Generative Models for Image Super-Resolution

Neural Networks (EC61409): Implementation Term Project

Shikhar Mohan (18EC10054)

19 November 2021

Abstract

SISR (Single Image Super Resolution) is a fundamental low-level image processing problem where a model is tasked with recovering an HR (high resolution) image from a LR (low resolution) image. Deep generative algorithms have found considerable success in this task and in this **implementation project**, we take a look at ESRGAN (Enhanced Super-Resolution Generative Adversarial Networks) [1], an influential paper by Wang *et al.* ESRGAN is a Generative Adversarial Network (GAN) based on SRGAN [2] where they enhance three key aspects of the same, which leads to samples with consistently and considerably better visual quality and more realistic textures compared to SRGAN. Namely, they first incorporate a better Residual-in-Residual Dense Block (RRDB), remove Batch Normalization (BN), they secondly incorporate Relativistic GAN from RaGAN [3] and thirdly use features before activation to compute an improved perceptual loss. In this project we primarily aim to provide a fresh code implementation of the ESRGAN architecture to ensure research reproducibility and useful comparison, and we then provide critical commentary on the architecture as well based on its usage feasibility in general as well as compared to other architectures and its performance on different metrics as well as how it fares in the Perception-Distortion tradeoff [4].

1 Introduction

The SISR task is a low-level, fundamental and challenging task with a variety of real-world applications, including (but not limited to) medical imaging, surveillance and security. The last few years have seen significant increase in the performance of Image Super-Resolution models, both in terms of visual quality metrics as well as distortion metrics. With the advent of deep learning, the SISR task has enjoyed great improvement as a multitude of techniques have been used to solve this problem such as CNNs SRCNN [5], GANs [1, 2], attention based models such as HAN+ [6] and SAN [7] and vision transformers such as the SwinIR [8]. These networks vary greatly in their architectures, choice of loss functions and training principles.

A basic GAN framework to tackle the SISR task is relatively straightforward to concieve, since it needs only an extra discriminator network alongside the expected low resolution (LR) to high resolution (HR) network which now serves as a generator in this adversarial setup. The discriminator determines whether the HR image is drawn from the data or sampled artificially, *i.e.* from the generator network. In this project we focus on ESRGAN, which improves upon SRGAN in three major aspects. First, they introduce a new Residual-in-Residual Dense Block (RRDB) which has a larger capacity and is easier to train. Moreover, they remove Batch Normalization (BN) [9] layers as well since stochasticity introduced by the varying mean and variance of features that usually aids training in most other networks where BN is beneficial, in a generative framework it ends up introducing artefacts as analysed in [10]. Residual scaling as in [10] is used as well. Second, they improve the discriminator by borrowing from Relativistic GAN [3], which aims to ask the question *"is one image more realistic than the other"* as opposed to the traditional *"which image is fake and which is real"*. Thirdly, they use the VGG [11] Layers before activation as opposed to after in SRGAN to calculate the perceptual loss. Both of the last two changes can be seen to improve the visual quality of generated textures sizeably.

1.1 Contributions

Our main contributions in this implementation project are as follows:

1. We design a modular and well-written and documented codebase based on PyTorch. Perfor-

mance of this codebase ensures research reproducibility and provides useful comparison for benchmarking. The codebase is modular and readable, and hence can be further extended to include more SR networks.

2. We provide critical commentary on the theoretical and practical nuances of this architecture. We also analyse how it fares on the Perception vs. Distortion tradeoff.

2 Background

In this section we provide a technical glance into the SISR task's formulation, evaluation metrics used, the general adversarial framework and the ESRGAN architecture.

2.1 Task Formulation

In the SISR task, our main goal is to obtain a high resolution (HR) image from a low resolution (LR) image. More specifically, given a downsampling factor α we have two images I_{HR} and I_{LR} of dimensions $C \times \alpha H \times \alpha W$ and $C \times H \times W$, where the latter is simply obtained by applying a gaussian filter and then downsampling the former. Our task is to build a model $\mathcal{G}$ which can recover I_{HR} back from I_{SR}. Formally, we need to construct $\mathcal{G}$ such that $I_{SR} = \mathcal{G}(I_{LR})$ where I_{SR} is as close to I_{HR} as possible. For training $\mathcal{G}$, I_{HR} are available, but not during inference time.

