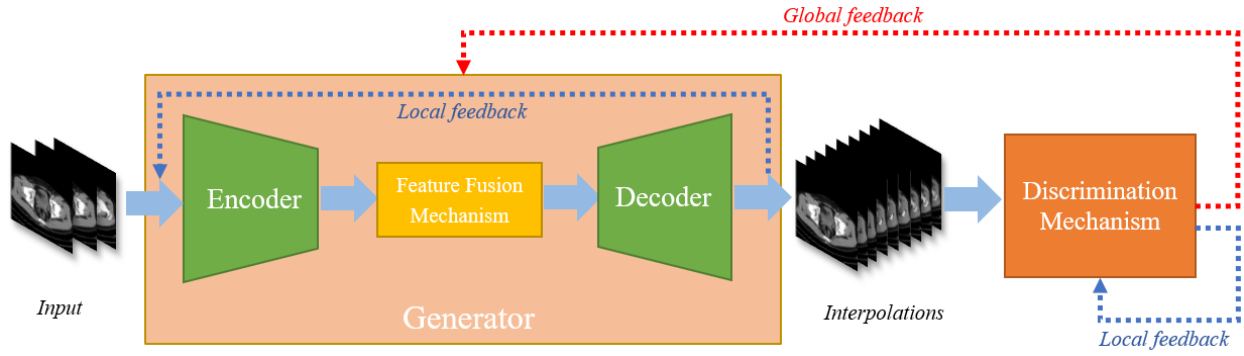


Figure 1-2: Overview of our system.

In our system, there are two kinds of negative feedback that maintain the balance of each component during training. The first one is global feedback, which exists between the generator and discrimination mechanisms, having effect on the whole generator. It corresponds to the extra assessment of the quality of overall results given by discrimination mechanism independent of the generator, which indicates the global updating direction for all parameters in generator. The other one is local feedback, which can be found within generator and discrimination mechanism. The scope of the local feedback is the individual components. For example, in generator, the local feedback can

be regarded as the bias of input images and reconstructed images, or the interpolations and their ground truth (if available), and it only has influence on the encoder and decoder parts, if the feature fusion mechanism is not parameter-oriented. In discrimination mechanism, the local feedback is used to improve the discriminating ability of itself, including the accuracy of predicting interpolation coefficient, determining the authenticity of input interpolations, etc.

Our system has a wide range of applications in the medical field. There are two main typical application scenarios: 2D volume projection and 3D volume reconstruction for the human body, summarized in **Figure 1-3**.

For 2D volume projection, it can be divided into orthogonal projection and weighted projection. Orthogonal projection refers to each pixel within slices having the same weight, for a given view, the projection is done undirected. Whereas weighted projection refers to the case when the weight of each pixel within slices obeys some kinds of customized distributions, such as Gaussian distribution or polynomial distribution, and the projection is done directionally. Orthogonal projection aims at visualizing overall structure and internal components of 2D volume, the projection results are similar to X-ray images. Weighted projection is used to show a specific part of body structure clearly and get rid of interference from other parts when doing volume projection, the weight distribution can be designed by ourselves. Both of which can be done by either standard coronal view and sagittal view, or even arbitrary view.

Our system also shows significant improvement for 3D volume reconstruction. We can take use of many existing software like "*Slicers*" and "*Osirix*" to do the 3D modeling from axial CT slices. Based on various reconstruction algorithms, which extract different regions of information from CT data, we are able to reconstruct internal human body structures such as skeletons and organs, as well as external silhouettes like skin.

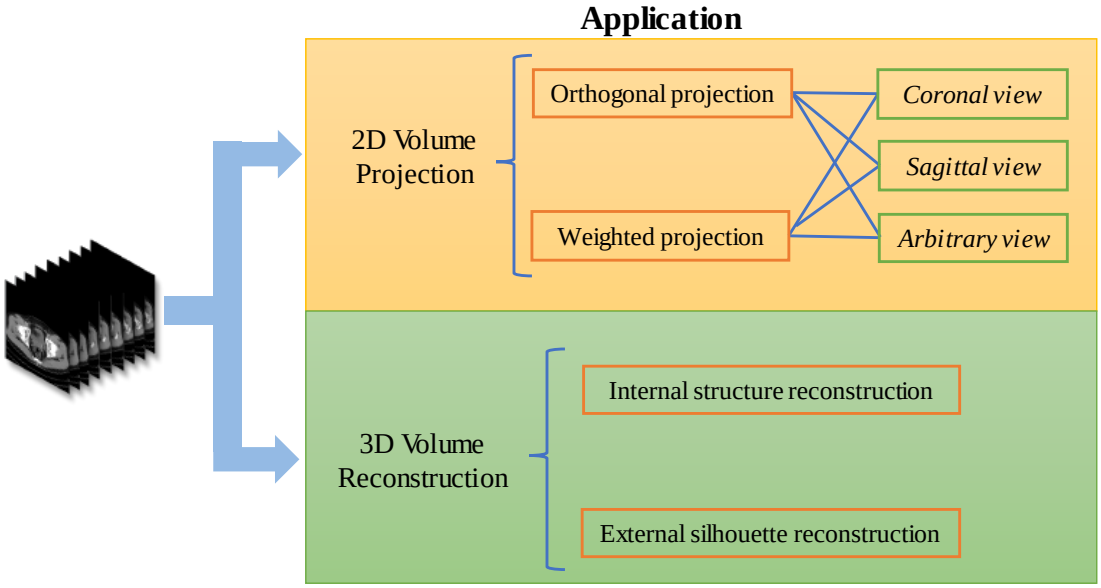


Figure 1-3: The application of our system.

1.3 Thesis Organization

In the thesis, Chapter 2 starts with a comprehensive literature review about deep learning, GANs and autoencoders. We first thoroughly introduce the general concept of deep learning and related background, such as CNN, activation function, loss function, and backpropagation, etc. Then we discuss its development status and several typical applications in medical fields, including disease diagnosis, lesions detection and segmentation. Later, we detailly illustrate the principle of GANs and its derivatives, which are the most popular generative models nowadays and have shown impressive ability for producing new photographs that look at least superficially authentic to humans. SRGAN and ESRGAN are especially emphasized in this section as they involve high-resolution image generation technologies that are closely related to our work. The idea of semantic interpolation is brought up with autoencoders in the following section, we elaborate the close relations among them. Finally, we introduce several reference based and no-reference based metrices for evaluating the quality of generated images.

Chapter 3 and Chapter 4 mainly describe the work we have done. In the first part of Chapter 3, we illustrate a state-of-the-art high-resolution interpolation network (HRINet) aiming at producing realistic and semantically smooth CT interpolations between two data points. We explicitly explain how we combine the idea of GANs and ACAI, and compensate their shortcomings individually to form the HRINet. Except for the network architecture, we introduce those novel characteristics one by one so as to show the rationality and correctness of our idea. Specifically, the network is denoted as HRINet-MSE or HRINet-Perceptual in terms of the choice of content loss. To achieve better performance, we propose a novel training method “alternatively supervising training” which combines supervised and unsupervised training steps alternatively. Later we analysis the experiments and results of HRINet and evaluate the performance of it among different metrices. We show HRINet makes great improvement for high-resolution CT interpolation qualitatively and quantitatively when comparing with other existing methods.

Chapter 4 shows two typical application scenarios of HRINet: 2D volume projection and 3D volume reconstruction. By projecting and modeling from original data and their interpolations created by HRINet, we illustrate how it make significant improvement in both cases.

Chapter 5 summarizes all the work we have done and draws the conclusion of this thesis. Finally, we analyze the pros and cons of our model, then discuss the future work that could possibly make improvement for the model.

1.4 Contributions

For semantically correct high-resolution CT interpolation methods based on HRINet, we make contributions as follows:

a) Semantically Correct Interpolation
   i. Proposed a novel training method “alternatively supervising training” for having the models get access to data space distribution, and balanced the model from being overly trained under monotonous training conditions.

b) High-resolution Interpolation
   i. Proposed two optional of networks (HRINet-MSE and HRINet-Perceptual) for producing high-resolution CT interpolations, and indicated the trade-off between the structural correctness and image sharpness.

   ii. Evaluated the performance of HRINet in the criteria of referenced and no-reference metrices for large gaps interpolations, adjacent slices interpolation, feature map reconstruction and representation learning.

c) Medical Application
   i. Showed the significant improvement of applying HRINet to generate interpolations that fill the gaps between CT slices for the task of 2D volume projection and 3D volume reconstruction.

1.4.1 Publications

1. Li, Jiawei, Jae Chul Koh, and Won-Sook Lee. "HRINet: Alternative Supervision Network for High-resolution CT image Interpolation." Accepted by *International Conference of Image Processing (ICIP)*, paper ID 2123, May 2020.

2. Li, Jiawei, Jae Chul Koh, and Won-Sook Lee. " Alternative Supervised and Unsupervised Learning for Image Interpolation and Projection Visualization" Under second-round review of *International Conference of Pattern Recognition (ICPR)*, paper ID 1133, April 2020.

Chapter 2. Literature Review

2.1 Deep Learning and Related Knowledge

Deep learning is the main direction of artificial intelligence development in recent years, and has achieved great success in academia and industry. Compared with the traditional manual customized rules by experts, deep learning makes the computer acquire knowledge from previous experience through multiple levels of learning to solve a variety of complex problems. The important knowledge related to deep learning is as follows.

(a) Convolutional neural network

Convolutional neural network (CNN) [53] extracts feature from local response, this is because in the field of image recognition, which is one of a typical application of convolutional neural network, the combination of a one pixel and other adjacent pixels is usually meaningful. But the further the two pixels apart from each other, the less correlation can be found between them. Thus, the convolutional neural network uses a fixed-size convolution kernel to extract features from an image, and the kernels can share features with each other, so that even when processing high-dimensional data, the number of parameters will not change. Besides, the number of extracted feature maps is identical with the number of convolution kernels. Compared with the fully connected neural network, the convolutional neural network greatly reduces the computational cost by decreasing the number of parameters that are required for model training.

Nowadays, convolutional neural networks are currently widely used in the field of computer vision. Many current mainstream network architectures and their derivatives such as VGG [50], ResNet [54], InceptionNet [55], and DenseNet [56] all use convolutional neural networks. In addition, in image super-resolution tasks, the deconvolution network is commonly applied to reduce the number of image channels and enlarge the size of the feature map.

(b) Activation function

The so-called activation function is a function that runs on the neuron of the artificial neural network and is responsible for mapping the input of the neuron to the output. The activation function can convert the output of the neuron from linear to nonlinear mapping [57][59]. After calculating the inner product of the input vector and the weight from the previous step, the neuron adds the offset term and applies the activation function to complete the whole process. The definition of activation function indicates that it is a kind of nonlinear mapping between real numbers, and it is derivable in most cases. The purpose of the activation function is to map the linear transformation into a high dimensional nonlinear space [57][58]. *Sigmoid* is one of the most widely used activation function, and its mathematical definition is the following:

$$Sigmoid = \frac{1}{1 + e^{-x}} \tag{2-1}$$

In general, a sigmoid function is monotonic, and has a first derivative which is bell shaped. A sigmoid function is constrained by a pair of horizontal asymptotes as $x \to \pm\infty$.The sigmoid function is convex for values less than 0, and it is concave for values more than 0. Because of this, the sigmoid function and its affine compositions can possess multiple optima. Besides, Sigmoid is a soft saturation activation function. When *x* tends to infinity, the derivative will be close to 0, which will cause the gradient vanishing problem, making the training stop to converge in advance [58]. Although sigmoid has these shortcomings, its physical meaning is close to the behavior of biological neurons, that is, the output range (0, 1) can be expressed as a probability, or it can be applied to the cross-entropy loss function.

The appearance of the *ReLU* activation function greatly alleviates the gradient vanishing problem [59]. ReLU is a piecewise function whose definition is shown in equation 2-2.

$$ReLU = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases} \tag{2-2}$$

ReLU allows the size of neural network models can be further extended. In the case when the input $x > 0$, it has identical continuous derivatives. When $x \leq 0$, the neuron will saturate, and the weights of neurons will stop to update. The appearance of ReLU function is the key to further develop the neural network. Another important property of ReLU is that it makes the neural network have the ability to express its coefficient, and improve the performance of very deep neural networks.

Other activation functions such as *LeakyReLU*, *PReLU* and *Tanh* all play important roles in different situation. And the choice of activation function has become one of the most important factors that determine the performance of the model.

(c) Loss function

The loss function, also called the objective function, is a method of evaluating how well a specific algorithm models the given data. Typically, it calculates the error between the prediction of neural networks and the real results defined by humans. The smaller the loss is, the more accurate prediction of the model made, and it also means the prediction closer to the real data distribution. The direction of optimization is defined by the bias calculated from the loss function which generates gradients for backpropagation and updates all the weights of parameters [60]. Broadly, loss functions can be classified into two major categories depending upon the type of learning task we are dealing with — Regression losses and Classification losses.

For regression losses, two most commonly used loss functions are *Mean Absolute Error (MAE)* and *Mean Square Error (MSE)*, sometimes they are also called *L1 loss* and *L2 loss*. The Mathematical expression of both are the following.

$$MAE = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N} \tag{2-3}$$

$$MSE = \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N} \tag{2-4}$$

Where N refers to is a vector of N predictions is generated from a sample of dataset on all variables, and y_i s the vector of observed values of the variable being predicted, with $\hat{y}_i$ being the predicted values.

MAE measures the average sum of absolute differences between predictions and actual observations. It's only concerned with the average magnitude of error irrespective of their direction. On the other hand, MSE measures the average squared difference between predictions and actual observations. Like MAE, it as well measures the magnitude of error without considering their direction. However, due to squaring, predictions which are far away from actual values are penalized heavily in comparison to less deviated predictions.

For classification losses, the *Cross-entropy Loss* is the common setting for classification problems [61], which is shown in equation 2-5. The loss increases as the predicted probability diverges from the actual label. Cross-entropy loss can measure the degree of difference between two different probability distributions in the same random variable. In deep learning, it is expressed as the difference between the real probability distribution and the predicted probability distribution. The smaller the value of cross-entropy, the better the model prediction effect.

$$\mathcal{H} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \, log \, \hat{y}_i + (1 - y_i) \, log(1 - \hat{y}_i)] \tag{2-5}$$

(d) Backpropagation

Neural network has forward and backward propagation. The forward propagation refers to making the feedforward prediction of the model, which is the process of getting training data and then calculating the prediction value through a neural network. And backpropagation is the process of modifying the model. The mathematical basis of backpropagation is the so-called "chain rule", that is, reversely derives the differentiation of the neural network layer by layer. And the dynamic programming based algorithm reduces the computational cost for updating the whole network.

(e) Optimization

The optimization algorithm is a guide for updating the weights of the neural network in order to obtain the optimal value of the objective function which contains the setting of data batch size, the learning rate and attenuation, moving average and other strategies to speed up the optimization and iteration of the neural network [62]. Optimizers carry out the optimization algorithm, and they tie together the loss function and model parameters by updating the model in response to the output of the loss function. In simpler terms, optimizers shape and mold models into its most accurate possible form by futzing with the weights. The loss function and gradient are the guide to point out the right direction for the optimizer to move forward [63].

Optimizers like SGD [64], RMSProp, AdaGrad [65] and Adam [66] have shown the ability of decreasing the gradient effectively and they are specifically designed for training deep neural networks. Among them, Adam is the most popular one nowadays because of its stability and adaptive feature.

Adam can be regarded as a combination of RMSprop and SGD with momentum. It uses the squared gradients to scale the learning rate like RMSprop, and it takes advantage of momentum by using the moving average of the gradient instead of the gradient itself [62][66]. Moreover, Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. It uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network [63].

2.1.1 Application in Medical Field

Deep learning, as a kind of novel technology and strong tools that has greatly prompted the development of medical level in recent years. Take medical imaging, for example, people have benefited a lot from recent advances in image classification and target detection [4]. And many studies have achieved good results in the complex diagnosis of different disciplines like dermatology, radiology, ophthalmology, and pathology, etc. Deep learning can provide doctors with auxiliary opinions for diagnosis and sometimes achieves doctor-level accuracy on a large number of diagnostic tasks.

(a) Disease diagnosis

The first appearance of deep learning applied in the medical field can be traced back to 1995. At that time, o et al. [1] described a CNN classification model to detect lung nodules on chest X-rays. They used 55 chest x-rays and a CNN with 2 hidden layers to output whether a region had a lung nodule or not. The relative availability of chest x-ray images has likely accelerated deep learning progress in this modality. Rajkomar et al. [2] augmented 1850 chest x-ray images into 150,000 training samples and used a modified pre-trained GoogLeNet [3] to classify the orientation of the images into frontal or lateral views with near 100% accuracy. Hosseini-Asl et al. [5] achieved state of the art results in diagnosing patients with Alzheimer's Disease versus normal by employing 3-D CNNs in an autoencoder architecture, pre-trained on the CADDementia dataset to learn generic brain structural features, and finally get the accuracy as high as 99%.

(b) Lesions detection

Detection, sometimes referred to as computer-aided detection (CADe) is a hot area of research because losing lesions during scanning can have serious consequences for both the patient and clinician. The purpose of CADe is to find or localize abnormal or suspicious regions in structural images, and thus to alert clinicians. Although CADe is well established in medical imaging, deep learning methods have improved its performance in different clinical applications. For example, Ciresan et al. [10] used 11 to 13 layers CNNs to identify mitotic figures in 50 breast histology images from the MITOS dataset. Their approach achieved precision and recall scores of 0.88 and 0.70 respectively. For lung lesions detection, Ciompi et al. [11] applied Overfeat to 2D slices of CT lung scans oriented in the coronal, axial and sagittal planes, to predict the presence of nodules within and around lung fissures.

(c) Segmentation

CT and MRI image segmentation research covers a variety of organs such as liver, prostate and knee cartilage, but a large amount of work has focused on brain segmentation, including tumor

segmentation [6][8]. Traditionally, medical anatomical segmentation is done manually, and the clinician draws the contour slice by slice through the entire MRI or CT volume stack, so the ideal method is to achieve a solution that can automatically complete this tedious work with high accuracy. Moeskops et al. [7] used 3 CNNs, each with a different 2D input patch size, and running in parallel to classify and segment the MRI brain images of 22 preterm infants and 35 adults into different tissue categories. The advantage of using 3 different input patch sizes is that each focus on capturing different aspects of the image, with the smallest patch focused on local textures while the larger patch sizes capture spatial features. Kleesiek et al. [9] used 3D convolutional architecture for skull extraction, a technique that was not limited to non-enhanced T1-weighted MR images. While training their 3D CNN, they constructed mini batches of multiple cubes that were larger than the actual input to their 3D CNN for computational efficiency. Over four different data sets, their method achieved the highest average specificity measures in comparison to several commonly used tools (i.e., BET, BSE, ROBEX, and 3dSkullStrip).

(d) Others

Deeping learning is also widely used in other medical area such as anatomy localization [12][13][14], clinical notes auxiliary analysis [15] and protein sequencing [16], etc., where CNNs and RNNs have shown great success compared with traditional approaches.

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) is a class of deep learning frameworks first proposed by Goodfellow et al. in 2014 [17], and applying to the unsupervised model, which opened up a new idea in the field of data generation. Given a training set, the technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers and have many realistic characteristics. Except for unsupervised learning, GANs have also proven useful for both supervised learning and semi-supervised learning task including representation learning [18], image inpainting [19], image super-resolution [20][21] and neural style transfer [22][23], etc.

2.2.1 Principle

The GANs are trained by two neural networks in adversarial, the two neural networks are generator and discriminator. GANs do not have too many restrictions on the choice of generator and discriminator models. In the initial version, a multi-layer perceptron network was used for training. GANs were inspired by the game theory, the generator and discriminator will complete with each other to achieve the Nash equilibrium in the training process. The architecture of GANs is illustrated in **Figure 2-1**.

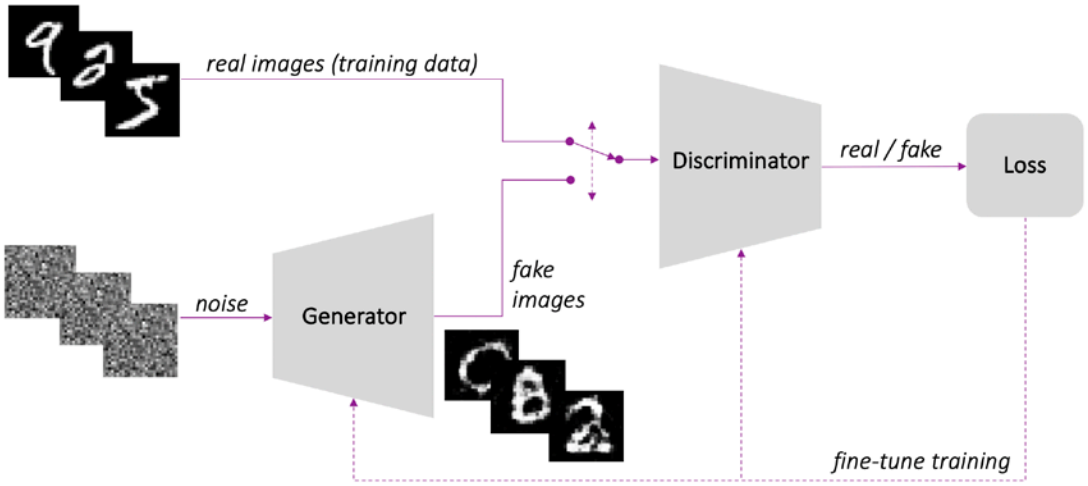


Figure 2-1: The general architecture of GANs, from [17].

The main principle of generator G is to generate fake data to fit the potential distribution of real data as much as possible, while the main principle of discriminator G is to correctly distinguish real data from fake data by the difference of their distribution. The input of the generator is a random noise vector z, and z obey a uniform or normal distribution in most cases. The noise is mapped to a new data space via generator to obtain a fake sample, $G(z)$, which is a multi-dimensional vector. The discriminator D is a binary classifier, it takes both the real sample from the dataset, and the fake sample generated by generator as the input, and the output of discriminator represents the probability that the sample is a real rather than a fake. When the discriminator is distinguishable from whether the data comes from the real dataset or the generator, the optimal state is reached. At that point, we denote that the generator has learned the distribution of real data.

GANs take unsupervised adversarial training, by minimizing Jensen-Shannon divergence between real data distribution $\mathbb{P}_{\boldsymbol{r}}$ and prior distribution $\mathbb{P}_{\boldsymbol{g}}$. The overall the minimax objective for optimization is:

$$\min_G \max_D \mathbb{E}_{x\sim\mathbb{P}_r(x)}\left[log\big(D(x)\big)\right] + \mathbb{E}_{x\sim\mathbb{P}_g(x)}\left[log\big(1-D(\tilde{x})\big)\right] \tag{2-6}$$

Where x is sampled from real data distribution $\mathbb{P}_{\boldsymbol{r}}(x)$, and $\tilde{x}$ is sampled from the prior distribution $\mathbb{P}_{\boldsymbol{g}}(x)$ such as uniform or Gaussian distribution, which can also be written as $D(G(z))$. $\mathbb{E}(\cdot)$ represents the expectation.

Given the generator, we need to minimize **2-6** to obtain the optimal solution. In continuous space, the objective function **2-6** achieves its minimum value at

$$D_G^*(x) = \frac{\mathbb{P}_r(x)}{\mathbb{P}_r(x) + \mathbb{P}_g(x)} \tag{2-7}$$

And this is the optimal solution of discriminator. From **2-7**, the discriminator of GAN estimates the ratio of two probability densities, which is the key difference from Markov chain or lower bound based methods.

In order to train the generator G minimize $\log(1 - D(\tilde{x}))$ and train the discriminator D to maximize $\log(D(x))$, we typically use an alternative training method, this is, First, we fix G and optimize D to maximize the discrimination accuracy of D. Then, we fix D and optimize G to minimize the discrimination accuracy of D. This process alternates and we could achieve the global optimal solution if and only if $\mathbb{P}_r = \mathbb{P}_g$. In the training process, we empirically update the parameters of D for k times and then update the parameters of G once.

2.2.2 GANs for Image Super-Resolution

Super resolution (SR), is the process of upscaling and or improving the details within an image by reconstructing from low resolution images to achieve higher resolution. Previous SR algorithms such as bilinear, bicubic and SRCNN [47] lead to good results but may also introduce ghosting and sawtooth, and sometimes the sharpness of SR results is not satisfied. The appearance of conditional GANs addresses those issues and even makes the generator have the ability to generate vivid and photo-realistic SR images with large upscaling factors.

Super-resolution Generative Adversarial Networks (SRGAN) [20] first applied GANs to image super-resolution tasks and achieved impressive results. It used an adversarial objective function that promotes super-resolved outputs that lie close to the manifold of natural images, as shown in **Figure 2-2**. Since the previous super-resolution approaches used the MSE as the loss function, the generated images are always blurry and lacking sharp textures because of the overly smoothing effect of MSE [20][24]. Although the signal-to-noise ratio is relatively high, it lacks high-frequency components, and there is still a sense of blur in human perceptual level. By applying the perceptual loss for the network instead of MSE loss, that is, an advanced semantic loss function that was defined on the intermediate feature representation of a pretrained VGG-16 network in the form of L1 distance, which greatly improves the authenticity of generated images. Second, they introduce an extra adversarial loss item to balance a min-max game between a generator and a discriminator. In addition, the ResNet [54] based architecture is used in feature extraction. When the network becomes very deep, it avoids the problem of gradient vanishing and explosion during the process of gradient backpropagation, which makes the performance of feature extraction to be better. The overall framework basically favors outputs that are perceptually close to human perceptual level which is measured by a novel criterion Mean Opinion Score (MOS).

Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN) [21] builds upon SRGAN [20] by removing batch normalization and incorporating dense blocks. Each dense block's input is also connected to the output of the respective block making a residual connection over each dense block. ESRGAN also has a global residual connection to enforce residual learning. Moreover, the authors introduce an enhanced discriminator called Relativistic GAN [25]. The training is performed on a total of 3,450 images from the DIV2K [26] and Flicker2K datasets employing augmentation via the L1 loss function first and then using the trained model using perceptual loss. The patch size for training is set to 128×128, having a network depth of 23 blocks. Each block contains five convolutional layers, each with 64 feature maps. The visual results are comparatively better as compared to SRGAN.

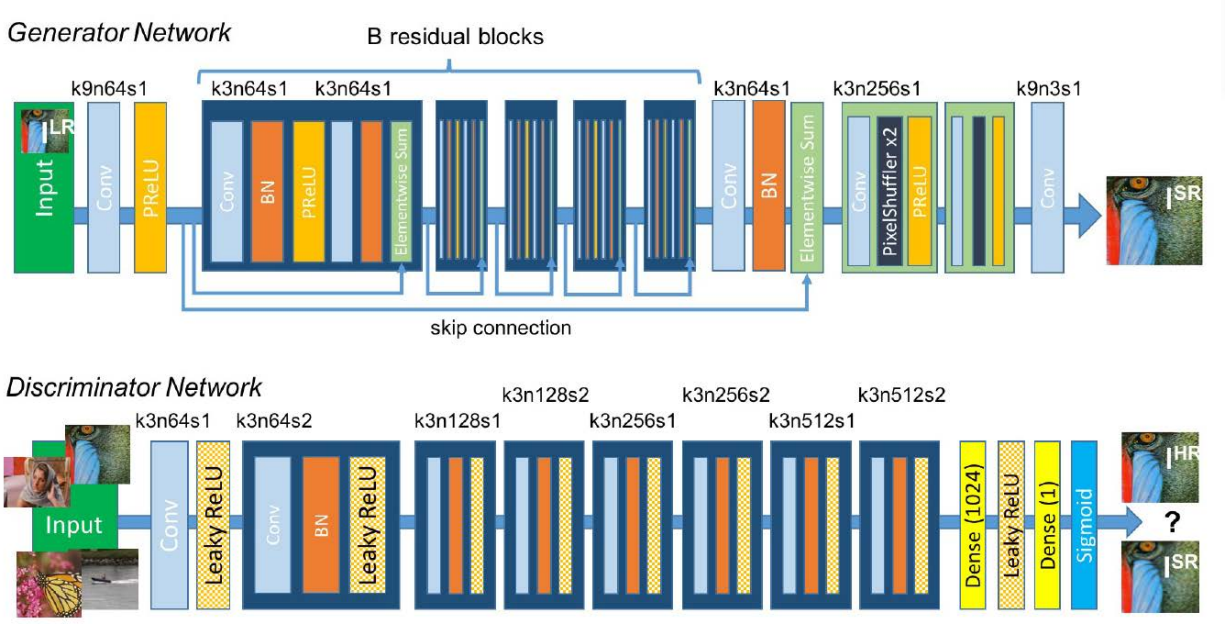


Figure 2-2: The architecture of generator and discriminator in SRGAN, reprinted from [20].

The EnhanceNet [27] is designed focuses on creating faithful texture details in high-resolution super-resolved images. The key issue with regular image quality metrices (such as PSNR and SSIM) is that they are not always consistent with the perceptual quality of the image, which may lead to overly smoothed images that do not have sharp textures. To address that issue, EnhanceNet used two extra loss terms except for the regular pixel-level MSE loss. The first one is the perceptual loss as illustrated above in SRGAN [20] and ESRGAN [21] models. The second one is the texture matching loss is used to match the texture for low-resolution and high-resolution images and is quantified as the L1 loss between gram matrices computed from deep features. They also adopted the adversarialy training process and try to make the SR network fool the discriminator. For network architecture, they built EnhanceNet based on the Fully Convolutional Network [28] and residual learning principle [29]. Their results showed that only applied pixel-level loss can obtain the best PSNR, and adding the additional loss term with the adversarial training can bring more realistic and better perceptual outputs. However, on the downside, the proposed adversarial training could create visible artifacts when super-resolving highly textured regions.

2.2.3 WGAN, WAN-GP and other GANs Derivatives

Though GANs have powerful learning abilities and a wide range of applications, there are still some problems with GAN. For example, the GANs are extremely hard to train and the adversarial balance can easily be broken during training, the network is very sensitive to the choice of model and parameters, the loss function of the generator and discriminator cannot indicate the training process, etc.[30] Besides, the “model collapse” is another a critical issue for GANs and results in the generated samples lacks diversity [31].

To enhance the GANs stability, Wasserstein GAN (WGAN) [32] studied these problems theoretically and gave simple but effective solutions. Since the real data distribution and prior

distribution can have very little overlap, in extreme cases, the Jensen-Shannon (JS) divergence of objective function can be a constant, which causes the vanishing gradient problem while using the gradient descent method to optimize GANs. The authors analysed the contradiction among them, and proposed WGAN optimize to the objective function by using the Earth-Mover (SM) distance to replace the JS divergence for evaluating the distribution distance between real data and the generated data, and stabilize the training process at the same time. Wasserstein distance is smoother and stable compared with JS divergence, thus even if there is no overlap between two distributions, the model is still able to make a corresponding measurement and provide a meaningful gradient. Concretely, a critic function that builds on the Lipschitz constraint is used to represent the discriminator. The equation for calculating Wasserstein distance can be denoted by the following.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[||x - y||\big] \tag{2-8}$$

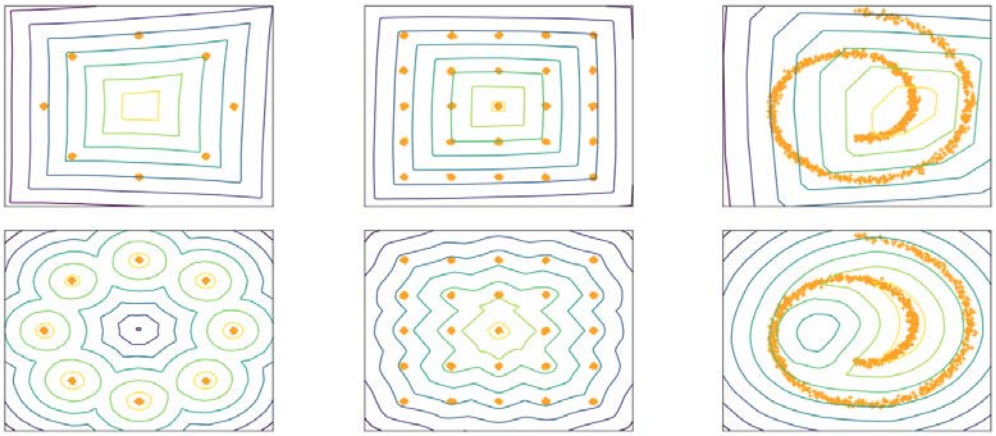
Where $\mathbb{P}_r$ is the data distribution, $\mathbb{P}_g$ is mathematically defined as the greatest lower bound (infimum) for any transport plan. $\prod(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$.

However, the equation 2-9 for the Wasserstein distance is highly intractable. Using the Kantorovich-Rubinstein duality, we can simplify the calculation to

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)] \tag{2-9}$$

where *sup* is the least upper bound and f is a 1-Lipschitz function following the constraint $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$.

While WGAN makes significant progress towards stable training of GANs, it still generate poorly samples or fail to converge in some settings. In light of that, Gulrajani et al. [33] find that the training failures are often due to the use of weight clipping in WGAN to enforce a Lipschitz constraint on the critic, which can lead to pathological behavior. Concretely, if the clipping parameter is large, then it will take a long time for any weights to reach the limit, thereby making it harder to train the critic till optimality. If the clipping is small, it can easily lead to vanishing gradients when the number of layers is huge, or when the batch normalization (BN) [34] is removed. And that attribute makes model performance very sensitive to this hyperparameter. In **Figure 2-4**, when batch normalization is removed, the discriminator moves from diminishing gradients to exploding gradients when the weight clipping parameter ***c*** increases from 0.01 to 0.1.



8 Gaussians 25 Gaussians Swiss Roll

Figure 2-3: Value surfaces of WGAN critics trained to optimality on toy datasets using (top) weight clipping and (bottom) gradient penalty, reprinted from [33]. Critics trained with weight clipping fail to capture higher moments of the data distribution. The generator is held fixed at the real data plus Gaussian noise.

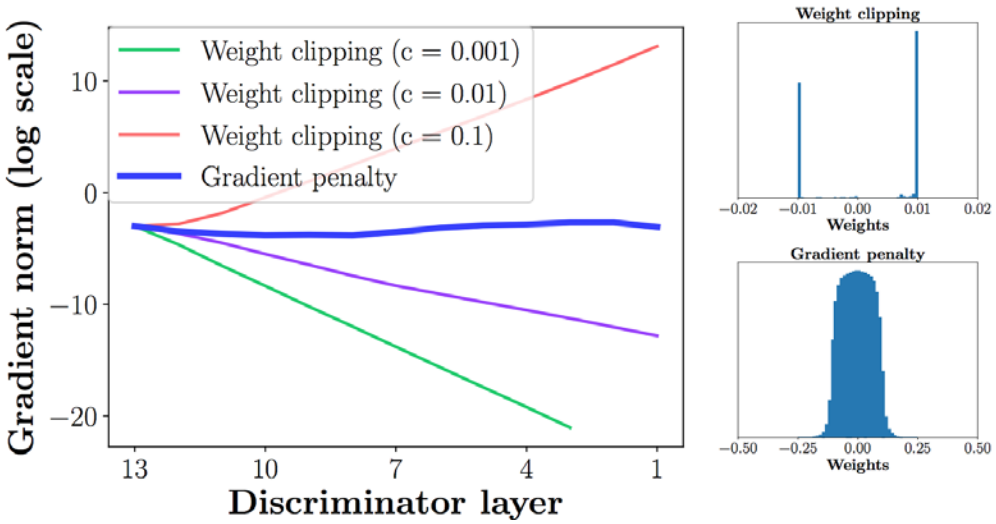


Figure 2-4: Gradient norms of deep WGAN critics during training on the Swiss Roll dataset either explode or vanish when using weight clipping, but not when using a gradient penalty (right), weight clipping (top) pushes weights towards two values (the extremes of the clipping range), unlike gradient penalty (bottom), reprinted from [33].

They propose WGAN with gradient penalty (WGAN-GP) [33] for enforcing the Lipschitz constraint. Instead of clipping weights, penalize the norm of the gradient of the critic with respect to its input. In terms of gradient penalty, it is a differentiable function $\boldsymbol{f}$ is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere. And points interpolated between the real and generated data should have a gradient norm of 1 for f. Therefore, instead of applying clipping, WGAN-GP penalizes the model if the gradient norm moves away from its target norm value 1. To

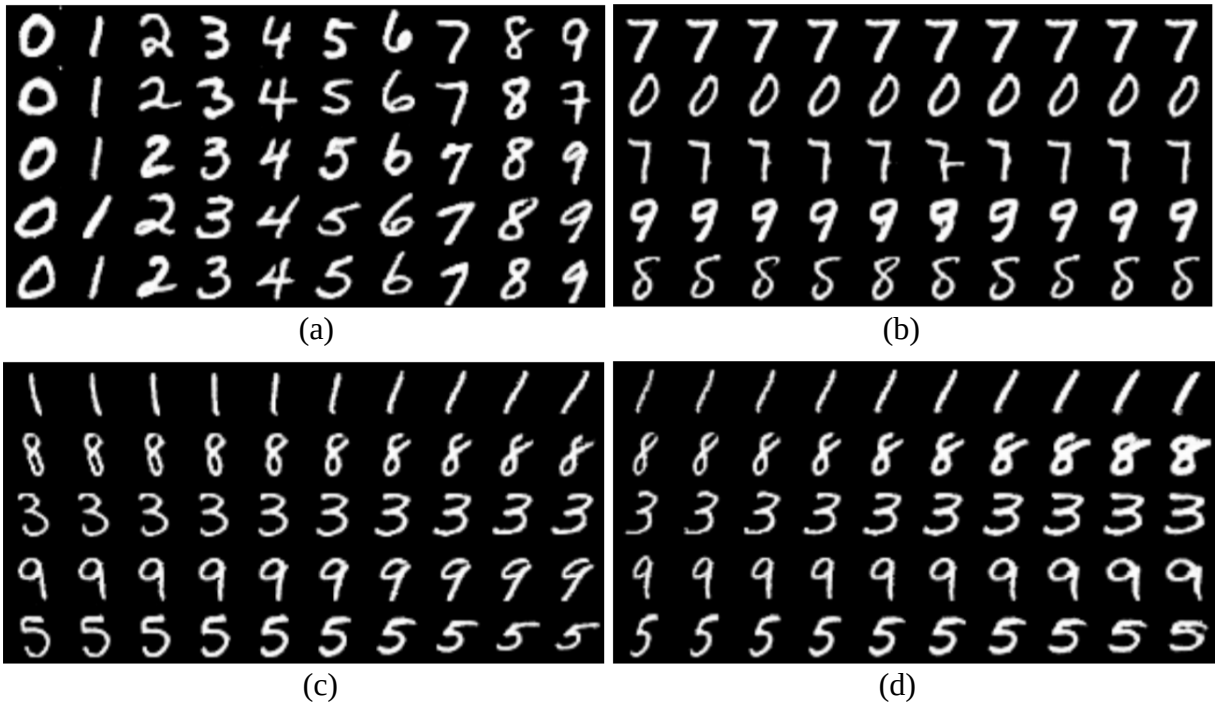
circumvent tractability issues, they enforce a soft version of the constraint with a penalty on the gradient norm for random samples $\hat{x}\sim\mathbb{P}_{\hat{x}}$. The new objective becomes

$$L = \underset{\tilde{x}\sim\mathbb{P}_g}{\mathbb{E}}[D(\tilde{x})] - \underset{x\sim\mathbb{P}_r}{\mathbb{E}}[D(x)] + \lambda \underset{\hat{x}\sim\mathbb{P}_{\hat{x}}}{\mathbb{E}}\left[\left(\left|\left|\nabla_{\hat{x}}D(\hat{x})\right|\right|_2 - 1\right)^2\right] \tag{2-10}$$

Where $\hat{x}$ sampled from $\tilde{x}$ and x with t uniformly sampled in the range of [0, 1], and λ is set to 10 constantly. x is used to calculate the gradient norm is any points sampled between $\mathbb{P}_g$and $\mathbb{P}_r$.

Except for the new cost function, the authors also removed the BN layers for the critic, because BN creates correlations between samples in the same batch which could weaken the effectiveness of the gradient penalty. In a word, their method converges faster and generates higher-quality samples than WGAN with weight clipping.

Information maximizing GAN (InfoGAN) [35] is an information-theoretic extension of GANs that is able to learn disentangled features in a completely unsupervised way. A disentangled representation is one which explicitly represents the salient features of a data instance. InfoGAN modified the objective of GANs by maximizing the mutual information between a fixed small subset of GAN's noise variables and observations to learn meaningful representations.



(a) (b)

(c) (d)

Figure 2-5: Manipulating latent codes on MNIST, reprinted from [44]. (a) Varying c_1 on InfoGAN (Digit type) (b) Varying c_1 on regular GAN (No clear meaning). (c) Varying c_2 from −2 to 2 on InfoGAN (Rotation). (d) Varying c_3 from −2 to 2 on InfoGAN (Width).

In GANs, there are no explicit restrictions on the manners in which the generator may use the noise. As a result, it is possible to use noise in a highly entangled way, and the noise does not correspond to the semantic features of the data. However, it makes sense to semantically decompose a domain based on the semantic features of the data under consideration. InfoGAN use this method

by decomposing the input noise vector into two parts: (i) *z* which is treated as a source of noise, (ii) *c* called the latent code and targeted at the salient structured semantic features of the data distribution. Thus, the generator network combines the incompressible noise *z* and the latent code *c* to be *G*(*z*, *c*). In order to emphasis the effect of the latent code *c* and avoid it being ignored by the generator, they proposed the information-theoretic regularization in equation 2-11 to regulate the min-max objective by subtracting the regular GANs cost function with an extra term $I(c; G(z, c))$, which refers to the maximized mutual information.

$$\min_G \max_D V_1 = V(D, G) - \lambda I\big(c; G(z, c)\big) \tag{2-11}$$

Figure 2-5 shows their experiment on MNIST dataset. By varying different dimensions of the latent code *c*, some patterns can be found in the generated image, for example, c_1 determining the digit type, c_2 and c_3 controlling the rotation angle and width of digits. For comparison, varying the latent code *c* for GANs has no clear meaning due to the highly entangled feature.

Other works such as Qi et al. propose Loss-Sensitive GAN (LSGAN) [36] to control the Lipschitz constant of the discriminator by adding regularization terms defined on input examples. Donahue et al. propose Bidirectional GANs (BiGANs) [37] to map the real data to the latent variable space. Miyato et al. proposed spectral normalization [38] to normalizing the weight matrices impose regularization on the space outside of the supports of the generator and data distributions without introducing somewhat heuristic means. They all made great contribution to the improvement of GANs.

2.3 Autoencoders

Autoencoders [39] contain encoder and decoder, both of which nowadays refer to multi-layer neural networks. Autoencoders are often used to learn efficient data coding in an unsupervised manner, the main components can be divided into input layers, hidden layers and output layers. Autoencoders have been proved useful for encoding and decoding, dimensionality reduction, feature extraction. With the development of research, people have made great progress of autoencoders and produced various variants such as Variational Autoencoder (VAE) [40], Denoising Autoencoder (DAE) [41], Adversarial Autoencoder (AAE) [42], etc.

2.3.1 AE and VAE

Autoencoders first encode the initial input vector to generate latent code as encoded information, and try to decode the latent coder to reconstruct the input by minimizing the error between the initial input and the reconstructed vector. The most basic autoencoder structure is one which simply maps input data points through a "bottleneck" layer whose dimensionality is smaller than the input, which is shown in **Figure 2-6**. The model contains an encoder function $g(.)$ parameterized by ϕ and a decoder function $f(.)$ parameterized by θ. For encoder, the mapping from data space into latent space can be denoted as $f_\theta: \mathcal{X} \to \mathcal{F}$, and the reverse process for decoder is $g_\phi: \mathcal{F} \to \mathcal{X}$.

During this setup, f_θ and g_ϕ are both neural networks which respectively map the input x to a deterministic latent code z and then back to a reconstructed input $\hat{x} = f_\theta(g_\phi(x))$. Typically, f_θ and g_ϕ are trained simultaneously with respect to $||x - \hat{x}||^2$. The overall loss function for the autoencoder is:

$$L_{AE}(\theta, \phi) = \frac{1}{n}\sum_{i=1}^{n}[x^{(i)} - f_\theta(g_\phi(x^{(i)}))]^2 \tag{2-12}$$

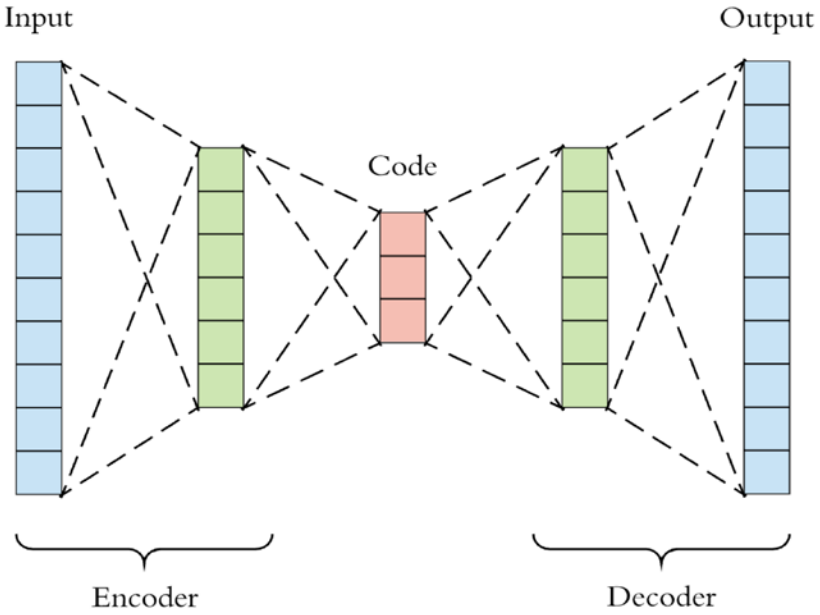


Figure 2-6: The general architecture of autoencoders.

Autoencoders are closely related to Principal Component Analysis (PCA) [43] because they both can reduce dimensionality in an unsupervised manner. If the activation function used within the autoencoder is linear within each layer, the latent variables present at the bottleneck directly correspond to the principal components of PCA. Generally, the activation function used in autoencoders is non-linear, typical activation functions are ReLU and sigmoid. Compared with PCA, autoencoders usually perform better in extraction features resulting from the attribute of deep neural networks. And the difference would become more obvious when the dimension of input is quite large.

VAE [40] inherits the architecture of traditional autoencoders and uses this to learn a data generating distribution, which allows us to take random samples from the latent space. These random samples can then be decoded by the decoder network to generate unique images that have similar characteristics to those that the network was trained on.

The main principle of VAE is, instead of mapping the input into a fixed vector, VAE maps it into a distribution p_θ parameterized by θ. Prior $p_\theta(\mathrm{z})$, likelihood $p_\theta(x|z)$ and posterior $p_\theta(z|x)$ determine the relationship between the data input x and the latent encoding vector z. The optimal parameter θ^* that maximizes the probability of generating real data samples can be denoted as the following.

$$p_\theta(x^{(i)}) = \int p_\theta(x^{(i)}|z)\ p_\theta(z)\, dz \tag{2-13}$$

$$\theta^* = \arg\max_\theta \prod_{i=1}^{n} p_\theta(x^{(i)})$$

$$= arg\, max_\theta \sum_{i=1}^{n} log p_\theta(x^{(i)}) \tag{2-14}$$

In VAE, the Kullback-Leibler (KL) divergence is used to quantify the distance between posterior $q_\phi(z|x)$ and the real distribution $p_\theta(z|x)$. And the KL divergence $D_{KL}(q_\phi(z|x)||p_\theta(z|x))$ measures the bias if the distribution $p_\theta(.)$ is used to represent $q_\phi(.)$, as illustrated in **Figure 2-7**.

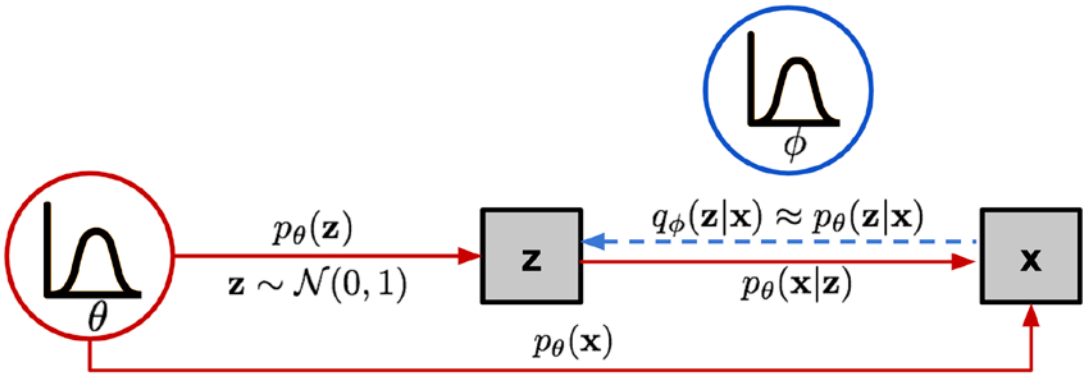


Figure 2-7: The principle of VAE, from [40]. Red solid lines denote the generative distribution $p_\theta(.)$ and blue dashed lines denote the distribution $q_\phi(z|x)$ to approximate the intractable posterior $p_\theta(z|x)$ by minimizing the KL divergence between them.

2.4 Semantic Interpolation

2.4.1 Background Knowledge

Interpolation, defined as creating a number of new data points within the range of a discrete set of known data points. linear interpolation represents the linear mixing of two data known data points x_1 and x_2 in the form of $\hat{x} = \alpha x_1 + (1-\alpha)x_2$, where α is the interpolation coefficient in the range of 0 to 1. For high-dimensional data like images, linear interpolation among them are overly simplistic and not considered on the notion of "semantic" because the interpolating is on the data space rather than in realistic data points, which corresponds to the simply mixing the pixel values proportionally. "Semantic interpolation" refers to creating interpolations that have similar data distribution with known data points, in other words, interpolations and data points should be on the same manifold if mapped to the high-dimensional space.

Typically, semantic interpolations can be acquired by the mixing of latent codes (latent representation), which refers to the low-dimensional and abstract features of the inputs. The more abstract of the latent codes, the more structural and semantic information they are able to represent. It turns out that methods such as PCA and autoencoder are very useful for dimensionality reduction and therefore very effective for generating latent code [41].

