IE643: Deep Learning : Theory and Practice **Jul-Nov 2018**

End-term Project Report : Generative Adversarial Nets for Single Image Super-Resolution

Student Name: Videsh Suman *Roll No: 150040095*

Abstract

Generative Adversarial Networks (GAN) are an alternate way of estimating generative models via an adversarial process where a game theoretic approach is involved to learn to generate from the training distribution through minimax two-player game. As a part of my initial contributions to this project, I had also implemented a vanilla GAN[1] to generate handwritten digits from a noisy latent space. Single image super-resolution (SISR) has been a notoriously challenging ill-posed problem, which aims to obtain a high-resolution output from one of its low-resolution versions. Despite the breakthroughs in accuracy and speed of SISR using faster and deeper convolutional neural networks (CNN), one central problem to recover the finer texture details super-resolving at large upscaling factors remains largely unsolved. With this motivation, I have attempted to implement the idea proposed by Christian Ledig et al.[3] to use a GAN comprising of deep convolutional networks for upscaling (4× factor) natural images to produce the then state-of-the-art photo-realistic super-resolved images.

1 Introduction

In the adversarial nets framework, first proposed by Goodfellow et al.[1], the generative model is pitted against an adversary, simultaneously training two models: the generative model G that captures the data distribution, and the discriminative model D which estimates the probability of a sample coming from the training data rather than G . The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game.
Super-resolution (SR) refers to the task of restoring high-resolution images from one or more low-resolution observations of the same scene[7]. The highly challenging task of estimating a high-resolution (HR) image just from its low-resolution (LR) counterpart is referred to as single image super-resolution (SISR). The ill-posed nature of the underdetermined super-resolution is particularly pronounced for high upscaling factors, for which texture detail in the reconstructed SISR images is typically absent. The optimization target of supervised SISR algorithms is commonly the (a) minimization of the mean squared error (MSE) between the recovered HR image and the ground truth which also maximizes the peak signal-to-noise ratio (PSNR) and/or (b) maximization of the Structural Similarity Index (SSIM).

As far as the scope of this project is concerned, I have implemented the vanilla GAN[1] trained on the MNIST dataset which generates similar handwritten digits from a single dimensional noisy latent space, and the SRGAN[3], a generative adversarial network (GAN) for SSIR, which was first proposed by Ledig et al. in 2016. To achieve photo-realistic results, I have used the perceptual loss function which consists of an adversarial loss and a content loss. The adversarial loss intends to push the solution to the natural image manifold using a discriminator network that is trained to differentiate between the super-resolved images and original photo-realistic images, while the content loss is motivated by perceptual similarity instead of similarity in pixel space. A deep residual network with skip connections has been used as the generator network which was able to recover photo-realistic textures from heavily downsampled images on publicly available natural images. An example photo-realistic image that was super-resolved with a 4× upscaling factor is shown in Figure 1.

Figure 1: Super-resolved image (left) and the original HR image (right). [4× upscaling]

2 Literature & Problem Definition

As first described by Goodfellow et al.[1] in 2014, the adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons. To learn the generators distribution p_g over data x, a prior is defined on input noise variables $p_z(z)$, then a mapping is represented to data space as $G(z;\theta_g)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. A second multilayer perceptron $D(x;\theta_d)$ is also defined which outputs a single scalar. $D(x)$ represents the probability that x came from the data rather than p_g. Now, D is trained to maximize the probability of assigning the correct label to both training examples and samples from G. Also, G is simultaneously trained to minimize $log(1-D(G(z)))$. In other words, D and G play the following two-player minimax game with value function $V(G,D)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{p_data(x)}[logD(x)] + \mathbb{E}_{zp_z(z)}[log(1-D(G(z)))]$$

In practice, this equation may not provide sufficient gradient for G to learn well. Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, $log(1-D(G(z)))$ saturates. Rather than training G to minimize $log(1-D(G(z)))$, it should be trained to maximize $logD(G(z))$. This objective function results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning [4]. The algorithm for training a multilayer perceptron GAN as stated in [1], has been mentioned in Algorithm 1. The gradient-based updates can use any standard gradient-based learning rule.

