Deep Learning for Image Processing

Final Report

Zhetao Zhuang

Supervisor: Miguel Rodrigues

Second Assessor: Miguel Rio

March 2017

DECLARATION

I have read and understood the College and Department's statements and guidelines concerning plagiarism.

I declare that all material described in this report is all my own work except where explicitly and individually indicated in the text. This includes ideas described in the text, figures and computer programs.

Name: Zhetao Zhuang

Signature: Zhetao Zhuang

Date: 20/03/2017

Table of Contents

Deep Learning for Image Super-Resolution

Zhetao Zhuang

Department of Engineering, University College London

Abstract. This paper compares several learning models for image super resolution (SR) and analyzes a deep learning model which is inspired by DCGAN[1] and Residual Network[2]. It learns a hierarchy of representations from object parts to scenes in both the generator and discriminator. The mapping is represented as a deep convolutional neural network (CNN)[3] that takes the low-resolution image as the input and outputs the high-resolution one. The layers are reformulated as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. This model shows a faster learning speed than old fashioned architectures, a lighter structure and state-of-the-art restoration quality.

1 Introduction

Single image super-resolution (SISR) is a classical problem in computer vision. It is widely used in applications such as object detection, security and medical imaging where more image details are required on demand. SISR is one of the core research directions in computer vision community. Statistical image priors or internal patch recurrence are utilized in early methods such as bicubic interpolation and Lanczos resampling[4] which proves to be powerful . Most of state-of-the-art methods for single image super-resolution are example based, for example, Neighbor Embedding[5]

method interpolates the patch subspace and sparse coding[6] method learns a low and high resolution pair based on sparse signal representation. Internal similarities of the same image are either exploited or the mapping functions are learned from external low and high resolution exemplar pairs by using these methods. Although they are often provided with abundant samples, it is difficult for them to model the data effectively.

Recently, some widely used learning model have been applied to learn an end-to-end mapping between the low/high resolution images. Random Forest [7] and CNN[3] have been used with large improvements in accuracy. CNN is one of the representative methods for learning a mapping from LR to HR in an end-to-end manner. It helps to build most sophisticated computer vision applications such as auto-tagging of friends , facial security features, self-driving cars, gesture recognition, automatic number plate recognition, etc. One of the most typical methods,namely SRCNN[8], engineered features are not required while these are often important in other methods such as sparse representation[9]. SRCNN achieves fast speed for practical on-line usage with moderate numbers of filters and layers. The result can be improved further with more larger filter size and filter numbers. Also, it shows a state-of-the-art performance.

However SRCNN has some limitations. For example, it is obvious that the net work only works for a single scale. Second and it relies on the context of relative small image regions. An improved model called VDSR[10] uses much deeper layers with larger receptive field. It helps to overcome the problem of small contextual information occurred in SRCNN and residual learning with large learning rate is applied to accelerate the learning speed when the network is very deep,for example,20 layers. In addition, this model copes with multiscale SR problem in a single network. Compared to SRCNN and VDSR, DRCN[17], which also uses deep network and large receptive field,

learns the features of input images in a recursive manner where the number of parameters to be learned are effectively reduced by using the same weight. Both VDSR and DRCN are relatively accurate and fast in comparison to state-of-the-art methods such as A+[18] and sparse coding.

The proposed method in this paper is actually the combination of CNN,RN(Residual Net)[11] and General Adversarial Net[12]. It contains the two parts which are a generative model with RN and a discriminative model respectively. The generator network relies on RN modules to train substantially faster than more old-fashioned architectures.

2 Background

In order to have a better understanding of the deep learning models for image super resolution. Here is a brief introduction to the challenges faced by computer vision,some basic knowledge of CNN and why it can be used efficiently for the computer vision related tasks.

2.1 Challenges in Computer Vision

There are some key roadblocks in computer vision which may make it difficult for the tasks related to object detecting and recognizing. For example, the same object can have different positions and angles in an image depending on the relative position of the object and the observer, some of the images might blend into the background. Also, different images can have different illumination and some part of the images might be hidden. These problems can increase the complexity in related tasks, thus make it more

difficult to classify or recognizing an image. We will come across these problems later in the experiment section. Some examples are shown in Picture 1 and Picture 2.

Picture 1 Difference in viewpoint

Picture 2 Difference in illumination

2.2 Image super-resolution

The recovery of the high resolution image based on the input images and the imaging mode. The goal of super-resolution is to find some better models that can map the low resolution images to high resolution ones. It also suffers the problems described before. The accuracy of imaging model is vital for super-resolution and an incorrect modeling can actually degrade the image further. In deep learning, the amount of contextual information,which should be exploited to infer missing high frequency components, are determined by the receptive field of a convolutional network. In addition, more neighbor pixels are collected and analyzed as a result of more contextual information which helps us to understand on what may be lost. Moreover, the depth of the network also plays a pivotal role in figuring out the image details. Deep network, usually with 20 or more layers, can extract more images details and improve the performance of the model. However, larger and deeper network means that it is much harder to train the model and computational abilities should be considered.

2.3 Convolutional Neural Networks

A CNN typical consists of 3 type of layers:convolution layer,pooling layer and fully connected layer. *Local receptive fields, feature map, shared weights and pooling* are the four basic ideals in CNN. For *local receptive fields*,lets consider inputs as a 28×28 neurons. In the first hidden layer, each neuron is connected to a small region of the input neurons, for example, a 5×5 region(or patch). That region in the input image is called the *local receptive field,* as shown in Fig.1. A weight and an overall bias will be learned in each connection and hidden neuron respectively.

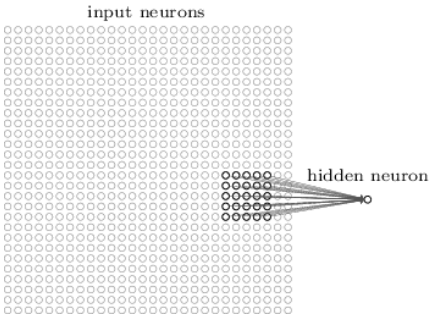


Fig.1 Local Receptive Field

Then the local receptive field is moved across the entire input image. In the first hidden layer, each local receptive field is corresponding to a different hidden neuron , as shown in Fig.2 and Fig.3.

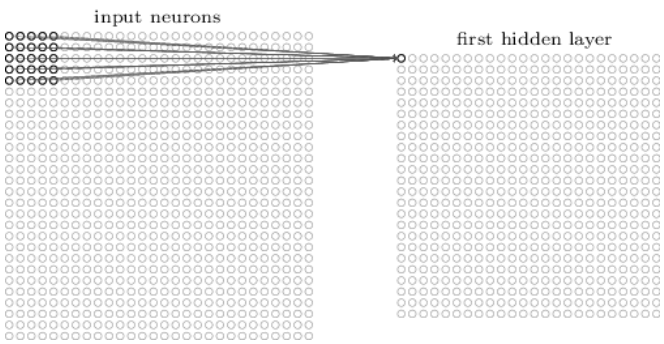


Fig.2 First hidden neuron from top-left corner pixel

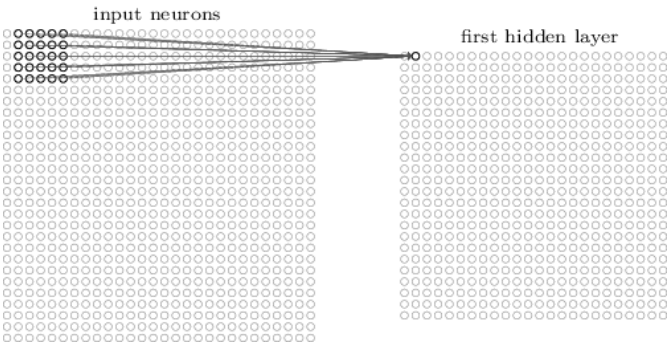


Fig.3 Slide the local receptive field over by one

This results a 24×24 hidden layer with each hidden neuron has a bias and 5×5 weights

connected to its local receptive field. In addition, each of the 24×24 hidden neurons shares the same weights and bias. Therefore, the parameters in the network are largely reduced. A *feature map is t*he map from the input layer to the hidden layer and the *shared weights* are defined by the feature map. There can be several maps,for example,in this case the hidden layer can be 24×24×3 with 3 feature maps, as shown in Fig.4.