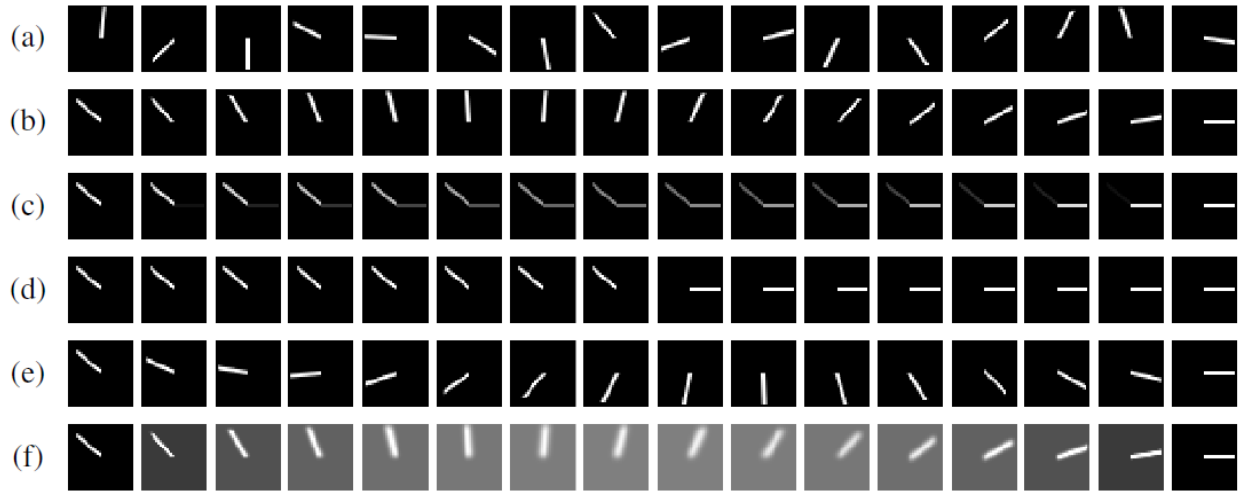


Figure 2-8: Examples of interpolations for synthetic lines dataset, reprinted from [44]. (a) Random samples from the dataset. (b) A perfect interpolation from $\frac{11}{14}\pi$ to 0 (ground truth). (c) Interpolating in data space rather than latent space. (d) An interpolation but with abruptly changes from one image to the other. (e) A smooth interpolation which takes a longer path from the start to end point than necessary. (f) An interpolation which takes the correct path but where intermediate points are not realistic.

A high-quality interpolation should have two characteristics: First, that intermediate points along the interpolation are indistinguishable from real data; and second, that the intermediate points provide a semantically smooth morphing between the endpoints. However, the latter characteristic is hard to enforce because it requires defining a notion of semantic similarity for a given dataset, which is often hard to explicitly codify. **Figure 2-8** elaborates the concept of high-quality interpolation. In **Figure 2-8 (b)**, lines are produced manually with identical angle difference between adjacent neighbours from $\frac{11}{14}\pi$ to 0, which is considered the perfect interpolation and meets the two characteristics mentioned above. **Figure 2-8 (c)** shows interpolating in data space rather than latent space, created by simply mixing the pixel values for two data points. Clearly, interpolating in this way produces points not on the data manifold, which means it is not semantic. **Figure 2-8 (d)** describes the case that interpolations have abruptly changes from one image to the other, rather than smoothly changing. Besides, high quality interpolations should also promise that taking the shortest path from the start to end point on the manifold and intermediate results are clear and realistic, which are qualified in **Figure 2-8 (e)** and **(f)** respectively.

2.4.2 Autoencoders and Interpolation

Since autoencoders are useful for dimensionality reduction, it can be used to generate the latent representation of input. For a given input $x \in \mathbb{R}^{d_x}$, the latent code $z \in \mathbb{R}^{d_z}$ is generated by passing through the encoder $f_\theta(x)$ parametrized by θ. And the decoder $g_\phi(z)$ attempt to produce the approximate reconstruction from the latent code *z*. The whole can be represented as

$$z = f_\theta(x) \tag{2-15}$$

$$x_\alpha = g_\phi(z) \tag{2-16}$$

Because L2 distance between *x* and x_α is minimized with the training of encoder and decoder, the latent code *z* has the ability to be a reliable abstract representation which represents the mapped result of high-dimensional input. Similar to PCA [43], the main components of the input are converted into compressed information and remained in the latent code *z*, while some fine details might be lost due to the "bottleneck" architecture of autoencoders.

Interpolating using an autoencoder describes the process of decoding a mixture of two latent codes using the decoder g_ϕ. Typically, the latent codes are combined via a convex combination, so that interpolation amounts to computing $\hat{x}_\alpha = g_\phi(\alpha z_1 + (1-\alpha)z_2)$ for some $\alpha \in [0,1]$ where $z_1 = f_\theta(x_1)$ and $z_2 = f_\theta(x_2)$ are the latent codes corresponding to data points x_1 and x_2. Ideally, adjusting α from 0 to 1 will produce a sequence of realistic data points where each subsequent $\hat{x}_\alpha$ is progressively less semantically similar to x_1 and more semantically similar to x_2. By decoding the convex combination of the latent codes for two data points, the autoencoder can produce an output which semantically mixes characteristics from the data points.

2.4.3 Adversarially Constrained Autoencoder Interpolation

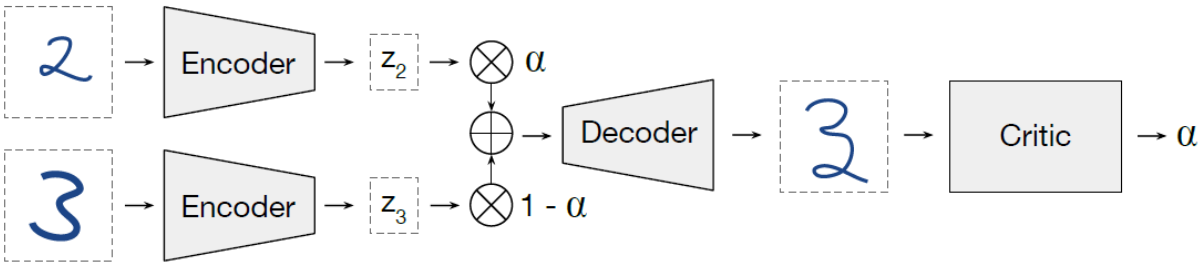


Figure 2-9: Adversarially Constrained Autoencoder Interpolation (ACAI), reprinted from [44]. A critic network tries to predict the interpolation coefficient α corresponding to an interpolated datapoint. The autoencoder is trained to fool the critic into outputting $\alpha = 0$.

Although autoencoders can generate semantic interpolations between data points to some extent, the results are not very realistic because simply mixing the latent code z_1 and z_2 may not guarantee that the interpolations are realistic data points and lying on the manifold. In fact, results produced from autoencoder sometimes are of low quality and not much different from the one that is

interpolated in data space. However, we rarely have access to the underlying data manifold, but we still hope that decoded points along the interpolation smoothly traverse the underlying manifold of the data instead of interpolating in data space.

To address above issue, Berthelot et al. proposed Adversarially Constrained Autoencoder Interpolation (ACAI) [44] to improve data interpolation and representation learning in autoencoders. They propose a regularizer which encourages interpolated data points to appear realistic, or more specifically, to appear indistinguishable from reconstructions of real data points. They introduce an extra critic network fed with interpolations of existing data points $\hat{x}_\alpha$, which is formed by

$$\hat{x}_\alpha = g_\phi\big(\alpha f_\theta(x_1) + (1-\alpha) f_\theta(x_2)\big) \tag{2-17}$$

The goal of critic is to predict the input coefficient α from $\hat{x}_\alpha$. And they constrain α in the range of [0, 0.5] during training to resolve the ambiguity between predicting α and $1-\alpha$. The autoencoder is trained to fool the critic to think that is always zero, in other words, making the critic to always think that an interpolated input is non-interpolated. This is achieved by adding an additional term to the autoencoder's loss to optimize its parameters to fool the critic. Formally, the critic and autoencoder are trained simultaneously to minimize

$$\mathcal{L}_d = \big|\big|d_\omega(\hat{x}_\alpha) - \alpha\big|\big|^2 + ||d_\omega(\gamma x + (1-\gamma) g_\phi(f_\theta(x))||^2 \tag{2-18}$$

$$\mathcal{L}_{f,g} = \Big|\Big|x - g_\phi\big(f_\theta(x)\big)\Big|\Big|^2 + \lambda\big|\big|d_\omega(\hat{x}_\alpha)\big|\big|^2 \tag{2-19}$$

Where $d_\omega(x)$ is the critic network, γ is a scalar hyperparameter in the range of [0, 1], and λ controls the weight of the regularization term to make the critic output 0 regardless of the value of α.

Ideally in the end, the autoencoder successfully "wins" this adversarial game by producing interpolated points which are indistinguishable from reconstructed data, that means the autoencoder is capable of producing semantically smooth interpolations under the "supervision" of critic. In practice they found that ACAI can provide a semantically smooth morphing between the endpoints and improve representation learning performance compared with basic autoencoders.

2.5 Image Quality Assessment

Image quality refers to visual attributes of images and focuses on the perceptual assessments of viewers. In general, image quality assessment (IQA) methods include subjective methods based on humans' perception (i.e., how realistic the image looks) and objective computational methods. the objective IQA methods are further divided into three types [67]: full-reference methods performing assessment using reference images, reduced-reference methods based on comparisons of extracted features, and no-reference methods (i.e., blind IQA) without any reference images.

2.5.1 Peak Signal-to-Noise Ratio

Peak signal-to-noise ratio (PSNR) is one of the most popular reconstruction quality measurements of lossy transformation (e.g., image compression, image inpainting). In general, PSNR

is defined via the maximum pixel value (denoted as L) and the mean squared error (MSE) between images. Given the ground truth image I with N pixels and the reconstruction $\hat{I}$, the PSNR between I and $\hat{I}$ are defined as follows:

$$PSNR = 10 \cdot log_{10}\left(\frac{L^2}{\frac{1}{N}\sum_{i=1}^{N}(I_i - \hat{I}_i)^2}\right) \tag{2-20}$$

However, PSNR highly relies on the pixel-level MSE, only caring about the differences between corresponding pixels instead of visual perception, it often leads to poor performance in representing the reconstruction quality in real scenes, where we are usually more concerned with human perceptions. However, due to the necessity to compare with literature works and the lack of completely accurate perceptual metrices, PSNR is still currently the most widely used evaluation criteria.

2.5.2 Structural Similarity Index

Structural similarity index (SSIM) [68] is proposed for measuring the structural similarity between images, based on independent comparisons in terms of luminance, contrast, and structures from the perspective of adapted HSV. Besides, SSIM is a full reference metric, in other words, the measurement or prediction of image quality is based on an initial uncompressed or distortion-free image as reference. The overall definition of SSIM is the following:

$$SSIM = \frac{2(\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2-21}$$

Where μ_x, μ_y and σ_{xy} stand for the local means, standard deviations, and cross-covariance for the image x.

2.5.3 No-reference Ma

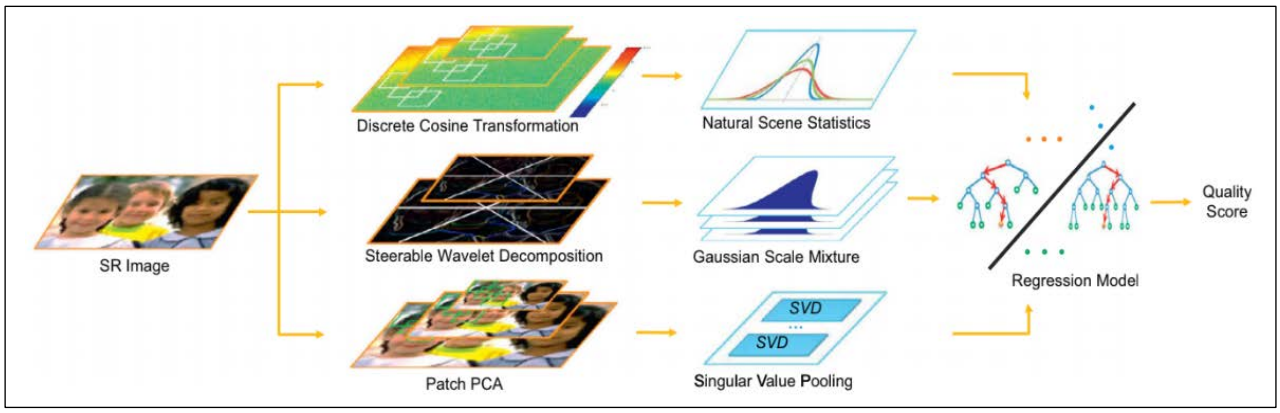


Figure 2-10: Main steps of no-reference metric Ma [52].

Ma et al. [52] proposed a no-reference metric for measuring image perceptual quality by learning from visual perceptual scores and directly predicting the quality scores without ground-truth images. The metrices are especially useful in the case when ground truth images are not available. Specifically, they design three types of low-level statistical features in both spatial and frequency domains to quantify artifacts, and learn a two-stage regression model to predict the quality scores of generated images without referring ground-truth images. **Figure 2-10** illustrates the main steps of the proposed no-reference metric. For each input generated image, statistics computed from the spatial and frequency domains are used as features to represent generated images. Each set of extracted features are trained in separate ensemble regression trees, and a linear regression model is used to predict a quality score by learning from a large number of visual perceptual scores.

2.5.4 Natural Image Quality Evaluator

Natural Image Quality Evaluator (NIQE) [69] is another popular no-reference image quality metric making use of measurable deviations from statistical regularities observed in natural images, without exposure to distorted images. NIQE is based on the construction of a "quality aware" collection of statistical features based on a simple and successful space domain natural scene statistic (NSS) model.

Concretely, it is applied by computing a number of identical NSS features from patches of the same size $P \times P$ from the image to be quality analyzed, fitting them with the multivariate Gaussian (MVG) model, then comparing its MVG fit to the natural MVG model. The quality of the distorted image is expressed as the distance between the quality aware NSS feature model and the MVG fit to the features extracted from the distorted image:

$$D(v_1, v_2, \sum_1, \sum_2) = \sqrt{\left((v_1 - v_2)^T \left(\frac{\sum_1 + \sum_2}{2}\right)^{-1} (v_1 - v_2)\right)} \tag{2-22}$$

Where v_1, v_2 and $\sum_1, \sum_2$ are the mean vectors and covariance matrices of the natural MVG model and the distorted image's MVG model.

The basic NIQE model for calculating perceptual score is trained on large databases of human opinions of distorted nature images, however, we can customize the model by changing the image database for training, which makes the metric to have excellent generalization ability.

Chapter 3. Semantically Correct High-resolution Interpolation

In order to produce a number of semantically correct and high-resolution CT interpolations between known CT slices, we propose a novel neural network “High-resolution interpolation network (HRINet)”. Interpolations produced by HRINet are filled with fine details and sharp edges. To further improve the results in terms of semantic correctness, we apply alternatively supervising training which balance the effect of supervised training and unsupervised training.

3.1 High-resolution Interpolation Network

Though ACAI and other existing networks perform well in interpolation related tasks in the size of 32^2 or 64^2, none of them focus on high-resolution images. When the size of the input becomes relatively large, for example, 256^2 or more, fine details and texture of input images can hardly be interpolated correctly. The results sometimes are blurry and not realistic.

High-resolution interpolation network (HRINet) is the network that we proposed dedicated to producing high-resolution interpolations for CT slices. In this section, we first explain the motivation, and then introduce the main principle of HRINet. Finally, we explicitly elaborate the state-of-the-art technologies involved in HRINet and the reasons we introduce.

3.1.1 Network Architecture

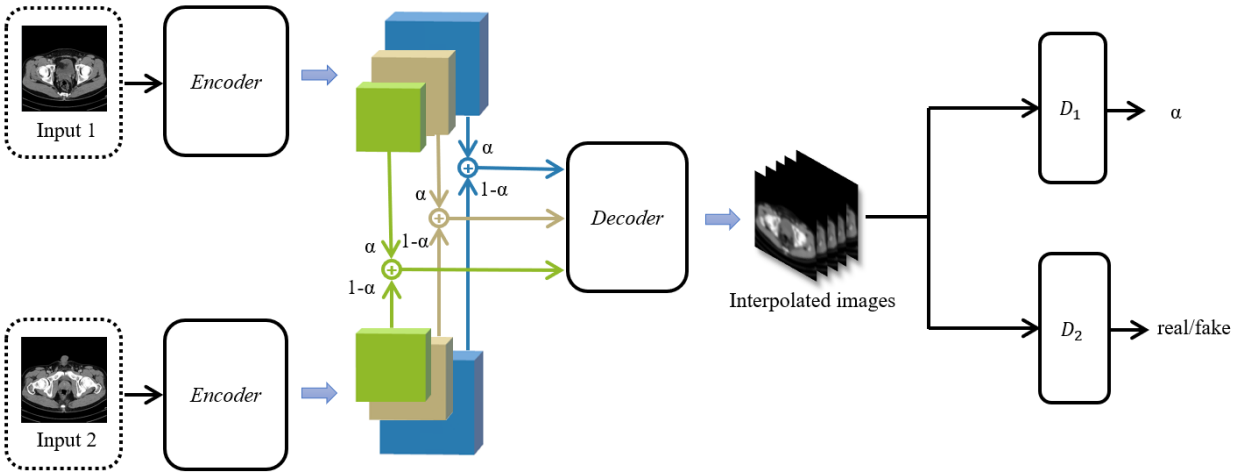


Figure 3-1: The overview of our model. An encoder is trained to produce low-level feature maps which are interpolated with interpolation coefficient α, and the decoder tries to disentangle the latent representation and mapping it into data space. D_1 is the discriminator tries to predict α, and D_2 is trained to distinguish whether the input is real or not. The overall autoencoder can be regarded as Generator, trained to making the D_1 output $\alpha = 0$ consistently.

The architecture of HRINet is illustrated in **Figure 3-1**. As mentioned in **1.2**, the generator contains encoder, decoder and feature fusion mechanism, the discrimination mechanism includes D_1 and D_2 and give feedbacks to the generator according to the overall quality, texture sharpness, structural correctness and semantic smoothness extend of produced interpolations.

The input images are chosen from our CT dataset. Every time the interpolating operation is done between two input images as a pair of end points during training, for example, the *input 1* (denoted as x_1) and *input 2* (denoted as x_1) in **Figure 3-1**. An encoder is trained to produce low-level feature maps instead of latent code for a pair of input images. The feature maps of two data points are mixed according to the interpolation coefficient α, sampled from the uniform distribution $\alpha \in \mathcal{U}(0, I)$. This process can be described as $\alpha f_\theta(x_1) + (1-\alpha) f_\theta(x_2)$. Then mixed low-level feature maps are decoded by a decoder, this is a reversed process of what happened in the encoder, it maps the latent representation into data space, so the output of decoder are also images denoted as interpolations.

D_1 is a discriminator tries to predict input coefficient α for every interpolation. The goal of D_1 is to encourage interpolated data points to appear semantically smooth, ideally, they are located somewhere on the path when traversing endpoints on the manifold. And the autoencoder is trained to fool the D_1 to think that α is zero consistently, in other words, making the D_1 to always think that an interpolated input is non-interpolated. D_2 is the other discriminator trained to distinguish whether the input is real images or interpolations. The purpose of introducing D_2 is to further improve the quality of results, and encourage the autoencoder to produce realistic images, more specifically, to be indistinguishable from real data points in terms of texture, style and layout, etc.

Thus, the overall autoencoder can be regarded as a generator, and there exists two adversarial games between generator (G) and discriminators (D_1 and D_2). Assuming in the end the autoencoder successfully "wins" these games by producing interpolated results which are indistinguishable from real data, that means the autoencoder is capable of producing semantically smooth and high quality interpolations under the "supervision" of discriminators.

3.1.2 Low-level Feature Maps Mixing

Nearly all other autoencoder based models such as ACAI trained the encoder to produce latent codes for the input, which are usually low-dimensional vectors, and then by linear blending them to achieve the effect of abstract feature fusion in latent space before decoded by corresponding decoder. One of the problems is that the latent codes produced in that way are highly entangled, in other words, their physical meanings are not interpretable to us and can hardly be constrained or customized. Besides, the highly compressed latent representations lost too much information in the process of dimensionality reduction by encoder. Fine details like texture, edges or other subtleties are reluctant to be reconstructed or interpolated.

Therefore, instead of simply mixing the latent codes, we proposed a novel feature map mixing approach. Concretely, we mixed the low-level feature maps that intercepted from the encoder, as illustrated in **Figure 3-2**. Those low-level feature maps $\phi_{i,j}(x)$ are linearly mixed in term of interpolation coefficient α. Specifically, i, j stands for the feature map after the i^{th} pooling and j^{th} convolution. This process can be described as the following.

$$\hat{\phi}_{i,j}(\hat{x}_\alpha) = \underset{\theta,\phi}{Concat}\left[\alpha\phi_{i,j}(x_1) + (1-\alpha)\phi_{i,j}(x_2)\right] \tag{3-1}$$

The mixed feature maps are concatenated with the output of corresponding decoder block, for example, "DE5" and "DE6" takes the mixed feature maps from "EN5" and "EN6", the feature delivery between "EN7" and "DE7" can be regarded as ordinary latent codes fusion.

The new features mixing method we proposed would blend the latent features in a more hierarchical way. In the diagram of **Figure 3-2**, the mixing operation happens in three different size of feature maps, and these feature maps with various sizes and channels would represent different levels of features or characteristics of input images. The larger the size of the feature map is, the more subtle and local features will be captured. Thus, compared with simply mixing the latent code [44][45], our method makes better use of hierarchical attributes of the features. And the bottom-up reconstruction strategy recovers more details from the mixed feature maps. Besides, our method has a lower information compressed ratio and information entropy among autoencoders, it means more information is delivered from encoder to decoder and leads to better reconstruction results.

The reason we chose low-level feature maps for feature fusion is that it captures generic features of the inputs, like rough shape or topology, while high-level feature maps contain more concrete subtleties and are close to data space. Mixing those high-level feature maps results in interpolating in data space rather than latent space, in that case, the autoencoder cannot access the manifold during training. Interpolations produced in this way are not realistic and they are apparently distinguishable from the dataset.

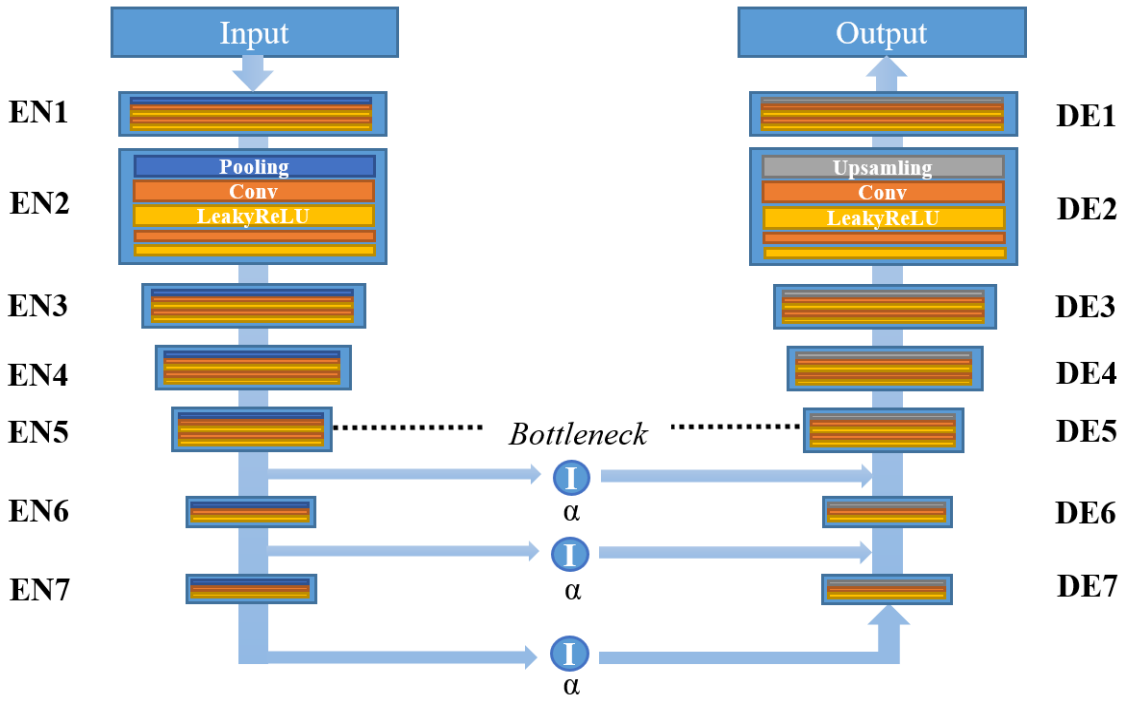


Figure 3-2: Network architecture of the autoencoder part. "EN" or "DE" stands for encoder or decoder block. The low-level feature maps mixing is done beyond the autoencoder part, and $\boldsymbol{\alpha}$ is interpolation coefficient. Bottleneck is where the numbers of channels getting decreased dramatically.

In addition, we get rid of all batch normalization (BN) layers within the autoencoder for the reason that BN layers have been proved are more likely to bring artifacts when the network is deeper and trained under adversarial framework [21]. Because BN layers normalize the features using mean and variance in a batch during training and use estimated mean and variance of the whole training dataset during testing. When the statistics of training and testing datasets differ a lot, BN layers tend to introduce unpleasant artifacts and limit the generalization ability. Removing BN layers helps to improve generalization ability and to reduce computational complexity of the model.