After first being proposed as fully connected multilayer perceptron GAN, various GAN architectures have been proposed in the recent times, the most impactful one being the deep convolutional GAN (DCGAN) by Redford et al.[5] that was used for the unsupervised representation learning task in natural images. Empirically, its been observed that GANs provide a powerful framework for generating plausible-looking natural images with high perceptual quality. The GAN procedure encourages the reconstructions to move towards regions of the search space with high probability of containing photo-realistic images and thus closer to the natural image manifold as shown in Figure 2.
As stated earlier, most of the super-resolution algorithms have been optimized using the metrics of peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM). PSNR is measured in decibels and used

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. They had used $k = 1$, the least expensive option, in their experiments[1].

for number of training iterations **do**
  for k steps **do**
    Sample minibatch of m noise samples $z^{(1)}, ..., z^{(m)}$ from noise prior $p_g(z)$
    Sample minibatch of m examples $x^{(1)}, ..., x^{(m)}$ from data generating distribution $p_data(x)$
    Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [logD(x^{(i)}) + log(1 - D(G(z^{(i)})))]$$

  end for
  Sample minibatch of m noise samples $z^{(1)}, ..., z^{(m)}$ from noise prior $p_g(z)$
  Update the generator by ascending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [log(D(G(z^{(i)})))]$$

end for

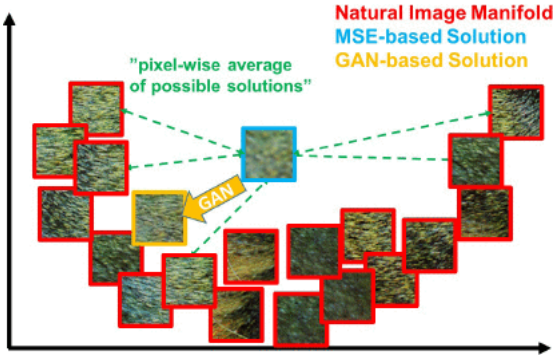

Figure 2: (Source - Ledig et al.[3]) Illustration of patches from the natural image manifold (red) and super-resolved patches obtained with MSE (blue) and GAN (orange). The MSE-based solution appears overly smooth due to the pixel-wise average of possible solutions in the pixel space, while GAN drives the reconstruction towards the natural image manifold producing perceptually more convincing solutions.

to be a common measure used to evaluate and compare SISR algorithms by capturing a higher degree of smoothness in the reconstructed image closer to the ideal one, while SSIM claims to take into account the similarity of the edges (high frequency content) between the denoised image and the ideal one. For a good SSIM measure (closer to 1), an algorithm needs to remove the noise while also preserving the edges of the objects and so, this measure is an improvement over the PSNR metric. However, the ability of PSNR and SSIM to capture perceptually relevant differences, such as high texture detail, seems limited as they are defined based on pixel-wise image differences.
This is illustrated in Figure 3, where highest PSNR and SSIM values do not necessarily reflect perceptually better SISR result, which essentially means that the SISR reconstructed image might not be photo-realistic even with higher PSNR or SSIM.

The major contributions of Ledig et al.[3] were (a) designing the first very deep (16 blocks) ResNet architecture (SRResNet) optimized for MSE; (b) proposing SRGAN which is a GAN-based network optimized for a new perceptual loss, replacing the MSE-based content loss with a loss calculated on feature maps of the VGG network, which are supposed to be more invariant to changes in pixel space, and (c) confirming with an extensive mean opinion score (MOS) test taken by 26 raters on 100 sets of SISR generated images with respect to the original (from three public benchmark datasets) that SRGAN is the new state of the art, by

