Digital Image Processing Final Report

Report about Accurate Image Super-Resolution Using Very Deep Convolutional Networks

Dachuan Chen
10828241

Department of Electrical Engineering
Chung Yuan Christian University
2022/12/23

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Introduction about the topic of the report.

For games to generate frames of image in real-time, it can use up nearly all of the computational power of GPU for some less expensive machines. But if we can use a super-resolution algorithm to generate a higher resolution image, we can use the same amount of computational power to generate a higher resolution image. This can reduce the amount of computational power needed to generate frames of image in real-time. In this report, I will introduce the super-resolution algorithm and the method to implement it.

The following report is splitted into 2 parts. The first part is the introduction of the super-resolution algorithm and the method to implement it, containing the idea summarized from the paper I selected. The second part is the experimental results and the conclusion based on the implementation of the algorithm.

Chapter 2

Content of the Paper

The source of "Accurate Image Super-Resolution Using Very Deep Convolutional Networks" is downloaded from IEEE Xplore.

This paper proposed a very deep CNN residual network for image super-resolution, which is a successor of SRCNN model with even better performance. With the utilization of residual learning, the proposed network can be trained with a lot deeper layers, less parameters, and converge faster than SRCNN model.

The standard the research team used is the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM). The PSNR is used to measure the quality of model output high resolution (HR) image, and the SSIM is a measure of the similarity between ground truth image and output HR image.

According to wikipedia, the PSNR (in dB) is defined via Mean Square Error (MSE):

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(I_{i,j} - I'_{i,j}\right)^2 \tag{2.1}$$

where $I_{i,j}$ is the pixel value of the ground truth image, $I'_{i,j}$ is the pixel value of the model output image (noisy approximation), m is the height of the image, and n is the width of the image.

$$\begin{aligned} PSNR &= 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20\log_{10}\left(MAX_I\right) - 10\log_{10}\left(MSE\right) \end{aligned} \tag{2.2}$$

where MAX_I is the maximum value of the ground truth image. PSNR is mostly used to measure the quality of the model output image in the selected paper.

According to wikipedia, the SSIM is defined as:

$$SSIM = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.3}$$

where x stands for the ground truth image, y stands for the model output image, μ_x and μ_y are the mean of x and y, respectively, σ_x^2 and σ_y^2 are the variance of x and y, σ_{xy} is the covariance of x and y, and $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ are two variables to stabilize the division with weak denominator, L as the dynamic range of the pixel-values (typically this is $2^{\#\text{bits per pixel}} - 1$), $k_1 = 0.01$ and $k_2 = 0.03$ by default.

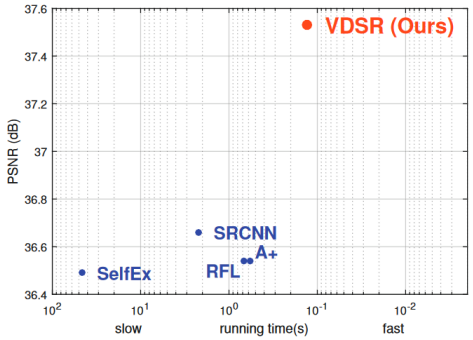


Figure 1: Our VDSR improves PSNR for scale factor ×2 on dataset Set5 in comparison to the state-of-the-art methods (SRCNN uses the public slower implementation using CPU). VDSR outperforms SRCNN by a large margin (0.87 dB).

Figure 2.1: Proposed model performance comparison.

2.1 Introduction

People have used bicubic interpolation, bilinear interpolation, lanczons resampling, ..., and SRCNN for image super-resolution (SR).

Since this paper is the successor of SRCNN, it mainly focuses on the difference with it. For the downsides of SRCNN:

- Relies on the context of small image regions.
- Training converges too slowly.
- The network only works for a single scale. (1x, 1.5x, 2x, ...)

