Project II: Blind Face Super-Resolution

Li Hantao, G2101725H, MSAI, hli038@e.ntu.edu.sg

I. Dataset Analysis

This project will use the data selected from the FFHQ Dataset [1]. We can only utilize the 4,000 images provided in the training set to train the network (unlike Project I, where we can use the validation set to fine-tune the model). It is crucial to analyze the target dataset for artificial intelligence tasks before we begin implementing the model. The image size in this dataset is 512. Examples are shown below.

Distinguishable from the face images in the Project I, these images do not show apparent bias on overall image brightness, contrast, and saturation are average, and there are no obvious overexposure, underexposure, and white balance offset. This may indicate that one of the augmentation methods, color jittering, will not be appropriate here anymore, which will be verified in the following section.

II. Data Augmentation

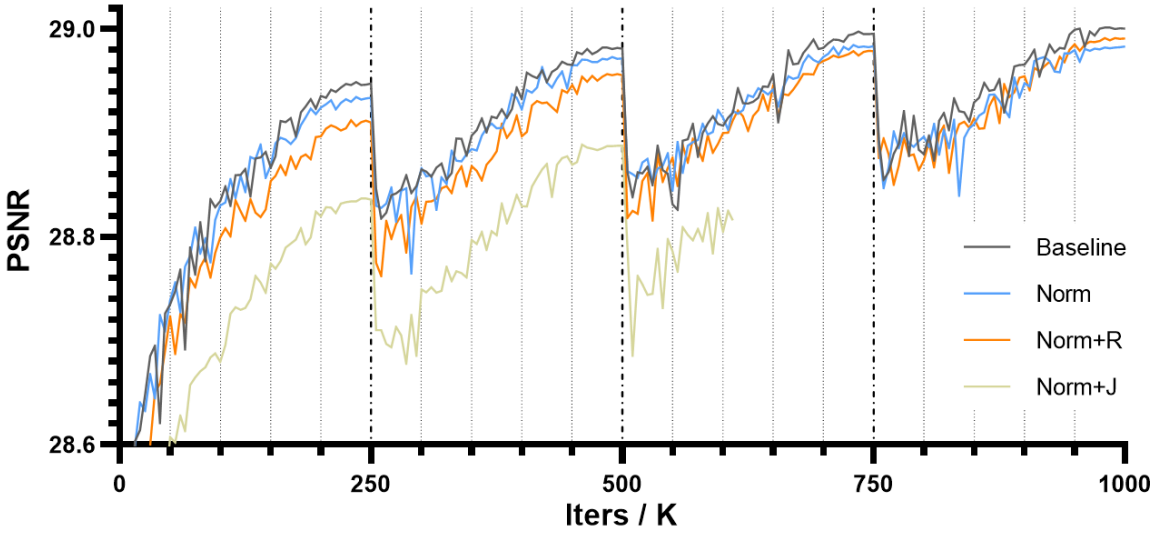
In this project, the model training pipeline is implemented by the mmediting codebase [2], which has many popular image-restoring methods with modular design and provides detailed documentation. For the circumstance that we only have 4,000 training images to train an SR model, it is essential to implement enough skills in the data augmentation pipeline to avoid overfitting as much as possible. Thus, we design a series of ablation studies and contrast experiments to find a better pipeline of data augmentation. However, due to the GPU limit, that is, the computational resource limit, we cannot run full network training many times to test the result of different data augmentation methods. In this stage, we only run the MSRResNet [3] model with 20 block depth (which is just under the restriction of parameter number) for a limited number of iterations (250k * 4) with 8 as the batch size. The loss function is L1 loss. If we notice that one method cannot make any enhancement from the training curve, the training procedure will be halted to preserve the computation resource. There is no doubt that we cannot find the authentic trend and enhancement of augmentations in such a small number of epochs, but this seems the only reasonable way for this project.

