NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

AI6126 Advanced Computer Vision

Individual Project 2

School of Computer Science and Engineering

Programme: MSAI

Date: 26 April 2024

Authored By: Tan Jie Heng Alfred (G2304193L)

Contents

1. Introduction

In this project, we attempted to recover super-resolution (SR) images of a human face from a low-resolution (LR) one. For the training of our model, we have 4000 high-resolution (HR) images of a human face, and we are required to perform a series of degradation to reach a synthetic LR one. For our validation set, we 400 pairs of corresponding LR-HR images whose degradation process is unknown to us. Examples of HR from the training set and LR-HR images from the validation set are shown in Figure 1.

Figure 1: Example images from training and validation sets. Left shows a HR image from the training set, middle shows the LR image from validation set, and right shows the corresponding HR image from validation set

To evaluate the performance of our model, we will be using the peak signal-to-noise ratio (PSNR). For a given image $I(c,x,y)$ of dimension $C \times H \times W$, we can calculate its PSNR with respect to its transformed image $f\big(I(c,x,y)\big)$ as,

$$\mathrm{PSNR}\Big(I(c,x,y), f\big(I(c,x,y)\big)\Big) = 10 \times \log_{10}\left[\frac{255}{\mathrm{MSE}\Big(I(c,x,y), f\big(I(c,x,y)\big)\Big)}\right],$$

$$\text{where } \mathrm{MSE}\Big(I(x,y), f\big(I(x,y)\big)\Big) = \frac{1}{CHW}\sqrt{\sum_{c,x,y}[I(c,x,y) - f(I(c,x,y)]^2}.$$

Here, the transformation f is composed of two functions, $f(X) = S_{\boldsymbol{\theta}}\big(D(X)\big)$, where $S_{\boldsymbol{\theta}}$ is our SR-network that recovers the HR image, and D is the unknown degradation process. Notice that if f is the identity map, then our SR image looks exactly like the HR image, with $\mathrm{MSE} = 0$ and PSNR approaching infinity. As such, a higher PSNR indicates that our SR-network is producing images closer to the HR images.

2. Degradation Pipeline and Model Selection

Since we are provided with only the HR images for training, we have to set up a degradation pipeline to obtain the LR images that we will attempt to perform SR on. For the purpose of our experiments, we use the same degradation pipeline as the one used in Real-ESRNet[1]. In particular, we have a two-order degradation where we perform blurring with a filter kernel, random resizing of the images, adding both noise, and simulating a JPEG compression. Then, to obtain the final LR image, we perform a sinc kernel blurring and resizing, where the order in which perform these

[1] https://github.com/xinntao/Real-ESRGAN/blob/master/realesrgan/models/realesrnet_model.py

two operations are random. This degradation process aims to simulate a real-world degradation process of obtaining a LR image (see Figure 2), so that our SR network could perform well on real-world images.

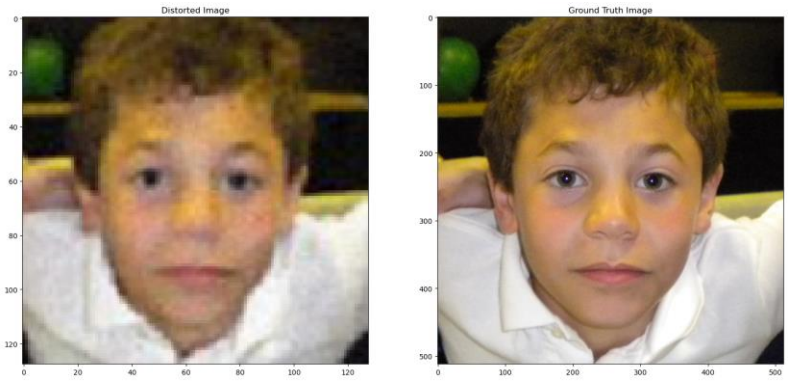

Figure 2: Left is the degraded, LR image of the corresponding HR, ground-truth image on the right.

For our model, we selected SR-ResNet[2], which has 20 residual blocks in the hidden layers, and a residual block between the LR input and SR output, among other things. For the latter, a bilinear interpolation was performed to resolve the differences in resolution. The residual connections allow us to have more layers and extract more information, while there are still paths for the gradient update to flow.

