Super-Resolution Imaging Using Convolutional Neural Networks

Yingyi Sun(✉), Wenhua Xu, Jie Zhang, Jian Xiong, and Guan Gui

College of Telecommunication and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
{1017010607,guiguan}@njupt.edu.cn

Abstract. Convolutional neural networks (CNN) have been applied to many classic problems in computer vision. This paper utilized CNNs to reconstruct super-resolution images from low-resolution images. To improve the performance of our model, four optimizations were added in the training process. After comparing the models with these four optimizations, Adam and RMSProp were found to achieve the optimal performance in peak signal to noise ratio (PSNR) and structural similarity index (SSIM). Considering both reconstruction accuracy and training speed, simulation results suggest that RMSProp optimization in the most scenarios.

Keywords: Super-resolution imaging · Convolutional neural networks · Gradient descent · Adam · RMSprop

1 Introduction

Recent years, with the development of optimized algorithms, mass data, and the increased performance of hardware, deep learning has received a lot of attentions in many aspects, such as super-resolution imaging, image recognition, etc. Among these topics, super-resolution imaging has been deeply studied. The goal of super-resolution imaging is to rebuild high-resolution images from the low-resolution images, which is one of the classic problems in computer vision.

Dong et al. proposed a relative new method for super-resolution imaging based on deep learning [1]. They establish connections between the images with the low and high resolution correspondingly. It is an end-to-end method that generates one high-resolution image as the output from only one low-resolution image as the input. To realize the connections between the inputs and the correspondingly outputs, they use deep convolutional neural networks (CNN) to obtain the mapping between the inputs and outputs.

In this paper, we further study the CNN model [1] and then train our own CNN models with partial optimizations. We find that gradient descent algorithm, which is the core of CNN's weight update, can be optimized by some other strategies in deep learning field. We improve one of the best models which have good performance by these optimizations, such as stochastic gradient descent (SGD), adaptive gradient (AdaGrad), adaptive moment estimation (Adam), and RMSprop. Among these

Q. Liang et al. (eds.), *Communications, Signal Processing, and Systems*, Lecture Notes in Electrical Engineering 516,
https://doi.org/10.1007/978-981-13-6504-1_8

optimizations, we find that RMSprop and Adam have the best performance on peak signal to noise ratio (PSNR), and AdaGrad and SGD are relatively poorer.

The remainder of this paper is organized as follows. Section 2 presents the related works about super-resolution imaging. In Sect. 3, we utilize some optimization algorithms to improve our proposed model. Experimental results are conducted to confirm the proposed method in Sect. 4.

2 Related Works

As a classic problem in computer vision, super-resolution has been studied in various methods. To solve this underdetermined inverse problem, the early methods are mostly based on sampling theory [2–4]. And the limitations that the detailed textures are not as good as expected are then discovered.

Recently, deep learning has been used in various fields, and no exception in super-resolution imaging. Convolutional neural networks have been invented for decades and many computer vision fields, such as pedestrian recognition [5], image classification [6, 7], object detection [8], and face recognition [9].

Dong et al. [1] use their deep CNNs models to rebuild high-resolution images. According to their results, 9-1-5 model has the best score considering performance and running time, under contrast of bicubic interpolation. Their deep CNNs have three convolution layers, each of which followed by an activation layer. The rectified linear unit (ReLU) [10] is used as the activation layer. ReLU makes the model converge faster in the case of high performance. The good performance of the deep CNNs models is shown in their research, with a lot of experimental results as support.

Figure 1 shows the structure of the model. This model has three convolutional layers (yellow), each of which is followed by an activation layer (green) named ReLU, except the third convolutional layer. Mean squared error (MSE) is used as loss function (orange) to calculate the loss between data and label. The light blue blocks represent the input and output data.

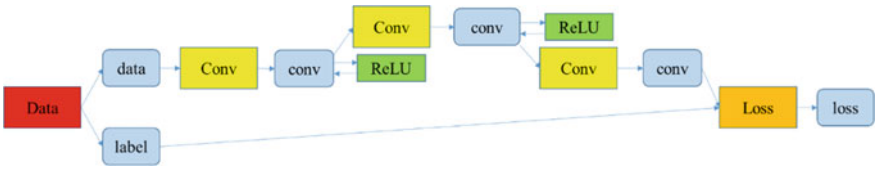

Fig. 1. Model structure of CNNs

3 Optimization Methods

3.1 Model Structure

Dong et al. [1] proposed their 9-1-5 model to train the convolutional neural networks for super-resolution imaging. As shown in Fig. 1, the model contains three

