Deep Learning basierte Auflösungserhöhung im Kontext der Videocodierung

Deep Learning based Super-Resolution in the context of Video Coding

Masterarbeit

Von / *by*
Nianlong Gu
Matr.-Nr. 374011

Diese Arbeit wurde vorgelegt am / *the present work was submitted to*
Lehrstuhl und Institut für Nachrichtentechnik

Univ.-Prof. Dr.-Ing. Jens-Rainer Ohm

Aachen, 1. June 2018

Contents

Chapter 1

Introduction

Single image super resolution (SISR) aims to recover the high-resolution image given the low-resolution image. This is an ill-posed problem whose solution is undetermined [1]. Early researches adopt interpolation methods, such as bicubic interpolation and discrete cosine transform (DCT) based interpolation. The state-of-art method is to use deep learning based convolution neural network (CNN) to up-sample low-resolution images. Among earliest works, Dong et al. [1] designed a 3-layer CNN model, named SRCNN, to achieve end-to-end mapping between low and full resolution images. They also proved the relationship between SRCNN and sparse-coding-based methods [2]. SRCNN model shows great improvement in accuracy compared with interpolation methods or sparse-coding-based methods. Kim et al. [3] went one step further by designing a very deep CNN model, named VDSR. Different from [1], in [3] the authors dropped the structural mapping to sparse-coding-based methods. Instead, they increased the number of CNN layers to 20, and proved that deeper networks can dig out more useful information and further increase the accuracy. Besides, they used the concept of residual learning by adding a shortcut from input to output into the CNN model.

Current researches on image super resolution can be classified into two classes. One class is to further improve the accuracy by making modification to the deep neural network structure. Kim et al. [4] proposed a deeply-recursive convolution network (DRCN) based on the VDSR model. It also uses a 20-layer convolution network, but the difference from VDSR is that in DRCN there are several recursive layers, and these recursive layers share the same filter kernel. Testing results show that DRCN can further improve the performance over VDSR by a small degree.
Another class is to explore shallow convolution networks with equally good performance and less computation cost. Li et al. [5] designed a 5-layer residual neural network for up-sampling and coding artifacts removal. This network is much shallower than VDSR. It adopts multi-scale feature extraction by using multi-scale filter size in the second and fourth layer. They showed that this 5-layer model can achieve slightly better performance than VDSR when tested on video sequences.

Recently, SR also shows its potential in video coding [6][7]. As the resolution of video sequence increases, more efficient video coding schemes are required to reduce the bit rate while maintaining a low level of distortion. It has been shown that compared with directly coding a frame at its full resolution, lower rate distortion cost can probably be achieved by first down-sampling some of the coding units and then coding them, especially in low bit rate scenarios [6][8]. In order to save more bit rate via low resolution coding scheme, it is essential to find effective SR methods which can recover the high resolution image with

good quality and show good robustness against coding artifacts.

The reminder of this thesis is organized as follows. In Chapter 2 the fundamentals of deep-learning based SR models and evaluation methods will be introduced. In Chapter 3, the performance of current SISR models is analyzed in the frequency domain. An auto-encoder [9] based SR model is designed and implemented into intra-frame video coding scheme in Chapter 4. In Chapter 5, we analyze the performance of auto-encoder model and compare BD-rate gains with the state-of-art work. Finally, this thesis is concluded in Chapter 6.

Chapter 2

Fundamentals

This chapter will provide an overview of existing deep-learning based SISR models, the SRCNN and the VDSR, as well as a 5-layer CNN model. Then we will compare their structures and main properties. Necessary evaluation measures and tools will be introduced.

2.1 Deep-learning based SISR Models

Given a low-resolution image, the task of a SISR model is to recover the corresponding full-resolution image, through a series of convolution and non-linear operations. The general structure of deep-learning based SISR is shown in Figure 2.1. The low-resolution image is first up-scaled via interpolation methods such as bicubic or Scalable Extensions of the High Efficiency Video Coding (SHVC) interpolation. The interpolated image, which is regarded as the input, is feed into a convolution network for processing. The output of the convolution network is the up-sampled image with high accuracy. The CNN part is a network with multiple convolution layers serially concatenated, and there is a non-linear activation function at the output of each hidden layer. Deep-learning based SISR models can be classified into non-residual and residual learning according to the structure of convolution network.

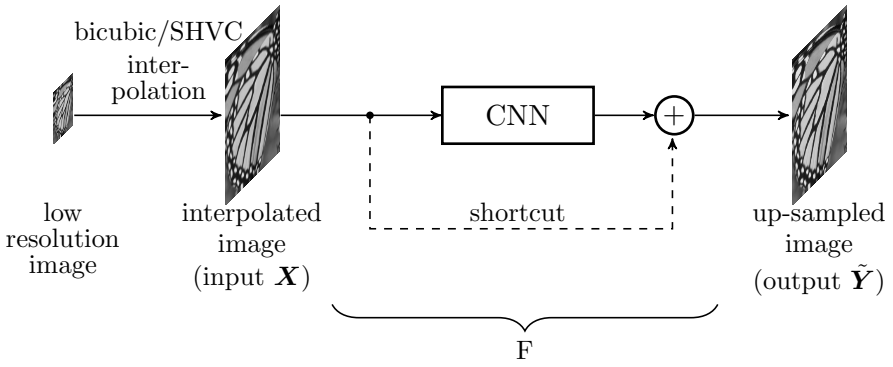


Figure 2.1: The general structure of a deep-learning based SISR model.

When there is no shortcut from the input to the output of CNN, this kind of deep-learning model is non-residual learning. What the convolution network learns is the non-linear mapping from interpolated images to high-resolution images. Let $\boldsymbol{X}$ be the input image, $\tilde{\boldsymbol{Y}}$ be the output image and F be the mapping function, then the CNN model can be expressed mathematically:

$$\tilde{\boldsymbol{Y}} = \mathrm{F}(\boldsymbol{X}) \tag{2.1}$$

When there is a shortcut from the input to output of CNN, as is shown in Figure 2.1, this type of learning model is residual learning. In a residual learning model, what the convolution network learns is the non-linear mapping from interpolated images to residual images. The residual images are ground truth images minus interpolated images. Let $\boldsymbol{X}$ be the input image, $\tilde{\boldsymbol{Y}}$ be the output image and G be the mapping function from the interpolated image to the residual, then the overall function of CNN model can be expressed mathematically:

$$\tilde{\boldsymbol{Y}} = \boldsymbol{X} + \mathrm{G}(\boldsymbol{X}) \tag{2.2}$$

In this section we briefly review some basic concepts in the field of deep-learning based SISR. Then three deep-learning based SISR models: the SRCNN, the VDSR and the 5-layer CNN model will be introduced. The SRCNN is a non-residual learning model while the VDSR and the 5-layer CNN model are residual learning models. We will introduce these three models from the aspect of network structure, training and main characteristics.

2.1.1 Basic Concepts in Deep-Learning based SISR

In this section we review some basic concepts for deep-learning based SISR, including convolution and deconvolution, stride and padding.

Convolution

Suppose the input image is $\boldsymbol{X}$ and the convolution filter is $\mathbf{W}$, the convolution operation in deep-learning is shown in Figure 2.2.

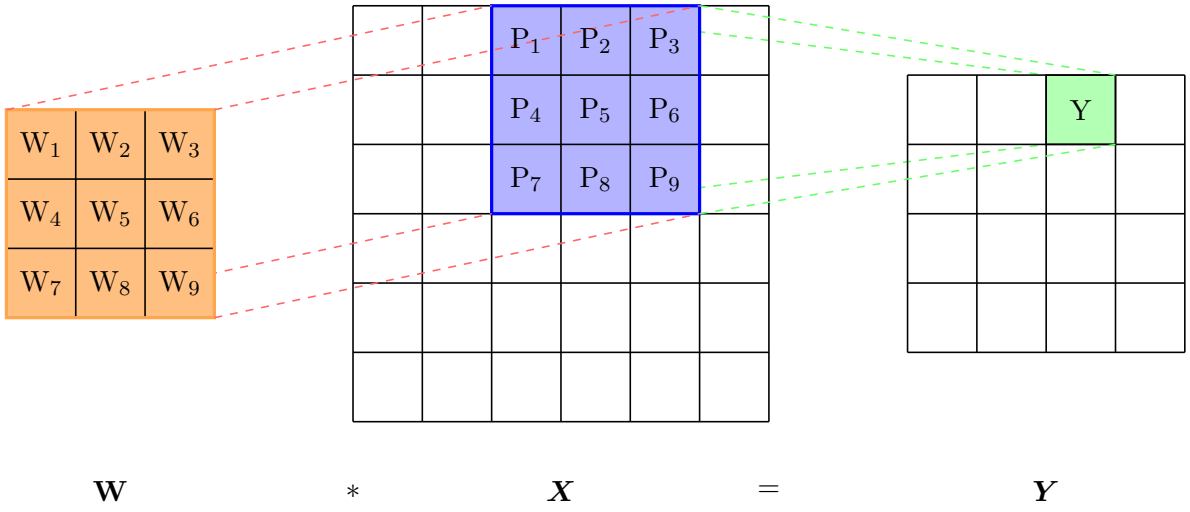


Figure 2.2: Convolution operation.

The value of the pixel in the output image can be expressed by:

$$\mathrm{Y} = \sum_{i=1}^{9} \mathrm{W}_i \cdot \mathrm{P}_i \tag{2.3}$$

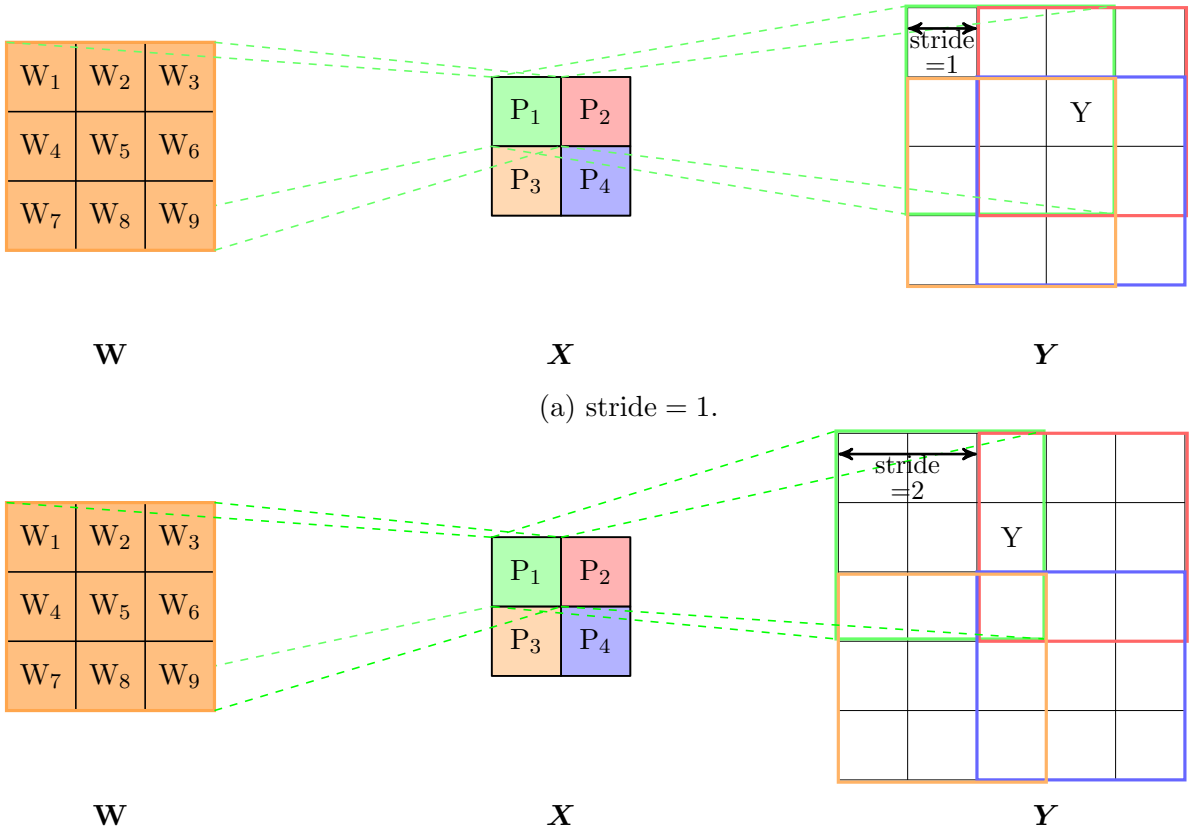
According to (2.3), the convolution operation in deep-learning networks is actually cross-correlation, since the filter is convolved with the input matrix without reversing the filter.

After convolution, the output image has a smaller size than the input image. In deep-learning networks, the common choice is to make the input image and the output image of the convolution have the same size. Therefore, the input image is first enlarged by padding. The padding method is to add extra pixels around the border of an image. Main padding methods include zero-padding which adds zeros around the image border, and replicate padding which replicates the pixel values in the image border.

The filter convolves with the input matrix, and move by certain pixels after each computation. The step size of moving is called stride. In convolution operation, stride can determine the scale of the convolution output. When the stride is 1, the output has the same scale as the input. When the stride is 2, the output of the convolution operation is down-scale into 1/2 of the original size.

Therefore, we can down-scale the convolution output by adjust the stride. In practice, the convolution neural network can be used to train a down-sampling model. This is will discussed in Section 4.2.

Deconvolution



(a) stride = 1.

(b) stride = 2.

Figure 2.3: Deconvolution operation.

Deconvolution is also called transposed convolution. For input image $\boldsymbol{X}$ and deconvolution

filter $\mathbf{W}$, the deconvolution output can be written as $\boldsymbol{Y} = \text{Deconv}(\mathbf{W}, \boldsymbol{X})$. Figure 2.3 shows deconvolution in detail.

As shown in Figure 2.3, during the deconvolution operation, the filter kernel is multiplied by a pixel in the input image, and becomes a matrix weighted by the pixel value. For each pixel, we can get a weighted-filter. These weighted filters are partially overlapped. We can get the deconvolution output by aggregating all the weighted filters.

The distance between two successive weighted filters in the output image is the stride. When the stride is 1, it means the step size between two weighted filters is 1. In this case, the output has the same scale as the input. As shown in Figure 2.3a, the pixel Y in the deconvolution image $\boldsymbol{Y}$ for stride 1 can be expressed by

$$\mathrm{Y} = \mathrm{P}_1 \cdot \mathrm{W}_6 + \mathrm{P}_2 \cdot \mathrm{W}_5 + \mathrm{P}_3 \cdot \mathrm{W}_3 + \mathrm{P}_4 \cdot \mathrm{W}_2 \tag{2.4}$$

When the stride is 2, the output is up-scaled by a factor of 2. As shown in Figure 2.3b, the pixel Y in the deconvolution image $\boldsymbol{Y}$ for stride 2 can be expressed by

$$\mathrm{Y} = \mathrm{P}_1 \cdot \mathrm{W}_6 + \mathrm{P}_2 \cdot \mathrm{W}_4 \tag{2.5}$$

The deconvolution neural network can be used to train a up-sampling model, and this will further discussed in Section 4.2.

2.1.2 SRCNN

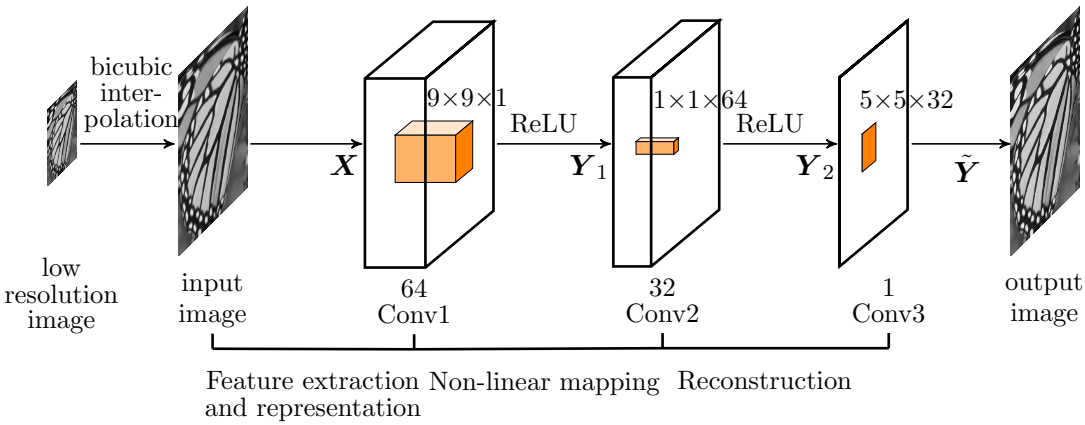


Figure 2.4: The structure of the SRCNN model [1]. For each layer, the number above represents the size of the filter kernel, and the number below represents the number of channels. In this thesis we follow this notation style.

The structure of the SRCNN model [1] is shown in Figure 2.4. As a non-residual learning model, the SRCNN model contains three convolution layers.
The first layer has a kernel size 9×9×1, which means when convolving the input matrix has

1 channel and the size of the convolution filter is 9×9×1. The number of channels of first layers is 64, which implies there are 64 different convolution filters for the first layer, and therefore the output of this layer has 64 channels. Then activation function Rectified Linear Unit (ReLU) is applied to the convolution results. It has been proven that compared with other activation functions such as tanh, using ReLU can help to reduce gradient vanishing problem, which makes it very suitable in deep network training [10]. Let $\boldsymbol{X}$ be the input of the first convolution layer, $\mathbf{W}_1$ and $\mathbf{B}_1$ be the filter and bias of the first layer and '*' represent the convolution operation. This layer can be mathematically expressed by

$$\begin{aligned} \boldsymbol{Y}_1 &= \mathrm{F}_1(\boldsymbol{X}) \\ \mathrm{F}_1(\boldsymbol{X}) &= \max(\mathbf{W}_1 * \boldsymbol{X} + \mathbf{B}_1, 0) \end{aligned} \tag{2.6}$$

The second layer takes the output of the first layer as input. The kernel size of this layer is 1×1×64, and the number of convolution output channels is 32. The activation function of this layer is also ReLU. This layer can be mathematically expressed by

$$\begin{aligned} \boldsymbol{Y}_2 &= \mathrm{F}_2(\boldsymbol{Y}_1) \\ \mathrm{F}_2(\boldsymbol{Y}_1) &= \max(\mathbf{W}_2 * \boldsymbol{Y}_1 + \mathbf{B}_2, 0) \end{aligned} \tag{2.7}$$

The last layer of this network takes the output of the second layer as input, and its output is the up-sampled image. Therefore, this layer has only one channel in the case of luma-only images, and has 3 channels in the case of luma-and-chroma images. Note that the last layer does not have activation function. We use $\tilde{\boldsymbol{Y}}$ to express the final output of this model, then mathematical description of this layer is

$$\begin{aligned} \tilde{\boldsymbol{Y}} &= \mathrm{F}_3(\boldsymbol{Y}_2) \\ \mathrm{F}_3(\boldsymbol{Y}_2) &= \mathbf{W}_3 * \boldsymbol{Y}_2 + \mathbf{B}_3 \end{aligned} \tag{2.8}$$

Combining (2.6) to (2.8) we can get the overall expression of the SRCNN model:

$$\tilde{\boldsymbol{Y}} = \mathrm{F}_3(\mathrm{F}_2(\mathrm{F}_1(\boldsymbol{X}))) \tag{2.9}$$

During the training of the SRCNN, the low-resolution images are first up-scaled via the bicubic interpolation method. The bicubic interpolated images are used as input data for the training of the SRCNN model. For each iteration, a batch of input image $\{\boldsymbol{X}_i\}$ is fed into the network and the output batch of the SRCNN can be expressed by $\{\mathrm{F}(\boldsymbol{X}_i, \Theta)\}$, where Θ are the network parameters. Suppose that the ground truth high-resolution image batch is $\{\boldsymbol{Y}_i\}$ and the batch size is n, then the loss function is the square of L^2 norm of the error signal, averaged over the batch size. This type of loss function favors a high peak signal-to-noise ratio (PSNR).

$$L_{\mathrm{SRCNN}}(\Theta) = \frac{1}{n} \sum_{i=1}^{n} ||\mathrm{F}(\boldsymbol{X}_i, \Theta) - \boldsymbol{Y}_i||^2 \tag{2.10}$$

The main characteristics of the SRCNN is that it is the pioneer in the field of deep-learning based SISR, and the authors in [1] established the relationship between SRCNN and sparse-coding-based SR methods from three aspects:

1. In sparse-coding-based methods, the sparse coding solver first extract each overlapping patch, then project it onto a low resolution dictionary. Supposing that the patch size is $f_1 \times f_1$ and this dictionary has n_1 coefficients, this is corresponding to convolving the input matrix with a filter which has n_1 channels and a size of $f_1 \times f_1$.

