MSc Thesis
Master of Science in Computing Science

Supervised by:
Prof. dr. Elena Marchiori

Radboud Universiteit

JPEG Compression Artifact Removal with Convolutional Neural Networks

Guido C.A. Zuidhof (s4160703)

Nijmegen, March 2017

Summary

This thesis describes a method for automatic reduction of compression artifacts introduced by JPEG image compression. The aim is to best reconstruct images of any dimension to the state prior to compression.

0.0 (1) Provide an actual summary

Contents

Todo list

CHAPTER 1 Background

1.1 JPEG compression

JPEG is a lossy compression method for digital images. It was named after the *Joint Photographics Experts Group* which authored the standard. This method aims to make the image more compressible by throwing away information which leads to a small difference in perceived quality. In particular it removes high frequency changes in intensity, and often also hue information but not brightness.

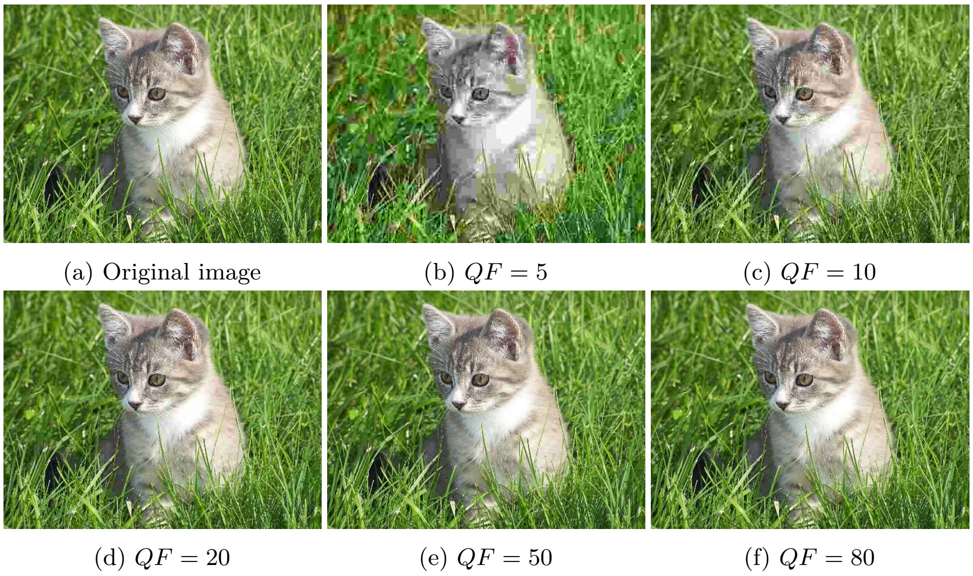
(a) Original image (b) $QF = 5$ (c) $QF = 10$
(d) $QF = 20$ (e) $QF = 50$ (f) $QF = 80$

Figure 1.1: Example image after JPEG compression with varying quality factors. Blocking artifacts and hue information loss are especially visible with low quality factor settings.

JPEG compression has one main parameter, the *quality factor (QF)*, which is a value between 0 and 100. A lower quality factor means that more high frequency information is thrown away, which allows for a greater compression ratio, but at the cost of quality loss. In figure 1.2 an image is shown at various quality factor settings.

1.1.1 JPEG compression overview

Different encoding methods exist within the JPEG standard, but the most common is JFIF encoding which consists roughly of the following steps:

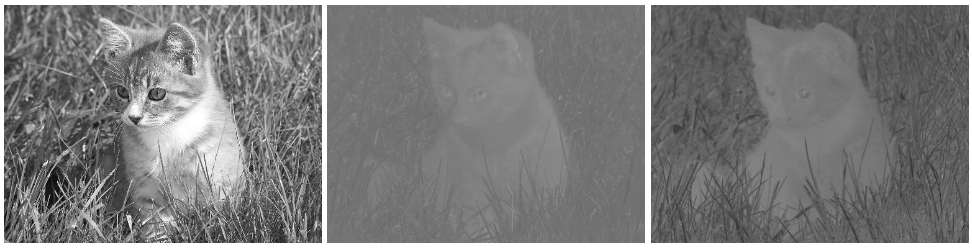
(a) Luma (Y) (b) Blue difference (C_b) (c) Red difference (C_r)

Figure 1.2: Color components of our example image in $Y'C_BC_R$ color space.

Transformation to $Y'C_BC_R$ color space The image is converted from RGB color space to the $Y'C_BC_R$ color space. The first component, Y, indicates the brightness. The other two components C_B and C_R represent the *chrominance value* (color). The reason this color space is useful is because the human perceptual system is especially sensitive to changes in brightness, but less so to small differences in color. In later steps this allows for targeted downsampling. Also, there are miscellaneous other small advantages to compressing in this data format.

