

---

# Image Super Resolution

---

Csaba Kilian

## Abstract

Using artificial neural networks to upscale images create a major improvement over traditional image interpolation algorithms. This paper compares existing approaches using autoencoders and residual networks and aims to combine ideas from both to create a new model whose accuracy has significantly improved over existing image interpolation algorithms. Using an input image (224x224x3) of birds we create a low resolution counterpart (56x56x3) using nearest neighbours interpolation and our model tries to upscale it by a factor of 4 on each axis. Our model achieves 0.0031 mean squared error score, and approximately 26.01dB peak signal to noise ratio - 29.5% improvement over bilinear interpolation.

## 1 Introduction

In the last few years, computational power of GPUs have increased to the point where it's feasible to use neural networks rather than traditional scaling algorithms to upscale image data. Neural network architectures have made great improvements over traditional re-sampling algorithms such as nearest neighbours, bilinear, bicubic or even Lanczos. The purpose of this project was to test recently published architectures and how they compare with traditional algorithms in upscaling an image 4x in both directions, and to create a more accurate model that improves on previous results. The models tested contained autoencoders, deep convolutional networks with skip layers, dense residual networks and generative adversarial networks.

The best tested model chosen was a deep convolutional network with 13 convolutional layers of a progressively increasing and then decreasing kernel size and 3 long skip connections. The rationale for this is so that information is carried over from the encoding phase into the decoding phase and to prevent the gradients from vanishing. Generative adversarial networks and deep residual networks were also implemented but required significantly more computational power than was available, more training time and did not yield good results in the 1st 40 training epochs used for model selection.

Mean squared error was used to compare the models and the best model was trained until it achieved a comparable score with more powerful architectures.

## 2 Related Works

Our hypothesis was to use autoencoders to learn how to upscale an image (super resolution). Autoencoders used for super resolution have a nice property that they tend to converge very fast [1]. To improve accuracy of reconstructed images, we investigated how skip connections can be used to carry information from the encoding phase into the decoding phase so that the network learns the differences between the low resolution image and the reconstruction [2].

This was motivated by exploring the use of residual dense networks for super resolution [3]. Their approach is to use residual blocks for learning the residuals,  $F(x) - x$  where  $F$  is a feature mapping, both globally and locally. Each residual block contains many convolutional layers and after each convolutional layer, the outputs of all previous layers are concatenated. These residual blocks are arranged sequentially and the global feature mapping is learnt by concatenating the output of each

residual block before upsampling. This way the network keeps track of the input states at each stage and effectively is focused on learning the residuals between an input and a feature mapping [3].

Major improvement has been the use of generative adversarial networks. Super resolution GANs differ from the previous techniques in 2 ways: they train 2 different neural networks, a generator that transforms low resolution into a high resolution image and a discriminator network that is presented with real and generated high resolution images and tries to discriminate between the 2 classes. The discriminator is rewarded for accurately distinguishing the fakes from the real ones whilst the generator is rewarded for being able to fool the discriminator network. The other major difference is that GANs incorporate perceptual loss (such as features extracted from a pre-trained image classifier such as VGG net) into their loss functions. The generator's loss function becomes a weighted sum of pixel-wise euclidean distance loss (mean squared error, etc.), the negative log of the probability that the generated image is real, and the euclidean distance between features extracted from the generated image and the reference image [4].

Residual dense network and generative adversarial networks were both implemented but not used for model selection due to the computational complexity and training time required. However, residual dense networks informed our decision to enhance autoencoders with skip connections.

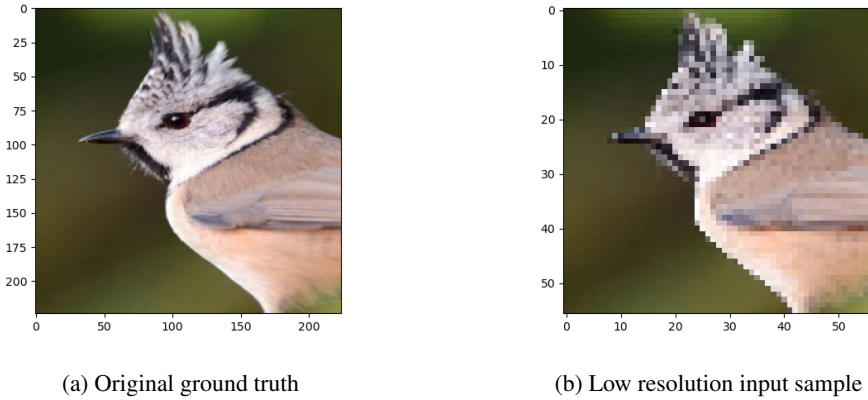


Figure 1: Corresponding ground truth and input into the network

### 3 Data

The data set used is a collection of 39209 images of 260 species of birds (224 x 224 x 3) [5]. This data set is used because many bird species have densely packed, colourful and reflective feathers and so a lot of information is lost during downsizing. In addition, because they fly most of the pictures are presented against a uniform background. For this super resolution learning task, we're testing how well can the model recognize certain objects against a backdrop and generate a high resolution and accurate reconstruction.

