

**SUPERRESOLUTION RECURRENT CONVOLUTIONAL NEURAL NETWORKS
FOR LEARNING WITH MULTI-RESOLUTION WHOLE SLIDE IMAGES**

by

Huu Dat Bui

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science

(Computer Science)

at the

UNIVERSITY OF WISCONSIN–WHITEWATER

November, 2018

Graduate Studies

The members of the Committee approve the thesis of

Huu Dat Bui presented on 30 November 2018

(Dr. Lopamudra Mukherjee, Chair)

(Dr. Hien Nguyen)

(Dr. Jiazhen Zhou)

SUPERRESOLUTION RECURRENT CONVOLUTIONAL NEURAL NETWORKS FOR LEARNING WITH MULTI-RESOLUTION WHOLE SLIDE IMAGES

By

Huu Dat Bui

Under the supervision of Associate Professor Lopamudra Mukherjee

At the University of Wisconsin-Whitewater

A recurrent convolutional neural network is supervised machine learning way to process images that has both properties of convolutional and recurrent networks. We propose Convolutional Neural Network(CNN) based approach and its advanced recurrent version(RCNN) to solve the problem of enhancing the resolution of images obtained from a low magnification scanner, also known as the image super-resolution (SR) problem. The given class of scanner produces microscopic images relatively fast and storage efficiently. However, those scanners generate comparatively low quality images than images from complex and sophisticated scanners and do not have the necessary resolution for diagnostic or clinical researches, therefore low resolutions scanners are not in demand.

The motivation of this study is to determine whether an image with low resolution could be enhanced by applying deep learning framework such that it would serve the same diagnostic purpose as a high resolution image from expensive scanners or microscopes. We presented novel network design and complex loss function. We validate these resolution improvements with computational analysis to show an enhanced image give the same quantitative results. In summary, our extensive experiments demonstrate that this method indeed produces images which are same quality to images from high resolution scanners. This approach opens up new application possibilities for using low-resolution scanners not only in terms of cost but also in access and speed of scanning for both research and possible clinical use.

TABLE OF CONTENTS

	Page
LIST OF TABLES	iv
LIST OF FIGURES	v
1 Introduction	1
1.1 Neural Networks	2
1.1.1 Convolutional Neural Networks	2
1.1.2 Recurrent Neural Networks	3
1.2 Deep Learning	4
1.3 Application Domain	5
2 Related Work	8
3 Methods	10
3.1 Single-resolution convolutional neural network	11
3.2 Multi-resolution convolutional neural network	13
3.2.1 CNN sub-network	13
3.2.2 Recurrent Convolution Network	14
3.3 Training and loss function	15
4 Experiments	18
4.1 Single-resolution convolutional neural network	19
4.2 Multi-resolution convolutional neural network	22
LIST OF REFERENCES	27
APPENDIX Python Code	34

LIST OF TABLES

Table	Page
4.1 Quantitative results from reconstructed Breast images.	19
4.2 Quantitative results from reconstructed Kidney images.	22
4.3 Quantitative results from reconstructed Breast images.	26
4.4 Quantitative results from reconstructed Kidney images.	26
4.5 Quantitative results from reconstructed Pancreatic images.	26

LIST OF FIGURES

Figure	Page
3.1 Architecture single-resolution convolutional neural network for image superresolution.	11
3.2 Architecture of our proposed recurrent convolutional neural network for image superresolution.	16
4.1 Results of reconstruction of breast TMA with CNN: Columns 1 and 3 show high and low resolution images and Column 2 shows the reconstructed image.	20
4.2 Results of reconstruction of Pancreatic TMA with CNN: Columns 1 and 3 show high and low resolution images and Column 2 shows the reconstructed image.	21
4.3 Results of reconstruction of breast TMA with RCNN: Columns 1 and 3 show high and low resolution images and Column 2 shows the reconstructed image.	23
4.4 Results of reconstruction of Kidney TMA with RCNN: Columns 1 and 3 show high and low resolution images and Column 2 shows the reconstructed image.	24
4.5 Results of reconstruction of Pancreatic TMA with RCNN: Columns 1 and 3 show high and low resolution images and Column 2 shows the reconstructed image.	25

Chapter 1

Introduction

Super-resolution (SR) problem is a well known class of problems, which task is constructing a high-resolution (HR) image from a low-resolution (LR) image without loss of information. In general, this problem has high degree of complexity and difficult to formulate universal rules for the low to high transformation. With development of computational techniques and computing machinery, convolutional neural networks(CNN) have become main driving tool to solve this problem, since they have the ability to find highly non-linear complex functions that constitute the mapping from the low-resolution to the high resolution image. Several recent results have shown state of the art results for the superresolution problem [14, 15, 25]. However, since these CNN frameworks involve a large number of parameters, empirical evidence has shown that such models need to be trained on large datasets to show reproducible accuracy and avoid overfitting. This is not a problem for most applications in computer vision, where labeled datasets in order of millions or larger (eg. ImageNet, TinyImages and etc.) are readily available. But for other application domains particularly microscopic or medical imaging, such large sample sizes are hard to acquire, given that each image has to be acquired individually, with significant human involvement. Therefore, lack of training data is a limiting requirement for biological applications, when used with Deep Network architectures that can easily overfit on small training samples.

1.1 Neural Networks

Neural Network, or An Artificial Neural Network is an information processing archetype that is inspired by the way human brain process information. The main element of this approach is the innovative system of units(neurons) interconnected into a large network with different structures. Neural network belongs to class supervised machine learning framework that can be trained by learning series of sample to perform some function, without any predefined rules. Neural network has been recently widely used and became driving technique to solve complex problems in different domains. Unfortunately, neural network was invited prior to computer's era, and suffered from at least one major obstacle and went through several epochs. The initial period of neural networks was begun with enthusiasm followed by a period of frustration and depression. Only relatively few scientists have made improvements at the time with lack of financial and professional support. These explorers have created a solid technology which outperformed the limitations identified by Minsky and Papert [53]. Minsky and Papert summed up and showed a general feeling of frustration against neural networks among researchers, and was thus accepted by most without further analysis. The evolution of cost efficient and high-performance computers boosted usage of neural networks for complex tasks that cannot be solved in traditional way.

In the context of super-resolution problem, the main idea of neural networks approach is to find a non-linear function that maps a low-resolution as input to high-resolution output, or in other words create a programmable rule that able to produce a relevant output depends on an input .

1.1.1 Convolutional Neural Networks

Convolutional Neural Network(ConvNet) is special class of feed forward artificial neural network with layers stacked in a queue and commonly used in computer vision to process images and videos including classification, recognition, transformation and even compression.

The main element in CNN is convolutional operation which responsible for information acquisition and combination. The first application of CNN was provided by LeCun in 1989 [42], in this article showed how handwritten zip code digits can be recognized by trained CNN.

Generally CNN architecture includes consecutive chain of convolution, pooling, activation, and classification (fully connected). Convolutional layer extracts fetures and produces feature maps by convolving a kernel with special size and shape across the input image. The main purpose of Pooling layer is to decode the output of preceding convolutional layers by applying the average or maximum of pixels in particular area and pass these values to the next layer. Rectified Linear Unit (ReLU) and its modifications such as Leaky ReLU are the most commonly used activation functions in the middle layers. ReLU linear transformation of input data by cutting any negative input values to zero(or close to zero values) while positive input values are passed as output [27]. To measure correctness of a prediction on an input data, the loss is used to compute the distance between the output of the last layer with expected result. Finally, parameters of the network are found by minimizing a loss function between prediction and ground truth labels with regularization constraints, and the network weights are updated at each iteration (e.g., using stochastic gradient descent or its modification Adam-optimization) using backpropagation until convergence.

1.1.2 Recurrent Neural Networks

A recurrent neural network (RNN)[6, 23, 82, 52] is a class of artificial neural network where connections between nodes form a directed graph along a sequence. Here, the input/outputs are not independent, rather such models recompute the similar function for each element in the sequence, with the intermediate/final output of subsequent elements in the network being dependent on the previous computations occurring earlier in the sequence.

The traditional RNN preserves an inherited state h at time t that is the output of a non-linear mapping from its input x_t and the previous state h_{t-1} :

$$h_t = \sigma(W \times x_t + U \times h_{t-1} + b) \quad [1.1]$$

where weight matrices W and U are shared over time.

RNN networks are widely applied for speech recognition [20] and machine translation [2] problems. However, relatively recent new applications have been found in image processing domain for this type of neurons system [32, 71]. Fundamentally, they take advantage of the nature of recurrent networks' architecture and produce more relevant outputs.

1.2 Deep Learning

Deep learning refers to neural networks with many layers (usually more than five) that extract a hierarchy of features from an input (e.g. raw image). It is a new and popular type of machine learning technique that extract a complex system of features from images due to their self-learning ability as opposed to the hand-crafted feature extraction in classical machine learning algorithms. They achieve impressive results and generalizability by training on large amount of data. The rapid increase in GPU processing power has enabled the development of state-of-the-art deep learning algorithms. This allowed training of deep learning algorithms with millions of images and provided robustness to variations in images.

