Super Resolution

Group OptiCode

Department of Electronic and Telecommunication Engineering

University of Moratuwa

Name	Index Number
YALEGAMA M.M.O.A.B.	200740V
INDRAPALA S.A.	200232P

This report is submitted as a partial fulfilment of the module

EN3160 – Fundamentals of Image Processing and Machine Vision

Github Repository: *https://github.com/oshanyalegama/EN3610-Assignment-04.git*.

01st September 2023

Contents

1 *Abstract*

This report details our proposed solution for the Nitre 2018 challenge. The challenge consists of producing SR images for the LR images provided by the DIV2K dataset. The dataset consists of 4 tracks, LR bicubic, realistic mild, realistic difficult and realistic wild, named Tracks 1,2,3, and 4 respectively. WDSR model was used for Track 1 and EDSR model with pretrained weights was used for Tracks 2 and 3. Track 4 was not completed due to insufficient computing resources. Experiments were conducted to test the models' performance for each track. The models produced satisfactory results achieving beyond 24 dB PSNR for Track 1 and beyond 18dB PSNR for Tracks 2 and 3. Further improvements are considered in the discussion.

2 Introduction

The Super Resolution project task is based on the NTIRE 2018 Challenge on Single Image Super-Resolution which aims to restore rich details in a low resolution image. This challenge consists of 4 tracks.

1. Track 1 (Classic Bicubic $\times 8$): This track uses bicubic downscaling with factor $\times 8$.
2. Track 2 (Realistic Mild $\times 4$): In this track, each ground truth image is downgraded ($\times 4$) to LR images. The degradation operators are the same within each image space and for all the images in train, validation, and test sets.
3. Track 3 (Realistic Difficult $\times 4$): This track is similar to Track 2 but with a stronger degradation.
4. Track 4 (Realistic Wild $\times 4$): This track is similar to Tracks 2 and 3. The degradation operators are the same within an image space but different from one image to another.

The challenge uses the DIV2K Dataset. This dataset has 1000 DIVerse 2K resolution RGB images with 800 for training, 100 for validation and 100 for testing purposes.

3 Related Work

Some of the earliest methods of solving Super Resolution was to use interpolation technqiues based on sampling theory. However, those methods show inaccuracies when predicting detailed textures. Some studies have also used natural image statistics as a solution.

Advanced methods however, try to map functions between low resolution and high resolution image pairs. These methods rely on techniques such as neighbour embedding, sparse coding, or clustering patch spaces to learn corresponding functions.

Recently, deep neural networks, particularly CNNs, have significantly improved Single Image Super Resolution. For example, residual networks, allow for much deeper network structures and achieve better results. It demonstrated that techniques like skip connections and recursive convolution help alleviate the challenge of preserving identity information in the super resolution process. Similarly, image restoration using encoder decoder networks and symmetric skip connections lead to faster and improved convergence in the restoration process.

Many super-resolution algorithms in deep learning first use bicubic interpolation to increase the size of the input image before feeding it into the network. Alternatively, some methods incorporate upsampling modules at the end of the network, which reduces computational requirements without sacrificing model capacity. However, this approach struggles with handling multi-scale problems within a single framework. This issue is addressed by EDSR by leveraging the relationships between learned features for each scale and proposing a new multi-scale model that efficiently generates high reso-

lution images for various scales. Here a training method that employs multiple scales for both single- and multi-scale models has been developed.

4 Methodology

Our method mainly used two models. For track 1 (x8 bicubic interpolation) we used Wide Activation for Efficient and Accurate Image Super-Resolution(WDSR) and for track 2 (realistic mild) and 3 (realistic difficult) we used EDSR (Enhanced Deep Residual Network).

EDSR is essentially a Residual Neural Network used for Super Resolution (SR). The difference is that in this network intermediary blocks used for batch normalization are removed which would otherwise reduce the flexibility of the model due to normalization of the features. This increases performance and also training capacity due to less complex architecture.