2. The sparse coding solver will recursively process the n_1 coefficients and map them into a high resolution dictionary which contains n_2 coefficients. This operation is non-linear, so it is corresponding to the non-linear activation function (ReLU) of the first layer and the whole second layer of the SRCNN model.

3. The high-resolution patch-wise representations are then projected onto the high resolution patches. This is corresponding to the third layer of the SRCNN model.

Although the results of other deep learning models (VDSR, DRCN, etc) shows that this strict analogy to sparse-based-coding may not be necessary, this correspondence does provide us with an intuition of what goal CNN models are trying to achieve in each layer. Besides, some characteristics of the SRCNN model are also adopted by other CNN models, such as using ReLU as activation function and leaving the last layer without any activation function.

2.1.3 VDSR

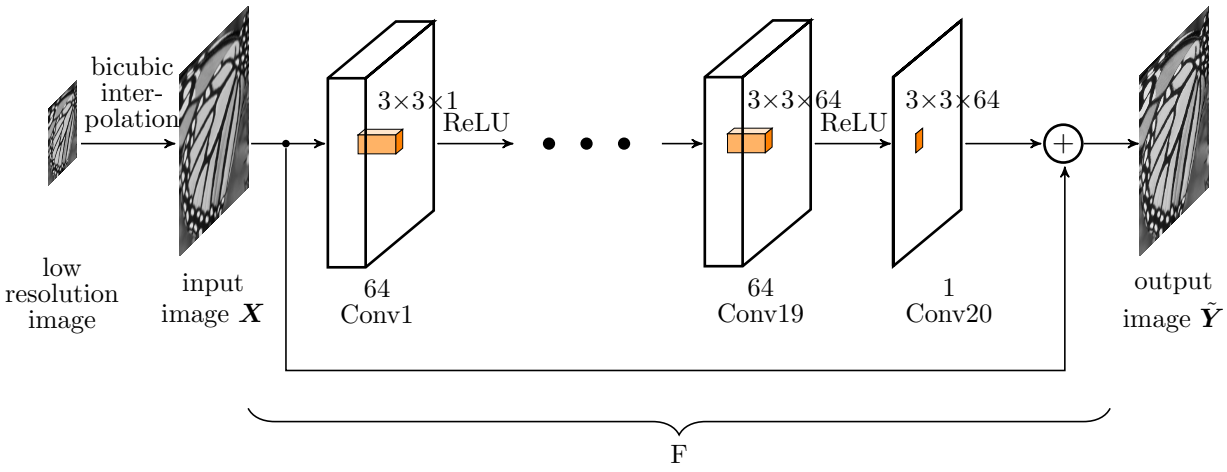


Figure 2.5: The structure of the VDSR model [3].

The structure of the VDSR model [3] is shown in Figure 2.5. Like the SRCNN model, the low resolution image is first up-sampled via bicubic interpolation. Then there are two branches for the interpolated image. One branch is the CNN network which contains 20 convolution layers. Except the first and the last layer, all the other layers have the same structure: 64 channels, a filter kernel with size $3 \times 3 \times 64$, and ReLU as the activation function. The first layer has a filter kernel with size $3 \times 3 \times 1$ and uses the ReLU as the activation function. The last layer is similar to that of the SRCNN, whose function is linear reconstruction. Therefore, the last layer has the same number of channels as the input, the filter size is also

$3 \times 3\times 64$, and there is no activation function for this layer. Another branch is the shortcut from the input to the output of the convolution layers. The final output of the VDSR is the sum of input image and output of the last convolution layer. We use $\boldsymbol{X}$ to express the the bicubic-interpolated input image. $\tilde{\boldsymbol{Y}}$ is the final output of the VDSR. F_i represents the non-linear function of i^{th} convolution layer. Mathematical description of this model is

$$\begin{aligned} \tilde{\boldsymbol{Y}} &= \mathrm{F}(\boldsymbol{X}) \\ \mathrm{F}(\boldsymbol{X}) &= \boldsymbol{X} + \mathrm{F}_{20}(\mathrm{F}_{19}(...\mathrm{F}_1(\boldsymbol{X}))...) \end{aligned} \tag{2.11}$$

In [3] the authors introduced the concept of the receptive filed and stated that a larger receptive filed is beneficial to the learning accuracy. The receptive field is used to express the input image filed which is related to one pixel in the output image. This measurement is closely related with the filter size and depth of the network. Suppose that the network has N layers, and the filter size of layer i is $f_i \times f_i$, then the receptive field F_{r} can be expressed by

$$F_{\mathrm{r}} = \sum_{i=1}^{N}(f_i - 1) + 1 \ \times \ \sum_{i=1}^{N}(f_i - 1) + 1 \tag{2.12}$$

For example, the receptive field of the SRCNN model is 13×13, while the receptive field of the VDSR is 41×41.

The training configuration of the VDSR is similar to that of the SRCNN. The low-resolution images are first up-scaled via the bicubic interpolation method. The bicubic interpolated images are used as input data for the training of the VDSR model. Stochastic gradient descent method is used to minimize the loss function.

$$L_{\mathrm{VDSR}}(\Theta) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}||\mathrm{F}(\boldsymbol{X}_i, \Theta) - \boldsymbol{Y}_i||^2 \tag{2.13}$$

This loss function is almost identical to (2.10), since both loss functions are mean square error (MSE) based.

One big issue in the training of very deep convolution network is gradient exploding and vanishing problem. In GDS minimization, the learning rate is multiplied with gradient to determine the step size. In order to stabilize the training the learning rate usually starts with a high value and decreases exponentially every certain epochs. When the learning rate is high, the step size can probably becomes too large to make the loss value diverge. When the learning rate decrease exponentially to a tiny value, the step size may become so small that the learning converges very slowly. To tackle this issue, in [3] the author adopted adjustable gradient clipping to limit the step size within a reasonable range. They showed that using gradient clipping can make the training quickly converge.

The main contribution of the VDSR model is that it first showed the feasibility of using very deep networks to deal with the SISR problem. Different from the SRCNN which only has 3 convolution layers, the VDSR model has a deeper network. The authors in [3] did not pursue a strong analogy to sparse-coding-based methods in network structure. Instead, they designed a 20-layer residual network with general setting (3×3 filter, 64 channels)

and showed that the VDSR model can achieve higher accuracy than the SRCNN model. Motivated by the success of the VDSR, models with even deeper network are tested and proved to be able to achieve better performance [3].

Besides, the VDSR model adopted residual learning structure to increase the convergence speed, and proposed the adjustable gradient clipping method to deal with the gradient vanishing and exploding problems. All these works provide useful guidance and experiences for the future design of deep-learning networks in the field of SISR.

5-layer CNN Model

In this thesis we designed a 5-layer CNN model based on the structure of the VDSR model. The 5-layer CNN model is also a residual-learning model. Compared with the VDSR model, the only difference is that the 5-layer CNN model only has 5 convolution layers, while the VDSR has 20 convolution layers. Except that, all the other configurations about the network structure are the same. For the training of the 5-layer CNN model, the loss function is the same as the loss function of the VDSR:

$$L_{\text{5layerCNN}} = L_{\text{VDSR}} \tag{2.14}$$

All the hyper-parameters of training are the same as those of the VDSR model. The purpose of using the 5-layer CNN model is to test the influence of network depth on the performance of the deep learning model.

2.1.4 Comparison of SRCNN, VDSR and 5-layer CNN

In this section we provide a summarized comparison of the SRCNN, the VDSR and the 5-layer CNN models in terms of network structure and training configurations.

Table 2.1: Comparison of VDSR and SCRNN in terms of network structure.

model	input	number of layers	activation function	residual learning	filter size	receptive field	loss function
SRCNN	bicubic interpolated images	3	ReLU	no	9×9/1×1/5×5	13×13	$L_{\text{SRCNN}}(\Theta)$
VDSR	bicubic interpolated images	20	ReLU	yes	3×3 for all layers	41×41	$L_{\text{VDSR}}(\Theta)$
5-layer CNN	bicubic interpolated images	5	ReLU	yes	3×3 for all layers	11×11	$L_{\text{5layerCNN}}(\Theta)$

From the comparison, we can see that the VDSR model and the SRCNN model have the same input, activation function and equivalent loss function. The main differences are the network depth and using skip connection or not. Besides, the only difference between the VDSR and the 5-layer CNN is the network depth. In [3] the authors proved that the VDSR model has a better performance than the SRCNN model due to the usage of the skip connection and gradient clipping methods.

2.2 Quality Evaluation

Numerically evaluation of the performance of SR models involves using different measures such as Peak Signal-to-Noise Ratio (PSNR) and structure similarity index (SSIM) [11][12]. These measures are related with the perceived quality of an image with respect to the ground truth image. In this section we will briefly review existing measures for image quality evaluation. A new measure, named Energy Signal-to-Noise Ratio (ESNR), which has good flexibility in analyzing the performance of SR models in frequency domain will be mainly discussed. Besides, we will introduce a windowing method which is useful for analysis in the frequency domain.

2.2.1 PSNR

PSNR represents the ratio of maximum possible energy of an image to the averaged noise energy, expressed in logarithmic decibel scale.
When computing PSNR, we first compute the MSE of the input image $\boldsymbol{X}$ with respect to the ground truth image $\boldsymbol{Y}$. Suppose that the image size is $M \times N$, then MSE can be expressed by

$$\mathrm{MSE} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |X(m,n) - Y(m,n)|^2 \tag{2.15}$$

The PSNR can be computed by

$$\mathrm{PSNR} = 10\log\left(\frac{A_{\max}^2}{\mathrm{MSE}}\right) \tag{2.16}$$

In (2.16), the $A_{\max}$ in the numerator is the maximum possible pixel value in an image. If the bit depth of an image is expressed by B, then $A_{\max} = 2^B - 1$. For example $A_{\max}$ is 255 for 8-bit color images. The denominator MSE is the average squared fluctuation of the input image $\boldsymbol{X}$, compared with the ground truth image $\boldsymbol{Y}$.

2.2.2 Energy Signal-to-Noise Ratio

In order to evaluate the quality of images in frequency domain, we defined a new measure, named Energy Signal-to-Noise Ratio (ESNR), in a similar way to defining PSNR. ESNR is defined as the ratio between the spectrum energy of ground truth image and the energy of the error spectrum, expressed in logarithmic decibel scale.

We can get the spectrum of the input image $\boldsymbol{X}$ and the ground truth image $\boldsymbol{Y}$ by Discrete Fourier Transform (DFT). Suppose that $\boldsymbol{S}_x$ and $\boldsymbol{S}_y$ are the spectrum of $\boldsymbol{X}$ and $\boldsymbol{Y}$, then $\boldsymbol{S}_x - \boldsymbol{S}_y$ is the error spectrum. The energy of error spectrum is

$$E_{\mathrm{error}} = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} |S_x(k,l) - S_y(k,l)|^2 \tag{2.17}$$

The energy of ground truth image spectrum is

$$E_{\mathrm{raw}} = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} |S_y(k,l)|^2 \tag{2.18}$$

Then ESNR can be expressed mathematically by

$$\mathrm{ESNR} = 10\log\left(\frac{E_{\mathrm{raw}}}{E_{\mathrm{error}}}\right) = 10\log\left(\frac{\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|S_y(k,l)|^2}{\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|S_x(k,l)-S_y(k,l)|^2}\right) \tag{2.19}$$

Compared with the PSNR, the main difference is that the ESNR expresses that ratio between the energy of the ground-truth image and the error signal energy, while the PSNR is defined by the ratio between the fixed maximum possible image energy and the error signal energy.

In the following, we will discuss the main properties of the ESNR in four aspects.

First of all, for the same raw image, when we use the ESNR and the PSNR to evaluate the input images with different quality, the change in the ESNR value is the same as the change in the PSNR value. This property is useful when we use the ESNR to evaluate the quality improvement of one input image over another input image. For example, the 1 dB gain in the PSNR and in the ESNR means the same degree of improvement in the image quality.

This can be proven mathematically. Let $\boldsymbol{Y}$ be the 8-bit-color raw image, and $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ be the two input images with different quality. The PSNR values for these two input images are

$$\begin{aligned}\mathrm{PSNR}_1 &= 10\log\left(\frac{255^2}{\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|X_1(m,n)-Y(m,n)|^2}\right)\\ \mathrm{PSNR}_2 &= 10\log\left(\frac{255^2}{\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|X_2(m,n)-Y(m,n)|^2}\right)\end{aligned} \tag{2.20}$$

Therefore, the change in the PSNR value is

$$\Delta\mathrm{PSNR} = \mathrm{PSNR}_2 - \mathrm{PSNR}_1 = 10\log\left(\frac{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|X_1(m,n)-Y(m,n)|^2}{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|X_2(m,n)-Y(m,n)|^2}\right) \tag{2.21}$$

The ESNR values for input images $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are

$$\begin{aligned}\mathrm{ESNR}_1 &= 10\log\left(\frac{\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|S_y(k,l)|^2}{\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|S_{x1}(k,l)-S_y(k,l)|^2}\right)\\ \mathrm{ESNR}_2 &= 10\log\left(\frac{\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|S_y(k,l)|^2}{\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|S_{x2}(k,l)-S_y(k,l)|^2}\right)\end{aligned} \tag{2.22}$$

The change in the ESNR value is

$$\Delta\mathrm{ESNR} = \mathrm{ESNR}_2 - \mathrm{ESNR}_1 = 10\log\left(\frac{\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|S_{x1}(k,l)-S_y(k,l)|^2}{\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|S_{x2}(k,l)-S_y(k,l)|^2}\right) \tag{2.23}$$

For two images $\boldsymbol{X}$, $\boldsymbol{Y}$ and their DFT $\boldsymbol{S}_x$, $\boldsymbol{S}_y$, due to the linearity property of the DFT, $\boldsymbol{S}_x - \boldsymbol{S}_y$ is the DFT of $\boldsymbol{X} - \boldsymbol{Y}$. According to Parseval's theorem, it holds

$$\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|X(m,n)-Y(m,n)|^2 = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|S_x(k,l)-S_y(k,l)|^2 \tag{2.24}$$

Therefore, we can write (2.21) as

$$\Delta\text{PSNR} = 10\log\left(\frac{\frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|S_{x1}(k,l)-S_y(k,l)|^2}{\frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|S_{x2}(k,l)-S_y(k,l)|^2}\right) \tag{2.25}$$

The $\frac{1}{MN}$ in both the numerator and the denominator can be canceled, then we get the conclusion

$$\Delta\text{PSNR} = \Delta\text{ESNR} \tag{2.26}$$

Another property of the ESNR is that it guarantees that whenever the error energy is at the same level as the energy of the ground truth image, the ESNR value will be always 0 dB, while the PSNR value in this case still depends on the energy of the ground truth image, which means that it varies for different images. In fact, for 8-bit color images, the PSNR value can reach 0 dB only when the input image is all-0 and the ground truth image is all-255, or vice versa. We can test a special case where the error energy and raw image energy are at the same level by setting the input image into all-zero. The PSNR value and the ESNR value are then computed according to the definition. The testing results are shown in Table 2.2. These testing results clearly show that in this special case the PSNR value is non-zero and varies over different images, while the ESNR value is 0 dB for all tested images.

Table 2.2: Comparison of PSNR and ESNR on Set5. The input images are set to all-zero, and the PSNR and the ESNR value are computed between input images and raw images.

measure	baby	bird	butterfly	head	woman
PSNR	4.57 dB	8.51 dB	5.58 dB	8.65 dB	5.48 dB
ESNR	0 dB	0 dB	0 dB	0 dB	0 dB

Besides, the ESNR value can provide a clear indication about the relative magnitude of error spectrum energy with respect to raw spectrum energy. We can write (2.19) as

$$\frac{E_{\text{error}}}{E_{\text{raw}}} = 10^{-\frac{\text{ESNR}}{10}} \tag{2.27}$$

This equation shows that there is a direct mapping from the ESNR value to the ratio of error spectrum energy to raw spectrum energy. Given an ESNR value, we can easily estimate the relative magnitude of error energy. For example, if the ESNR is 3 dB, the error spectrum energy is around 1/2 of the raw spectrum energy. If the ESNR is 30 dB,

then the error energy is around 0.1% of the raw spectrum energy.

Moreover, the ESNR presents good flexibility in evaluating the image quality in frequency domain. For example, in order to evaluate the image quality within the upper half frequency, we can define a measure, named $\mathrm{ESNR}^{\mathrm{up}}$, which is the ratio of raw energy in upper half frequency $E^{\mathrm{up}}_{\mathrm{raw}}$ to error energy in upper half frequency $E^{\mathrm{up}}_{\mathrm{error}}$, expressed in logarithmic decibel scale:

$$\mathrm{ESNR}^{\mathrm{up}} = 10\log\left(\frac{E^{\mathrm{up}}_{\mathrm{raw}}}{E^{\mathrm{up}}_{\mathrm{error}}}\right) = 10\log\left(\frac{\sum_{k,l\in[\frac{\pi}{2},\pi]}|S_y(k,l)|^2}{\sum_{k,l\in[\frac{\pi}{2},\pi]}|S_x(k,l)-S_y(k,l)|^2}\right) \tag{2.28}$$

where $\boldsymbol{S}_x$ and $\boldsymbol{S}_y$ are the DFT results of the input image $\boldsymbol{X}$ and ground truth image $\boldsymbol{Y}$. Similarly, we can define $\mathrm{ESNR}^{\mathrm{low}}$, which is the ratio of raw energy in lower half frequency $E^{\mathrm{low}}_{\mathrm{raw}}$ to error energy in lower half frequency $E^{\mathrm{low}}_{\mathrm{error}}$, expressed in logarithmic decibel scale:

$$\mathrm{ESNR}^{\mathrm{low}} = 10\log\left(\frac{E^{\mathrm{low}}_{\mathrm{raw}}}{E^{\mathrm{low}}_{\mathrm{error}}}\right) = 10\log\left(\frac{\sum_{k,l\in[0,\frac{\pi}{2}]}|S_y(k,l)|^2}{\sum_{k,l\in[0,\frac{\pi}{2}]}|S_x(k,l)-S_y(k,l)|^2}\right) \tag{2.29}$$

These two measures can be used the evaluate the quality of images in upper half frequency and lower half frequency range. We can also define the ESNR within arbitrary frequency range $[a,b]$:

$$\mathrm{ESNR}^{[a,b]} = 10\log\left(\frac{\sum_{k,l\in[a,b]}|S_y(k,l)|^2}{\sum_{k,l\in[a,b]}|S_x(k,l)-S_y(k,l)|^2}\right) \tag{2.30}$$

This measure can effectively evaluate the image quality within arbitrary frequency range.

