Super-Resolution using Deep Learning

CSC420H1: Project Report (CosmicMangos)

Michal Fishkin
Siddhant Jain
Robert Wu

June 11, 2021

Abstract

Images with higher resolutions capture and preserve more details, and are usually easier to work with. While data and information are valuable, so are bandwidth and storage. Sometimes images are of lower resolution than we'd like, like in photographs taken by low-resolution cameras and compressed images that are more suitable for fast data transfer. Benefits of lower resolutions include higher framerates, lower latency and less expensive storage. Unfortunately, as a natural consequence, these lower-resolution images almost always contain less information than their higher-resolution counterparts.

Introduction

In order to work with more information, it's useful to produce an image with higher resolution, what we refer to as "super-resolution" (SR). This task allows you to extract more information than was in the original images and hopefully produce a more pleasant image to the human eye. However, this problem is ill-posed. An ill-posed problem is defined as one that violates one of the following properties.

1. A solution exists;
2. The solution is unique;
3. The solution changes smoothly with initial conditions.

As super-resolution is about synthesizing pixels and details that don't exist in the original images, we can assume that in general, the second property is violated. Thus the various solutions to the problem will have their own ways of "predicting" or "generating" the synthetic pixels. They can employ local generation, such as interpolation and image pyramids, or even deep learning, which can identify larger patterns and contexts from training examples to make better predictions.

Literature Review

To begin, we'll mention some simple classical methods used to solve this problem. These are methods introduced in our lectures in CSC420, and they helped us develop an intuition for the SR problem.

- Copying existing pixels.
- Image filters.
- Interpolation (linear interpolation in particular).
- Image Pyramids

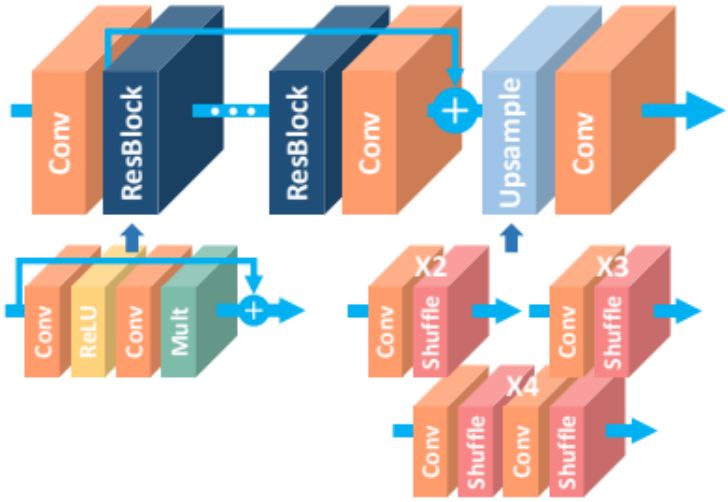

Figure 1: Enhanced Deep Super-Resolution Network (EDSRN). [6]

Of course, to produce an SR image, classical methods have access only to the pixels in the original image. This limitation could be overcome using previous "learned" context, as the last majority of images that we work with will have similar patterns to other images. As there is an extensive amount of work in this field, we have reviewed some of the deep learning algorithms proposed in the recent literature.

- Super-resolution through neighbor embedding, [1] a novel method for solving single-image super-resolution problems, inspired by locally linear embedding (LLE). This early solution leverages training examples to provide local geometry of the manifolds formed from patches taken from low and high resolution images. [1]
- A Fully Progressive Approach to Single-Image Super-Resolution. [2] The proposed method uses a progressive (from easy to hard) learning process aided by Generative Adversarial Networks (GANs). Interestingly, the authors reduce memory usage and training time while still increasing accuracy, with the use of an asymmetric architecture of Dense Compression Units (DCUs). [2]

- Deep Learning for Single Image Super-Resolution: A Brief Review. [3] This is a review paper of the existing solutions to SISR, and we can evaluate them in choosing algorithms and considering caveats. While we didn't quite use the implementations presented in this publication, we took as insight the challenges the authors faced: "the acceleration of deep models, the extensive comprehension of deep models and thecriteria for designing and evaluating the objective functions." [2]

- Neural-Enhance, [4] a deep learning implementation using extensive neural networks that "hallucinate" details of images. This relies on Lasagne, using Theano to support Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM). [5] Based on the author's descriptions, we hypothesize this solution might be rather sensitive to noise.

- Enhanced Deep Super-Resolution Network (EDSRN). [6] A residual network without unnecessary modules and progressively expanding model sizes in training. The authors propose a "new multi-scale deep super-resolution system (MDSR) and training method,which can reconstruct high-resolution images of differentupscaling factors in a single model." [6]

- Image Super-Resolution using Convolution Neural Networks and Autoencoders (CNNs+Autoencoders). [7]. This tutorial demonstrates how to combine CNNs and Autoencoders to generatively predict and enhance images. [7] This solution was appealing because it's quite explicit about its architecture and provides easy ways to tune the hyperparameters and adjust the structure.

