# Fixing Blocking Artifacts from JPEG Images

By Oren Kohavi, Michael Li, and Jesse Gallant

*"Those are some crisp and compressed images!"*

## INTRODUCTION

Image compression is necessary for handling high-resolution images in the modern world, but when images are restricted to low-bandwidth or low-storage applications, high levels of compression must be used, leading to loss of information (lossy compression), but also visible loss in quality.

In the case of JPEG Images, the most popular form of lossy compression, images with high compression tend to appear blocky, and smooth curves will often become jagged. This is because JPEG compresses each 8x8-pixel region of an image separately, and borders between tiles of an image often appear jarring and unnatural.

## DATA

We are using the Kodak Lossless Image dataset linked in the project handout. There are 24 images, each either 768x512 or 512x768. Each image contains 3 channels, representing the RGB value of corresponding pixels. With our preprocessing methodology (detailed below), this should result in roughly 150k input/label pairs. (512(height) * 768(width) * 24(images in dataset) / 64(pixels per block) = 147,456)

## METHOD

The architecture of our model is detailed in the paper: It consists of a series of convolution layers and residual blocks (each residual block consisting primarily of batch-normalization, with some 1x1 convolution mixed in) The model will be trained using inputs of size 24x24x3 and outputs of size 8x8x3, which will be generated by a preprocessing script by splitting up larger images. The large input images to this preprocessing script will be lossless images from our dataset.

"Similar to JPEG, we partition an image into 8×8 blocks and process each block separately. We use a convolutional neural network that inputs a block together with its adjacent blocks (a 24 × 24 image), and outputs the processed block in the center." - BlockCNN

Anticipated challenges include the large number of layers, which might make training slow, and the complex structure of the residual block, which will be non-trivial to implement in Tensorflow.

## RELATED WORKS

We drew inspiration from "BlockCNN: A Deep Network for Artifact Removal and Image Compression" where they demonstrated a method for artifact removal and image compression to minimize image size while retaining as much image information as possible.

## RESULTS

Success for this project was defined as producing an output image that has higher similarity to the original image than the compressed JPEG image does. We achieved this goal, with our deblocked images looking much better than the compressed version.

On the right, we have a segment of one of the Kodak images depicted. The original image is displayed on top. Below are the two images. One of them is compressed (on the right). In this image we an see blocking artifacts that make the image look pixelated low quality.

**ORIGINAL**

The left image is our networks output image. Although the de-blocked and the compressed image are the same size, the de-blocked image is much less pixelated. The de-blocked image is visually much smoother than the compressed image. The biggest improvement is that the networks output image has much fewer blocking artifacts which means our network was a success!

**DE-BLOCKED**

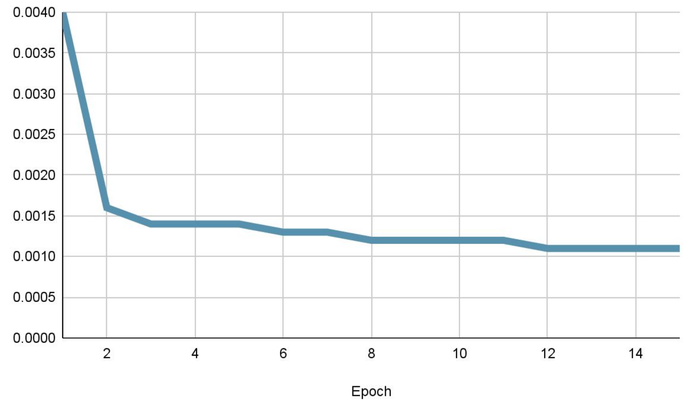
**COMPRESSED**

## DISCUSSION

As seen in the results of our implementation of BlockCNN, our network was very successful at de-blocking compressed images. Although successful, it is important to note the drawbacks to our implementation. The biggest drawback to our network is the training data. The Kodak image set is mostly of outdoor images whose subject is animals or everyday objects. If our network were given an image of something unlike any of our training data, it might have a hard time successfully de-blocking it.

The other biggest downside to our network is its complexity. The network takes a long time to train because it has many convolve layers. After training, it is possible that the network might be inefficient on a phone.

**Loss per Epoch:**




Training Loss: 0.00138