Firstly, we apply the optimal DA setting we selected in Project I (except the crop) since both two projects are training on the face image. We test the following method in a series of experiments:

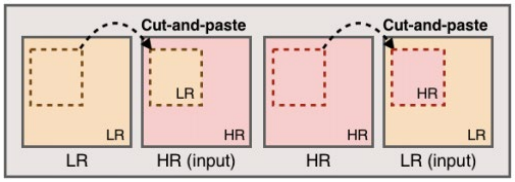
a) Normalization.
b) Random rotate the image with range in 25°, Prob = 0.5.
c) Color Jittering, brightness = 16, hue = 9, contrast and saturation range = (0.75, 1.25), Prob = 0.5.

The training curves of this series of experiments are shown below. We compare these three methods to locate the optimal one. However, distinct from other CV tasks such as segmentation and detection, we notice that normalization operation in SR tasks is harmful to the final PSNR. It can be easily explained that the consistency of RGB value in the input and output image is vital for SR tasks; thus, the normalization that will disorder the ratio of original RGB value cannot be utilized in this task. We can find that the Color Jittering (Norm+J) operation impacts the PSNR, which may be because the bias on overall image brightness, contrast, and saturation is more consistent now. The gap between w/ and w/o J is too massive to

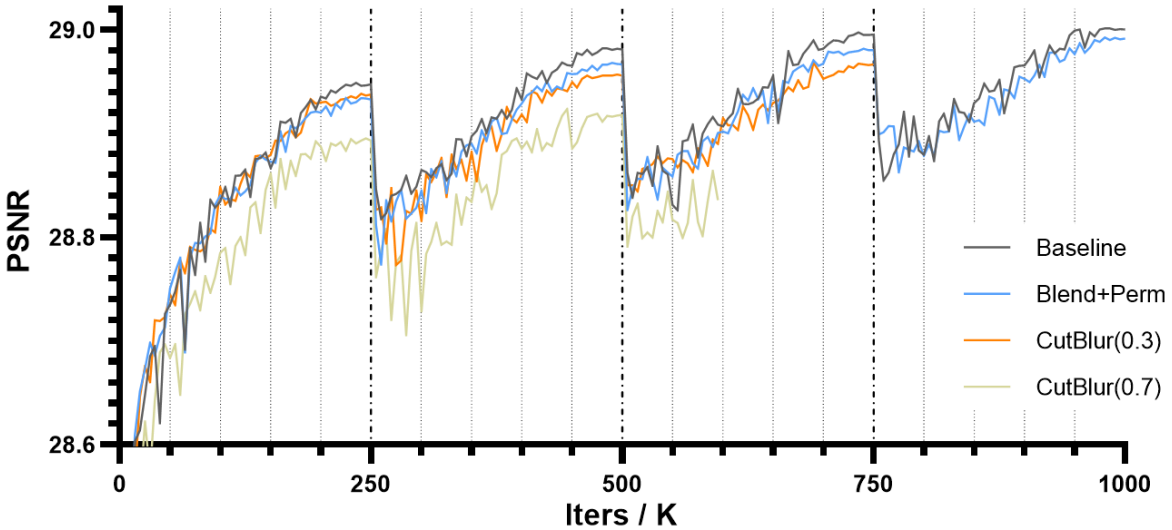
resume the training, so we suspend it after 600K iterations. The Random Rotation enhances the PSNR after 1000K iterations' training.



Secondly, we seek other approaches to DA that may improve the result. The researcher proposed an efficient pipeline for SR tasks in CVPR 2020, including CutBlur, RGB permutation, and Blend [4].