For training this network $\mathcal{G}$, we employ a loss function $l_{SR}(I_1, I_2)$ which has lower values for I_1 and I_2 similar by some understanding. Here, for a batch (size N) of pairs of HR and LR images $(I_{HR}, I_{SR})_i$ where $0 \leq i < N$ we define $\mathcal{L}$ as follows.

$$\mathcal{L} = \frac{1}{N} \sum_{0 \leq i < N} l_{SR}(\mathcal{G}(I_{LR}), I_{HR})$$

We obtain $\mathcal{G}$ that minimizes $\mathcal{L}$, which is our desired Super-Resolution network.

2.2 Metrics

There are two families of metrics that we employ for this task. One being no-reference Visual Quality Assessment (VQA) metrics such as PI and NIQE [12] and the other being full-reference distortion metrics such as MSE or PSNR. Visual Quality Assessment metrics effectively judge how "normal" or similar to a naturally occurring image our input image is, whereas distortion metrics judge how similar to the corresponding HR image the generated SR image is. Technically, the SISR task is an ill-posed problem since there can be multiple HR images corresponding to a single SR image, which is why full-reference distortion metrics aren't the most reliable, but we need them to some extent. Blau *et al.* in [4] talk about the theoretically opposing nature of these both families of metrics, which is why we observe that even though this model is not the best PSNR performer, it excels in VQA metrics and produces realistic textures and images. Conversely it has also been observed that models that perform very well on the distortion metrics do not generate visually pleasing images.

2.3 Adversarial Framework

As defined by Goodfellow *et al.* in [13], a generic adversarial framework consists of two networks $\mathcal{G}$ and $\mathcal{D}$ with parameters $\theta_\mathcal{G}$ and $\theta_\mathcal{D}$, of the generator and discriminator respectively. The two networks are trained jointly in a min-max manner, where the discriminator $\mathcal{D}$ tries to distinguish between real images drawn from the dataset and fake images generated by $\mathcal{G}$, whereas the generator tries to generate convincing enough synthetic images to fool the discriminator. Formally speaking, the training occurs with the objective function being the following min-max problem:

$$\begin{aligned} \min_{\theta_\mathcal{G}} \max_{\theta\mathcal{D}} \mathbb{E}_{I_{HR} \sim p_{data}}[\log \mathcal{D}(I_{HR})] + \\ \mathbb{R}_{I_{LR} \sim p_\mathcal{G}}[\log(1 - \mathcal{D}(\mathcal{G}(I_{LR})))] \end{aligned} \tag{1}$$

Training GANs is notoriously difficult as there are many things which can go wrong, including (but not limited to) mode collapse and cyclicity in discriminator and generator error (*i.e.* neither of the two converge). To avoid this, extensive hyperparameter tuning is performed and many different loss functions and training schemes are incorporated.

2.4 ESRGAN

In this subsection we describe the various components of ESRGAN and explain the overall architecture. The framework of SRGAN as in figure 2 is maintained, but individual components are greatly changed as explained below.

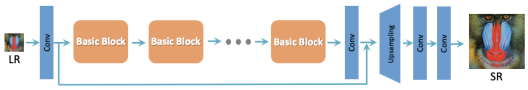


Figure 1: SRGAN generator architecture.

2.4.1 RRDB

A new residual block named RRDB is introduced, which is the same as Residual Dense Block in SRGAN except with more parameters and made easier to train. Batch Normalisation layers are removed as the regularising signal they offer usually in terms of variation in mean and variance results in visual artefacts in the final generated image. This effect is increased substantially in a GAN setup when the variation in mean and variances is high.

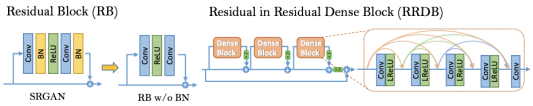


Figure 2: (Left) BN layer is removed (Right) RRDB block is used with β being the residual scaling parameter.

Residual scaling, *i.e.* scaling the residual connections with a coefficient $\beta \in [0, 1]$ to maintain stability is incorporated. Moreover, a smaller parameter initialisation is used, since it seems to make training much easier.

