Research on Image Up-scaling and Super-resolution Based on Convolutional Neural Network

Zongxi Cheng*

Graduate School of Art and Science, Boston University, Massachusetts, US, 02215

*Corresponding author: zongxialcheng@gmail.com

Abstract. Image upsampling and super-resolution techniques are of great significance in many fields, which can improve image quality and enhance information extraction capabilities. Image upsampling and super-resolution techniques have wide applications in the real world, such as image quality enhancement, satellite and remote sensing images, and security and surveillance. This paper describes and analyzes a machine learning approach to up-scaling a low resolution images using Convolutional Neural Networks. We build on previous works on single image super-resolution. We train 3 structurally similar but different models and obtain an improvement of up to 3% compared to the more common image upscaling method, Bicubic Interpolation.

Keywords: Image Up-scaling, Image Super-resolution, Computer Vision, Convolutional Neural Network.

1. Introduction

The problem of up-scaling an image is a classic problem in computer science [1]. Before the advent and success of Neural Networks, solutions to these problems have been restricted to taking some weighted average of neighboring pixels, in some form or another [2]. We try to solve the problem of upscaling low-resolution images, using Convolutional Neural Networks which will learn a non-linear mapping from low-resolution images to high-resolution images [3].

Previous approaches to this problem either try to exploit similarities within the same imageor learn mapping from pairs of high-resolution and low-resolution image [4].

One of the most representative methods for paired-image-based image up-scaling is the sparse-coding-based method [5]. It involves several steps including preprocessing data, overlapping patches, encoding patches by dictionaries, and optimizing the dictionaries. The approach is not straightforward and the optimization of the model is relatively hard to accomplish. Another paper [6], using Convolutional Neural Networks(CNN) produces an end-to-end mapping between paired low resolution and high-resolution images, instead of the pipeline that [6] used. We decided to use the approach in [7], reproduce the work and make optimization to the model.

We want our models to produce better quality images compared to the more common image up-scaling scheme, Bicubic Interpolation. We will be using the Mean Squared Error (MSE) to evaluate the accuracy and quality of predicted images [8].

2. Background

In this section, we describe some important concepts that are useful to understand this paper and finally discuss our data pre-processing.

2.1. Bicubic Interpolation

Bicubic interpolation is used extensively in our work [9]. It is an extension of cubic interpolation for interpolating data points on a two-dimensional regular grid [10]. In image processing, it is widely used to do image resampling since the images it produces are smoother and have fewer interpolation artifacts than other approaches like bilinear or nearest-neighbor interpolation. In this project we used it to re-size images to appropriate dimensions during the our data pre-processing stage.

2.2. Keras with TensorFlow backend

In this project we utilize Keras AP to build and train our models [11]. Keras is a high level neural network API written in Python and capable of running on top of TensorFlow [12], CNTK, or Theano back-ends. It includes plenty of ready to use functionality that helps people to utilize the power of TensorFlow. TensorFlow is one of the most famous systems for deep learning that operates at large scale and in heterogeneous environments. It uses dataflow graphs to represent computation, and allows mapping nodes of a dataflow graph across many machines or many computational devices within a machine. It gives flexibility to the application developers by enabling them to experiment with novel optimization and training algorithms.

2.3. Convolutional Neural Networks

Despite the many representative methods for image super-resolution, we were using an end-to-end Convolutional Neural Networks to attack this problem. A Convolutional Neural Network, also known as CNN, is a class of deep, feed-forward Artificial Neural Network(ANN). CNN has gained popularity recently because its high performance in image classification problems. Of several keys reasons to its success, 1) The success in implementing and training on modern powerful GPUs; 2) The speed and quality provided by Rectified Linear Unit (ReLU); The abundant data available ready for training large and complex models make CNN very attractive for image related problems. For further discussion on CNN please refer to [1]. In this project we explore variations of the CNN structure described.

2.4. Data Pre-processing

We found the training set used in the works of [13], which consisted of 91 images of varying dimensions. From each image, we extract sub-images of dimensions 78px × 78px, to obtain a total of 741 sub-images of equal dimensions. We did this to avoid dealing with a lot of data at once, which can be very memory-intensive during model training.

We perform the following operations on each sub-image to produce a pair of high-resolution and low-resolution image. A sub-image, X, is re-sampled to be a $\frac{1}{3}$ of it's original size. All our experiments use a magnification of size 3, but all our methods can be extended to any magnification size. The reduced image is then scaled up to iitsoriginal size using bicubic interpolation – call this result X'. The original subimage, X and the pre-processed sub-image, X' form a training set pair. Even though both X and X' ve the same size, we refer to sub-image X' as low-resolution. This is the only data pre-processing step.