For the optimizing of the model, we chose the following loss function:

$$\mathcal{L}_{SR}(I,S) = (1-\alpha)\|SR - I\|_1 + \alpha\left\|S_{\boldsymbol{\theta}}\left(R_{128\times128}(I)\right) - I\right\|_1, \alpha \in [0,1]$$

where $S_{\boldsymbol{\theta}}$ refers to the SR network, $R_{n\times n}$ refers to the resizing operation to size $n \times n,$ and $SR = S_{\boldsymbol{\theta}}\left(D(I)\right)$ is the SR image obtained from degrading the HR image I and putting it through the SR network. The first summand is a simple $L1$ loss between the HR image, I, and the SR image. For the second summand, we note that we are resizing the original HR image to 128×128 so that, after putting it through the SR network, we obtain an image of the same dimension as the HR image. Intuitively, the HR image I should not differ much from the SR image $S_{\boldsymbol{\theta}}\left(R_{128\times128}(I)\right)$, if only a resizing operation is done. As such, this second summand can be viewed as an additional regularization term, with factor α, preventing the network from producing images too far from the ground truth. We found that by letting α decrease gradually with the number of epochs, the model produces higher quality SR image. In particular, we let α decrease from 0.8 to 0.4, following the equation,

$$\alpha = 0.4 + 0.2\left(1 + \cos\left(\frac{5000 \times \text{epoch} + \text{iter}}{5000 \times \text{epochs}}\right)\pi\right),$$

where epoch and epochs represent the current epoch run and total number of epochs respectively, and iter refers to the current iteration (each image is considered an iteration). This update is motivated by the cosine annealing scheduler. We decided to use a decreasing scale factor, because as the model trains, it should

[2] https://github.com/XPixelGroup/BasicSR/blob/master/basicsr/archs/srresnet_arch.py

learn to better model the SR images from the LR ones and does not require as much guidance from the regularizer. Alternatively, we can view this loss as gradually introducing more degradation onto the HR images, starting from mostly just a simple resizing. We hypothesized that this helps the model learn better.

3. Experiments and Results

Due to time and resource constraints, we ran a total of 10 epochs for the training at a batch size of 5, for two experimental set ups – one whose loss is $\mathcal{L}_{SR}$ (Experiment 1) as outlined above, and one whose loss is just a $L1$-loss between the SR image output and the HR image (Experiment 2). For both experiments, we used Adam optimizer, with learning rate 0.001 and weight decay of 0.0005. We also used a cosine annealing scheduler. Both experiments were conducted on an Nvidia RTX 4060 Ti (16Gb). We show the average training and validation losses and PSNRs in Figure 3 to 6 below.

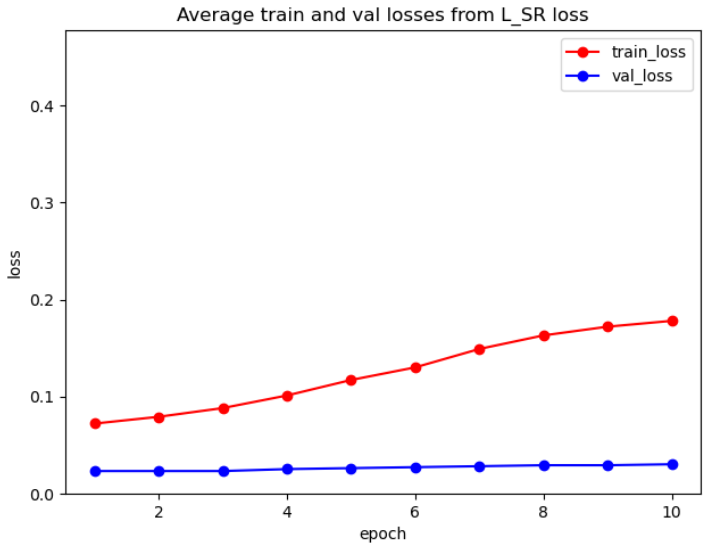

Figure 3: Average training and validation losses from experiment 1 (using $\mathcal{L}_{SR}$ loss)