3.1.3 PatchGAN Regularizer

In the architecture of the HRINet, D_2 is a Markovian discriminator (PatchGAN) firstly proposed by Zhu et al. [22][23]. PatchGAN focus on the local response of the input image x, by mapping the whole image into $N \times N$ patches, which are also called receptive fields. Each patch $x_{i,j}$ corresponds the single local prediction $p_{i,j}$made by PatchGAN as illustrated in **Figure 3-3**, where $i, j \in [1, N]$. The length of receptive field ℓ_{n+1} after the n^{th} convolution can be deducted by the following formula:

$$\ell_{n+1} = (\ell_n - 1) * strides + kernel_{size} \tag{3-2}$$

Thus, the ultimate prediction that D_2 made after the n times of convolution is:

$$p_{n+1}^{\ell} = \frac{1}{W_N \times H_N} \sum_{i=1}^{W_N} \sum_{j=1}^{H_N} \left(D_{2_\psi} \left(\hat{I}_\alpha \; or \; I \right)_{i,j} \right) \tag{3-3}$$

In general, PatchGAN determines whether every patch of image is “real” or “fake”, whereas regular GAN discriminator (ImageGAN) predicts the authenticity of the whole image and PixelGAN makes predictions for every single pixel. The comparison between them is shown in **Figure 3-4**.

We applied PatchGAN for discriminator because it can promote shape output for the generator. Because PatchGAN focuses on local response of images rather than global one, it is good for modeling only high-frequency components, whereas low-frequency counterparts are restricted by L2 distance among the autoencoder. And it suffers less impact from the whole semantics of images, making it a better regularized discriminator for regularizing details and texture.

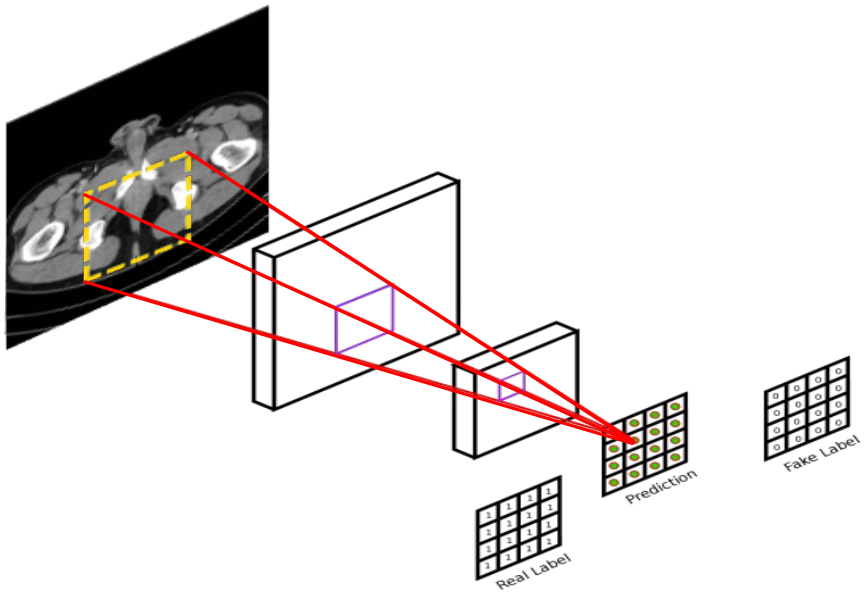


Figure 3-3: The principle of PatchGAN. Each patch is mapped into a single prediction of the output vector of D_2.

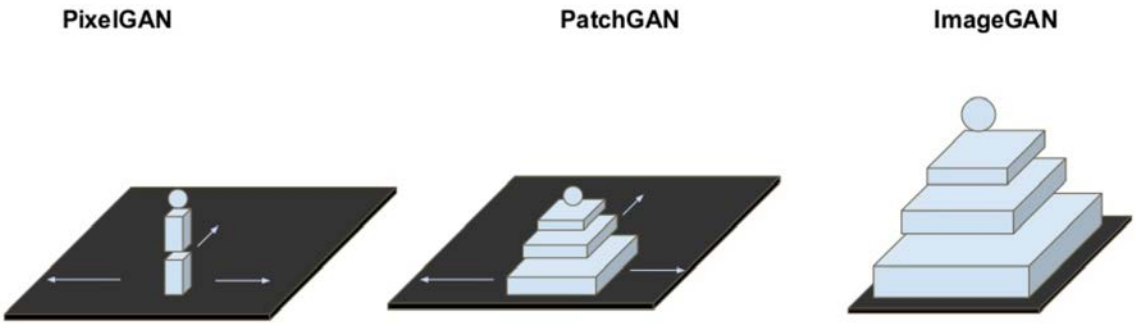


Figure 3-4: The comparison of PixelGAN, PatchGAN and ImageGAN.

3.2 Alternatively Supervising Training

In order to produce visually pleasing interpolations, more constraints need to be added during training to further improve the performance of the generator. It is unavoidable that the generator tends to generate images with incorrect shape or details as illustrated in **Figure 1 (c)**. In practice there are numerous paths connecting two data points $A, B \in \mathbb{R}^{d_x}$, denoted as $\wp_i: A \to B$, the range of i goes from 0 to $+\infty$. And only one of them $\wp_K$ corresponds to the ground truth of interpolations among all possibilities, which is what we desired. Typically, the generator would randomly choose one that the prior is inaccessible to us. And that is very hard to control in the latent space, because the generator does not know which path to take while traversing on the manifold is considered "correct" by human cognition, since we never know the real distribution of the whole dataset in the latent space.

To make our results becomes more realistic in terms of perception quality, we penalize the generator when choosing an "inappropriate" path between end points for interpolating. This is achieved by having the generator get access to the real data space, and minimize a kind of distance between interpolations and ground truth. However, having the generator to be overly trained in data space might weaken the embedding ability, which is not what we expected. To balance both impacts, we proposed alternatively supervising training methods to make the generator produce semantic meaningful results. Specifically, we make the generator train in two different stages: the unsupervised training step (Step I) and supervised training stage (Step II).

3.2.1 Unsupervised Training Step (Step I)

In unsupervised training step, the generator (*G*) is trained with discriminators D_1 (parametrized by ω) and D_2 (parametrized by ψ). We use $f_\theta(.)$ and $g_\phi(.)$ to denote the encoder (parametrized by θ) and decoder (parametrized by ϕ) within *G*, the encoder and decoder are trained simultaneously with some kinds of loss function. For a given interpolation coefficient $\alpha \in [0,1]$, the convex combination of latent code can be written by: $\hat{z}_\alpha = \alpha f_\theta(x_1) + (1-\alpha) f_\theta(x_2)$, so the interpolated result is $\hat{x}_\alpha = g_\phi(\hat{z}_\alpha)$.

The overall objective for generator in Step I is:

$$\mathcal{L}_{ae}^{I} = \underbrace{\mathcal{L}_{cont}}_{content\ loss} + \underbrace{\lambda \underbrace{\mathcal{L}_{ac}}_{constrain\ loss}}_{adversarial} + \underbrace{\beta \underbrace{\mathcal{L}_{Gen}}_{adversarial\ loss}} \tag{3-4}$$

The details of content loss $\mathcal{L}_{cont}$ are discussed in **3.2.3**, for adversarial constrain loss $\mathcal{L}_{ac}$ and general adversarial constrain loss $\mathcal{L}_{Gen}$, the definitions are the following.

$$\mathcal{L}_{ac} = \left|\left|D_{1_\omega}(\hat{x}_\alpha)\right|\right|^2 \tag{3-5}$$

$$\mathcal{L}_{Gen} = log\left(1 - D_{2_\psi}(\hat{x}_\alpha)\right) \tag{3-6}$$

For adversarial constraint loss, we keep the same loss item in ACAI, which purpose is trying to fool the D_1 to think that the input is always non-interpolated one. The adversarial loss is a common

generative loss in GANs, making *G* generate indistinguishable results for D_2, so as to further encourage the network to favor solutions that reside on the manifold.

The losses of D_1 and D_2 are the following:

$$\mathcal{L}_{d_1=}||D_{1_\omega}(\gamma x + (1-\gamma)g_\phi(f(x))||^2 + \left|\left|D_{1_\omega}(\hat{x}_\alpha) - \alpha\right|\right|^2 \tag{3-7}$$

$$\mathcal{L}_{d_2=} log\left(1 - D_{2_\psi}(x)\right) + log\left(D_{2_\psi}(\hat{x}_\alpha)\right) \tag{3-8}$$

where, γ is a scalar hyperparameter, the item $||D_{1_\omega}(\gamma x + (1-\gamma)De_\phi(En_\theta(x))||^2$ helps stabilize the training process in the early period of training, that is , when the autoencoder's reconstructions are relatively poor [26]. And the item $||D_{1_\omega}(\hat{x}_\alpha) - \alpha||^2$ penalizes the bias of predicting the α from $\hat{x}_\alpha$ for D_1.

3.2.2 Supervised Training Step (Step II)

In Step II, we customize every input batch to be a group of *n* consecutive CT slices chosen from the same patient. For each patient, the space between slices is a known constant. The first and last slices of the group would be two inputs images for interpolation, which are corresponding to a pair of endpoints in dataspace, the semantic distance on the manifold is relatively farther when compared with two adjacent slices for interpolation, we describe this case as "interpolating on a large gap". The generator will produce $n-2$ interpolations (endpoints are excluded) and α is equally spaced from $(0, 1)$. Then we try to minimize the distance between the interpolated result and corresponding real slices to eliminate the unrealistic artifacts, as shown in **Figure 3-5**. The optimization objective of generator in Step II is:

$$\mathcal{L}_{ae}^{II} = \frac{\epsilon}{n-2}\sum_{i=0}^{n-2}\mathcal{L}_{cont}^{i} \tag{3-9}$$

The format of content loss $\mathcal{L}_{cont}^{i}$ are described in **3.2.3**, and ϵ is a weight scalar hyperparameter. Typically, the further apart the two endpoints are on the manifold, the harder to do semantically smooth interpolation between them. The purpose of training the model to interpolate on large gaps is to improve the robustness and stability of the model, and eliminate those artifacts caused by choosing an "inappropriate" path while traversing on the manifold.

These two training steps run alternatively during training, to offset the shortcomings of the either when used alone. And we found that the regularization effects of both steps all have great impacts on the generator. The whole training process can be described as below.

ALGORITHM 1: Alternatively supervising training

Initialize $G_{\theta,\phi}$, D_{1_ω} , D_{2_ψ}

for *i = 0,1,…,N:*

  if *i* is even *(unsupervised training)*, **do**

    Input: Two endpoint images $x_1, x_2 \in \mathbb{R}^{n\times m\times 1}$ and interpolation coefficient α.

    while not converge, **do**:

      $x_\alpha \leftarrow G_{\theta,\phi}(f_\alpha{:}\sim(x_1, Ix_2))$

      $L \leftarrow \left|\left|x - G_{\theta,\phi}(x)\right|\right|_2^2 + \left|\left|D_{1_\omega}(\hat{x}_\alpha)\right|\right|^2 + \mathbb{E}_{x\sim\mathbb{P}_g}[\log\left(1 - D_{2_\psi}(\hat{x}_\alpha)\right)]$

    end

  else *(supervised training)*, **do**

    Input: A series of sequential images $x_1, \ldots, x_n \in \mathbb{R}^{n\times m\times 1}$ and interpolation coefficient $\alpha \in \mathcal{U}(0, I)$, a pretrained network P_φ

    while not converge, **do**:

      $L \leftarrow \frac{\lambda}{n}\sum ||P_\varphi(x) - P_\varphi(\hat{x}_\alpha)||^2$

    end

end

Besides, the alternative training shows the possibility to combine two or more totally different training steps in the same model, that means several optimization methods could be applied simultaneously without conflict. It has the same effect on including all training steps in once back propagation but might become impractical due to occupying too much memory.

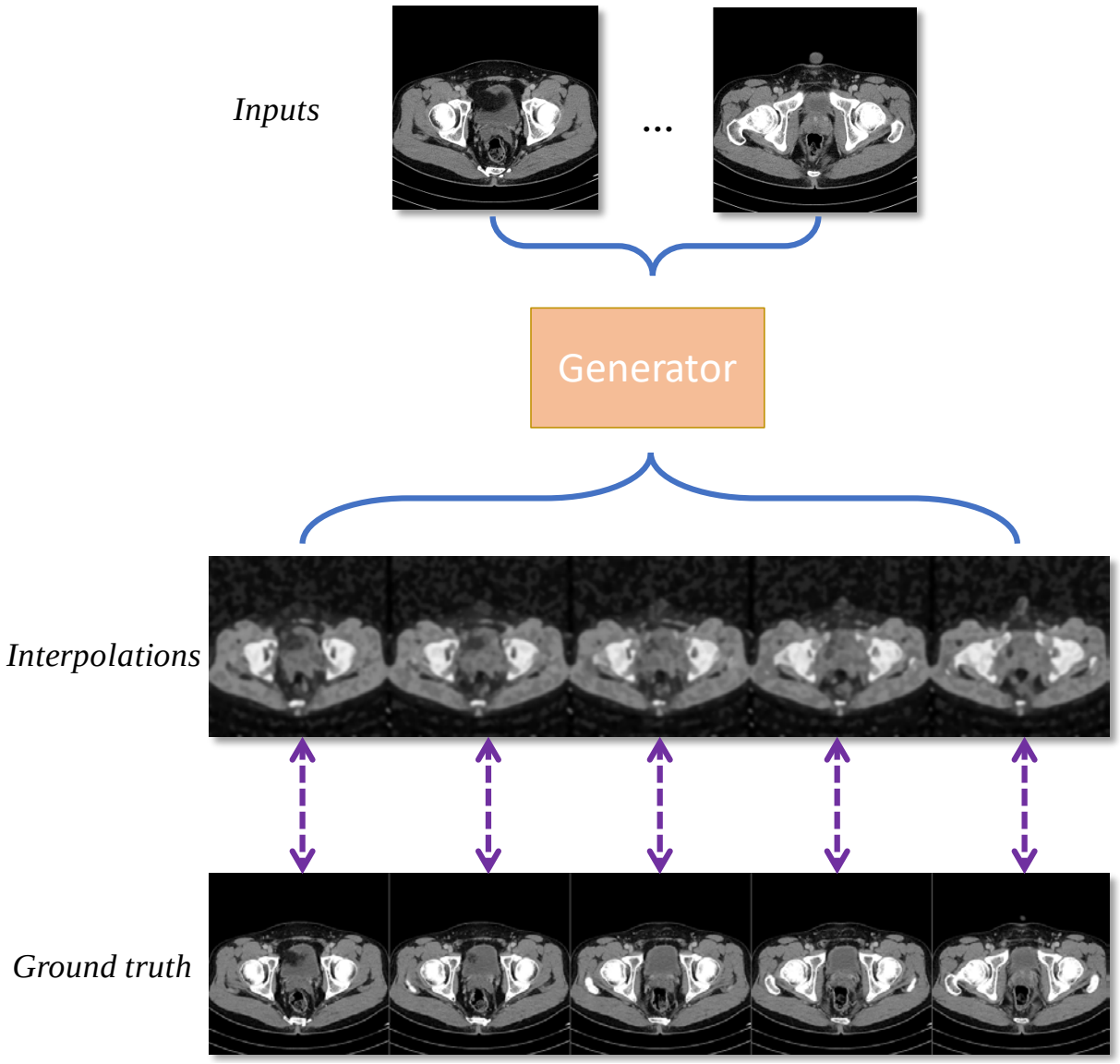


Figure 3-5: The principle optimization objective in Supervised training step (Step II). For a pair of inputs with large gap, the content loss $\mathcal{L}_{cont}$ is calculated between their interpolations and corresponding ground truth.

3.2.3 Content Loss Function

There are a variety of ways to calculate content loss in image generation tasks, the two most popular are pixel-wise MSE loss [32][47] and perceptual loss [20][48][49]. We proposed both choices for content loss in our model, if the pixel-wise MSE loss is applied, we named it as HRINet-MSE, the same as the corresponding item in ACAI [44]. Likewise, if the perceptual loss is applied, we call that HRINet-Perceptual.

The perceptual loss is defined on another pre-trained network p_φ (parametrized by φ). With $p_\varphi^{i,j}$ we indicate the feature map obtained by the jth convolution (after activation) before the ith pooling layer within the loss network. The perceptual loss is calculated as the euclidean distance between the feature representations of generated image $\hat{I}_\alpha$ and the reference image I:

$$\mathcal{L}_{feat}^{p_\varphi^{i,j}} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(p_\varphi^{i,j}\left(\hat{I}_\alpha\right)_{x,y} - p_\varphi^{i,j}(I)_{x,y} \right) \tag{3-10}$$

Here $W_{i,j}$ and $H_{i,j}$ describe the dimensions of the respective feature maps within the loss network $p_\varphi^{i,j}$.

Instead of using the standard pre-trained VGG network [50] for perceptual loss calculation, which is commonly used in natural image super-resolution [20][21], we used a pre-trained encoder to be a network for loss calculation. The encoder part is copied and separated from the original autoencoder, getting pre-trained in training stage I. Details of how we pretrained it are elaborated in **3.3.2.4.** The network is fixed in subsequent training and only for calculating perceptual loss, which means gradients stop to propagate through p_φ, and all parameters in p_φ would not be updated during training.

Thus, the content loss $\mathcal{L}_{cont}$ in unsupervised training step (Step I) is:

$$\mathcal{L}_{cont} = \underbrace{\left|\left|x - g_\phi\left(f_\theta(x)\right)\right|\right|^2}_{MSE\ content\ loss} \ or \ \underbrace{||p_\varphi(x) - p_\varphi(g_\phi\left(f_\theta(x)\right))||^2}_{perceptual\ content\ loss} \tag{3-11}$$

Depending on whether pixel-wise MSE content loss is applied or perceptual content loss is applied. Note that during this step, the content loss can be regarded as reconstruction error where x and $\mathrm{g}_\phi\left(\mathrm{f}_\theta(\mathrm{x})\right)$ refer to the input image and the reconstructed result. Step I is still be considered unsupervised training step relatively because the generated interpolation $\hat{x}_\alpha$ are only for the purpose of calculating regularization items $\mathcal{L}_{ac}$ and $\mathcal{L}_{Gen}$ instead of being used to minimize the distance between ground truth as illustrated in **3.2.2**.

Similarly, the content loss $\mathcal{L}_{cont}^{i}$ in the supervised training step (Step II) is defined as:

$$\mathcal{L}_{cont}^{i} = \underbrace{\left|\left|x - \hat{x}_\alpha\right|\right|^2}_{MSE\ content\ loss} \ or \ \underbrace{\left|\left|P_\varphi(x) - P_\varphi(\hat{x}_\alpha)\right|\right|^2}_{perceptual\ content\ loss} \tag{3-12}$$

Then we could rewrite the optimization objective of generator for HRINet-MSE or HRINet-Perceptual in Step II to be:

$$\mathcal{L}_{ae}^{II} = \frac{\epsilon}{n-2}\sum_{i=0}^{n-2}\left(\left|\left|x-\hat{x}_{\alpha}\right|\right|^{2} \ or \ \left|\left|P_{\varphi}(x)-P_{\varphi}(\hat{x}_{\alpha})\right|\right|^{2}\right) \tag{3-13}$$

3.3 Experiments

In the section, we first show the overview of our dataset, and then explore the role each main component in our model to elaborate its influence when adjusting some hyperparameters of which during training. Quantitatively and qualitatively results explain the reason we set those hyperparameters to specified values.

3.3.1 Dataset Description

The Dataset contains around 1200 CT slices from 10 different patients provided by the *Korea University Anam Hospital*. Patients are among different ages and gender. All slices are taken from the torso cross section, including abdomen, pelvis, etc. The slice thickness varies from *3mm* to *5mm*. The original data is resized into 512^2, and the relative length of each pixel cube is from *0.53mm* to *0.69mm*, calculated by the actual scanning range. Detailed descriptions of each patient's CT data is listed in **Table 3-1**. The ratio for the purpose of training and testing is 9:2. An overview of the dataset can be found in **Figure 3-6.**

Table 3-1: Description of the dataset. The rows with shadow are test sets.

Patients number	Number of slices	Space (*mm*)	Actual length (*mm*)	Pixel cube length (*mm*)	Number of interpolations created
1	159	3	272	0.53	5
2	159	3	272	0.53	5
3	159	3	272	0.53	5
4	106	5	353	0.69	7
6	106	5	353	0.69	7
7	106	5	353	0.69	7
8	106	5	353	0.69	7
9	106	5	353	0.69	7
10	106	5	353	0.69	7

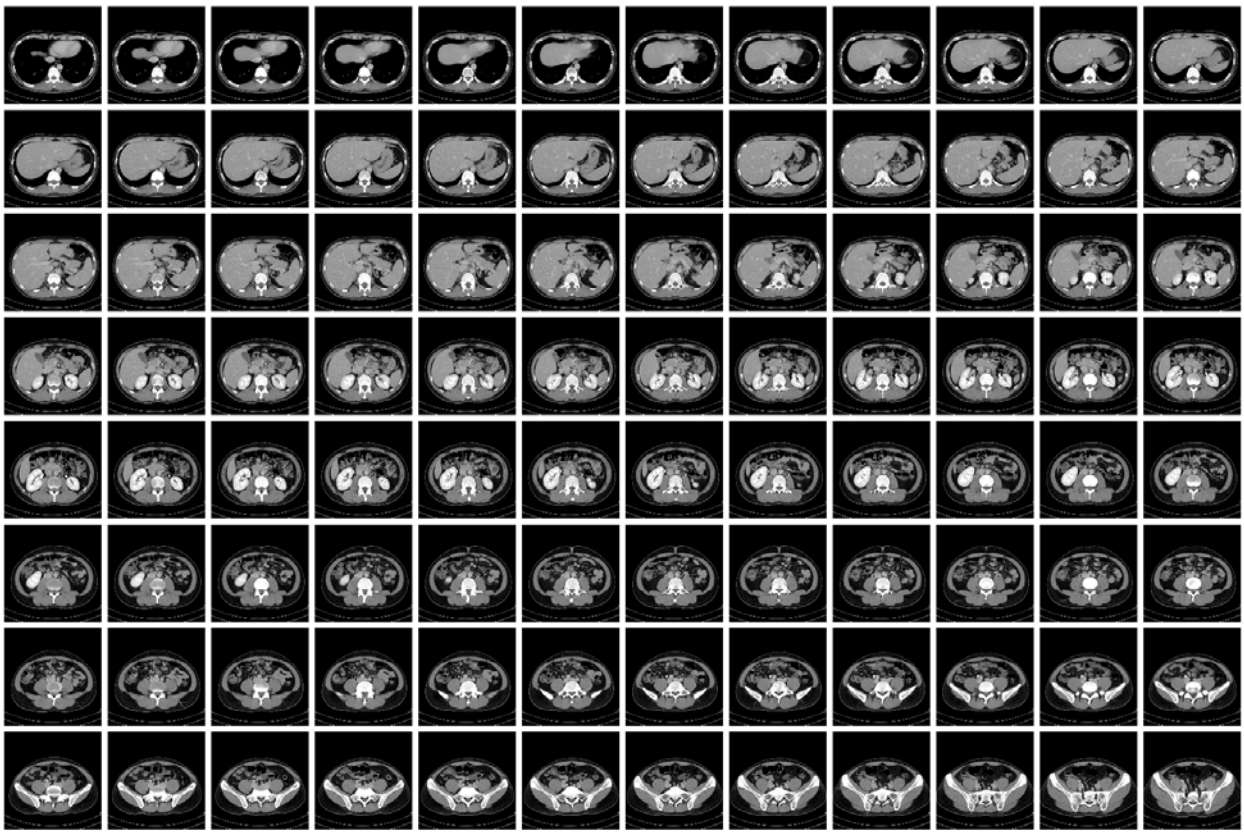


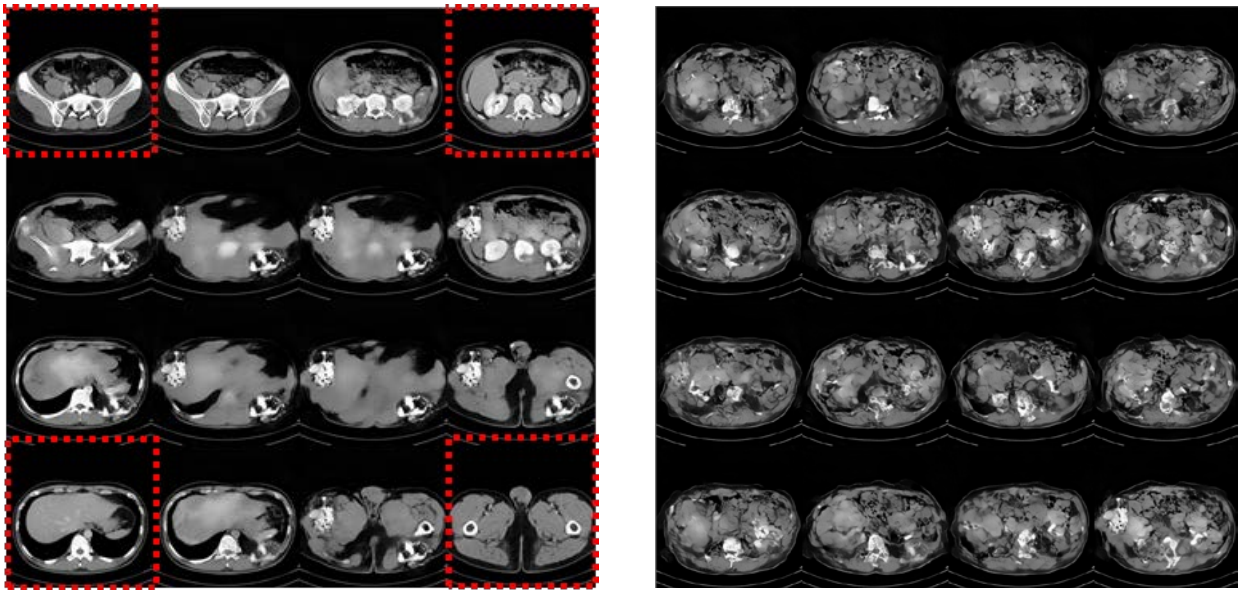
Figure 3-6: Samples from CT slices dataset.