convolutional layers, each of which is connected to an activation layer named ReLU except the third convolutional layer. The operation of ReLU is:

$$O = \max(0, I) \tag{1}$$

where O denotes the output of the ReLU as well as I, the input. Thinking of a convolutional layer and a corresponding activation layer as a pair, the operation of the first two pairs is:

$$Y_i = \max(0, W_i * X_i + B_i) \tag{2}$$

where i equals to 1 or 2 denoting the first or second pair and X_i or Y_i represent the input or output of each pair, respectively. Here, W_i corresponds to the weight coefficient of the filter, as well as B_i the bias coefficient. These parameters like W_i and B_i are achieved by minimizing the loss function, which calculates the mean squared error (MSE) of the reconstructed image and the corresponding high-resolution image. The formula for MSE is:

$$L = \frac{1}{n}\sum_{i=1}^{n} \|X_{ri} - X_{hi}\|^2 \tag{3}$$

where n is the number of training samples, and X_{ri} or X_{hi} represent the reconstructed image or the corresponding high-resolution image, respectively. L is the function of the filter parameters, such as weight and bias coefficients.

3.2 Some Optimization Algorithms

3.2.1 Stochastic Gradient Descent (SGD) Algorithm

Gradient descent (GD) is widely used in deep learning field. In this paper, it can help the CNNs determine the most suitable parameters of the filters as soon as possible, under the premise of good performance. As a result of GD's easily falling into a local optimal solution, this paper replaces GD with SGD. SGD in this article refers to mini-batch SGD. The difference of GD and SGD is only the quantity of the training samples. The training sample in GD is the whole data set, while SGD uses only a part, which calculates some training samples once a time instead of all. Samples for training in SGD are randomly selected at each iteration.

For example, the weight coefficients are updated as

$$W_{i+1} = W_i - \eta \frac{\partial L}{\partial W_i} \tag{4}$$

Here, η is learning rate, L denotes the loss function, and W represents the weight coefficients. The iteration number is denoted by i.

3.2.2 Momentum Algorithm

Momentum algorithm is such a method that it simulates the concept of momentum in physics to optimize the SGD algorithm. The most notable feature is that the last gradient is taken into account in current gradient calculation. It can be expressed as

$$W_{i+1} = \mu W_i - \eta \frac{\partial L}{\partial W_i} \tag{5}$$

Here, the added $\mu \in (0, 1)$ is momentum factor, which controls the effect of the last gradient on the current gradient. The general value of μ is 0.95.

3.2.3 AdaGrad Algorithm

The parameters in algorithms above, such as momentum algorithm, are needed to be set based on experience. Therefore, the determination of specific values of these parameters is seem to be fixed, at most in several common values. Obviously, such setting method cannot be convinced.

AdaGrad algorithm as well as the following methods can set parameters adaptively. After calculating the gradient, AdaGrad algorithm calculates the cumulative squared gradients for the next updating.

3.2.4 RMSProp Algorithm

Considering to the non-convex properties of the CNNs and RMSProp algorithm's adaptive to non-convex, RMSProp algorithm can be used for the optimization. Based on the momentum and AdaGrad algorithms, RMSProp combines the superiority of both two algorithms. It takes into account both the second-order moments of the gradient and the historical effects.

3.2.5 Adam Algorithm

Considering the advantage of AdaGrad and RMSProp on dealing with sparse gradients and non-stationary objectives, Adam algorithm can be seen as a combination of AdaGrad and RMSProp. It makes full use of first and second moments of the gradients to adjust the learning rate of each parameter, respectively.

Adam is superior to the other optimization algorithms in terms of performance. Yet it calculates both the first and second moments of the gradients, the training speed is relatively slower. In spite of this, Adam is still found to be robust in problems with large datasets or parameters with high dimension and is widely applied.

3.3 Comparison of Several Optimization Algorithms

In Sect. 3.2, this paper introduces some optimization algorithms for traditional gradient descent methods. The following part we will discuss these algorithms, such as SGD, AdaGrad, RMSProp, and Adam.

3.3.1 GD and SGD

In GD algorithm, the loss functions need to traverse the entire training data, which is waste of time and is more possible to fall into local optimal solutions. Therefore, in deep learning field like this super-resolution imaging problem, the CNNs model falls into the unwanted solutions easily.

Compared to GD, the samples for training in SGD are randomly selected at each iteration. In this way, we can avoid the local optimal solutions and save much running time.

3.3.2 AdaGrad, RMSProp, and Adam

As shown in the algorithms, AdaGrad makes the use of cumulative squared gradients to minimize the loss functions. As a result, it can make the learning rate adaptive. However, the training speed of AdaGrad is relatively slower.