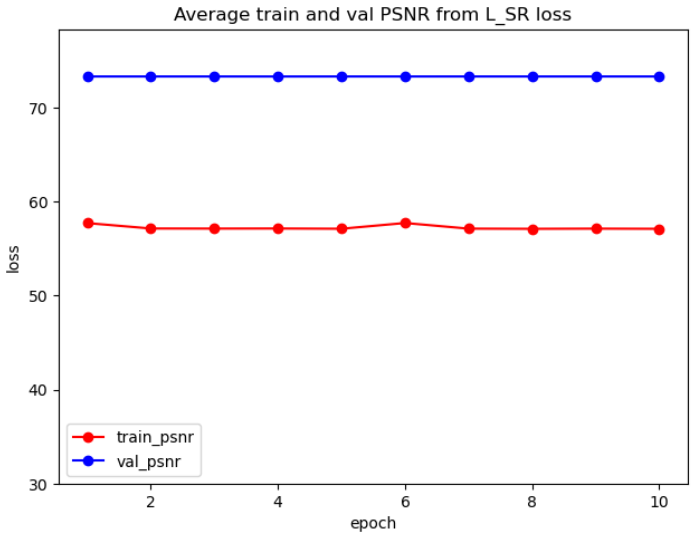

Figure 4: Average training and validation PSNRs from experiment 1 (using $\mathcal{L}_{SR}$ loss)

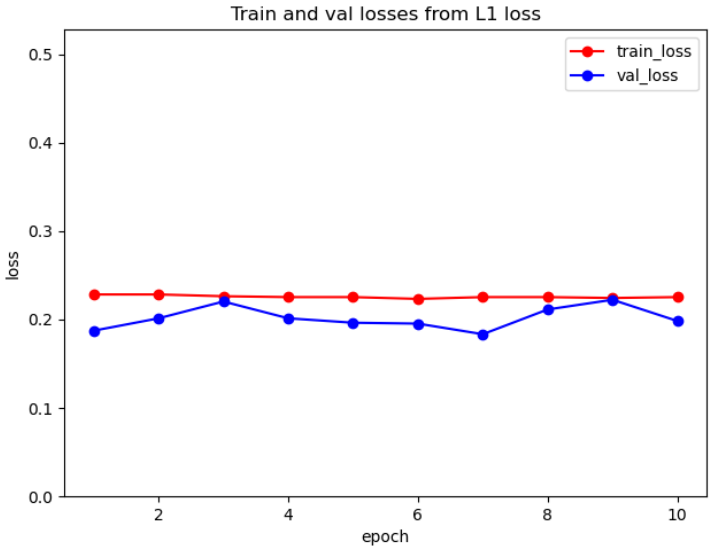

Figure 5: Average training and validation losses from experiment 2 (using $L1$-loss)

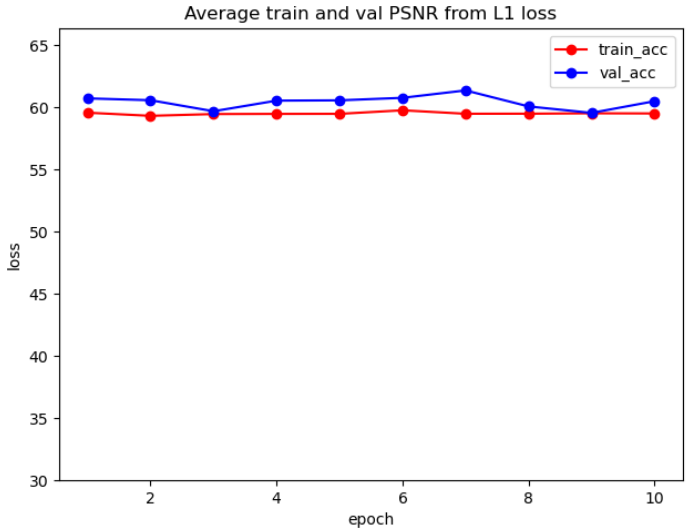

Figure 6: Average training and validation PSNRs from experiment 2 (using $L1$-loss)

The validation PSNRs when using $\mathcal{L}_{SR}$ is consistently higher than just using $L1$-loss between the SR and HR images. This suggests that the $\mathcal{L}_{SR}$ guided the model to create a more 'accurate' SR image, reducing the MSE of the SR image with respect to the HR image. Example images from experiment 1 and 2 are shown in Figure 7. Although it does not remove all the noise in the image, the model in experiment 1 at least produces an image that resembles the LR image. This is likely because of the regularization term, which guided the model to preserve the general structure and texture of the ground truth images, through the HR images. Notice also that the training losses in experiment 1 increases steadily. Typically, this is a cause for concern, but in our case the loss calculation is changing with the number of iterations. This also suggests that the model struggles to optimize the first summand $\|SR - I\|_1$ in $\mathcal{L}_{SR}$, even after many iterations. Finally, we should mention that the computational cost of experiment 1 is appreciably higher than that of experiment 2, by virtue of having two forward propagation – once to calculate each summand.