There are several types of deep learning approaches that have been developed for different purposes, such as object detection and recognition, image classification and segmentation, speech recognition, and etc. Some of the known deep learning algorithms are stacked auto-encoders, deep Boltzmann machines, deep neural networks, and convolutional neural networks (CNNs). CNNs are the most commonly applied to image and video processing. For medical imaging, particularly convolutional networks have rapidly become a methodology of choice for analyzing a wide variety of medical images [3, 22, 29, 46, 78, 65, 60]. Such applications include classification of diseases, lesion classification and localization/segmentation, and registration [49]. Oktay has applied a multi-stream convolutional neural network to reconstruct high-resolution cardiac MRI from low-resolution input MRI volumes for similar problem domain [56]. In digital pathology, though deep/machine learning have been used for other purposes [31, 40, 51, 72], such as computer aided diagnosis, detection of tumors in WSI, cancer

staging, prediction of cardiovascular risk factors from retinal fundus photographs and determining a comprehensive way to reconstruct super-resolution (enhanced) images from low resolution scanners have not been studied before.

1.3 Application Domain

Histopathology is still one of the main approach in disease diagnose, with a technology development it is common to use computer technology to create image-based information environment(Digital Pathology) and manage information generated from a digital slide in order to microscopic examination of tissue and study the manifestations of disease. The key element that enables digital pathology is virtual microscopy or *Whole slide imaging (WSI)*. Here we describe a problem scenario in digital pathology from microscopic imaging prospective where the SR problem can be adapted to address an important challenge in this context and how to address the issue of small sample size in this context. *Whole slide imaging (WSI)* or virtual microscopy is a technology that involves high-speed, digital acquisition of images representing entire stained tissue sections from glass slides in a format that allows them to be viewed by a pathologist on a computer monitor, where the image can be magnified and navigated spatially in much the same way as standard microscopy. This method is popular due to education and clinical demands[74, 75, 59]. Although modern whole slide scanners can now scan tissue slides with high resolution in a relatively short period of time, significant challenges, including high cost of equipment and data storage and the most relevant question of whether pathological diagnoses rendered using WSI are comparable with diagnoses made by microscopy, still remain unsolved [58]. Despite these issues, WSI can have numerous advantages for pathologists as well. Namely, the ability manipulate digital slides in shared fashion that leading to convenient access, regardless of location of the pathologist, which in turn results in faster ways to get alternative opinions, digital conferences, and decentralized primary diagnostic reviews. Digital storage also allows integration of digital slides into the patient's electronic profile as well as easy access to archived slides [19, 76]. Nevertheless these advantages, data storage and communication remain major drawbacks in high resolution digital pathology[58], in addition to the

prohibitive cost of high resolution scanners. One potential way to address these issues is to use LR images from low magnification slide scanners. Such devices are widely available, easy to use, relatively cheap and can also quickly produce images- with smaller storage requirements. However, LR images can increase the chance of misdiagnosis and false treatment if used as the primary source by a pathologist. For instance, cancer grading normally requires identifying tumor cells based on size and morphology assessments[1], which can be easily distorted in low magnification images. Addressing these concerns requires a way to improve the resolution of the images on-the-fly, without substantial increase in storage and computational requirements.

One of possible way to address the discussed issues is post-process low-resolution images to enhance the quality and resolution, in order to achieve relatively equally similar to those images acquired from high resolution scanners. This problem commonly solved by Superresolution (SR) methods, particularly Deep Learning(DL) approach has been frequently used in recent times. Specifically, CNN based SR methods have shown state of the art results for a number of applications [15], including for the application of interest here[55]. Despite this, one of the main challenges that limit the applicability of DL methods for such biomedical applications, is the sample size of training data. As mentioned earlier, due to a large number of parameters in CNNs, they require a large number of training samples (often in the order of millions) to generalize well to unknown data. Applications such as ours do not satisfy this desiderata since it is generally limited to smaller dataset sizes, due to the complexities associated with the acquisition process.

As a solution to the above problem, researchers have proposed algorithms focusing CNN initialization tricks and modifications to CNN architecture designed for general purpose CNN architectures. Our approach to solving this problem draws upon the inherent nature of the superresolution problem. Suppose I_1 and I_h represents a particular low and high resolution image pair. If I_h is a significant resolution ratio higher than I_1 , learning their direct transformation function $f(I_1) = I_h$ can be challenging leading to a overparameterized system. But if we had access to some intermediate resolutions say I_2, \dots, I_{h-1} (with a smaller resolution

change between consecutive images), it makes intuitive sense that transformation which converts an image of a given resolution, into the closest high resolution image would roughly be the same across all the resolutions considered, if we assume that resolution changes vary smoothly across the sequence. Having more image pairs (I_{k-1}, I_k) for $k = 2 \dots h$. to train, it may be computationally easier to learn a smooth function \hat{f} , such that $\hat{f}(I_{k-1}) \approx I_k$ for all k . In this paper we formalize this notion and develop a recurrent neural network(RCNN) to learn from multi-resolution slide scanner images.

Chapter 2

Related Work

This section reviews dominated deep learning frameworks have been used to solve a super-resolution problem. We mainly focus on single-resolution and complex multi-resolution architectures.

Single-resolution architecture Stacked collaborative local auto-encoders are used [12] to construct the LR image layer by layer. [57] suggested a method for SR based on an extension of the predictive convolutional sparse coding framework. A multiple layer convolutional neural network (CNN), similar to our model, inspired by sparse-coding methods is proposed in [14, 15, 16]. Chen [10] proposed to use multi-stage trainable nonlinear reaction diffusion (TNRD) as an alternative to CNN where the weights and the nonlinearity is trainable. Wang et. al [73] trained a cascaded sparse coding network from end to end inspired by LISTA (Learning iterative shrinkage and thresholding algorithm)[24] to fully exploit the natural sparsity of images. Recently, [62] proposed a method for automated texture synthesis in reconstructed images by using a perceptual loss focusing on creating realistic textures. Several recent ideas have involved reducing the training complexity of the learning models using approaches such as Laplacian Pyramids [41], removing unnecessary components of CNN [48] and addressing the mutual dependencies of low and high resolution images using Deep Back-Projection Networks[26]. In addition, Generative adversarial networks (GAN) have also been used for the problem of single image super-resolution, these include [44, 77, 47, 35]. Other deep network based models for image super-resolution problem includes [37, 38, 70, 63]. We also briefly review SR approaches for sequence data such as videos. Most of the existing deep learning-based

video super-resolution methods using motion information inherent in video to generate a single HR output frame from multiple LR input frames. Kappeler [36] wraps video frames from the preceding and subsequent LR frames onto the current one using the optical flow method and pass them through a CNN that produces the output frame. Caballero [8] follows the same approach but replace the optical flow model with a trainable motion compensation network. Huang et al.[32] use a bidirectional recurrent architecture for video super-resolution with shallow networks but do not use any explicit motion compensation in their model. Other notable works include [61, 69].

Multi-resolution architecture Recurrent neural network (RNN) is well known among scientists and has different utilization [9, 18, 30, 43], but most famous and prosperous applications assign to the processing of sequential data such as handwriting recognition [43] and speech recognition [28].

One of successful application a multi-dimensional RNN (MDRNN) justify handwriting recognition was proposed in [21]. MDRNN has a conducted architecture in that it process an image as 2D sequential data. However, MDRNN has a single hidden layer, which cannot produce the feature hierarchy as CNN. Another type is a hierarchical RNN or the Neural Abstraction Pyramid (NAP) is proposed for image processing [4]. NAP is a biology-inspired architecture with both vertical and lateral recurrent connectivity, through which the image interpretation is gradually refined to resolve visual ambiguities. Important that though the general framework of NAP has recurrent and feedback connections, for object recognition only a feed-forward version was tested. The recurrent NAP was used for other tasks such as image reconstruction.

Relatively recent Recurrent Neural Networks have been used to address superresolution problem [32] and opposite compression issue [71], their application shows significant improvement compare to traditional architectures. In [32] Huang and Wang use bidirectional multi-frame network to improve video resolution by modelling temporal dependency in video sequences. However, we are not aware of recurrent network application in the context of up-scaling task.

Chapter 3

Methods

This chapter describes the our single-resolution [54] and its advanced modification using multi-resolution models for reconstructing high-resolution images from low-resolution inputs. First, we briefly define the problem via formal setting. Let H and L denote the high and low resolution image sets respectively. For training/learning, we assume that the corresponding high resolution image H_i for each low resolution image L_i is available. We extract patches from the low resolution image L_i and represent each patch as a high-dimensional vector (where the vector for j th patch is referred to as l_{ji}). The goal of the training process is to learn a non-linear function \mathbf{f} , when applied to l_{ji} , transforms it into a high-resolution reconstructed patch r_{ji} . That is $r_{ji} = \mathbf{f}(l_{ji})$. Then, we aggregate all such patches to form the reconstructed high resolution image R . The objective driving the training process often minimizes some metric of difference between R and H . Most CNN [14, 15, 16] based models employ a number of convolutional layers to learn the complex mapping between the imaging domains. But there are some salient properties of our data which makes it hard to apply these existing approaches directly to our problem. First, these models are trained by upscaling the LR input image to the size of HR image, before it is passed through the CNN layers, which often leads to higher computational times to train the model. Secondly, these methods are not designed to handle cases where the LR image may be from a different modality, such as in our case, where the complexity of the transformation is much greater. Also, most CNN based methods use mean square error(MSE) as the metric to evaluate the similarity of H and R , that is $\mathbf{MSE}(H, R) = \sum_i \|H_i - R_i\|_2^2$, where R_i is the reconstruction of the i th image. Such a loss function is easy to minimize, but it correlates poorly with human perception of image quality and as a result, the resultant images

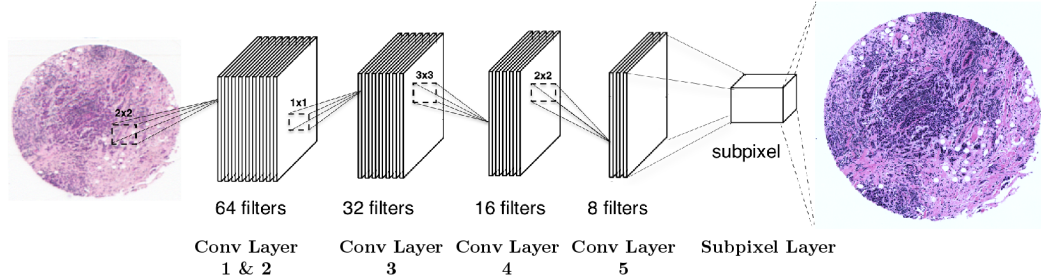


Figure 3.1 Architecture single-resolution convolutional neural network for image superresolution.

are sometimes blurry and/or lacking the high-frequency components of the original images. We address this issue by adding image saliency based terms in the objective.