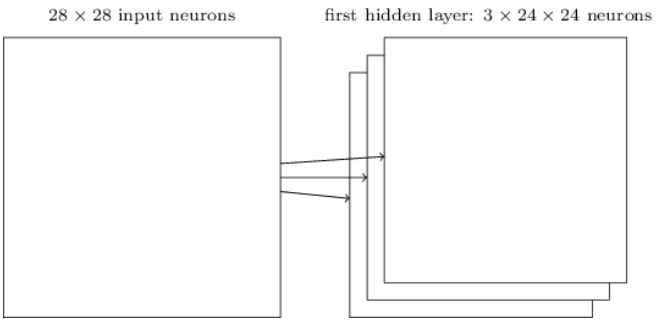


Fig.4 First hidden layer with 3 feature maps

Apart from the convolutional layers above, *pooling layers* are also involved in CNN. Usually, we use pooling layers after convolutional layers immediately. The purpose of pooling layers is to simplify the content in the output from the convolutional layer and reduce the risk of overfitting. One popular method for pooling is called *max-pooling*. The maximum activation in the 2×2 input region in *max-pooling is* simply output by a pooling unit. In this case, a 24×24 layer is reduced to 12×12 layer, as illustrated in Fig.5.

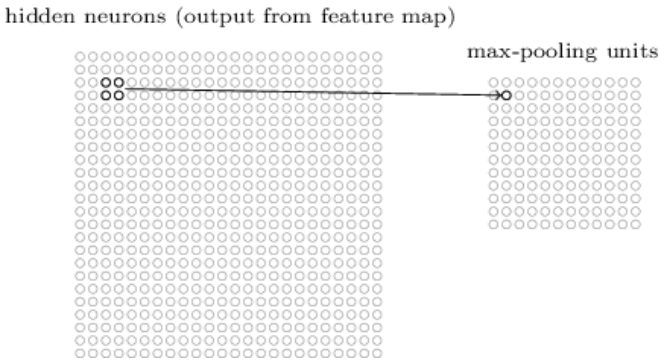


Fig.5 Each unit in the pooling layer may summarize a region of 2×2 neurons in the previous layer.

A complete convolutional neural network is formed by putting these layers together as

shown in Fig.6. Started with 28×28 input neurons,the neural network is then followed by a convolutional layer which uses a 3 feature maps and 5×5 local receptive field.

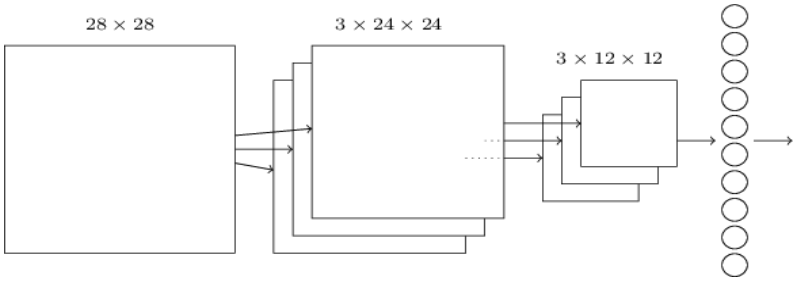


Fig.6

A layer of 3×24×24 hidden feature neurons is built. Then a max-pooling layer is applied to 2×2 regions, in each of the 3 feature maps. A layer of 3×12×12 hidden feature neurons is completed. A *fully-connected layer* is shown in the final layer of connections in the network. Every neuron from the max-pooled layer is connected to the every output of neurons in the fully-connected layer.

2.4 Feature extraction using convolution

One of the most important part in CNN is using convolution to extract features from the images.The convolutional theorem explained that the convolution of **f** and **g** is equal to the element wise product of their Fourier Transforms:

$$F(\mathrm{f} * g) = F(f) \cdot F(g)$$

This allows us to transform an natural image to the representation of magnitude and phase. From the past papers and experiments we know that magnitude spectra of all natural images are quite similar and most information in the image is carried in the phase, not the amplitude. Therefore, what convolution really does is to extract those information carried in the phase of images by using 2D Fourier Transforms. Fig.7 shows how convolution actually works in a 2D matrix.

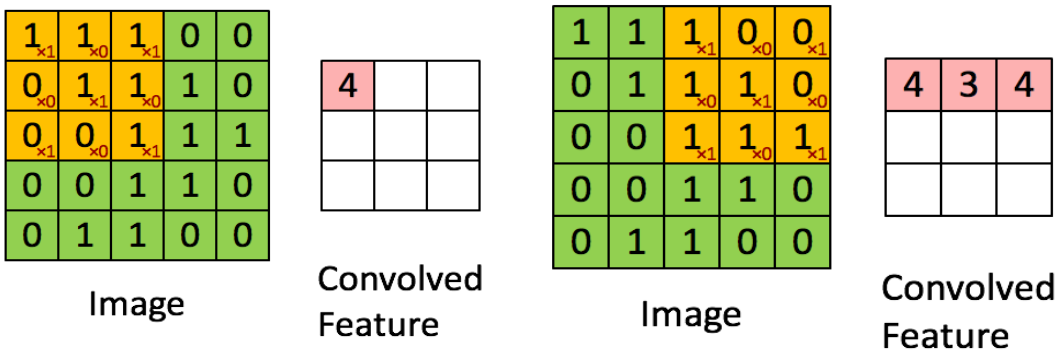


Fig.7[16] The Convolution operation. The output matrix is called Convolved Feature or Feature Map.

Being stationary is one property of natural images. This means that the features that we learn at one part of the image can also be applied to other parts of the image and the same features can be used at all locations[16]. This is why we can see that CNNs uses shared weights and bias. Learned features over small patches can convolve with larger image so that a feature activation value is obtain at each location in the image. Those features containing larger information will get a higher value after convolution so that the desired feature are extracted from the input images.

2.5 Why CNN works better

For those traditional methods described before, they are not capable of catering to vast amount of variations in images,for example,variations in viewpoints and difference in illumination.In deep CNN, small pieces of information are modeled consecutively and combined deeper in network. One easy way to understand it is that we can think the first layer will try to detect the edges and templates are formed for edge detection. Then in the following layers,these templates are combined to form simple shapes and eventually into different object positions,scales,illumination,etc. The final layers will match an input image with all the templates and the final layer aggregates them together with different weights. Therefore, complex variations and behaviour can be modeled by deep CNNs while giving highly accurate predictions.

3 Related Work

Sparse coding,SRCNN,VDSR and DRCN are typical methods for SR. It is necessary to compare and analyze these models to help us have a better understanding for SR when using deep learning methods. More effort will be spent on how SRCNN works as it is the fundamental of understanding how deep learning can be applied to SR.

3.1 Sparse Coding

This method is one of the typical external example based image super resolution. The patches are extracted and whitened from the image and a low resolution dictionary is used to encode the patches. Then high-resolution patches are reconstructed through the sparse coefficient which are passed into a high resolution dictionary. The output is formed by aggregating the reconstructed patches. However in an unified optimization framework, the rest of the steps are rarely optimized or considered.

3.2 SRCNN

This method consists of three layers: patch extraction/representation, non-linear mapping and reconstruction. Filters of spatial sizes 9×9, 1×1, and 5×5 were used respectively. Lets denote the low-resolution input image as X and the ground truth high-resolution image as Y. The purpose is to generate an output image F(X) that is similar to Y as much as possible. Three main steps are involved in learning the mapping :

1. First step is to initialize a set of patches which are used to extract features from low-resolution images X and then represent each of them as a high dimension vector.

The first layer can be expressed as F1 by the following equation:

$$F1(X) = \max(0, W1 * X + B1) \qquad (1)$$

W1 is a set of filter with size c×f1×f1×n1, where c, n1 represents the channel of the input image and number of output filters in layer respectively. The patch size is f1×f1 and B1 is biases with n1 dimension.

2. After the extraction of features with n1 dimension in the first layer, the second convolutional layer is defined where each n1-dimensional vectors are mapped non-linearly to an n2-dimensional vector. This can be demonstrated by the equation:

$$F2(X) = \max(0, W2 * F1(X) + B2) \qquad (2)$$

where W2 has a size of n1×f2×f2×n2 and B2 is a n2-dimensional.