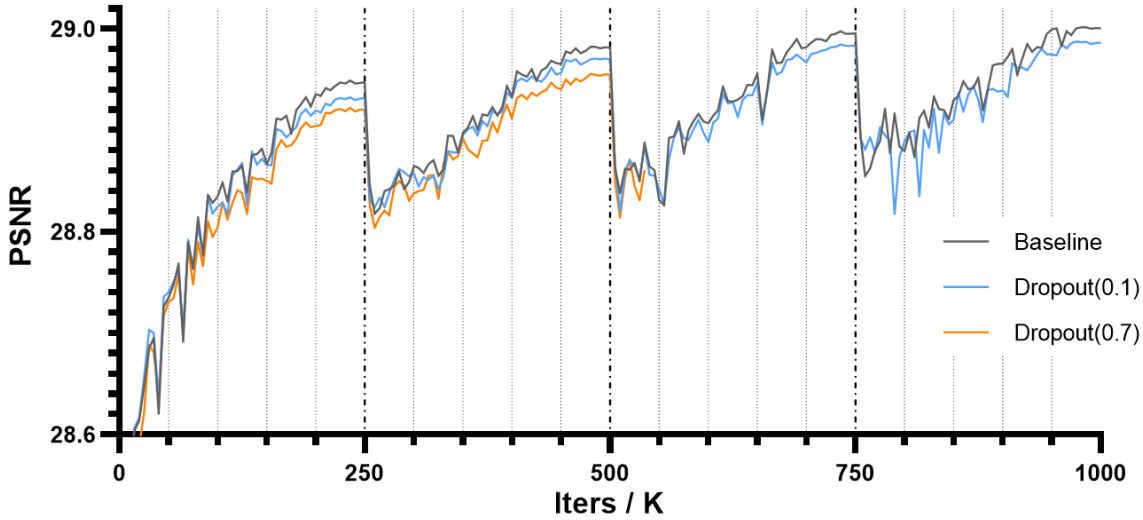


CutBlur cut and paste a low-resolution image patch into its corresponding ground-truth high-resolution image patch, as shown above. Left to right are examples of Blend, RGB Permutation, and CutBlur. We implement them with a series of experiments, where RGB permutation and Blend are achieved by color jittering functions with a single parameter of brightness and hue. The default rate α of implementing Cut Blur is 0.7, and we add an additional training with α = 0.3. The training curves are shown below.



The probabilities of Blend + Permutation and CutBlur are 0.1 and 0.2, respectively. The result of B+P shows a protentional trend to enhance the PSNR in the increased number of iterations. On the other hand, both two rates' CutBlur cannot improve the result of training. That may be because the face image in the training set is all in the middle part, while the method similar to CutMix focuses on making items in the different areas more robust. Thus, we stopped them after 750K and 600K iterations.

Thirdly, another research finds that Dropout in the last conv layer can enhance the PSNR [5]. This paper proposes two optimal dropout probabilities of 0.1 and 0.7. We test both of them, which are shown below. The training results show that the dropout cannot make improvements in this task. Our iterations may not be enough to let those DA or dropout method show their effect. However, due to the computation source limitation, the final training procedure may not be long enough as well. Thus, we have to abandon all of those unstable DA methods.

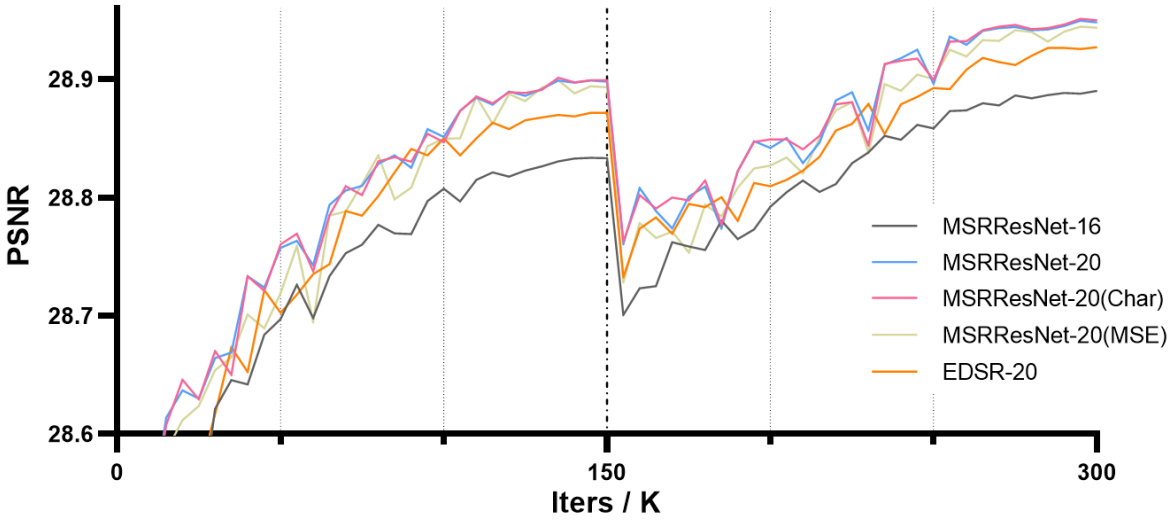


After the whole testing procedure, we decided to use the following method:

a) Random Rotate the image with range in 25°, Prob = 0.3.
b) Color Jittering, brightness =0.2, Prob = 0.08.
c) Color Jittering, hue = 0.1, Prob = 0.08.

III. Loss Function and Model

In testing loss function and model selection, we use an iteration setting of 2*150K instead of 4*250K because the effect of different losses and models does not need such a long time to show up. For the loss function selection, we tested the L1 loss, MSE loss, and Charbonnier loss similarly. For the model selection, due to the limitation of parameters, we only try the MSRResNet-16 (Baseline), MSRResNet-20, and EDSR-20 [6]. Their training curves of them are shown below. The PSNR of Charbonnier loss is slightly higher than the L1 loss. Thus, we choose the MSRResNet-20 with Charbonnier loss as our final model.



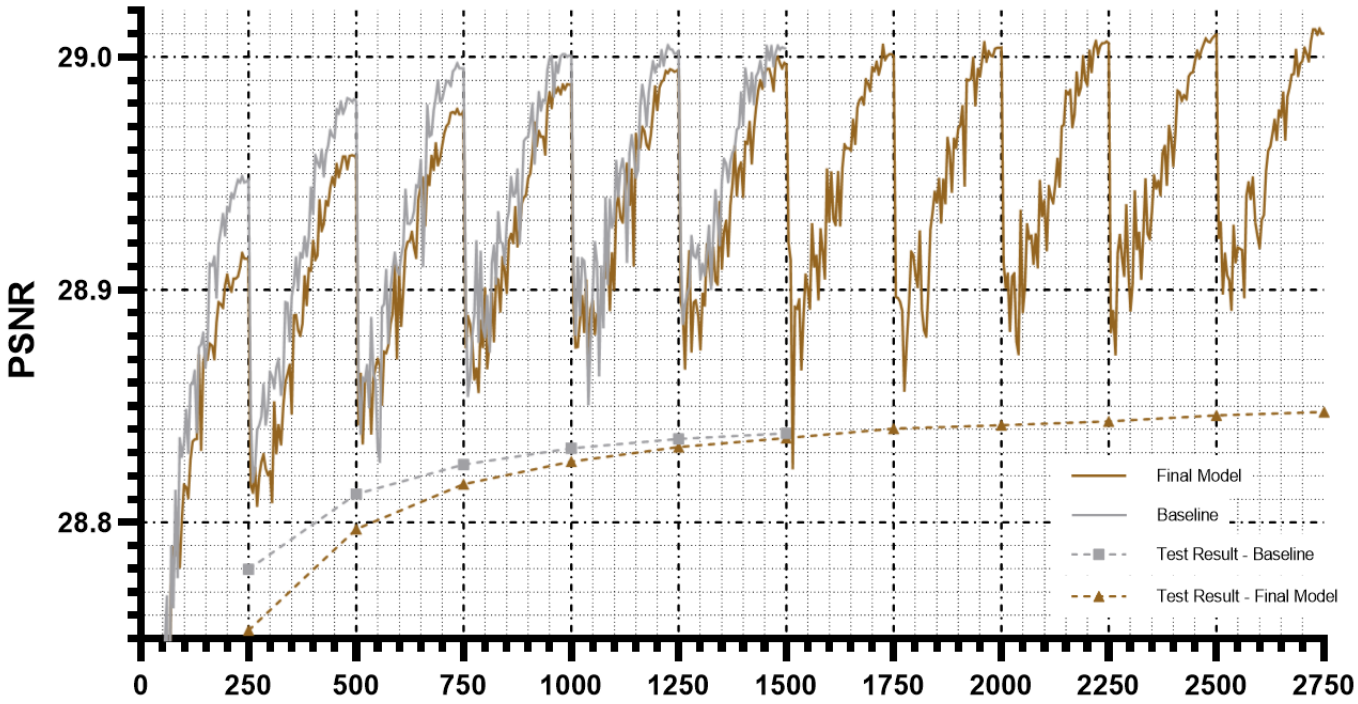
IV. Result

We complete all the training and testing procedures in the following two platforms in Table I.

TABLE I

Platform I	Platform II	Parameters
GPU: NVIDIA GeForce RTX 3090 CPU: Intel(R) Xeon(R) Gold 6330 Ubuntu~18.04 PyTorch: 1.8.1+cu111	GPU: NVIDIA TITAN Xp CPU: Intel(R) Xeon(R) CPU E5-2620 v4 Ubuntu~18.04 PyTorch: 1.8.1+cu111	Flops: 46.5 GFLOPs (For 3*128*128) Params: 1.81 M

We train the MSRResNet-20 model with 2750K as the iteration setting. To ensure the effect of data augmentation methods, we resume the baseline training parallel with the training procedure until the performance of DA is greater than the baseline. The training and testing curves (we only test the model of 2750K iterations through CodaLab) are shown below. The PSNR of baseline and the final model seem to be identical after 1500k iterations; thus, we terminate the training process of baseline at that time. The PSNR in both the validation and test set increases throughout the training procedure, while the best result is shown in the 2750k checkpoint with a **PSNR of 28.8475 on test set and 29.0122 on validation set**.



V. Conclusion

In this project, we implement a series of ablation studies and contrast experiments to find a better pipeline of data augmentation, a better loss function, and a better network backbone. However, even if we implement augmentation and dropout methods proposed in CVPR recently [4-5], none of them works. On the one hand, it may be because of the limited iteration numbers, which cannot show the effect of anti-overfitting methods. On the other hand, the PSNR result of the final model still showed a continuous and stable rise trending in the later stage of the training process; that is, the first derivative of the result did not decline, indicating that continuous training should further improve the result. Nevertheless, due to the limited GPU resources, we cannot afford a lengthier or larger cluster GPU for training. Even this training process took a GTX 3090 nearly seven days' computation. In summary, the final model can be improved efficiently by a longer training time or a larger GPU cluster.

[1] https://github.com/NVlabs/ffhq-dataset
[2] https://github.com/open-mmlab/mmediting
[3] Ledig, C., Theis, L., Huszár, F., Caballero, J., Aitken, A.P., Tejani, A., Totz, J., Wang, Z., & Shi, W. (2017). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 105-114.
[4] Yoo, J., Ahn, N., & Sohn, K. (2020). Rethinking Data Augmentation for Image Super-resolution: A Comprehensive Analysis and a New Strategy. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 8372-8381.
[5] Kong, X., Liu, X., Gu, J., Qiao, Y., & Dong, C. (2021). Reflash Dropout in Image Super-Resolution. *ArXiv, abs/2112.12089*.
[6] Lim, B., Son, S., Kim, H., Nah, S., & Lee, K.M. (2017). Enhanced Deep Residual Networks for Single Image Super-Resolution. 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 1132-1140.