3.1 Single-resolution convolutional neural network

Here, we describe the architecture single-resolution convolutional neural network 3.1 that underlies in our proposed Convolutional Recurrent Neural Network. **Feature extraction layer:** The first step in the convolution process is to extract features from the low resolution input images. Note that for most feature extraction methods such as Haar, DCT etc, the key problem can be posed as the task of learning a function \hat{f} , which takes as input the low resolution images and outputs the learned features $\hat{f}(L_i)$. Therefore the feature extraction process can be learned as a layer of the convolutional neural network, which constitutes the first layer of our network. This can be expressed as

$$Y_1 = \sigma(\theta_1 \times L + b_1) \quad [3.1]$$

where L is the entire corpus of low resolution images and θ_1 and b_1 represent the weights and biases of the first layer. The weights are composed of $n_1 = 64$ convolutions on each image patch, with each convolution filter being of size 2×2 . Therefore this layer has 64 filters, each of size 2×2 . The bias vector is of size $b_i \in R^{n_1}$. We keep filter sizes small at this level, so as it extract more fine grained features from each patch. The $\sigma(x)$ function implements a ReLU function, which can be written as $\sigma(x) = \max(0, x)$.

Feature mapping layer The second layer is similar to the previous layer except the filter sizes are set to 1×1 . The number of filters are still set to 64. The purpose of this layer is to obtain a weighted sum pool of features across various feature-maps of the previous layer. The output of this layer is referred to as Y_2 .

Intermediate convolutional layers: The feature extraction layer is followed by three convolutional layers. In this setting, we assume that for the i th layer ($i \in \{3, 4, 5\}$), the previous layer output is given Y_{i-1} , which is then served as input to the i th layer. The convolutional filter functions in these intermediate layers can be written as follows:

$$Y_i = \sigma(\theta_i \times Y_{i-1} + b_i) \quad i \in 3, 4, 5 \quad [3.2]$$

where θ_i and b_i represent the weights and biases of the i th layer. Each of the weights θ_i is composed of n_i filters of size $n_{i-1} \times f_i \times f_i$. We set $n_i = 2^{8-i}$. This makes $n_3 = 32$ and the number of filter decreases by a factor of 2, with each subsequent layer. We observe this has computational advantages, without noticeable decay in reconstruction performance. The filter sizes f_i are set to $\{3, 2, 1\}$ for each of the three layers respectively. This is akin to first applying the non-linear mapping to 3×3 patch of the feature map and then progressively reducing the size to 1. This structure is inspired by hierarchal CNN models, as described in [33].

Subpixel layer: The purpose of the final (6th) layer is to increase the resolution of the LR image to convert it to a HR image from the learnt LR feature maps. For this, we use a subpixel layer similar to the one proposed in [66]. The advantage of using such sub-pixel layer is that other previous layers operate on the reduced LR image, which reduce the computational and memory complexity substantially.

The upscaling of the LR image to the size of the HR image is implemented as a convolution with a filter θ_{sub} whose stride is $\frac{1}{r}$ (r is the resolution ratio between the HR and LR images). Let the size of the filter θ_{sub} be f_{sub} . A convolution with stride of $\frac{1}{r}$ in the LR space with a filter θ_{sub} (weight spacing $\frac{1}{r}$) would activate different parts of θ_{sub} for the convolution. The weights that fall between the pixels will not be activated. The patterns are activated at periodic intervals of $mod(x, r)$ and $mod(y, r)$ where x, y are the pixel position in HR space. Alternatively, this

can be implemented as a filter θ_6 , whose size is $n_5 \times r^2 \times f_6 \times f_6$, given that $f_6 = \frac{f_{sub}}{r}$ and $mod(f_{sub}, r) = 0$. This can be written as

$$Y_6 = \gamma(\theta_6 \times Y_5 + b_6) \quad [3.3]$$

where γ is periodic shuffling operator which rearranges r^2 channels of the output to the size of the HR image (See [67] for the detailed reasoning).

3.2 Multi-resolution convolutional neural network

In addition to low and high resolution images L and H respectively, we use two more intermediate resolutions of all images, we call these sets I^1 and I^2 respectively. These image sets can be ordered in terms of increasing resolution as L, I^1, I^2, H . For training/learning, we assume image to image correspondence among these four sets are known. Here, we propose a special designed recurrent convolutional network (RCNN) which uses a CNN sub-architecture to map the low-resolution images to next high resolution one. The three CNN sub-networks simulate our successfully applied network presented in [54], one for each input-target pair (L, I^1) , (I^1, I^2) and (I^2, H) respectively. These three CNN sub-networks are interconnected in a recurrent manner. Furthermore, we impose that the CNN pipelines share similar weights, to ensure that function learned for each pair of images is roughly the same. We describe the details of our model next.

3.2.1 CNN sub-network

Here we describe the basic structure of each CNN sub-network and its constituent layers.

Feature Extraction-Mapping Layer: The first step in the convolution process is to extract features from the low resolution input images. The feature extraction process can be learned as a layer of the convolutional neural network, which constitutes the first layer of our network. This can be expressed as

$$\hat{Y}_1 = \sigma(\theta_1 \times L + b_1) \quad [3.4]$$

where L is the entire corpus of low resolution images and θ_1 and b_1 represent the weights and biases of the first layer. The weights are composed of $n_1 = 64$ convolutions on each image patch, with each convolution filter being of size 2×2 . Therefore this layer has 64 filters, each of size 2×2 . The bias vector is of size $b_i \in R^{n_1}$. We keep filter sizes small at this level, so as it extract more fine grained features from each patch. The $\sigma(x)$ function implements a ReLU function, which can be written as $\sigma(x) = \max(0, x)$. This is followed by a sum pooling layer, to obtain a weighted sum pool of features across various feature-maps of the previous layer. The output of this layer is referred to as Y_1 .

Convolutional Layers: The feature extraction layer is followed by three convolutional layers. In this setting, we assume that for the i th layer ($i \in \{2, 3, 4\}$), the previous layer output is given Y_{i-1} , which is then served as input to the i th layer. The convolutional filter functions in these intermediate layers can be written as follows:

$$Y_i = \sigma(\theta_i \times Y_{i-1} + b_i) \quad i \in 2, 3, 4 \quad [3.5]$$

where θ_i and b_i represent the weights and biases of the i th layer. Each of the weights θ_i is composed of n_i filters of size $n_{i-1} \times f_i \times f_i$. We set $n_i = 2^{7-i}$. This makes $n_2 = 32$ and the number of filter decreases by a factor of 2, with each subsequent layer. We observe this has computational advantages, without noticeable decay in reconstruction performance. The filter sizes f_i are set to $\{3, 2, 1\}$ for each of the three layers respectively.

Subpixel Layer: The purpose of the subpixel layer is to increase the resolution of the lower resolution image to convert it to a higher resolution image from the learnt LR feature maps or vice versa. This layer is identical to the subpixel layer in our single-resolution network.

3.2.2 Recurrent Convolution Network

There are two types of connections used to connect the 3 CNN subnetworks, see Figure 3.2. The first type of recurrent convolutions, denoted by dotted lines, aim to model the dependency across images of the closeby resolution difference. These convolutions connect adjacent hidden layers of successive images(ordered by resolutions), where the current hidden layer is

conditioned on the hidden layer at the next higher resolution image. Since the image sizes differ at between these two layers, we use a subpixel layer (described next) to convert the input to appropriate sizes. The second type of recurrent convolutions, are denoted by dashed lines. This is used to model the dependency of a given hidden layer of the current image on the hidden layer at the immediate previous layer of the next higher resolution image. This endows the hidden layer with not only the output of the previous layer, but also information about how a higher resolution image has evolved. Once again, we add a subpixel layer to account for the size difference in the images being considered.