3.3.2 Training Details and Parameters

3.3.2.1 Non-relevant Input Pairs

In the unsupervised training step (Step I), every pair of input image is randomly chosen from the whole CT dataset, which means there is not special constrain about the correlation that the input images must have, for example, they are neither required to be adjacent slices in original dataset nor must be selected from the same patients. We denoted that the input image pair (I_1, I_2) is independent in spatial and practical level.

The reason we used non-relevant images as input during training is that it helps improve the stability and robustness of our model. Although generating interpolations from a pair of unrelated images are meaningless to us, it can make the model, especially the discriminators, get access to the manifold in multiple different ways. For instance, having D_1 to predict input interpolation coefficient α even when the practical meaning of interpolated results is not clear at the human level. In addition, it can also prevent overfitting when the size of the dataset is relatively small. **Figure 3-7 (a)** shows the interpolations generated by unrelated input images which maybe nonsense to human but is beneficial for training, **Figure 3-7 (b)** shows some random samples obtained by first sampling in the latent space based on a reference distribution of input vectors, and then being decoded by decoder. The distribution of these samples can reflect the extent of manifold learning of the generator.



(a) (b)

Figure 3-7: (a) Interpolations without clear meaning generated by unrelated input images (the four corners encircled in red dash lines), shown in the form of 2D-grid interpolation. (b) Random samples based on a reference distribution in latent space.

3.3.2.2 Features Map Sizes for Mixing

The size of feature maps (latent code) that are mixed in the latent space is of vital importance to the performance of the model. Like described in **3.1.2**, feature maps with various sizes and channels represent different levels of features or characteristics of input images, the larger the size of the feature map is, the more subtle and local features will be captured. Selecting inappropriate sizes result in interpolating in data space rather than latent space, or it might blur the outputs. In both cases, the autoencoder cannot access the manifold during training. Interpolations produced in this way are not realistic and they are apparently distinguishable from the dataset.

For 256^2 and 512^2 inputs, we customized our feature maps that are mixed into three different sizes, with the ratio of 16/1, 32/1 and 64/1 of the original input image width, as shown in

Table *3-2*. For contrast, the ACAI model only mixes the latent code with width to be 16/1 of originals.

The visualization of low-level feature maps for an input image is shown in **Figure 3-8**. We can find that larger ones capture more local features and subtleties of the input, like edges or texture. The smaller ones gather more global features including background, topology and structural information. For a given input image in **Figure 3-8**, **Table 3-3** lists examples of feature map with various sizes, all of them are visualized after activation function and are normalized to the range of [0, 1]. In a word, by blending different sizes of low-level feature maps, we make features be fused in a more hierarchical way.

Table 3-2: The contrast of feature map sizes be passed through for interpolation between ACAI and HIRNet.

Network	ACAI		**HRINet**(Ours)	
Input size *(w ×h ×c)*	256×256×1	512×512×1	256×256×1	512×512×1
Feature map size *(w ×h ×c)*	16×16×16	32×32×16	16×16×16	32×32×16
	-	-	8×8×8	16×16×8
	-	-	4×4×8	8×8×4

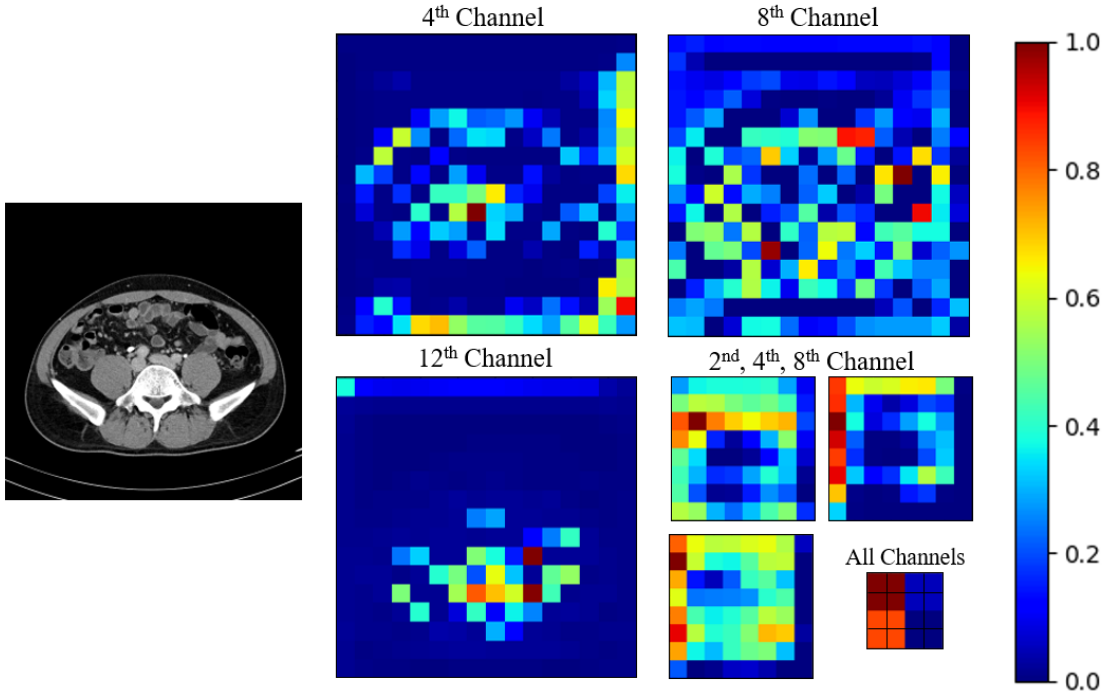
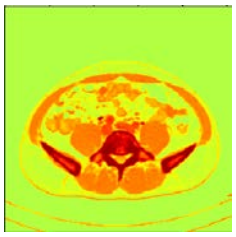
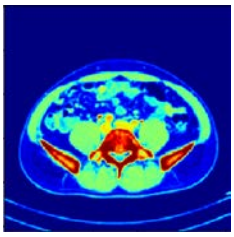
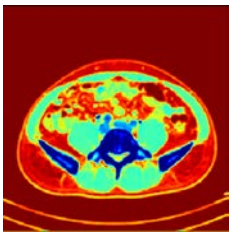
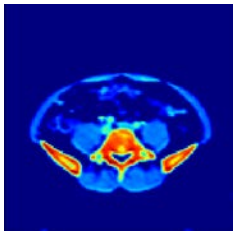
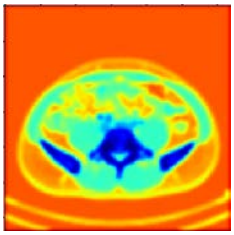
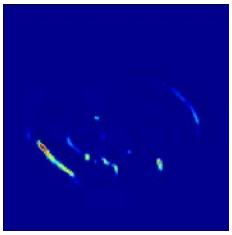
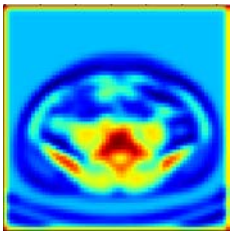
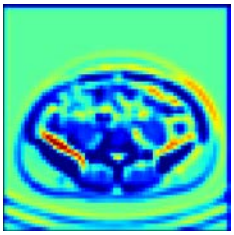
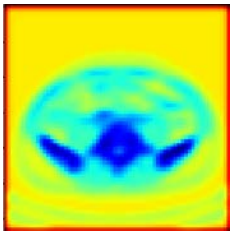
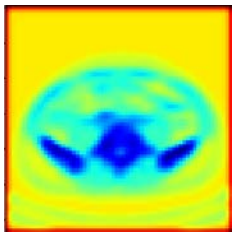
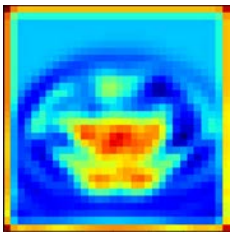
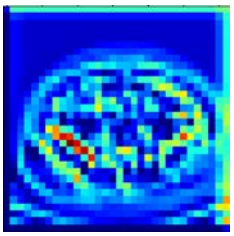
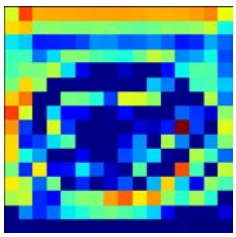
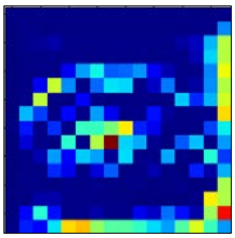
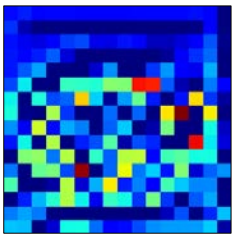
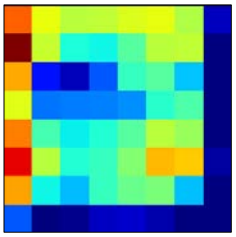
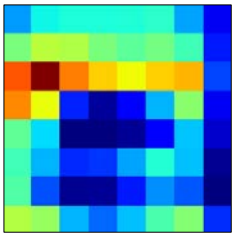
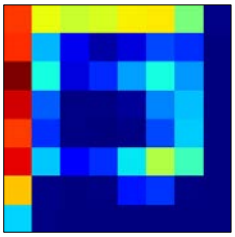


Figure 3-8: Low-level feature maps for a specific input image (left, size of 256^2) with size of 16^2, 8^2 and 4^2 before mixing. All features are normalized to the range of [0, 1].

Table 3-3: Examples of feature maps with various sizes for a specific input image in Figure 3-8, the 2^{th}, 4^{th} and 8^{th} channels are visualized after activation function and are normalized to the range of [0, 1].

Size(W×H)	*2th* Channel	*4th* Channel	*8th* Channel	Legend
256×256				
128×128				
64×64				
32×32				
16×16				
8×8				

3.3.2.3 PatchGAN Receptive Field

In order to make the model to achieve the best status, we try to find the best receptive field parameters for PatchGAN discriminator D_2. We test the effect of varying the patch size N of D_2, and evaluate the performance of our model under different parameters setting, from 1×1 "PixelGAN" to 256×256 "ImageGAN" (for inputs with size of 256^2). **Figure 3-9** manifests that the training collapse with 1×1 PixelGAN, the mixed regions becomes blurry and desaturated under merely one-pixel receptive field. The 16×16 PatchGAN and 256×256 PatchGAN encourage shaper results, but some regions of interpolation are still unclear or having tilting artifacts, makes the results in low qualities. The best results are given by 70×70 PatchGAN, they have the most sharp edges and subtleties. In that case, the D_2 outputs a vector with the dimension of $30\times30\times1$ for 256^2 inputs, the ultimate prediction p_n^ℓ is a scalar and calculated as the average of the output vector.

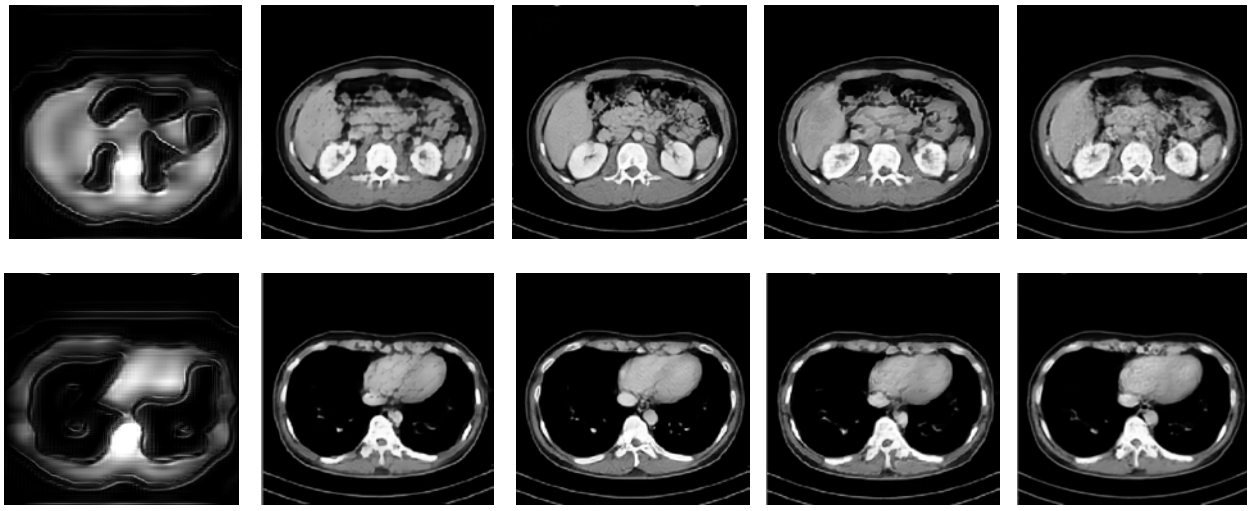


Figure 3-9: Examples from different sizes of PatchGAN receptive field. The receptive field applied for training increases from left to right. The best results are when the patch size equals to 70×70.

3.3.2.4 Training process and parameters setting

The model is parameter sensitive and hard to train at once. To speed up the convergence, we divided our training process into two stages:

In the first stage, we remove the adversarial loss item L_{Gen} and pretrain the generator merely with discriminator D_1 for 5×10^5 iterations, so as to make the generator have ability to generate interpolated results but in low quality.

In the second stage, we used the pre-trained network and made it train with discriminators D_1 and D_2 simultaneously for another 5×10^5 iterations. We set generative adversarial coefficient $\beta = 5\times10^{-4}$ for 256^2 inputs and $\beta = 5\times10^{-3}$ for 512^2 inputs. We also found that β is a key hyperparameters and large β value might mislead the gradient descent direction for the generator that

results in the whole training collapse, even when the generator has preliminary generation ability. Because in the early period of training, when the discriminating ability of D_2 is relatively low, it might give the prediction with huge bias. And the bias is accumulated not only in the gradients of D_2 itself but also the generator when the gradients are doing backpropagation. Setting β to be a very small value can stabilize the training process for both D_2 and G. Besides, to further improve the stability, we applied WGAN-GP framework for GANs part of our model, and made the G get updated once for every five times of updating D_2.

For the supervised training step (step II), we set the weight scalar hyperparameter $\epsilon = 2 \times 10^{-5}$, and had n to be a large enough number $n = 7$ to guarantee that the model is stable and robust enough to meet the larger gap interpolations requirements in practice. In other words, for a series of $n = 7$ sequential slices, the model generates 5 intermediate slices $\widehat{I_2}, \dots, \widehat{I_6}$ which are used for calculating content loss in a round under the promise that I_1 and I_7 are two endpoints for interpolation. The network p for calculating perceptual loss is intercepted from the pre-trained generator in the first stage, specifically, we use the feature maps that engendered after the 8^{th} convolutional layers and before the 4^{th} pooling layers in the encoder. The reason we use the pre-trained encoder for loss network rather than standard VGG-19 [50] is that the VGG network is pretrained on the ImageNet dataset [51] for image classification, so it can used for natural image generation tasks but not suitable for medical image generation. There we create our own version of loss network by intercepting and copying part of the encoder, since the autoencoder (generator) has preliminary generating ability, it means the encoder is eligible for deep feature extraction. Another benefit of that, in our case, the loss network is pre-trained on the same CT dataset as the other components of our model, we do not need to worried about the difference of dataset distribution between pre-trained loss network and generator, which might be a potential problem for natural image generation tasks.

During these two stages, we kept the adversarial constraint coefficient $\lambda = 0.5$ consistently, the same as the original setting in ACAI model [44]. And we set the learning rate to be 1×10^{-4} with weight decay ratio equals to 1×10^{-5}.

Chapter 4. Evaluation

We evaluated the performance of our models without adversarial components (D_1 and D_2) and evaluated the effect of supervised content loss applied in the second stage of alternative training. For contrast, if neither adversarial components nor alternative training are applied during the whole training process, that is, only the PatchGAN discriminator (D_2) is added to the original model and the alternative training part is removed, we named that model to be HRINet (PatchGAN only).

We compared our models with autoencoder and ACAI quantitatively and qualitatively among several tasks including large gaps interpolation, adjacent slices interpolation and features reconstruction, etc. For ablation study, we explored the influence of feature maps mixing, receptive field of PatchGAN and the types of content loss.

4.1 Large Gaps Interpolation

For evaluation on the task of on large gaps interpolation, we keep the same parameter setting as training process, we made $n = 7$, and the models generate 5 intermediate slices $\hat{I}_2, \dots, \hat{I}_6$ for comparison. **Figure 4-1**, **Figure 4-2** and **Figure 4-3** show some visual examples generated by different models.

Figure 4-4 shows qualitative comparison and locally enlarged view. Both PSNR and SSIM are calculated for specific interpolated results which are intercepted from a series of continuous interpolations. The original results can be found in **Figure 4-1** and **Figure 4-3**. PSNR and SSIM quantitatively measure the distortion and structural similarity between the results and originals. However, both quality metrices can hardly represent high-frequency components and sometimes deviate from human perception. PSNR and SSIM both have mean square error items, so MSE-based solutions are more likely to lead to high scores but often result in overly smoothing effects. It corresponds to the results in **Figure 4-4**, where ACAI and HRINet-MSE have spatial MSE loss items and they get higher PSNR/SSIM scores, but clearly much blurrier than HRINet-Perceptual.

We observe that there's a trade-off between the structural correctness and sharpness. Overall, MSE-based solutions are good at minimizing the structural bias like incorrect shapes or positions, and therefore they have better structural correctness. But they are reluctant to capture the high-frequency bias from ground truth, like textures and fine details, they tend to minimize the average of local pixels bias but ignore individual differences. Conversely, perception-based solutions could extract those high-frequency and deep features by a neural network, and produce results with similar texture, but the overall structural information might be lost in the loss network, thus it can hardly retrieve global information.

Besides, **Figure 4-4** also shows the benefits of having the model get access to the real data distribution during supervised training step, no matter in MSE loss way or perceptual loss way. We observe that in large gap interpolation tasks, both ACAI and HRINet (PatchGAN only) cannot correctly "guess" the "trend of bone shape changing" which result in poorly interpolations. In other words, the model unable to find a semantic meaningful path to go through between two endpoints of the manifold among numerous "possible choices", though the interpolating process is still be considered "semantic", it not corresponds to the human perception and thus those interpolations are not realistic. This phenomenon is in line with our assumption described in **3.2**.

Quantitative results are summarized in **Table 4-1,** we calculate average PSNR, SSIM, RMSE and Ma et al. [52] scores for each of the models respectively over one hundred of images. The results show that HRINet-MSE and HRINet-Perceptual often lead to the best or the second-best scores. As expected, the models with spatial MSE loss item (ACAI and HRINet-MSE) perform well in pixel-wise distance metrices (PSNR, SSIM and RMSE), while HRINet-Perceptual and HRINet (PatchGAN only) do well in no-reference perceptual similarity metrices. And perceptual based metrices like Ma et al. [52] has been proved to be closer to human visual judgement.

Table 4-1: Different metrices for evaluating the performance of our model. The best result is highlighted in bold and second-best is underlined. (PSNR/SSIM and Ma et al. are the higher the better, RMSE is the lower the better)

Metrices	**Size**	**AE**	**ACAI**	**HRINet (PatchGAN only)**	**HRINet-MSE**	**HRINet-Perceptual**	**Ground truth**
PSNR	256^2	22.730	<u>22.800</u>	22.130	**25.576**	22.654	∞/1.0
	512^2	21.937	<u>21.968</u>	20.141	**21.824**	20.826	∞/1.0
SSIM	256^2	0.957	<u>0.957</u>	0.951	**0.977**	0.958	1.0
	512^2	0.907	<u>0.909</u>	0.872	**0.909**	0.894	1.0
RMSE	256^2	16.985	16.845	18.072	**12.059**	<u>16.6326</u>	0
	512^2	18.394	**18.290**	22.435	<u>18.305</u>	20.4879	0
***Ma et al.*[52]**	256^2	2.811	2.843	<u>2.882</u>	2.856	**2.883**	2.950
	512^2	2.601	2.577	2.893	<u>2.934</u>	**2.961**	2.911

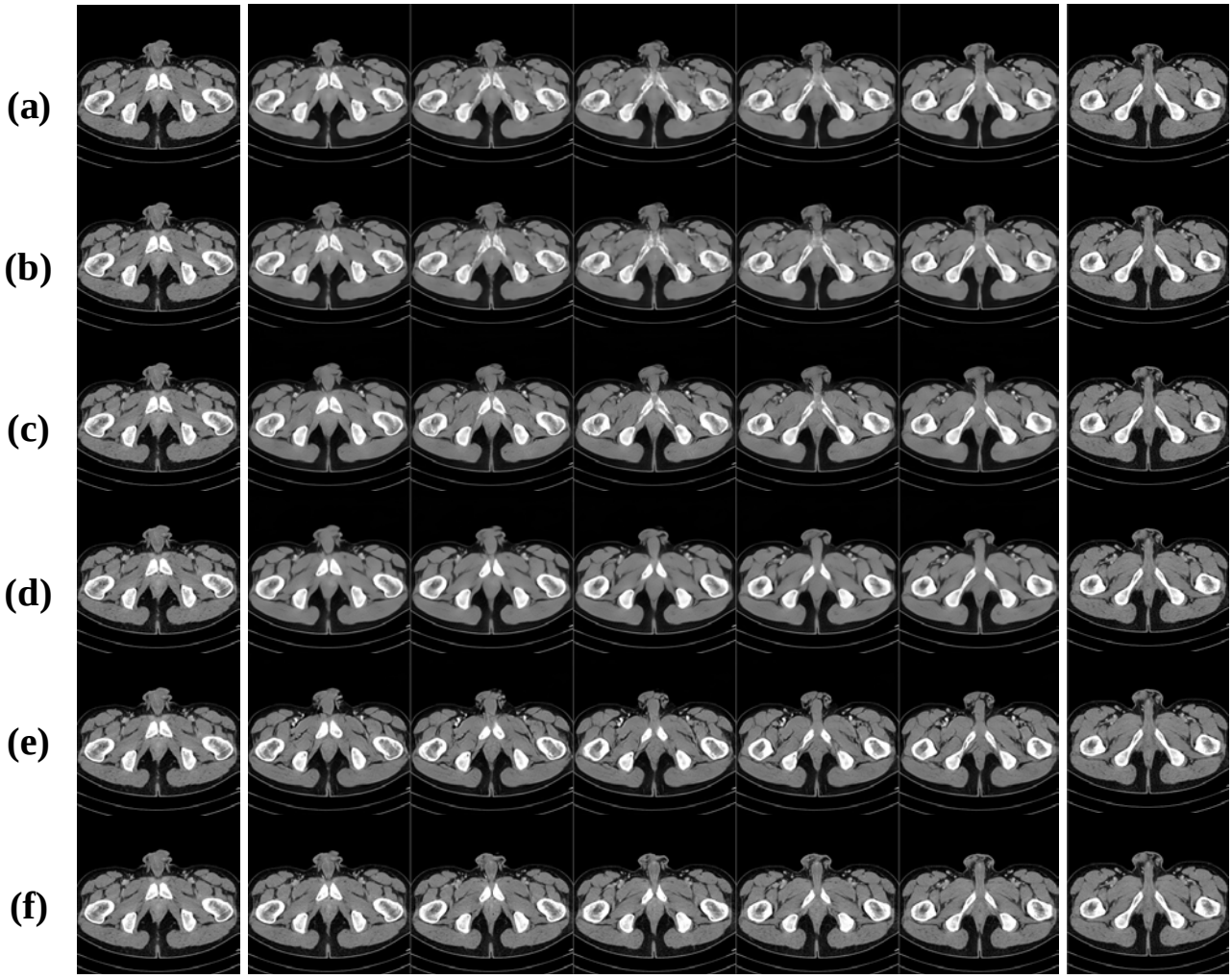


Figure 4-1: Examples of interpolating on a large gap where ground truth is available. The leftmost and rightmost columns are known slices as inputs, and the gap between them is five consecutive slices. Results are produced by (a) AE, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual, (f) Ground truth.