For the proposed model:

- Big image (size) can be used for training.
- Model using residual-learning, gradient clipping, and huge learning rate (contrary to SRCNN) can speed up convergence to its maximum.
- Proposed model is sufficient for multiple scales SR.

2.2 Related Work

Learn and compare with SRCNN by deep-learning-based SR approach.

Difference with SRCNN:

- (Model) SRCNN takes too much time to train, but proposed 20 layers model of 3×3 filters does a better job.
- (Training) SRCNN uses HR image to carry the input to the end layer, but proposed model process residual images directly making it much faster.
- (Scale Factor) Proposed model can be used for multiple scales, and by zero padding every layer it output the same size of HR image.

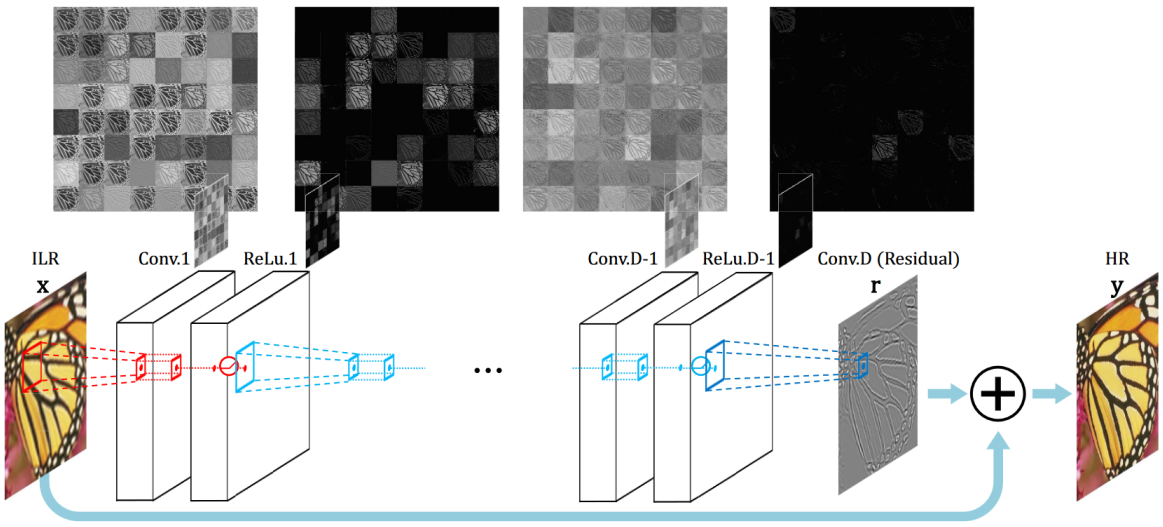


Figure 2: Our Network Structure. We cascade a pair of layers (convolutional and nonlinear) repeatedly. An interpolated low-resolution (ILR) image goes through layers and transforms into a high-resolution (HR) image. The network predicts a residual image and the addition of ILR and the residual gives the desired output. We use 64 filters for each convolutional layer and some sample feature maps are drawn for visualization. Most features after applying rectified linear units (ReLu) are zero.

Figure 2.2: Proposed model structure.

2.3 Proposed Method

2.3.1 Proposed Network

Further details of the proposed model:

- ILR: Input low resolution image. Generating from downscaled ground truth HR image and use bicubic interpolation to scale up to the original size.
- HR: Output high resolution image. Generating from combining input LR image and residual image (generated with residual-network).
- Zero padding: Zero padding is used to make the output image the same size as the ground truth HR image (convolution crops images).

2.3.2 Training

- Basic
  - x is the input LR image.
  - y is the ground truth HR image.
  - y' is the output HR image.
  - $\{x^{(i)}, y'^{(i)}\}$ is the data set.
  - f is the model.
  - $f(x)$ is the output of the model. (residual image)
  - Objective function: $L(f) = \frac{1}{2}||y' - f(x)||^2$.
- Residual-Learining
  - $r = y' - x$ is the residual image.