3.3 Training and loss function

The objective function, based on which the CNN is trained, is crucial in determining the quality of the high resolution reconstructions. Most SR systems minimize the pixel-wise mean squared error (MSE) between the HR and the reconstructed image, which while easy to optimize, often correlates poorly with human perception of image quality. This is because MSE estimator returns the average of a number of possible solutions, which does not perform well for high-dimensional data [62]. The paper by [39] shows that two very different reconstructions of the same image, can have the same MSE error and reconstructions based on MSE alone has been shown to be blurry and/or lack high frequency components of the original image [62, 68]. To address this issue, we train our CNN using linear combination function of Multi-scale structured similarity (**MSSIM**) in addition to mean square error between the reconstructed image (R) and the high resolution image (H). We briefly describe this objective next. In particular we choose the **MSSIM**, since it better calibrated to capture perceptual metrics of image quality. Also, its pixel-wise gradient has a simple analytical form and is inexpensive to compute and therefore can be easily incorporated in gradient descent based back-propagation. **MSSIM** is the multi-scale extension of structured similarity (**SIM**), which is defined based on the following parameters. Let x and y be two patches of equal size from the two images H and R being compared. Assume μ_x (μ_y) denote the mean, σ_x^2 (σ_y^2) denote the variance of the patch x (y)

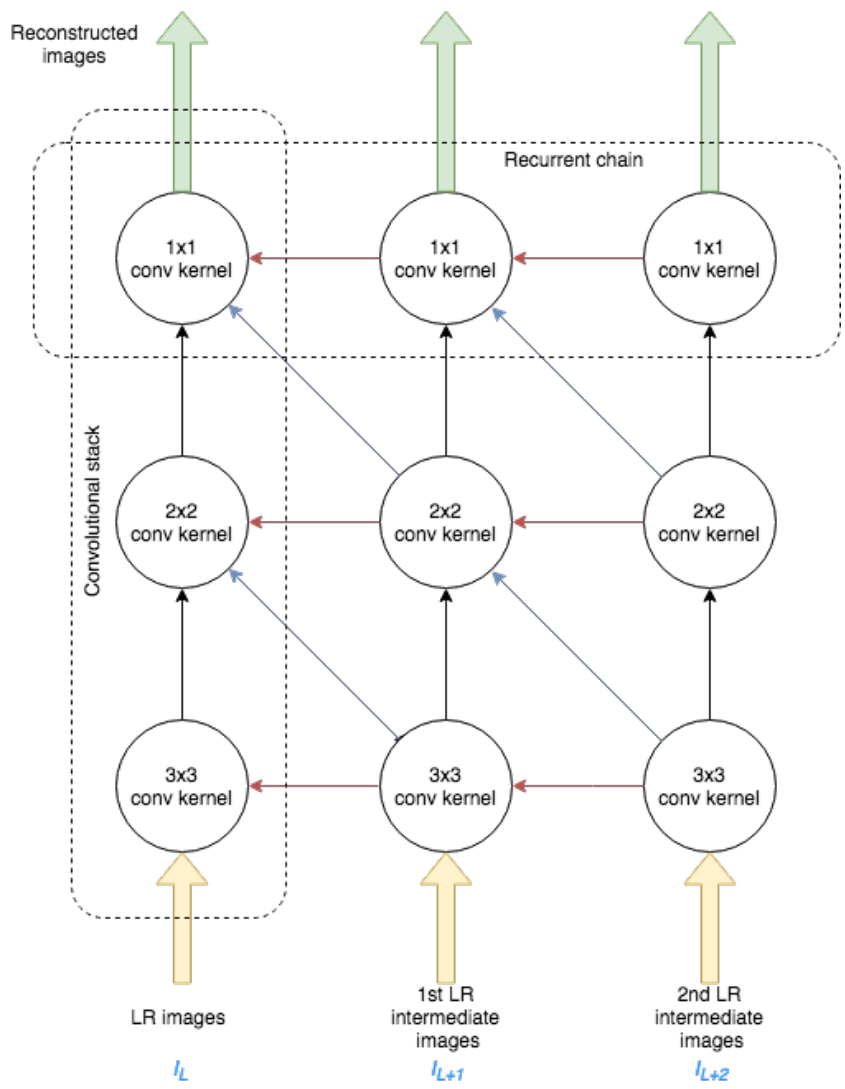


Figure 3.2 Architecture of our proposed recurrent convolutional neural network for image superresolution.

respectively, and σ_{xy} denote their covariance. Therefore, the **SIM** function can be defined as:

$$\mathbf{SIM}(x, y) = I(x, y)^\alpha C(x, y)^\beta S(x, y)^\gamma \quad [3.6]$$

were $I(x, y) = \frac{(2\mu_x\mu_y+c_1)}{(\mu_x^2+\mu_y^2+c_1)}$ is the luminance based comparison, $C(x, y) = \frac{(2\sigma_x\sigma_y+c_2)}{(\sigma_x^2+\sigma_y^2+c_2)}$ is a measure of contrast difference and $S(x, y) = \frac{\sigma_{xy}+c_3}{\sigma_x\sigma_y+c_3}$ is the measure of structural differences between the two images. c_i for $i = \{1, 2, 3\}$ are small values added for numerical stability and the α , β and γ are the relative exponent weights in the combination. The structured similarity between the images H and R is averaged over all corresponding patches x and y . This single-scale measure assumes a fixed image sampling density and viewing distance, and may only be appropriate for certain range of image scales. To make it more broadly applicable, a variant of **SIM**, called the multi-scale structured similarity (**MSSIM**) has been proposed. Here, the input x and y are iteratively downsampled by a factor of 2 with a low-pass filter, (with scale 1 denoting the original scale). The contrast and structural components of SIM are calculated at all scales (denoted by C_p and S_p for scale p). The luminance component is applied only at the highest scale(say P). The multi-scale structured similarity function can be written as

$$\mathbf{MSSIM}(x, y) = I_P(x, y)^\alpha \prod_{p=1}^P C_p(x, y)^{\beta_p} S_p(x, y)^{\gamma_p} \quad [3.7]$$

In our case, all the weights in the exponents are kept the same. We compute the **MSSIM** using 4 different scales, and use window sizes of 4×4 to calculate the metrics across both images.

Our loss function can be written as follows:

$$L(H, R) = \rho \mathbf{MSE}(H, R) + (1 - \rho) \mathbf{MSSIM}(H, R) \quad [3.8]$$

where ρ is between 0 and 1. Since both terms in the objective is differentiable, we can train the neural network using gradient descent, adopting standard back propagation methods.

Chapter 4

Experiments

This chapter presents conducted experiments to evaluate performance of the proposed approach on three tissue microarray (TMA) datasets, a Breast TMA dataset consisting of 182 images [11], and a Kidney TMA dataset with 381 images[5] and a Pancreatic TMA dataset with 180 images. Tissue microarray or TMA is one of recently developed direction in pathology. A microarray consists numbers of tiny tissue fragments from different studies assembled in one histologic slide, and therefore provides high efficient analyze ability of multiple specimens simultaneously. Tissue microarrays are paraffin blocks produced by extracting cylindrical tissue cores from different paraffin donor blocks and re-embedding these into a single microarray block at defined array coordinates.[34]

To measure the reconstruction quality of our deep learning framework, eight different metrics have been used to calculate distance between a high-resolution image as ground truth and a reconstructed image : 1) root mean square error (RMSE), 2) signal to noise ratio (SNR), 3) structured similarity (SSIM) and 4) Mutual Information (MI) 5) Multiscale Structured Similarity (MSSIM) 6) Information Fidelity Criteria(IFC) [64] 7) Noise quality measure(NQM)[13] and 8) Weighted peak signal-to-noise ratio (WSNR) [7]. RMSE should be as low as possible, whereas SNR, SSIM(1 being the maximum), MSSIM (1 being the maximum) and the remaining metrics, should be high for good reconstruction.

Moreover, to achieve provided results we trained our two type of networks separately for each TMA datasets, so 6 sets of network parameters have been acquired. This allows to achieve the high accuracy in the image reconstruction process.

4.1 Single-resolution convolutional neural network

First, we compare our single-resolution method with 6 other approaches: bicubic interpolation, which is a standard baseline, the patch based sparse coding approach (ScR) in [80, 81, 79], the deep learning approach (CSCN) [73, 50], the convolutional neural network based framework (FSRCNN) [16], a CNN model which uses a subpixel layer (ESCNN), a sparse coding based Dictionary learning method implemented using deep learning (SCDL) [55] and a GAN based implementation of SR (SRGAN)[45, 17]. All methods were trained with the same training batch of images and number of epochs.

Table 4.1 Quantitative results from reconstructed Breast images.

Breast TMA								
Method	RMSE	SNR	SSIM	MI	MSSIM	IFC	NQM	WSNR
Ours	18.39	22.99	0.7968	0.2784	0.94	2.86	19.40	23.73
Bicubic	34.94	17.39	0.48	0.20	.77	1.09	9.05	16.97
ScR	32.32	18.28	0.60	0.23	0.82	0.98	8.44	17.59
CSCN	18.82	22.87	0.66	0.16	.65	.32	3.6	18.48
FSRCNN	20.11	22.26	0.65	0.15	0.68	0.40	3.13	16.51
ESCNN	26.63	19.74	0.71	0.21	0.85	1.65	10.39	20.0
SCDL	16.69	23.92	0.69	0.23	0.85	1.27	9.31	20.71
SRGAN	29.76	18.74	0.73	.19	0.81	1.22	9.70	19.50

The table 4.1 shows that our single-resolution approach beats well known state of arts in most of metrics. Only the patch based sparse coding approach (ScR) has relatively better root mean square error (RMSE) and signal to noise ratio (SNR). The table 4.2 contains metrics for Kidney TMA dataset, that has significantly better result compare to other approaches. We did not perform comparison for Pancreatic TMA dataset, but visual results Figure 4.1 and Figure 4.2 show same level of quality with breast and kidney datasets.