Because deep learning is computationally expensive, we will test and evaluate the most promising methods that are feasible to implement. We will consider EDSRN, Neural-Enhance, and CNNs+Autoencoders to measure their results against classical methods.

Methodology, Experiments, Results

Figure 2: Preprocessed images (80×80).

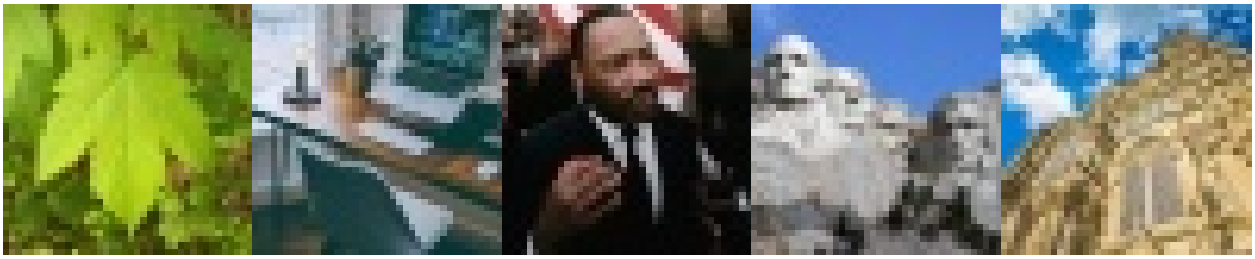

Figure 3: Reduced images (40×40).

For evaluation, we chose a small but varied test set of five images, each unique in same aspect. See the `small-test-set` folder in our repository for the originals.

- **Green Maple Leaf**: Small details of close-up object.
- **Home Office**: Many objects in one setting.
- **Martin Luther King Jr.**: A person's face and body language.
- **Mount Rushmore**: Details of far objects.
- **Salisbury Cathedral**: Upward angle at a façade.

In order to evaluate the methods on the same standards, we chose 80×80 as the base resolution, and 40×40 as the reduced resolution. As the original images had a variety of resolutions, we had to resize them down and "centre-crop" them (cropping the largest centre-square) using `preprocessing/preprocessed.py` [Figure 2]. Then, to produce the reduced resolution images, we used `cv2.INTER_AREA` [10] in `preprocessing/reduce.py` to produce our 40×40 images [Figure 3].

We will use the four methods to generate SR images from the reduced test set, that are 80×80. These dimensions are chosen because we will compare the resulting SR images to the original images to evaluate how well each method predicts/generates the missing pixels.

Figure 4: Super-resolution using linear interpolation.

Figure 5: Super-resolution using EDSRN.

Our linear interpolation method was implemented in `linear_inter.ipynb` and run on all the images to produce the SR images seen in [Figure 4]. In `linear_inter.ipynb`, we also employed EDSRN to produce images in [Figure 5].

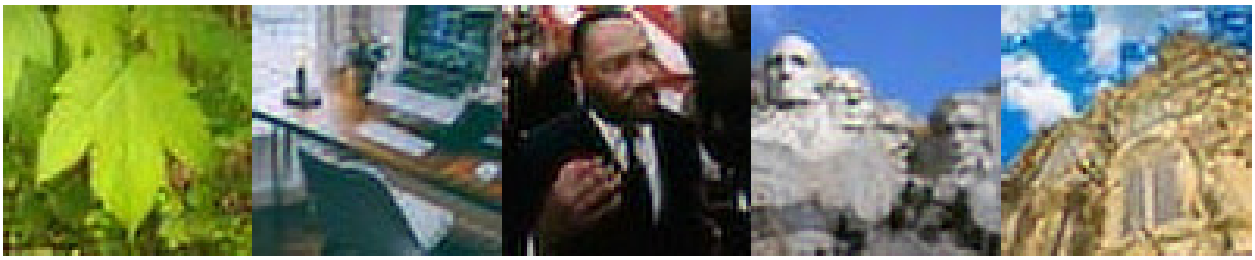

Figure 6: Super-resolution using Neural-Enhance.

Using the existing implementation of Neural Enhance [4], deployed to a dockerized container on Google Compute Platform (GCP) we generated the SR images in [Figure 6]. Neural Enhance provides pre-trained models of a purely CNN nature. Neural Enhance contains 16 Convolution layers and 4 max-pooling layers.

Figure 7: Super-resolution using CNNs and Autoencoders.