The images were further flipped horizontally at random. This was done to augment the training samples [6]. The transformation was applied during batch loading time, so the data set was extended rather than modified in place. In addition, pixel values were linearly scaled to be between 0 and 1 in order to reduce burn in time and prevent gradient explosion.

While our original proposal called for creating low resolution samples via lossy JPEG compression, this proved time consuming with 10000's of images. Instead, we simulated the low resolution image by adding a preprocessing layer to every model that scales the input image by 1/4 using the worst possible - nearest neighbour interpolation. Figure 1a shows the original high resolution image and 1b shows the corresponding input into the network.

## 4 Methodology

In the initial model selection phase, 2 classes of models were created: reference pseudo-models using only traditional interpolation methods to upscale the low resolution image, and neural networks.

The reference models that we compare our deep learning methods against are Nearest Neighbour, Bilinear, Bicubic and Gaussian interpolations. They only upscale a low resolution image by 4 in each axis using their respective kernel.

For the deep learning models, our method was to create a relatively shallow autoencoder, and add layers to increase accuracy. Model A uses 8 convolutional layers, 2 max pooling and 2 up sampling layers. Our approach was to upscale the image using a traditional interpolation algorithm and then feed it through many convolutional layers.

Model B progressively also added 2 skip-connections, both before max pooling operations to after the up-sample operations. Our hypothesis is that the network would learn the residuals between the encoder and the decoder layers.

Model D is an improvement on model B and uses more convolutional layers and 3 skip connections. A major difference is that the initial up-sampling before the encoding layers is done using bilinear instead of nearest neighbours approximation. Bilinear (based on mean squared error) is about 2x as accurate as nearest neighbours and is a better starting point for the network to learn from.

Other models that were considered as a basis was Super Resolution Generative Adversarial Network (SRGAN) [4], and Residual Dense Network for Image Super-Resolution (RDN) [3]. SRGAN has been found to be computationally prohibitive, and RDN could not be reproduced using our dataset to the same accuracy (at least 35dB peak signal to noise ratio) that the paper claimed.

All the models were run for 40 epochs using Adam optimizer with a learning rate of 0.01 with a pre-seeded random number generator. 5% of the training set was reserved for validation.

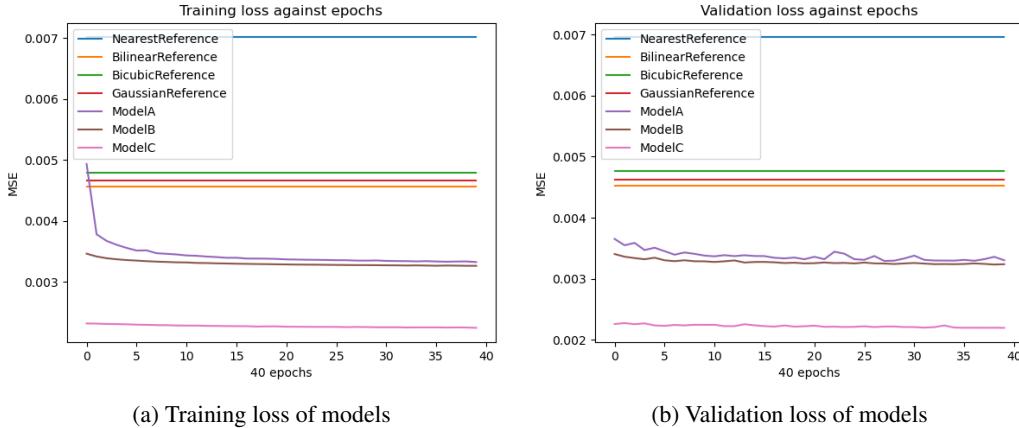


Figure 2: Corresponding ground truth and input into the network

Model D which is an extension of model B was chosen for a longer training run (Model C was lost and could no longer be recreated). It was trained for 200 epochs using a batch size of 8 and learning rate scheduling decreasing 10 fold / 50 epochs. It was evaluated on a separate test set.

