Removing JPEG Blocking Artifacts Using Machine Learning

Jonathan Quijas **Olac Fuentes**
Department of Computer Science
University of Texas at El Paso

***Abstract*—JPEG is a commonly used image compression method. While it normally yields very good compression ratios, it also introduces blocking artifacts and quantization noise. In this paper, we present a method to remove noise and blocking effects from JPEG-compressed images. We use machine learning techniques to predict DCT coefficients and pixel values in a compressed image. Results show a decrease in mean square error between our predicted images and the original uncompressed images when compared to the compressed images, as well as a clear reduction of blocking artifacts.**

***Index Terms*—Image compression, JPEG, feed-forward neural networks, artifact removal**

I. Introduction

The most common format for digital image compression is JPEG. It compresses 8 by 8 pixel blocks using the discrete cosine transform (DCT) by quantizing pixel information to reduce the size of files. When compressing an image using JPEG, the image is transformed from the RGB color space into the Y'CbCr color space, which separates the luminance and color components into different channels. Because the human visual system has poorer frequency response to color than to luminance, color components (CbCr) are subsampled, greatly reducing an image's file size with minimal perceivable effect [1]. Luminance components are then reduced by quantization in the DCT domain, discarding the luminance (Y') channel's low-frequency components and preserving an image's overall structure [2]. Since the DCT is local to each 8 by 8 block, blocking artifacts are created when an image is heavily compressed.

Wavelet-based approaches to compression have been proposed [3], using extracted edge information and exploiting correlations among wavelet coefficients to detect and smooth blocks in uniform background regions [4]. Other compression techniques consider complex models with high computational cost, and some require user input parameters. When post-processing is finished, some detail is commonly lost and the resulting image is usually blurred.

Our goal is to preserve as much detail as possible while removing the block-like discontinuities in the compressed image. We approached this problem with a machine learning solution, building a set of predictors that would improve our compressed, blocky images. We trained a set of feed-forward neural networks to predict the values of the discarded DCT coefficients for each 8 by 8 block, as well as approximate the pixel intensities for all pixels at the block boundaries. Once learning is finished, the resulting weights can be stored to easily process any number of new images.

(a) Original (b) Compressed (Q = 25)

Fig. 1: Effects of heavy compression

II. Effects of JPEG Compression and Our Goals

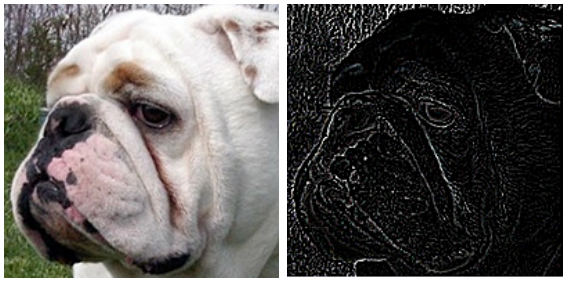
JPEG is a lossy compression format. When an 8 by 8 image block is transformed into the DCT domain, the resulting coefficients are quantized, or integer divided, by their corresponding JPEG-standard quantization matrix coefficients. This can be denoted as

$$B_{i,j} = \text{round}\left(\frac{D_{i,j}}{Q_{i,j}}\right) \text{ for } i = 0,\ldots,7; j = 0,\ldots,7 \quad (1)$$

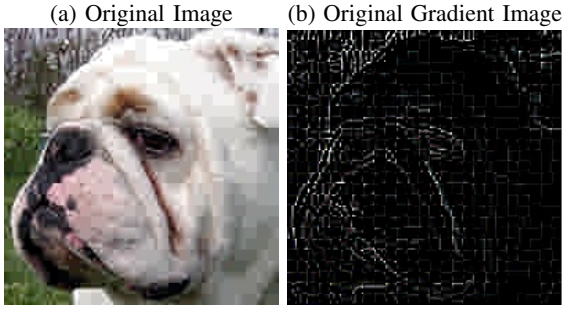
where B is the quantized DCT coefficient matrix, Q is the JPEG quatization matrix, and D is the original DCT coefficient matrix. The image block is then transformed back from the DCT domain to the spatial domain. This quantization process creates an information loss which cannot be regained, but only approximated. The reason there are blocking artifacts created is that the Discrete Cosine Transform is performed on independent, non-overlapping blocks of 8 by 8 pixels. When heavy compression is done, more information is lost, and pixel gradients become larger at these block boundaries.

We are interested in improving the quality of compressed images by reducing noise and block discontinuities found on 8 by 8 image blocks, while preserving high-frequency details, i.e. edges and textures. We are also interested in analysing and improving the performance of vision algorithms which may fail on heavily compressed images.