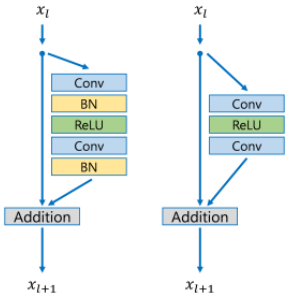

Figure 1: SRResNet vs EDSR

We were able to train the model faster by using pretrained weights for the model trained on x4 bicubic LR images. It is also worth noting that this model supports multiple image scales.

WDSR is very similar to the EDSR baseline model. WDSR reduces the width of feature maps before ReLU function in each residual block which was shown to increase the performance on Track 1 dataset. The exclusion of some convolution layers from the EDSR model also made it possible to increase the performance and speed of the model.

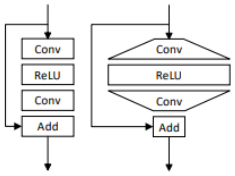

Figure 2: EDSR vs WDSR

We chose to train our models using L1 loss instead of L2. This is not only because it generally maximizes PSNR but it also provides better convergence than L2.

5 Diagrams and Explanations

The proposed models were run for 30,000 steps and their variation in loss and PSNR calculated for a subset of the validation set is as follows.

5.1 Track 1

```
1000/30000: loss = 13.889, PSNR = 23.997782 (152.74s)
2000/30000: loss = 10.744, PSNR = 24.195242 (88.02s)
3000/30000: loss = 10.500, PSNR = 24.282177 (89.01s)
4000/30000: loss = 10.312, PSNR = 24.357700 (88.99s)
5000/30000: loss = 10.105, PSNR = 24.398777 (89.17s)
6000/30000: loss = 10.135, PSNR = 24.444382 (88.76s)
7000/30000: loss = 10.028, PSNR = 24.430384 (88.93s)
8000/30000: loss = 10.063, PSNR = 24.443981 (89.26s)
9000/30000: loss = 9.888, PSNR = 24.469423 (90.08s)
10000/30000: loss = 9.818, PSNR = 24.493036 (89.69s)
11000/30000: loss = 9.978, PSNR = 24.515066 (90.31s)
12000/30000: loss = 9.799, PSNR = 24.545444 (89.32s)
13000/30000: loss = 9.865, PSNR = 24.536831 (88.91s)
14000/30000: loss = 9.697, PSNR = 24.515274 (89.19s)
15000/30000: loss = 9.830, PSNR = 24.536041 (89.00s)
16000/30000: loss = 9.744, PSNR = 24.582371 (89.41s)
17000/30000: loss = 9.781, PSNR = 24.573681 (89.14s)
18000/30000: loss = 9.755, PSNR = 24.558270 (88.82s)
19000/30000: loss = 9.765, PSNR = 24.571535 (88.74s)
20000/30000: loss = 9.849, PSNR = 24.575314 (88.73s)
21000/30000: loss = 9.667, PSNR = 24.593439 (88.59s)
22000/30000: loss = 9.750, PSNR = 24.576242 (88.81s)
23000/30000: loss = 9.680, PSNR = 24.605598 (88.77s)
24000/30000: loss = 9.738, PSNR = 24.633823 (89.32s)
25000/30000: loss = 9.613, PSNR = 24.596094 (88.81s)
26000/30000: loss = 9.630, PSNR = 24.577185 (89.04s)
27000/30000: loss = 9.610, PSNR = 24.584637 (88.82s)
28000/30000: loss = 9.611, PSNR = 24.604729 (88.99s)
29000/30000: loss = 9.673, PSNR = 24.623158 (89.02s)
30000/30000: loss = 9.607, PSNR = 24.611004 (88.63s)
```

Figure 3: Performance on Track 1 (x8 bicubic)