3. The final stage is the aggregation and average of the high-resolution patches obtained in previous layer. The third convolutional layer can be defined to produce the output image by the equation:

$$F(X) = W3 * F2(X) + B3 \qquad (3)$$

where W3 is n2×f3×f3×n3 and B3 is a n3-dimensional vector. The average of high-resolution patches depends on whether they are in image domain or not. The loss function used in this model is Mean Square Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left\| F(Xi;\theta) - Yi \right\|^{2} \qquad (4)$$

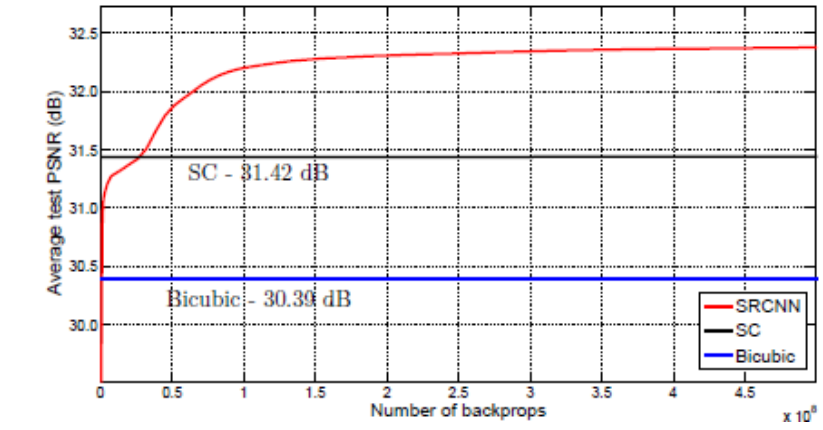
4. where n represents the number of training examples,F(Xi;θ) is the output of the model and Yi is ground truth image. Stochastic gradient descent with back propagation is used to minimized the loss function. The weights are updated by the following

equation:

$$\Delta_{i+1} = 0.9 \cdot \Delta_i + \eta \cdot \frac{\partial L}{\partial W_i^l}, W_{i+1}^l = W_i^l + \Delta_{i+1} \qquad (5)$$

where $l \in \{1,2,3\}$ and i are the indices of layers, ŋ is the learning rate. The weight W of each layer is initialized by Gaussian distribution with zero mean and standard deviation of 0.001 and 0 for bias.

There are several appealing properties for SRCNN. First, it gives a much better accuracy when comparing with state-of-the-art example-based methods described before. Second, the result can be further improved by using larger dataset or a larger model. Third, this model shows a faster training speed with moderate number of layers and filters. Fig.8 shows a comparison of different method on an example.



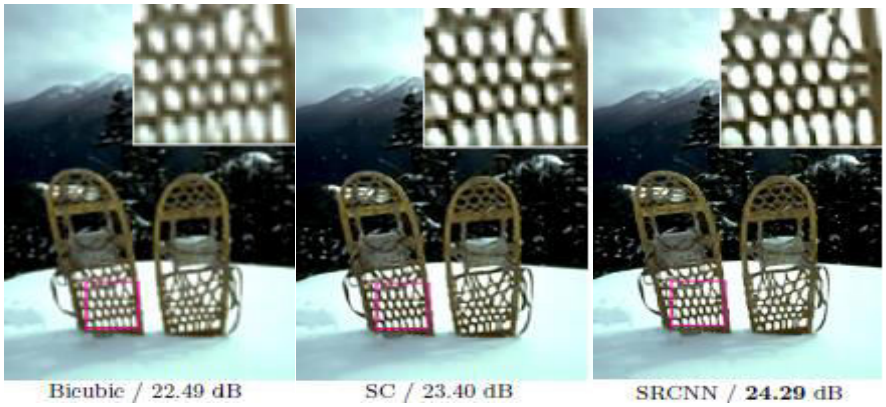


Fig. 8[8] PSNR of SRCNN exceed SC in a few number of iterations

3.3 VDSR

This method uses 20 weight layers with 64 filters(filter size is 3×3) for each convolutional layer. The network is very deep (20vs3 [8]) and information used for reconstruction (receptive field) is much larger (41×41). Fig.9 demonstrates the structure of the model.

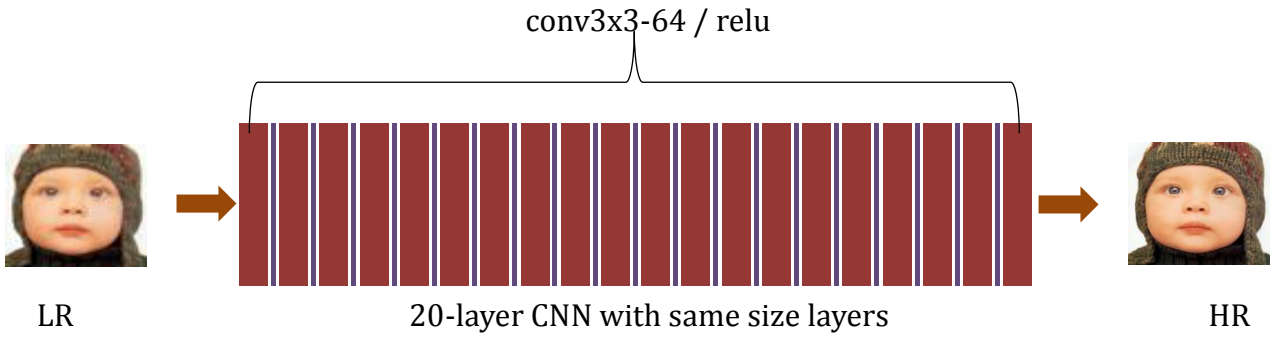
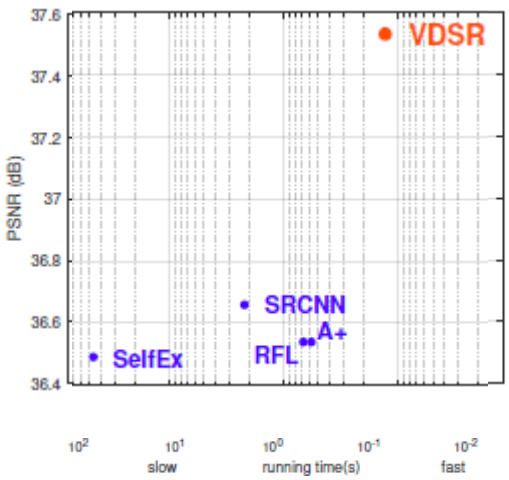


Fig. 9 Simple illustration of VDSR

SRCNN carries the input to the end layer and residuals are reconstructed which is very similar to what an auto encoder does. Thus, training time is decreased due to this structure. While in VDSR, the residual image are modeled directly so much faster convergence is achieved. Comparing to many other deep learning models for SR, which result in worse performance when increasing the depth, performance in this method is boosted significantly with deep structure. Fig.10 shows the running time for different models.



Ground Truth(PSNR) SRCNN(27.02) VDSR(27.32)

Fig. 10[10] The running time for VDSR is much smaller than the others and it outperforms other models.

Also, VDSR can cope with multiple scales. From Table.1 we can see that when training scale is not equal to testing scale then the performance will degrade. For example, with training factor 2, the PSNR for testing factor 3,4 reduce to 30.43 and 28.43. However, if the multiple training scale factor are used during training,for example factor {2,3,4}, then the result will actually be improved when compared to use single training factor(37.10 vs 37.06,33.27 vs 30.43 and 30.95 vs 28.43).

Test/Train	2x	3x	4x	2,3x	2,4x	3,4x	2,3,4x	Bicubic
2x	37.10	30.05	28.13	37.09	37.03	32.43	37.06	33.66
3x	30.43	32.89	30.50	33.22	31.20	33.24	33.27	30.39
4x	28.43	28.73	30.84	28.70	30.86	30.94	30.95	28.42

Table.1[10]: Scale Factor Experiment. Several models are trained with different scale sets. Quantitative evaluation (PSNR) on dataset 'Set5'is provided for scale factors 2,3 and 4.