2.4.2 Relativistic Discriminator

A standard discriminator which tries to classify the images into the two categories of real or fake, corresponding to images drawn from the wild and images synthetically generated by a GAN, but a Relativistic Discriminator on the other hand tries to give a score for a batch of natural and synthetic images. The output of the modified objective here is closer to 1 when the fake image is more fake than the natural image and 0 otherwise. Formally,

$$\begin{aligned} \mathcal{D}_R(I_{HR}, I_{SR}) &= \sigma(C(I_{HR}) - \mathbb{E}[C(I_{SR})]) \\ \mathcal{D}_R(I_{HR}, I_{SR}) &= \sigma(C(I_{SR}) - \mathbb{E}[C(I_{HR})]) \end{aligned} \tag{2}$$

The corresponding discriminator loss here is:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}_\mathcal{R}} = &- \mathbb{E}_{x_r}[log\mathcal{D}_\mathcal{R}(x_r, x_f)] \\ &\mathbb{E}_{x_f}[log(1 - \mathcal{D}_R(x_f, x_r))] \end{aligned} \tag{3}$$

and the generator loss being:

$$\begin{aligned} \mathcal{L}_\mathcal{G} = &- \mathbb{E}_{x_f}[log\mathcal{D}_\mathcal{R}(x_r, x_f)] \\ &\mathbb{E}_{x_r}[log(1 - \mathcal{D}_R(x_f, x_r))] \end{aligned} \tag{4}$$

Note that the generator in this case benefits from the gradients obtained from both the real data being identified as real and fake being identified as fake, which is not the case in a standard GAN architecture.

2.4.3 Perceptual Loss

A pretrained VGG16 network is used, from which the feature activations from the 35th layer are obtained to create the Perceptual Loss. This is much better than using the layers after activation in VGG as done in SRGAN and allows for much more visually appealing images. This is against the general convention but it works because the features after the activation are generally very sparse and don't help much.

2.4.4 Network Interpolation

A technique with potential to remove the noise present in GAN training which leads to inefficiency is explored in this section, but not used in the final implementation due to its lack of results. A PSNR based generator is first trained, $\mathcal{G}_{PSNR}$ and a GAN based generator
$mathcalG_{GAN}$ is trained as well. As an attempt to remove the noise from GAN training, interpolation is performed between these two networks with a parameter α as described below:

$$\theta_G = \alpha\theta_G^{PSNR} + (1 - \alpha)\theta_G^{GAN} \tag{5}$$

where $\alpha \in [0, 1]$. Apart from reducing noise induced artefacts caused in GAN training, this also allows for the perception vs distortion tradeoff to be engaged with differently without having to retrain the entire network.

3 Experiments

In this section we describe the experimentation setup, such as the hyperparemeters, training schemes, training time as well as the datasets used.

3.1 Training

A scaling factor of 4 was used to obtain the downsampled images from 128×128 HR images. A bicubic kernel was used to downsample the images to form the SR set, each image of size 32×32.

The training process occurs in two phases, the first one where only the generator is trained using an Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a Cosine Annealing LR scheduler for scheduling the learning rate. In the second phase, which is the adversarial training phase an initial learning rate of 2×10^{-5} is used, which is then halved at multiple stages as LR scheduling (multiscale LR scheduling). The Generator pretraining serves to avoid undesired minima in the loss landscape during the adversarial training phase. Moreover, a pretrained Generator gives the Discriminator good super-resolved images as opposed to randomly generated black/discolored images. This lets the discriminator focus better on textures. Our reproduced code on PyTorch trains for nearly 30 hours on an Nvidia Tesla P100 16GB GPU (Google Colab Pro).

3.2 Data

We use 800 images from the DIV2K [14] dataset, which we center-crop 128×128 for our HR images and perform bicubic downsampling on the HR images to obtain the LR images. During training, incorporate random flipping and rotation data augmentation techniques to help our model learn real world structure better.

3.3 Results

The ESRGAN architecture outperformed the state of the art techniques in 2018 back when it was published and is still one of the best performers in terms of VQA metrics.