And finally, in using CNNs and Autoencoders, we followed a Towards Data Science tutorial [7] and trained the model on the "Labeled Faces in the Wild" dataset [8] [9] using a 5% partition and maximum of 100 epochs. For this method, we adjusted the hyper-parameters, and cleaned the validation set with minor data processing and adding our test images. Furthermore, we added a normalizing function `normalize()` as the produced images had scaling that slightly exceeded the upper bounds of pixel values. Using 50% pixelation (roughly equivalent to the factor of 2 in our downsampling preprocessing reduction), the cost function reached converged at the plateau in 51 epochs, when our final results were produced [Figure 7]. Looking at these images qualitatively, we have high hopes for this method.

Evaluation

In measuring the quality of the super-resolution (SR) outputs numerically, we used the following metrics, the latter two of which we devised to be creative.

1. Mean Squared-Error (MSE): Mean pixel-by-pixel squared differences between the original image and the SR output. Implemented in `mse.py`.

2. Mean Squared-Gradient-Errors (MSGE): Mean pixel-by-pixel squared gradient (computed using Sobel) differences between the original image and the SR output. *The values are expressed as pairwise gradient components* (G_x, G_y). Implemented in `MSGE.py`.

3. Corner Detection (Count): (SR Corner Count - Original Corner Count). This metric is a real number for each single method/image pair, and should be interpreted by *magnitude*. Implemented in `corner-detection.py`.

4. Corner Detection (Nearest Corner Distance): (D_{OtoSR}, D_{SRtoO}) where D_{OtoSR} is defined as the average distance between corners in the original images and their closest neighbour pixel in the SR image and D_{SRtoO} is defined as the average distance between corners in the SR images and their closest neighbour pixel in the original image. This is roughly equivalent to the effect of Corner Matching as we're finding the nearest corners of another image with the same perspective.

Results

Using the SR images generated from our methods [Figures 4, 5, 6, 7], we used our evaluation methods to produce the following results.

Green Maple Leaf Results				
	MSE	MSGE	Corner Count*	Nearest-Corner
Linear Interpolation	61.52	(274982, 222540)	−524	(1.375, 0.412)
EDSRN	55.04	(31710, 26342)	−216	(0.850, 0.570)
Neural-Enhance	187.30	(47674, 35089)	−308	(0.814, 0.412)
CNNs+Autoencoders	84.54	(31374, 20961)	−177	(0.796, 0.572)

Home Office Results				
	MSE	MSGE	Corner Count*	Nearest-Corner
Linear Interpolation	72.01	(330256, 419536)	87	(0.453, 0.614)
EDSRN	62.65	(39649, 51109)	−199	(0.500, 0.268)
Neural-Enhance	334.57	(74994, 74682)	−86	(0.709, 0.464)
CNNs+Autoencoders	62.43	(31148, 37969)	−2	(0.494, 0.594)

Martin Luther King Jr. Results				
	MSE	MSGE	Corner Count*	Nearest-Corner
Linear Interpolation	52.20	(574745, 433759)	7	(0.816, 0.700)
EDSRN	42.21	(53709, 36984)	59	(0.663, 0.713)
Neural-Enhance	259.24	(87961, 61993)	46	(0.604, 0.660)
CNNs+Autoencoders	43.17	(34841, 25462)	−27	(0.762, 0.656)

Mount Rushmore Results				
	MSE	MSGE	Corner Count*	Nearest-Corner
Linear Interpolation	59.88	(312489, 377194)	179	(0.674, 0.822)
EDSRN	53.06	(38247, 45897)	−201	(0.809, 0.540)
Neural-Enhance	197.42	(43156, 59944)	−30	(0.774, 0.653)
CNNs+Autoencoders	53.36	(30383, 38007)	−253	(0.878, 0.495)

Salisbury Cathedral Results				
	MSE	MSGE	Corner Count*	Nearest-Corner
Linear Interpolation	86.20	(438483, 437998)	−543	(0.617, 0.444)
EDSRN	79.80	(67422, 65934)	−276	(0.580, 0.490)
Neural-Enhance	414.76	(89386, 91589)	−293	(0.557, 0.462)
CNNs+Autoencoders	80.17	(52132, 55888)	−420	(0.590, 0.459)

** Note that Corner Counts are expressed as signed differences, and the quality of the SR solution via preservation of corners should be interpreted using the magnitude of the differences.*

Conclusion

After reviewing the results, we made the following observations.