After introducing the main properties and concepts of the ESNR, in the following, we will discuss two ESNR-related measures which are very useful to help us to analyze the image performance in frequency domain. These two measures are named as weight coefficient and contribution coefficient.

Weight Coefficient

When using $\mathrm{ESNR}^{\mathrm{low}}$ and $\mathrm{ESNR}^{\mathrm{up}}$ to analyze the quality of the image, it is important to predict either the change of the $\mathrm{ESNR}^{\mathrm{up}}$ or the change of the $\mathrm{ESNR}^{\mathrm{low}}$ has a larger contribution to the change of the overall ESNR value. For example, suppose that there are two cases. In one case the $\mathrm{ESNR}^{\mathrm{low}}$ is increased by 1 dB, and in another case the $\mathrm{ESNR}^{\mathrm{up}}$ is increased by 1 dB. We need to know in which case the overall ESNR value has a larger gain.

To figure out this, we can first deduce the analytical relationship between $\mathrm{ESNR}^{\mathrm{low}}$, $\mathrm{ESNR}^{\mathrm{up}}$ and the overall ESNR.
First we have the energy relationship between the overall energy and the energy in the lower half and the upper half frequency.

$$\begin{aligned} E_{\mathrm{raw}} &= E^{\mathrm{up}}_{\mathrm{raw}} + E^{\mathrm{low}}_{\mathrm{raw}} \\ E_{\mathrm{error}} &= E^{\mathrm{up}}_{\mathrm{error}} + E^{\mathrm{low}}_{\mathrm{error}} \end{aligned} \tag{2.31}$$

Let α be the ratio of the raw image energy in the upper half frequency $E_{\text{raw}}^{\text{up}}$ to the raw image energy in the overall frequency E_{raw}, then we have

$$\alpha = \frac{E_{\text{raw}}^{\text{up}}}{E_{\text{raw}}} , \quad 1-\alpha = \frac{E_{\text{raw}}^{\text{low}}}{E_{\text{raw}}} \tag{2.32}$$

It holds that

$$\frac{E_{\text{error}}}{E_{\text{raw}}} = \alpha \cdot \frac{E_{\text{error}}^{\text{up}}}{E_{\text{raw}}^{\text{up}}} + (1-\alpha) \cdot \frac{E_{\text{error}}^{\text{low}}}{E_{\text{raw}}^{\text{low}}} \tag{2.33}$$

According to (2.19), (2.29) and (2.28), we can use ESNR values to express (2.33):

$$10^{-\frac{\text{ESNR}}{10}} = \alpha \cdot 10^{-\frac{\text{ESNR}^{\text{up}}}{10}} + (1-\alpha) \cdot 10^{-\frac{\text{ESNR}^{\text{low}}}{10}} \tag{2.34}$$

We can then get the relationship between the ESNR and the ESNR^{up} and the ESNR^{low}.

$$\text{ESNR} = -10\log(\alpha \cdot 10^{-\frac{\text{ESNR}^{\text{up}}}{10}} + (1-\alpha) \cdot 10^{-\frac{\text{ESNR}^{\text{low}}}{10}}) \tag{2.35}$$

Therefore, the ESNR can be regarded as a function of the ESNR^{up} and the ESNR^{low}, with a parameter α only dependent on the raw image.

In order to evaluate the change of which value, the ESNR^{up} or the ESNR^{low} has a larger influence to the value of the overall ESNR, we can analyze the total derivative of the ESNR:

$$\text{d ESNR} = \frac{\partial \text{ESNR}}{\partial \text{ESNR}^{\text{up}}} \cdot \text{d ESNR}^{\text{up}} + \frac{\partial \text{ESNR}}{\partial \text{ESNR}^{\text{low}}} \cdot \text{d ESNR}^{\text{low}} \tag{2.36}$$

Then we compute the partial derivative of the ESNR with respect to the ESNR^{up} according to the chain rule:

$$\begin{aligned} \frac{\partial \text{ESNR}}{\partial \text{ESNR}^{\text{up}}} &= -\frac{10}{\ln(10)} \cdot \frac{\alpha \cdot 10^{-\frac{\text{ESNR}^{\text{up}}}{10}} \cdot \ln(10) \cdot \frac{-1}{10}}{\alpha \cdot 10^{-\frac{\text{ESNR}^{\text{up}}}{10}} + (1-\alpha) \cdot 10^{-\frac{\text{ESNR}^{\text{low}}}{10}}} \\ &= \frac{\alpha \cdot 10^{-\frac{\text{ESNR}^{\text{up}}}{10}}}{\alpha \cdot 10^{-\frac{\text{ESNR}^{\text{up}}}{10}} + (1-\alpha) \cdot 10^{-\frac{\text{ESNR}^{\text{low}}}{10}}} \end{aligned} \tag{2.37}$$

Substitute (2.28), (2.29) and (2.32) into (2.37), we can get a more compact expression of this partial derivative:

$$\begin{aligned} \frac{\partial \text{ESNR}}{\partial \text{ESNR}^{\text{up}}} &= \frac{\frac{E_{\text{raw}}^{\text{up}}}{E_{\text{raw}}} \cdot \frac{E_{\text{error}}^{\text{up}}}{E_{\text{raw}}^{\text{up}}}}{\frac{E_{\text{raw}}^{\text{up}}}{E_{\text{raw}}} \cdot \frac{E_{\text{error}}^{\text{up}}}{E_{\text{raw}}^{\text{up}}} + \frac{E_{\text{raw}}^{\text{low}}}{E_{\text{raw}}} \cdot \frac{E_{\text{error}}^{\text{low}}}{E_{\text{raw}}^{\text{low}}}} \\ &= \frac{E_{\text{error}}^{\text{up}}}{E_{\text{error}}^{\text{up}} + E_{\text{error}}^{\text{low}}} \\ &= \frac{E_{\text{error}}^{\text{up}}}{E_{\text{error}}} \end{aligned} \tag{2.38}$$

Similarly, the partial derivative of the ESNR with respect to the ESNR^{low} is

$$\frac{\partial \text{ESNR}}{\partial \text{ESNR}^{\text{low}}} = \frac{E_{\text{error}}^{\text{low}}}{E_{\text{error}}} \tag{2.39}$$

By substituting (2.38) and (2.39) into (2.36), we can get the expression of the total derivative of the ESNR:

$$\mathrm{d\ ESNR} = \frac{E_{\mathrm{error}}^{\mathrm{up}}}{E_{\mathrm{error}}} \cdot \mathrm{d\ ESNR}^{\mathrm{up}} + \frac{E_{\mathrm{error}}^{\mathrm{low}}}{E_{\mathrm{error}}} \cdot \mathrm{d\ ESNR}^{\mathrm{low}} \tag{2.40}$$

Let

$$w^{\mathrm{u}} = \frac{E_{\mathrm{error}}^{\mathrm{up}}}{E_{\mathrm{error}}}\ , \quad 1 - w^{\mathrm{u}} = \frac{E_{\mathrm{error}}^{\mathrm{low}}}{E_{\mathrm{error}}} \tag{2.41}$$

then the total derivative of the ESNR can be expressed by

$$\mathrm{d\ ESNR} = w^{\mathrm{u}} \cdot \mathrm{d\ ESNR}^{\mathrm{up}} + (1 - w^{\mathrm{u}}) \cdot \mathrm{d\ ESNR}^{\mathrm{low}} \tag{2.42}$$

We define the factor w^{u} as the weight coefficient of the upper half frequency. It is the ratio of the error energy in the upper half frequency to the overall energy. For simplicity we call w^{u} as upper-half weight coefficient, and in this case $1-w^{\mathrm{u}}$ is the lower-half weight coefficient.

The upper-half weight coefficient indicates the importance of the influence of the quality change in the upper half frequency on the overall quality change. For example, let upper-half weight coefficient w^{u} be 0.9 and the lower-half weight coefficient be 0.1. In this case, if the $\mathrm{ESNR}^{\mathrm{up}}$ is increased by 1 dB, the overall ESNR will increase by 0.9 dB. On the contrary, the overall ESNR only increases by 0.1 dB if the $\mathrm{ESNR}^{\mathrm{low}}$ is increased by 1 dB. This implies that the improvement in the $\mathrm{ESNR}^{\mathrm{up}}$ has a higher influence factor on the change in the overall ESNR.

Moreover, according to the definition, the weight coefficient in the upper half frequency is the ratio of the error energy in the upper half frequency to the overall error energy. Therefore, there is a direct connection between the importance of performance change in a frequency range and the ratio of the error energy within that frequency range. If the ratio of the error energy is higher within a frequency range, then the performance change within that frequency range has larger influence on the overall performance change.

Therefore, the weight coefficient can provide us with guidance when we want to further improve the quality of the input image. We should put more efforts into improving the performance in the frequency range where the weight coefficient is high. In other words, we should try to reduce the error energy in the frequency range where the ratio of error energy is high. This conclusion fits well with our intuition.

In order to validate the weight coefficient, we test two SISR methods. One method is the bicubic method. We use bicubic down-sampling to down-sample the raw image and get the low-resolution image. Then we use bicubic interpolation to up-sample the low-resolution image to get the up-sampled image. Another method is the SHVC method. First SHVC down-sampling is used to down-sample the raw image. Then we use SHVC interpolation to up-sample the low-resolution image to get the up-sampled image.

In this test, the bicubic method and the SHVC method are used to first down-sampled then interpolate the image 'head.bmp' in Set5. We can compare the PSNR, the ESNR values and the upper-half weight coefficients between the bicubic interpolated image and the

Table 2.3: Comparison of the PSNR, the ESNR and the upper-half weight coefficient between the bicubic interpolated image and the SHVC interpolated image. Tested on the 'head.bmp' in Set5.

Method	PSNR	ESNR	ESNRlow	ESNRup	w^{u}
Bicubic	34.86 dB	26.20 dB	32.98 dB	2.31 dB	0.7906
SHVC	35.16 dB	26.50 dB	35.19 dB	2.23 dB	0.8650

SHVC interpolated image. As shown in Table 2.3, the SHVC interpolation method achieves a higher PSNR. We can see that for the bicubic interpolated image, the w^{u} is 0.7906, so the change in the $\mathrm{ESNR}^{\mathrm{up}}$ has a larger influence on the change in the overall ESNR. The w^{u} for the SHVC interpolated image is 0.8650, which means that the performance change in the upper half frequency has a dominant influence on the overall performance change. These results imply that when we design a new SISR model to further improve the overall performance of the up-sampled images, we should put more effort into improving the performance in the upper half frequency, instead of the lower half frequency.

However, a large w^{u} does not necessarily guarantee that it is easy to improve the image quality in the upper half frequency. Especially in the SISR problems, since most of the information in the upper half frequency is lost during the down-sampling process, it is very difficult to recover the information in the upper half frequency, even using deep-learning based models. In this case, when we design a deep-learning based model and achieve some gain in the PSNR, this gain may not mainly come from the upper half frequency as expected. Therefore, we need another measure to analyze the performance gain in frequency domain.

Contribution Coefficient

The measure contribution coefficient can be used for comparing the performance of two SISR models. For example, if we want to compare the performance between the SRCNN model and the VDSR model, we can use these two models to up-sample the low-resolution image separately, and get two up-sampled images with different qualities. Then the contribution coefficient can be used to analyze from which frequency range the performance gain of the image with higher quality comes.

First of all we will give the definition and deduction of this measure. Although (2.42) expresses the relationship between differentials which are vanishingly small, we can still use this equation to approximate the relationship between the small change in the ESNR and the small changes in the $\mathrm{ESNR}^{\mathrm{up}}$ and the $\mathrm{ESNR}^{\mathrm{low}}$.

For example, let $\boldsymbol{Y}$ be the raw image and $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ be the two up-sampled images with different qualities, evaluated by ESNR_1 to ESNR_2. Suppose that the change in the $\mathrm{ESNR}^{\mathrm{up}}$

and the $\mathrm{ESNR}^{\mathrm{low}}$ are

$$\begin{aligned}\Delta\mathrm{ESNR}^{\mathrm{up}} &= \mathrm{ESNR}_2^{\mathrm{up}} - \mathrm{ESNR}_1^{\mathrm{up}} \\ \Delta\mathrm{ESNR}^{\mathrm{low}} &= \mathrm{ESNR}_2^{\mathrm{low}} - \mathrm{ESNR}_1^{\mathrm{low}}\end{aligned} \tag{2.43}$$

We use w_1^{u} and w_2^{u} to express the upper-half weight coefficient for $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$. In (2.38) we know that what w^{u} represents is the partial derivative of the ESNR with respect to the $\mathrm{ESNR}^{\mathrm{up}}$, so the value of w^{u} is not a constant in the period from ESNR_1 to ESNR_2. However, we can use $\bar{w^{\mathrm{u}}} = \frac{w_1^{\mathrm{u}} + w_2^{\mathrm{u}}}{2}$ to approximate the equivalent effective value of the w^{u} within this period. Then the change in the overall ESNR value can be estimated by

$$\Delta\mathrm{ESNR}_{\mathrm{e}} = \bar{w^{\mathrm{u}}} \cdot \Delta\mathrm{ESNR}^{\mathrm{up}} + (1 - \bar{w^{\mathrm{u}}}) \cdot \Delta\mathrm{ESNR}^{\mathrm{low}} \tag{2.44}$$

Equation 2.44 shows that the overall change in the ESNR can be estimated by the weighted sum of the change in $\mathrm{ESNR}^{\mathrm{up}}$ and $\mathrm{ESNR}^{\mathrm{low}}$. For example, if there is no performance change in the lower half frequency ($\Delta\ \mathrm{ESNR}^{\mathrm{low}} = 0$), then the change in the overall ESNR completely depends on the change in the $\mathrm{ESNR}^{\mathrm{up}}$:

$$\Delta\mathrm{ESNR}_{\mathrm{e}} = \bar{w^{\mathrm{u}}} \cdot \Delta\mathrm{ESNR}^{\mathrm{up}} \tag{2.45}$$

Therefore, we can define the term

$$C^{\mathrm{u}} = \bar{w^{\mathrm{u}}} \cdot \Delta\mathrm{ESNR}^{\mathrm{up}} \tag{2.46}$$

as the contribution coefficient in the upper-half frequency. It represents the part of the change in the overall ESNR value caused by the change in the $\mathrm{ESNR}^{\mathrm{up}}$. Note that the unit for C^{u} is dB. For short, we call C^{u} upper-half contribution coefficient. Similarly, we can define the lower-half contribution coefficient C^{ℓ}:

$$C^{\ell} = (1 - \bar{w^{\mathrm{u}}}) \cdot \Delta\mathrm{ESNR}^{\mathrm{low}} \tag{2.47}$$

C^{ℓ} represents the part of the ESNR change caused by the performance change in the lower half frequency. The overall estimated ESNR change is the sum of the C^{u} and C^{ℓ}:

$$\Delta\mathrm{ESNR}_{\mathrm{e}} = C^{\mathrm{u}} + C^{\ell} \tag{2.48}$$

For example, suppose that $C^{\mathrm{u}} = 0.9$ dB and $C^{\ell} = 0.1$ dB, then this means that the performance change in upper half frequency causes a 0.9 dB increase in the overall ESNR, and the performance change in lower half frequency causes a 0.1 dB increase in the overall ESNR. Therefore, the overall ESNR is increased by 1 dB. Note that C^{u} could be negative, and this means that the performance change in the upper-half frequency has a negative contribution to the over performance change. Similarly, C^{ℓ} could also be negative.

We can use an example to show the usage of the contribution coefficient C^{u} and C^{ℓ}. We would like to compare the performance of bicubic method and SHVC method, and analyze from which frequency range, either the lower half frequency or the upper half frequency, the performance gain of the SHVC method against bicubic method comes. From Table 2.3, we can compute the change in the $\mathrm{ESNR}^{\mathrm{up}}$, the $\mathrm{ESNR}^{\mathrm{low}}$ and the overall ESNR. The average weight coefficient of these two methods are $\bar{w_u} = \frac{w_{\mathrm{Bicubic}}^{\mathrm{u}} + w_{\mathrm{SHVC}}^{\mathrm{u}}}{2} = 0.8278$. Then we can compute the estimate change in the ESNR according to (2.44) and compare with the actual

Table 2.4: The change in the ESNR, the ESNR^{up}, the ESNR^{low} and the C^{u} from the bicubic interpolated image to the SHVC interpolated image. Tested on the 'head.bmp' in Set5.

Method	ΔPSNR	ΔESNR	$\Delta\text{ESNR}^{\text{low}}$	$\Delta\text{ESNR}^{\text{up}}$	$\bar{w}^{\text{u}}$	C^{ℓ}	C^{u}	$\Delta\text{ESNR}_{\text{e}}$
Bicubic→SHVC	0.30 dB	0.30 dB	2.21 dB	-0.08 dB	0.8278	0.33 dB	-0.02 dB	0.31 dB

change in the ESNR. Finally, we can compute the upper-half and lower-half contribution coefficients according to 2.46 and 2.47 and use them to evaluate the contribution of the performance gain in the upper half frequency and the lower half frequency. The testing results are shown in Table 2.4.

As shown in Table 2.4, since the SHVC method has a smaller ESNR^{up} than the bicubic method, the C^{u} has a negative value -0.02 dB, which implies that the performance loss of the SHVC in the upper half frequency will cause a 0.02 dB drop in the overall ESNR. On the contrary, the contribution coefficient for the lower half frequency is 0.33 dB, which implies that the performance gain in the lower half frequency causes a 0.33 dB increase in the overall ESNR. Therefore, the estimated change in the ESNR value is 0.31 dB. There is only 0.01 dB difference between the estimated change and the actual change in the ESNR value, which validates the effectiveness of the approximation in (2.44). This fits well with our knowledge, since the SHVC method cannot recover information in the upper half frequency in theory.

2.2.3 Rectangular Ring Spectrum

In this section we give the definition of rectangular ring spectrum and show how to use this tool to express the ESNR spectrum, weight spectrum and contribution spectrum which represent the ESNR value, the weight coefficient and the contribution coefficient over frequency respectively.

We use a rectangular-shape spectrum instead of a circular shape spectrum, since in testing we find that for an ideally interpolated image which has the same contents as the raw image in the lower half frequency and no information in the upper half frequency, its spectrum is a rectangular shape with half the size of the raw image spectrum. The division of the spectrum into rectangular ring shape is shown in Figure 2.6. For an image $\boldsymbol{X}$, we can first apply DFT to get its spectrum $\boldsymbol{S}_x$. Then we can divide the spectrum into L frequency bands, and each band is a rectangular ring with a bandwidth $\frac{\pi}{L}$. This definition is similar to the definition of ring spectra in [13], and the only difference is that here each ring has a rectangular shape. Note that in this thesis, the default value of L is 40, which means the bandwidth of a rectangular ring is $\frac{\pi}{40}$.