RMSProp optimizes AdaGrad using the Newton iteration method in the process of accumulating squared gradient. The added coefficient in RMSProp is used to control the acquisition of historical squared gradients.

Adam combines the advantages of both AdaGrad and RMSProp. It can calculate different adaptive learning rates for different parameters and is suitable for big or high-dimensional data. Beside these, Adam needs smaller requirements for memory.

The detailed performance comparisons of these algorithms are discussed in Sect. 4.

4 Experiments

In this section, we first discuss the determination of training data and label. Next, we add the four optimizations discussed in Sect. 3 to the model and compare the performance of them, respectively. The last but not least, we examine the training loss and speed of the model with four optimizations.

4.1 Training Data

To train our model for super-resolution imaging, we need to determine the data and label in training sets. The purpose of our model is to rebuild the images, so the super-resolution images generated from low-resolution images by CNNs are seen as data. In order to continuously improve the performance of the model, it needs the label as contrast. The corresponding high-resolution images are used as label, which is the same as data as the input of the loss layer. Mean square error (MSE) is used as a loss function in this paper.

To realize the process above, the size of data and label should be calculated ahead of time. For example, the sizes of these three types of convolution kernels in the 9-1-5 model are set to 9 by 9, 1 by 1, and 5 by 5, respectively. With the stride of 1, the size of data and label can be 33 by 33 and 21 by 21, respectively.

4.2 Comparison of the Performance

Figure 2 shows the performance of bicubic reconstructed image and another four images with optimizations. It displays one original image and five reconstructed images as contrast. The first image is the original image, which is grayed out. The reconstructed image below the original one is using bicubic interpolation, which is obviously fuzzy. The right four images are clearer, which reflects on the effects of these optimizations.

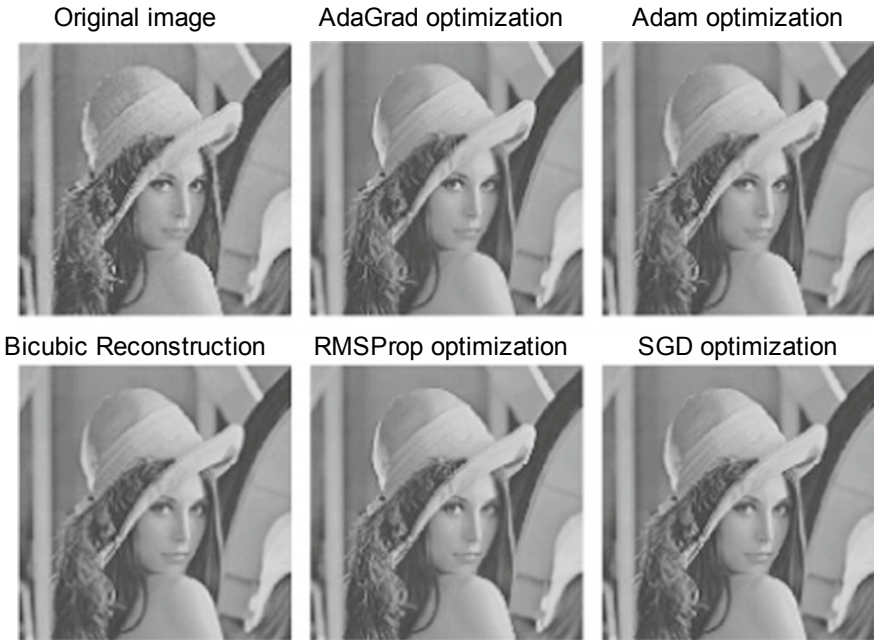

Fig. 2. Original and five reconstructed images

Table 1 displays the specific PSNR and SSIM of five reconstructed images. It is clear that Adam and RMSProp achieve the highest PSNR 33.30228 and 33.10496 dB, respectively. As a result, Adam is the best choice in terms of performance and the

Table 1. PSNR and SSIM of five reconstructed images

Type	PSNR (dB)	SSIM
Bicubic	31.67762	0.859749
AdaGrad	32.20875	0.867244
Adam	33.30228	0.882372
RMSProp	33.10496	0.879238
SGD	32.42305	0.874023

performance of RMSProp is slightly worse. It shows that Adam has the highest scores of SSIM with 0.882372 while RMSProp ranks second with 0.879238, followed by SGD and AdaGrad, respectively.

