Regenerating Compressed JPEG Images

Project Report

Zafir Khalid
ID: 40152164
zafirmk0@gmail.com

December 2022

Gina Cody School of Engineering and Computer Science
Concordia University
Montreal, Canada

Professor - Dorra Rihai

COMP 432 - Machine Learning

Abstract

JPEG image compression is a form of lossy compression where data is lost. As such, these files are generally small in size and low quality (Figure 2). The main objective of this machine learning project is to restore the information that was lost due to this lossy compression. In other words, given a compressed JPEG image - the goal of this project is to regenerate a higher quality image (Figure 3).

Using a kaggle dataset that provides 855 high quality colored , an image compression pipeline created using OpenCV and a convolutional neural network created using Tensorflow - a successful model was implemented with an average SSIM score of 94% and an average training accuracy of 80%.

Figure 1: Original Img

Figure 2: Compressed Img

Figure 3: Regenerated Img

Keywords— Convolutional Neural Network, Images, Super resolution

1 Overview

At a high level, the layout of the project is done in different modules resembling a pipeline infrastructure. This is to ensure the smoothness of data flow and readability of code. Shown in Figure 4 below, the original (high quality) image is passed through an image compression module - through which a compressed, low quality image is retrieved. The low quality image maintains the same size as the input image. Note for this project all images (low quality and high quality) are of the shape (256 x 256 x 3). There are 855 high quality images and their corresponding 855 low quality images.

Figure 4: Processing Pipeline

The *DataLoader* class is responsible for the image compression and image normalization. Image normalization is a process where in pixel values are scaled to be in range $[0, 1]$ - this is to ensure ease of training and numerical stability. Since the activation functions used in CNNs (such as the sigmoid or the ReLU function) are typically defined on the range 0 to 1. Using inputs that are already scaled to this range can help ensure that the activations fall within the desired range and can help the network converge faster.

Once the entire set of low resolution images has been generated and saved - a train/test split is applied to the entire dataset before it is sent on its way to a convolutional neural network for training and testing. For this particular problem the train/test split applied was 90% for training and 10% testing (with a portion reserved as the validation set). A 90-10 split was chosen because of the number of images available. Even with 10% there are still a sufficient amount of images to test on. Below is the structure of the training, testing and validation set.

```
Shape of training images:  (769, 256, 256, 3)
Shape of test images:  (78, 256, 256, 3)
Shape of validation images:  (8, 256, 256, 3)
```

2 Model Architecture

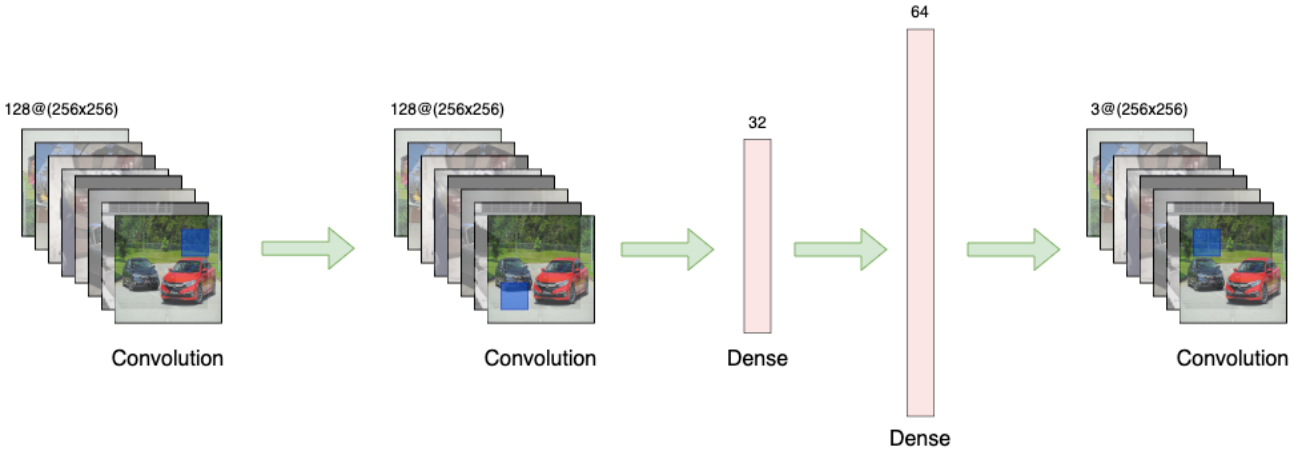

Figure 5: CNN Architecture

The convolutional neural network (CNN) architecture used is a fairly standard architecture for image super resolution tasks. The use of convolutional layers allows the network to learn local patterns in the input image, which is important for preserving the high-frequency details that are lost during downscaling. The ReLU (rectified linear unit) activation function applied along with each convolutional layer. This is commonly used in CNNs as it allows the model to learn non-linear relationships between the input and output. For the first two convolutional layers a 128 by 128 kernel is used along with padding set to same - so as to ensure that the image does not reduce in size whilst passing through the layers. The use of dense layers, also known as fully-connected layers, allows the network to learn global patterns in the input image. The ReLU activation function is again used to introduce non-linearity to the model.