In the following, we will use the definition of the rectangular ring spectrum to introduce the concept of the ESNR spectrum, weight spectrum and the contribution spectrum.

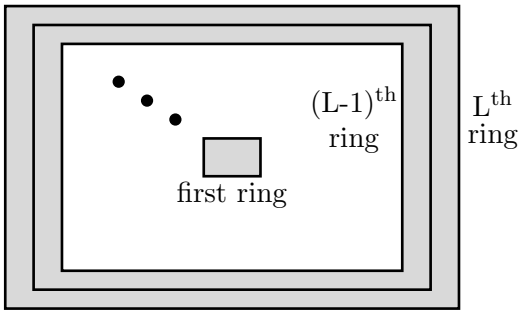


Figure 2.6: Division of rings in the rectangular ring spectrum.

ESNR Spectrum

The energy of the i^{th} ring can be expressed by

$$E^i = \sum_{k,l \in i^{\text{th}}\text{ring}} |S_x(k,l)|^2 \tag{2.49}$$

E^i represents the energy within frequency range $[\frac{i-1}{L}\pi,\ \frac{i}{L}\pi]$, and we call E^i the rectangular ring energy spectrum.

The rectangular ring energy spectrum can be used to express the ESNR spectrum. Let $\boldsymbol{S}_x$ and $\boldsymbol{S}_y$ be the spectrum of the input image and the ground truth (raw) image respectively, then the $\boldsymbol{S}_x - \boldsymbol{S}_y$ represents the error spectrum. According to (2.49), We can get the rectangular ring energy spectrum of the raw image and the error image:

$$\begin{aligned} E^i_{\text{raw}} &= \sum_{k,l \in i^{\text{th}}\text{ring}} |S_y(k,l)|^2 \\ E^i_{\text{error}} &= \sum_{k,l \in i^{\text{th}}\text{ring}} |S_x(k,l) - S_y(k,l)|^2 \end{aligned} \tag{2.50}$$

According to definition of the ESNR, we can define the ESNR spectrum.

$$\text{ESNR}^i = 10\log\left(\frac{E^i_{\text{raw}}}{E^i_{\text{error}}}\right) \tag{2.51}$$

ESNR^i expresses the ESNR value over frequency range $[\frac{i-1}{L}\pi,\ \frac{i}{L}\pi]$.

Weight Spectrum

In frequency analysis, it is useful to know how much the performance change in the ESNR^i within frequency range $[\frac{i-1}{L}\pi, \frac{i}{L}\pi]$ influences the overall performance change in the ESNR value. In order to analyze this, we can further extend the definition of upper-half weight coefficient w^{u} in Section 2.2.2 by introducing the concept of the weight spectrum, denoted as w^i.

The weight spectrum w^i is defined as the ratio of the error energy E^i_{error} within frequency range $[\frac{i-1}{L}\pi, \frac{i}{L}\pi]$ to the error energy E_{error} of the overall frequency:

$$w^i = \frac{E^i_{\text{error}}}{E_{\text{error}}} \tag{2.52}$$

The total derivative of the ESNR value can be expressed by

$$\mathrm{d\ ESNR} = \sum_{i=1}^{L} w^i \cdot \mathrm{d\ ESNR}^i \tag{2.53}$$

We can deduce this equation in a similar way to the procedure in Section 2.2.2. First of all, we have the equation

$$\frac{E_{\mathrm{error}}}{E_{\mathrm{raw}}} = \sum_{i=1}^{L} \frac{E^i_{\mathrm{error}}}{E_{\mathrm{raw}}} \tag{2.54}$$

Let α_i be the ratio of the raw image energy in frequency range $[\frac{i-1}{L}\pi, \frac{i}{L}\pi]$ to the overall raw image energy

$$\alpha_i = \frac{E^i_{\mathrm{raw}}}{E_{\mathrm{raw}}} \tag{2.55}$$

then (2.54) can be written as

$$\frac{E_{\mathrm{error}}}{E_{\mathrm{raw}}} = \sum_{i=1}^{L} \alpha_i \cdot \frac{E^i_{\mathrm{error}}}{E^i_{\mathrm{raw}}} \tag{2.56}$$

Substituting (2.27) and (2.51) into (2.56), we can get the relationship between the ESNR and the ESNR^i:

$$\mathrm{ESNR} = -10\log(\sum_{i=1}^{L} \alpha_i \cdot 10^{-\frac{\mathrm{ESNR}^i}{10}}) \tag{2.57}$$

Therefore, the total derivative of the ESNR can written as

$$\mathrm{d\ ESNR} = \sum_{i=1}^{L} \frac{\partial \mathrm{ESNR}}{\partial \mathrm{ESNR}^i} \cdot \mathrm{d\ ESNR}^i \tag{2.58}$$

The partial derivative $\frac{\partial \mathrm{ESNR}}{\partial \mathrm{ESNR}^i}$ can be computed according to the chain rule:

$$\begin{aligned}
\frac{\partial \mathrm{ESNR}}{\partial \mathrm{ESNR}^i} &= -\frac{10}{\ln(10)} \cdot \frac{\alpha_i \cdot 10^{-\frac{\mathrm{ESNR}^i}{10}} \cdot \ln(10) \cdot \frac{-1}{10}}{\sum_{k=1}^{L} \alpha_k \cdot 10^{-\frac{\mathrm{ESNR}^k}{10}}} \\
&= \frac{\alpha_i \cdot 10^{-\frac{\mathrm{ESNR}^i}{10}}}{\sum_{k=1}^{L} \alpha_k \cdot 10^{-\frac{\mathrm{ESNR}^k}{10}}} \\
&= \frac{\frac{E^i_{\mathrm{raw}}}{E_{\mathrm{raw}}} \cdot \frac{E^i_{\mathrm{error}}}{E^i_{\mathrm{raw}}}}{\sum_{k=1}^{L} \frac{E^k_{\mathrm{raw}}}{E_{\mathrm{raw}}} \cdot \frac{E^k_{\mathrm{error}}}{E^k_{\mathrm{raw}}}} \\
&= \frac{E^i_{\mathrm{error}}}{\sum_{k=1}^{L} E^k_{\mathrm{error}}} \\
&= \frac{E^i_{\mathrm{error}}}{E_{\mathrm{error}}}
\end{aligned} \tag{2.59}$$

Therefore, we can write (2.58) as

$$\mathrm{d\ ESNR} = \sum_{i=1}^{L} \frac{E_{\mathrm{error}}^{i}}{E_{\mathrm{error}}} \cdot \mathrm{d\ ESNR}^{i} \tag{2.60}$$

When using w^i to represent $\frac{E_{\mathrm{error}}^{i}}{E_{\mathrm{error}}}$, we get the same expression as (2.53).

(2.60) shows that the weight spectrum w_i quantifies the influence of the change in the ESNR^i on the change of the ESNR, over different frequency ranges. The weight spectrum provides us with guidance when we are trying to further improve the quality of the input image. We should put more effort into the frequency range where the weight spectrum has a larger value, since the performance gain within that frequency range has the most obvious influence on the overall performance gain.

Moreover, according to the definition (2.52), the weight spectrum can be regarded as the error energy distribution over frequency. In order to improve the performance by a large margin, we should try to reduce the error energy where the error distribution is high, and we can get this information from the weight spectrum.

Note that the weight spectrum shows the distribution of error energy in percentage, but not the actual magnitude of the error energy, so we cannot use weight spectrum to compare the relative magnitude of the error energy of different models at certain frequency range.

Contribution Spectrum

When we comparing two up-sampled images, one of which has better quality than the other, we would like to know in detail what the percentage of the contribution in frequency range $[\frac{i-1}{L}\pi,\ \frac{i}{L}\pi]$ to the overall gain is. In this case, we can extend the definition of the contribution coefficient into the context of rectangular ring spectrum.

Suppose that we have two up-sampled image, $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, and the raw image $\boldsymbol{Y}$, we can compute the ESNR spectrum and the weight spectrum for $\boldsymbol{X}_1$ and $\boldsymbol{X}_1$, denoted as ESNR_1^i, ESNR_2^i, w_1^i and w_2^i. Then the difference of the ESNR spectrum is

$$\Delta\mathrm{ESNR}^i = \mathrm{ESNR}_2^i - \mathrm{ESNR}_1^i \tag{2.61}$$

Since the weight spectrum from ESNR_1^i to ESNR_2^i is not constant, we can use the average weight spectrum $\bar{w^i} = \frac{w_1^i + w_2^i}{2}$ to approximate the equivalent effective weight spectrum. According to (2.53), the change in the overall ESNR value $\Delta\mathrm{ESNR} = \mathrm{ESNR}_2 - \mathrm{ESNR}_1$ can be approximated by

$$\Delta\mathrm{ESNR}_{\mathrm{e}} = \sum_{i=1}^{L} \bar{w^i} \cdot \Delta\mathrm{ESNR}^i \tag{2.62}$$

Therefore, we can define the contribution spectrum as

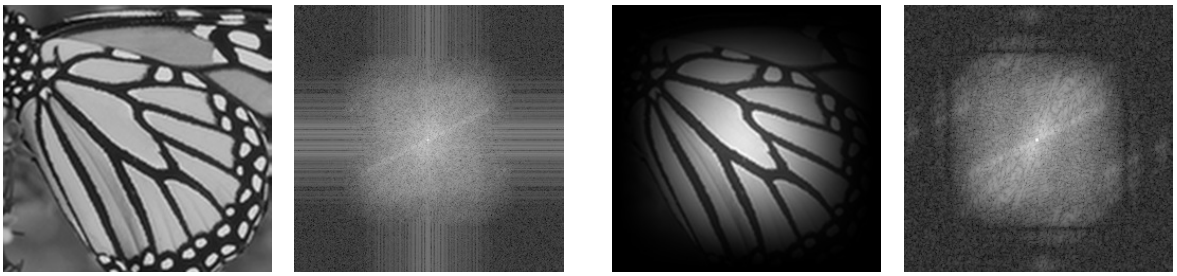
$$C^i = \bar{w^i} \cdot \Delta\mathrm{ESNR}^i \tag{2.63}$$

The contribution spectrum C^i represents the part of the change in ESNR value (dB) caused by the performance change within frequency range $[\frac{i-1}{L}\pi, \frac{i}{L}\pi]$. By aggregating the C^i over frequency, we will get the estimated change in the overall ESNR value.

2.2.4 Windowing

In frequency analysis, we can evaluate the quality by visualizing the spectrum of an image. A typical procedure of visualization is first applying DFT to the image to get the spectrum, then shift the spectrum to move the low resolution components to the center and high resolution components to the border, and finally convert the magnitude of the spectrum into logarithmic scale by applying a logarithmic function.

When analyzing the spectrum, we notice that there are high frequency components which should not exist in theory. For example, we can use SHVC tools to down-sample an image $\boldsymbol{X}$ into half size, then up-sample it into the original size via SHVC interpolation. Theoretically, the SHVC-interpolated image keeps the most information in the lower half frequency and almost all-zero in upper half frequency. However, in the experiment we find that there are many high frequency components in the spectrum. As is shown in Figure 2.7a, when directly visualizing the spectrum of the SHVC-interpolated image, we can see that there are many 'stripes' in the upper half frequency. This is because that DFT will assume that the image is periodic. After periodically extended the image, there will be discontinuity at the border, and this discontinuity will bring some extra noise in the frequency domain. We call this effect as boundary discontinuity effect.



(a) No windowing applied. (b) Windowing applied.

Figure 2.7: Spectrum of the SHVC-interpolated image when applying windowing or not, tested on the 'butterfly' image in Set5.

One effective way to remove the boundary discontinuity effect is windowing. In this thesis, we weighted the data by the multiplication of a hanning window [14], then applied DFT to the weighted image and visualized the spectrum. Due to the property of hanning window, the boundary pixels of the weighted image are zero, so there is no discontinuity at the boundary after periodically extend the image. Therefore, the boundary discontinuity effect will be mitigated in this way.

Figure 2.7b shows the spectrum of the SHVC-interpolated image, multiplied by the hanning window. We can see that there is less distortion in the spectrum, compared with the image

without windowing. However, one side effect of applying the hanning window is that there exist side lobes in frequency domain, as shown in Figure 2.7b we can notice that there are small amount of spectrum components in the upper half frequency.

Chapter 3

Performance Evaluation of Current SISR Models

In this chapter, we will evaluate the performance of current SISR methods in frequency domain. The performance evaluation mainly includes four aspects. First, the testing results of different models and their weight spectrum are provided and analyzed. Then we will compare the performance between interpolation methods and deep-learning based methods to check where the gain of deep-learning based methods comes in frequency domain. Next the influence of the network depth on the performance will be analyzed by comparing the performance of the 5-layer CNN model and the VDSR model. Afterwards, we try to show the influence of different down-sampling and interpolation methods on the training and performance of deep-learning based networks.
Finally, we will briefly discuss the motivation of designing a new deep-learning based SISR model for better performance.

3.1 Performance of SISR Models

In this section, we first provide the testing results of different SISR models in terms of the PSNR, the ESNR, the ESNR^{up}, the ESNR^{up}, and the upper-half weight coefficient w^{u}.

The SISR methods under evaluation mainly include two parts. One part are the traditional interpolation methods, including bicubic method, SHVC method, and the ideal method. The bicubic method means using bicubic down-sampling and interpolation to down-sample the raw image then up-sample the low-resolution image. The SHVC method means using SHVC down-sampling and interpolation to first down-sample the raw image then up-sample the down-sampled image. The ideal method means that by manipulating the spectrum we preserve the complete information in the lower half frequency while remove all components in the upper half frequency.
Another part are deep learning based models, including the SRCNN model, the VDSR model and the 5-layer CNN model. Here for the training of all these three deep-learning based SISR models, we use bicubic method to generate the training and testing data.

In Table 3.1 we notice that for bicubic and SHVC methods, the ESNR^{up} is not zero, but in theory this value should be 0 dB, since there should be little spectrum component in the upper half frequency of bicubic or SHVC interpolated images. The main reason for this phenomenon is the boundary discontinuity effect. For comparison, we first multiply the up-sampled image and the raw image with a hanning window, then compute the ESNR^{up} value. The results are shown in Table 3.2.

Table 3.1: Testing results of SISR models. The deep-learning models use bicubic interpolated images for training. All results are averaged over all images in Set5 and Set14 respectively.

Method	Set5					Set14				
	PSNR (dB)	ESNR (dB)	$ESNR^{low}$ (dB)	$ESNR^{up}$ (dB)	w^u	PSNR (dB)	ESNR (dB)	$ESNR^{low}$ (dB)	$ESNR^{up}$ (dB)	w^u
Bicubic	33.66	27.12	31.64	1.55	0.6339	30.23	24.11	29.33	0.72	0.6889
SHVC	34.39	27.84	34.04	1.53	0.7489	30.73	24.61	31.51	0.62	0.7863
SRCNN	36.66	30.11	39.11	3.12	0.8706	32.45	26.34	35.45	1.88	0.8732
5-layer CNN	37.05	30.50	39.77	3.47	0.8786	32.73	26.61	35.87	2.15	0.8764
VDSR	37.43	30.89	40.45	3.82	0.8871	32.95	26.83	36.27	2.34	0.8811
Ideal Interpolation	34.14	27.59	∞	0	1	31.16	25.04	∞	0	1

Table 3.2: Comparison of the $ESNR^{up}$ of bicubic and SHVC interpolated images, computed with and without windowing.

Method	Set5		Set14	
	no windowing	windowing	no windowing	windowing
Bicubic	1.55 dB	0.04 dB	0.72 dB	0.26 dB
SHVC	1.53 dB	0.04 dB	0.62 dB	0.12 dB

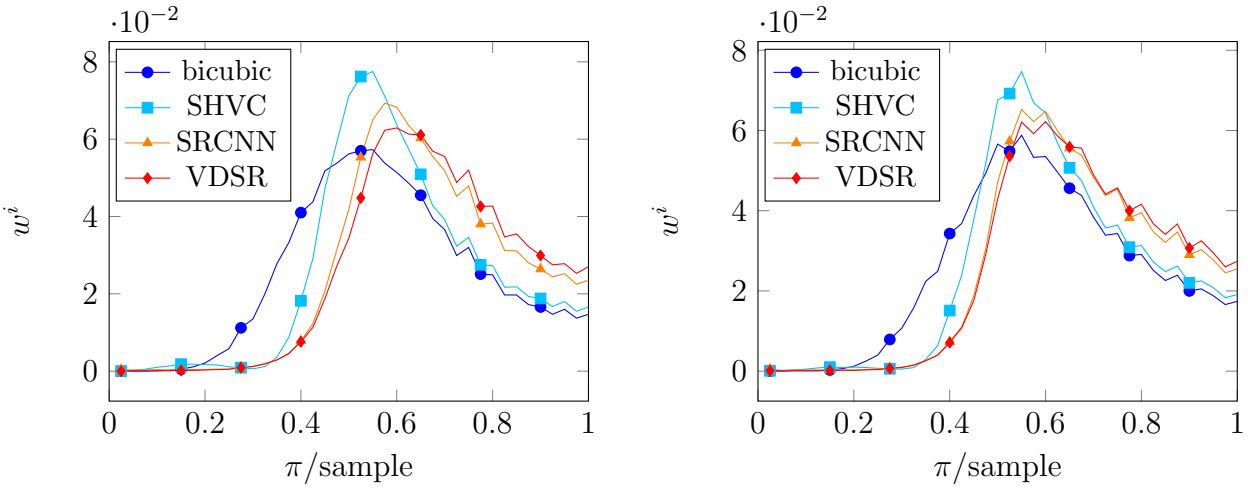
From Table 3.2 we can see that after applying a hanning window to remove the boundary discontinuity effect, the $ESNR^{up}$ value of both interpolation methods is much closer to 0 dB, which supports our assumption that boundary discontinuity effect is the main reason. Moreover, the frequency response of the filter for bicubic and SHVC interpolation has side lobes. This also brings some spectrum components in the upper half frequency.

From the results in Table 3.1, we can have a basic impression of different SISR models. First, compared with the interpolation methods such as the Bicubic and SHVC, the deep-learning based methods achieve performance gain in both the lower half frequency and the upper half frequency. For deep-learning based methods, when we increase the network depth from 3 to 5 then to 20, there are gains in both the lower half frequency and the upper half frequency. However, for these comparisons we cannot clearly know the gain in which frequency range is more important. This will be analyzed in the following section.

The upper-half weight coefficient w^u in Table 3.1 also gives us useful information. Compared with interpolation methods, the deep-learning based methods has a larger w^u. Besides, when increasing the network depth, the value w^u also increase.
The increase in the value w^u represents the increase in the ratio of the error energy in the upper half frequency to the overall error energy. In other words, when using deep-learning based models and increasing the network depth, the error energy in the upper half frequency accounts for more percent of the overall error energy. For example, for the VDSR model the

w^{u} is around 0.88, which means the upper-half error energy accounts for 88% of the overall error energy.
In Section 2.2.2 we showed that the w^{u} quantifies the influence of the performance change in the upper-half frequency on the overall performance change. Therefore, due to a large w^{u} for the VDSR, if we want to further improve the performance over the VDSR model, we should find effective methods to improve the performance in the upper half frequency.