3.4 Our Implementation

Due to GPU compute insufficiencies (server timeouts on Google Colab), the entire network could not be trained but the images getting better every 5th epoch can be observed on the implementation performed. A Pytorch Lightning wrapper is used to create a GAN class wherein two instances of the Discriminator and Generator nn.Module classes are created. These instances of the networks are then used to train both the discriminator and generator jointly as described in the original ESRGAN paper. The Colab Notebook needs to be run with GPU hardware acceleration enabled, where the rest of the model's files need to be cloned from a github repository (corresponding shell instructions are embedded in the colab notebook cells). A tensorboard instance will show the desired training plots for GAN training and generator pretraining and will show the output of the test batch at every 5th epoch of training. The pytorch-lightning wrapper stores the weights at the end of every epoch in the current directory, same as where the tensorboard instance saves the desired logs.

Future work here would involve making the code even more modular, maintaining a proper documentation and adding more SISR models and datasets for useful comparison. Repositories like these help research greatly by giving scientists a single place where they can obtain all implementations from. Some examples of such repositories include Lightly-AI for Self-Supervised Learning and Detectron by Facebook for Image Detection related tasks.

4 Discussion

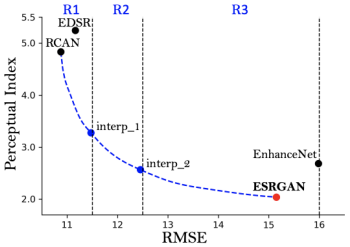
4.1 Strengths and Weaknesses

The ESRGAN framework like many other generative modelling based techniques for the SISR task performs better on VQA metrics as compared to distortion based metrics. This is true because there are multiple possible HR images for a given LR image, so even though an evaluated PSNR might be lower than that of some other architectures, it can be clearly observed the the images obtained are visually aesthetically pleasing. Moreover, this network takes very long to train, nearly 30 hours and takes significant resources to train as well as perform inference. This means that this technique is ill suited for more mobile and low cost applications. However, this is a criticism not specific at all to ESRGAN as the training time and computation cost is always high for DNNs or GANs.

4.2 Perception vs. Distortion

The ESRGAN framework is the winner of PIRM-SR challenge, which is the first challenge that evaluates networks on three regions. Performance of

these networks is plotted on a PI vs RMSE graph and the graph is divided into three regions corresponding to three ranges in the RMSE, and the lowest PI for each range is the winner for that particular region. This is done so because according to the influential work done by Blau *et al.* in [4], VQA and distortion metrics work in opposing directions of each other. It is easy to see where ESRGAN lies in on this graph in figure (2).



Results on PIRM self_val dataset

Method	PI	RMSE
ESRGAN	**2.040**	15.15
interp_2	2.567	12.45
EnhanceNet	2.688	15.99
interp_1	3.279	11.47
RCAN	4.831	10.87
EDSR	5.243	11.16

Figure 3: Perception-distortion plane on PIRM self validation dataset.

The ESRGAN architecture interfaces with the Perception vs. Distortion tradeoff by giving a major preference to Perception while maintaining distortion in manageable levels.

5 Conclusion

In this project we successfully design a codebase for the SISR task GAN architecture ESRGAN. This well functioning codebase improves research reproducibility and provides useful comparison for future research. This codebase is written using the latest relevant libraries, is readable and modular and can hence be extended to other SISR algorithms to build a reliable repository of the same. Moreover, we provide critical commentary on the theoretical and practical nuances of this technique, namely its inapplicability in more mobile form factors of application and its preference towards perception in the perception vs. distortion tradeoff, which leads to better performance in the VQA metrics and more visually pleasing generated super-resolved images.

References

[1] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European conference on computer vision (ECCV) workshops*, pp. 0–0, 2018.

[2] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," 2017.

[3] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard gan," *arXiv preprint arXiv:1807.00734*, 2018.

[4] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

[5] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," 2015.

[6] B. Niu, W. Wen, W. Ren, X. Zhang, L. Yang, S. Wang, K. Zhang, X. Cao, and H. Shen, "Single image super-resolution via a holistic attention network," in *European Conference on Computer Vision*, pp. 191–207, Springer, 2020.

[7] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11065–11074, 2019.

[8] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "Swinir: Image restoration using swin transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1833–1844, 2021.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[10] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 136–144, 2017.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[12] A. Mittal, R. Soundararajan, and A. C. Bovik, "Making a "completely blind" image quality analyzer," *IEEE Signal processing letters*, vol. 20, no. 3, pp. 209–212, 2012.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[14] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1122–1131, 2017.