Figure 4: Track 1 Before and after Super Resolution

5.2 Track 2

```
hey pal
1000/30000: loss = 19.621, PSNR = 19.991533 (75.10s)
2000/30000: loss = 19.463, PSNR = 20.035566 (46.12s)
3000/30000: loss = 19.438, PSNR = 19.941133 (45.83s)
4000/30000: loss = 19.378, PSNR = 19.944569 (45.25s)
5000/30000: loss = 19.287, PSNR = 19.880016 (44.86s)
6000/30000: loss = 19.292, PSNR = 19.917753 (44.82s)
7000/30000: loss = 19.272, PSNR = 19.853485 (44.56s)
8000/30000: loss = 19.152, PSNR = 19.911205 (44.14s)
9000/30000: loss = 19.321, PSNR = 19.950693 (44.34s)
10000/30000: loss = 19.246, PSNR = 19.981686 (44.20s)
11000/30000: loss = 19.248, PSNR = 19.936172 (44.07s)
12000/30000: loss = 19.079, PSNR = 19.926788 (43.85s)
13000/30000: loss = 19.140, PSNR = 19.908270 (43.52s)
14000/30000: loss = 19.117, PSNR = 19.922337 (43.66s)
15000/30000: loss = 19.207, PSNR = 19.978611 (43.37s)
16000/30000: loss = 19.153, PSNR = 19.916180 (42.76s)
17000/30000: loss = 19.144, PSNR = 19.861607 (44.98s)
18000/30000: loss = 19.095, PSNR = 19.933353 (44.25s)
19000/30000: loss = 19.096, PSNR = 19.909576 (43.62s)
20000/30000: loss = 19.207, PSNR = 19.981756 (43.06s)
21000/30000: loss = 19.059, PSNR = 19.919460 (43.11s)
22000/30000: loss = 19.141, PSNR = 19.992916 (42.99s)
23000/30000: loss = 19.149, PSNR = 19.951080 (43.19s)
24000/30000: loss = 19.119, PSNR = 20.005999 (43.57s)
25000/30000: loss = 19.116, PSNR = 19.974634 (42.88s)
26000/30000: loss = 19.134, PSNR = 19.988174 (42.59s)
27000/30000: loss = 19.127, PSNR = 19.863209 (42.05s)
28000/30000: loss = 19.007, PSNR = 19.974281 (42.66s)
29000/30000: loss = 19.041, PSNR = 19.778862 (42.41s)
30000/30000: loss = 19.009, PSNR = 19.914282 (42.28s)
```

Figure 5: Performance on Track 2 (mild)

Figure 6: Track 2 Before and after Super Resolution

5.3 Track 3

```
1000/30000: loss = 19.091, PSNR = 20.106394 (72.16s)
2000/30000: loss = 18.729, PSNR = 20.256115 (44.38s)
3000/30000: loss = 18.873, PSNR = 20.158848 (44.29s)
4000/30000: loss = 18.784, PSNR = 20.233143 (44.11s)
5000/30000: loss = 18.515, PSNR = 20.229769 (43.69s)
6000/30000: loss = 18.468, PSNR = 20.246822 (43.70s)
7000/30000: loss = 18.550, PSNR = 20.134329 (43.83s)
8000/30000: loss = 18.694, PSNR = 20.150806 (43.70s)
9000/30000: loss = 18.545, PSNR = 20.157576 (43.81s)
10000/30000: loss = 18.611, PSNR = 20.159012 (43.88s)
11000/30000: loss = 18.486, PSNR = 20.235878 (43.93s)
12000/30000: loss = 18.440, PSNR = 20.140896 (43.84s)
13000/30000: loss = 18.376, PSNR = 20.046175 (43.85s)
14000/30000: loss = 18.379, PSNR = 20.172649 (42.46s)
15000/30000: loss = 18.297, PSNR = 20.212688 (42.30s)
16000/30000: loss = 18.338, PSNR = 20.063261 (42.37s)
17000/30000: loss = 18.344, PSNR = 20.258432 (42.29s)
18000/30000: loss = 18.302, PSNR = 20.203945 (41.73s)
19000/30000: loss = 18.475, PSNR = 20.104092 (42.29s)
20000/30000: loss = 18.432, PSNR = 20.155394 (42.25s)
21000/30000: loss = 18.207, PSNR = 20.240499 (41.97s)
22000/30000: loss = 18.305, PSNR = 20.045643 (43.74s)
23000/30000: loss = 18.227, PSNR = 20.228439 (41.07s)
24000/30000: loss = 18.371, PSNR = 20.228603 (42.20s)
25000/30000: loss = 18.396, PSNR = 20.117878 (42.35s)
26000/30000: loss = 18.218, PSNR = 20.185062 (43.42s)
27000/30000: loss = 18.279, PSNR = 20.208668 (43.31s)
28000/30000: loss = 18.351, PSNR = 20.307934 (43.69s)
29000/30000: loss = 18.340, PSNR = 20.175758 (41.64s)
30000/30000: loss = 18.279, PSNR = 20.328146 (43.17s)
```