In order to analyze the performance of different SISR models in frequency domain in a more detailed way, we can compare the ESNR spectrum and the weight spectrum.



(a) Set5.

(b) Set14.

Figure 3.1: Comparison of weight spectrum of different SISR methods. The curves are averaged over all images in Set5 and Set14 respectively.

As is shown in Figure 3.1, the weight spectrum does not show the absolute magnitude of the error energy over different frequency, since the error energy has been normalized. Instead, they give us the information about the distribution of the error energy over frequency, so we can also regard it as the error energy distribution spectrum. Note that in fact this spectrum is actually a histogram of the error energy distribution, since according to the definition of rectangular ring spectrum, the minimum bandwidth is $\frac{\pi}{40}$.

Several conclusions can be drawn from the weight spectrum in Figure 3.1. First, the distributions of the error energy mainly center at the frequency range $[0.45\pi, 0.7\pi]$. This indicates that the information loss in the frequency range $[0.45\pi, 0.7\pi]$ is the main source of the information loss in SISR problems. Improving the performance within this frequency range can bring most obvious gain in the overall performance. Second, compared with interpolation methods such as the bicubic and the SHVC, the error distribution of the deep-learning based models moves towards higher frequency. Most error energy distributes in the upper half frequency. Therefore, when we want to design a new deep-learning model to further improve the performance, more attention should be paid to improving the performance in the upper half frequency.

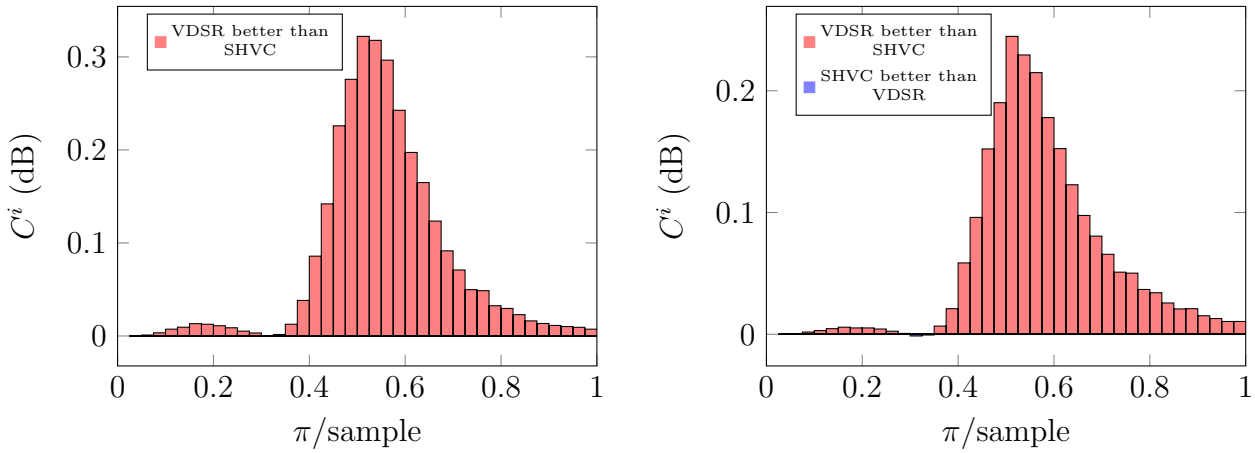
3.2 Comparison of Interpolation and Deep-Learning Models

After giving the testing testing results and analyzing the weight spectrum, in this section we compare the performance between interpolation methods and the deep-learning based methods. Our goal is to figure out from which frequency range the gain of the deep-learning based SISR models mainly comes, and how much the contribution to the overall gain is.

First we can compare the performance gain between the SHVC methods and the VDSR model. By using contribution coefficient, we can analyze where the gain of the VDSR against the SHVC comes from, either in the lower half frequency or the upper half frequency.

Table 3.3: Analysis of performance gain of the VDSR model against the SHVC method. All results are averaged over all images in Set5 and Set14 respectively.

Data	Methods	ΔPSNR	ΔESNR	$\Delta\text{ESNR}^{\text{low}}$	$\Delta\text{ESNR}^{\text{up}}$	$\bar{w}^{\text{u}}$	C^{ℓ}	C^{u}	$\Delta\text{ESNR}_{\text{e}}$
Set5	SHVC→VDSR	3.04 dB	3.05 dB	6.41 dB	2.29 dB	0.8180	1.17 dB	1.87 dB	3.04 dB
Set14	SHVC→VDSR	2.22 dB	2.22 dB	4.76 dB	1.72 dB	0.8337	0.79 dB	1.44 dB	2.23 dB



(a) Set5. (b) Set14.

Figure 3.2: Contribution spectrum of between the SHVC method and the VDSR model. The curves are averaged over all images in Set5 and Set14 respectively.

In Table 3.3, we notice that when tested on Set5 ΔPSNR ≠ ΔESNR. This is actually caused by round-off errors. Besides, comparing the value between the value of the estimated change and the actual change in the ESNR, we can find that there is only 0.1 dB difference between the ΔESNR and the $\Delta\text{ESNR}_{\text{e}}$. This further validates the approximation used in (2.44).

When tested on Set5, the performance gain of the VDSR in the upper half frequency accounts for a 1.87 dB gain in the overall ESNR, while the performance gain in the lower half frequency accounts for a 1.17 dB gain in the overall ESNR. This means 60% of the performance gain comes from the upper half frequency, while 40% of the performance gain comes from the lower half frequency. The similar relationship can be observed when tested on Set14.

We can further analyze the performance contribution over frequency by visualizing the contribution spectrum. As shown in Figure 3.2, compared with the SHVC methos, the performance gain of the VDSR model mainly comes from frequency range $[0.4\pi, 0.7\pi]$.

3.3 Influence of Network Depth

In this section we try to analyze the influence of the network depth on the performance of the deep-learning network. Two models are under test. One model is the VDSR model which has 20 layers and the other model is the 5-layer CNN model with 5 convolution layers.

We can first use contribution coefficient C^{u} to estimate the contribution of the gain in the lower-half frequency and the upper-half frequency to the overall performance gain.

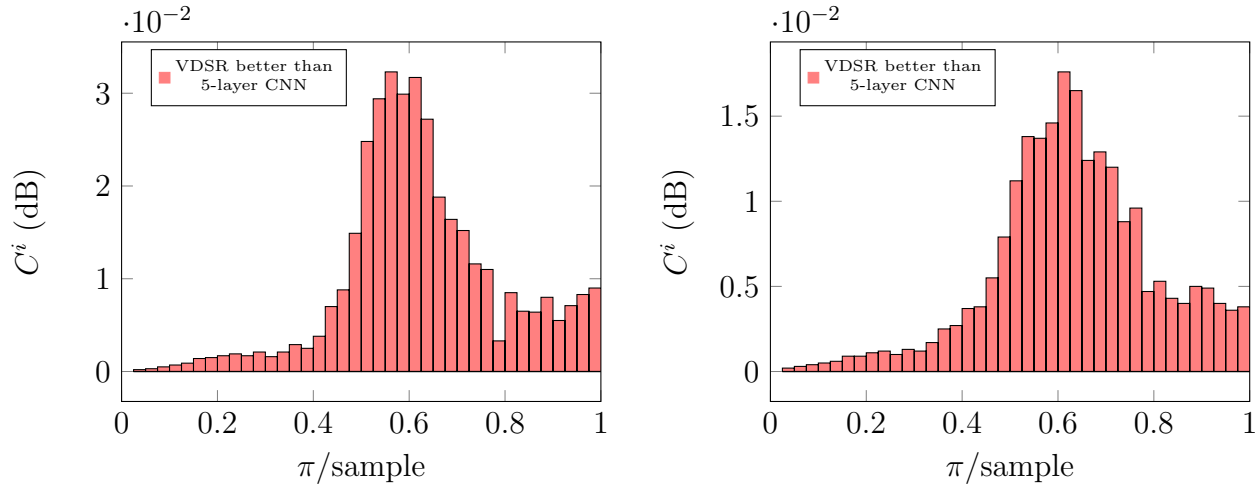
Table 3.4: Analysis of performance gain of the VDSR model against the 5-layer CNN. All results are averaged over all images in Set5 and Set14 respectively.

Data	Methods	ΔPSNR	ΔESNR	$\Delta\mathrm{ESNR}^{\mathrm{low}}$	$\Delta\mathrm{ESNR}^{\mathrm{up}}$	$\bar{w^{\mathrm{u}}}$	C^{ℓ}	C^{u}	$\Delta\mathrm{ESNR}_{\mathrm{e}}$
Set5	5-layer CNN→VDSR	0.38 dB	0.39 dB	0.68 dB	0.35 dB	0.8829	0.08 dB	0.31 dB	0.39 dB
Set14	5-layer CNN→VDSR	0.22 dB	0.22 dB	0.4 dB	0.19 dB	0.8788	0.05 dB	0.17 dB	0.22 dB

As shown in Figure 3.4, when tested on Set5, the upper-half contribution coefficient is 0.31 dB while the lower half contribution coefficient is only 0.08 dB. This means that when increasing the network depths from 5 layers to 20 layers, the performance gain in the upper half frequency accounts for around 80% of the overall ESNR gain. This conclusion also holds when tested on Set14.

We can plot the contribution spectrum to evaluate the contribution ratio over frequency. From Figure 3.3 we can see that the gain of the VDSR model mainly comes from the frequency range $[0.5\pi,0.7\pi]$. Besides, in the frequency range $[0.8\pi, \pi]$ a slight contribution to the overall gain can be observed.

Therefore, based on the analysis above, a deeper network can help to recover more information in the upper half frequency range.



(a) Set5.

(b) Set14.

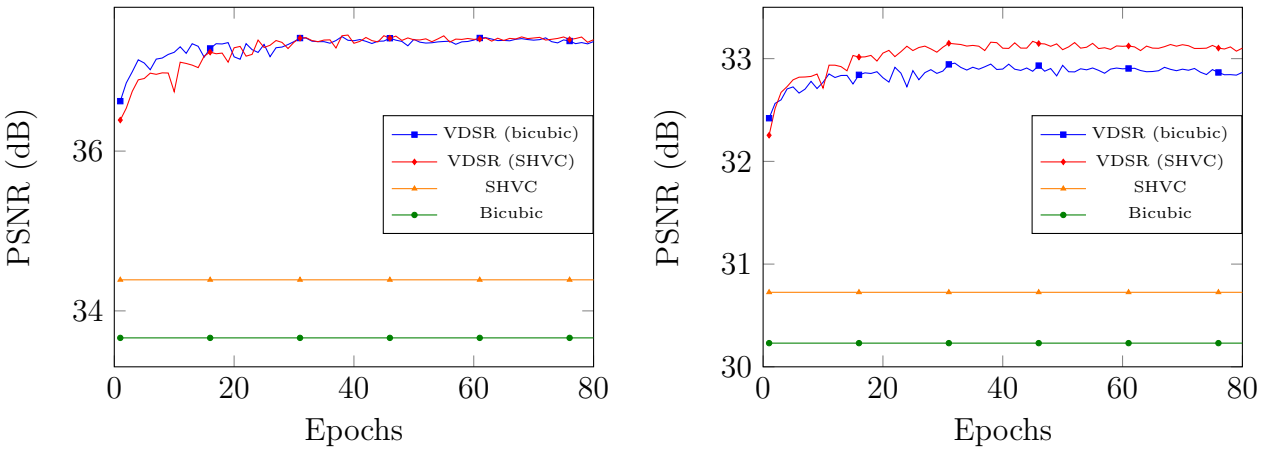
Figure 3.3: Contribution spectrum of between the 5-layer CNN and the VDSR model. The curves are averaged over all images in Set5 and Set14 respectively.

3.4 Influence of Down-sampling Methods

In current deep-learning based SR methods, such as SRCNN and VDSR, the common flow in training and testing is first using bicubic down-sampling to get the low-resolution images, then upscaling low-resolution images into input images with full size via the bicubic interpolation. Input images are then fed into the CNN models to get the high-resolution images. Therefore, bicubic down-sampling and interpolation methods are chosen as the default down-sampling and interpolation method. Note that the interpolation method is corresponding to the down-sampling method, so we use 'bicubic method' to represent the process of first bicubic down-sampling then bicubic interpolation.

In this section we will try to explain how different down-sampling methods influence the training and performance of the deep-learning based SISR model. The VDSR model is trained using bicubic interpolated images or SHVC-interpolated images as training data. For simplicity, we call the VDSR model using bicubic interpolated images for training as bicubic-based VDSR, and the VDSR model using SHVC interpolated images for training as SHVC-based VDSR.

First we can analyze the influence of down-sampling methods on the training of the deep-learning networks by comparing the training curve. The training curve is plotted based on the testing results on Set5 and Set14. The results are shown in Figure 3.4. From this figure we can notice that when using bicubic interpolated images as training data, it takes only 10 epochs for the VDSR model to start converging, while the VDSR model converges after 20 epochs when using SHVC-interpolated images for training. Therefore, when using SHVC interpolated images for training, it takes more epochs for the VDSR model to converge.



(a) VDSR, Set5.

(b) VDSR, Set14.

Figure 3.4: Performance curve comparison between bicubic and SHVC down-sampling, tested on VDSR model.

In order to analyze the influence of down-sampling methods on the performance of the deep-learning based models, we can compare the performance after convergence, and analyze where the performance gain of SHVC-based VDSR model comes from in frequency domain. The performances of the bicubic-based VDSR and the SHVC-based VDSR are shown in Table 3.5.

Table 3.5: Testing results of SISR models. The VDSR model uses bicubic interpolated images for training. All results are averaged over all images in Set5 and Set14 respectively.

Method	Set5					Set14				
	PSNR (dB)	ESNR (dB)	ESNR^{low} (dB)	ESNR^{up} (dB)	w^{u}	PSNR (dB)	ESNR (dB)	ESNR^{low} (dB)	ESNR^{up} (dB)	w^{u}
VDSR (Bicubic)	37.43	30.89	40.45	3.82	0.8871	32.95	26.83	36.27	2.34	0.8811
VDSR (SHVC)	37.45	30.91	42.33	3.66	0.9244	33.16	27.04	38.95	2.30	0.9322

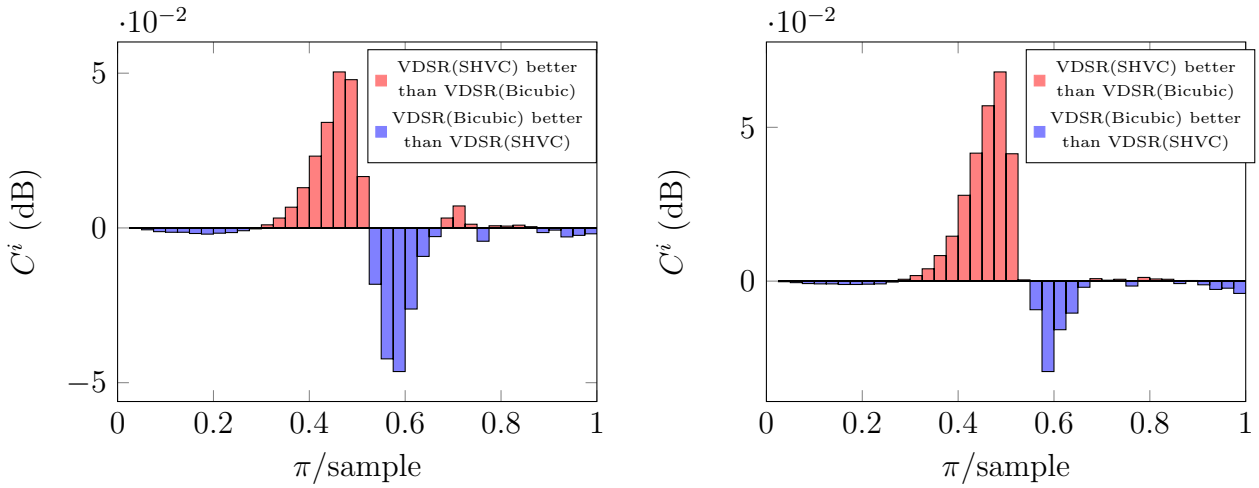
From Table 3.5 we can see that the SHVC-based VDSR model achieve a slightly higher PSNR. When comparing the ESNR^{up} and the ESNR^{low}, we can see that the SHVC-based VDSR model has a higher ESNR^{low} and a lower ESNR^{up}. This results imply that compared with the bicubic-based VDSR model, the SHVC-based VDSR model has better performance in the lower half frequency, while worse performance in the upper half frequency.

In the next, we will analyze how much the performance change in the lower half frequency and the upper half frequency contributes to the overall performance gain. From Table 3.6 we can notice that when tested on Set5, the C^{u} is -0.14 dB, which means that compared with the bicubic-based VDSR, the the change of performance of SHVC-based VDSR in the upper half frequency accounts for a 0.14 dB drop in the overall ESNR. The C^{u} almost completely cancels out the positive contribution in the lower half frequency. Therefore, there is only tiny increase in the overall ESNR. This means that the performance change in

Table 3.6: Analysis of performance gain of the SHVC-based VDSR against the bicubic-based VDSR. All results are averaged over all images in Set5 and Set14 respectively.

Data	Methods	ΔPSNR	ΔESNR	$\Delta\text{ESNR}^{\text{low}}$	$\Delta\text{ESNR}^{\text{up}}$	$\bar{w}^{\text{u}}$	C^{ℓ}	C^{u}	$\Delta\text{ESNR}_{\text{e}}$
Set5	VDSR(Bicubic)→ VDSR(SHVC)	0.02 dB	0.02 dB	1.88 dB	-0.16 dB	0.9058	0.17 dB	-0.14 dB	0.03 dB
Set14	VDSR(Bicubic)→ VDSR(SHVC)	0.21 dB	0.21 dB	2.68 dB	-0.04 dB	0.9067	0.25 dB	-0.04 dB	0.21 dB

the upper half frequency has a negative contribution to the overall performance gain.



(a) Set5. (b) Set14.

Figure 3.5: Contribution spectrum of between the SHVC-based VDSR and the bicubic-based VDSR model. The curves are averaged over all images in Set5 and Set14 respectively.

We can also use the contribution spectrum to see more clearly in which frequency range the contribution spectrum is negative. As shown in Figure 3.5, when analyzing the performance gain of the SHVC-based VDSR against the bicubic-based VDSR, we can notice that there are positive contributions to the overall gain in frequency range $[0.4\pi, 0.5\pi]$. However, there are comparable negative contributions to the overall gain in frequency range $[0.5\pi, 0.6\pi]$, and these negative contributions cancel out the positive contributions. Therefore, no obvious improvement of the SHVC-based VDSR in the overall performance is observed.

These analysis results show that compared with the bicubic-based VDSR, the SHVC-based VDSR has better performance in the lower half frequency, but worse performance in the upper half frequency. This implies that different down-sampling methods has different influence on the performance of the the deep-learning based SISR models.

Therefore, if we want to designing a deep-learning based SISR model and evaluate its performance, more attention should be paid to the performance gain in the upper half frequency range, and down-sampling methods should also be taken into consideration. It is worth checking if we can train a deep-learning network for both the down-sampling and the up-sampling process, in the hope that more information in the upper half frequency can be recovered. This is our motivation of designing an auto-encoder based model, and this will be further discussed in the following chapter.