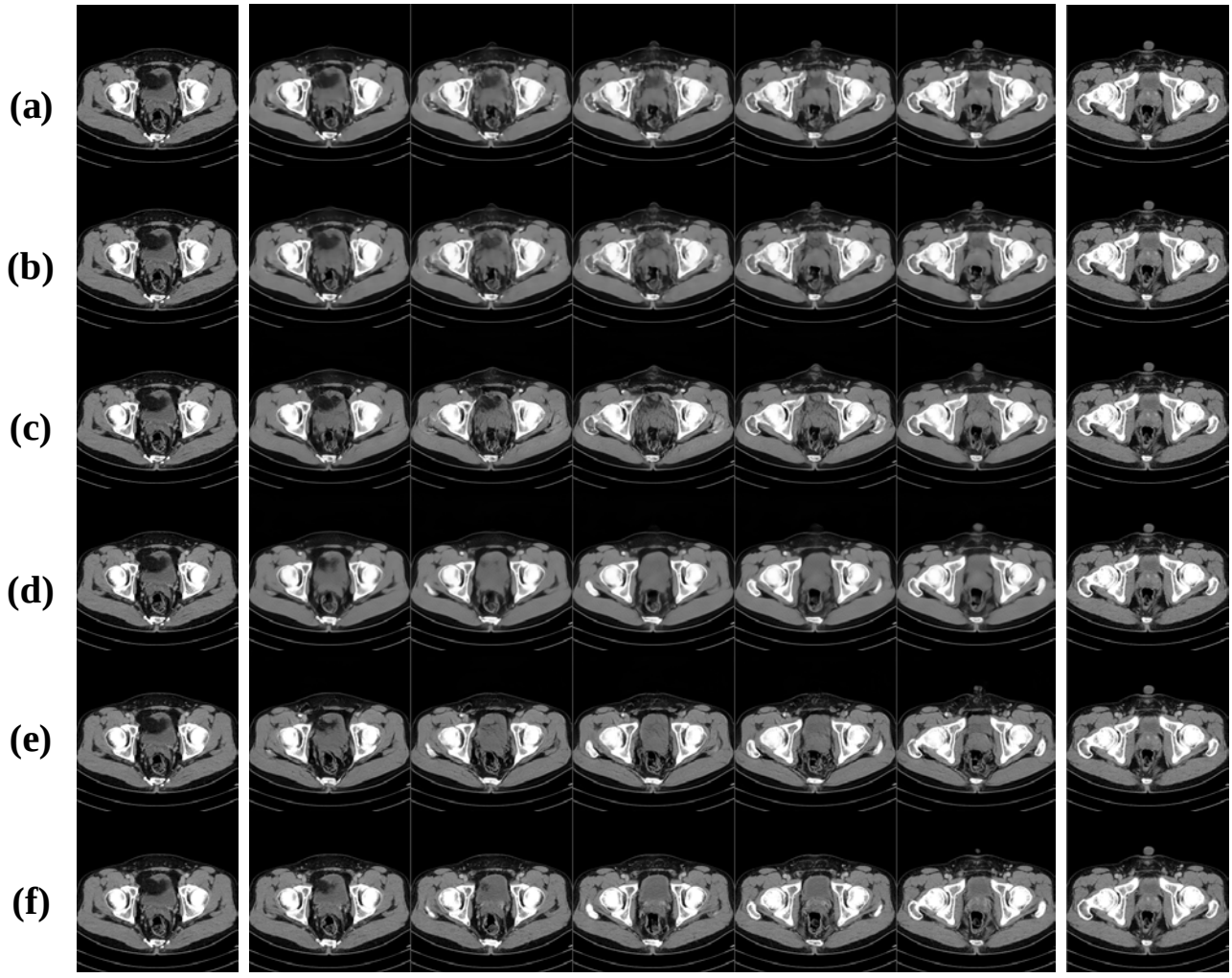


Figure 4-2: Examples of interpolating on a large gap where ground truth is available. The leftmost and rightmost columns are known slices as inputs, and the gap between them is five consecutive slices. Results are produced by (a) AE, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual, (f) Ground truth.

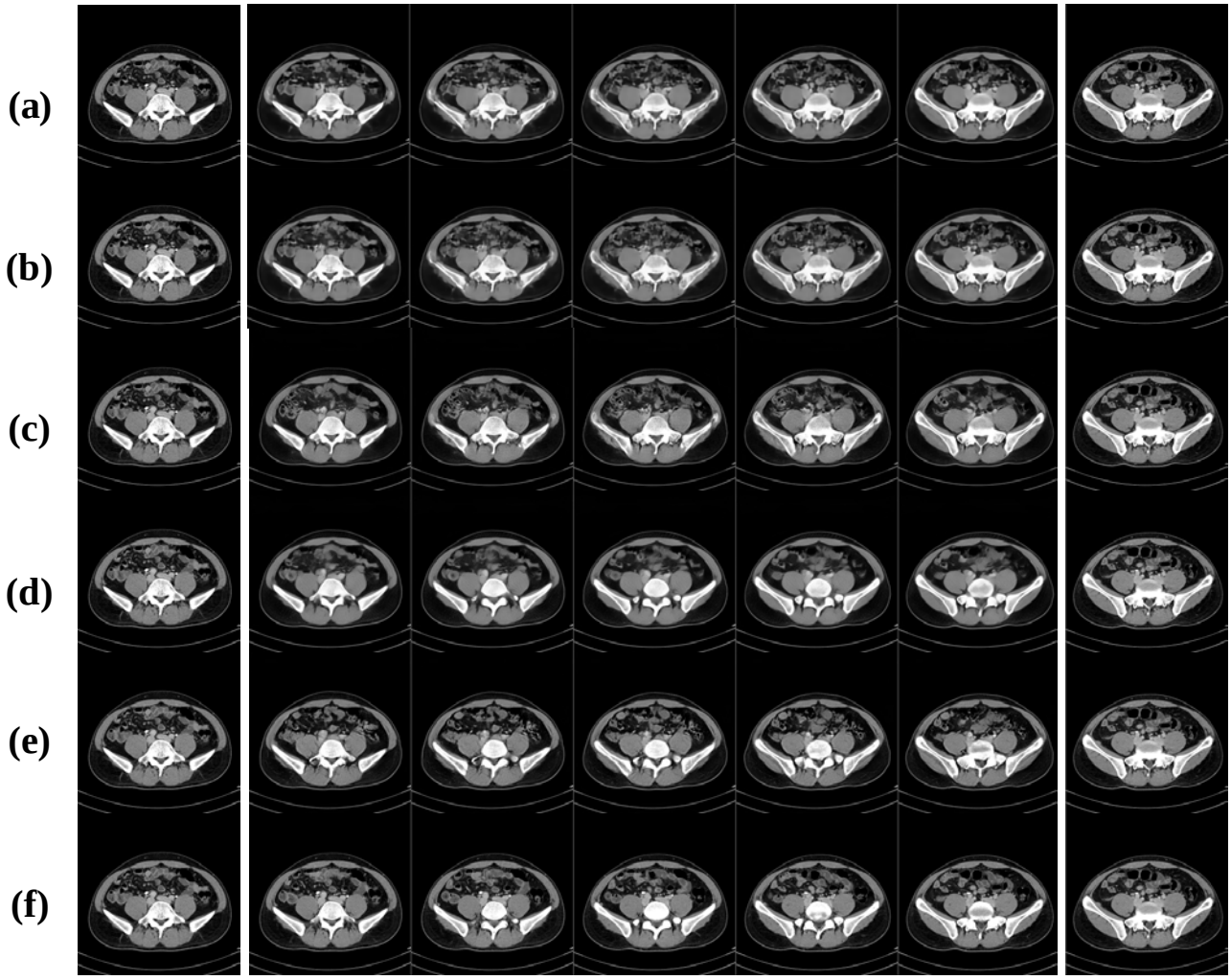


Figure 4-3: Examples of interpolating on a large gap where ground truth is available. The leftmost and rightmost columns are known slices as inputs, and the gap between them is five consecutive slices. Results are produced by (a) AE, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual, (f) Ground truth.

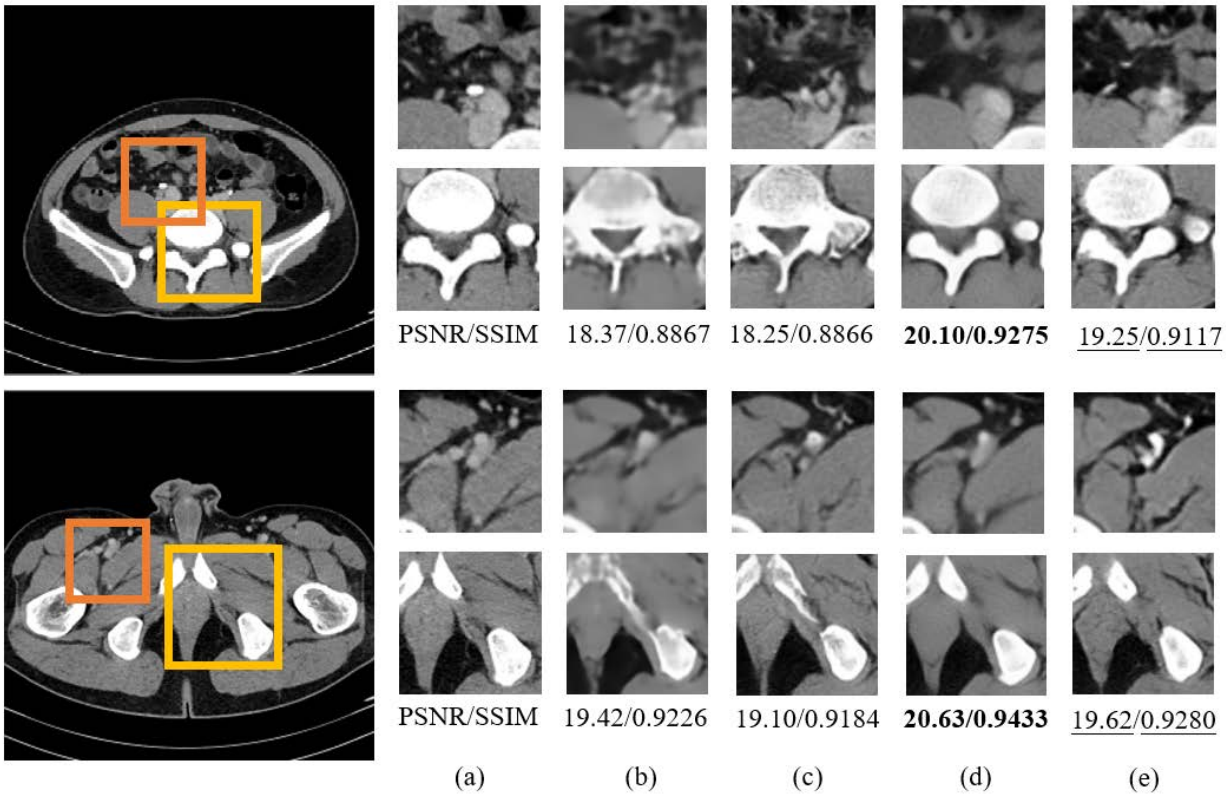


Figure 4-4: Qualitative results and locally enlarged view. The upper left and lower left are ground truth of interpolations, other results are intercepted from a series of continuous interpolations. The best result is highlighted in bold and second best is underlined. (a) Ground truth, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual.

4.2 Adjacent Slices Interpolation

Figure 4-5 and **Figure 4-6** show results of interpolating between spatially adjacent CT slices from the same patients. Note that this is the case that is closest to the actual application when the ground truth is not available, the model runs in a purely unsupervised way. Our model meets the need for creating semantic interpolations between known slices with minimum adjacent distance. Results show that both HRINet-MSE and HRINet-Perceptual are able to create sharp and realistic interpolations. In theory, our model can create as many interpolations as needed as long as smoothly morphing the interpolation coefficient α from 0 to 1.

To evaluate the performance of our model in the current stage, only no-reference metrices like NIQE, Ma et al. [52] are qualified for measurement because of lack of ground truth. **Table 4-2** summarizes the quantitative comparison among AE, ACAI, HRINet (PatchGAN only), HRINet-MSE and HRINet-Perceptual. Specifically, the NIQE model is trained on the original CT dataset with all real data for measuring the quality of images accurately. We found that the non-MSE-based solutions are likely lead to higher scores, because the interpolations produced in these ways are sharper than the counterparts.

Table 4-2: No-reference metrices for evaluating the performance of our model. The best result is highlighted in bold and second best is underlined. (NIQE the lower the better, Ma et al. [52] the higher the better)

Metrices	Size	AE	ACAI	HRINet (PatchGAN only)	HRINet-MSE	HRINet-Perceptual
NIQE	256^2	2.440	2.425	<u>2.254</u>	2.258	**2.237**
	512^2	2.445	2.417	<u>2.359</u>	2.364	**2.351**
Ma et al. [52]	256^2	2.815	2.852	<u>2.878</u>	2.859	**2.891**
	512^2	2.550	2.562	2.891	<u>2.918</u>	**2.937**

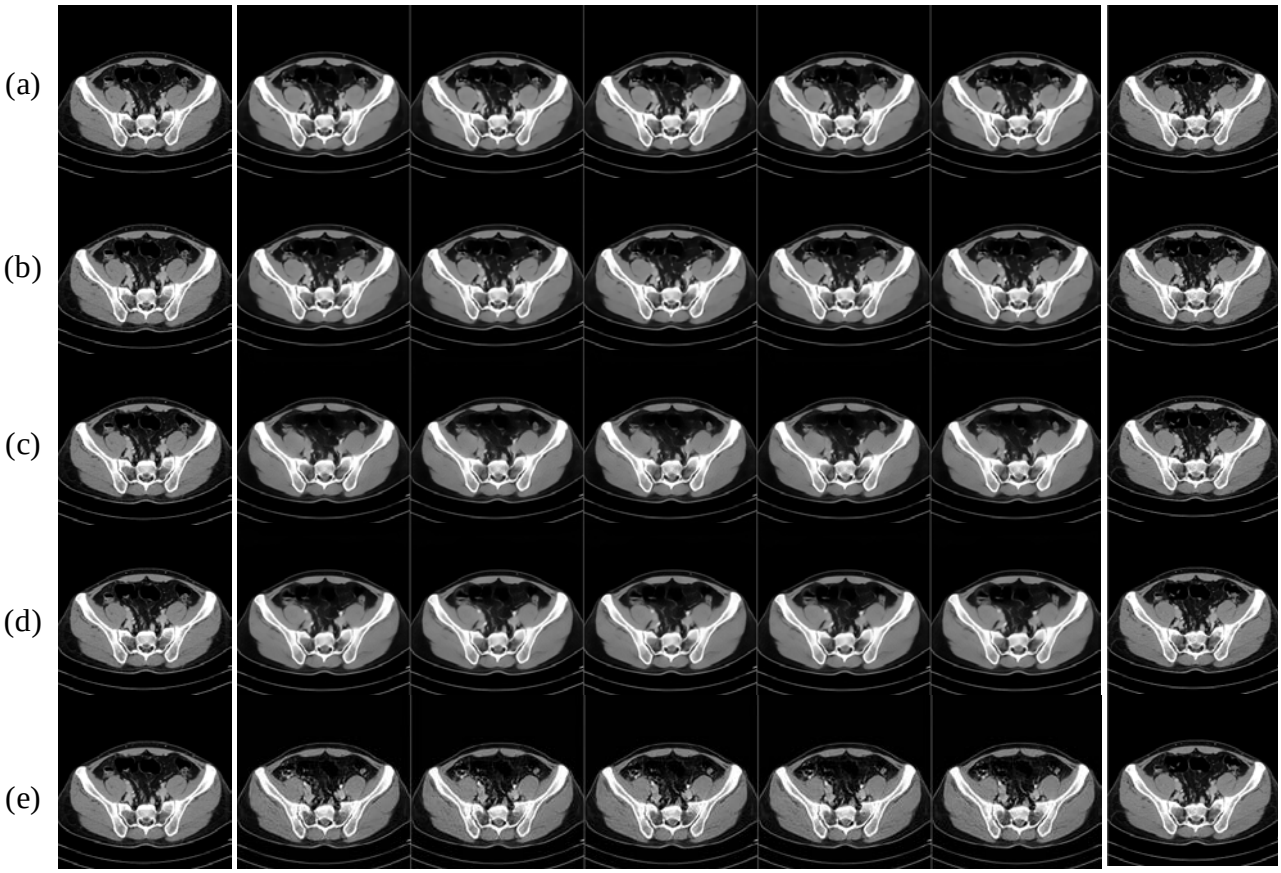


Figure 4-5: Examples of interpolating between adjacent slices. The left most column and right most column are input images, the middle ones are interpolations. Results are produced by (a) AE, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual

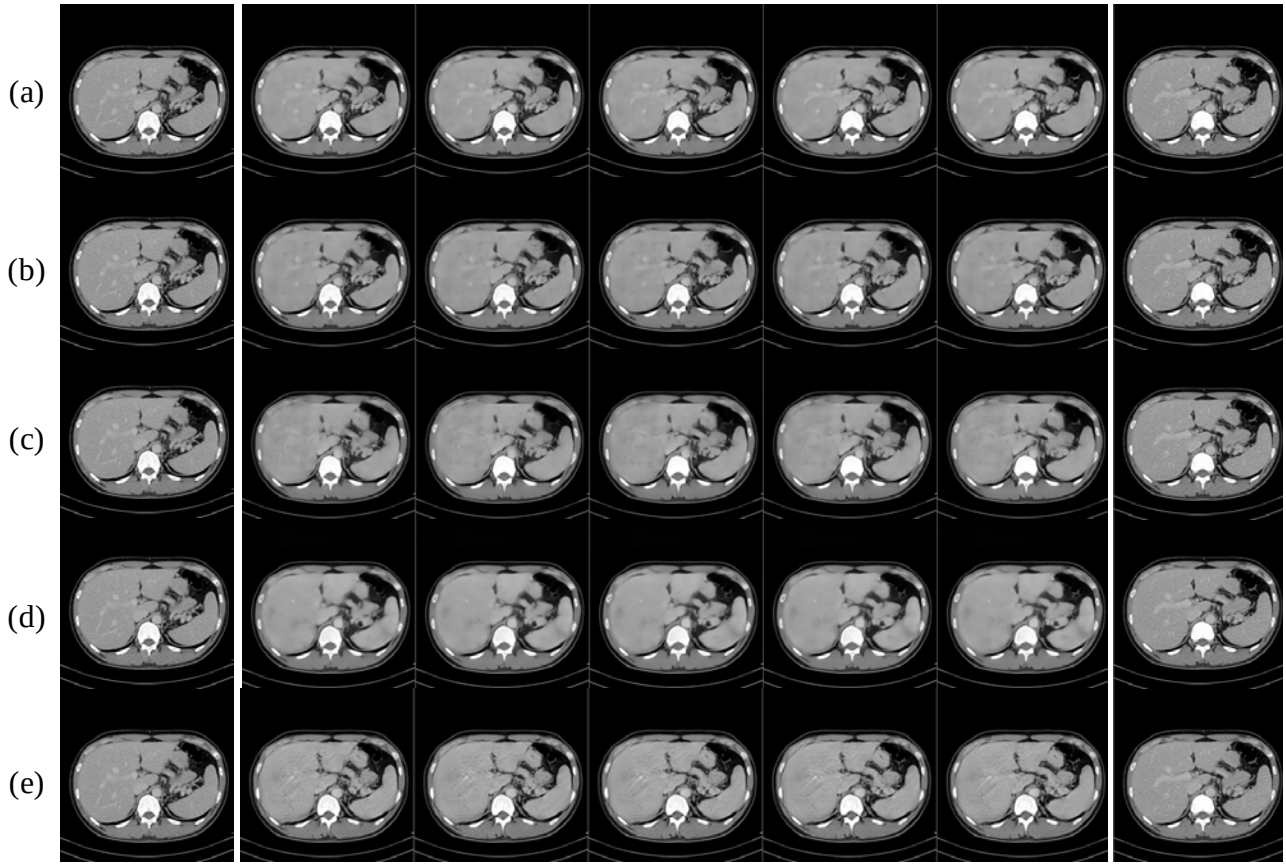


Figure 4-6: Examples of interpolating between adjacent slices. The left most column and right most column are input images, the middle ones are interpolations. Results are produced by (a) AE, (b) ACAI, (c) HRINet (PatchGAN only), (d) HRINet-MSE, (e) HRINet-Perceptual

4.3 Feature Reconstruction and Representation Learning

We have so far solely focused on measuring the quality of interpolations of different autoencoders. Now, we turn to the question of whether improved interpolation is associated with improved representation learning. In other words, we seek to test whether models perform better in feature extraction and reconstruction. Specifically, the autoencoder part of each model maps images into latent representation and then does the reverse process, during the process, features get entangled and disentangled. This can be done by having interpolation coefficient α equal to 0 or 1 consistently during the test.

Figure 4-7 shows the feature reconstruction results of each model. We observe that results from AE, ACAI and HRINet-MSE are smoother than HRINet-Perceptual, this is because of the overly smoothing effect of MSE loss items in their model, as mentioned in **2.2.2**. To further analyze the reconstruction abilities, we calculate the average PSNR, SSIM, L1 and L2 loss with ground truth. **Table 4-3** shows that solutions with MSE loss items like ACAI, HRINet-MSE are likely to have

lower distortion and higher structural similarity, and they achieve better non-perceptual numerical results.

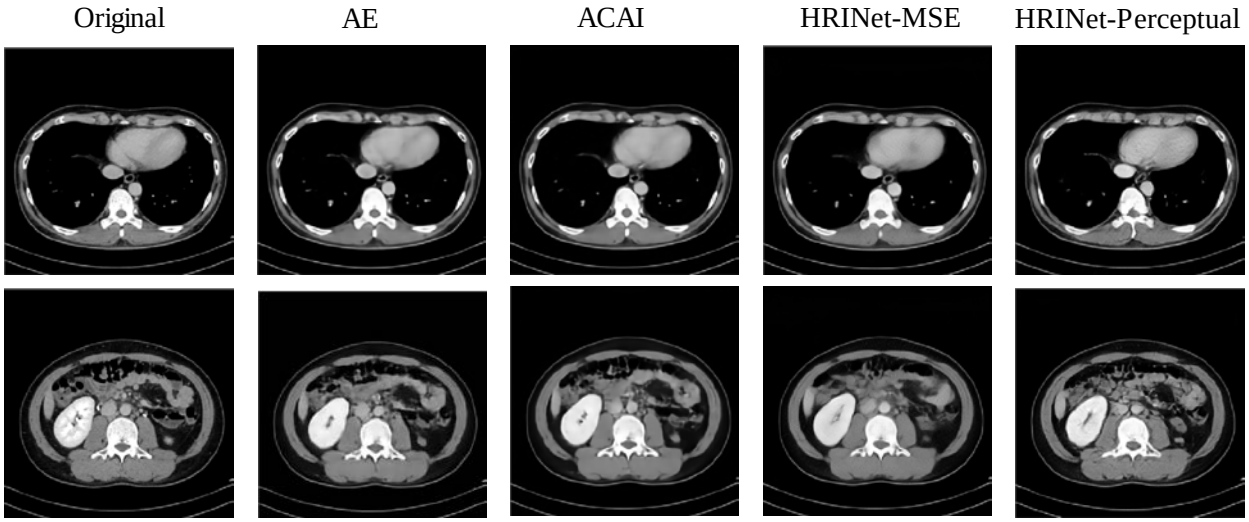


Figure 4-7: Feature reconstruction results from the autoencoder part of each model.

Table 4-3: Different metrices for evaluating the performance of feature reconstruction. Tests are done for 256^2 inputs. The best result is highlighted in bold and second best is underlined. (PSNR and SSIM are the higher the better, L1 and L2 loss are the lower the better.)

Metrices	**AE**	**ACAI**	**HRINet-MSE**	**HRINet-Perceptual**
PSNR	23.469	<u>23.860</u>	**25.748**	23.478
SSIM	0.942	<u>0.961</u>	**0.982**	0.953
L1	3.837	<u>3.513</u>	**3.234**	3.802
L2	14.723	<u>12.340</u>	**10.458**	14.455

Besides, **Figure 4-8** shows the visualization of some random samples sampled from reference distribution in latent space, which can reflect what kind of features that models extract from the inputs. We observe that the learnt distributions of AE and ACAI are very different from that of HRINet-MSE and HRINet-Perceptual. The former models mainly learnt structural features, and the latter learnt those structural features as well as invisible fine-grained features.

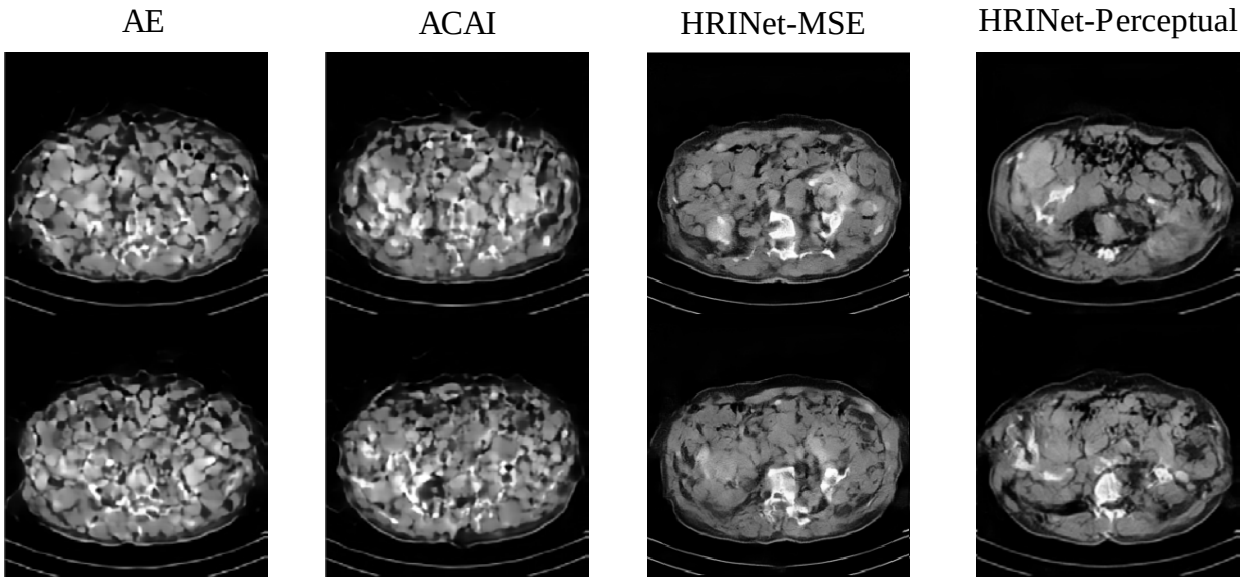


Figure 4-8: Random samples based on a reference distribution in latent space.

