Selected Topics in Visual Recognition using Deep Learning
HW4 Report: Image Super Resolution

310706004 Hsin-Ying Lien

I. Introduction

In this assignment, I have tried to perform image super resolution with several models, such as VDSR, EDSR...etc. I finally used "super-image" library, which provides state-of-the-art image super resolution models for PyTorch, to train EDSR model with proper hyperparameters setting to accomplish the image super resolution task on the given high-resolution image dataset. It reached a final peak signal to noise ratio about 28.1565 on the test low-resolution image dataset.

Code GitHub link: https://github.com/angelalien/VRDL-HW4
Dataset: 291 high-resolution training images, 14 low-resolution testing images

II. Methodology

I finally implemented EDSR model to perform image super resolution by referring to the training step on the official website of super-image library.

1. Data Pre-process

For image file preparation:

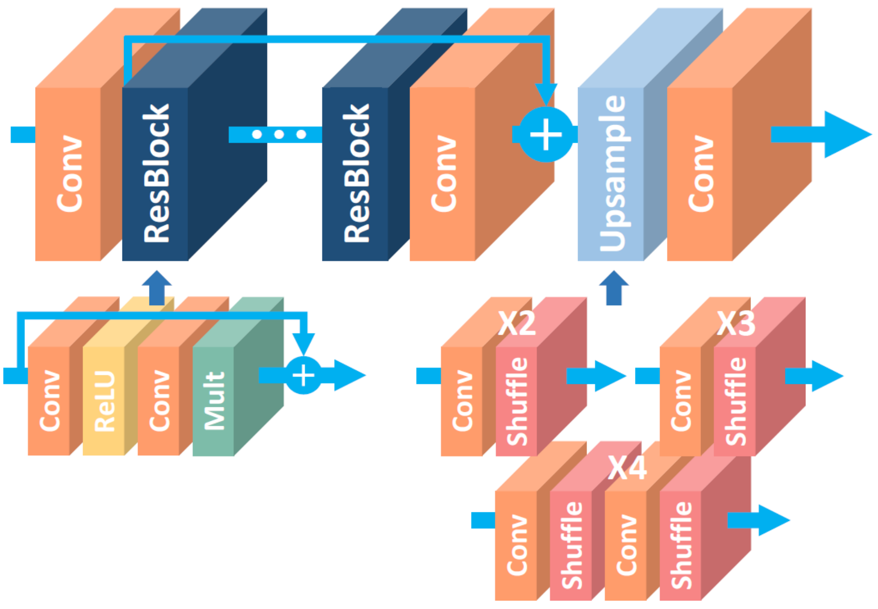
1) Download the training dataset from Google Drive, and split it into training dataset and validation dataset with the proportion of 9:1.
2) Create low-resolution images by resizing the original high-resolution images into 1/3 (equal to 1/upscaling factor)
3) Save both low- and high-resolution image file directories into the dictionary for building the training and validation dataset.

For data augmentation:

1) Crop the given training image into four corners and the central crop, with the size of 1/2 height and 1/2 width.
2) For each cropped image, resize it to the original scale for high-resolution image, and resize it to the 1/3 scale for low-resolution image with bicubic interpolation.
3) For each resized image, randomly crop it with patch size of 12.
4) Horizontally and vertically flip the image randomly.
5) Rotate the image by angle of 90.

2. Model Architecture

EDSR (enhanced deep super-resolution network) is a model that uses both deeper and wider architecture (32 ResBlocks and 256 channels) to improve performance. It uses both global and local skip connections, and up-scaling is done at the end of the network. It doesn't use batch normalization layers (input and output have similar distributions, normalizing intermediate features may not be desirable) instead it uses constant scaling layers to ensure stable training. An L1 loss function (absolute error) is used instead of L2 (MSE), the authors showed better performance empirically and it requires less computation.