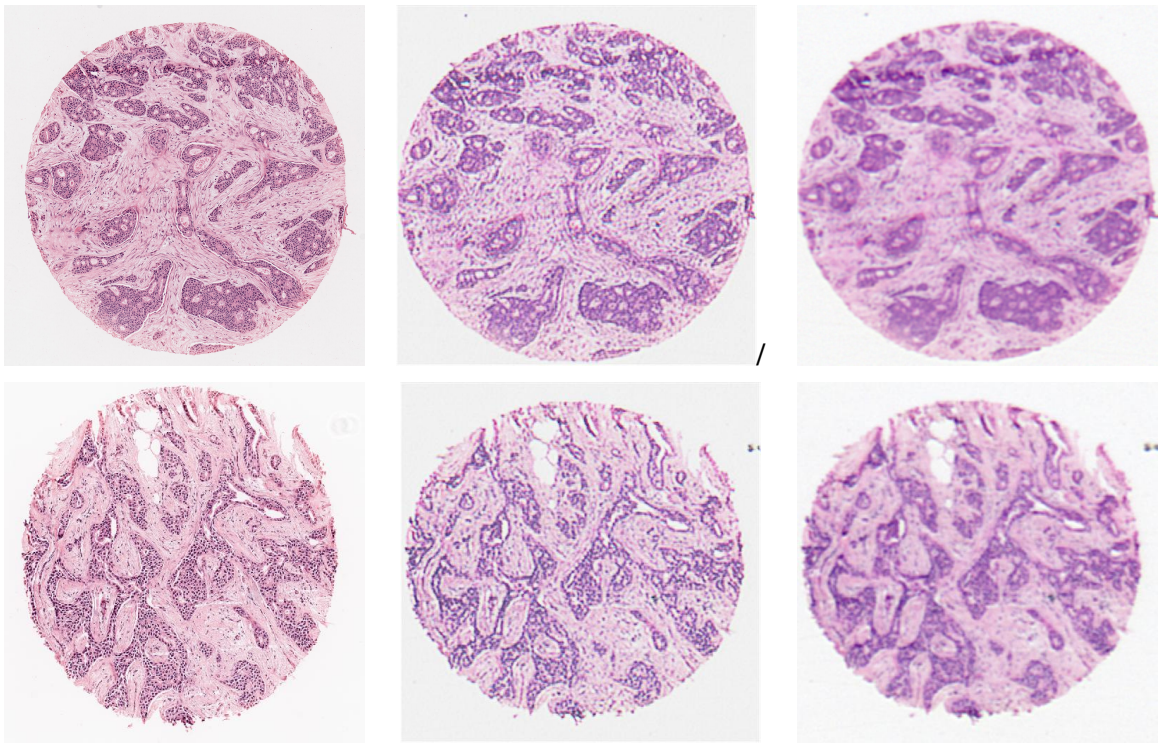


Figure 4.1 Results of reconstruction of breast TMA with CNN: Columns 1 and 3 show high and low resolution images and Column 2 shows the reconstructed image.

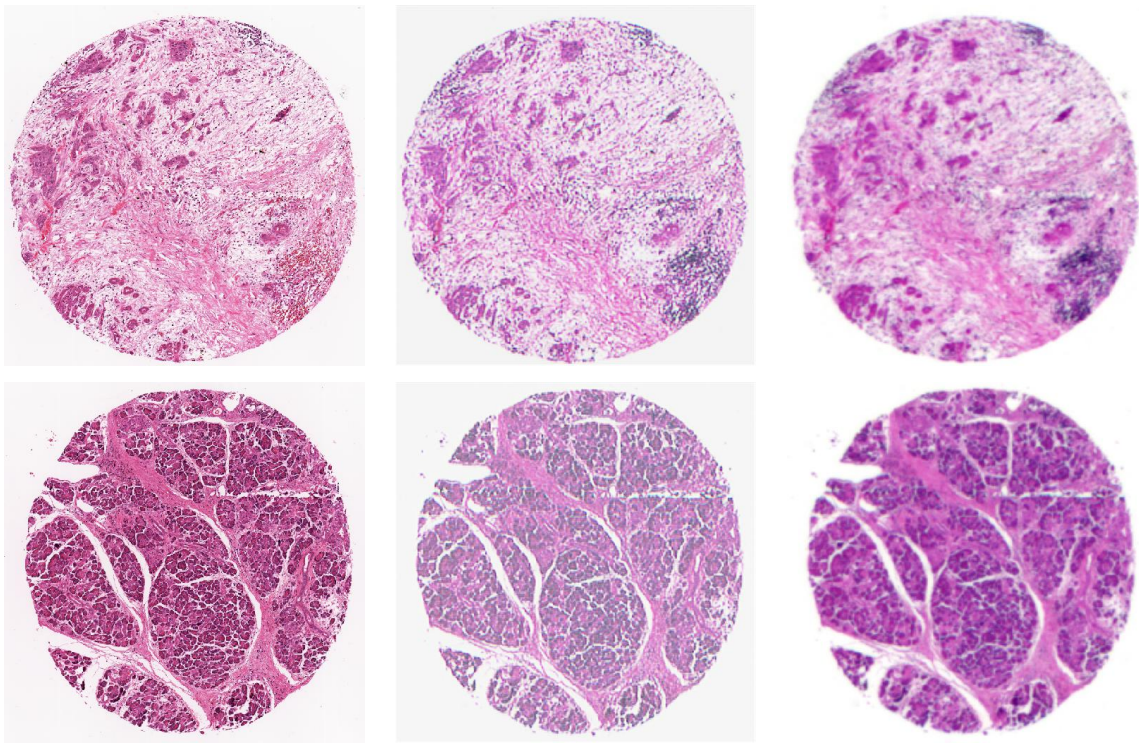


Figure 4.2 Results of reconstruction of Pancreatic TMA with CNN: Columns 1 and 3 show high and low resolution images and Column 2 shows the reconstructed image.

Table 4.2 Quantitative results from reconstructed Kidney images.

Kidney TMA								
Method	RMSE	SNR	SSIM	MI	MSSIM	IFC	NQM	WSNR
Ours	20.48	21.96	.75	0.21	0.93	1.99	13.41	30.05
Bicubic	36.66	16.39	0.49	0.16	0.71	.60	1.07	17.46
ScR	32.32	18.28	0.60	0.19	0.78	0.84	5.26	19.31
CSCN	31.55	18.20	.47	0.18	0.66	0.61	5.36	17.30
FSRCNN	35.86	18.43	.58	0.15	0.64	0.25	4.94	20.40
ESCNN	33.53	17.69	0.62	0.17	0.80	0.90	4.24	19.75
SCDL	22.34	21.18	.63	0.20	0.87	1.93	8.05	24.36
SRGAN	29.11	18.92	0.72	.15	0.81	0.87	7.028	23.31

4.2 Multi-resolution convolutional neural network

Here, we show that multi-resolution approach improves results and metrics compare to single-resolution one. Results are shown in Tables 4.3, 4.4 and 4.5.

Qualitative images of the reconstruction are shown in Figure 4.3 for Breast TMA, Figure 4.4 for Kidney and Figure 4.5 for Pancreatic TMA.

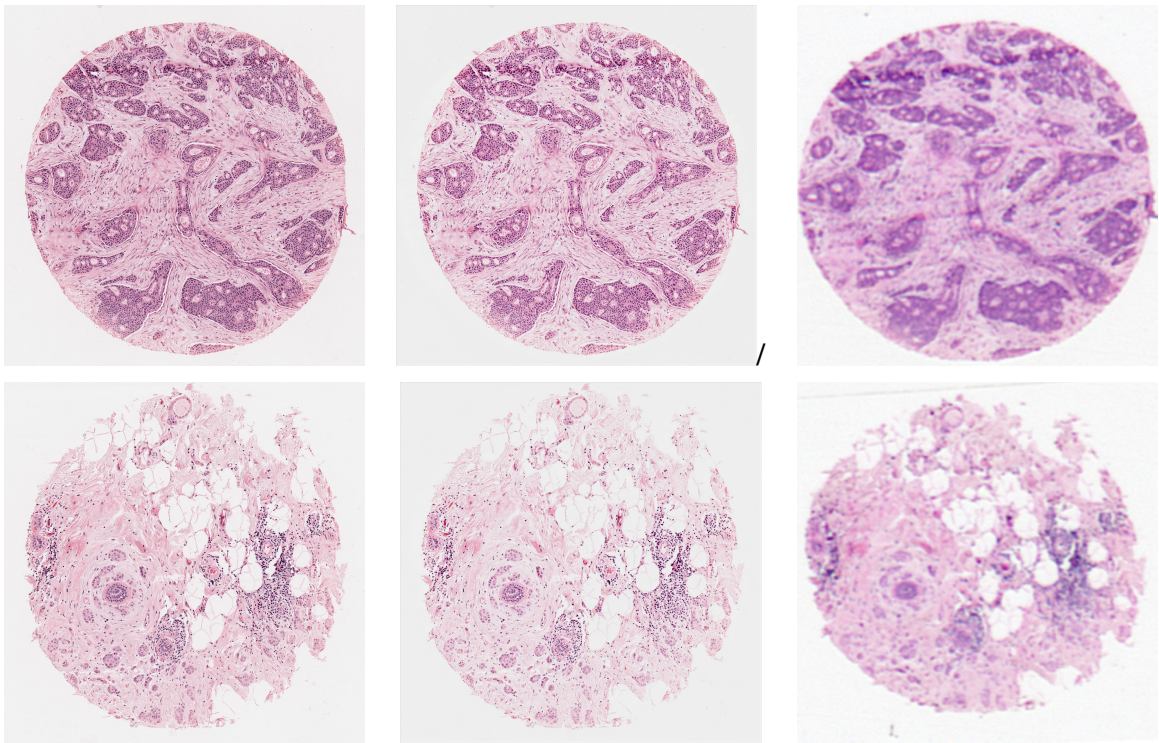


Figure 4.3 Results of reconstruction of breast TMA with RCNN: Columns 1 and 3 show high and low resolution images and Column 2 shows the reconstructed image.