3.4 DCRN

When increasing the depth of the network by adding new weight layer introduces more parameters which could result in overfitting problem and make the model become to huge to be stored and retrieved. DCRN helps to reduce the number of parameters to be

learned effectively by repeatedly applying the same convolutional layer as many times as desired. If we compare this model to VDSR, we can see that DCRN is almost a recursive version of VDSR. The model are divided in to three part as shown in Fig.11.

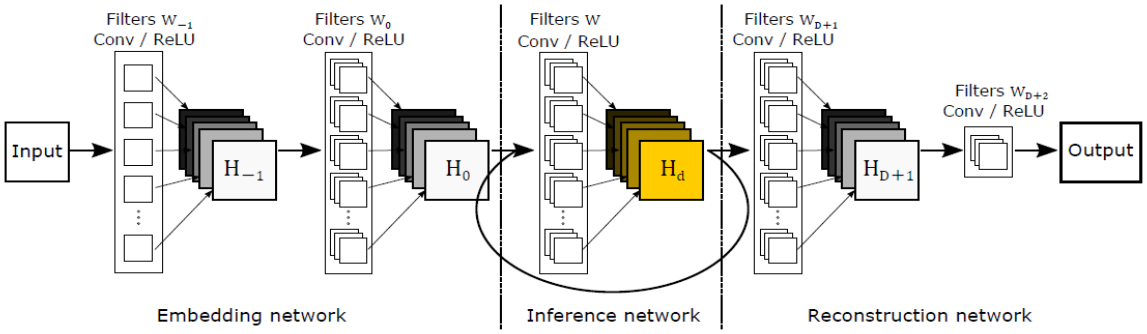
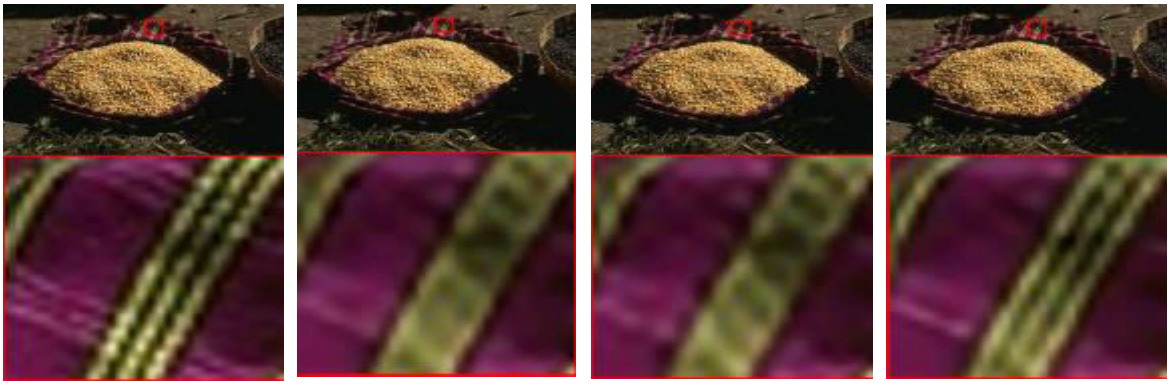


Fig. 11 Architecture of DRCN

The embedding network(2 layers) takes the input image (grayscale or RGB) and represents it as a set of feature maps. Then the output from embedding net is fed into inference network(16 layers which use the same weight in a recursive manner) which is the main component that analyzes a large image region by recursive layers. Each recursion applies the same convolution followed by a rectified linear unit. The reconstruction network(2 layers) is used to reconstruct the high resolution images by combing the output from recursion layers and input layer. 20 weight layers (of filter size 3×3) with 256 filters for each convolutional layer and receptive field 41×41 are used.To ease the difficulty of training the model, all recursions are supervised which means all predictions resulting from different levels of recursions are combined to deliver a more accurate final prediction. Also, input is connected to the output for reconstruction. Some results from the corresponding paper are shown in Fig.12.



Ground Truth(PSNR) A+(24.24) SRCNN(24.48) DRCN(25.76)

Fig. 12[17] The PSNR result of *B100* with scale factor ×2

4 Proposed Model

4.1 Proposed Network

For image super resolution, the proposed model is inspired by DCGAN[1] and Residual Network[2]. The generator part G captures the sample distribution and the discriminator part D estimates the probability that the sample comes from the real data rather than G. Early in learning, when G is poor, D can reject samples from G with high confidence because they are clearly different from the training data. The goal is to maximize the probability of D making a mistake. More specifically, we are training G so that if we input a sample to G and feed the output of G ,namely the sample distribution,to D so that D can not tell whether the sample is from sample distribution or data distribution,i.e D(x)=1/2. D(x) represents the probability that x came from the data rather than G and G(x) is a differentiable function represented by a multilayer perceptron. Through updating the parameters, both G and D are optimized. In other words, D and G play the following two-player minimax game with value function V (G,D):

$$\min_{G}\max_{D} V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_{z}(z)}[\log(1-D(G(z)))] \quad (6)$$

where p_Z is input noise variables.

For G fixed, the optimal discriminator D is :

$$D_G^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)} \quad (7)$$

The global optimality of the training criterion V (G;D) is achieved if and only if:

$$p_g = p_{data} \quad (8)$$

where p_g is generator's distribution and p_{data} is data distribution.

The advantage of adversarial networks is that they can represent very sharp distributions. Fig.13 shows the basic layout of the model.

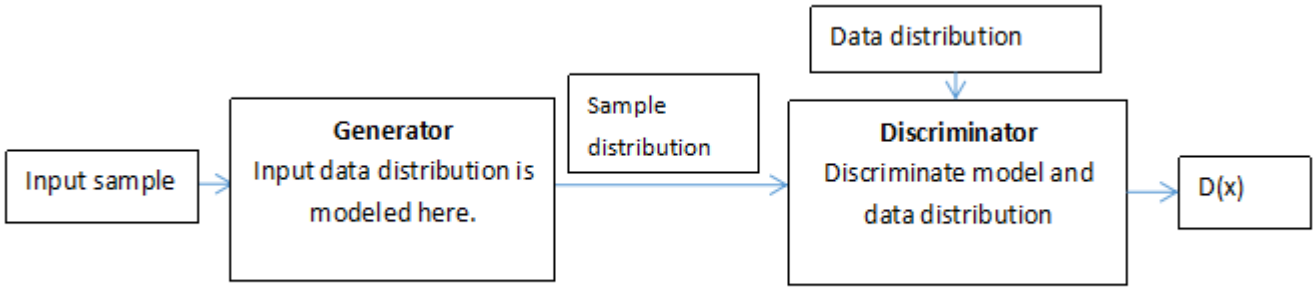


Fig.13 Proposed network layout

For the generator part, residual network is applied for improving the performance and prevents the model from saturation and vanishing/exploding problems. Deep residual nets can easily enjoy accuracy gains from greatly increased depth, producing results substantially better than normal networks. In the proposed model, the residual block used is an improved version of the original one which gives a better performance. The structure is shown in Fig.14 and Fig.15

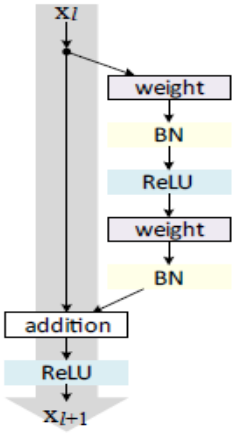


Fig.14[11] Residual Bock with identity mapping

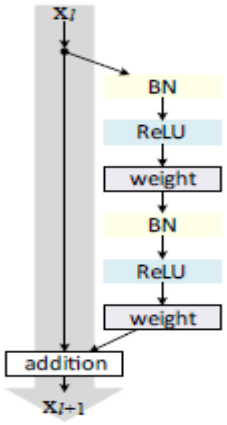


Fig.15[11] Improved Residual Bock

4.2 Structure for Experiment

The input has 3 channels and the generator consists of 3 layers with 256,128,96 channels respectively. The filter size between each layer is 3×3. Between first and second layer two residual block with identity mapping is add as well as second and third layer. After the second layer, the feature size is up scaled by 2 times. The structure of generator is simplified in Fig.16 where b,H,W and c represent batch size, height, width and number of channels respectively. The reason for up scaling is that the input image is 4 times smaller than the ground truth image.