Chapter 4

Autoencoder for SISR and Implementation

After providing a detailed motivation for designing an autoencoder model for SISR, this chapter describes the structure and the implementation of the autoencoder model. Inclusion into High Efficiency Video Coding (HEVC) intra-frame coding will also be discussed.

4.1 Concept Motivation

The general structure of current deep-learning based SISR models is shown in Figure 4.1. The raw image is down-sampled by the bicubic method or the SHVC method, then the down-sampled image is up-scaled into the original size via bicubic interpolation or SHVC interpolation. This interpolated image is fed into the deep-learning network to generate the up-sampled image.

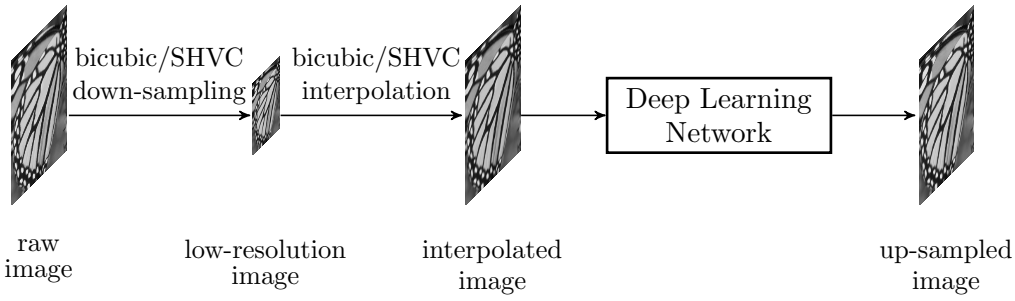


Figure 4.1: General structure of current deep-learning based SISR models.

According to the analysis in Chapter 3, on the one hand, if we want to further improve the performance over current deep-learning models, we need to recover more information in the upper half frequency. On the other hand, the down-sampling and interpolation method has an influence in the performance of the deep-learning network.

Traditional interpolation methods aim to preserve the lower half frequency components with almost no alias. Compared with the bicubic interpolation, the SHVC interpolation keeps more information within the lower half frequency with little alias, but the testing results in Section 3.4 show that using SHVC interpolation does not help the deep-learning model to recover more information in the upper half frequency. On the contrary, we observed a performance drop in the upper half frequency when using SHVC-interpolated images for training. This indicates that we should use a neural network to learn a proper down-sampling method, such that more information in the upper half frequency can be

recovered.

In this thesis, an auto-encoder SISR model is proposed to learn both the down-sampling and the up-sampling.

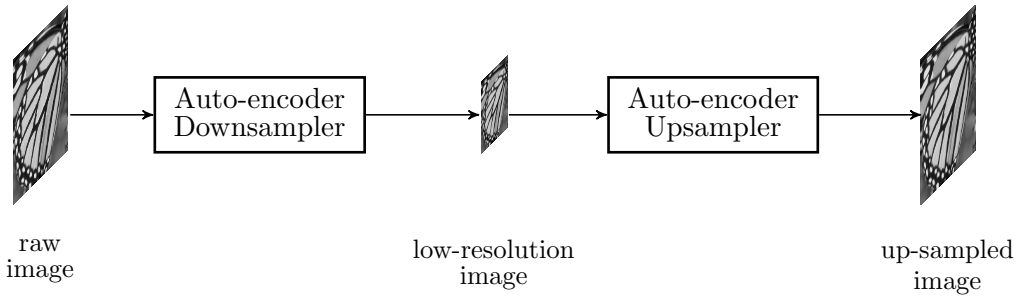


Figure 4.2: General structure of the auto-encoder SISR model.

The general structure of an auto-encoder SISR model is shown in Figure 4.2. The raw image is fist down-sampled through the auto-encoder down-sampler, then the low resolution image is up-sampled via the auto-encoder up-sampler. The down-sampler is the former part of the auto-encoder, which is a convolution neural network. The up-sampler, on the other hand, is the latter part of the auto-encoder, which is a deconvolution neural network. In Section 4.2, we will discuss the implementation of the auto-encoder model.

4.2 Autoencoder Model Implementation

In this section we will discuss in detail the implementation of the auto-encoder model, including the network structure, main characteristics and training configurations.

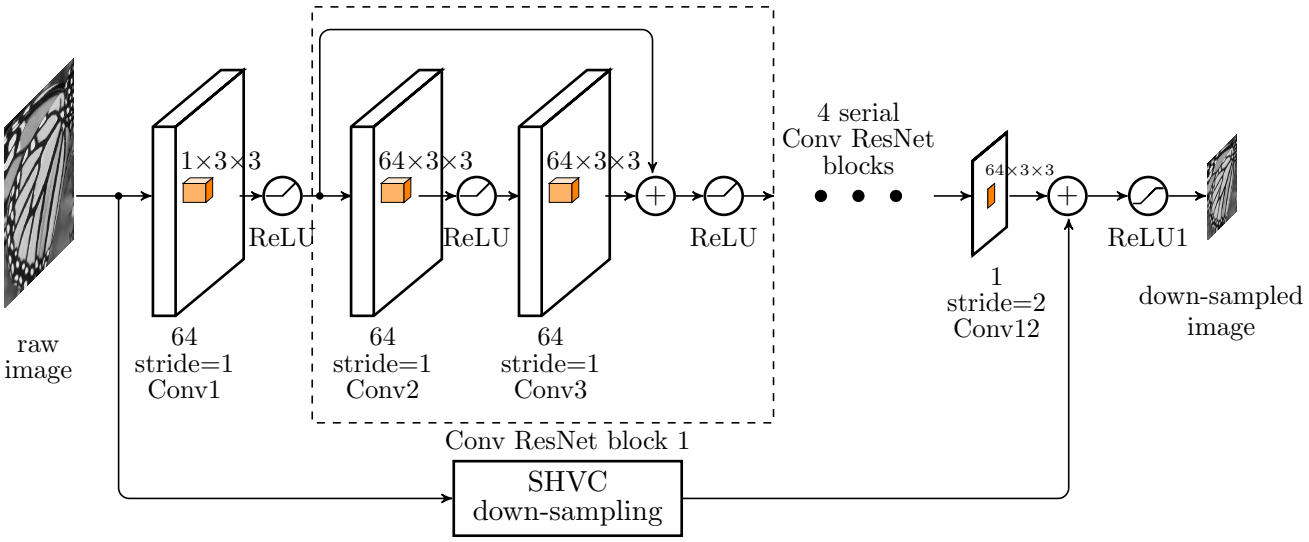


Figure 4.3: Structure of the auto-encoder down-sampler.

The structure of the auto-encoder model can be divided into two parts. The first part is the auto-encoder down-sampler which down-samples the full-resolution image into the low-resolution image. The default down-sampling scale is 2. The second part is the autoencoder up-sampler which up-samples the low-resolution image. Note that here the image pixel values are normalized into the range [0,1].

The network structure of the autoencoder down-sampler is shown in Figure 4.3. This is a CNN model with skip connections. There are 12 convolution layer in total. Except the last layer which has only 1 channel, the convolution output of all the other layers has 64 channels. As for the filter kernel size, the first layer has a kernel size $1\times3\times3$, and the filter size for the other layers is $64\times3\times3$. In Section 2.1.1 we introduced the concept of stride. In this auto-encoder down-sampler, the former 11 layers have the same height and width as the full-resolution image. The down-scaling is performed in the last convolution layer. Therefore, the stride is 2 for the last convolution layer, and 1 for all the other layers. Besides, zero-padding is used to maintain the same size between the convolution input and output when stride equals 1.

Moreover, the last convolution layer has a different activation function, denoted by ReLU1.

$$\mathrm{ReLU1}(x) = \min(\max(x, 0), 1) \tag{4.1}$$

Since we need to use the output of the autoencoder down-sampler as the down-sampled image, the normalized pixel value of the output must be within the range [0,1]. Therefore, the ReLU1 activation function used is to limit the pixel value within the range [0,1].

In Figure 4.3 we can notice that there is a structure named convolution ResNet block [15]. A basic ResNet block contains two layers. There is the shortcut from the input of the first layer to the convolution output of the second layer. The input of the block and the convolution output of the second layer are summed and activated by the ReLU function. In [15] the authors showed that the very deep network constructed by such a ResNet block has an advantage in convergence speed and accuracy. Therefore we adopted this structure in building the autoencoder model. In the autoencoder down-sampler there are 5 concatenated ResNet blocks.

Besides, in the autoencoder down-sampler there is a shortcut from the input to the output. In the shortcut the SHVC down-sampling method is used to down-sample the raw image. Therefore, the SHVC-down-sampled image and the image down-sampled by the neural network are summed up and then activated by the ReLU1 function. There are reasons for using this shortcut.
First, the network can converge faster by adding this shortcut, since the SHVC-down sampled image contains most of the information in the lower half frequency, what neural networks learn is the residual signal. In [3] the authors showed that such a residual-learning structure can help to speed up the convergence and improve accuracy.
Second, when trained without the SHVC-downsampling shortcut, the autoencoder down-sampler tends to introduce much alias in the frequency domain. However, in experiments

we find that too much alias in the down-sampled image will reduce the robustness against the coding artifacts when we implement the SISR model into the HEVC intra-frame video coding scheme. Therefore, by adding this SHVC down-sampling shortcut, we can limit the amount of alias by limiting the MSE between the SHVC downsampled image and the autoencoder down-sampled image.

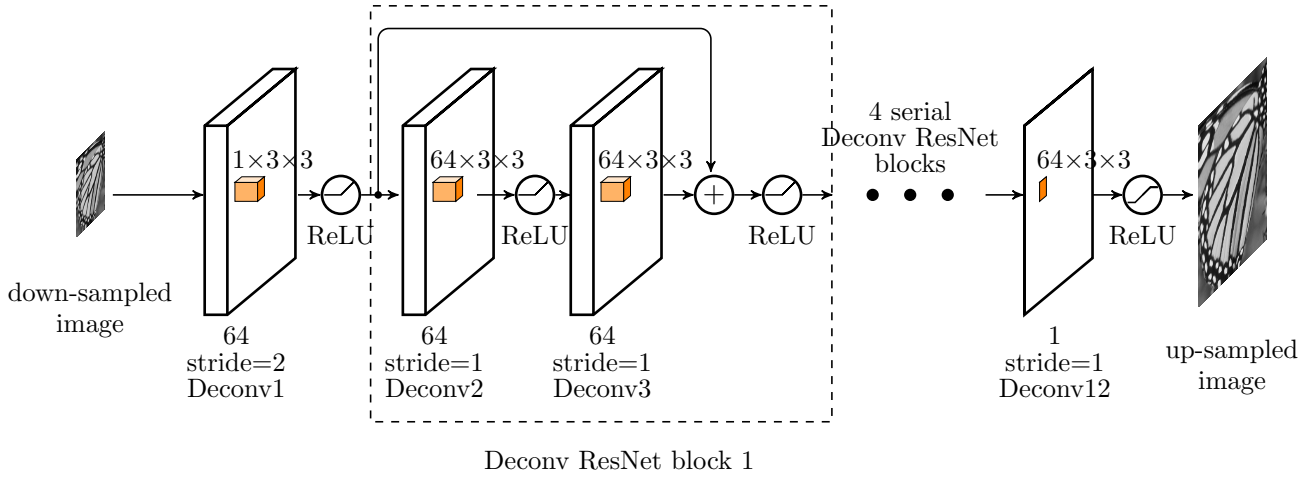


Figure 4.4: Structure of the autoencoder up-sampler.

The structure of the autoencoder up-sampler is shown in Figure 4.4. There are 12 deconvolution layers in the autoencoder up-sampler. The deconvolution operation has been introduced in Section 2.1.1. The network structure of the autoencoder up-sampler is similar to that of the autoencoder down-sampler. The output of the last layer has only 1 channel while the output of all the other layers has 64 channels. The first layer has a kernel size 1×3×3, and the filter size for all the other layers is 64×3×3. The first deconvolution layer up-scales the down-sampled image into by a factor of 2, and the following layers maintain the same size as the full-resolution image. Therefore, the stride is 2 for the first deconvolution layer, and 1 for all the other deconvolution layers. Besides, there is a ReLU activation function for each deconvolution layer.

In Figure 4.4 we can see that there are also ResNet blocks in the autoencoder up-sampler. In total there are 5 concatenated deconvolution ResNet blocks. The skip connections in these concatenated deconvolution ResNet blocks can help to speed up the convergence and avoid the gradient vanishing problem which happens frequently in training of deep neural networks.

Different from the autoencoder down-sampler, there is no direct SHVC interpolation shortcut from the input to the output of the autoencoder up-sampler. On the one hand, in testing we find that with concatenated ResNet blocks this model can up-sample the low-resolution with high accuracy. On the other hand, in the up-sampling part we do not need to control the loss between the auto-encoder up-sampled images and the SHVC-interpolated images. Therefore, there is no need to add such a SHVC-interpolation shortcut.

In the training of the complete autoencoder model, there are two main tasks. The first task is to train an autoencoder down-sampler with good robustness against coding artifacts. To

do this we can try to reduce the loss between the autoencoder down-sampled image and the SHVC-downsampled image. The second task is to train an autoencoder up-sampler which recover the high-resolution image with high accuracy compared with the raw image. For this task we can try to reduce the loss between the autoencoder up-sampled image and the raw image. These two tasks are related with each other. Therefore, this can be regarded as a multi-task learning problem [16].

In [16] the authors showed that learning multiple related tasks simultaneously can be advantageous in terms of accuracy, compared with learning these tasks independently. In this thesis we adopted the concept of multi-task learning. The overall loss function is the weighted sum of the loss in the autoencoder down-sampler part and the loss in the autoencoder up-sampler part.

Let $\{\boldsymbol{Y}_i\}$ be the raw image batch, which is the input of the autoencoder down-sampler. $\{\boldsymbol{X}_{\text{auto},i}\}$ represents the autoencoder down-sampled image batch, and $\{\boldsymbol{X}_{\text{SHVC},i}\}$ represents the SHVC down-sampled image batch. The autoencoder up-sampled image batch can be represented by $\{\boldsymbol{Y}_{\text{auto},i}\}$. The batch size is n.
The loss function in the autoencoder down-sampler part is the square of the L^2 norm of the difference between the autoencoder down-sampled image and the SHVC down-sampled image, summed over the batch.

$$L_{\text{down}} = \sum_{i}^{n} ||\boldsymbol{X}_{\text{auto},i} - \boldsymbol{X}_{\text{SHVC},i}||^2 \tag{4.2}$$

The loss function in the autoencoder up-sampler part is the square of the L^2 norm of the difference between the autoencoder up-sampled image and the raw image, summed over the batch.

$$L_{\text{up}} = \sum_{i}^{n} ||\boldsymbol{Y}_{\text{auto},i} - \boldsymbol{Y}_i||^2 \tag{4.3}$$

The overall loss function used for minimization in the training of the autoencoder model is the weighted sum of the L_{up} and L_{down}. We define a hyper-parameter β, and the overall loss function can be expressed by

$$L = \beta \cdot L_{\text{up}} + (1 - \beta) \cdot L_{\text{down}} \ , \qquad 0 < \beta < 1 \tag{4.4}$$

The parameter β represents the weight of the loss L_{up} in the overall loss. In testing we find that if β is close to 1, the autoencoder model will focus on improving the accuracy. In this case there will be much alias in the down-sampled image, and the model is less robust against coding artifacts. If β is closer, the auto-encoder down-sampled image has less alias, but the autoencoder up-sampled image has worse quality. Therefore, a proper value of β should be determined by testing such that not only the autoencoder up-sampler can recover the full-resolution image with high accuracy, but also the autoencoder down-sampler shows good robustness against coding artifacts. In this thesis, we choose 0.8 as the value of the β.

4.3 Autoencoder Dynamic-resolution Conversion in HEVC

After giving an discussion about the implementation of the autoencoder model, this section introduces how to include the SISR model into the HEVC intra-frame video coding. What we try to do is to check if more bit rate can be saved by including the SISR model into the intra-frame coding scheme, compared with the full-resolution coding scheme.

First we will introduce some basic concepts in HEVC coding. In HEVC, one factor evaluating the quality of the coding is quantization parameter (QP) [17]. The QP ranges from 0 to 51, and the mapping of the QP to the quantization step size is logarithmic. The increase of QP by 6 is corresponding to the doubling of quantization step size. Therefore, the higher the QP value is, the lower the bit rate is and the more coding artifacts there will be.

When we need to compare two coding schemes, we can compare the rate-distortion (R-D) cost. Suppose that the raw image is $\boldsymbol{X}$ and the coded image is $\tilde{\boldsymbol{X}}$, the sum of the squared difference (SSD) between the raw image and the coded image is the squared L^2 norm of the difference signal:

$$D = ||\boldsymbol{X} - \tilde{\boldsymbol{X}}||^2 \tag{4.5}$$

Let R be the bit rate for the coding of the image. According to the HEVC reference document (HM 16), the R-D cost can be expressed by

$$\begin{aligned} J &= D + \lambda \cdot R \\ \lambda &= \alpha \cdot w \cdot 2^{(\mathrm{QP}-12)/3.0} \end{aligned} \tag{4.6}$$

For intra-frame coding and non-reference images, $\alpha = 1$ and $w = 0.57$. In this thesis, we use these values by default.

In HEVC coding, we can compare the performance of coding schemes by analyzing the rate-distortion (R-D) curve. In a R-D curve, the x-axis is the bit rate, and the y-axis is the PSNR value. One measure related with the R-D curve is the Bjøntegaard-Delta (BD) metric [18]. The Bjøntegaard's metric is used to compute the average gain in PSNR (BD-PSNR) or the average percentage of saving in bitrate (BD-rate) between two RD curves. The motivation of implementing the SISR model in the HEVC intra-frame video coding is to achieve a high BD-rate gain. In the following, we will introduce the framework of a full-and-low resolution hybrid intra-frame video coding scheme.
The framework of the full-and-low resolution hybrid intra-frame coding is shown in Figure 4.5. This is a simplified version of the framework proposed in [5]. The input video signal $\boldsymbol{X}$ is first sliced into a set of small blocks with size 64×64, named as coding tree units (CTU). For each CTU, there are two coding modes to select. One mode is the full-resolution coding mode, which means that the CTU is coded at the full resolution. Another mode is the low-resolution coding. The CTU is first down-sampled into half resolution before coding. The coded CTU is then up-sampled to the original resolution. A mode selection procedure is used to select the coding mode with smaller rate-distortion cost. Finally the coded CTUs are constructed into the coded frame.

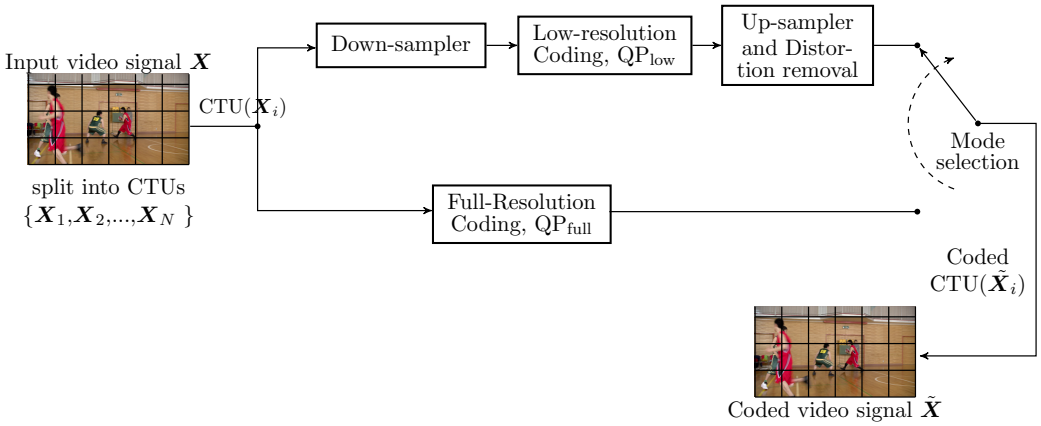


Figure 4.5: Framework of the full-and-low resolution hybrid intra-frame coding scheme.