4.3 Comparison of the Training Process

Section 4.2 discusses the performance of the model with four optimizations, respectively. Yet considering performance alone to choose the most suitable model cannot be convinced. Thus, the training process is also recorded to be the basis for selecting the most suitable model.

Figure 3 describes the changes in training loss of the models with optimizations like AdaGrad, Adam, RMSProp, and SGD during the training process. It is obvious that AdaGrad and Adam have the minimal training loss after convergence.

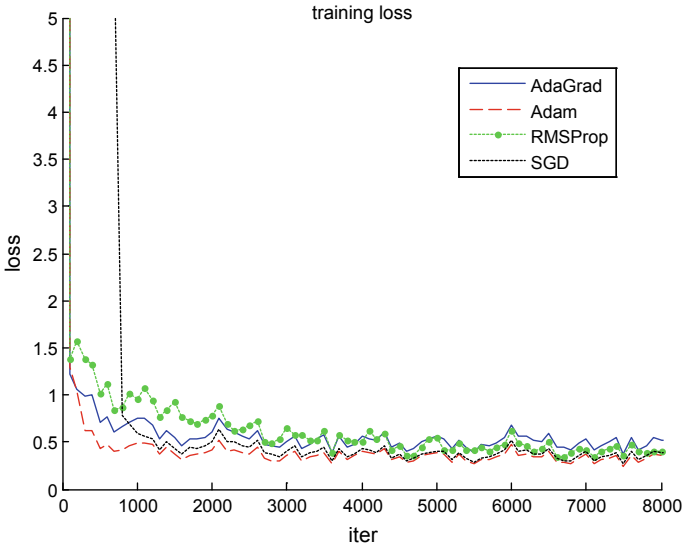

Fig. 3. Comparison of the training loss of the four models

In addition, Adam has another advantage that it makes the model convergences faster. As a result, considering the performance and effect of convergence, Adam is superior to other optimizations.

Table 2 displays the accurate values of running time for every 100 iterations. As described in Fig. 3, the fastest training speed for RMSProp and SGD is about 52.1928 and 51.85337 seconds for every 100 iterations. In contrast, Adam's training speed is two to three times slower. As a result, considering the performance, speed of convergence, and training time, the model with RMSProp optimization is the most suitable.

Table 2. Average time required for 100 iterations

Type	Time (s/100 iterations)
AdaGrad	272.9645
Adam	130.0681
RMSProp	52.1928
SGD	51.85337

5 Conclusion

In this paper, we first utilized the convolutional neural networks to reconstruct the super-resolution images from the corresponding low-resolution images. Then, we introduced four optimization algorithms and discussed the relationships between them. At the end, we analyzed the performance and training process of the models with these four optimizations, and we found that in the case of all these factors, RMSProp may be the most suitable optimization algorithm for the super-resolution imaging.

References

1. Dong C, Chen CL, He K, Tang X. Image Super-Resolution Using Deep Convolutional Networks. IEEE Trans Pattern Anal Mach Intell. 2016;38(2):295–307.
2. J. Allebach and P. W. B. T.-I. C. on I. P. Wong 1996. Proceedings, "Edge-directed interpolation," *Int. Conf. Image Process.*, vol. 3, no. 3, pp. 707–710, 1996.
3. Li X, Orchard MT. New edge-directed interpolation. IEEE Trans Image Process. 2001;10 (10):1521–7.
4. Zhang L, Wu X. An edge-guided image interpolation algorithm via directional filtering and data fusion. IEEE Trans Image Process. 2006;15(8):2226–38.
5. Ouyang W, X. B. T.-I. I. C. on Wang CV. Joint deep learning for pedestrian detection. In: IEEE international conference on computer vision; 2014. p. 2056–63.
6. He K, Zhang X, Ren S, Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans Pattern Anal Mach Intell. 2015;37(9):1904–16.
7. Krizhevsky A, Sutskever I, GEBT-IC on Hinton NIPS. ImageNet classification with deep convolutional neural networks. In: International conference on neural information processing systems; 2012. p. 1097–105.
8. Ouyang W, Zeng X, Wang X, Qiu S, Luo P, Tian Y, Li H, Yang S, Wang Z, Li H. DeepID-Net: deformable deep convolutional neural networks for object detection. IEEE Trans. Pattern Anal. Mach. Intell. 2014;99:2403–12.
9. Sun Y, Wang X, Tang X. Deep learning face representation by joint identification-verification, vol. 27; 2014. p. 1988–96.
10. Nair V, GEBT-IC on IC on Hinton ML. Rectified linear units improve restricted boltzmann machines. In: International conference on international conference on machine learning; 2010, p. 807–14.