3. Models and Experiments

First we will describe our different models, and then specify our experiments.

3.1. Models

We structured our models following the same overall organization described in the works of (Dong et al. 2014), which consists of 3 main stages. The first stage is Patch extraction, which conceptually extracts patches of features from the input image. The second stage, Non-linear Mapping, introduces non-linearity into the mapping our models are learning. It simply applies a non-linear function to the extracted features. The third stage is Image Reconstruction. This stage reconstructs our final image from the non-linearly mapped features. Following this general structure, we have experimented with 3 different models that are described below.

The 915 Model is the model used in the paper we are trying to replicate. This model has 3 convolutional layers. The first layer (corresponding to Patch extraction) consists of 64 convolutional filters of size 9 with ReLu activation. The second layer consists of 32 convolutional filters of size 1 with ReLu activation. This layer essentially takes the positive part of the inputs from the previous

layers. The Reconstruction stage has 3 convolutional filter, each corresponding to the 3 color channels (RGB). Each convolutional filter of this layer has size 5 with linear activation. This stage is considered as an averaging step to stitch together the extracted feature patches, which is why a linear activation makes sense here. This model uses layers with convolutional filters of size 9,1, and 5 in order, which is why it's named the 915 Model.

The 5515 Model is almost identical to the 915 Model, but replaces the the feature extraction stage with 2 convolutional layers. The first layer consists of 128 convolutional filters of size 5 with ReLu activation, followed by 64 convolutional filters of size 5 again with ReLu activation. The last two layers stay exactly the same as the 915 Model

The 75315 Model is a 5 layer model. This model uses the first 3 layers for the feature extraction stage. The first layer consists of 96 convolutional filters of size 7, followed by 72 filters of size 5, and finally a set of 48 filters of size 3. All filters in the first 3 layers use ReLu activation. The last 2 layers stay exactly the same as previous models.

3.2. Experiments

We built and run all models on Python 3, using the neural network package, Keras with TensorFlow back-end. We train each model on many sub-images, each with size 78px × 78px. The choice of the size of the sub-images is purely artbitrary and was largely motivated by the capability of the machines we used for training. We used the train test split function from Sklearn package (Pedregosa et al. 2011) to split our data set into training and test set with ratio of 4:1. Generally, we run each model for at least 100 epochs, which we found was enough for convergence. Next we describe, the optimizer and the loss function used.

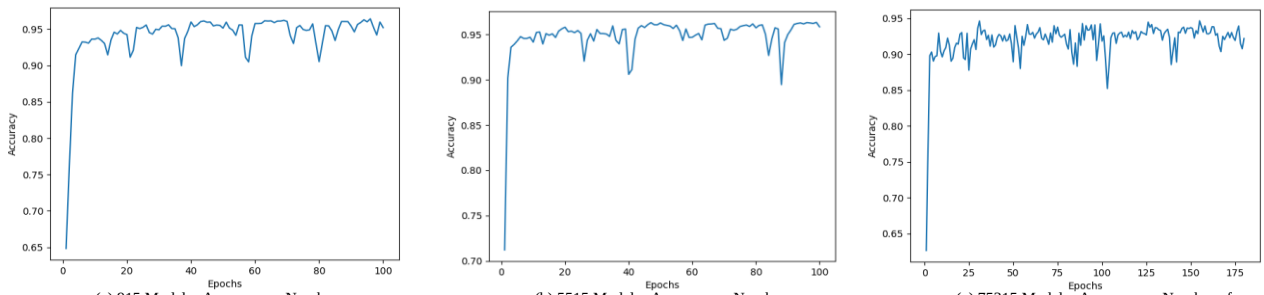
We used the Mean Squared Error (MSE) as the loss function, which has a better correlation with a signal quality metric called Peak Signal to Noise Ratio (PSNR). All the convolutional layers have no padding to avoid border effects during training. As a result all out outputs are smaller than the input. For example, the output of a 78px × 78px image using the the 915 Model will be a 72px × 72px image. The MSE loss function is evaluated only using the differences in the central pixels.

We used the Adam optimizer from Keras package with its default configuration for all our training. We have tried other optimizers from Keras package, such as Adagrad, Adadelta, Nadam, and SGD, but cause all our models to diverge. While we tried to tune these optimizers to get our models to converge, we didn't spend significant time configuring them. Thus, spending some more time configuring optimizers might cause our models to converge, giving us options of optimizers to choose from.

4. Results

4.1. Quality Measure

As described in the previous section, the MSE is used as the loss function. Figure 1 below show the progress of the normalized MSE as the number of epochs the specific model is trained over is increased. In the Figure 1 below we display Accuracy, which is just the value obtained from 1 − Normalized MSE.