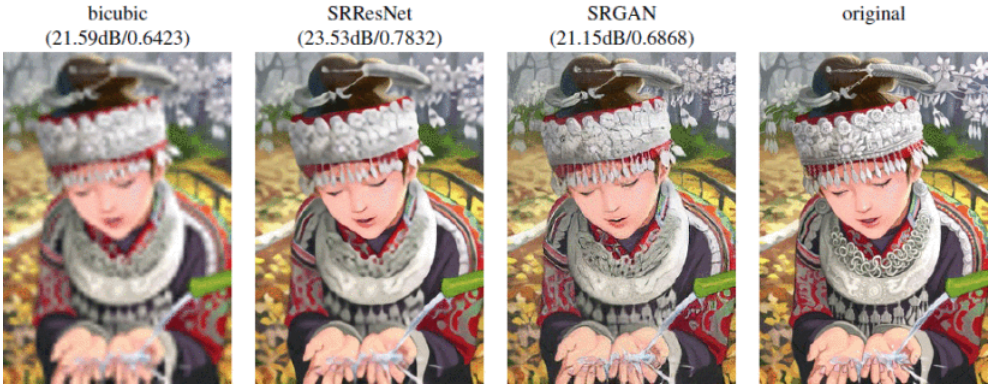

Figure 3: (Source - Ledig et al.[3]) From left to right: Bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

a large margin, for the estimation of photo-realistic SR images with high upscaling factors (4×).
In SISR the aim is to estimate a high-resolution, superresolved image I^{SR} from a low-resolution input image I^{LR}. Here I^{LR} is the low-resolution version of its high-resolution counterpart I^{HR}. The high-resolution images are only available during training. In training, I^{LR} is obtained by applying a Gaussian filter to I^{HR} followed by a downsampling operation with downsampling factor r. For an image with C color channels, we describe I^{LR} by a real-valued tensor of size $W \times H \times C$ and I^{HR}, I^{SR} by $rW \times rH \times C$ respectively.

The ultimate goal is to train a generating function G that estimates for a given LR input image its corresponding HR counterpart. To achieve this, a generator network is trained as a feed-forward CNN G_{θ_G} parametrized by θ_G. Here, $\theta_G = W_{1:L}; b_{1:L}$ denotes the weights and biases of a L-layer deep network and is obtained by optimizing a SR-specific loss function l^{SR}. Inspired by Ledig et al., this work has used a specifically designed perceptual loss l^{SR} as a weighted combination of several loss components that model distinct desirable characteristics of the recovered SR image.

Following the adversarial architecture, a discriminator network D_{θ_D} is optimized in an alternating manner along with G_{θ_G} to solve the adversarial min-max problem. The general idea behind this formulation is that it allows one to train a generative model G with the goal of fooling a differentiable discriminator D that is trained to distinguish super-resolved images from real images. With this approach the generator can learn to create solutions that are highly similar to real images and thus difficult to classify by D. This encourages perceptually superior solutions residing in the subspace, the manifold, of natural images. This is in contrast to SR solutions obtained by minimizing pixel-wise error measurements, such as the MSE. Figure 4 shows the network architecture that was used in the first SRGAN model.

The perceptual loss function l^{SR} is critical for the performance of the generator network. While l^{SR} has mainly been modeled based on the MSE in the past, Ledig et al.[3] improved on this to design a loss function that assesses a solution with respect to perceptually relevant characteristics. The perceptual loss is formulated as the weighted sum of a content loss (l^{SR}_X) and an adversarial loss component as:

$$l^{SR} = \underbrace{\underbrace{l^{SR}_X}_{\text{content loss}} + \underbrace{10^3 l^{SR}_{Gen}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}}$$

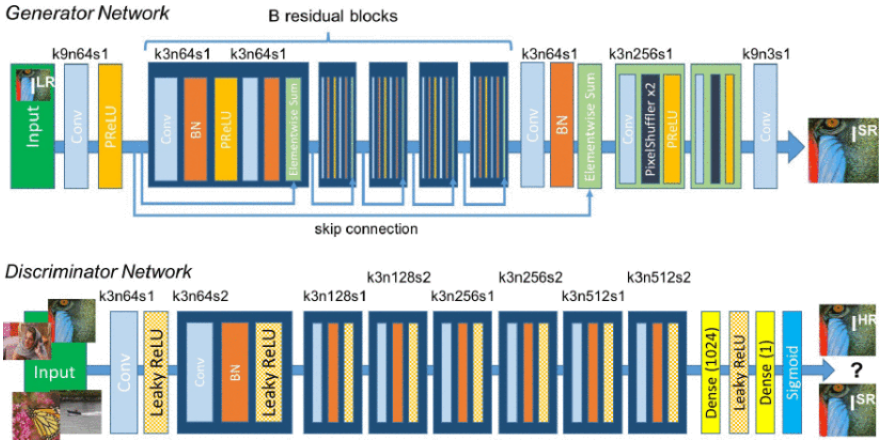

Figure 4: (Source - Ledig et al.[3]) Architecture of generator and discriminator network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