## 5 Results

Model D produced a substantial improvement in mean squared error as well as perceptually. Model D yielded a mean squared error of 0.0031 on the test set. In contrast to the bilinear upscaling (MSE: 0.0044), this is 29.5% improvement. The peak signal to noise ratio (PSNR) achieved by model D is approx. 26dB.

Tables 2 and 3 show 4 test samples. Although this is subjective, the model has perceptually outperformed bilinear upscaling. Despite that the initial layer of Model D is a bilinear upscaling, the model

Number	Layer	input dim	output dim
1	UpSampling (bilinear)	56x56x3	224x224x3
2	Conv. 2d - kernel size: 3x3, stride: 1, padding: 1,	224x224x3	224x224x32
3	Conv. 2d - kernel size: 3x3, stride: 1, padding: 1,	224x224x32	224x224x64
4	Conv. 2d - kernel size: 3x3, stride: 1, padding: 1,	224x224x64	224x224x64
5	Max pooling - kernel size: 2x2, stride: 2, padding: 0	224x224x64	112x112x64
6	Conv. 2d - kernel size: 3x3, stride: 1, padding: 1,	112x112x64	112x112x128
7	Conv. 2d - kernel size: 3x3, stride: 1, padding: 1,	112x112x128	112x112x128
8	Max pooling - kernel size: 2x2, stride: 2, padding: 0	112x112x128	56x56x128
9	Conv. 2d - kernel size: 3x3, stride: 1, padding: 0,	56x56x128	56x56x256
10	Conv. 2d - kernel size: 5x5, stride: 1, padding: 2	56x56x256	56x56x256
11	Conv. 2d - kernel size: 5x5, stride: 1, padding: 2	56x56x256	56x56x256
12	Conv. 2d - kernel size: 3x3, stride: 1, padding: 0	56x56x256	56x56x256
13	Add layers (9 + 12)	56x56x256	56x56x256
14	Conv. 2d Transpose - kernel size: 3x3, stride: 1, padding: 2	56x56x256	112x112x256
15	Conv. 2d - kernel size: 3x3, stride: 1, padding: 0	112x112x256	112x112x128
16	Conv. 2d - kernel size: 3x3, stride: 1, padding: 0	112x112x256	112x112x128
17	Add layers (6 + 16)	112x112x128	112x112x128
18	Conv. 2d Transpose - kernel size: 3x3, stride: 1, padding: 2	112x112x128	224x224x128
19	Conv. 2d - kernel size: 3x3, stride: 1, padding: 0	224x224x128	224x224x64
20	Conv. 2d - kernel size: 3x3, stride: 1, padding: 0	224x224x64	224x224x32
21	Add layers (2 + 20)	224x224x32	224x224x32
22	Conv. 2d - kernel size: 1x1, stride: 1, padding: 0	224x224x32	224x224x3

Table 1: Model D architecture

can smooth out object edges really well and remove pixelation. The model predictions also contain 2 observed peculiarities:

1. The model paints in circular spots when trying to reconstruct highly detailed and granular textures (Tables 2 and 3)
2. The model is unable to reconstruct the light reflection on the eyes of birds (Tables 2 and 3)

## 6 Discussion

Adding more convolutional layers were found to increase accuracy and skip connections greatly helped in making the model learn the residuals between a bilinear upsampling and the ground truth. Despite the better accuracy of using skip connections, using our implementation of Residual Dense Network only achieved approximately 6 PSNR in the 1st 40 training epochs - substantially worse than even nearest neighbours interpolation. A common feature of other networks that we tried implementing is that upsampling is done in the last layer before the output. This decreases model complexity but greatly decreases MSE scores in the initial training epochs. In contrast, upsampling before the autoencoder phase provides really high gains in PSNR even in the 1st epoch. The question is whether the quick gains are offset by a limit on MSE decreases in the later training epochs. After all rather than learning from the low resolution sample, the autoencoder starts off with an upscaled version of the low resolution sample.

Using Adam optimizer was found to converge the fastest to below 0.0040 mean squared error. Learning rate scheduling was found to marginally improve mean squared error from 0.0040 to 0.0031.

Although the model is really good at reconstructing edges, its ability to reproduce fine details in texture is lacking.

As mentioned in the previous section, it is seldom able to reconstruct light reflection on the eyes of birds. It can be seen in the examples that light reflecting on the eyes are lost in the low resolution samples. One possible remedy for this would be for the model to augment the low resolution image by adding random Gaussian noise after the upscaling layer but before the convolutional layers.