Neural networks are a well-known machine learning technique. Their powerful learning capabilities allow them to learn different types of non-linear functions. Once trained, the

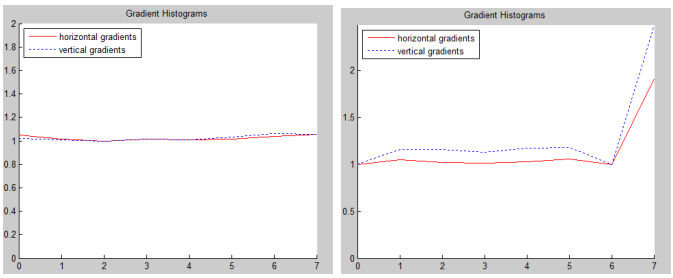


(a) Original Image (b) Original Gradient Image



(c) Compressed Image (d) New Gradient Image

Fig. 2: The effects of heavy compression are shown. After heavy compression, ringing and blocking artifacts are introduced into the compressed image. Although detail is lost, the overall image structure can still be seen.



(a) Original (b) Compressed (Q = 25)

Fig. 3: An image's normalized gradient histograms (original and compressed), each with eight bins representing the pixels corresponding index in an 8 by 8 pixel group. The gradients are the largest at the block boundaries.

weights learned can be stored, and prediction on new data instances is fast. Because of these qualities, they naturally became a candidate solution to this problem.

III. Enhancing the Image

When an image is heavily compressed, several of its high-frequency components are discarded, resulting in loss of detail and texture flattening. We want an image processing framework that allows for the file size reduction benefits of JPEG while ridding the image of blocking artifacts and noise. As mentioned before, many image deblocking algorithms used today are computationally expensive, and can increase the compressed image's mean square error (MSE). This translates to a smaller peak signal-to-noise ratio (PSNR), deviating from what should be a signal and into distorting noise. This led our research to take a machine learning approach to develop an efficient and effective method to rid the image of noise and recover true signal values.

In machine learning, training a predictor usually takes a larger amount of the time compared to making a prediction. This is especially true for complex algorithms such as neural networks. Once learning is finished, however, the learned parameters can be stored and reused to predict values on any new data instance efficiently. Prediction using a neural network is fast, being just a non-linear combination of the instance's attributes and the learned weights.

A. Learning DCT Coefficients

A big part of information loss in JPEG compression happens when the DCT coefficients are quantized. We want to recover such lost information by approximating the original non-quantized DCT values corresponding to the image prior to compression. We approached this by using a trained network's prediction as the true DCT value for its corresponding DCT coefficient. A DCT-predicting neural network receives as input the quantized DCT coefficients in an 8 by 8 pixel block and predicts a DCT coefficient corresponding to its index position in the pixel block. After all coefficients are predicted, we perform an inverse Discrete Cosine Transform to bring back the image to the space domain. We trained 64 neural networks to predict the values of an image block's 64 DCT coefficients, training one network for each coefficient.

B. Learning Pixel Values

Blocking artifacts are noticeable because the change of pixel intensities are larger at the boundaries between two adjacent 8 by 8 pixel blocks. By reducing these intensity differences, blocking artifacts become less evident. Because of this, we developed the idea of approximating pixel values located at these border positions. We trained a set of neural networks to predict the luminance intensity values at image block boundary indexes (mod 8 == 0, mod 8 == 1), using the pixels' neighborhoods as instance attributes. We experimented with square neighborhoods of various sizes (k = 3, 5, and 7). Our tests gave the most favorable results when we trained using a pixel's 49-cell neighborhood (k = 7). We considered vertical and horizontal data separately. A pixel-predicting neural network takes as input a pixel neighborhood and predicts a target pixel's value. After prediction, all pixels at pixel block boundaries are replaced with the predicted values. This process yields an evident reduction of blocking artifacts.

C. Smoothing

When an image is compressed using JPEG, a certain degree of the image's high-frequency components are removed, but some are also introduced in the form of blocking artifacts and aliasing, i.e. ringing artifacts around areas of interest. An image can be smoothed with a Gaussian filter

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

to mitigate undesired high-frequency noise.

We convolved a Gaussian kernel with the compressed images to remove some of the noise and artifacts introduced by compression. Although we measured a decrease in error after the convolution, this did not yield a significant increase in image quality, i.e. PSNR. However, when smoothing was combined with neural network prediction, it boosted the overall performance of our method. Our tests indicate that a Gaussian kernel of size 3 by 3 pixels with standard deviation of .4 yields the best results in terms of error decrease.

IV. Processing Methods

We begin the construction of our framework by collecting a set of *n* images compressed with the same quality parameter *q*. We then proceed to smooth our entire image set using a Gaussian filter with the parameter values mentioned above. Thus, we describe our new set of smoothed images as

$$S(k) = I(k) \circledast G \text{ for } k = 1, 2, \ldots, n, \tag{3}$$

where *S* is the new set of smoothed images, *I* is the original set of compressed images, *G* is our Gaussian filter, and $\circledast$ is the convolution operator.