  - Objective function: $L(f) = \frac{1}{2}||r - f(x)||^2$. (since this model is trained with residual images)

- Training details
  - Custom loss layer: residual estimate (r) + network input ILR image (x) + ground truth HR image (y).
  - Loss: Euclidean distance between $x + r$ and y. (straight line distance between two pixels)
  - Regression: Mini-batch gradient descent based on back-propagation.
  - Momentum: 0.9.
  - Regularized by weight decay (L_2 penalty multiplied by 0.0001).

- High learning rates for very deep Networks
  - It is not realistic to converge 20 layers of SRCNN network, but it can be done within 4 hours with the proposed model.
  - Gradient clipping: $-\frac{\theta}{\gamma} <$ learning rate $< \frac{\theta}{\gamma}$, where γ is the current learning rate, θ is the weights assigned to particular features. With this, the learning rate is maximized and clipped to prevent exploding gradients.

- Multi-Scale
  - Proposed model can be used for multiple scales.
  - Input batch size is equal to the size of receptive field (64).
  - The source images can be different scales. (1.5x720p, 2x540p, 3x360p, ...)

2.4 Understanding Properties

In this section, the paper discuss the following properties of the proposed model:

- Show large depth is necessary for SR.
- Show residual-learning network converges much faster than the standard CNN (SRCNN).
- Single network performs as well as method using multiple networks trained for specific scale.

2.4.1 The Deeper, the Better

- 3×3 filters are used for all layers.
- For depth D, the receptive field size is $(2D + 1) \times (2D + 1)$.
- Large receptive field, more context to predict image details.
- Very deep networks can exploit high nonlinearities. (19 rectified linear units).
- In most cases, performance increases as depth increases.

Epoch	10	20	40	80
Residual	36.90	36.64	37.12	37.05
Non-Residual	27.42	19.59	31.38	35.66
Difference	9.48	17.05	5.74	1.39

(a) Initial learning rate 0.1

Epoch	10	20	40	80
Residual	36.74	36.87	36.91	36.93
Non-Residual	30.33	33.59	36.26	36.42
Difference	6.41	3.28	0.65	0.52

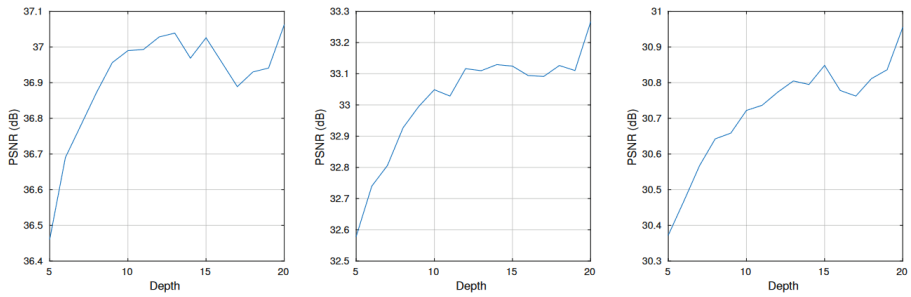
(b) Initial learning rate 0.01

Epoch	10	20	40	80
Residual	36.31	36.46	36.52	36.52
Non-Residual	33.97	35.08	36.11	36.11
Difference	2.35	1.38	0.42	0.40

(c) Initial learning rate 0.001

Table 1: Performance table (PSNR) for residual and non-residual networks ('Set5' dataset, $\times 2$). Residual networks rapidly approach their convergence within 10 epochs.