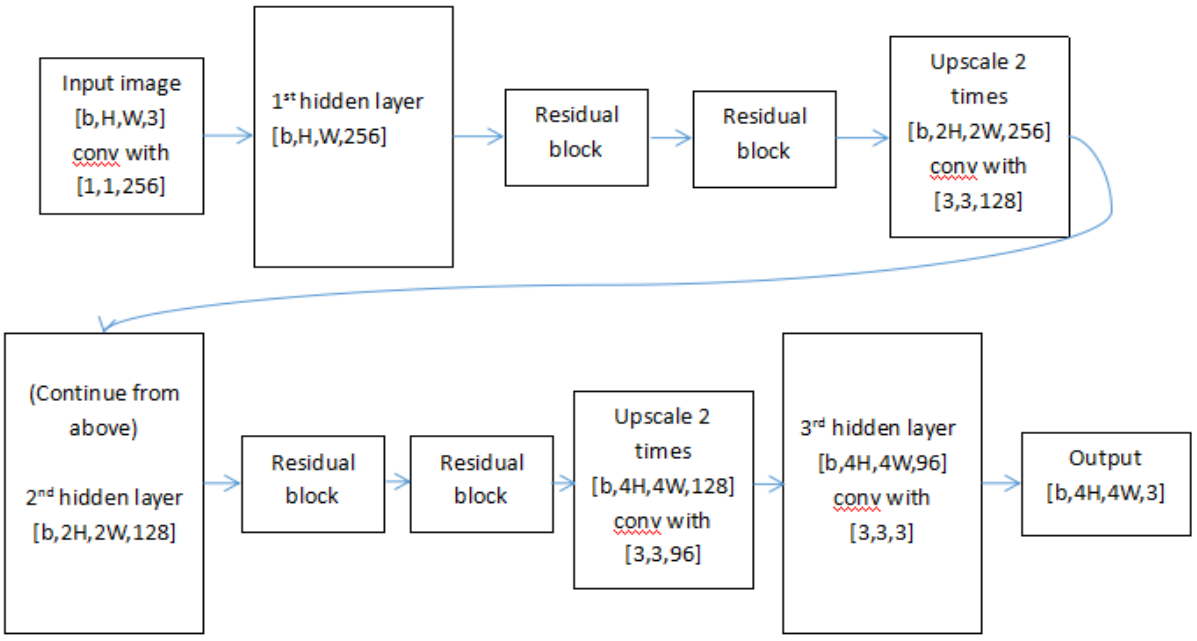


Fig.16 Simplified generator structure for experiment

The structure for discriminator is much simpler as it does not has residual network because it will not provide any improvement during the experiment. The model is thus a fully convolutional model where 4 layers with 64,128,256,512 channels in each layer are used and the output of the discriminator will be a single value. The loss function applied in this model is Mean Square Error (MSE). Stochastic gradient descent with back propagation is used to minimized the loss function. In addition, the loss function of the generator has a term that measures the L1 difference between the sample input and the output image produced by the generator. The initial learning is set to be 0.0002 and it will be halved after certain batches,i.e. 1000 batches. Decreasing learning rate can help to gain more details of the features although this could slow down the training time a bit.

5 Experiment

In this section, four different deep learning models, SRCNN,VDSR,DRCN and proposed model,are implemented. For all SRCNN, VDSR and DCRN, the training set consists of 91 images proposed in [19] and the **Set5**[8] are used to evaluate the performance of upscaling factor 3. The datasets used for proposed model is the Large-scale CelebFaces Attributes (CelebA) Dataset with the Align&Cropped Images version.

The training parameters for SRCNN are set to be f1=9,f2=1,f3=5,n1=64 and n2=32 with learning rate 0.0001. VDSR use a network of depth 20 with filter size 3×3 and batches of size 64. Momentum and weight decay parameters are set to 0.9 and 0.0001 respectively along with a initial learning rate of 0.1 which is then decreased by a factor of 10 every 20 epochs. For DRCN,20 convolutional layers (receptive field of 41 by 41)

with 16 recursions are used(256 filters of the size 3 × 3 for all weight layers) . It has the same momentum and weight decay parameters as VDSR for the convenience of comparison. Training images are of size 41 by 41 patches with stride 21 and 64 patches are used as a mini-batch for stochastic gradient descent.

5.1 SRCNN and VDSR

For comparison, the two models are put in the same sub section.In the training phase,the ground truth are prepared as 33×33 and 41×41 sub images randomly cropped from training images for SRCNN and VDSR respectively. The output in SRCNN after each layer decreases due to the convolution. However,in VDSR, the output after each layer remain the same due to zero padding in each layer,so the resulting output for the former one is 21×21 and the latter one is 41×41. MSE loss is evaluated between the corresponding output and labels. After the implementation of SRCNN ,the result of mean square error(MSE) for this model is shown in Fig.17, the horizontal axis represents the number of iterations and is the same for the following figures. From this figure we can see that the MSE converge to a value around 4.9 in the first 1000 iterations and can hardly be reduced further. The reasons why this could happen can be related to the relative small filter size and shallow depth of the network.

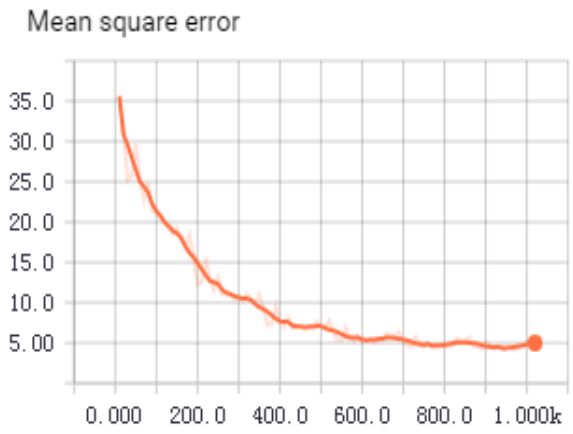
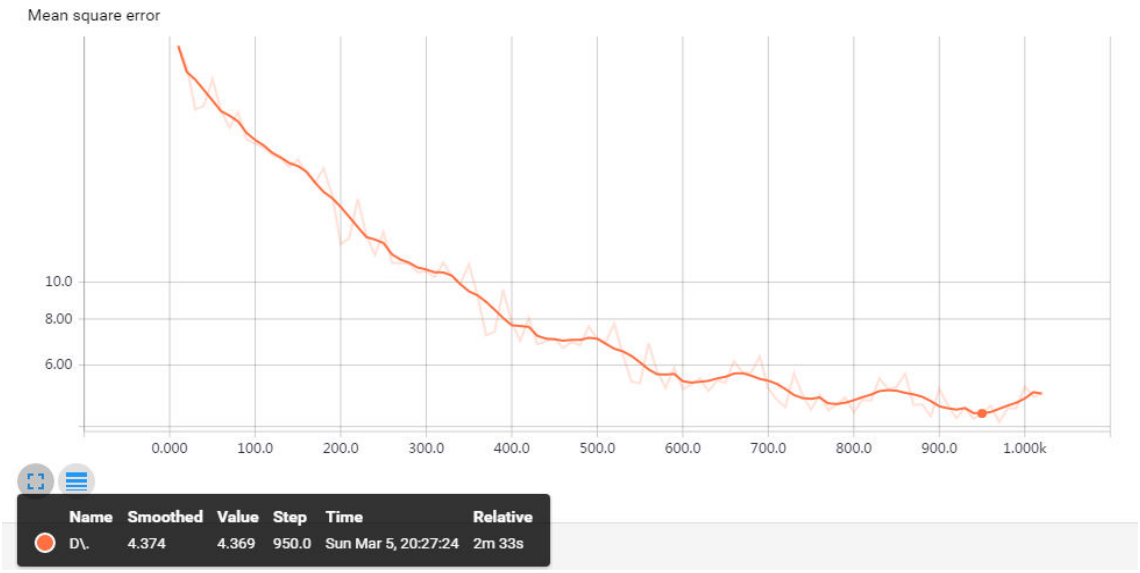


Fig.17 Mean square error for SRCNN for scale factor 3 in the first 1000 iteration

For SRCNN, the experiment result could be further improved by taking one of the two steps:

- Increase the number of filters. For example, change n1=64,n2=32 to n1=128,n2= 64. The resulting PSNR from the 'bird_GT.bmp' in Set5 increases from 35.47dB to 35.50dB. Also the MSE reduced to a lower minimum of 4.37 in the end of first 1000 iterations when compared to former one , which is 4.98. The The result is shown in Fig.18

- Increase the filter size. For example, enlarge the filter size from 9×9, 1×1, and 5×5 to 11 ×11, 1×1, and 7×7. The result will be further improved. This is well consistent what discussed before that larger filter size will give us better contextual information.