The pixel-wise MSE loss is calculated as:

$$l^{SR}_{MSE} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH}(I^{HR}_{x,y} - G_{\theta_G}(I^{LR})_{x,y})^2$$

This is the most widely used optimization target for image SR on which many state-of-the-art approaches have been relying. However, while achieving particularly high PSNR,solutions of MSE optimization problems often lack high frequency content which results in perceptually unsatisfying solutions with overly smooth textures. Instead of relying on pixel-wise losses they built on the ideas of using a loss function that is closer to perceptual similarity. They defined the VGG loss based on the ReLU activation layers of the pre-trained 19 layer VGG network described by Simonyan and Zisserman. $\phi_{i,j}$ indicates the feature map obtained by the j^{th} convolution (after activation) before the i^{th} maxpooling layer within the VGG19 network. Then the VGG loss is defined as the euclidean distance between the feature representations of a reconstructed image G_{θ_G} (I^{LR}) and the reference image I^{HR}:

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}}\sum_{x=1}^{W_{i,j}}\sum_{y=1}^{H_{i,j}}(\phi_{i,j}(I_{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I_{LR}))_{x,y})^2$$

Here $W_{i,j}$ and $H_{i,j}$ describe the dimensions of the respective feature maps within the VGG network.
In addition to the content losses described so far, the generative component of the GAN is also added to the perceptual loss. This is supposed to encourage the network to favor solutions that reside on the manifold of natural images, by trying to fool the discriminator network. The generative loss l^{SR}_{Gen} is defined based on the probabilities of the discriminator $D_{\theta_D}(G_{\theta_G}(I^{LR}))$ over all training samples as:

$$l^{SR}_{Gen} = \sum_{n=1}^{N} -logD_{\theta_D}(G_{\theta_G}(I^{LR}))$$

Here, $D_{\theta_D}(G_{\theta_G}(I^{LR}))$ is the probability that the reconstructed image $(G_{\theta_G}(I^{LR}))$ is a natural HR image.

3 Implementation Details

3.1 Vanilla GAN

Three-layered feed-forward networks were used to train both the discriminator and the generator in order to generate the images from the MNIST dataset. The latent space of tensor size 100×1 was chosen as the input to the generator. This network is highly inspired by the very first GAN devised by Goodfellow[1]. Adam optimizer[2] with learning rate 2×10^{-4} was used to train both the networks. Dropout(30%)[6] were used for all the layers in the discriminator network. A batch size of 100 was used to perform 119,900 training steps. The results were obtained in the form of image visualizations, no comparison was made with any other generative baseline.

3.2 SRGAN

The network architecture has already been shown in the Figure 4. Much of the training heuristics has been directly implemented from the work by Ledig et al.[3]. The deep generator network (SRResNet) has been composed of 5 residual blocks with identical layout. Specifically, I have used two convolutional layers with small 3×3 kernels and 64 feature maps followed by batch-normalization layers and ParametricReLU as the activation function. The increase in resolution of the input image is achieved by two trained sub-pixel convolution layers. To discriminate real HR images from generated SR samples I trained a discriminator network as shown in Figure 4. LeakyReLU activation ($\alpha = 0.2$) is used, avoiding max-pooling throughout the network. The discriminator network is trained to solve the maximization problem. It contains eight convolutional layers with an increasing number of 3×3 filter kernels, increasing by a factor of 2 from 64 to 512 kernels as in the VGG network. Strided convolutions are used to reduce the image resolution each time the number of features is doubled. The resulting 512 feature maps are followed by two dense layers and a final sigmoid activation function to obtain a probability for sample classification.
The loss function for training the generator was a combination of the MSE loss, VGG loss, the total variation regularization and the adversarial loss. The training was performed on 64×64 size HR patches randomly cropped from 17,125 images in the Pascal VOC2012 Dataset. The LR images were obtained by downsampling the HR images (BGR, C = 3) using bicubic interpolation with downsampling factor $r = 4$. During training, a mini-batch of 64 images was chosen for each interation. The SRResNet was trained for 100 epochs on Adam optimiser with learning rate of 10^{-4}; this pretrained network was used as the initialization for the generator network to avoid undesirable local optima. No other baseline apart from the pretrained SRResNet was used to compare the results with the SRGAN.