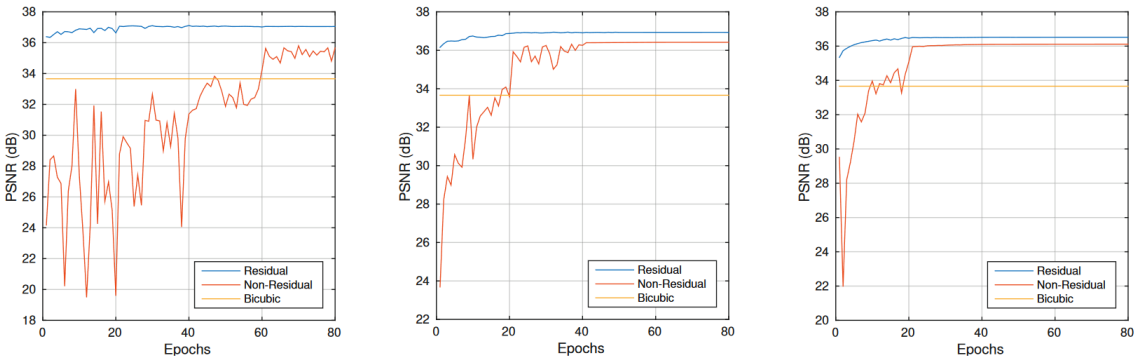
Figure 2.3: Performance difference between residual and non-residual networks.



(a) Test Scale Factor 2 **(b)** Test Scale Factor 3 **(c)** Test Scale Factor 4

Figure 3: Depth vs Performance

Figure 2.4: Performance difference between layer depth.



(a) Initial learning rate 0.1 **(b)** Initial learning rate 0.01 **(c)** Initial learning rate 0.001

Figure 4: Performance curve for residual and non-residual networks. Two networks are tested under 'Set5' dataset with scale factor 2. Residual networks quickly reach state-of-the-art performance within a few epochs, whereas non-residual networks (which models high-resolution image directly) take many epochs to reach maximum performance. Moreover, the final accuracy is higher for residual networks.

Figure 2.5: Performance difference between residual and non-residual networks.

Test / Train	×2	×3	×4	×2,3	×2,4	×3,4	×2,3,4	Bicubic
×2	37.10	30.05	28.13	37.09	37.03	32.43	37.06	33.66
×3	30.42	32.89	30.50	33.22	31.20	33.24	33.27	30.39
×4	28.43	28.73	30.84	28.70	30.86	30.94	30.95	28.42

Table 2: Scale Factor Experiment. Several models are trained with different scale sets. Quantitative evaluation (PSNR) on dataset 'Set5' is provided for scale factors 2,3 and 4. Red color indicates that test scale is included during training. Models trained with multiple scales perform well on the trained scales.

Figure 2.6: Performance difference between model trained with different scales images.

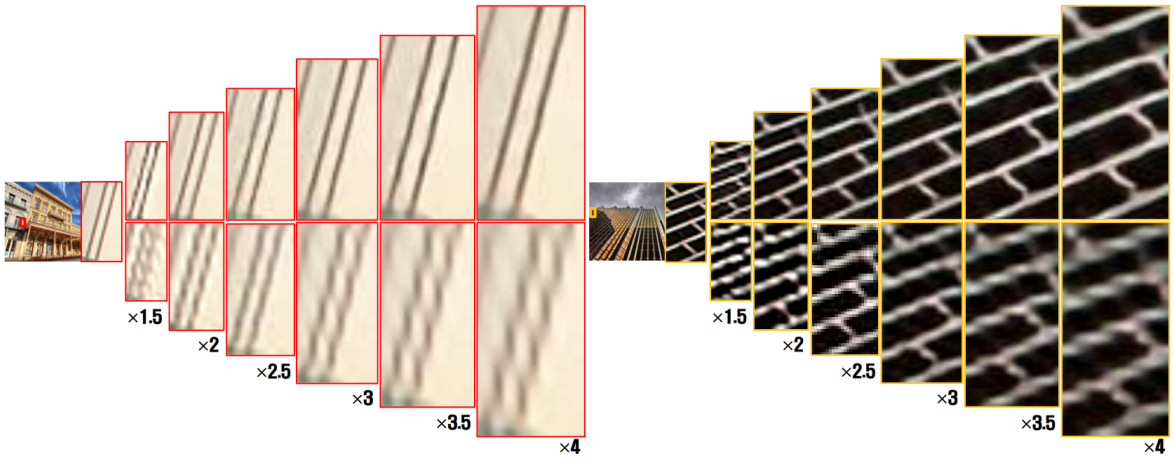


Figure 5: (Top) Our results using a single network for all scale factors. Super-resolved images over all scales are clean and sharp. (Bottom) Results of Dong et al. [5] (×3 model used for all scales). Result images are not visually pleasing. To handle multiple scales, existing methods require multiple networks.

