VRDL HW4 Report

310551055 鄭伯俞

Table of Contents

GitHub

Link to the repo of my code : https://github.com/Nemo1999/VRDL_HW4

Reference

- Paper that introduce SRResNet :
  Ledig, Christian et al. “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017): n. pag. Crossref. Web.

- Implementation of SRResNet :
  https://github.com/cszn/KAIR

Introduction

In homework4, we are asked to train image super-resolution (**SR**) models with an upscaling factor of 3. The quality of predicted HR images in the testing stage is measured in peak signal-to-noise ratio (**PSNR**).

PSNR is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. It is inversely proportional to **MSE reconstruction** error. As shown in the following formula.

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

The model I chose for this homework is **SRResnet** , a SR model proposed in the referenced paper. I didn't use any pretrained model, the model is trained from scratch and the final best testing PSNR is **"28.1258"**. Some predicted images in the test set are shown below.

LR image	HR image (Predicted)
	

Methodology

Data pre-process

The pre-processing of the training dataset is defined in dataset_sr.py. The pipeline is described below:

1. The images are converted to **single precision floating point numpy arrays** after being read from disks. The value for each pixel color intensity is normalized from [0,255] to [0,1] .
2. The image is **cropped to the largest possible size such that height and width is integral multiple of the scale factor** (scale factor is 3 in this homework). For example, if the original image has size 100x200, it will be cropped to 99x198, and the LR image will have dimension 33x66.
3. The LR image is obtained using bicubic interpolation implemented in KAIR_Repo/utils/utils_image.py.
4. Randomly crop a patch from both LR image and HR image on the corresponding location of the image (I use patch_size for HR =96 for my training config) . For example if we cut a 32x32 LR patch at location (11, 23) from the LR image, the corresponding HR patch will be 96x96 at location (33,69).
5. We apply random rotation (angle integral multiple of 90), and random LR flip to both LR and HR image. Same transformation is applied to LR and HR images in a pair.

At testing time, only stage 1. is applied.

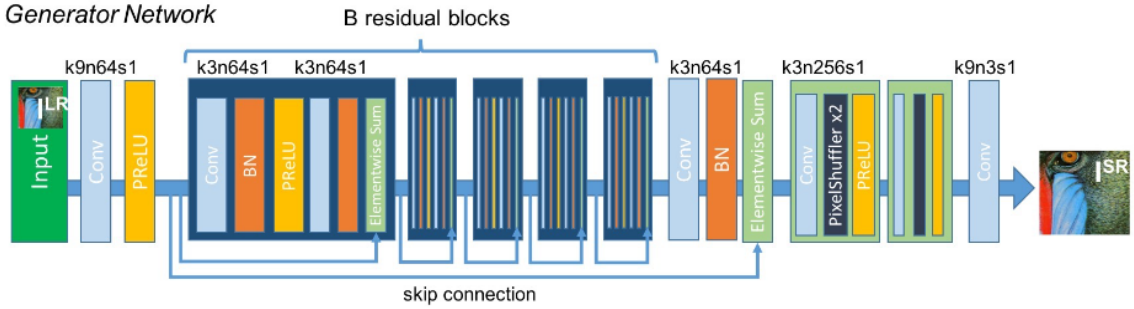
Model architecture

The model used in this homework is **msrresnet0** (which stands for Modified version of SRResNet version 0), which is implemented in KAIR project here .

The model architecture compose of 4 parts: Head→ Body → Upscale → Tail

1. Head (source)
   The head is simply a convolution layer that increases the **channel size from 3 to 64**.
2. Body (source)
   The body of msrresnet0 used comprises of **16 residual blocks**, each res block has a **Conv → ReLU → Conv** pipeline that calculates the residual, the input and output **channel sizes** for the convolution are both **64**, and the **kernel sizes are all 3x3**. The whole body is encapsulated in a **ShortcutBlock** (definition source / instantiation source) , that adds another shortcut connection from the beginning of the Body to the end of the Body.
3. Upscale (source)
   The upscale block for scale=3 is an **nearest upsampling** with scale=3 followed by a convolution and Relu block. For higher upsample scale, multiple upscale blocks are concatenated.
4. Tail (source)
   The tail block is defined as two convolution layers, the first use Relu activation, **the final layer use no bias and no activation**

The original proposed architecture for SRResNet is shown in the figure below.



Note that **MSRResNet differs from original SRResNet in the following 3 major ways:**

1. The head of MSRResNet doesn't use activation
2. Residual blocks of MSRResNet don't use batchnorm.
3. The upscaling block uses Nearest Upsampling → Conv → Relu , instead of Conv → PixelShuffling → PReLU

Hyperparameters

The hyperparameters used are defined here . I summarize the important settings below:

- Model: msrresnet0
- Loss: l1 (generate sharper image, but maybe not optimal psnr? I didn't try)
- Optimizer: adam , Lr=1e-4, Scheduler=MultiStepLR, mileStones= 100000 * [2,4,6,8,10]
- Patch_size = 96
- Batch_size = 1 (because images in training set have different sizes, batch size >1 cause error, I could have resized the images first to train faster)

Summary

In this homework, I learned to train super-resolution models, and during the process of writing this report, I practiced tracing source code of a DL project. I got down to the details of model architecture and compared the implementation to the original proposed architecture from the paper. Finally, in the hyper parameter section, I notice that I should have used L2 norm for training because it should give better PSNR results (by definition), and that I should resize the images to the same dimension to allow batch training, which should speed up the training process.