(a) 915 Model – Accuracy vs Number of epochs

(b) 5515 Model – Accuracy vs Number of epochs

(c) 75315 Model – Accuracy vs Number of epochs

Figure 1. Comparison of recognition effects of three models.

It is interesting to note that all the models converge very quickly. Therefore, it might be a good idea to train a more complex model, but for fewer epochs. An example of what our models can produce is given in Figure 2. While Bicubic interpolation produced images with average normalized MSE of 12%, the average error rates was about 9% for images produced by all the CNN models. While there is no significant quantitative difference in quality of images produced by the 3 models, people found the outputs of the deeper models more visually appealing. Although the observations collected from people's comments are not very scientific, there seem to be marginal gains in perceptual quality when we increase the depth of our network. Although the Accuracy of the 75315 Model is below the other models during training, it seems to generalize a little better. For example, all our models introduce some noise when the image we are trying to predict has several black patches. As you can see from Figure 3, the output from the 75315 Model introduces the least noise. For this specific image of a Zebra, we get a normalized MSE of 15.11%, 21.16%, 26.16%, and 17.85% for bicubic interpolation, 915, 5515 and 75315 respectively. Although the previous work (Dong et al. 2014) concluded that deeper is not better, we seem to be getting (marginally) better results (introduces less noise).

4.2. Learned Filters

Looking at the filters learned in the first layer for each of our models (Figure 4), we can see an interesting pattern. The filters learned by 915 Model shows little diversity – there are filters that capture a lot of detail and filters that capture very little detail, and small number of filters that capture specific patterns. In contrast, the other 2 models show a little more diversity, the 5515 Model being the most diverse. We believe the main reason for this is the filter size used. The smaller the filter size, the more detail the layer captures. Therefore, it might be a good idea to start with smaller filter size to capture smaller local details first, and then increase filter size layer by layer to incorporate a more global detail of the image.

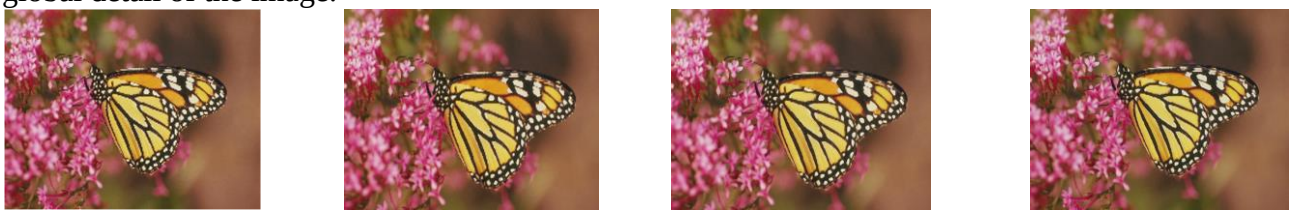


Figure 2. Output of different Models on the Same input: Bicubic interpolation, 915 Model, 5515 Model, 75315 Model, in order from left to right.

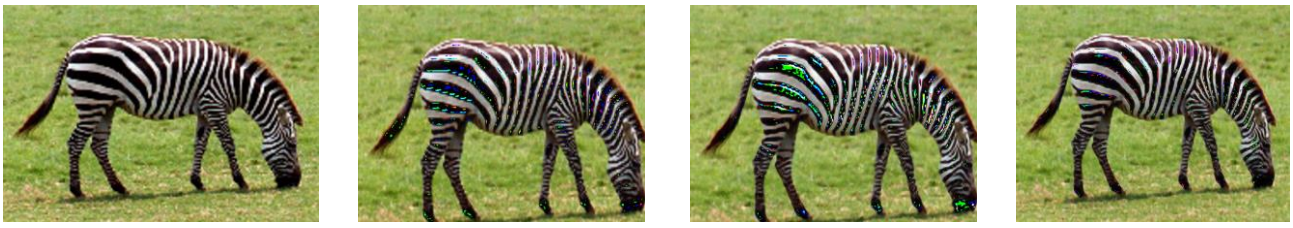


Figure 3. Output of different Models on the Same input: Bicubic interpolation, 915 Model, 5515 Model, 75315 Model, in order from left to right.

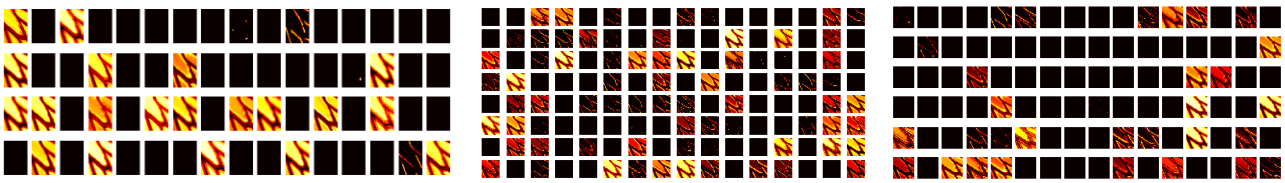


Figure 4. Convolutional filters learned in first layer, 915 Model, 5515 Model, 75315 Model, in order from left to right.