We then train a set of feedforward neural networks to predict DCT coefficient values and another set to predict pixel intensity values at block boundaries.

A. DCT Nets

- For each smoothed image s_k in our image set S,
  - For each non-overlapping 8 by 8 image pixel block,
    - Transform pixel block into DCT domain, extract this matrix as a vector and add it to our training set
- For each DCT coefficient *D(i,j)*, *i*=1,...,8; *j*=1,...,8
  - Train a feedforward neural network to predict the original DCT coefficient prior to compression
- After training, store learned weights

B. Pixel Nets

The following describes the process to build the training sets for the pixel predicting nets. We build a total of four training sets: A set for each direction (vertical or horizontal) at each target location (i mod 8 == 0, i mod 8 == 1, j mod 8 == 0 and j mod 8 == 1)

- For each smoothed image s_k in our image set S,
  - For each direction (vertical and horizontal)
    - For each target pixel (vertical:i mod 8 == 0 and i mod 8 == 1, horizontal:j mod 8 == 0 and j mod 8 == 1)
      - Add the pixel's neighborhood vector to the corresponding training set
- Train a feedforward neural network to predict the original pixel values prior to compression
- After training, store learned weights

C. Processing

- Predict image's block border pixel values; we call these *PIXPRED*
- Predict image's DCT coefficient values and transform back to pixel space; we call these *DCTPRED*
- Combine both groups of predicted pixel values, replacing the inside of the compressed pixels blocks with *DCTPRED* and the outermost pixels with *PIXPRED*

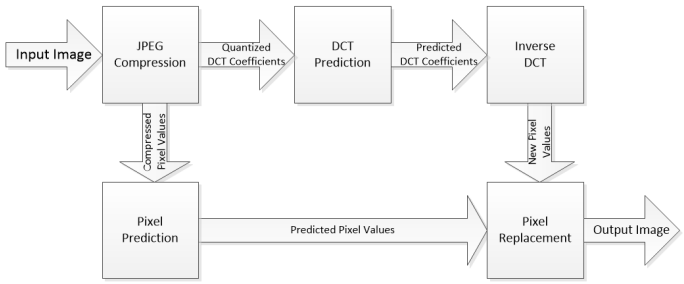


Fig. 4: Our method's pipeline.

V. Results

We tested our method with a total of 5 images. We compare our results to other deblocking methods, namely reapplication of JPEG [5] and UnBlock [6]. The following list defines the methods listed on the table:

- Gaussian: After smoothing the image with a Gaussian filter
- Re-JPEG: After reapplying JPEG compression
- UnBlock: After applying UnBlock algorithm
- DCT Prediction: Prediction of DCT coefficients
- Pixel Prediction: Prediction of border pixels
- Combined Predictions: Combination of predicted border pixels and pixels after DCT prediction
- Gaussian Combined Predictions: Combination of predicted border pixels and pixels after DCT prediction based off Gaussian-filtered image training

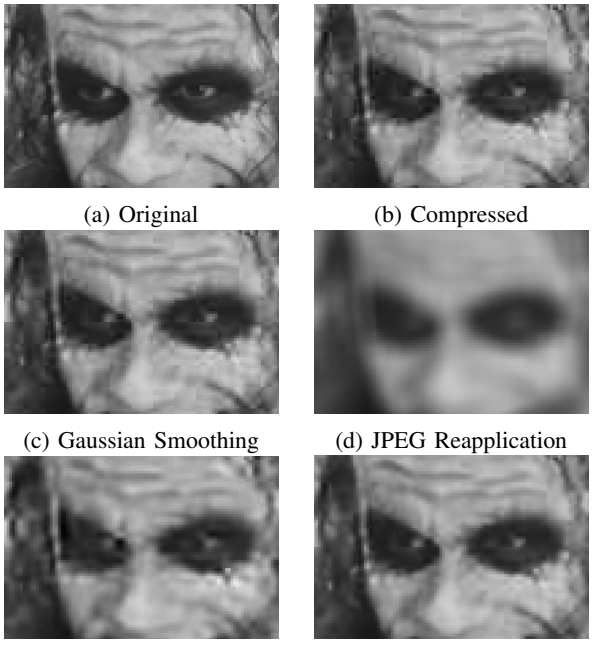
The following results are the average improvement percentage in mean square error (MSE) and peak signal-to-noise ratio (PSNR):

- The average MSE introduced by JPEG compression was 80.301
- The average PSNR was 29.265

Method	MSE %	PSNR %
Gaussian	4.950%	0.757%
Re-JPEG	-372.291%	-23.007%
UnBlock	-227.864%	-17.615%
DCT Prediction	6.879%	1.059%
Pixel Prediction	8.784%	1.365%
Combined Prediction	10.816%	1.700%
Gaussian Combined Prediction	11.143%	1.755%

Fig. 5: Reduction in mean-squared-error and peak-signal-to-noise-ratio for each of the algorithms tested.