Figure 2.7: Performance difference between model trained with different scales images.

2.4.2 Residual-Learning

Residual network converges much faster, performs better, and has less parameters to train.

2.4.3 Single Model for Multiple Scales

Proposed model trained with multiple scales images boosts the performance of the model.

2.5 Experimental Results

Describe datasets used and parameter settings.

2.5.1 Datasets for Training and Testing

- Training Dataset
  - 91 images from Yang et al, and 200 images from Berkeley Segmentation Dataset.
  - Data augmentation, including rotation and flipping is used.
- Test Dataset
  - Set5
  - Set14
  - Urban100
  - B100

Table 2.1: Training parameters

Parameter	Value
model depth	20
batch size	64
momentum	0.9
weight decay	0.0001
weight initialization	method in this paper
activation function	rectified linear units (ReLU)
training epochs	80
training iterations	9960
training batch size	64
training learning rate	0.1 (initial), decreased by factor of 10 every 20 epochs (3 times)
training time	4 hours (with NVIDIA GeForce Titan Z)

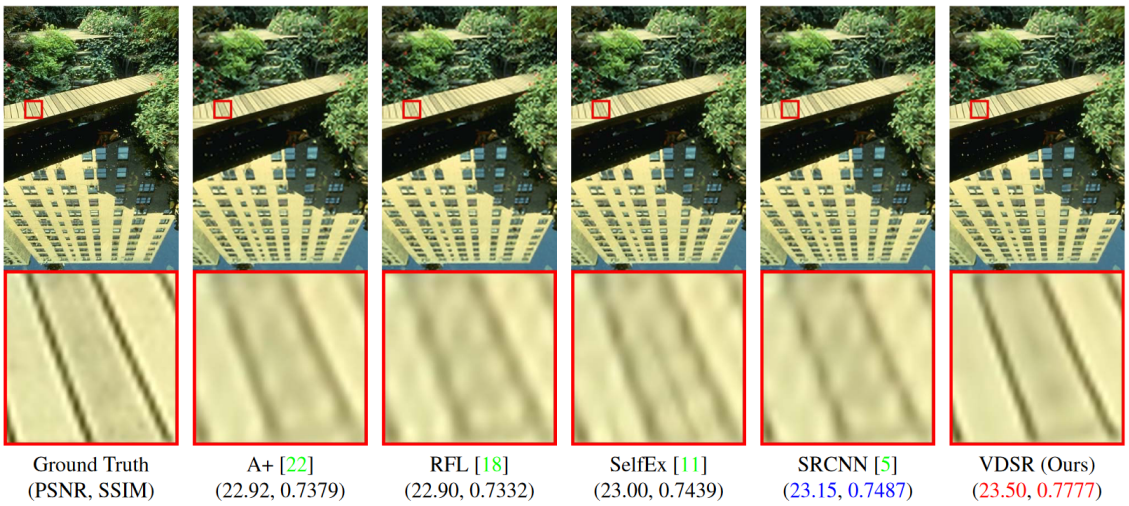


Figure 6: Super-resolution results of "148026" (*B100*) with scale factor ×3. VDSR recovers sharp lines.

Figure 2.8: Performance difference between models.

2.5.2 Training parameters

2.5.3 Benchmark

Used publickly available framework of Huang et al.

2.5.4 Comparison with the State-of-the-Art

When compared with A+, RFL, SelfEx, and SRCNN, the result is overwhelmingly good and outperform all of them. It's fast at the same time.