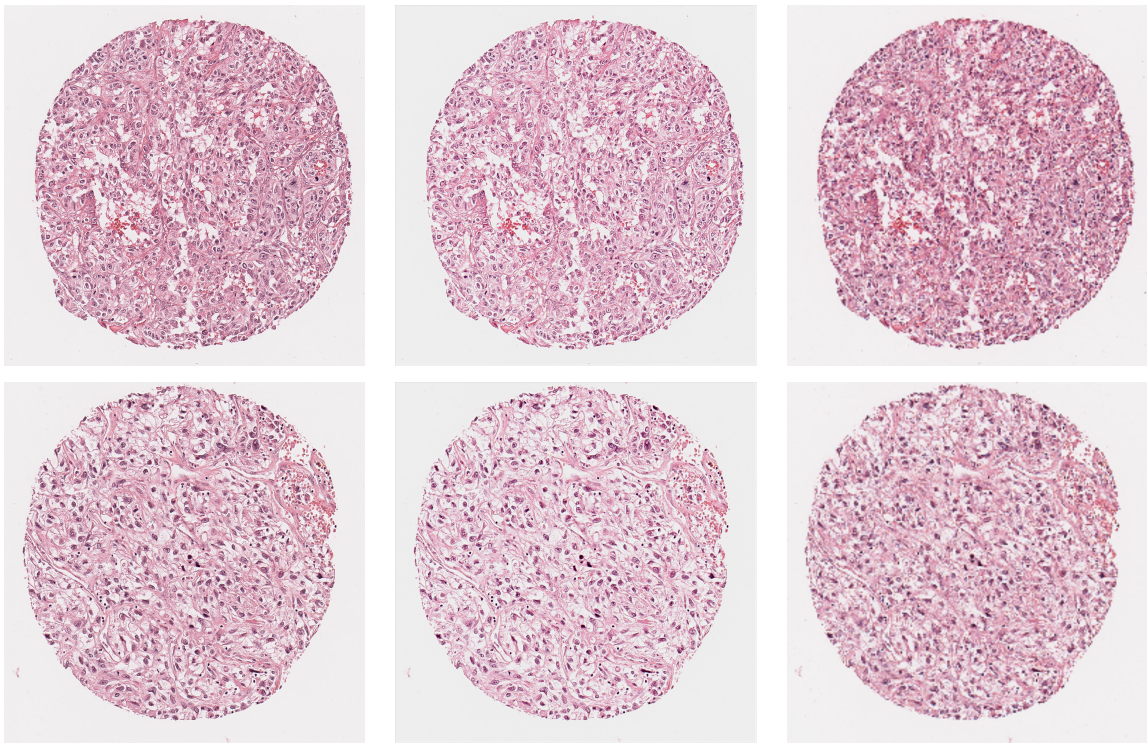


Figure 4.4 Results of reconstruction of Kidney TMA with RCNN: Columns 1 and 3 show high and low resolution images and Column 2 shows the reconstructed image.

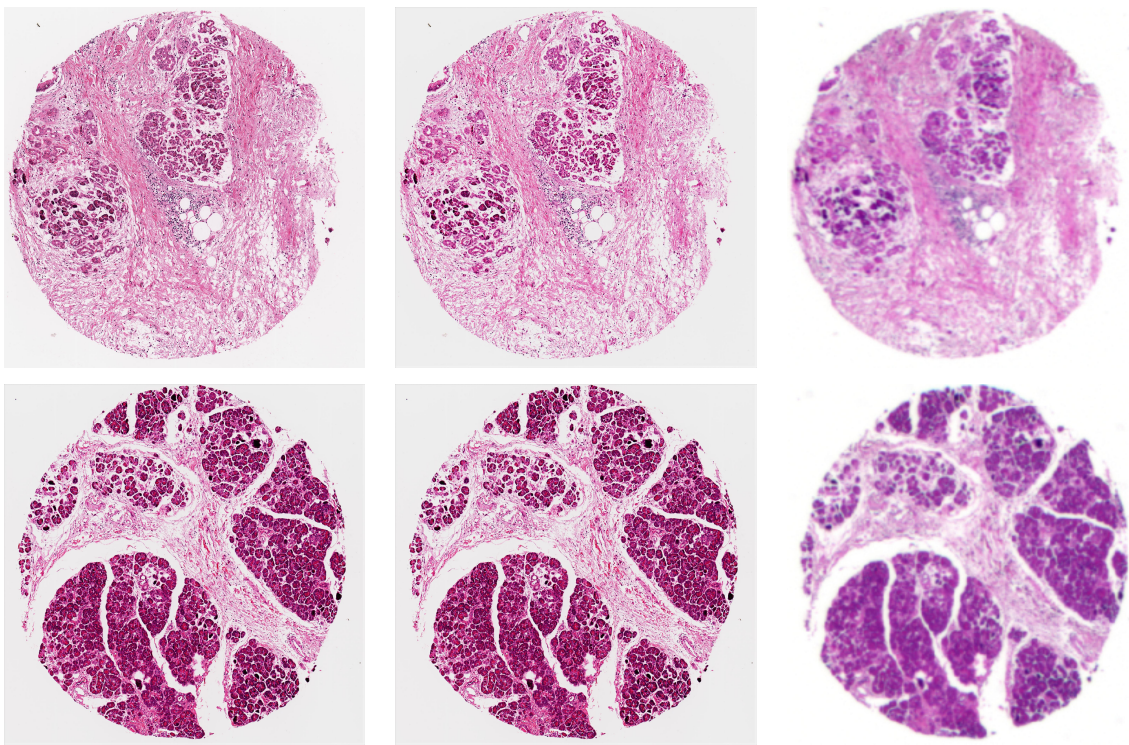


Figure 4.5 Results of reconstruction of Pancreatic TMA with RCNN: Columns 1 and 3 show high and low resolution images and Column 2 shows the reconstructed image.

Breast TMA					
Metric	CNN	RCNN	Metric	CNN	RCNN
RMSE	18.39	15.64	MSSIM	.94	0.95
SNR	22.99	24.36	ITC	2.86	3.16
SSIM	.79	.98	NQM	19.40	20.33
MI	.27	.31	WSNR	23.73	26.59

Table 4.3 Quantitative results from reconstructed Breast images.

Kidney TMA					
Metric	CNN	RCNN	Metric	CNN	RCNN
RMSE	20.48	11.60	MSSIM	.93	.97
SNR	27.04	28.31	ITC	1.99	3.04
SSIM	.75	.98	NQM	13.41	11.15
MI	.21	.35	WSNR	30.05	30.60

Table 4.4 Quantitative results from reconstructed Kidney images.

Pancreas TMA					
Metric	CNN	RCNN	Metric	CNN	RCNN
RMSE	-	20.32	MSSIM	-	0.93
SNR	-	22.07	ITC	-	2.20
SSIM	-	.96	NQM	-	16.94
MI	-	.29	WSNR	-	24.79

Table 4.5 Quantitative results from reconstructed Pancreatic images.

LIST OF REFERENCES

- [1] Tumor grade. <https://www.cancer.gov/about-cancer/diagnosis-staging/prognosis/tumor-grade-fact-sheet>.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [3] Yaniv Bar, Idit Diamant, Lior Wolf, and Hayit Greenspan. Deep learning with non-medical training used for chest pathology identification. In *Medical Imaging 2015: Computer-Aided Diagnosis*, volume 9414, page 94140V. International Society for Optics and Photonics, 2015.
- [4] Sven Behnke. *Hierarchical neural networks for image interpretation*, volume 2766. Springer, 2003.
- [5] Sara L Best et al. Collagen organization of renal cell carcinoma differs between low and high grade tumors. *Urology (submitted)*.
- [6] John A Bullinaria. Recurrent neural networks. *Neural Computation: Lecture*, 12, 2013.
- [7] Norman S Bunker. Optimization of weighted signal-to-noise ratio for a digital video encoder, June 11 1996. US Patent 5,525,984.
- [8] Jose Caballero, Christian Ledig, Andrew P Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *CVPR*, volume 1, page 7, 2017.
- [9] Gail A Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1):54–115, 1987.
- [10] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1256–1272, 2017.
- [11] Matthew W Conklin et al. Aligned collagen is a prognostic signature for survival in human breast carcinoma. *The American journal of pathology*, 178(3):1221–1232, 2011.