4 Results

4.1 Vanilla GAN

With increasing time steps the digits become clearer. The samples are obtained from fair random draws, not cherry-picked.

4.2 SRGAN

Due to the limitations of computation and time, I wasn't able to train the networks very effectively. The original work claims to have trained the networks on 135,000 images from the ImageNet dataset with a mini-

Figure 5: From left to right: (a) 2250 time steps, (b) 50500 time steps, (c) 119900 time steps

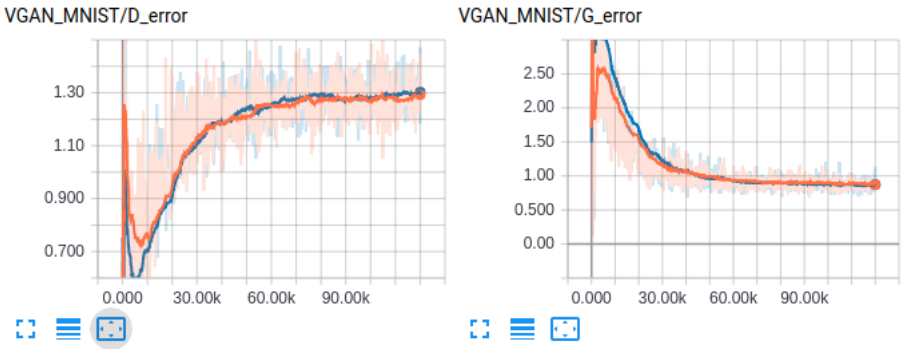

Figure 6: The plots of D-Error and G-Error across 200 epochs for two separate runs of training.

batch size of 16 images per iteration. In my implementation, I could only afford to train on 17,125 images (VOC2012) and then validate the model on the Set5 and Set14 datasets. It took $\sim$ 65 seconds per epoch to train the SRResNet for a mini-batch size of 64 images, while the SRGAN model took $\sim$ 115 per epoch for the same mini-batch size. Multi-step learning rate schedulers had to be used for faster convergence. Also, the generator used in the original work was 16 residual units deep, while I didn't make mine more than 5 units deep owing to lesser data available for training.

Figure 7: (a) Interpolated HR image, (b) Original HR image, (c) super-resolved image using SRGAN.

It's difficult to distinguish between the quality of images generated by SRGAN and SRResNet, unlike the results of the original work. Refer to Appendix for more results. The average PSNR values obtained by validating SRGAN on Set5 and Set14 are 29.05dB and 25.91dB respectively.

Figure 8: (a) Interpolated HR image, (b) Original HR image, (c) super-resolved image using SRResNet.

5 Conclusion

Through this project, I have been able to train adversarial networks successfully for a very specific use case. Although I wasn't able to achieve the best possible results, this experience provided me a hands-on experience with adversarial training. However, due to constraints in time, I couldn't train the network enough to be able to observe the effect of perceptual content loss in the VGG feature space, major highlight of the baseline research. Also, I could not explore into the problem of measuring a GAN's performance effectively.

6 Future Work

The field of AI is quite like a limitless playground, can be applied to different kinds of data, fostering ample avenues for interdisciplinary research. This idea of perceptual loss could be extended on other kinds of data from medial or satellite imaging domain. Though there is a lot of research happening in the area of deep Generative Adversarial Networks, training them effectively still remains a huge problem due to their instability. Besides, there aren't very standard metrics to gauge the performance of these adversarial networks. Hence, these are some important areas that need significant attention by AI researchers.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *arXiv*, 2014. `https://arxiv.org/abs/1406.2661`.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 2014. `https://arxiv.org/pdf/1412.6980.pdf`.