(a)

Number of filters	n1=64,n2=32	n1=128,n2=64
3x - PSNR	35.47	35.50

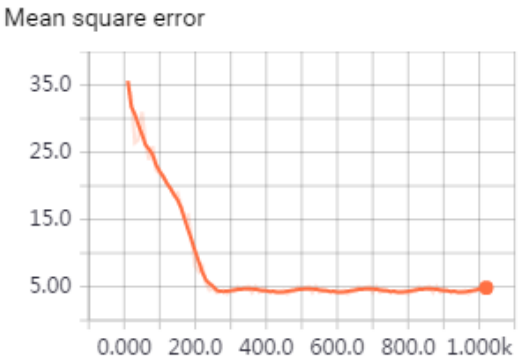
(b)

Fig.18 (a)&(b)n1=128 and n2=64,MSE and PSNR on 'bird_GT.bmp' in Set5

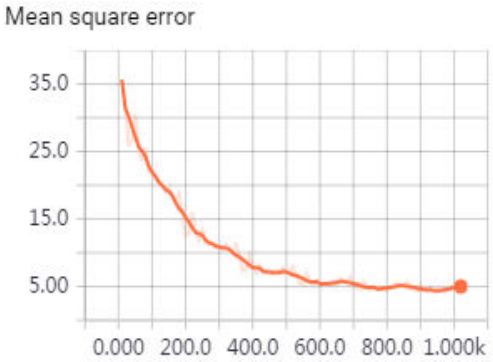
However, adding more layers to SRCNN does not give a much better result and in some cases it could actually degrade the performance. Also, more time is needed to train the model. I have tried to add 5,10 and 15 layers respectively. The filter size for each layer added is 3×3 and the output of each added layer are padded to be the same. In Fig.19(b), the MSE for the model decreases further(with a lower minimum around 3.9) and converges quicker when 5 more layers are added. However, in Fig.19(c)&(d), the MSE does not improve much when 10 and 15 more layers are used. More importantly, from Fig.19(e), the model could suffer a larger probability of converging to local minimum when more layers are added. From the table in Fig.19(a),we can see that the PSNR decreases a bit when the number of layers added is 10 and 15. This may due to the gradient vanishing/exploding problem when we increase the depth of the network or the internal covariate shift. In order to decrease the effect of internal covariate shift, I have added batch normalization[20] after each convolutional layer except the last layer for reconstruction during training and it should be removed during testing.

Number of layers added	**0**	**5**	**10**	**15**
3x - PSNR	35.47	35.51	35.46	35.44

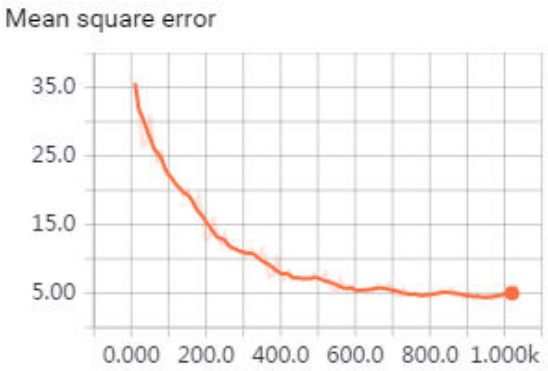
(a) result of PSNR when adding more layers

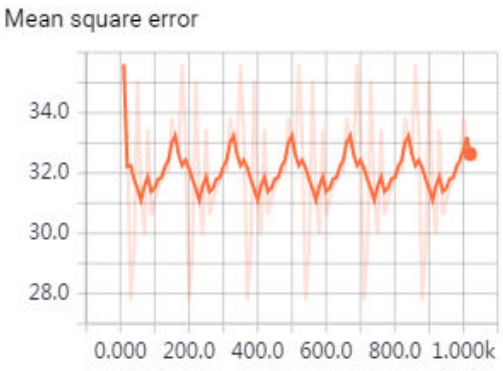


(b)5 layers added



(c)15 layers added





(d)10 layers added (e)10 layers added when the model failed to converge

Fig.19 (a)PSNR for the model (b)&(c)&(d)&(e) MSE for the model

In terms of VDSR, although it has 20 layers, this problem is solved by using gradient clipping and residual net. From Fig.20 we can see that the convergence of VDSR is surprisingly faster than SRCNN in just first 1000 iterations and the initial value of MSE is also much smaller. This is due to the residual learning in VDSR which adds the input image to the residual image directly to construct the high resolution image. Therefore, the MSE loss can be reduced much more quickly. The much larger receptive fields used increases the accuracy of this model. The performance of VDSR can be further improved by using more layers.

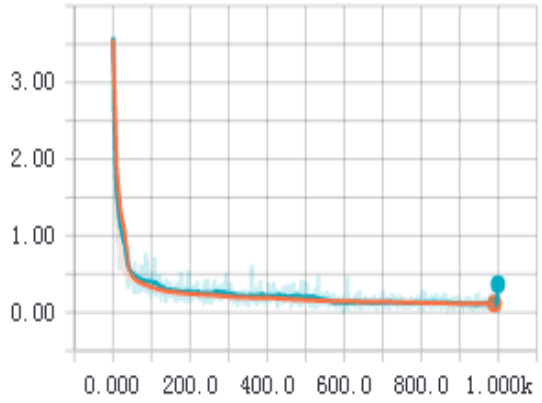


Fig.20 Mean square error for VDSR for scale factor 3

Fig.21 shows the result for residual and non-residual network with 10 weight layers and scale factor 3 to help us have a better understanding of the effect of residual learning. We can see that residual network converges much faster and it shows superior

performance at convergence.

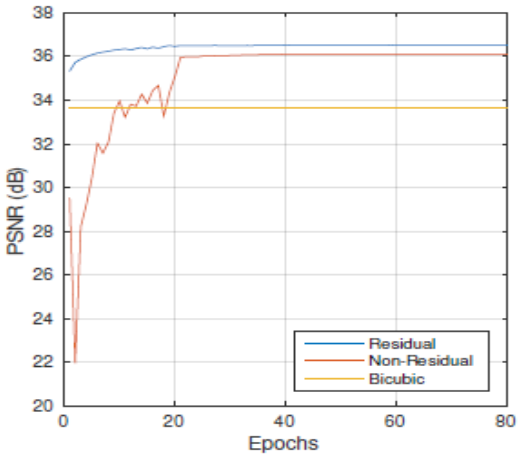


Fig.21 Initial learning rate 0.001

Another reason for faster learning is that larger learning rate is used in VDSR(0.0001 for SRCNN and 0.1 for VDSR). The PSNR of constructed high resolution image benchmark is shown in Table.2. Each value in the table is the average of different test images in Set5. We can see that VDSR also outperform SRCNN with a large increase in PSNR value. The result is consistent to the discussion before. Here are some reconstructed high resolution images in Fig.22 for comparison, the scale factor used is 3.

Scale	Bicubic	A+	SRCNN	VDSR
2x - PSNR	33.66	36.54	36.66	37.53
3x - PSNR	30.39	32.58	32.75	33.66
4x - PSNR	28.42	30.28	30.84	31.35

Table.2 PSNR for Bicubic, A+,SRCNN and VDSR based on Set5

(a) Bicubic,PSNR=32.58dB (b) SRCNN,PSNR=35.47dB (c) VDSR,PSNR=36.23dB

Fig.22 Reconstructed high resolution image for SRCNN and VDSR

After the experiment, we can see that VDSR outperforms SRCNN in four points:

- Larger receptive fields gives larger contextual information and better details of images
- Deeper network gives better accuracy
- Residual learning is used so better accuracy and faster training
- Larger learning rate makes the network converge much faster

5.2 DRCN

The implementation of DRCN is very similar to VDSR except the inference layer and its skip-connection. To have a better understanding of the structure of DRCN, I have quoted the figure from [17] as shown in Fig.23.