Overall, this architecture is well-suited for image super resolution tasks as it allows the model to learn both local and global patterns in the input image and make use of non-linear relationships between the input and output. The use of ReLU activation functions also helps to improve the model's performance by introducing non-linearity to the model and allowing the model to learn more complex patterns in the data. A final softmax layer is added after the last dense layer. The softmax function has the property of "squashing" the outputs of the neurons to be between 0 and 1, which can help to constrain the output of the network and improve its stability and generalization performance. The table below shows the number of parameters used across the architecture.

```
    Layer (type)                 Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0
 conv2d (Conv2D)             (None, 256, 256, 128)     3584
 conv2d_1 (Conv2D)           (None, 256, 256, 128)     147584
 dense (Dense)               (None, 256, 256, 32)      4128
 dense_1 (Dense)             (None, 256, 256, 64)      2112
 conv2d_2 (Conv2D)           (None, 256, 256, 3)       15555
=================================================================
Total params: 172,963
Trainable params: 172,963
```

3 Model Evaluation

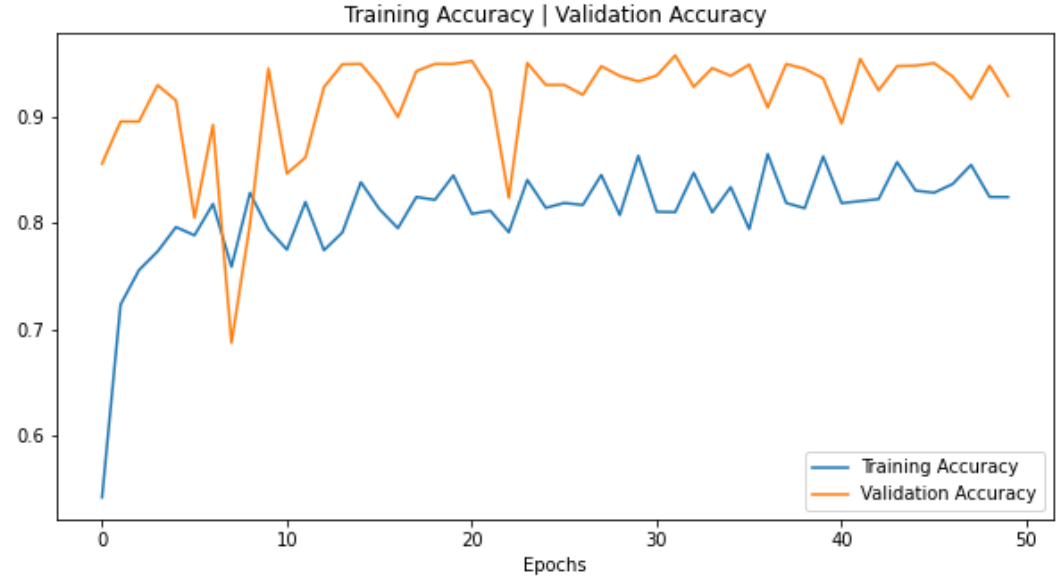

Figure 6: Training & Validation Accuracy

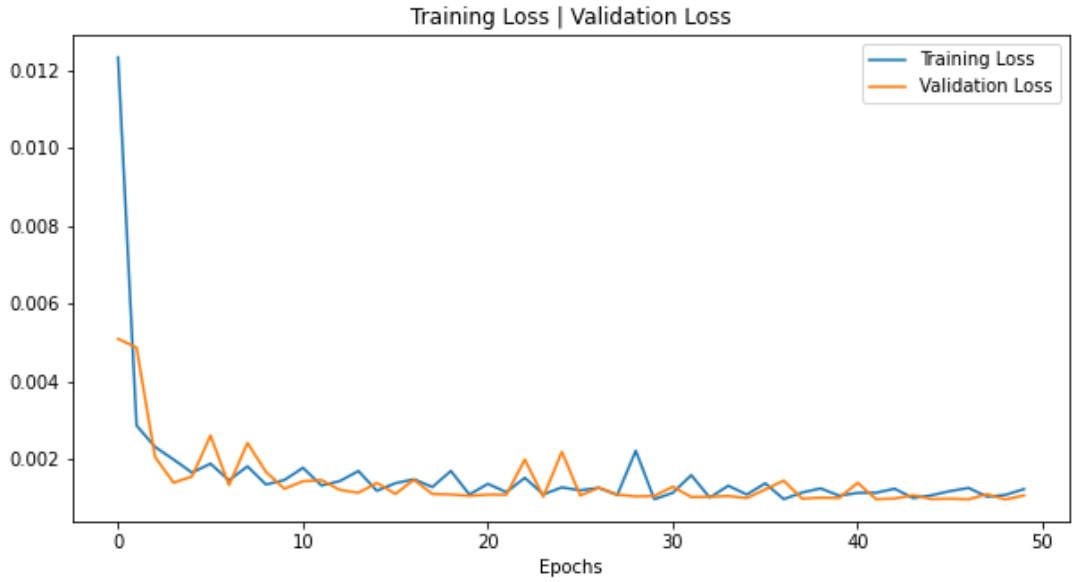

Figure 7: Training & Validation Loss

The training and validation accuracies shown for the model show a steady increase as the number of epochs continues towards 50. Training accuracy reaches nearly 84%, where as validation accuracy peaks at roughly 95%. The accuracy graphs are also consistent with the corresponding loss graphs - as the accuracy increases the loss values drop down. Both training loss and validation loss go down to roughly 0.001.

3.1 Structural Similarity Index (SSIM)