2.6 Conclusion

- Training very deep netwrok is hard due to slow convergence rate.
- Use residual-learning and extremely high learning rates to optimize fast.
- Convergence speed is maximized while limited by gradient clipping.
- Outperform any other models.

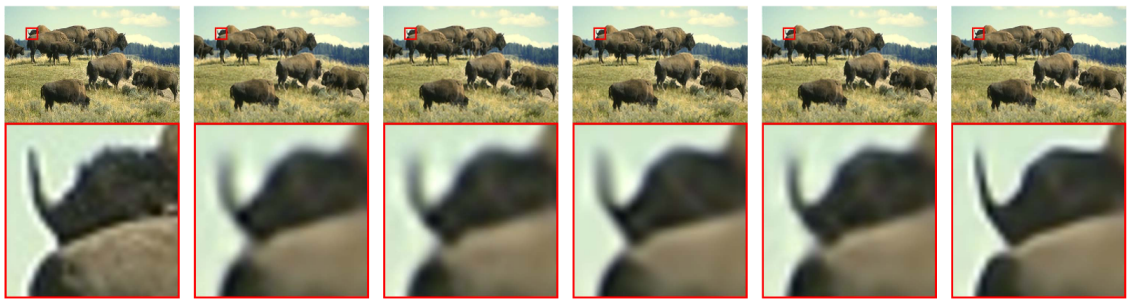


Figure 7: Super-resolution results of "38092" (*B100*) with scale factor $\times 3$. The horn in the image is sharp in the result of VDSR.

Figure 2.9: Performance difference between models.

Chapter 3

Experimental Results

Since it takes too much time to train a new model with modifications, the following section focuses on using trained weights provided by PyTorch implementor.

- PyTorch implementation: `https://github.com/Lornatang/VDSR-PyTorch` (include paper author MatLab implementation)
- Mine implementation: `https://github.com/belongtothenight/DIP_Code/tree/main/src/final` (modified config.py, add image preprocess file, add code structure)

3.1 Trained weights with Natural Dataset

In figure 3.1, the left image is the ground truth HR image, and the right is the SR image. The PSNR value is 37.41 dB. With the proposed model, it can't restore the image to the original one but still provide a very good details.

3.2 Trained weights with Generated Dataset

I use some pictures from recorded Minecraft gameplay and preprocess them to different scales. The following figure shows the performance difference different picture sizes. The PSNR and size are presented with percentage to demonstrate the performance difference.

Figure 3.1: Performance difference between models.

Figure 3.2: PSNR performance vs. different sizes of images with MC6 set.