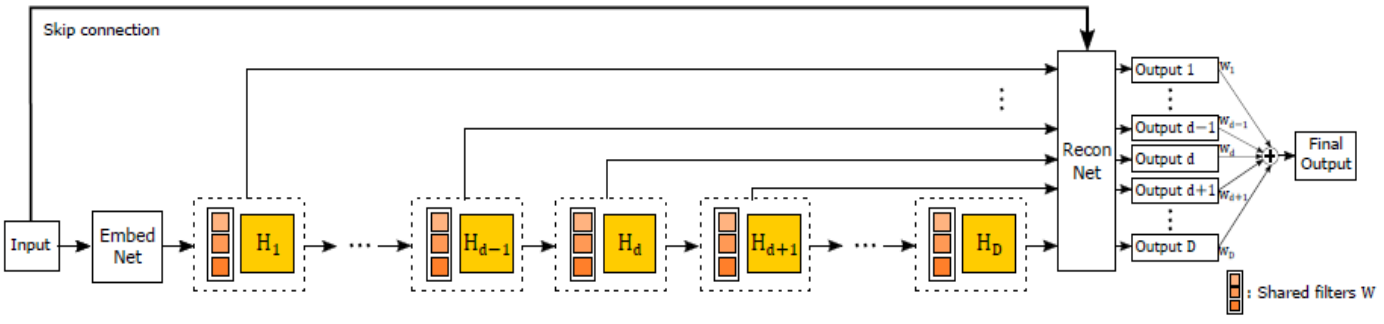


Fig.23[17] DRCN with recursive-supervision and skip-connection

The reconstruction network is shared for recursive predictions and all predictions from the intermediate recursion are used to obtain the final output. Due to the computational capacity of my computer, the training procedure is very slow and it takes about 5

minutes to complete a training iteration. Therefore, I used the trained parameters to generate some results which are shown in Fig.24.

(a) Bicubic,PSNR=33.91dB (b) SRCNN,PSNR=35.25dB (c) DRCN,PSNR=36.96dB

Scale	Bicubic	SRCNN	VDSR	DRCN
3x - PSNR	33.41	35.33	36.55	36.76

F

ig.24 Some result for DRCN on Set5

If we compare the result to SRCNN and VDSR, it is easy to find that DRCN works well in some cases, for example, better performance in Set5. However, one thing I would like to mention is that, from the experiment, I found it is much harder to train the network as the initial MSE is very large and longer training time needed. This may due to the fact that the recursive structure of the network suffers more of the gradient vanishing or exploding problems and less likely to converge to global maximum.

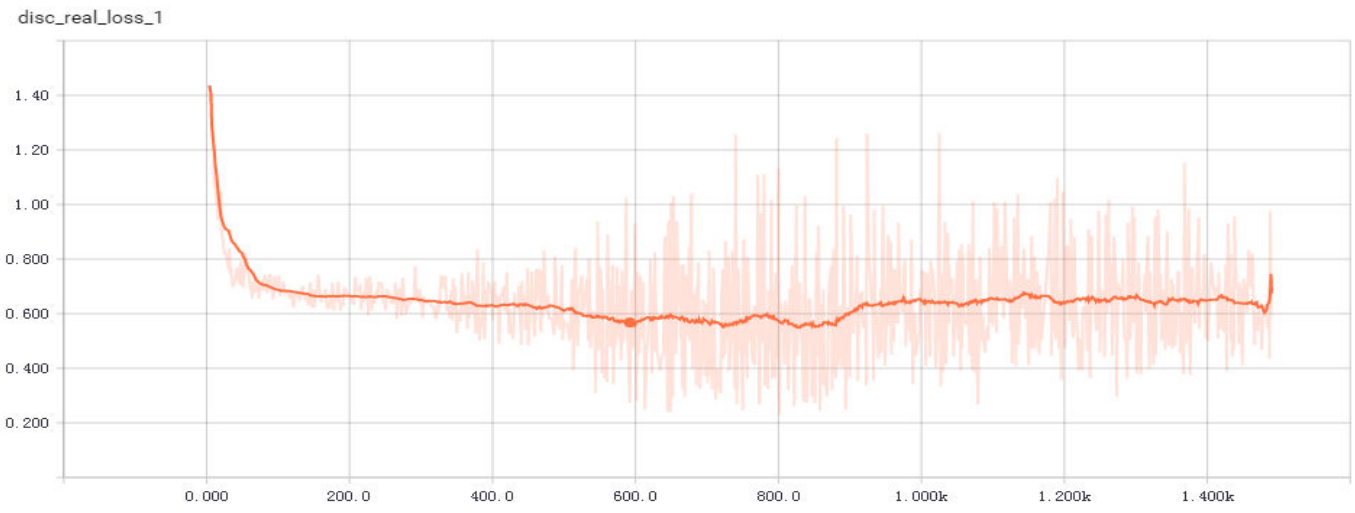
5.3 Proposed Model

In this section, based on some existing different version of this proposed model, a similar model is rebuilt. For academic use, I apply the knowledge from previous sections and the proposed model to generate some more state-of-the-art result.

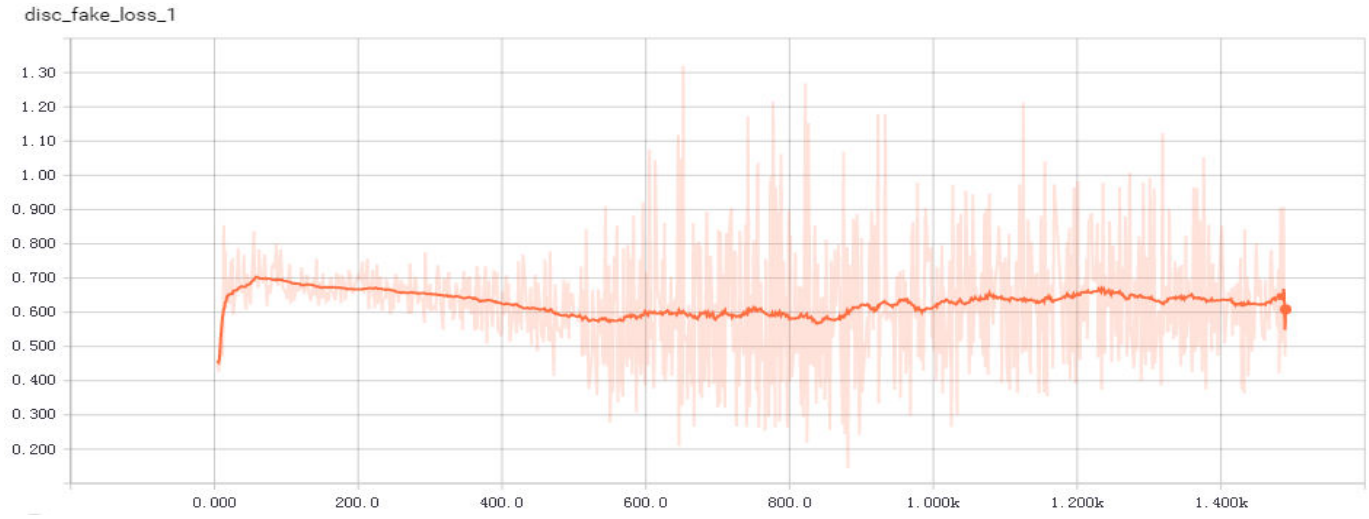
The proposed model is basically the combination of methods depicted before with the

structure of generator and discriminator model. The input images are randomly cropped from original image with some preprocessing such as flip and saturation and finally resized to 16×16. Then the input images are upscaled by a factor 4. The resulting 64x64 images display sharp features that are plausible based on the dataset that was used to train the neural net. The label images are of size 64×64.