The structural similarity index (SSIM) is a measure of the similarity between two images. It is a widely used index in image processing and computer vision that aims to quantify the perceived change in quality between two images. SSIM is based on the idea that the human visual system is more sensitive to some types of changes in images than others. It takes into account the luminance, contrast, and structure of the images and computes a score. SSIM is often used to compare the quality of image processing algorithms and to evaluate the performance of image compression techniques and super resolution techniques.

Figure 8: Training SSIM Score

Figure 9: Testing SSIM Score

SSIM is a better metric to evaluate the training process as it accurately compares two images. The training SSIM and validation SSIM scores reach nearly 95%. Compared to the accuracy scores these are far greater and a much better estimator for the performance of the model training. Moreover, the graph (which samples the first 20 images of the testing set) shows that the similarity between the high quality image and predicted images is greater than the similarity between the low quality images and high quality images. In other words, the model is able to successfully generate images that are of better quality than the low

resolution compressed images.

Note: Only the first 20 images of the test set are shown in the graph so that the graph is not too crowded.

4 Model Results

Below are some of the results obtained from the super resolution CNN model.

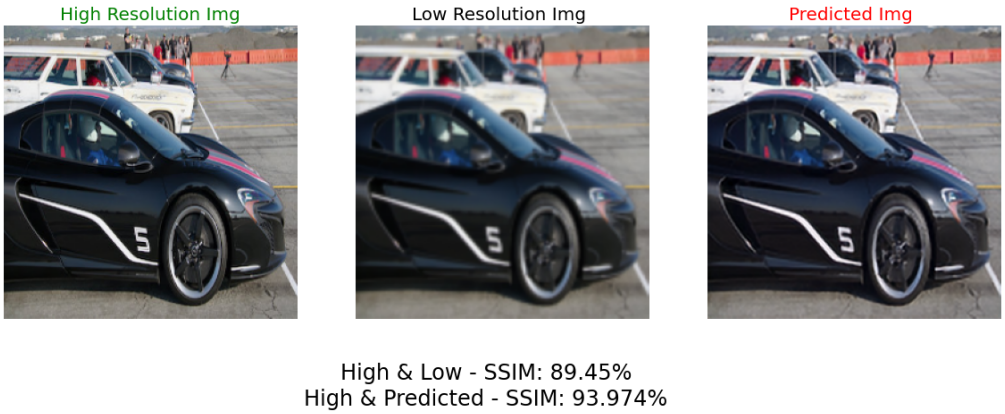

Figure 10: Example image of a subject

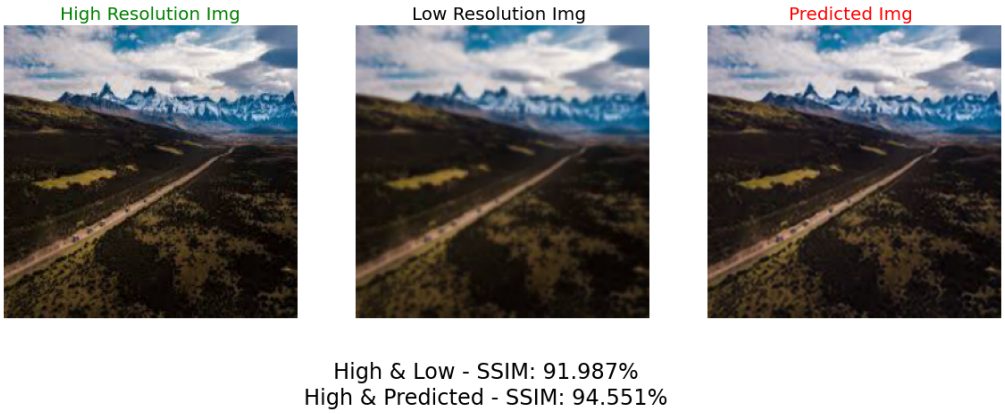

Figure 11: Example image of a landscape

Figure 12: Example image of a face

Figure 13: Example image of a person

4.1 Reproducibility of Results

In order to make predictions on images, use the *Prediction* class and in particular the *predictIndividual* method to make a single prediction or the *predictTestSet* method to predict on the entire test set.