Figure 7: Performance on Track 3 (difficult)

Figure 8: Track 3 Before and after Super Resolution

6 Results

The metric used to calculate the performance of each track is PSNR (Peak Signal to Noise ratio) relative to the corresponding high resolution image and is calculated using the equation,

$$PSNR = 10 \log_b (\frac{R^2}{MSE})$$

Where R = 255 and MSE is the mean square error between the SR image and High Resolu-

tion image. The PSNR is taken as the mean value for each image in the validation set.

Track	Model	PSNR (dB)
1	WDSR	23.60
2	Pretrained EDSR	18.79
3	Pretrained EDSR	19.32

7 Discussion

Track 1 gives visually pleasing results when the model is trained for a considerable number of steps as can be seen from the experiments. Track 2 and 3 however do not produce some of the finer textures and details with the same level of detail as Track 1 does. It's loss also doesn't reduce as significantly with the number of training steps. Future improvements of the model could attempt to solve this by using preprocessing to reduce some of the noise and blur in the image.

The two models train very fast for all tracks, never taking more than 3 minutes to train for 1000 steps. Memory allocation required was also not very significant taking less than 8 GB of memory at peak usage.

Track 4 (realistic wild) however, was not run by our models due to it demanding a higher memory allocation than that was available through our resources. Attempts at caching the dataset also produced no solution. In future improvements we hope to optimize the training pipeline, reduce model architecture complexity and utilize resources with higher performance so that Track 4 could also be supported.

All models were run using Kaggle, with a CPU of 30 GB RAM and utilizing the P100 GPU with 16 GB GPU memory.

8 Acknowledgements

We would like to express our gratitude to Dr. Ranga Rodrigo for his invaluable guidance and supportfor this module. His expertise greatly enriched our understanding of the subject matter. We are also thankful to Mr. Tharindu for his dedicated assistance throughout the course. Their contributions were instrumental in the completion of this report.

9 Conclusion

In this report, we deliberate our results on training two models, EDSR and WDSR to produce super resolution images from low resolution images of the DIV2K dataset. We demonstrate with experiments that WDSR shows visually pleasing and satisfactory results for Track 1, and EDSR shows satisfactory results with Tracks 2 and 3 with a weakness in producing finer textures and details. The performance requirements for track 4 are too demanding to be tested with our resources and were not included in the experiments. Overall the two models provided satisfactory results for the completion of the challenge.

10 References

Lim, B. et al. (2017) Enhanced deep residual networks for single image Super-Resolution, arXiv.org. Available at: *https://arxiv.org/abs/1707.02921*

Yu, J. et al. (2018) Wide activation for efficient and accurate image Super-Resolution, arXiv.org. Available at: *https://arxiv.org/abs/ 1808.08718*

NTIRE 2018 Challenge on Single Image Super-Resolution: Methods and Results. Available at: *https://NTIRE-2018-Challenge-CVPR-2018-paper.pdf*

Krasserm KRASSERM/super-resolution: Tensorflow 2.x based implementation of EDSR, WDSR and SRGAN for single image super-resolution, GitHub. Available at: *https://github.com/krasserm/super-resolution*