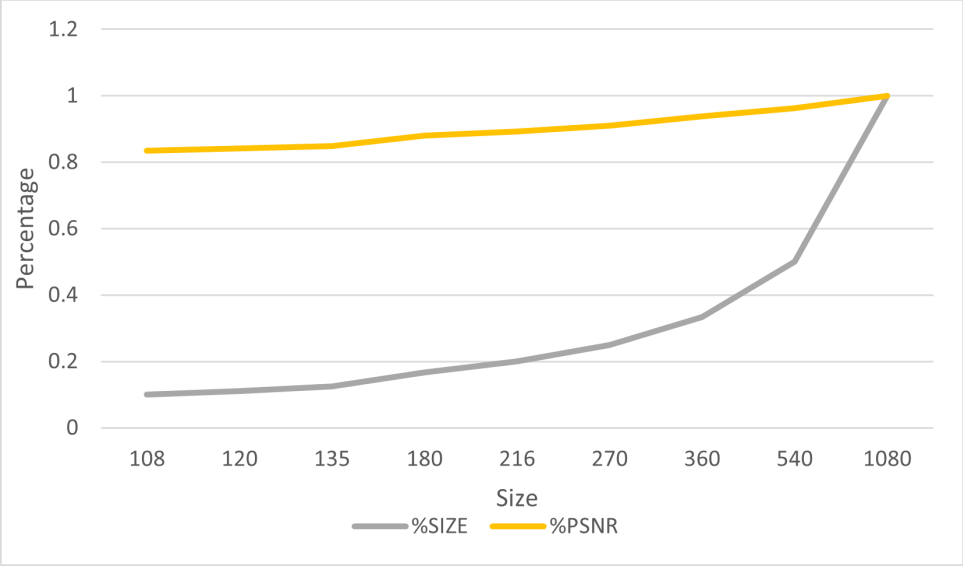


Figure 3.3: PSNR performance vs. different sizes of images with MC7 set.

Figure 3.4: PSNR performance vs. different sizes of images with MC8 set.

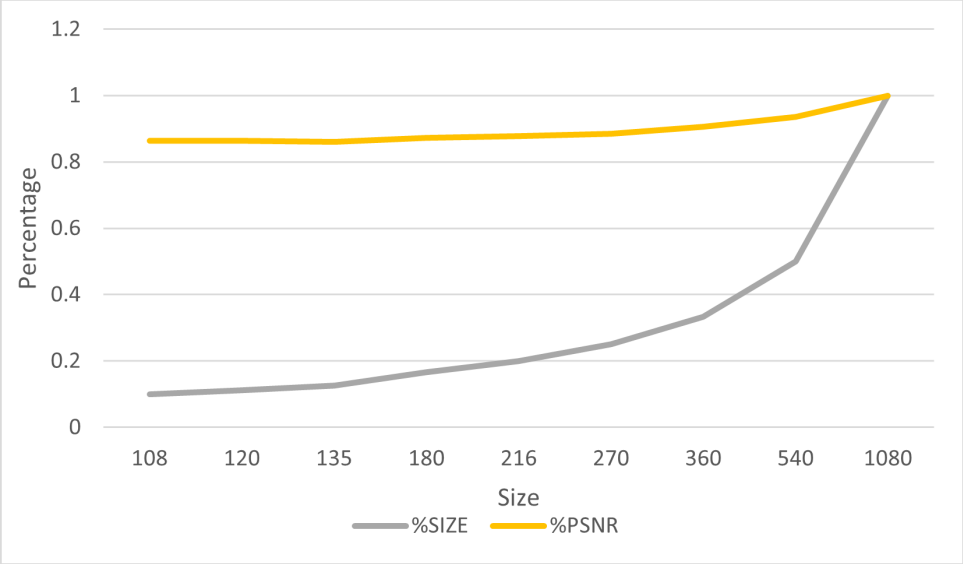


Figure 3.5: PSNR performance vs. different sizes of images with MC9 set.