- In Mean Squared-Error (MSE), Neural-Enhance was consistently the worst-performing method. This is likely due to the noise that Neural-Enhance generates everywhere on every image.
- With the exception of the Green Maple Leaf, CNNs and Autoencoders performed achieved the best MSE results or a very close second-best.
- Linear interpolation performed very poorly in the mean squared-gradient-errors. For some reason the errors are in six figures.
- Meanwhile, CNNs and Autoencoders performed well in this respect, with the lowest squared gradient (in both componenets) errors among all methods across all images.
- For the Green Maple Leaf and Salisbury Cathedral images, corner counts differed a lot for all methods, with three-figure corner count differences. Additionally, it would appear that every method lost hundreds of these perceived corners since their corner count differences are all negative for these two images.
- For the Martin Luther King Jr. image, corner counts from Corner Detection were the smallest in magnitude. This seems to agree with the visual inspection interpretation that the other images have many more corners and details, while this image of King had longer edges and more smooth color profiles.
- CNNs and Autoencoders had mixed results in corner detection. This is not what we expected, as we imagined that implementing deep learning to recognize natural and human-centred patterns would yield much better results than the classical methods we tested.
- Qualitatively, we comment on the quality of these SR images.
  - Linear interpolation produced images that look very similar to the reduced images, so this method is not very valuable.
  - OpenCV produced reasonably good images, but they're slightly blurry.
  - Neural-Enhanced produced images with considerable detail, but also with a lot of noise that make them look like they were repeatedly compressed and decompressed, or taken out of an old security feed.
  - CNNs and Autoencoders produced the best outputs for all five of our images.

We conclude that CNNs and Autoencoders are a great method for the super-resolution problem, likely attributing to its complexity and "information retention" by using autoencoders. While it performs just average in corner detection, it does well in MSE and MSGE, and produces images that are much more pleasing to the human eye.

Authors' Contributions

If you're unable to access our repository from Quercus, please see https://github.com/rusbridger/CSC420-Project.

Michal Fishkin

- Obtaining/uploading/sharing the Big Bang Theory dataset.
- Cleaned and labeled the dataset.
- Helped with the pre-processing procedure.
- Qualitative interpretation of results.
- Problem outline and visualizations.

Siddhant Jain

- Implementing linear interpolation and EDSRN SR methods.
  - `linear_inter.ipynb`
- Testing and adjusting hyperparameters for Neural-Enhance in Docker.
- MSE evaluation script.
  - `mse.py`

Robert Wu

- Preprocessing/filtering the Big Bang Theory face dataset and small test dataset (processed/reduced).
  - `preprocess.py` and `reduce.py`
  - https://drive.google.com/drive/folders/18EiU2Hwzs8mBduMjlha1m9O75ItpFmv5
- Testing and adjusting hyperparameters for CNNs+Autoencoders.
- Corner Detection and MSGE evaluation scripts and results formatting.
  - `corner-detection.py` and `gradients.py`

References

[1] Hong Chang, Dit-Yan Yeung, Yimin Xiong, *Super-resolution through neighbor embedding*, Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.; 1 (2004). Available at https://ieeexplore.ieee.org/abstract/document/1315043.

[2] Wang, Yifan and Perazzi, Federico and McWilliams, Brian and Sorkine-Hornung, Alexander and Sorkine-Hornung, Olga and Schroers, Christopher, *A Fully Progressive Approach to Single-Image Super-Resolution*, 2018. Available at https://igl.ethz.ch/projects/prosr/prosr-cvprw-2018-wang-et-al.pdf.

[3] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, Qingmin Liao, *Deep Learning for Single Image Super-Resolution: A Brief Review*, IEEE Transactions on Multimedia (TMM); 3 (2019). Available at https://arxiv.org/pdf/1808.03344v3.pdf.

[4] Alex J. Champandard, *Neural Enhance*, 2016. Available at https://github.com/alexjc/neural-enhance.

[5] Sander Dieleman and Jan Schlüter and Colin Raffel and Eben Olson and Søren Kaae Sønderby and Daniel Nouri et al, *Lasagne: First release.*, 2015. Available at http://dx.doi.org/10.5281/zenodo.27878

[6] Bee Lim and Sanghyun Son and Heewon Kim and Seungjun Nah and Kyoung Mu Lee, *Enhanced Deep Residual Networks for Single Image Super-Resolution*, CVPR 2017, Available at https://arxiv.org/abs/1707.02921.

[7] Harshil Patel, *Image Super-Resolution using Convolution Neural Networks and Auto-encoders*, Towards Data Science (May 16, 2020). Available at

- https://towardsdatascience.com/image-super-resolution-using-convolution-neural-networks-and-auto-encoders-28c9eceadf90
- https://github.com/harshilpatel99/image_superResolution

[8] Gary B. Huang and Manu Ramesh and Tamara Berg and Erik Learned-Miller, *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*, University of Massachusetts, Amherst (Oct, 2007). Available at http://vis-www.cs.umass.edu/lfw/

[9] Gary B. Huang Erik Learned-Miller, *Labeled Faces in the Wild: Updates and New Reporting Procedures*, University of Massachusetts, Amherst (May, 2014). Available at http://vis-www.cs.umass.edu/lfw/

[10] Wenru Dong, *What is OpenCV's INTER_AREA Actually Doing?*, Medium (Jun 24, 2018). Available at https://medium.com/@wenrudong/what-is-opencvs-inter-area-actually-doing-282a626a09b3