In the mode selection procedure, we need to compute the R-D cost of both full-resolution coding and low-resolution coding. For low resolution coding, the bitrate R is the bitrate required for coding the low-resolution image, and the distortion D is the SSD between the up-sampled image and the raw image. For full-resolution coding, R is the bitrate for the coding of the full-resolution image, and D is the SSD between the coded full-resolution image and the raw image. The distortion D is computed at the full-resolution level for a fair comparison. Besides, for colorful video sequence with three channels (Y,U,V), the distortion is the sum of the SSD for Y, U, and V channels, and the bitrate is the overall bitrate for these three channels.

In this framework, for full-resolution coding, all the distortion comes from coding artifacts and the bitrate is for coding the full resolution image. For low-resolution coding, the distortion comes from both the coding artifacts and the information loss caused by down-sampling and up-sampling, and the bitrate is for coding the low-resolution. If we choose the same QP for both coding modes, the full-resolution coding will have smaller distortion while larger bitrate, and the low-resolution coding will have smaller bitrate while larger distortion, so the R-D cost for both modes is not comparable. In [5] the authors showed that in order to make the RD cost of both coding modes comparable, there is an approximated relationship between the QP for full-resolution coding and the QP low-resolution coding:

$$\mathrm{QP}_{\mathrm{low}} = \mathrm{QP}_{\mathrm{full}} - 6 \tag{4.7}$$

In the following, we will introduce the detail of including the SISR model into the low-resolution coding mode of the hybrid coding framework.

Since the human visual system has a lower acuity for color differences than for luminance, in video coding the chroma components can be further sub-sampled in order to save bitrate. For example, in the YUV 4:2:0 format, the size of the chroma components is half of the size of the luma components. Due to the difference in scale, the down-sampling and up-sampling of the luma and chroma components should be handled separately.

The up-sampler and distortion removal part aims to up-sample the coded low-resolution image and remove the distortion caused by coding artifacts. In this part we adopt the

five-layer neural network model proposed in [5]. Note that the up-sampling and distortion removal of the luma components and the chroma components are performed by different models, and both models has a similar structure.

The structure of the model for the up-sampling and distortion removal of the luma components is shown in Figure 4.6.

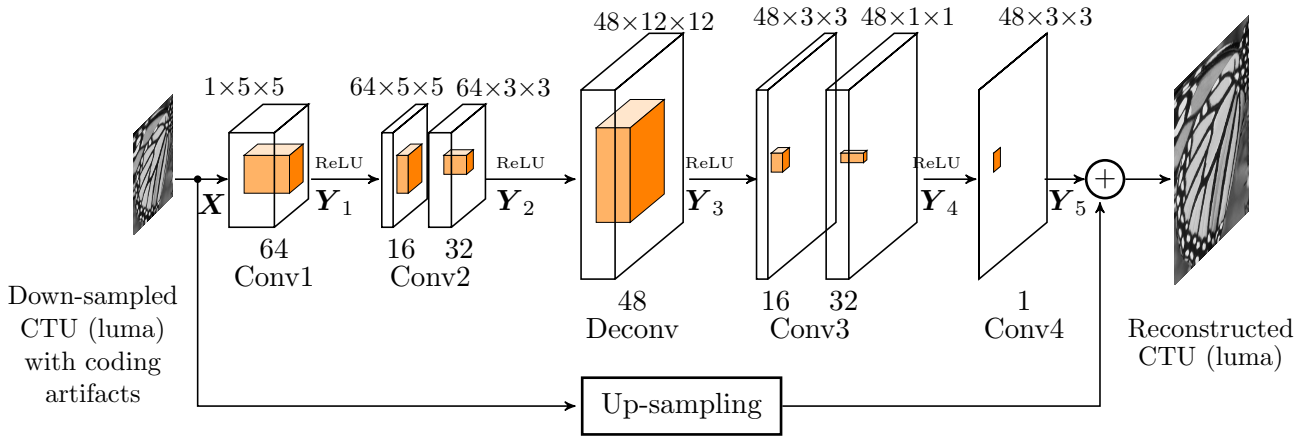


Figure 4.6: The structure of the model for up-sampling and distortion removal of the luma components [5]. The number above each layer is the filter kernel size.

As is shown in Figure 4.6, the down-sampled CTU (luma) is fed into the network for up-sampling and distortion removal. This network contains two branches. One branch is the up-sampling path. In this branch there is a up-sampler. This up-sampler can be the SHVC interpolation or a pre-trained deep-learning based up-sampler, such as the autoencoder up-sampler. Through this path the low-resolution CTU is up-sampled into the full-resolution CTU.

Another branch is 5-layer neural network. This branch aims to remove the distortion caused by coding artifacts and further enhance the quality of the reconstructed CTU. In this 5-layer network, the first two layers are convolution layers at the low-resolution level, followed by a deconvolution layer which up-scales the input by a factor of 2. The last two layers are convolution layers at the full-resolution level.
The main characteristic of this 5-layer network is that it adopts a multi-scale convolution structure at the second and the third convolution layers. Take the second convolution layer as an example, in this layer there are two different parts. The first part has 16 different filter kernels of size 64×5×5. The second part has 32 different filter kernels of size 64×3×3. The input of the second layer is the output of the first layer, $\boldsymbol{X}_1$. Let $\mathbf{W}_{21}$ and $\mathbf{B}_{21}$ be the filter kernels and bias of the first part, and $\mathbf{W}_{22}$ and $\mathbf{B}_{22}$ be the filter kernels and bias of the second part. Combined with the ReLU activation function, the output of the second layer $\boldsymbol{Y}_2$ can be expressed by

$$\boldsymbol{Y}_2 = \begin{cases} \max(0, \mathbf{W}_{21} * \boldsymbol{Y}_1 + \mathbf{B}_{21}) \\ \max(0, \mathbf{W}_{22} * \boldsymbol{Y}_1 + \mathbf{B}_{22}) \end{cases} \tag{4.8}$$

The output of these two parts are concatenated in the channel axis to generate the output of the second layer.
According to the discussion in Section 2.1.3, the filter size is related with the receptive field. In [5], the authors state that by using a multi-scale convolution structure, each layer can have information of various receptive fields, and this can help to improve the predictive accuracy of the neural network.

The model for the up-sampling and distortion removal of the chroma components of the CTUs is similar to that of the luma components.

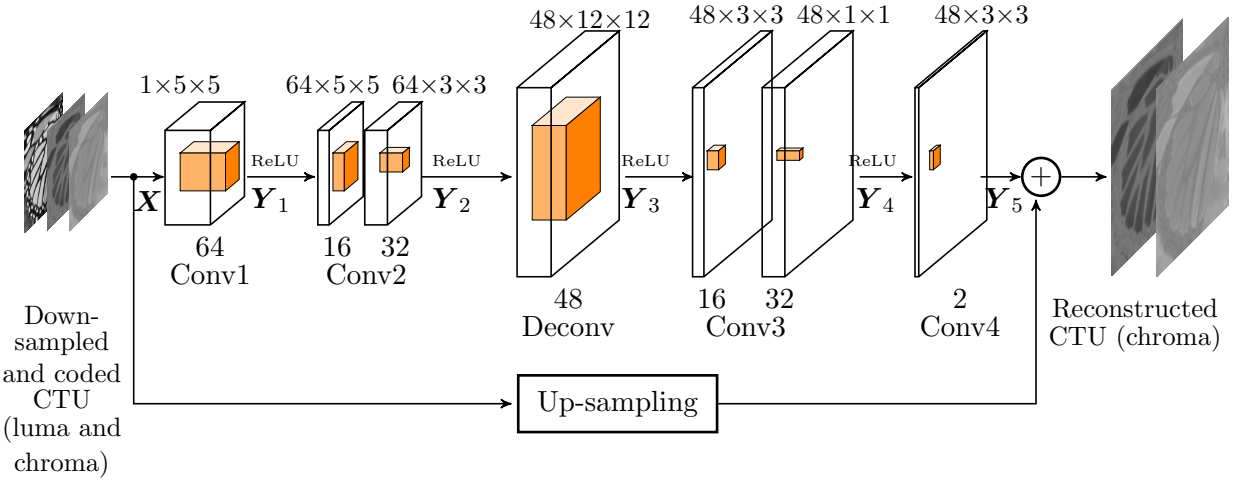


Figure 4.7: The structure of the model for up-sampling and distortion removal of the chroma components [5].

As shown in Figure 4.7, for the up-sampling of the chroma components, this network uses both luma and chroma channels as input. In [5] the authors showed that there exists correlations between luma channel and chroma channel. Motivated by this, they proposed to reconstruct the chroma from both chroma and luma.
Besides, in this thesis the default format of YUV files is 4:2:0, so in the original CTU the size of Y channel is 64×64, and the size of U and V channels is 32×32. In the down-sampled CTU, Y channel has a size of 32×32 while U and V channels have a size of 16×16. Therefore, in order to fed three channels into the neural network, the luma channel of the down-sampled CTU need to be further down-sampled into the size 16×16 by bicubic down-sampling.

In this thesis one of the main tasks is to test if the autoencoder model proposed in Section 4.2 shows some advantages relative to the SHVC down-sampling and up-sampling methods, when applied into the full-and-low resolution hybrid intra-frame video coding scheme. Therefore, two schemes are compared.
One scheme is the SHVC-based scheme. As shown in Figure 4.5, for the down-sampler part of the low-resolution coding mode, SHVC method is used for the down-sampling of both luma and chroma. Besides, SHVC method is used in the up-sampling shortcuts of the luma (Figure 4.6) and the chroma (Figure 4.7).

Another scheme is the autoencoder-based scheme. In this thesis we only training the autoencoder model for the luma. Therefore, in this autoencoder-based scheme, for down-sampling part, the autoencoder down-sampler is used for luma, and the SHVC method is used for the chroma. For the up-sampling shortcut, the autoencoder up-sampler is used for the luma model shown in Figure 4.6, while the SHVC method is used for the chroma model shown in Figure 4.7. Performance evaluation of these two schemes are given in the next chapter.

Chapter 5

Evaluation and Results of Autoencoder Model

This chapter gives the evaluation of the performance of the autoencoder-based SISR model. First the autoencoder model is tested on the data Set5 and Set14. We compare the results of the autoencoder model, the VDSR and the SHVC interpolation and analyze the performance in frequency domain. Then the SHVC-based and autoencoder-based full-and-low resolution hybrid intra-frame coding schemes are compared in terms of R-D curve and BD-rate.

5.1 Performance of Autoencoder-based SISR Model

In this section we first give the testing results in terms of the PSNR, the ESNR, the ESNR^{up}, the ESNR^{low}, as well as the upper-half weight coefficient w^{u}. Apart from the autoencoder model, two other models are also tested for comparison. One model is the SHVC interpolation which preserves most information in the lower half frequency with little alias. Another model is the VDSR model which achieves the nearly state-of-art performance. The testing results are shown in Table 5.1.

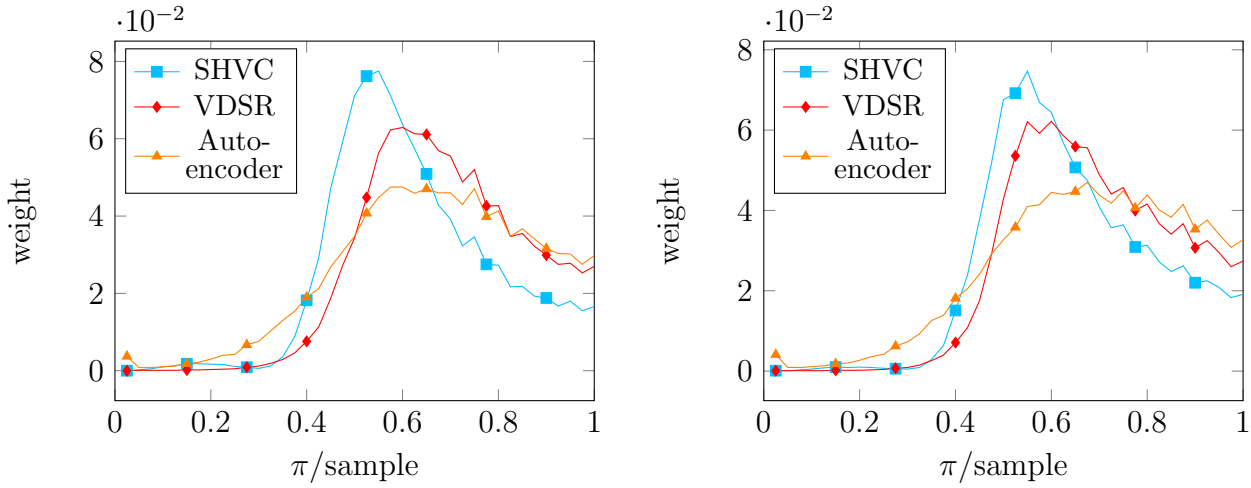
Table 5.1: Testing results of three SISR models. All results are averaged over all images in Set5 and Set14 respectively.

Method	Set5					Set14				
	PSNR (dB)	ESNR (dB)	ESNR^{low} (dB)	ESNR^{up} (dB)	w^{u}	PSNR (dB)	ESNR (dB)	ESNR^{low} (dB)	ESNR^{up} (dB)	w^{u}
SHVC	34.39	27.84	34.04	1.53	0.7489	30.73	24.61	31.51	0.62	0.7863
VDSR	37.43	30.89	40.45	3.82	0.8871	32.95	26.83	36.27	2.34	0.8811
Autoencoder	**38.09**	**31.54**	38.52	**4.97**	0.7916	**34.25**	**28.13**	35.35	**4.05**	0.8036

Comparing the results between the autoencoder model and the VDSR model, a higher PSNR value of the autoencoder model is observed. When analyzing the ESNR^{up} and ESNR^{low}, we can find that the autoencoder model has a better performance in the upper half frequency. For example, when tested on Set5, for the VDSR model the error energy in the upper half frequency is around the 42% of the raw image energy in the upper half frequency, while for the autoencoder model this ratio is reduced to 32%. However, in the upper half frequency, the autoencoder model has a slightly worse performance than the VDSR model.

The value of the ESNR, the ESNR^{up} and ESNR^{low} is directly related with the error energy. According to the comparison of the ESNR^{up} and ESNR^{low}, compared with the VDSR

model, the autoencoder model has more error energy in the lower half frequency and less error energy in the upper half frequency. Therefore, when comparing the upper-half weight coefficient w^u which represents the ratio of the error energy in the upper half frequency to the overall error energy, we find that the autoencoder model has a smaller w^{u} than the VDSR model. This can also be seen from the weight spectrum. As shown in Figure 5.1, for



(a) Set5. (b) Set14.

Figure 5.1: Comparison of weight spectrum of different SISR methods. The curves are averaged over all images in Set5 and Set14 respectively.

the SHVC interpolation, the error energy mainly distributes near frequency 0.5π. For the VDSR model, the distribution of the error energy shifts to the higher frequency and centers at 0.6π. However, the error distribution of the autoencoder model is more disperse over frequency.

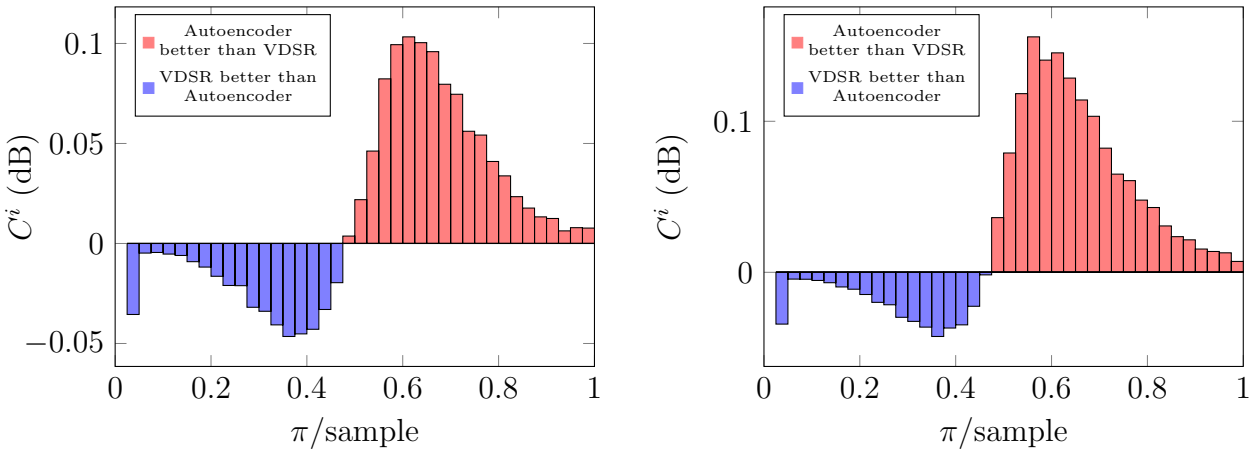
In Figure 5.1 we notice that the autoencoder model achieves a higher PSNR than the VDSR model. In order to analyze from which frequency part, either the lower half frequency or the upper half frequency, the gain in PSNR comes, we can compute the contribution coefficient C^{ℓ} and C^{u} from the results in Table 5.1. The testing results are shown in Table 5.2.

Table 5.2: Analysis of performance gain of the Autoencoder model against the VDSR method. All results are averaged over all images in Set5 and Set14 respectively.

Data	Methods	ΔPSNR	ΔESNR	$\Delta\mathrm{ESNR}^{\mathrm{low}}$	$\Delta\mathrm{ESNR}^{\mathrm{up}}$	$\bar{w^{\mathrm{u}}}$	C^{ℓ}	C^{u}	$\Delta\mathrm{ESNR}_{\mathrm{e}}$
Set5	VDSR→Autoencoder	0.66 dB	0.65 dB	-1.93 dB	1.15 dB	0.8394	-0.31 dB	0.97 dB	0.66 dB
Set14	VDSR→Autoencoder	1.30 dB	1.30 dB	-0.92 dB	1.71 dB	0.8424	-0.15 dB	1.45 dB	1.30 dB

From Table 5.2 we can see that given the change in the $\mathrm{ESNR}^{\mathrm{up}}$, the change in the $\mathrm{ESNR}^{\mathrm{low}}$ and the average upper-half weight coefficient, the change in the overall ESNR (or PSNR) can be accurately estimated according to (2.44). When tested on Set5, the performance gain of the autoencoder model in the upper half frequency accounts for a 0.97 dB gain in the

overall ESNR, while the performance loss in the lower half frequency accounts for a 0.31 dB drop in the overall ESNR. However, the positive upper-half contribution counteracts the negative lower-half contribution, so in total we observe an obvious gain in the overall ESNR. The same relationship can be observed from the results of Set14.
We can analyze the contribution to the overall PSNR gain over frequency by the contribution spectrum.