[3] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv*, 2016. `https://arxiv.org/abs/1609.04802`.

[4] Fei-Fei Li, Justin Johnson, and Serena Yeung. Cs231n lecture 13: Generative models, stanford university. *arXiv*, 2017. `http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf`.

[5] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv*, 2015. `https://arxiv.org/abs/1511.06434`.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014. `http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf`.

[7] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, and Jing-Hao Xue. Deep learning for single image super-resolution: A brief review. *arXiv*, 2018. `https://arxiv.org/abs/1808.03344`.

7 Appendix

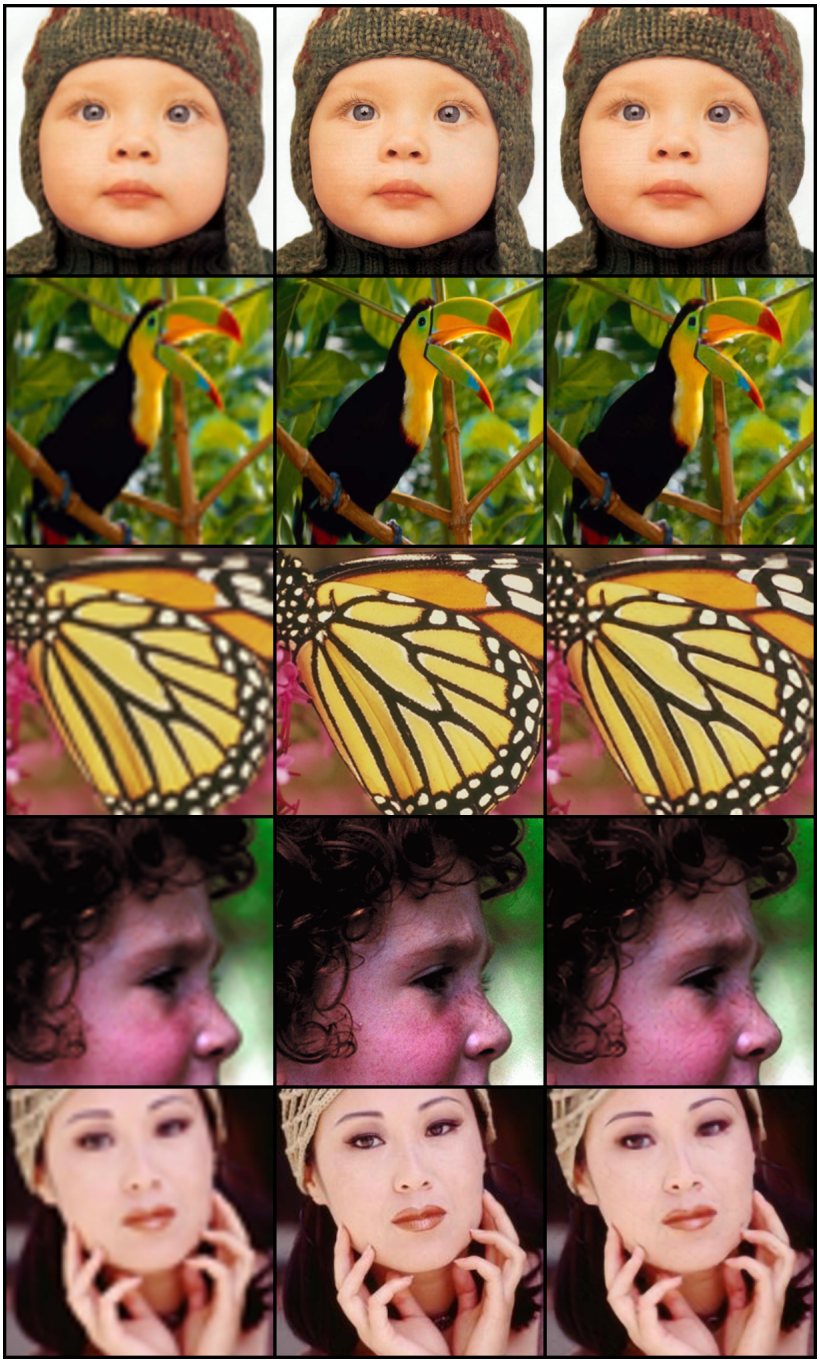

Figure 9: First column: Interpolated LR images, second column: riginal HR images,third column: super-resolved images using SRGAN after 40 epochs.