5. Conclusions

We build on previous works on deep learning and image super-resolution, by successfully reproducing the model described, and examining variations of that model. Our models learned an end-to-end mapping between the paired images with nearly complete absence of pre/post-processing of data. While training these models is slow (using Google Colaboratory improves training time significantly), up-scaling an image after the model is trained is fast. Thus, these models produce decent quality (up to 3% improvement from bicubic interpolation) without sacrificing speed.

Our modification to the number and size of each layer succeeded in yielding two better models. While these improvements are subtle, they do provide us with a direction for an improvement. Based on our observations, we believe that adding more layers to the described structure will make the network less prone to introducing noise and generalize a little better. It is also worth noting that the number of filters we choose at each level were completely dictated by the feasibility of training that model in the time frame we had. For example, when we reduce the number of filters in the first layer of the 75315 Model to 96, from the 128 filters in the 5515 Model, it was purely to make the training period feasible. Therefore, we also suggest careful experimentation with the number of filters in a layer (width of the network) as well. We have shown that our models converge very quickly with marginal improvement after 20 epochs. Therefore, it is possible to experiment with more complex models with variations in both depth and width with lower training epochs. Adding more training examples that are likely to induce noises (such as images with significant black or monochromatic patches) might also reduce the amount of noise introduced. We spent very little time configuring the optimizers we used. While this may turn out to be inconsequential, careful optimizer tuning might also improve the results.

Based on observations of filters from the first layers of various models, we believe that starting with small-size convolutional filters in the earlier layers, and progressively increasing filter size might yield better results. Further work and experiments need to be done to test these hypothesis.

References

[1] Yang W, Zhang X, Tian Y, et al. Deep learning for single image super-resolution: A brief review[J]. IEEE Transactions on Multimedia, 2019, 21(12): 3106-3121.

[2] Zhu X, Guo K, Fang H, et al. Cross view capture for stereo image super-resolution[J]. IEEE Transactions on Multimedia, 2021, 24: 3074-3086.

[3] Lan R, Sun L, Liu Z, et al. MADNet: A fast and lightweight network for single-image super resolution[J]. IEEE transactions on cybernetics, 2020, 51(3): 1443-1453.

[4] Shou Y, Meng T, Ai W, et al. Conversational emotion recognition studies based on graph convolutional neural networks and a dependent syntactic analysis[J]. Neurocomputing, 2022, 501: 629-639.

[5] Xue J, Zhao Y Q, Bu Y, et al. Spatial-spectral structured sparse low-rank representation for hyperspectral image super-resolution[J]. IEEE Transactions on Image Processing, 2021, 30: 3084-3097.

[6] Shou Y, Meng T, Ai W, et al. Object Detection in Medical Images Based on Hierarchical Transformer and Mask Mechanism[J]. Computational Intelligence and Neuroscience, 2022, 2022.

[7] Shou Y, Cao X, Meng D. Masked Contrastive Graph Representation Learning for Age Estimation[J]. arXiv preprint arXiv:2306.17798, 2023.

[8] Tian C, Xu Y, Zuo W, et al. Coarse-to-fine CNN for image super-resolution[J]. IEEE Transactions on Multimedia, 2020, 23: 1489-1502.

[9] Ying R K, Shou Y, Liu C. Prediction Model of Dow Jones Index Based on LSTM-Adaboost[C]//2021 International Conference on Communications, Information System and Computer Engineering (CISCE). IEEE, 2021: 808-812.

[10] Zhu X, Guo K, Ren S, et al. Lightweight image super-resolution with expectation-maximization attention mechanism[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2021, 32(3): 1273-1284.

[11] Meng T, Shou Y, Ai W, et al. A Multi-Message Passing Framework Based on Heterogeneous Graphs in Conversational Emotion Recognition[J]. Available at SSRN 4353605, 2021.

[12] Shou Y, Cao X, Meng D, et al. A Low-rank Matching Attention based Cross-modal Feature Fusion Method for Conversational Emotion Recognition[J]. arXiv preprint arXiv:2306.17799, 2023.

[13] Dong W, Zhou C, Wu F, et al. Model-guided deep hyperspectral image super-resolution[J]. IEEE Transactions on Image Processing, 2021, 30: 5754-5768.