- [12] Zhen Cui, Hong Chang, Shiguang Shan, Bineng Zhong, and Xilin Chen. Deep network cascade for image super-resolution. In *European Conference on Computer Vision*, pages 49–64. Springer, 2014.
- [13] Niranjana Damera-Venkata, Thomas D Kite, Wilson S Geisler, Brian L Evans, and Alan C Bovik. Image quality assessment based on a degradation model. *IEEE transactions on image processing*, 9(4):636–650, 2000.
- [14] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199. Springer, 2014.
- [15] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.
- [16] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, pages 391–407. Springer, 2016.
- [17] Hao Dong, Akara Supratak, Luo Mai, Fangde Liu, Axel Oehmichen, Simiao Yu, and Yike Guo. TensorLayer: A Versatile Library for Efficient Deep Learning Development. *ACM Multimedia*, 2017.
- [18] Benito Fernandez, Alexander G Parlos, and Wei Kang Tsai. Nonlinear dynamic system identification using artificial neural networks (anns). In *1990 IJCNN International Joint Conference on Neural Networks*, pages 133–141. IEEE, 1990.
- [19] John R Gilbertson, Jonhan Ho, Leslie Anthony, Drazen M Jukic, Yukako Yagi, and Anil V Parwani. Primary histologic diagnosis using automated whole slide imaging: a validation study. *BMC clinical pathology*, 6(1):4, 2006.
- [20] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 IEEE international conference on*, pages 6645–6649. IEEE, 2013.
- [21] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.
- [22] Hayit Greenspan, Bram Van Ginneken, and Ronald M Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016.
- [23] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

- [24] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *ICML*, pages 399–406, 2010.
- [25] Shuhang Gu, Wangmeng Zuo, Qi Xie, Deyu Meng, Xiangchu Feng, and Lei Zhang. Convolutional sparse coding for image super-resolution. In *IEEE ICCV*, pages 1823–1831, 2015.
- [26] Muhammad Haris, Greg Shakhnarovich, and Norimichi Ukita. Deep backprojection networks for super-resolution. In *Conference on Computer Vision and Pattern Recognition*, 2018.
- [27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [29] Shin Hoo-Chang, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285, 2016.
- [30] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.
- [31] Le Hou, Dimitris Samaras, Tahsin M Kurc, Yi Gao, James E Davis, and Joel H Saltz. Patch-based convolutional neural network for whole slide tissue image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2424–2433, 2016.
- [32] Yan Huang, Wei Wang, and Liang Wang. Bidirectional recurrent convolutional networks for multi-frame super-resolution. In *Advances in Neural Information Processing Systems*, pages 235–243, 2015.
- [33] Jörn-Henrik Jacobsen, Edouard Oyallon, Stéphane Mallat, and Arnold WM Smeulders. Multiscale hierarchical convolutional networks. *arXiv preprint arXiv:1703.04140*, 2017.
- [34] Nazar MT Jawhar. Tissue microarray: a rapidly evolving diagnostic and research tool. *Annals of Saudi medicine*, 29(2):123, 2009.
- [35] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

- [36] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2(2):109–122, 2016.
- [37] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1646–1654, 2016.
- [38] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1637–1645, 2016.
- [39] Tae Hyung Kim and Justin P. Haldar. The fourier radial error spectrum plot: A more nuanced quantitative evaluation of image reconstruction quality. In *Proceedings of the International conference on Biomedical Imaging(ISBI)*, 2018.
- [40] Sonal Kothari, John H Phan, Todd H Stokes, and May D Wang. Pathology imaging informatics for quantitative analysis of whole-slide images. *Journal of the American Medical Informatics Association*, 20(6):1099–1108, 2013.
- [41] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Fast and accurate image super-resolution with deep laplacian pyramid networks. *arXiv preprint arXiv:1710.01992*, 2017.
- [42] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
- [43] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.
- [44] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, 2016.
- [45] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017.
- [46] June-Goo Lee, Sanghoon Jun, Young-Won Cho, Hyunna Lee, Guk Bae Kim, Joon Beom Seo, and Namkug Kim. Deep learning in medical imaging: general overview. *Korean journal of radiology*, 18(4):570–584, 2017.

- [47] Jing Li, Zhichao Lu, Gang Zeng, Rui Gan, and Hongbin Zha. Similarity-aware patchwork assembly for depth image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3374–3381, 2014.
- [48] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *The IEEE conference on computer vision and pattern recognition (CVPR) workshops*, volume 1, page 4, 2017.
- [49] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.
- [50] Ding Liu, Zhaowen Wang, Bihan Wen, Jianchao Yang, Wei Han, and Thomas S Huang. Robust single image super-resolution via deep networks with sparse prior. *IEEE Transactions on Image Processing*, 25(7):3194–3207, 2016.
- [51] Anant Madabhushi and George Lee. Image analysis and machine learning in digital pathology: Challenges and opportunities, 2016.
- [52] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.
- [53] Marvin Minsky and Seymour Papert. Perceptron expanded edition, 1969.
- [54] Lopamudra Mukherjee, Adib Keikhosravi, Dat Bui, and Kevin W Eliceiri. Convolutional neural networks for whole slide image superresolution. *Biomedical Optics Express*, 9(11):5368–5386, 2018.
- [55] Lopamudra Mukherjee, Adib Keikhosravi, and Kevin Eliceiri. Neighborhood regularized image superresolution for applications to microscopic imaging. In *Proceedings of the International conference on Biomedical Imaging(ISBI)*, 2018.
- [56] Ozan Oktay, Wenjia Bai, Matthew Lee, Ricardo Guerrero, Konstantinos Kamnitsas, Jose Caballero, Antonio de Marvao, Stuart Cook, Declan ORegan, and Daniel Rueckert. Multi-input cardiac image super-resolution using convolutional neural networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 246–254. Springer, 2016.
- [57] Christian Osendorfer, Hubert Soyer, and Patrick Van Der Smagt. Image super-resolution with fast approximate convolutional sparse coding. In *International Conference on Neural Information Processing*, pages 250–257. Springer, 2014.
- [58] Liron Pantanowitz et al. Review of the current state of whole slide imaging in pathology. *Journal of pathology informatics*, 2, 2011.

- [59] Liron Pantanowitz, Maryanne Hornish, and Robert A Goulart. The impact of digital imaging in the field of cytopathology. *Cytojournal*, 6, 2009.
- [60] Adhish Prasoon, Kersten Petersen, Christian Igel, François Lauze, Erik Dam, and Mads Nielsen. Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. In *International conference on medical image computing and computer-assisted intervention*, pages 246–253. Springer, 2013.
- [61] M. S. M. Sajjadi, R. Vemulapalli, and M. Brown. Frame-recurrent video super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [62] Mehdi SM Sajjadi, Bernhard Scholkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. *arXiv preprint arXiv:1612.07919*, 2016.
- [63] Samuel Schulter, Christian Leistner, and Horst Bischof. Fast and accurate image upscaling with super-resolution forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3791–3799, 2015.
- [64] H. R. Sheikh, A. C. Bovik, and G. de Veciana. An information fidelity criterion for image quality assessment using natural scene statistics. *IEEE Transactions on Image Processing*, 14(12):2117–2128, 2005.
- [65] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.
- [66] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016.
- [67] Wenzhe Shi, Jose Caballero, Lucas Theis, Ferenc Huszar, Andrew Aitken, Christian Ledig, and Zehan Wang. Is the deconvolution layer the same as a convolutional layer? *arXiv preprint arXiv:1609.07009*, 2016.
- [68] Jake Snell, Karl Ridgeway, Renjie Liao, Brett D Roads, Michael C Mozer, and Richard S Zemel. Learning to generate images with perceptual similarity metrics. In *Image Processing (ICIP), 2017 IEEE International Conference on*, pages 4277–4281. IEEE, 2017.
- [69] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy*, pages 22–29, 2017.

- [70] Radu Timofte, Rasmus Rothe, and Luc Van Gool. Seven ways to improve example-based single image super resolution. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 1865–1873. IEEE, 2016.
- [71] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *CVPR*, pages 5435–5443, 2017.
- [72] Dayong Wang, Aditya Khosla, Rishab Gargeya, Humayun Irshad, and Andrew H Beck. Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718*, 2016.
- [73] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *2015 IEEE ICCV*, pages 370–378, Dec 2015.
- [74] Ronald S Weinstein et al. An array microscope for ultrarapid virtual slide processing and telepathology. design, fabrication, and validation study. *Human pathology*, 35(11):1303–1314, 2004.
- [75] David Wilbur et al. Whole-slide imaging digital pathology as a platform for teleconsultation: a pilot study using paired subspecialist correlations. *Archives of pathology & laboratory medicine*, 133(12):1949–1953, 2009.
- [76] David C Wilbur. Digital cytology: current state of the art and prospects for the future. *Acta cytologica*, 55(3):227–238, 2011.
- [77] Bingzhe Wu, Haodong Duan, Zhichao Liu, and Guangyu Sun. Srgan: Perceptual generative adversarial network for single image super resolution. *CoRR*, abs/1712.05927, 2017.
- [78] Yan Xu, Tao Mo, Qiwei Feng, Peilin Zhong, Maode Lai, I Eric, and Chao Chang. Deep learning of feature representation with multiple instance learning for medical image analysis. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1626–1630. IEEE, 2014.
- [79] Jianchao Yang, Zhaowen Wang, Zhe Lin, Scott Cohen, and Thomas Huang. Coupled dictionary training for image super-resolution. *IEEE transactions on image processing*, 21(8):3467–3478, 2012.
- [80] Jianchao Yang, John Wright, Thomas Huang, and Yi Ma. Image super-resolution as sparse representation of raw image patches. In *CVPR*, pages 1–8. IEEE, 2008.
- [81] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE TIP*, 19(11):2861–2873, 2010.
- [82] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