4.4 Ablation Study

4.4.1 Effects of Main Components

To figure out the influence of each main component to HRINet and how they improve the performance, we did ablation study for content loss, regularizes and training methods. In **Table 3-6**, we calculate two referenced metrices (PSNR/SSIM) and one perceptual metric (Ma et al. [52]) respectively for each parameter setting. The evaluation is done uniformly based on 256^2 input for the large gap interpolating task and some of the results are excerpted from tables above.

From **Table 4-4**, we observe that MSE content loss and alternative training are mainly served as reducing information loss and distortion as well as improving structure similarity. D_2 mainly improves those perceptual characteristics for high-frequency components. And D_1 has a significant impact for both referenced metrices and perceptual metrices, it enhances the quality of interpolations a lot. However, feature maps mixing mechanism has no obvious influence for PSNR/SSIM, but slightly improves the perceptual similarity. Compared with basic AE, HRINet increases the PSNR/SSIM score up to 12.52%/2.1% as well as Ma et al. [52] score up to 2.56%.

To show how the model learns to interpolate "semantically" step by step, we recorded the output statuses of the model among different training periods, with the increasing number of training epochs. We used non-relevant inputs to illustrate this idea in **Figure 4-9**, because the huge content difference between two input images makes manifold traversing becomes more difficult. In that case, if the model doesn't learn the manifold very well, it tends to produce results similar to data space mixing. Instead, the model would produce sharp interpolations when it performs well on embedding, though results may not have practical meaning.

Figure 4-9 illustrates the process of learning to interpolate "semantically". At the beginning, the autoencoder only trained with discriminator D_1, it gradually had the ability to generate interpolations, but results are much closer to data space mixing. Later, we added the PatchGAN regularizer D_2, it fluctuates the performance of the autoencoder a little bit at first. As training continues, the autoencoder becomes stable, it starts to produce sharp interpolations that are much different from previous results. We denote that the model has leant the data distribution and performed well on embedding.

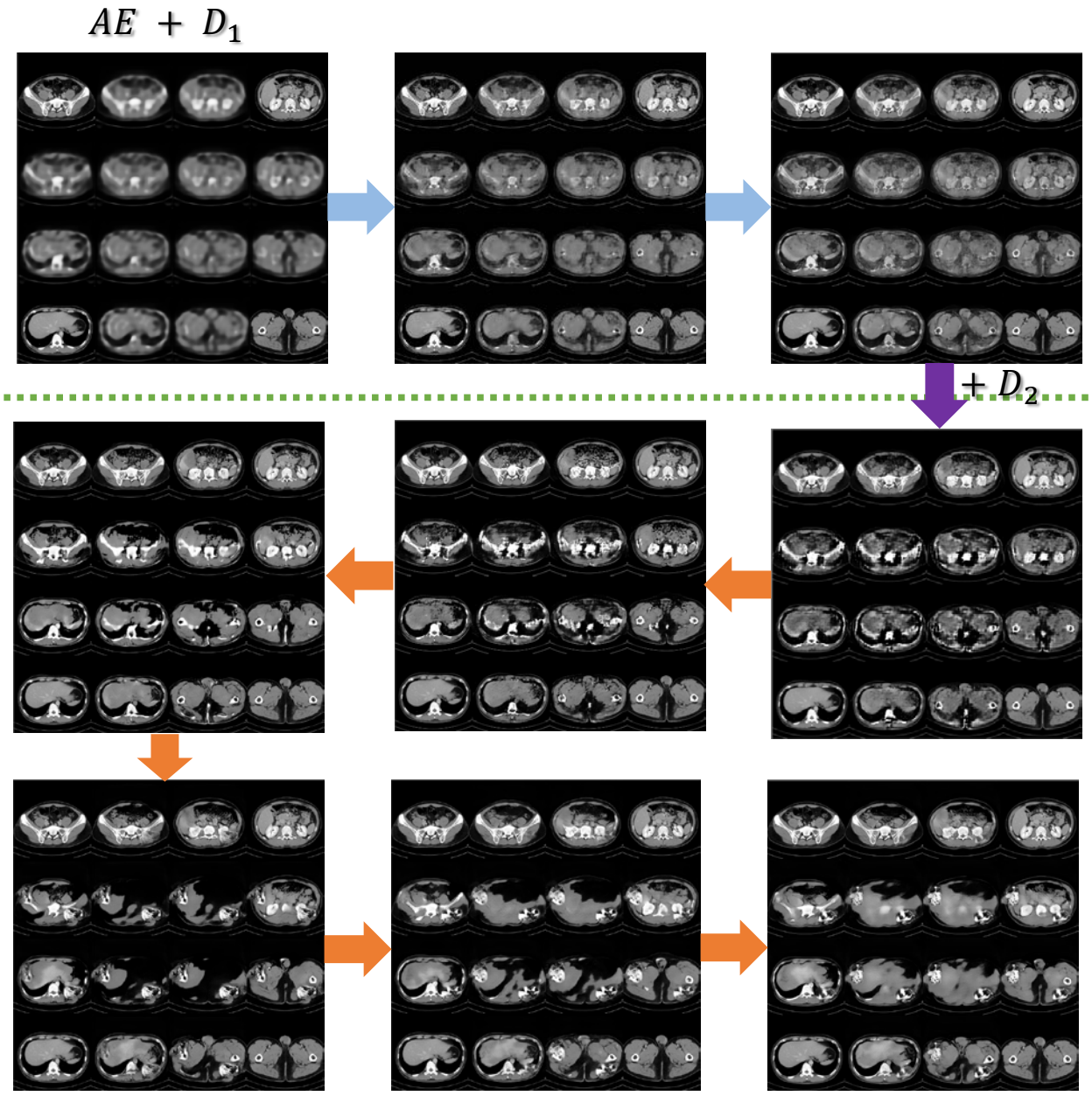


Figure 4-9: Interpolating on non-relevant inputs to show how the model learns to interpolate "semantically" step by step with the increasing number of training epochs. At the beginning, AE only trained with D_1. Later, we added the regularizer D_2.

Table 4-4: Ablation study for content loss, regularizes and training method. (PSNR/SSIM and Ma et al. [52] the higher the better)

MSE/Perceptual	Feature maps mixing	D_1	D_2	Alternative training	PSNR/SSIM	Ma et al. [52]
MSE	×	×	×	×	22.730/0.957	2.811
MSE	√	×	×	×	22.478/0.945	2.817
MSE	√	√	×	×	22.883/0.962	2.840
MSE	√	√	√	×	22.130/0.951	2.882
MSE	√	√	√	√	**25.576/0.977**	2.856
Perceptual	√	√	√	√	22.654/0.958	**2.883**

4.4.2 Receptive Field Size

In **3.3.2.3** we showed the visual results of modifying the size of receptive field for PatchGAN discriminator D_2, here we calculate the histogram intersection with ground truth to further illustrate why 70×70 is the best parameter setting.

Histogram intersection calculates the similarity of two discrete probability distributions (histograms), with possible value of the intersection lying between 0 and 1. If two discrete probability distributions are very different from each other, their histograms should have no overlap. Conversely, if two distributions' histogram intersection score is close to 1, that means they are in identical distribution. **Table 4-5** summaries the histogram interaction scores of various receptive field sizes, each score is calculated from the overlap between histograms in **Figure 4-10.**

Table 4-5: Histogram interaction scores with ground truth.

Receptive Field Size	Histogram Interaction Score
1×1 (PixelGAN)	0.62
16×16	0.73
70×70	**0.84**
142×142	0.71
256×256 (ImageGAN)	0.79

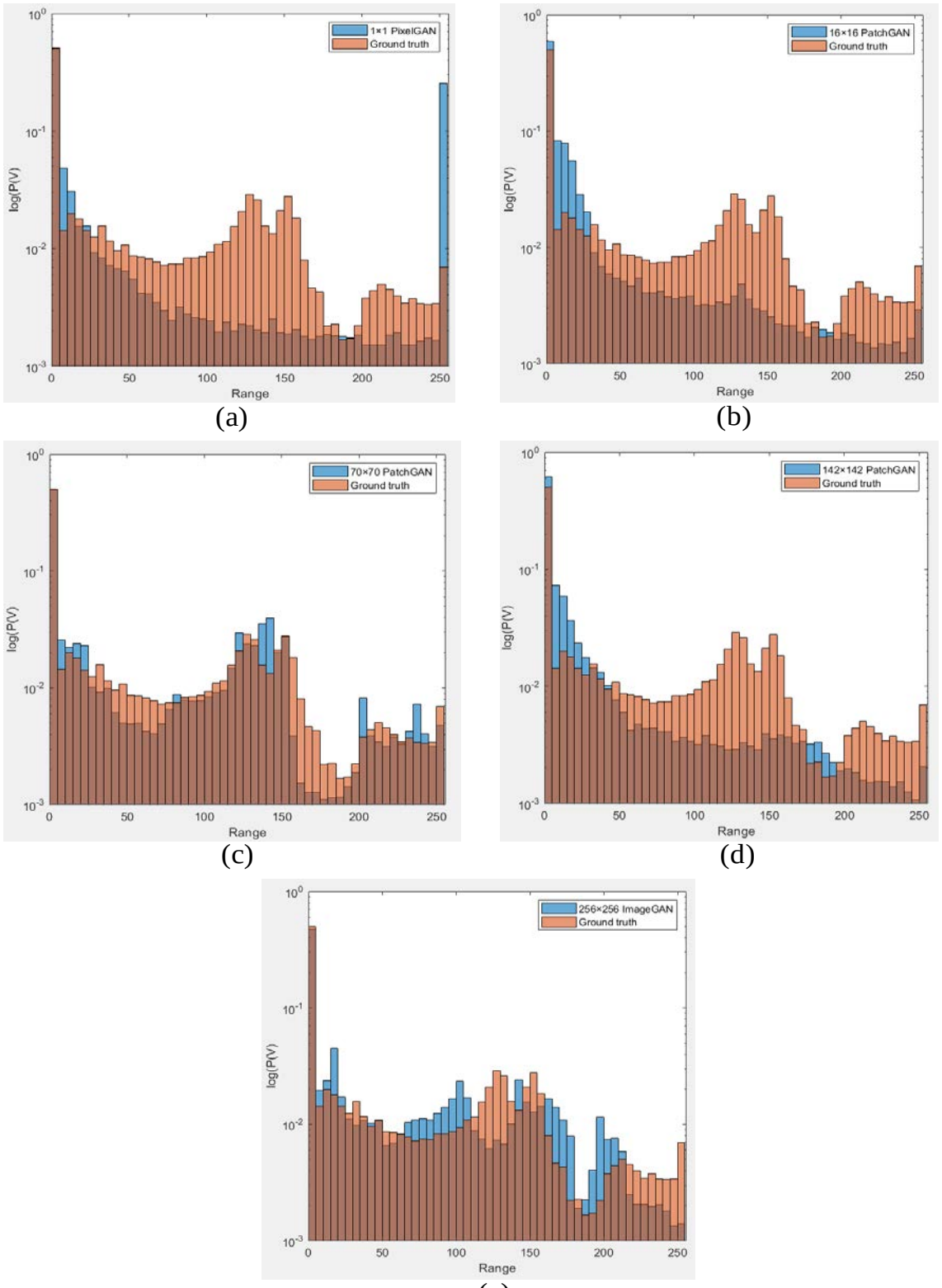


Figure 4-10: Histogram interaction with ground truth. Receptive field sizes of (a) 1×1 (PixelGAN) (b) 16×16 (c) 70×70 (d) 142×142 (e) 256×256 (ImageGAN).

Chapter 5. Application

5.1 2D Volume Projection

One of typical applications for high-resolution interpolation technology is 2D projection visualization. In this section, we introduce two kinds of projecting results created from original axial view CT slices and their interpolation: Orthogonal projection results and weighted projection results. We found that both HRINet-MSE and HRINet-Perceptual give similar results in 2D volume projection or 3D volume reconstruction, because the main difference between them are axial view details, those subtle discrepancies are almost indistinguishable after 2D projection or 3D reconstruction. Strictly speaking, all results in the current section and the following section are made from HRINet-Perceptual model.

Since HRINet creates high quality interpolations that fill gaps between slices, we are able to make smoother and sharper projection results easily. In fact, we create 5 intermediate interpolations between adjacent slices, the number of interpolations that required can be calculated by

$$n = \frac{gap\_distance * image\_size}{actual\_size}$$

Note that if all pixels in each slice are regarded as small cubic, the length of the cubic l_{pixel_cubic} is given by $\frac{atual\_size}{image\_size}$, thus the number of interpolations needed to filling the gaps n is $\frac{gap\_distance}{l_{pixel\_cubic}}$.

5.1.1 Orthogonal Projection

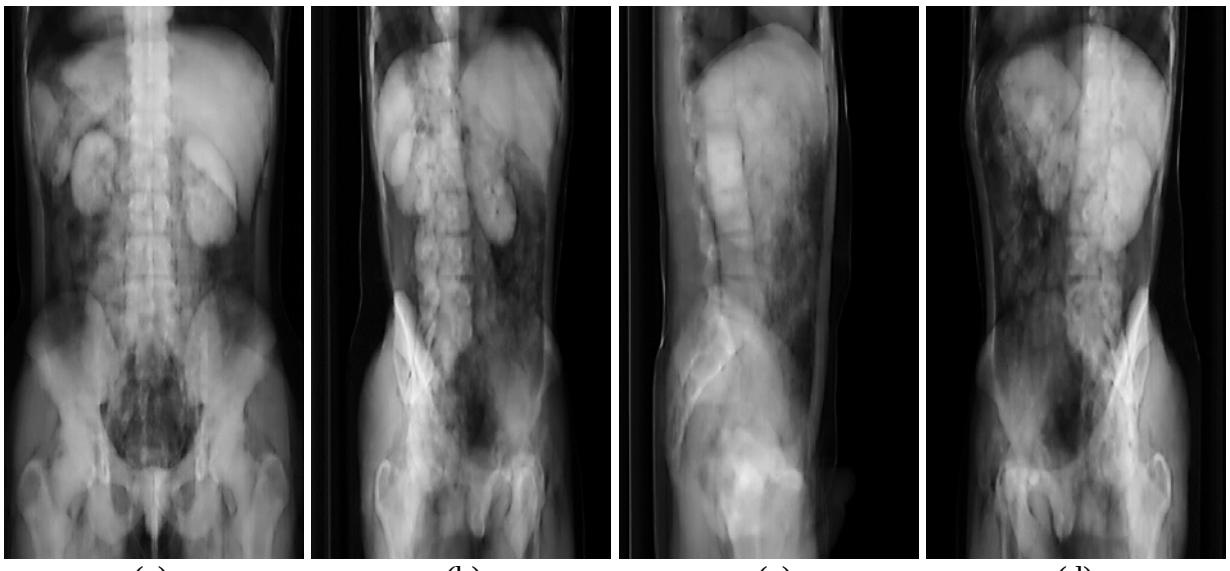
We created the orthogonal projection results of coronal view, sagittal view and even arbitrary view from the given axial view slices and their interpolations, results are shown in **Figure 5-1**. We used the Euler angle θ denotes the rotation angle according to horizontal axis, especially, it corresponds to sagittal view when $\theta = 0$, and corresponds to coronal view when $\theta = \pi/2$. This is achieved by projecting each axial view slice to a one-pixel width line for a given perspective, then placing them to the corresponding position in the projection results. **ALGORITHM 2** elaborates this process of creating orthogonal projection.

For orthogonal projection, we can assume that all pixels have the same weights and equals to one regardless of the relative position of the pixel. The number of interpolations that needed to filled the gaps between slices is calculated by the actual size of CT slices, so after the orthogonal projection reconstruction, we keep the original ratio of the size of human body organs and tissues that taken from axial view, and make sure that they are not distorted during reconstruction.

In **Figure 5-2**, we show the improvement on 2D orthogonal projection by comparing different approaches that deal with gaps before orthogonal projection. Since the HRINet with alternative supervision training can generate realistic interpolations, and keep the structural correctness and semantic smoothness at the same time, thus those interpolations are credited to filling gaps between slices. We can find out that the muscle tissues and bone outlines are properly reconstructed without too much blurry after 2D orthogonal projection, because interpolations compensate for the missing information for view changing. In **Figure 5-2 (c)**, we leave the unfilled regions to blank to illustrate how much extra information is needed when changing the view. The original data could only provide about five tenth of needed information, and its proportion would become smaller when larger gaps distance. However, both filling gaps by repeating pixel values (**Figure 5-2 (a)**) or doing bilinear interpolation (**Figure 5-2 (b)**) cannot provide extra information to eliminate the bias caused by view changing, as a result, they tend to produce blurry and unsmooth results.

ALGORITHM 2: Orthogonal projection

Input: CT slices $I_1, I_2, \dots, I_N \in \mathbb{R}^{n \times m \times 1}$, Euler angle θ.
for *k = 0,1,…,N-1:*
  for each pixel in i^{th} row and j^{th} column, **do**
    $M_\theta \leftarrow \theta$ // Transformation matrix
    $p_k(x, y) \leftarrow I_k(i, j) \otimes M_\theta$ // Relative coordinates after projection
  end
  $\hat{I}_k[t] \leftarrow \frac{1}{n}\sum p_k(x, y), t \in (0, n)$ // Sum by given axis
end



(a) (b) (c) (d)

Figure 5-1: Examples of 2D orthogonal projection reconstruction from any view. The Euler angle (a) $\theta = 0$ (coronal view); (b) $\theta = \pi/4$; (c) $\theta = \pi/2$ (sagittal view); (d) $\theta = 3\pi/4$.

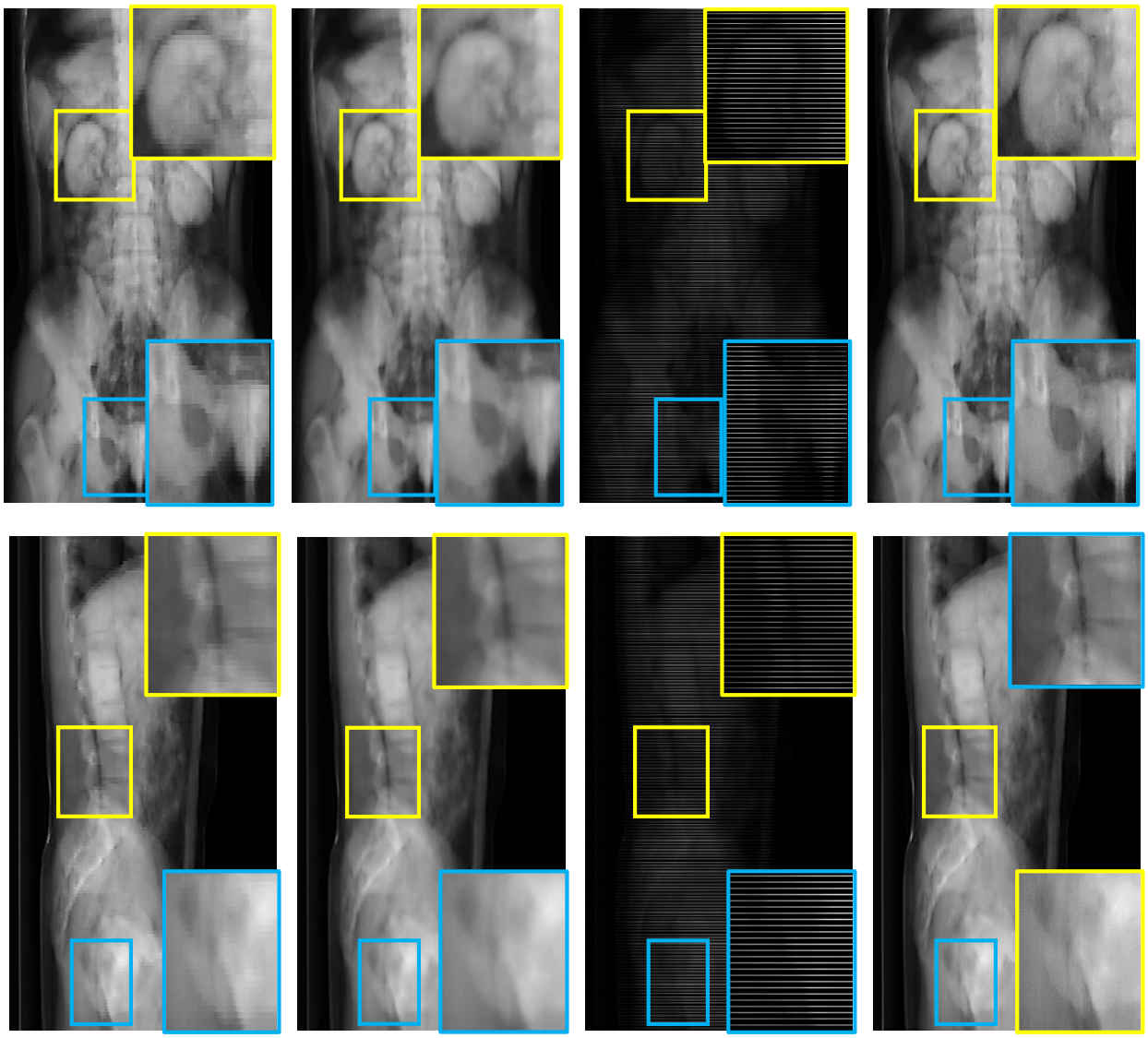


Figure 5-2: Comparison of different approaches dealing with gaps before orthogonal projection. (a) Repeat the previous line of pixels value (b) Bilinear interpolation (c) Not filling the gaps (f) HRINet interpolation.

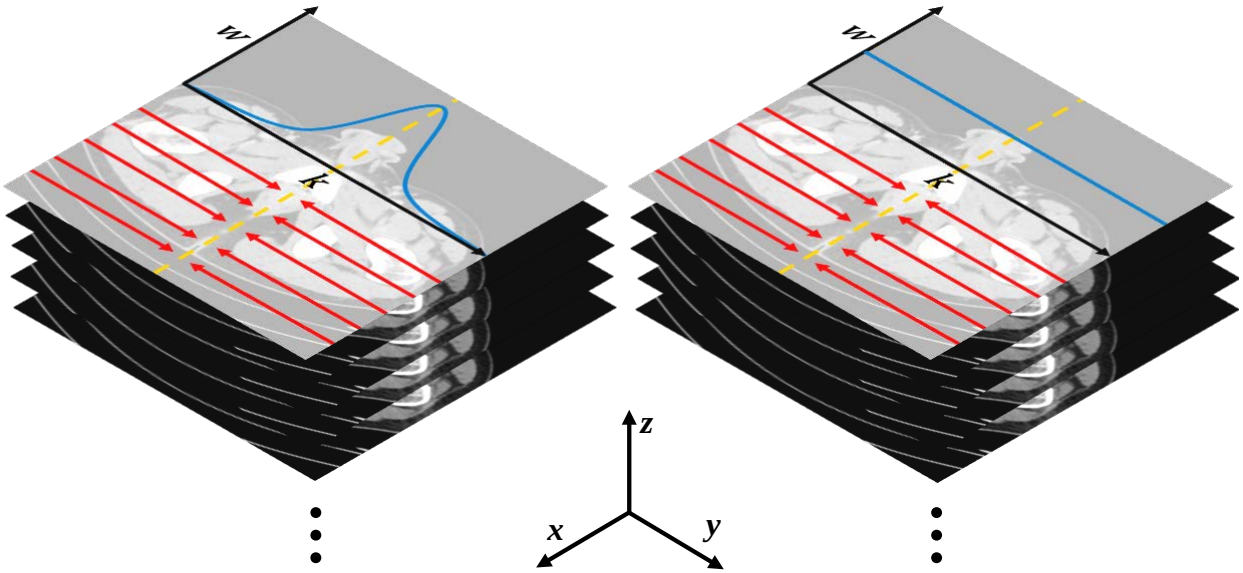
5.1.2 Weighted Projection

Expect for orthogonal projection, we also create perspective projection reconstruction by calculation weighted the sum of pixel values for a given view according to the relatively positions of pixels in axial view. For the contrast, we assigning them identical weight when creating orthogonal projection results. **Figure 5-3** illustrates how we create the sagittal view orthogonal projection and weighted projection from axial view images. Assuming k represents relative position of each pixel in axial view images and $k \in [0, n]$, where n refers to the width of images. The weight w that assigned to each pixel is mapped from k and n, denoted as $\mathcal{W}(.): w \leftarrow k$, where $\mathcal{W}(.)$ stands for the customized weights distribution.

We show some examples of weighted projection with different weight distributions in **Figure 5-4**. We can obtain the visualization of a specific body part by choosing a suitable weight distribution among pixels. For example, in **Figure 5-4 (d)**, by assigning the weights in Gaussian distribution with $\mu = n/2$ and $\sigma^2 = (n/11)^2$, we can easily get the sharp spine section without the interference of other body parts.