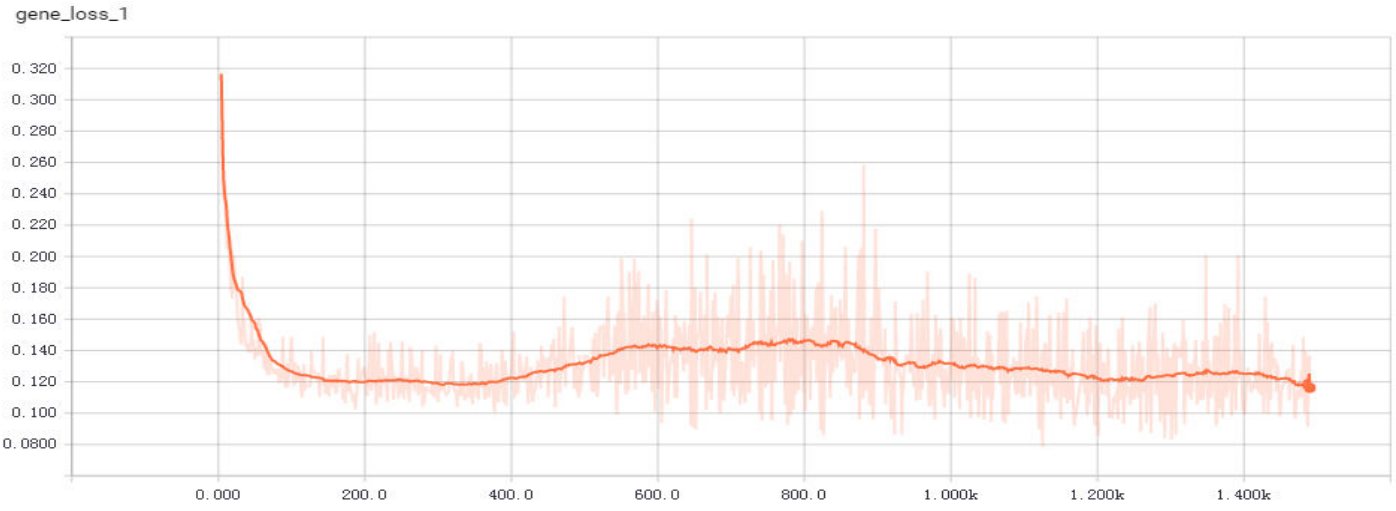
The model shows some interesting reconstructed image after about 1,500 training iterations,about 24,000 batches. Fig.25 shows the loss for generator and discriminator during the training, the horizontal axis represents the number of training iterations. The loss for discriminator from real data and generator converges very quickly in the first 200 iterations and tends to remain at an average of approximate 0.6 and 0.12 respectively. The loss for discriminator from fake data,where the input data to the discriminator comes from the output of generator,increases to around at 0.6 and remains at that level. The result is well consistent with the principle of GAN described before as the increase of loss from fake data will decrease the loss from real data until a balances is reached, that is they are almost equal to each other approximately at 0.6. This means the discriminator can not tell whether the data is from the output of generator or real data and both generator and discriminator is optimized.



(a) Loss for discriminator from real data



(b)Loss for discriminator from fake data

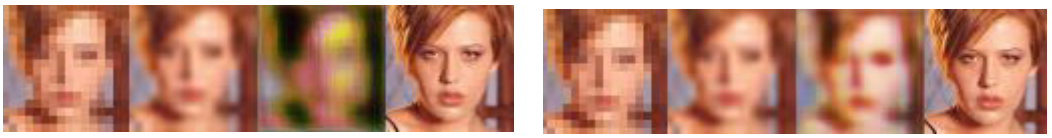


(c) Loss for generator

Fig.25 Loss for generator and discriminator

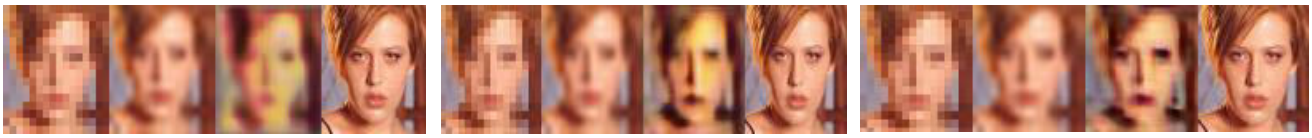
The loss function of the generator has a additional term that measures the L1 difference between the 16×16 input and down scaled version of the output(64×64 to 16×16) produced by the generator. This term is similar to the residual learning and the result proves that this term makes the convergence of the network greatly accelerate during the first iteration and this may also help to prevent the generator to end in local poor maximum. Fig.26 shows some result for with and without L1 term. The first column is input image, the second column is bicubic interpolation,the third column is network

output and the last one is ground truth image. We can see that the training result in first 10 iterations with L1 term shows more sharp features. More result with L1 term are shown in Fig.27. As we can see from Fig.27 in the third column,as the training iterations increases, more sharp features of the ground truth images are generated.



(a) (b)

Fig.26(a) Without L1 term,first 10 iterations(b) With L1 term ,first 10 iterations



(a)first 50 iterations (b)first 200 iterations (c)first 500 iterations

Fig.27 Training result for 50,200,500 iterations with L1 term

The final result for some examples are shown in Fig.28. A plausible reconstruction of the original face can be produced from the network.

(a) (b)

Fig.28 Some result generated randomly from the dataset after 1500 iterations

After the experiment, I found that although the input image is small and blurred, the output of this model could still give a sharp and large reconstructed image. However, the reconstruction can be poor if the picture is not well illuminated or face at an angle,for example, Fig28(b). This may due to the fact that most of the faces in the

dataset are well illuminated or looking straight ahead. Associated with the problems of computer vision mentioned in previous section, it will be more difficult reconstruct images from low resolution to high resolution if there is a difference in illumination or variations in view points.

6 Conclusion

In this work, I have mainly compared three different deep learning models SRCNN,VDSR and DRCN. The proposed model is built based on the methods used in these models. For SRCNN,it works well compared to some traditional models such as sparse coding in terms of training time and performance. However, it does have some limitations such as relatively low convergence and poor performance to different image scale. Then VDSR is introduced as an improvement to SRCNN and it achieves some pretty good result. DRCN reduced the number of parameters in deep models such as VDSR. There are also many other deep learning models which are proved to work well, for example, FSRCNN[14] and DDSRCNN[15]. From the analysis of different models above, it is clear that deep learning models can be well applied to image super resolution tasks and can generate some state-of-the-art result. The performance may yet to be further gained by trying different combinations of layer, filters or new structures.

7 Further work

Apart from deep learning, I have read some papers about the transfer learning that helps to improved the accuracy of models which perform similar tasks. It is interesting

to explore that whether the parameters trained in image classification problems can be applied to improve the result of image super resolution problems. Also, I am trying see if there is any possibility of adding rewards to the deep learning models when training. For example, higher reward is given to the model when output accuracy is relatively high comparing with some benchmarks so that a better learning policy is chosen during training steps.

Appendix

Code source: https://github.com/zceehua/3rd-year-project

Reference

[1]Alec Radford, Luke Metz, Soumith Chintala.Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

[2]Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun Microsoft Research.Deep Residual Learning for Image Recognition

[3]LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.,Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation pp. 541{551 (1989)[4]C. E. Duchon. Lanczos filtering in one and two dimensions. Journal of Applied Meteorology

[5]H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding

[6]Yang, J., Wright, J., Huang, T., Ma, Y.: Image super-resolution as sparse representation of raw image patches.

[7]S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests

[8]C. Dong, C. C. Loy, K. He, and X. Tang. Image superresolution using deep convolutional networks

[9]J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image superresolution via sparse representation.

[10]Accurate Image Super-Resolution Using Very Deep Convolutional Networks.Jiwon Kim, Jung Kwon Lee and Kyoung Mu Lee Department of ECE, ASRI, Seoul National University, Korea.

[11]Identity Mappings in Deep Residual Networks。Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun

[12]Generative Adversarial Nets.Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozairy, Aaron Courville, Yoshua Bengio.

[13]S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In *CVPR*,2015.

[14]Accelerating the Super-Resolution Convolutional Neural Network

Chao Dong, Chen Change Loy, and Xiaoou Tang Department of Information Engineering, The Chinese University of Hong Kong

[15]Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections

Xiao-Jiao Mao, Chunhua Shen, Yu-Bin Yang

[16]http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

[17]Deeply-Recursive Convolutional Network for Image Super-Resolution,Jiwon Kim ,Jung Kwon Lee ,Kyoung Mu Lee

[18]Adjusted Anchored Neighborhood Regression for Fast Super-Resolution (ACCV2014), Radu Timofte etal.

[19]J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image superresolution via sparse representation.

[20]Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift