**Summary**

Under the existing hardware conditions, we built a software platform using pytorch learning framework according to the requirements of the organizing committee. And complete the task of single picture super-resolution on this platform.

Firstly, we tested the effect and training time of classical models such as ESRGAN and DBPNGAN. We find that ESRGAN is effective, but it takes a long time. So we add Weight Normalization and adjust batch operation on ESRGAN to reduce the convergence time of ESRGAN to 3 days. After that, we did a series of experiments on the original model, such as replacing RRDB and modifying residual path.

1. **Environment**
   1. **Hardware and Software Platform**

| CPU | Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz |
| --- | --- |
| GPU | Tesla K40c |
| Memory | 23G |
| System | Centos 7.2.1511 |
| CUDA / cudnn | CUDA 9.0.176 / cudnn 7.4.2 |
| python | Python 3.6.5 :: Anaconda |
| pytorch | 1.0.0 |

Table 1-1 Hardware and Software Platform

* 1. **Train and test our model**

You can train and test the network by directly running the train.sh and test.sh files located in the script/code path.

Train: ./train.sh

Test: ./test.sh

1. Datasets

For training, we mainly use DIV2K and Flickr2K dataset, which are both 2K resolution dataset for image restoration tasks. Through training, it is found that Flickr2K data sets containing 26502K high-resolution images are helpful for convergence and training results. So we decided to use Flickr2K data set as training set and DIV2K data set as validation set.

We crop 480 × 480 input patches from HR image and its bicubic downsampled image as training output-input pairs. During the training process, we randomly flip the pictures horizontally and rotate them 90 degrees to enhance the training data set. For the test set, we used the test set provided by the PIRM-SR challenge

1. **Analysis of** Original **network**

ESRGAN is an outstanding model based on Gan to pursue image perception effect. It achieves good visual effects by using dense residual blocks, perceptual loss, and relative discriminators.

**3.1 Network Structure**

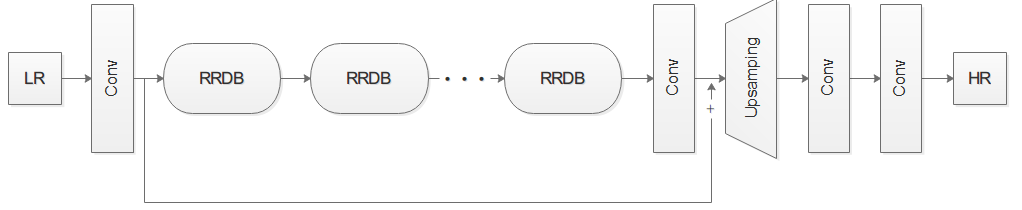


Figure 3-1-1 Generating network of original model

**Generation network**: The above figure is the generation network of ESRGAN. The LR image first passes through a convolution layer to get its feature information. Then the feature information is extracted by a residual network composed of 23 RRDBs. Finally, an HR image is obtained through a Pre-upsamping layer connected by the Upsamping layer and two Conv layers.

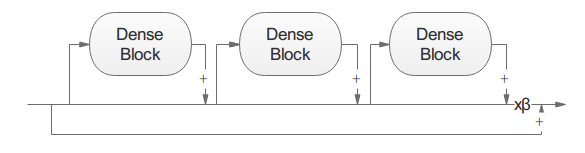
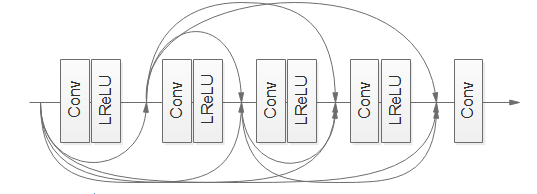
 

Figure 3-1-2 Residual in Residual Dense Block Figure 3-1-2 Dense Block

**Residual in Residual Dense Block (RRDB)**: RRDB has residual-in-residual structure and dense connection blocks. Residual-in-residual structure enables residual learning to proceed at different levels. As shown in the figure above, each RRDB contains three Dense Blocks. DenseBlock consists of densely connected five convolution layers. This connection mode improves network capacity.

**Relative discriminator**:ESRGAN uses a relative discriminator. Unlike the standard discriminator, the relative discriminator judges the probability that the true picture is more true than the false one and that the false picture is more false than the true one.

**Loss**: ESRGAN loss consists of perception loss, generation network countermeasure loss and pixel loss. Perception loss is calculated by using the feature information extracted from SRGAN network before activation and the feature information generated from the image. The loss function is expressed as follows：

|  |  |
| --- | --- |
|  |  |

In the formula,，*L*per: Perception loss,

*L*G : Generator against loss,

*L*pixel : Pixel loss

In the original model, both pixel loss and perception loss are calculated based on L1 loss. And the values of α and β are 5e-3 and 1e-2, respectively.

**3.2 The result of original model**

We trained the model for 160k with GPU K40, validated the model with our test set, and got the following result: the Perceptual index is about 2.17084.

Table 3-2-1 The result of original model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Iter | 140k | 145k | 150k | 155k | 160k |
| Pi | 2.1742 | 2.1703 | 2.1612 | 2.1703 | 2.1782 |
| Avg\_Pi | 2.17084 | | | | |

1. Optimizing Process

#### 4.1 Weight Normalization

Due to the limitation of equipment and time, the convergence rate of ESRGAN is too slow for us. Under these conditions, we added Weight Normalization to ESREGAN to speed up its training.

WN can normalize the weight, which is to decompose the weight vector into two parts: weight size and direction. The formulas for normalization (2) and gradient calculation (3) of network weight W are as follows:

|  |  |
| --- | --- |
|  |  |
|  |  |

From the above formulas, we can see that the weight W can be constrained by the size of vector V so that the training of network parameters is more robust than learning rate, so we can choose a larger learning rate to accelerate network training.

The following is a comparison of two models using WN and not using WN:

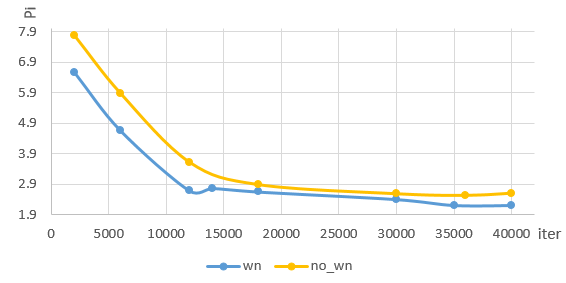


Figure 4-2-1 The comparison of two models using wn and not using wn

The learning rates of wn and no\_wn in the figure above are 1E-3 and 1E-4, respectively. (Note: When the learning rate of no\_wn is 1E-3, it can not be trained normally.)

As can be seen from the figure above, the fluctuation trend of the two models is basically the same, but the convergence effect of WN is faster. Although Weight Normalization can not directly improve the speed of model training, it can make the model have better convergence effect in the case of higher learning rate.

**4.2 Residual network structure**

**4.2.1 Residual pathway**

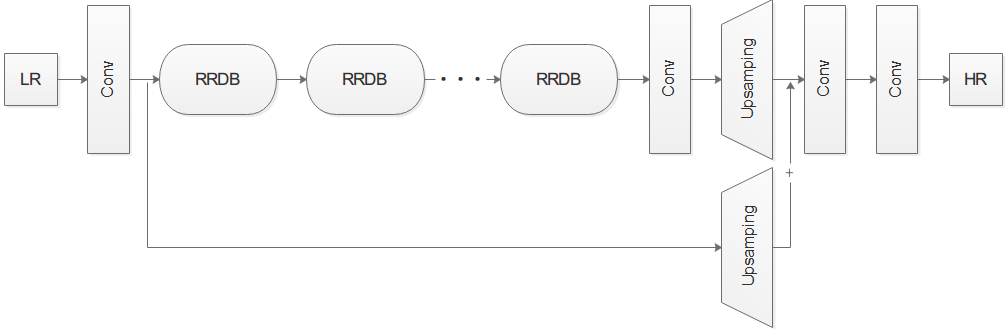


Figure 4-2-1 Pre-Upsamping-ResNet model

The original model adds the residual information directly to the LR information. In this case, the residual information is underutilized. Therefore, we changed the residual path and performed an Upsamping operation on LR information.

We trained 90K iterations with batch of 6 in size and got the result: the Perceptual index is about 2.15405.

**4.2.2 An unsuccessful attempt**

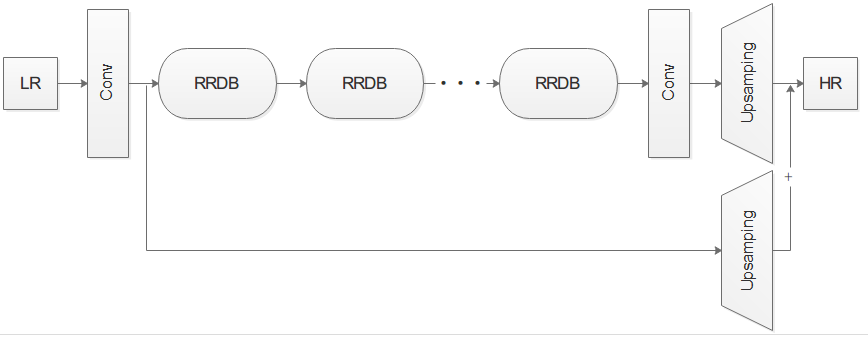


Figure 4-2-2 Model using direct-out layer

In our assumption, the high frequency information output by residual network should be as constant as possible and used more frequently. The last two convolution layers of the model are too simple compared with the complex residual network in front, and we doubt whether their advantages are stronger than their changes to high-frequency information. So we deleted two convolution layers and did a test.

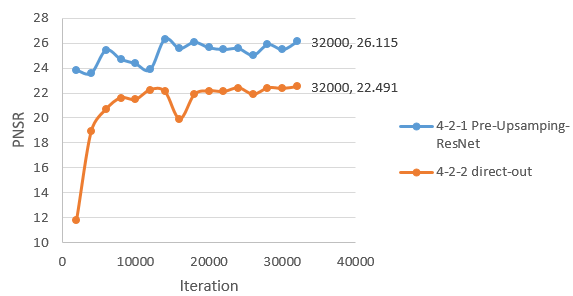


Figure 4-2-3 The result of model in figure[4-2-1] and figure[4-2-2]

As can be seen from the above figure, the results of the model were significantly worse after the output layer was changed to the direct-out layer. The Perceptual index of the model in 32K iterations is 5.7784. Compared with the model Pre-Upsamping-ResNet's result of 2.4141 times at 32K, it can be said that the effect is not ideal.

**Analysis of the Reasons for Failure**: After our discussion, using direct-out layer is just transferring the deleted two-layer convolution layer tasks to residual blocks. Compared with the improvement of speed and the utilization of high frequency information, the difficulty of model training is greatly increased. We have two solutions: 1. Increase the use of residual information. 2. Change the residual network model to make the correlation between high-frequency information and low-frequency information, that is to say, increase the error-correcting ability of the network. In the following work, we mainly focus on the second solution.

**4.3 Residual network structure**

**4.3.1 Introduction**

D-DBPN (Dense Deep Back-Projection Networks) won the first price on Region 2 in PIRM2018. The model focuses on increasing the sampling rate of SR features in different depths and distribute the tasks to calculate the reconstruction error to each stage. This schema enables the networks to preserve the HR components by learning various up- and down-sampling operators while generating deeper features.

In this study, we use the Dense Projection units of Back-projection stages in D-DBPN to replace Residual-in-Residual Dense Block (RRDB) in ESRGAN in order to gain better restoration of image features.

**4.3.2 Network architecture**

Below are two ideas we implemented:

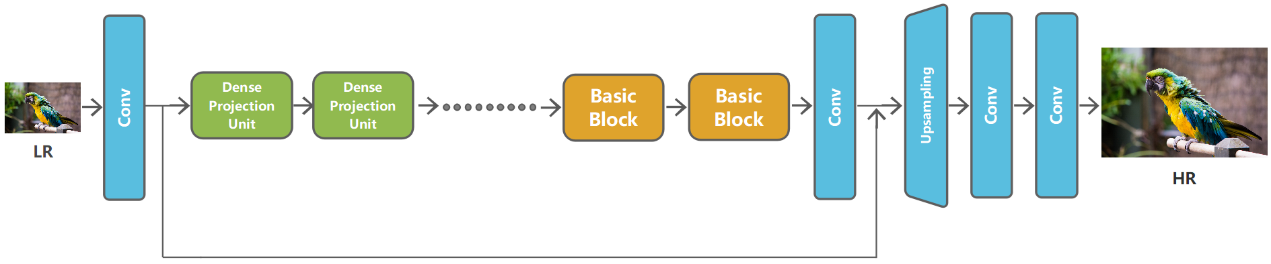


Figure 4-3-1 Modify RRDB x23 into Dense projection units x5+RRDB x13 using ESRGAN

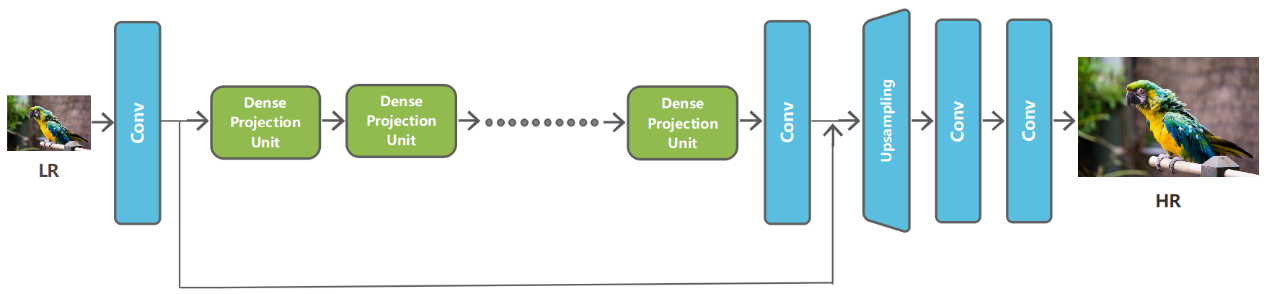


Figure 4-3-2 Modify RRDB x23 into Dense projection units x10 using ESRGAN

The structure of the dense projection units is as follows:

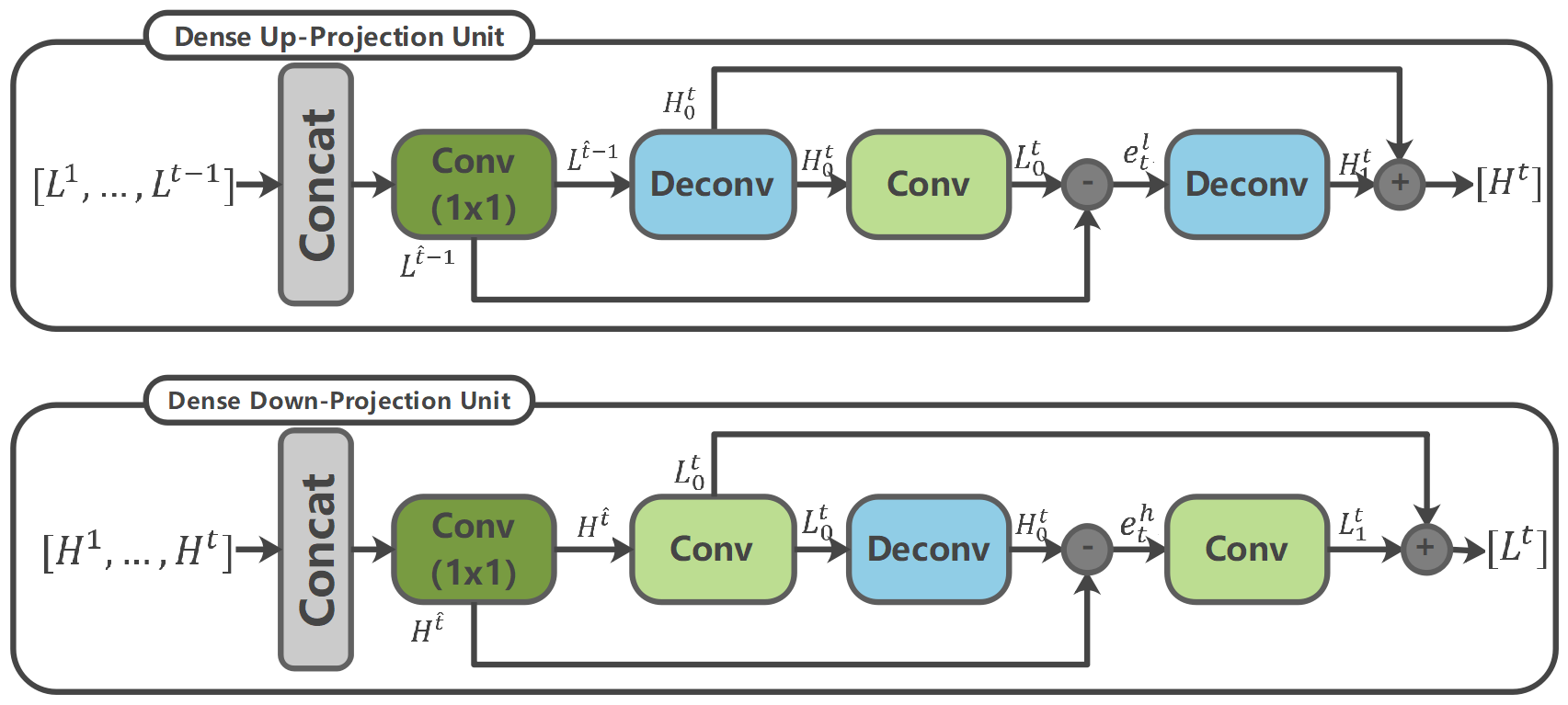


Figure 4-3-3 The structure of the dense projection units

Due to the limited computational power, we were unable to train the complete model. Hence, we used the DF2K data set to do the training after we set the batch-size to 7. The training frequency of plan 1 is 130k, and that of plan 2 is 70k. We used the PIRM2018 data set for verification, and the data is as follows:

Table 4-3-1 The result of these model

|  |  |  |  |
| --- | --- | --- | --- |
| Perceptual index | | | |
| DBPN\_GAN | RRDBx13\_Projectionx5 | Projectionx10 | ESRGAN |
| 2.198830551 | 2.1927 | 2.270088004 | 2.086542 |

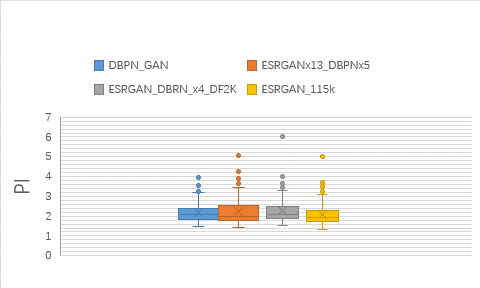


Figure 4-3-4 The variance of Pi Value

Compared with the original plan of using ESRGAN, the PI value of the modified model increases gradually with the increase of dense projection units. The modified model has a smaller batch size and fewer iterations, and the value of PI is higher than the original model. Through analysis, we believe that we added the corrective feedback mechanism achieved by up-down sampling to the model via appending Dense projection units, thus enables the modified model to restore detailed features better, which has a huge effect on improving PI values. In the meantime, we speed up the convergence of the model and reduce the training time of the model by adding dense projection units, this enables us to assign PSRN values similar to the original model under the premise of using smaller batch size and basically the same number of iterations.

**5. Result**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Parameters | | | | |
| ESRGAN | Residual network | Residual path | batch | Use wn | Perceptual loss |
| 23xRRDB | Normal | 23 | No | 2.0865 |
| 23xRRDB | Pre-Upsamping | 6 | yes | 2.1491 |
| RRDBx13\_DBPNx5 | Normal | 7 | no | 2.1927 |
| DBPNx10 | Normal | 7 | no | 2.270088004 |
| DPBNGAN |  |  | 7 | no | 2.198830551 |