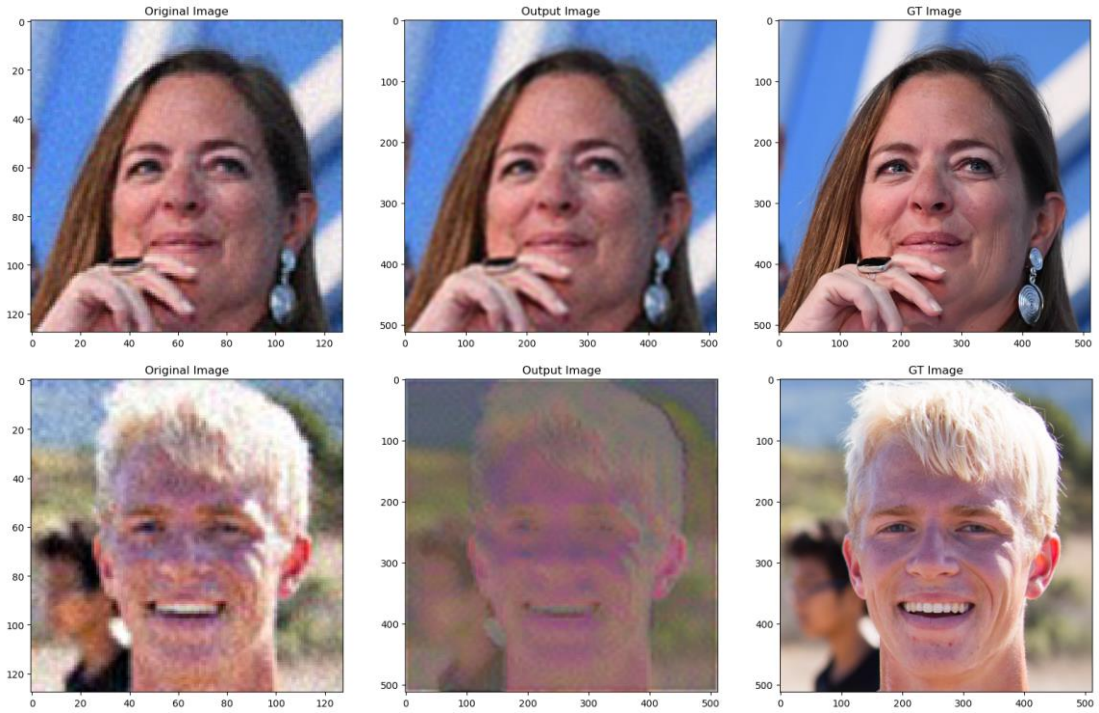

Figure 7: Example images from both experiments. Top row images are from experiment 1 and bottom row images are from experiment 2.

Because the loss calculation of experiment 1 is inconsistent across iterations and epochs, and we are hoping to maximize the PSNR score of the test set, we use the weights of the model that maximizes the validation PSNR score to produce the SR images on the test set. We reported a test set PSNR of 25.61326.

4. Conclusion

Through this project, we managed to train a network, using the SRResNet architecture, to recover a SR image from a LR image. We attempted to improve on the $L1$ loss that would have otherwise been used for the SR task, which has produced even noisier images and lower validation PSNR than our $\mathcal{L}_{SR}$ loss. We hypothesized that the second summand, $\left\|S_{\boldsymbol{\theta}}\left(R_{128\times128}(I)\right) - I\right\|_1$, in $\mathcal{L}_{SR}$ implicitly guided the model to reproduce images that do not deviate too far from the HR images. This prevents the model from generating artifacts as seen in the images produced by a simple $L1$ loss. In the future, however, we could provide more useful training signal by incorporating adversarial losses by using a GAN-based architecture.