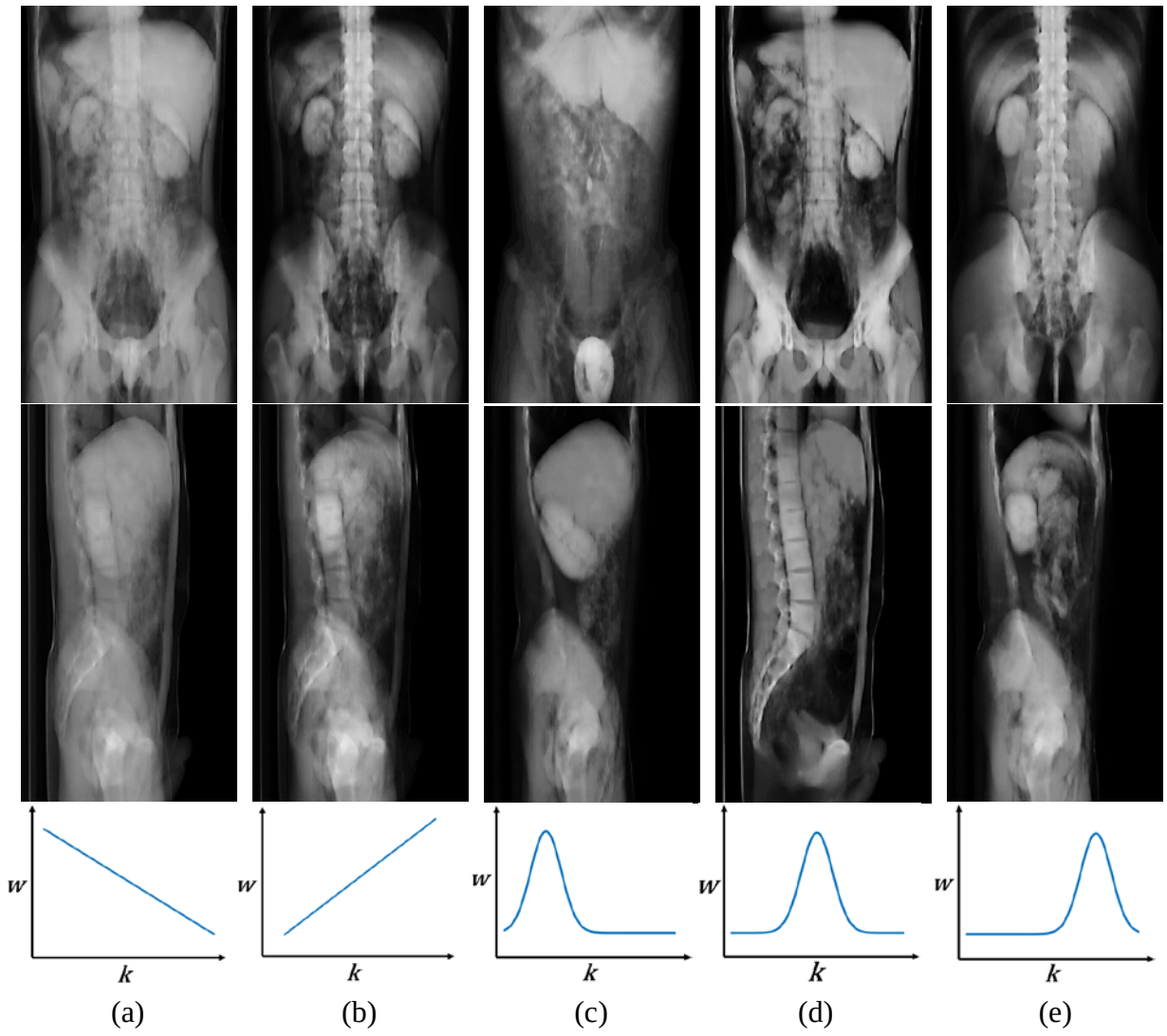
ALGORITHM 3: Weighted projection

Input: CT slices $I_1, I_2, \ldots, I_N \in \mathbb{R}^{n\times m\times 1}$, Euler angle θ, weights distribution $\mathcal{W}(.): w \leftarrow k, k \in (0, n)$
for *k = 0,1,…,N-1:*
  for each pixel in *i*th row and *j*th column, **do**
    $M_\theta \leftarrow \theta$ // Transformation matrix
    $p_k(x, y) \leftarrow I_k(i, j) \otimes M_\theta$ // Relative coordinates after projection
  end
  $\hat{I}_k[t] \leftarrow \frac{1}{n}\sum p_k(x, y) \odot \mathcal{W}(k)$ // Weighted sum by given axis
end



(a) (b)

Figure 5-3: Illustration of how we create the sagittal view orthogonal projection and weighted projection from axial view images. Red arrows stand for the direction we calculate weighted sum, which projecting whole images into one-pixel-width lines (yellow dash lines). (a) weighted projection, where the weight w obeys weights distribution $\mathcal{W}(k, n)$. (b) Orthogonal projection, where each pixel has identical weight.



(a) (b) (c) (d) (e)

Figure 5-4: Examples of weighted projection results. The last row shows the distribution of weights according to their relative positions in axial view, all of them are in the range of [0, 1]. (a) $w \propto \frac{1}{k}$; (b) $w \propto k$; (c) $w \in \mathcal{N}\left(\frac{n}{4}, \left(\frac{n}{11}\right)^2\right)$; (d) $w \in \mathcal{N}\left(\frac{n}{2}, \left(\frac{n}{11}\right)^2\right)$, where w is weight, k stands for relative position and $k \in [0, n], n$ is the width of the axial view image.

5.2 3D Volume Reconstruction

Another kind of significant application of high-resolution interpolations is 3D volume human body reconstruction. Since HRINet creates high quality interpolations that fill gaps between CT slices, we are able to make smoother 3D volume with those original slices and interpolations.

Figure 5-5 shows two kinds of human body models rendered on original axial view slices and their interpolations. We used a modeling software *"3D Slicers 4.10.2"* to create these models.

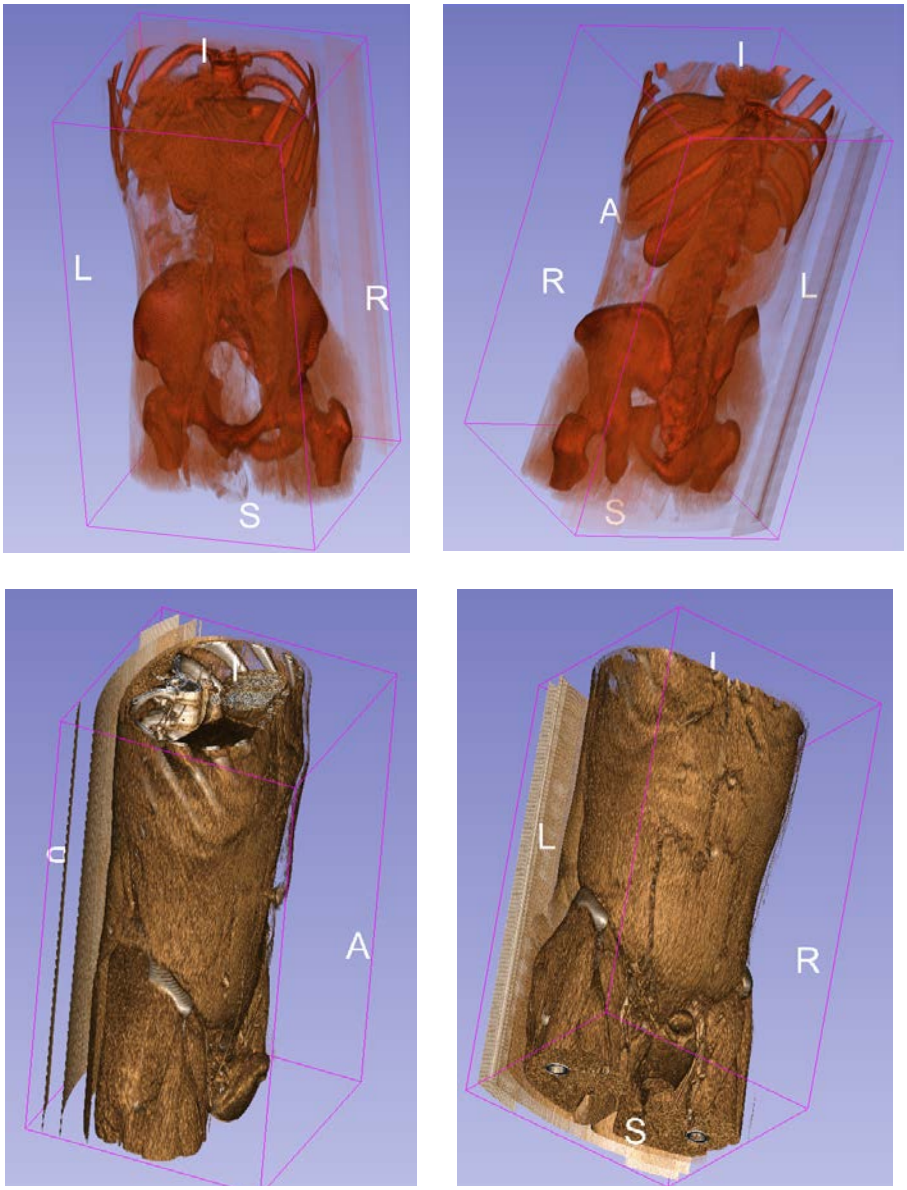


Figure 5-5: The inner organs and tissues model (first row) and overall structural model (second row) created from original axial view slices and their interpolations.

We illustrate the improvements of applying high-resolution interpolation in 3D volume reconstruction in **Figure 5-6**. Specifically, we create 5 intermediate interpolations between adjacent slices and the number is calculated by relative pixel size and the actual gaps distance. It is obvious that the silhouette of reconstructed organs and bones become much smoother than the one modeled on the original dataset. The new model gets rid of those zigzag edges because of smaller gaps between slices. In a word, the high quality and sharp interpolations guarantee the accuracy and fidelity of the reconstructed 3D model, this technology can be very helpful in the medical field.

Figure 5-6: The illustration of improvements on 3D volume reconstruction. (a) Modeling on original dataset. (b) Modeling with high quality interpolations.

Chapter 6. Conclusion

6.1 Conclusion

The majority of CT data are sampled at locations where the distance between consecutive slices is significantly larger than in-plane pixel size during scanning. In practice, the number of scans during in-vivo CT data scanning is limited for the purpose of avoiding the risks of radiation on patients as well as reducing the examining time. For that reason, there is spatial missing existed between adjacent CT slices. The missing parts, also called gaps, are sometimes neglectable when the distance of the gap is relatively large compared with the actual CT image size. Those large gaps between CT slices might lead to huge bias when doing further medical analysis such as 2D or 3D human body visualization.

(a) About problems and motivation

Image interpolation, known as creating a number of new slices between known CT slices to obtain an isotropic volume image by filling the gaps with created interpolations. Medical field puts high thresholds for interpolations to be used in practice, they need to be semantic and realistic enough, and resemble real data with only small error permitted. Existed models such as autoencoder and ACAI have shown ability to generate smooth interpolations for small images, but struggle with producing high-resolution CT interpolations. For one thing, they are trained in an unsupervised way which means that they lack regularization from real data to regularize their results, and that produce artifacts in their results. For another thing, high-resolutions results required more local features to be mixed in the latent space, by means of merely mixing the latent code can hardly produce interpolations with fine details and subtleties.

(b) About semantic correctness

In order to produce semantically correct interpolations, we apply alternatively supervising training which adding extra constraints to the generator in the dataspace, preventing it from generating interpolations with incorrect shape or details and overcome the “overlapping and blurry problems” of results. This is done by penalizing the generator when choosing an “inappropriate” path between datapoints on the manifold for interpolating. And then we show that the results are significant improved in the criteria of structural correctness of human tissues.

(c) About high-resolution

Since the size of original CT slices is 512^2, we maintain the resolution of our results to be 512^2 or 256^2 throughout the experiments. We apply PatchGAN discriminator as the texture and details regularizer in order to producing interpolations with sharp edges, which meets the requirement of

clear organ and tissues structure in the medical field. The discriminator penalizes the blurry regions in the results. Besides, the optional perceptual content loss overcome the overly smoothing effect caused by MSE content loss, which are beneficial to the high-resolution results as well.

(d) About high-resolution interpolation network (HRINet)

In the thesis, we propose a novel way of creating smooth CT interpolations with high quality, which is very useful to fill the gaps between adjacent CT slices. This is achieved by HRINet combining two phases of the training process, the unsupervised pre-training and alternatively supervising training. Besides, a lot of features we introduced in HRINet all make great contribution to the improvement.

Firstly, we modify the feature mixing and delivery mechanism in the autoencoder part. Instead of merely mixing latent codes, we mix low-level feature maps in order to fuse latent features in a more hierarchical way. Structurally, we also remove batch normalization layers within the autoencoder to improve the generalization ability of our model.

Secondly, we apply an extra PatchGAN discriminator which focuses on local response for the input image to eliminate local artifacts, by making generated results indistinguishable from real data.

To further regularize our model, we propose a novel training method “alternatively supervising training” to make the model produce interpolations more realistic corresponds to human cognition, it also minimizes the bias between results and ground truth to the great extend, and avoids the model to be overly trained in both latent space and data space.

In addition, we explore different possibilities for content loss, and discuss the pros and cos for MSE content loss and perceptual content loss respectively. We provide two viable versions of the model, they are HRINet-MSE and HRINet-Perceptual, depending on the type of content loss.

(e) About experiments and evaluation

Subsequently, we analyze the performance of our model among several tasks: Large gaps interpolations, adjacent slices interpolation and feature reconstruction. To provide comprehensive evaluation for our model, we evaluate it among various metrices, including reference based metrices (PSNR, SSIM, RMSE, L1, L2) and no-reference based metrices (Ma et al. [52], NIQE). We show our results are superior to other existing models such as autoencoder or ACAI quantitatively and qualitatively. We find that HRINet has made improvements in both accuracy and sharpness for CT data interpolation, and we also observe the trade-off between HRINet-MSE and HRINet-Perceptual in terms of the criterion of structural correctness and sharpness. For ablation study, we evaluate the effect of each main component such as regularizers, content loss, etc. To explore the best receptive field size for PatchGAN discriminator, we set up contrast experiments with different parameter setting, then compare intuitive results and histogram interaction scores respectively.

(f) About medical application

Finally, we show the applications for high-resolution interpolation approach in medical field, the 2D volume projection and 3D volume reconstruction. For 2D volume projection, it can be divided into orthogonal projection and weighted projection, depending on whether pixel values have identical weight or assigned weight according to customized weight distribution. And we also create projection

results for coronal view, sagittal view and even show the possibility for creating upon arbitrary view. For 3D volume reconstruction, we take advantage of existed software to render 3D models for internal structure and external silhouette. In the end, we illustrate the improvements of high-resolution interpolation in both 2D volume projection and 3D volume reconstruction, since it effectively fills those gaps with realistic interpolations. It eliminates zigzag edges in 2D and 3D results and provide sharper and clearer visualization with high accuracy and fidelity. All above prove the practical value of our model in medical field.

6.2 Future work

There is still room for further improvement for our model. One of the drawbacks is that some details like tissues and vessels in the CT images are not properly interpolated compared with skeletons and organs, this is because detailed information are mainly kept in high-level feature maps, merely mixing low-level feature maps can hardly do interpolations with those concrete details correctly. In the experiment, we find that mixing high-level feature maps would cause interpolating in data space rather than latent space, which is a more critical issue. Besides, the effectiveness of delivering features among the autoencoder is relatively low, because features between different levels of feature maps are highly coupled. Theoretically, if an appropriate decoupling strategy is applied, it would improve the feature delivering effectiveness a lot. Therefore, the way that features are mixed and delivered needs to be explored further.

References

[1] Lo, S. C., Lou, S. L., Lin, J. S., Freedman, M. T., Chien, M. V., & Mun, S. K. (1995). Artificial convolution neural network techniques and applications for lung nodule detection. *IEEE transactions on medical imaging*, *14*(4), 711-718.

[2] Rajkomar, A., Lingam, S., Taylor, A. G., Blum, M., & Mongan, J. (2017). High-throughput classification of radiographs using deep convolutional neural networks. *Journal of digital imaging*, *30*(1), 95-101.

[3] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[4] Ker, J., Wang, L., Rao, J., & Lim, T. (2017). Deep learning applications in medical image analysis. *Ieee Access*, *6*, 9375-9389.

[5] Wen, J., Thibeau-Sutre, E., Diaz-Melo, M., Samper-González, J., Routier, A., Bottani, S., ... & Alzheimer's Disease Neuroimaging Initiative. (2020). Convolutional Neural Networks for Classification of Alzheimer's Disease: Overview and Reproducible Evaluation. Medical Image Analysis, 101694.

[6] Akkus, Z., Galimzianova, A., Hoogi, A., Rubin, D. L., & Erickson, B. J. (2017). Deep learning for brain MRI segmentation: state of the art and future directions. *Journal of digital imaging*, *30*(4), 449-459.

[7] Moeskops, P., Viergever, M. A., Mendrik, A. M., De Vries, L. S., Benders, M. J., & Išgum, I. (2016). Automatic segmentation of MR brain images with a convolutional neural network. *IEEE transactions on medical imaging*, *35*(5), 1252-1261.

[8] Shen, D., Wu, G., & Suk, H. I. (2017). Deep learning in medical image analysis. *Annual review of biomedical engineering*, *19*, 221-248.

[9] Kleesiek, J., Urban, G., Hubert, A., Schwarz, D., Maier-Hein, K., Bendszus, M., & Biller, A. (2016). Deep MRI brain extraction: A 3D convolutional neural network for skull stripping. *NeuroImage*, *129*, 460-469.

[10] Cireşan, D. C., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2013, September). Mitosis detection in breast cancer histology images with deep neural networks. In *International conference on medical image computing and computer-assisted intervention* (pp. 411-418). Springer, Berlin, Heidelberg.

[11] Ciompi, F., de Hoop, B., van Riel, S. J., Chung, K., Scholten, E. T., Oudkerk, M., ... & van Ginneken, B. (2015). Automatic classification of pulmonary peri-fissural nodules in computed tomography using an ensemble of 2D views and a convolutional neural network out-of-the-box. *Medical image analysis*, *26*(1), 195-202.

[12] Yan, Z., Zhan, Y., Peng, Z., Liao, S., Shinagawa, Y., Metaxas, D. N., & Zhou, X. S. (2015, June). Bodypart recognition using multi-stage deep learning. In *International conference on information processing in medical imaging* (pp. 449-461). Springer, Cham.

[13] Roth, H. R., Lee, C. T., Shin, H. C., Seff, A., Kim, L., Yao, J., ... & Summers, R. M. (2015, April). Anatomy-specific classification of medical images using deep convolutional nets.

In *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)* (pp. 101-104). IEEE.

[14] Shin, H. C., Orton, M. R., Collins, D. J., Doran, S. J., & Leach, M. O. (2012). Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1930-1943.

[15] Ruan, W., & Lee, W. S. (2018, December). Recognising Named Entity of Medical Imaging Procedures in Clinical Notes. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (pp. 743-746). IEEE.

[16] Spencer, M., Eickholt, J., & Cheng, J. (2014). A deep learning network approach to ab initio protein secondary structure prediction. *IEEE/ACM transactions on computational biology and bioinformatics*, *12*(1), 103-112.

[17] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680).

[18] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[19] Yeh, R. A., Chen, C., Yian Lim, T., Schwing, A. G., Hasegawa-Johnson, M., & Do, M. N. (2017). Semantic image inpainting with deep generative models. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5485-5493).

[20] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., ... & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4681-4690).

[21] Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., ... & Change Loy, C. (2018). Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 0-0).

[22] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1125-1134).

[23] Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2223-2232).

[24] Boesen Lindbo Larsen, A., Kaae Sønderby, S., Larochelle, H., & Winther, O. (2015). Autoencoding beyond pixels using a learned similarity metric. *arXiv*, arXiv-1512..

[25] Jolicoeur-Martineau, A. (2018). The relativistic discriminator: a key element missing from standard GAN. *arXiv preprint arXiv:1807.00734*.

[26] Timofte, R., Agustsson, E., Van Gool, L., Yang, M. H., & Zhang, L. (2017). Ntire 2017 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 114-125).

[27] Sajjadi, M. S., Scholkopf, B., & Hirsch, M. (2017). Enhancenet: Single image super-resolution through automated texture synthesis. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 4491-4500).

[28] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).

[29] Kim, J., Kwon Lee, J., & Mu Lee, K. (2016). Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1646-1654).

[30] Pan, Z., Yu, W., Yi, X., Khan, A., Yuan, F., & Zheng, Y. (2019). Recent progress on generative adversarial networks (GANs): A survey. *IEEE Access*, *7*, 36322-36333.

[31] Wu, X., Xu, K., & Hall, P. (2017). A survey of image synthesis and editing with generative adversarial networks. *Tsinghua Science and Technology*, *22*(6), 660-674.

[32] Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

[33] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in neural information processing systems* (pp. 5767-5777).

[34] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[35] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems* (pp. 2172-2180).

[36] Qi, G. J. (2020). Loss-sensitive generative adversarial networks on lipschitz densities. *International Journal of Computer Vision*, *128*(5), 1118-1140.

[37] Donahue, J., Krähenbühl, P., & Darrell, T. (2016). Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

[38] Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

[39] Bourlard, H., & Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, *59*(4-5), 291-294.

[40] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[41] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, July). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (pp. 1096-1103).

[42] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

[43] Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, *2*(1-3), 37-52.

[44] Berthelot, D., Raffel, C., Roy, A., & Goodfellow, I. (2018). Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*.

[45] Krizhevsky, A., & Hinton, G. E. (2011, April). Using very deep autoencoders for content-based image retrieval. In *ESANN* (Vol. 1, p. 2).

[46] Li, J., Koh, J. C., & Lee, W. S. (2020). HRINet: Alternative Supervision Network for High-resolution CT image Interpolation. *arXiv preprint arXiv:2002.04455*.

[47] Dong, C., Loy, C. C., He, K., & Tang, X. (2015). Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, *38*(2), 295-307.

[48] Johnson, J., Alahi, A., & Fei-Fei, L. (2016, October). Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision* (pp. 694-711). Springer, Cham.

[49] Gatys, L., Ecker, A. S., & Bethge, M. (2015). Texture synthesis using convolutional neural networks. In *Advances in neural information processing systems* (pp. 262-270).

[50] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[51] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.

[52] Ma, C., Yang, C. Y., Yang, X., & Yang, M. H. (2017). Learning a no-reference quality metric for single-image super-resolution. *Computer Vision and Image Understanding*, *158*, 1-16.

[53] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

[54] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[55] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[56] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).

[57] Karlik, B., & Olgac, A. V. (2011). Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, *1*(4), 111-122.

[58] Sibi, P., Jones, S. A., & Siddarth, P. (2013). Analysis of different activation functions using back propagation neural networks. *Journal of theoretical and applied information technology*, *47*(3), 1264-1268.

[59] Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

[60] MacArthur, D. G., Balasubramanian, S., Frankish, A., Huang, N., Morris, J., Walter, K., ... & Albers, C. A. (2012). A systematic survey of loss-of-function variants in human protein-coding genes. *Science*, *335*(6070), 823-828.

[61] Zhang, Z., & Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems* (pp. 8778-8788).

[62] Taqi, A. M., Awad, A., Al-Azzo, F., & Milanova, M. (2018, April). The impact of multi-optimizers and data augmentation on TensorFlow convolutional neural network performance. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)* (pp. 140-145). IEEE.

[63] Bello, I., Zoph, B., Vasudevan, V., & Le, Q. V. (2017). Neural optimizer search with reinforcement learning. *arXiv preprint arXiv:1709.07417*.

[64] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177-186). Physica-Verlag HD.

[65] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, *12*(7).

[66] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[67] Wang, Z., Chen, J., & Hoi, S. C. (2020). Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

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

[69] Mittal, A., Soundararajan, R., & Bovik, A. C. (2012). Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, *20*(3), 209-212.

[70] Woodring, J. H., & Lee, C. (1992). The role and limitations of computed tomographic scanning in the evaluation of cervical trauma. *The Journal of trauma*, *33*(5), 698-708.

[71] Fahlbusch, R., & Samii, A. (2007). A review of cranial imaging techniques: potential and limitations. *Clinical neurosurgery*, *54*, 100.

[72] Caldwell, C. B., Mah, K., Skinner, M., & Danjoux, C. E. (2003). Can PET provide the 3D extent of tumor motion for individualized internal target volumes? A phantom study of the limitations of CT and the promise of PET. *International Journal of Radiation Oncology* Biology* Physics*, *55*(5), 1381-1393.

[73] Brenner, D. J., Elliston, C. D., Hall, E. J., & Berdon, W. E. (2001). Estimated risks of radiation-induced fatal cancer from pediatric CT. *American journal of roentgenology*, *176*(2), 289-296.

[74] Wicky, S., Blaser, P. F., Blanc, C. H., Leyvraz, P. F., Schnyder, P., & Meuli, R. A. (2000). Comparison between standard radiography and spiral CT with 3D reconstruction in the evaluation, classification and management of tibial plateau fractures. *European radiology*, *10*(8), 1227-1232.

Publication by the Author

[1] Li, Jiawei, Jae Chul Koh, and Won-Sook Lee. "HRINet: Alternative Supervision Network for High-resolution CT image Interpolation." Accepted by *International Conference of Image Processing (ICIP)*, paper ID 2123, May 2020.

[2] Li, Jiawei, Jae Chul Koh, and Won-Sook Lee. " Alternative Supervised and Unsupervised Learning for Image Interpolation and Projection Visualization" Under second-round review of *International Conference of Pattern Recognition (ICPR)*, paper ID 1133, April 2020.