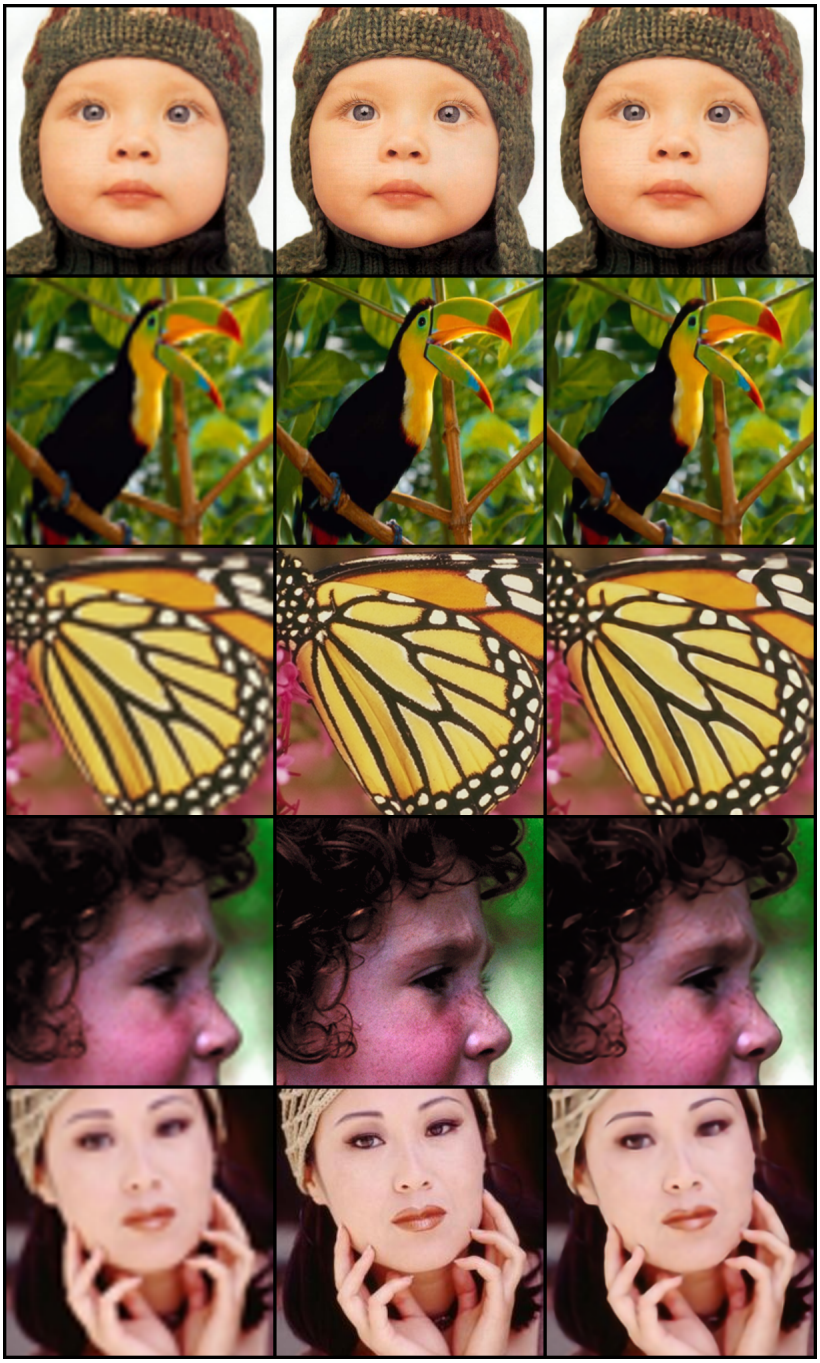

Figure 10: First column: Interpolated LR images, second column: riginal HR images,third column: super-resolved images using SRResNet after 100 epochs.