Another weakness of the model is reconstructing granular detailed textures. Because the details are lost in the low resolution sample, the model seems to extrapolate from the pixelated texture in the







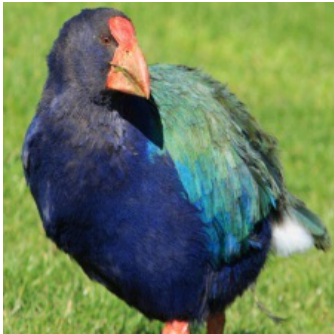

low resolution input		
bilinear upscaling		
model prediction		
ground truth		

Table 2: Test samples 1, 2

bilinear upscaling layer and just increases the density of the pixelation (ie. it replaces the coarse grained pixelation with fine grained dots), but is unable to learn different textures as in test sample 2.

In order to improve textural detail reconstruction there are a few methods to consider for the future:

1. Do not upscale an image to 4 times the low resolution size directly but construct a model that upscales to 2 times the low resolution size and a 4x upscaling thus becomes  $\hat{y} = F(F(x))$ .
2. Create a second autoencoder network  $G(x)$  that randomly takes small patches from the training samples (ie.  $5 \times 5 \times 3$ ), upscales them, adds Gaussian noise, and then learns to predict the target patches. Its prediction function would be to take every  $5 \times 5$  patch with a  $5 \times 5$  stride from the entire input image, and individually upscale, then combine them into a high resolution image.
3. Combine mean square error with a perceptual loss function such as using the feature map of a deep layer of VGGNet





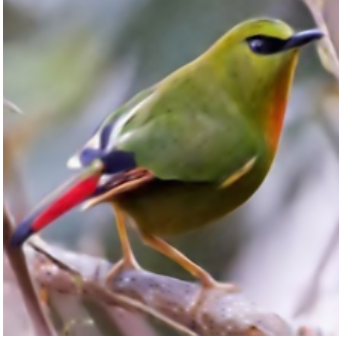
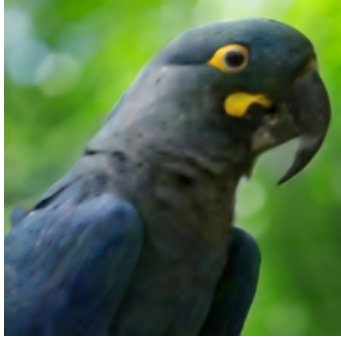


low resolution input		
bilinear upscaling		
model prediction		
ground truth		

Table 3: Test samples 3, 4

## 7 Conclusion

Our model using 16 convolutional layers and 3 skip connections connecting the encoding and the decoding phase has achieved significant improvement over traditional interpolation algorithms.

Although image super resolution is undergoing promising advances using generative adversarial networks, autoencoders can also obtain good (although not comparable) results using sufficiently deep convolutional layers. Super resolution using artificial neural networks is an innovative way to upscale and restore old pictures or films especially when the film negative is no longer available.

It is important to point out that the network is effectively trained to look at a low resolution image and "imagine" (extrapolate) a high resolution based on that. It would be unethical to use this technique to upscale low resolution footage from security cameras and make predictions because the reconstruction is based on training data taken from a limited sample space.

Nevertheless, for restoring old videos and pictures, artificial neural networks offer groundbreaking improvements over traditional image interpolation algorithms.

## 8 References

- [1] A. Palmer, "Single Image Super Resolution of Textures via CNNs", University of Waterloo, Waterloo, Ontario. Available: <https://cs.uwaterloo.ca/mli/palmer-super-resolution.pdf> [Accessed: 02-April-2021].
- [2] K. He, et al. "Deep Residual Learning for Image Recognition", Microsoft Research, 2015. [Online]. Available: <https://arxiv.org/pdf/1512.03385.pdf>. [Accessed: 03-April-2021].
- [3] Y. Fu, et al. "Residual Dense Network for Image Super-Resolution", [Online]. Available: <https://arxiv.org/pdf/1802.08797.pdf>. [Accessed: 03-April-2021].
- [4] A. Acosta, et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network", Twitter [Online]. Available: <https://arxiv.org/pdf/1609.04802.pdf>. [Accessed: 04-April-2021].
- [5] "265 Bird Species" [Online]. Available: <https://www.kaggle.com/gpiosenka/100-bird-species>. [Accessed: 25-March-2021].
- [6] J. Brownlee. "How to Configure Image Data Augmentation in Keras" [Online]. Available: <https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/>