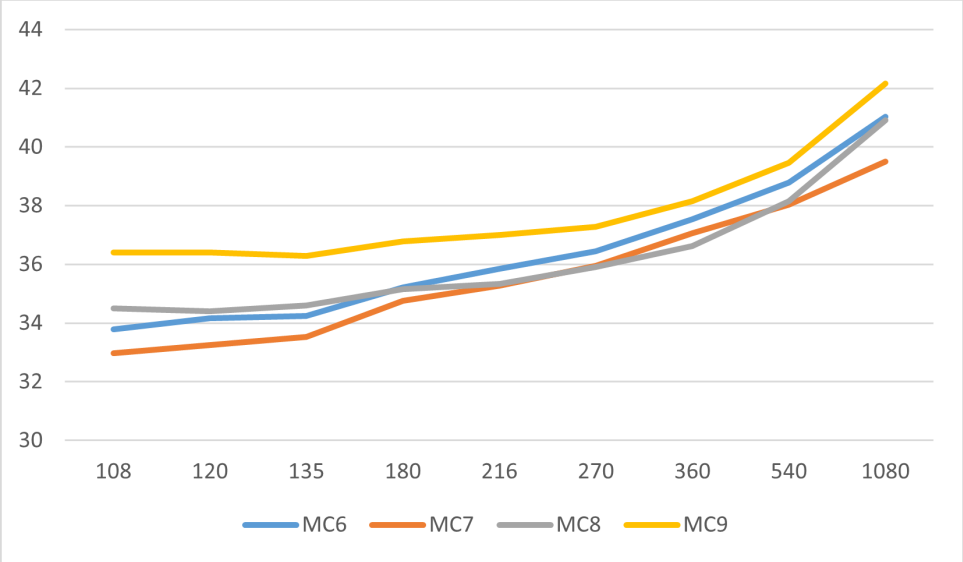


Figure 3.6: PSNR performance vs. different sizes of images with MC6-9 set.

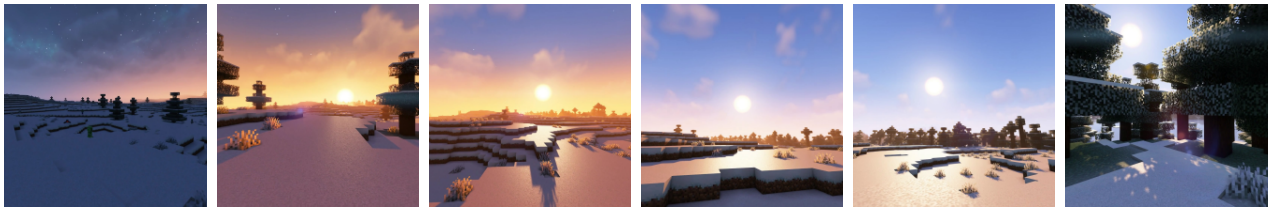


Figure 3.7: MC6 set.

From the figure 3.2-3.5, we can see that the performance of the model is affected by the size of the image. As image size increases, the performance also increases.

Figure 3.6 shows the PSNR performance difference different sets of images in figure 3.7-10.
When comparing with the set images, it indicates if the image have wider spread of pixel values the performance get worse.

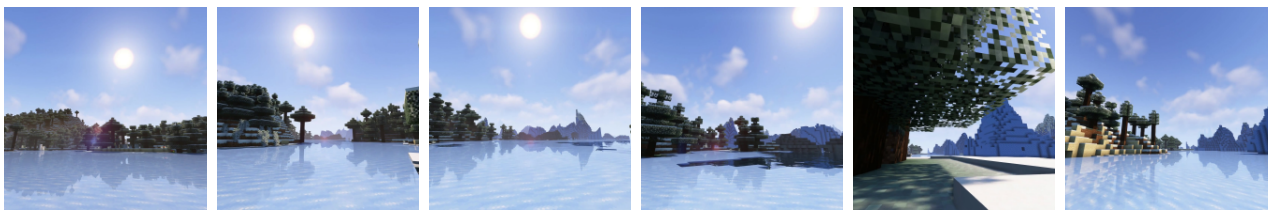


Figure 3.8: MC7 set.

Figure 3.9: MC8 set.

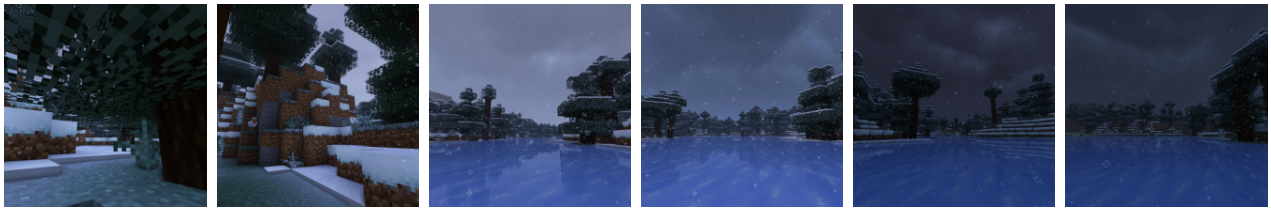


Figure 3.10: MC9 set.