APPENDIX

Python Code

The implementation of presented model in python code.

```
import numpy as np
import tensorflow as tf
from tensorflow.python.estimator.model_fn import ModeKeys as Modes

from config import FLAGS
from subpixel import phase_shift
from utils import tf_slice

LOG_EVERY_STEPS = 10

SUMMARY_EVERY_STEPS = 100

def model_fn(features, labels, mode, params):
    learning_rate = params.learning_rate
    devices = [('/device:%s' % d) for d in params.device.split(',')]
    for d in devices:
        with tf.device(d):
            with tf.name_scope('inputs'):
                lr_images = tf_slice(features[0], 0)
                int1_images = tf_slice(features[1], 0)
                int2_images = tf_slice(features[2], 0)
                hr_images = tf_slice(labels, 0)

    _, _, predictions = rcnn(lr_images, int1_images, int2_images, devices)

    if mode in (Modes.TRAIN, Modes.EVAL):
        with tf.name_scope('losses'):
            mse = tf.losses.mean_squared_error(hr_images, predictions)
            rmse = tf.sqrt(mse)
            psnr = tf_psnr(mse)
```

```

        ssim = tf_ssim(hr_images, predictions)
        loss = 0.75 * rmse + 0.25 * (1 - ssim)
    with tf.name_scope('train'):
        train_op = tf.train.AdamOptimizer(learning_rate)
            .minimize(loss, tf.train.get_global_step())

if mode in (Modes.TRAIN, Modes.EVAL):
    tf.summary.scalar('mse', mse)
    tf.summary.scalar('rmse', rmse)
    tf.summary.scalar('psnr', psnr)
    tf.summary.scalar('ssim', ssim)
    tf.summary.scalar('loss', loss)
    # tf.summary.image('predictions', predictions, max_outputs=1)

    summary_op = tf.summary.merge_all()
    summary_hook = tf.train.SummarySaverHook(save_steps=SUMMARY_EVERY_STEPS,
        output_dir=FLAGS.summaries_dir, summary_op=summary_op)

    logging_params = {'mse': mse, 'rmse': rmse, 'ssim': ssim,
        'psnr': psnr, 'loss': loss, 'step': tf.train.get_global_step()}
    logging_hook = tf.train.LoggingTensorHook(logging_params,
        every_n_iter=LOG_EVERY_STEPS)

    # eval_metric_ops = {
    #     "rmse": tf.metrics.root_mean_squared_error(features, predictions)
    # }
    estimator_spec = tf.estimator.EstimatorSpec(
        mode=mode,
        loss=mse,
        predictions=predictions,
        train_op=train_op,
        training_hooks=[logging_hook, summary_hook]
    )
else:
    # mode == Modes.PREDICT:
    export_outputs = {
        'predictions': tf.estimator.export.PredictOutput(
            {'high_res_images': predictions}
        )
    }
    estimator_spec = tf.estimator.EstimatorSpec(
        mode=mode,

```

```

        predictions=predictions,
        export_outputs=export_outputs
    )
return estimator_spec

def conv(inputs, weights):
    return tf.nn.conv2d(inputs, weights, strides=[1, 1, 1, 1], padding='SAME')

def cnn(lr_images, output_size, devices=['/device:CPU:0']):
    size = lr_images.get_shape().as_list()[1]
    ratio = int(output_size / size)
    output_channels = ratio * ratio if ratio > 1 else ratio
    filters_shape = [2, 1, 3, 2, 1]
    filters = [64, 32, 16, 8, output_channels]
    channels = lr_images.get_shape().as_list()[3]
    for d in devices:
        with tf.device(d):
            with tf.name_scope('weights'):
                w1 = tf.get_variable(initializer=tf.random_normal(
                    [filters_shape[0], filters_shape[0], channels, filters[0]],
                    stddev=1e-3), name='cnn_w1')
                w2 = tf.get_variable(initializer=tf.random_normal(
                    [filters_shape[1], filters_shape[1], filters[0], filters[1]],
                    stddev=1e-3), name='cnn_w2')
                w3 = tf.get_variable(initializer=tf.random_normal(
                    [filters_shape[2], filters_shape[2], filters[1], filters[2]],
                    stddev=1e-3), name='cnn_w3')
                w4 = tf.get_variable(initializer=tf.random_normal(
                    [filters_shape[3], filters_shape[3], filters[2], filters[3]],
                    stddev=1e-3), name='cnn_w4')
                w5 = tf.get_variable(initializer=tf.random_normal(
                    [filters_shape[4], filters_shape[4], filters[3], filters[4]],
                    stddev=1e-3), name='cnn_w5')
            with tf.name_scope('biases'):
                b1 = tf.get_variable(initializer=tf.zeros(filters[0]), name='cnn_b1')
                b2 = tf.get_variable(initializer=tf.zeros(filters[1]), name='cnn_b2')
                b3 = tf.get_variable(initializer=tf.zeros(filters[2]), name='cnn_b3')
                b4 = tf.get_variable(initializer=tf.zeros(filters[3]), name='cnn_b4')
                b5 = tf.get_variable(initializer=tf.zeros(filters[4]), name='cnn_b5')
            with tf.name_scope('predictions'):

```

```

conv1 = tf.nn.bias_add(conv(lr_images, w1), b1, name='conv_1')
conv1r = tf.nn.leaky_relu(conv1, name='relu_1')
conv2 = tf.nn.bias_add(conv(conv1r, w2), b2, name='conv_2')
conv2r = tf.nn.leaky_relu(conv2, name='relu_2')
conv3 = tf.nn.bias_add(conv(conv2r, w3), b3, name='conv_3')
conv3r = tf.nn.leaky_relu(conv3, name='relu_3')
conv4 = tf.nn.bias_add(conv(conv3r, w4), b4, name='conv_4')
conv4r = tf.nn.leaky_relu(conv4, name='relu_4')
conv5 = tf.nn.bias_add(conv(conv4r, w5), b5, name='conv_5')
upscaled = tf.tanh(phase_shift(conv5, ratio))
predictions = upscaled if ratio > 1 else conv5
return predictions

```

```

def rcnn(in_images, inter1, inter2, devices=['/device:CPU:0']):
    def hidden_layer(img1, img2, img3, w, wr, wt, b, number='1'):
        h_x1 = tf.nn.leaky_relu(tf.nn.bias_add(conv(img1, w), b, name='h_x1_' + number))

        r_x2 = tf.image.resize_bicubic(conv(h_x1, wr), [512, 512])
        t_x2 = tf.image.resize_bicubic(conv(img1, wt), [512, 512])
        x2 = tf.add(conv(img2, w), tf.add(r_x2, t_x2))
        h_x2 = tf.nn.leaky_relu(tf.nn.bias_add(x2, b, name='h_x2_' + number))

        r_x3 = tf.image.resize_bicubic(conv(h_x2, wr), [1024, 1024])
        t_x3 = tf.image.resize_bicubic(conv(img2, wt), [1024, 1024])
        x3 = tf.add(conv(img3, w), tf.add(r_x3, t_x3))
        h_x3 = tf.nn.leaky_relu(tf.nn.bias_add(x3, b, name='h_x3_' + number))

        return h_x1, h_x2, h_x3

    channels = in_images.get_shape().as_list()[3]
    fshape = [3, 1, 2]
    fnums = [32, 16, 4]

    with tf.variable_scope('first_layer', reuse=tf.AUTO_REUSE):
        # first hidden layer
        w1 = tf.get_variable(initializer=tf.random_normal(
            [fshape[0], fshape[0], channels, fnums[0]], stddev=1e-3), name='cnn_w1')
        wr1 = tf.get_variable(initializer=tf.random_normal(
            [1, 1, fnums[0], fnums[0]], stddev=1e-3), name='cnn_wr1')
        wt1 = tf.get_variable(initializer=tf.random_normal(
            [fshape[0], fshape[0], channels, fnums[0]], stddev=1e-3), name='cnn_wt1')

```

```

b1 = tf.get_variable(initializer=tf.zeros(fnums[0]), name='cnn_b1')

with tf.variable_scope('second_layer', reuse=tf.AUTO_REUSE):
    # second hidden layer
    w2 = tf.get_variable(initializer=tf.random_normal(
        [fshape[1], fshape[1], fnums[0], fnums[1]], stddev=1e-3), name='cnn_w2')
    wr2 = tf.get_variable(initializer=tf.random_normal(
        [1, 1, fnums[1], fnums[1]], stddev=1e-3), name='cnn_wr2')
    wt2 = tf.get_variable(initializer=tf.random_normal(
        [fshape[1], fshape[1], fnums[0], fnums[1]], stddev=1e-3), name='cnn_wt2')
    b2 = tf.get_variable(initializer=tf.zeros(fnums[1]), name='cnn_b2')

with tf.variable_scope('third_layer', reuse=tf.AUTO_REUSE):
    # third hidden layer
    w3 = tf.get_variable(initializer=tf.random_normal(
        [fshape[2], fshape[2], fnums[1], fnums[2]], stddev=1e-3), name='cnn_w3')
    wr3 = tf.get_variable(initializer=tf.random_normal(
        [1, 1, fnums[2], fnums[2]], stddev=1e-3), name='cnn_wr3')
    wt3 = tf.get_variable(initializer=tf.random_normal(
        [fshape[2], fshape[2], fnums[1], fnums[2]], stddev=1e-3), name='cnn_wt3')
    b3 = tf.get_variable(initializer=tf.zeros(fnums[2]), name='cnn_b3')

inter2 = tf.zeros(tf.shape(inter2))
for d in devices:
    with tf.device(d):
        l1_h1, l1_h2, l1_h3 = hidden_layer(in_images, inter1, inter2, w1, wr1, wt1, b1)

        l2_h1, l2_h2, l2_h3 = hidden_layer(l1_h1, l1_h2, l1_h3, w2, wr2, wt2, b2, '2')

        h1, h2, h3 = hidden_layer(l2_h1, l2_h2, l2_h3, w3, wr3, wt3, b3, '3')

        p1 = tf.tanh(phase_shift(h1, 2))
        p2 = tf.tanh(phase_shift(h2, 2))
        p3 = tf.tanh(phase_shift(h3, 2))

return p1, p2, p3

```