Figure 5 shows the results of applying the algorithms described above to our test images. It can be seen that our method reduces noise while preserving details and avoiding deviation from the original pixel values.

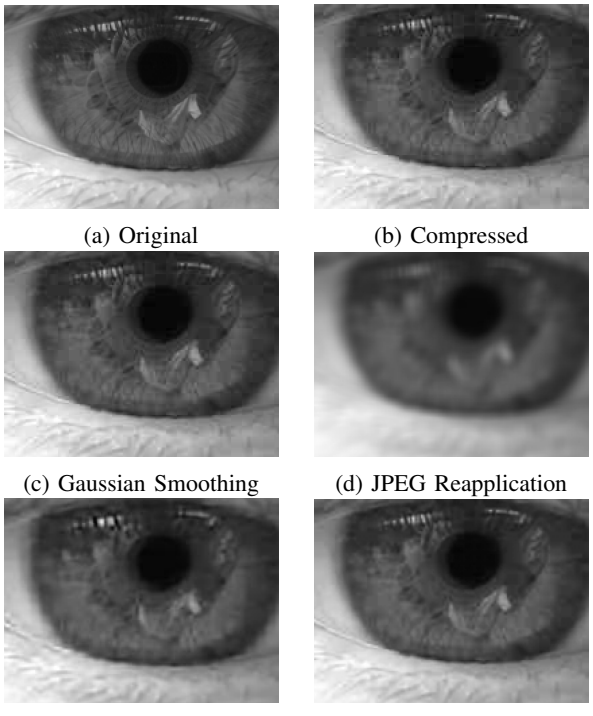


(a) Original (b) Compressed

(c) Gaussian Smoothing (d) JPEG Reapplication

(e) UnBlock (f) Our Method

Fig. 6: Face close-up

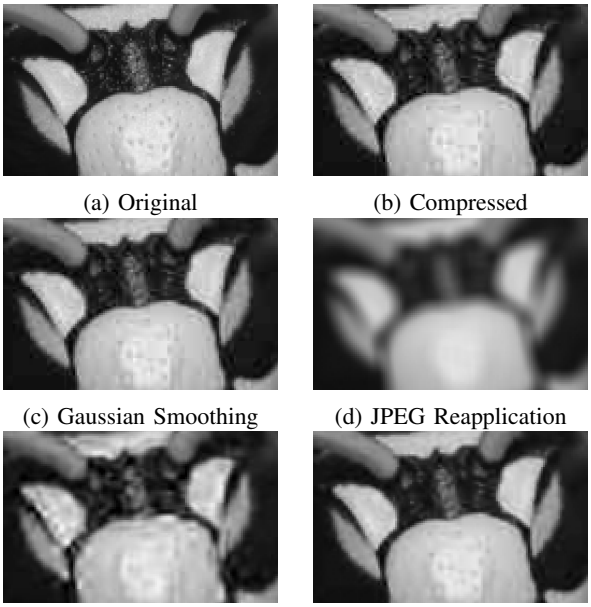


(a) Original (b) Compressed

(c) Gaussian Smoothing (d) JPEG Reapplication

(e) UnBlock (f) Our Method

Fig. 7: Eye close-up



(a) Original (b) Compressed

(c) Gaussian Smoothing (d) JPEG Reapplication

(e) UnBlock (f) Our Method

Fig. 8: A wasp's head

VI. Conclusions

We presented a machine learning method that allows to process images affected with blocking artifacts and noise and improve image quality. Our method yields a reduction in MSE and increase in PSNR, as well as an evident reduction of blocking artifacts. Processing is fast, and no user-input is necessary.

References

[1] C. J. van den Branden Lambrecht, Ed., *Vision Models and Applications to Image and Video Processing*. Springer, 2001, ch. 10, pp. 202, 208–209.

[2] G. K. Wallace, "The jpeg still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, Apr. 1991.

[3] Y. Yang and N. P. Galatsanos, "Removal of compression artifacts using projections onto convex sets and line process modeling," *IEEE Transactions on Image Processing*, vol. 6, pp. 1345–1357, 1997.

[4] Z. Xiong, M. T. Orchard, and Y.-Q. Zhang, "A deblocking algorithm for jpeg compressed images using overcomplete wavelet representations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 2, pp. 433–437, 1997.

[5] A. Nosratinia, "Enhancement of jpeg-compressed images by re-application of jpeg," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 27, no. 1-2, pp. 69–79, 2001.

[6] J. P. Costella, "The unblock algorithm," 2006.