(a) Set5. (b) Set14.

Figure 5.2: Contribution spectrum of of the Autoencoder model against the VDSR model. The curves are averaged over all images in Set5 and Set14 respectively.

As shown in Figure 5.2, when comparing the autoencoder model with the VDSR model, there are positive contribution to the overall PSNR gain in the upper half frequency while negative contribution to the overall PSNR gain in the lower half frequency. By comparing the red area and the blue area in Figure 5.2, we notice that the positive contribution is larger than the negative contribution, so the overall PSNR of the autoencoder model is higher than that of the VDSR model. In experiments we find these negative contributions are caused by the alias introduced by the autoencoder down-sampler.
These results show that in order to improve the performance of the SISR models, it is possible to sacrifice some quality in the lower half frequency and improve the quality in the upper half frequency. The alias introduced by the autoencoder down-sampler may play a role in recovering the information in the upper half frequency.

5.2 Rate Distortion Analysis

In this section, we analyze the rate-distortion performance of the full-and-low resolution hybrid intra-frame coding scheme via R-D curve and BD-rate. The performance of the SHVC-based scheme and the autoencoder-based scheme introduced in Section 4.3 will be analyzed and compared.

Figure 5.3 shows the R-D curves of several typical test sequences. The first frame of these test sequences is used. Three coding scheme is under test: the full resolution coding scheme,

the SHVC-based hybrid coding scheme and the autoencoder-based hybrid coding scheme. In this test, the QPs value for the full resolution coding are {32, 37, 42, 47}. According to the introduction in Section 4.3, the relationship between the QP_{full} for full resolution coding and the QP_{low} for low resolution coding is $QP_{full} = QP_{low} + 6$. Therefore, the QP_{low} values are {26, 31, 36, 41}.

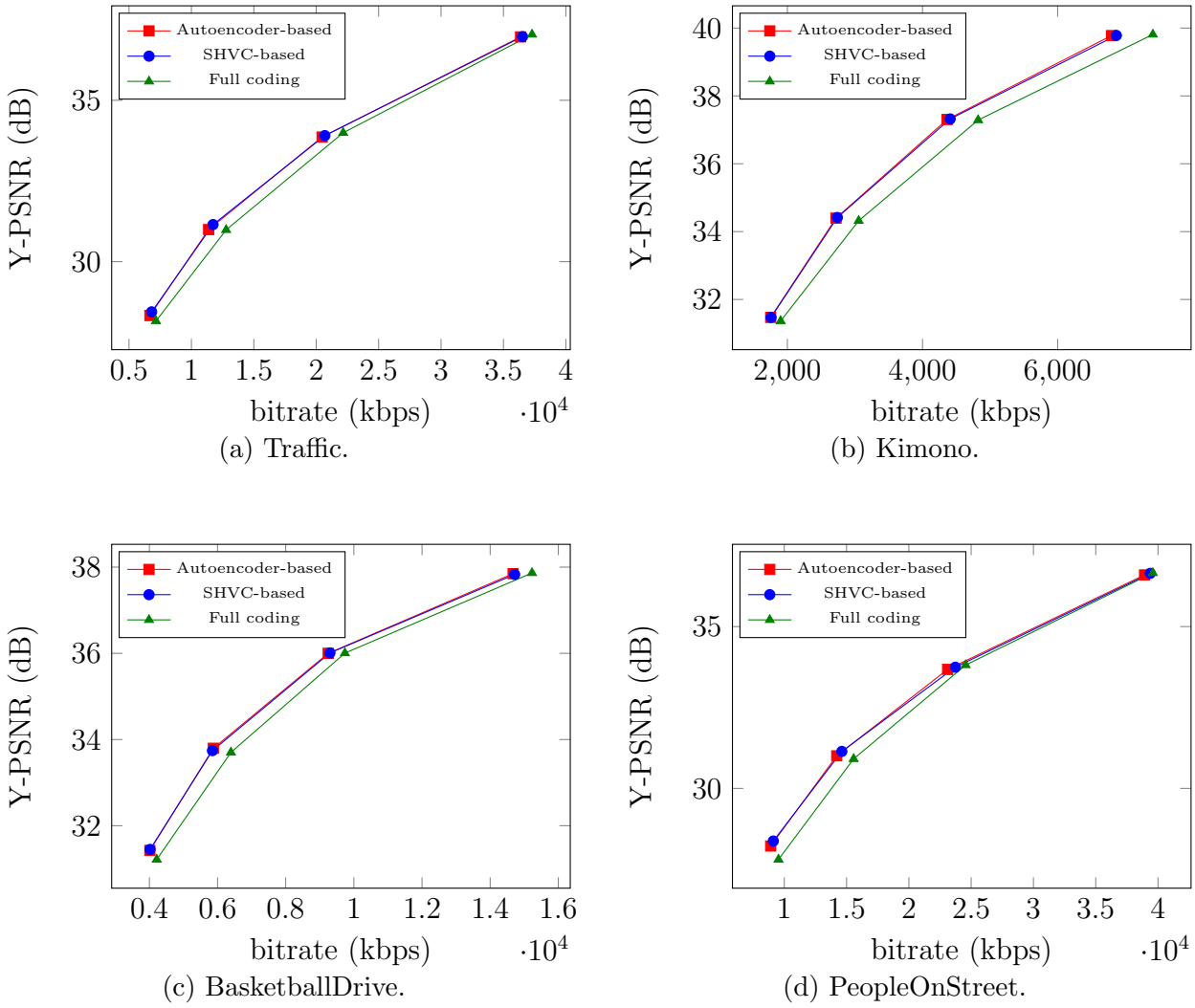
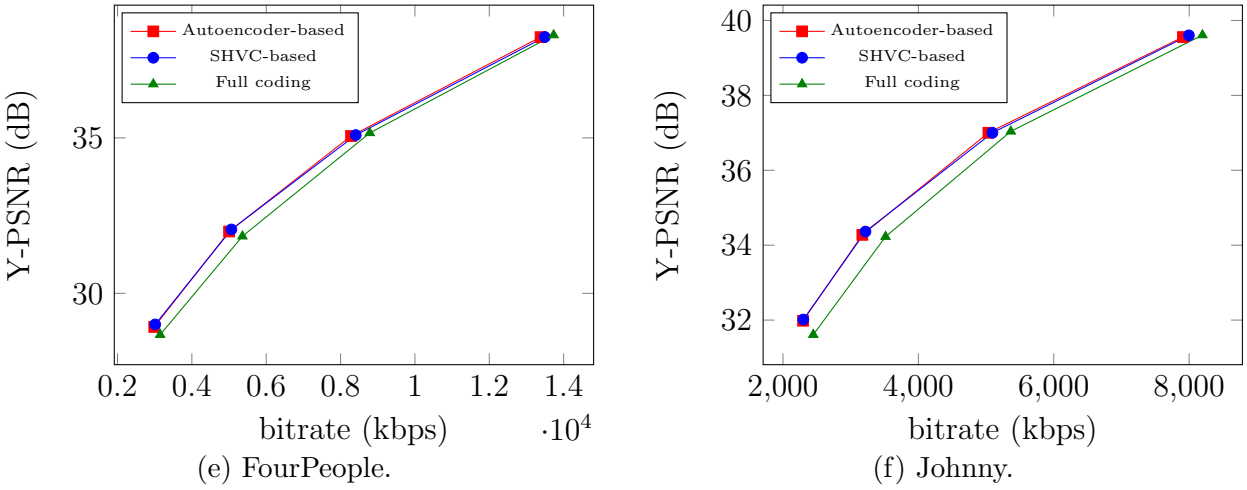


Figure 5.3: R-D curves of several typical test sequences.

From Figure 5.3 we can see that compared with the SHVC-based hybrid coding scheme, the autoencoder-based hybrid coding scheme achieve slight bitrate reduction for QP_{full} ={32, 37}. We can compute and compare the BD-rate of the SHVC-based hybrid coding scheme and the autoencoder-based coding scheme. Two cases are tested. The first case is set the range of the QP for full resolution coding as QP_{full}={32, 37, 42, 47}. In the second case the range of QP is QP_{full}={22, 27, 32, 37}. The results are shown in Table 5.3 and 5.4.



(e) FourPeople. (f) Johnny.

Figure 5.3: R-D curves of several typical test sequences.

From the results in Table 5.3 and 5.4, we can see that the autoencoder-based hybrid coding scheme can achieve a better performance on average for Y, U, V channel, and the improvement is around 0.3%. This results implies that autoencoder down-sampler and up-sampler are proved to be advantageous relative to SHVC methods in the context of hybrid resolution intra-frame video coding. Besides, either for SHVC based model or auto encoder based model, full-and-low hybrid coding scheme doesn't show obvious improvement over full resolution coding scheme for the coding of V channel.

Table 5.3: Comparison of BD rate of SHVC-based scheme and autoencoder-based scheme. The range of QP_{full} is {32, 37, 42, 47}.

Sequences	SHVC-based coding scheme			Autoencoder-based coding scheme		
	Y	U	V	Y	U	V
Traffic	-7.3%	-3.5%	11.0%	**-7.4%**	**-4.3%**	**-8.7%**
PeopleOnStreet	-6.0%	-4.0%	-2.1%	**-6.9%**	**-5.3%**	**-5.2%**
Kimono	-9.9%	-10.4%	-11.5%	**-10.5%**	**-11.3%**	**10.4%**
ParkScene	-5.3%	-13.8%	17.1%	-5.3%	-13.8%	**12.1%**
Cactus	-5.8%	7.5%	9.9%	**-5.9%**	7.8%	**9.1%**
BQTerrace	-2.8%	-1.3%	2.5%	**-2.9%**	**-1.4%**	**0.4%**
BasketballDrive	-6.8%	3.2%	0.1%	**-7.2%**	3.7%	0.6%
FourPeople	-5.8%	4.1%	-0.4%	**-6.3%**	**2.6%**	1.0%
Johnny	-7.1%	2.5%	3.2%	**-7.6%**	3.2%	4.8%
KristenAndSara	-5.8%	-2.9%	-3.1%	**-6.3%**	-2.1%	-2.2%
Average	-6.3%	-1.9%	5.0%	**-6.6%**	**-2.1%**	**4.0%**

Table 5.4: Comparison of BD rate of SHVC-based scheme and autoencoder-based scheme. The range of QP_{full} is {22, 27, 32, 37}.

Sequences	SHVC-based coding scheme			Autoencoder-based coding scheme		
	Y	U	V	Y	U	V
Traffic	-1.0%	-1.3%	0.8%	**-1.1%**	-1.3%	**0.5%**
PeopleOnStreet	-0.4%	-1.2%	-1.5%	**-0.6%**	**-2.0%**	**-2.3%**
Kimono	-5.3%	-6.2%	0.8%	**-6.0%**	**-6.9%**	**0.5%**
ParkScene	-0.7%	-3.2%	3.2%	-0.7%	**-3.3%**	**2.5%**
Cactus	-0.8%	0.4%	0.2%	-0.8%	**0.3%**	0.2%
BQTerrace	-0.1%	-0.1%	-0.1%	**-0.2%**	-0.1%	**-0.4%**
BasketballDrive	-1.7%	0.1%	-0.7%	**-2.0%**	**0.0%**	-0.4%
FourPeople	-0.8%	-0.9%	-1.5%	**-1.3%**	**-1.6%**	-1.2%
Johnny	-1.9%	1.3%	-0.2%	**-2.1%**	**-0.1%**	**-0.7%**
KristenAndSara	-1.6%	-0.5%	-1.3%	**-1.8%**	**-1.0%**	**-2.1%**
Average	-1.43%	-1.16%	-0.03%	**-1.66%**	**-1.60%**	**-0.34%**

Chapter 6

Conclusion

Single image super resolution (SISR) is an ill-posed problem in computer vision and shows its potential in the context of video coding. Since the SRCNN model was first proposed, it has become the current research focus in this field that training deep-learning based model to perform super resolution. For example, the VDSR model adopted a residual network with 20 convolution layers. The authors in [3] showed that by increasing the network depth and adding a skip connection, the VDSR model can achieve higher accuracy with a faster convergence speed. Other models such as DRCN were proposed to further improve the performance over the VDSR model.

In this thesis, we aimed to analyze the performance of different SISR model in the frequency domain. In analogy to the definition of the PSNR, one measure named Energy Signal-to-Noise Ratio (ESNR) was designed. The ESNR evaluates the ratio of the error energy to the raw image energy in the logarithmic scale. During analyzing the properties of the ESNR, two important measures, weight spectrum and contribution spectrum, were proposed. The weight spectrum represents the distribution of error energy over frequency. It was proved theoretically that in order to achieve more obvious gain in the PSNR, more efforts should be put into improving the performance in the frequency range where is error distribution is high. When comparing the performance of two SISR models, the contribution spectrum can be used to show how much the contribution of the performance change over a certain frequency range to the overall gain in the ESNR value.

By using these measures, we analyzed the performance of current SISR models from various aspects. By comparing the performance of the VDSR model and the SHVC method, we found that the performance gain of the VDSR model comes not only from the upper half frequency but also from the lower half frequency. The comparison between the 5-layer CNN model and the VDSR model revealed that for deep-learning based SISR model, increasing the network depth can help to recover more information in the upper half frequency. Moreover, in this thesis we first found that different down-sampling methods have an deep influence on the training and performance of the deep-learning based SISR models. Using down-sampling and interpolation methods with almost no alias for training is not helpful for the network to recover the information in the upper half frequency.

Based on these conclusions, an autoencoder model which can learn both down-sampling and up-sampling operations simultaneously was designed, with the hope that the autoencoder model can learn a proper down-sampling method, such that more information in the upper half frequency range can be recovered. Testing results showed that compared with the VDSR model this autoencoder model can achieve higher PSNR values. Analysis in

frequency domain revealed that this autoencoder model has a worse performance in the lower half frequency, but a better performance in the upper half frequency. By implementing it into the full-and-low hybrid resolution intra-frame video coding scheme, the autoencoder model also showed an improvement over the SHVC-based coding scheme.

Nevertheless, it still remains unclear that how the autoencoder model manage to recover more information in the upper half frequency. By analyzing the low-resolution image down-sampled by the autoencoder down-sampler, it can be found that the image down-sampled by the autoencoder down-sampler has more alias than the SHVC down-sampled image. One possible assumption is that these alias contain useful information which can help to recover upper half frequency components. If we figure out the main features of these alias, and how they are related with the upper half frequency components, then these alias can be useful in the context of video coding. For example, in the transmitter part, we can first extract the alias generated by the autoencoder down-sampler. This alias signal and the SHVC down-sampled image are coded and transmitted separately. In the receiver part, the received alias could be used as parameters for the up-sampling of the coded low-resolution image. It is worthing checking if more bitrate can be saved in this way.

Appendix A

List Of Abbreviations

Table A.1: List of abbreviations.

Abbreviation	Full Name
SISR	Single Image Super Resolution
SR	Super Resolution
CNN	Convolution Neural Network
DCT	Discrete Cosine Transform
SRCNN	Super Resolution Convolution Neural Network
VDSR	Very Deep network for Super Resolution
DRCN	Deeply Recursive Convolution Network
HEVC	High Efficiency Video Coding
SHVC	Scalable Extensions of the High Efficiency Video Coding
ReLU	Rectified Linear Unit
PSNR	Peak Signal-to-Noise Ratio
GDS	Gradient Descent
ESNR	Energy Signal-to-Noise Ratio
MSE	Mean Square Error
DFT	Discrete Fourier Transform
QP	Quantization Parameter
BD	Bjøntegaard-Delta
CTU	Coding Tree Unit
R-D	Rate-Distortion
SSD	Sum of Squared Differences

Appendix B

List Of Symbols

Table B.1: List of symbols.

Symbol	Meaning
$\boldsymbol{X}$	input image in spatial domain
$\boldsymbol{Y}$	raw image in spatial domain
$\boldsymbol{S}_x$	spectrum of image $\boldsymbol{X}$
$\boldsymbol{S}_y$	spectrum of image $\boldsymbol{Y}$
$L(\Theta)$	loss function
F_{r}	receptive field
w^{u}	upper-half weight coefficient
C^{ℓ}	lower-half contribution coefficient
C^{u}	upper-half contribution coefficient
α	ratio of the upper-half frequency raw energy to the overall raw energy
E^i	energy spectrum
ESNR^i	ESNR spectrum
w^i	weight spectrum
C^i	contribution spectrum

Bibliography

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. "Image super-resolution using deep convolutional networks". In: *IEEE transactions on pattern analysis and machine intelligence* 38.2 (2016), pages 295–307 (cited on pages 1, 6, 7).

[2] Yang Jianchao, John Wright, Thomas Huang, and Yi Ma. "Image super-resolution as sparse representation of raw image patches". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*. 2008, pages 1–8 (cited on page 1).

[3] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. "Accurate image super-resolution using very deep convolutional networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pages 1646–1654 (cited on pages 1, 8–10, 37, 51).

[4] 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*. 2016, pages 1637–1645 (cited on page 1).

[5] Yue Li, Dong Liu, Houqiang Li, Li Li, Feng Wu, Hong Zhang, and Haitao Yang. "Convolutional neural network-based block up-sampling for intra frame coding". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2017) (cited on pages 1, 40–43).

[6] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. "Image super-resolution using deep convolutional networks". In: *IEEE transactions on pattern analysis and machine intelligence* 38.2 (2016), pages 295–307 (cited on page 1).

[7] Ren Yang, Mai Xu, Zulin Wang, and Zhenyu Guan. "Enhancing Quality for HEVC Compressed Videos". In: *arXiv preprint arXiv:1709.06734* (2017) (cited on page 1).

[8] Feng Jiang, Wen Tao, Shaohui Liu, Jie Ren, Xun Guo, and Debin Zhao. "An end-to-end compression framework based on convolutional neural networks". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2017) (cited on page 1).

[9] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. "Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pages 2802–2810. URL: `http://papers.nips.cc/paper/6172-image-restoration-using-very-deep-convolutional-encoder-decoder-networks-with-symmetric-skip-connections` (cited on page 2).

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pages 1097–1105 (cited on page 7).

[11] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pages 600–612 (cited on page 11).

[12] Alain Hore and Djemel Ziou. “Image quality metrics: PSNR vs. SSIM”. In: *Pattern recognition (icpr), 2010 20th international conference on*. IEEE. 2010, pages 2366–2369 (cited on page 11).

[13] Jens Ohm. *Multimedia content analysis*. Springer, 2016, page 111 (cited on page 19).

[14] Mitsuo Takeda, Hideki Ina, and Seiji Kobayashi. “Fourier-transform method of fringe-pattern analysis for computer-based topography and interferometry”. In: *JosA* 72.1 (1982), pages 156–160 (cited on page 23).

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pages 770–778 (cited on page 37).

[16] Theodoros Evgeniou and Massimiliano Pontil. “Regularized multi–task learning”. In: *KDD*. 2004 (cited on page 39).

[17] Gary J Sullivan, Jens Ohm, Woo-Jin Han, and Thomas Wiegand. “Overview of the high efficiency video coding (HEVC) standard”. In: *IEEE Transactions on circuits and systems for video technology* 22.12 (2012), pages 1649–1668 (cited on page 40).

[18] G Bjontegaard. “Calculation of average PSNR differences between RD-Curves”. In: (Jan. 2001) (cited on page 40).