3. Hyperparameters

I set the hyperparameters by referring to the paper "Enhanced Deep Residual Networks for Single Image Super-Resolution".

n_resblocks = 32 (number of residual blocks)
n_feats = 256 (number of filters)
res_scale = 0.1 (residual scaling)
patch_size = 12 (the size I set for randomly crop the image)
num_train_epochs = 300

III. Experimental Results and Observations

Model	Hyperparameters	Pre-process	PSNR
EDSR	n_resblocks=32, n_feats=256, res_scale=0.1, patch_size=12, epoch=100	five crop, random flip/rotate	27.8935
	n_resblocks=32, n_feats=256, res_scale=0.1, patch_size=12, epoch=150	five crop, random flip/rotate	28.0049
	n_resblocks=32, n_feats=256, res_scale=0.1, patch_size=12, epoch=200	five crop, random flip/rotate	28.0898
	n_resblocks=32, n_feats=256, res_scale=0.1, patch_size=12, epoch=300	five crop, random flip/rotate	28.1565
MDSR	patch_size=12, epoch=100	five crop, random flip/rotate	27.2224
AWSRN	bam=true, n_resblocks= 4, n_feats=32, block_feats=128, n_awru=4, epoch=100	five crop, random flip/rotate	26.6837

I have tried to train different models and adjust the hyperparameters setting to reach a higher performance. During the process, I got some observation results.

1) EDSR model outperforms other models
   The models I have tried include VDSR, EDSR, MDSR, MSRN, and AWSRN. I finally got the best performance with EDSR, which removes the batch normalization layers. MSRN and AWSRN doesn't performs well, maybe due to the hyperparameter setting. In MDSR, pre-processing modules are located at the head of networks to reduce the variance from input images of different scales, which is useful for multi-scale training but doesn't help the performance.
2) The enhanced FiveCrop method improves the performance so much
   After I implemented the process of cropping and resizing the image, I got much better performance with the PSNR increasing from 26.6484 to 27.8722. I think that it is because the training image data is five times the size of the original data, which augments the information for model learning.
3) Some methods of data augmentation don't help so much
   When I was training VDSR model, the results didn't become much better after adding gaussian noise, randomly flipping and rotating images. The PSNR just increases by about 0.05~0.1.

4) FiveCropped Images mix original images doesn’t help
   I originally increased the training data with the enhanced FiveCrop method mentioned above. I also tried to add the original uncropped images into the training data, but the result didn’t become better. I think that it is because the images added is duplicate with the cropped images, there is no new information provided.
5) Increase the model architecture scale makes the result a little better
   When training EDSR model, the PSNR of results increases by about 0.1 after I expand the model by increasing the number of residual blocks and filters, and set residual scaling as 0.1 for the computational efficiency.
6) SGD optimizer may be better than Adam in VDSR
   When training VDSR model, I find that the performance is a little better when I use SGD optimizer with learning rate of 0.001 instead of Adam optimizer with learning rate of 0.1. The PSNR of images increases by about 0.1.
7) The proportion of train-test split has little influence
   I have tried the proportion of 8:2 and 9:1. The latter one only increases the PSNR of images by 0.02, so I think that it is probably just a little variation.

IV. Summary

Nowadays, it’s easy for us to implement several models to deal with image super resolution tasks by referring to some websites or using existing tools. It is convenient to train model by using existing library and adjusting hyperparameters, but less flexible to change the model architecture. We can also refer to others’ GitHub to build our model more flexibly.

In this assignment, I find that the data augmentation plays an important role for the training results. We can expand the data augmentation method based on the original function from the existing libraries to increase more information and improve the training results. Besides, proper choose of model and some hyperparameter setting also help improve the performance. We can refer to some advices from papers, and make some experiments to find the most proper setting for our dataset. As a result, data augmentation and proper setting of hyperparameter will lead to better performance for image super resolution.

V. Reference

Super-Image Library
https://eugenesiow.github.io/super-image/

VDSR PyTorch Implementation
https://github.com/2KangHo/vdsr_pytorch

Enhanced Deep Residual Networks for Single Image Super-Resolution
https://openaccess.thecvf.com/content_cvpr_2017_workshops/w12/papers/Lim_Enhanced_Deep_Residual_CVPR_2017_paper.pdf