Downsampling the C_B and C_R components In this step the spatial resolution of the chroma components is reduced by downsampling. This step may be skipped for high quality factor settings.

Splitting into 8×8 blocks Each image channel is split into blocks of 8×8 pixels, starting from the top left. Depending on the chrome subsampling applied in the previous step, the *minimum coded unit (MCU)* block size may be larger than 8×8; most commonly being 16×16 due to a reduction by a factor 2 of the chroma component in both directions. Prior work in JPEG artifact reduction often does not consider the larger blocking effect introduced in the colors of an image, and instead only evaluate on grayscale images.

If necessary, the bottom and right of the images are padded so that the image size is a multiple of the MCU size in both dimensions. Multiple different padding strategies exist in order to prevent artifacts introduced by this padding.

Discrete cosine transform Every 8×8 block is converted to a representation in the frequency domain using a normalized, two-dimensional type-II *discrete cosine transform*(DCT)[9]. This is a method for generating the cosine transform without using the fourier transform. To do this, the image is first zero-centered by subtracting 128 from every pixel (for an 8-bit image). We define a two-dimensional matrix C as:

$$C(k,n) = \frac{1}{\sqrt{N}} \; for \; k,n = 0$$

$$C(k,n) = \sqrt{\frac{2}{N}} \; cos(\pi(2k+1)\frac{n}{2N}) \; otherwise$$

For image U the DCT coefficients matrix G is then given by

$$G = CUC^T.$$

This leads to a matrix of coefficients with high values generally situated in the top left corner (which correspond to the DC coefficient and low frequency AC coefficients). See figure 1.3.

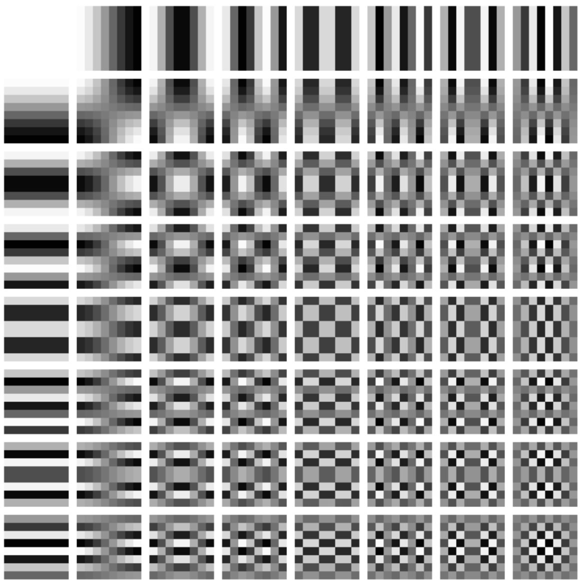

Figure 1.3: *DCT patterns:* the values in the DCT coefficient matrix are weights for each of these patterns (known as *basis functions*) which together construct the original 8×8 image

Quantization Every component in the frequency domain is now divided by a specific value. These values are given by the quantization matrix (which is different for different quality settings of the JPEG compression).

With Q the quantization matrix, the quantized coefficient matrix K is given by:

$$K(k,n) = round(\frac{G(k,n)}{Q(k,n)}) \; for \; k = 0,1,2,...,7; n = 0,1,2,...,7$$

This rounding to the nearest integer is the only lossy part in the encoding, aside from possible downsampling of the C_B and C_R components.

Entropy encoding The values in this matrix are now losslessly compressed. The components are arranged in a “zigzag” order starting in the top left corner, which is favorable for run length encoding due to the zero values at the end.

Decoding Decoding the image simply involves performing the opposite of all previous steps in reverse order. Of course, due to the rounding operation in the quantization step and the possible downsampling of chroma components the decoded image is likely different from the original. This difference may be visible to the human psychoperceptual system in the form of the artifacts described next.

1.2 Artifacts

Due to loss of information, visible artifacts are introduced to the image. Four main types of artifacts are generally observed under JPEG compression of images with a low quality setting.

Ringing artifacts JPEG operates in the frequency domain, representing the image as a sum of oscillating waves. This is not suited for sharp edges, especially with low quality settings, these high frequency waves are rounded to zero. These artifacts are particulary visible in very high contrast images with sharp edges, such as text or simple vector images.

Block coding artifacts Due to the JPEG compression method working on 8×8 blocks, discontinuities at the block boundaries may become visible. With a very low quality setting, only the DC coefficient may be available for every block, leading that block to be a single color. See figure 1.4 for an example image where these artifacts are clearly visible.

Color distortion When the chroma components are subsampled, much of the fine detail is lost. The colors appear more washed out as they correspond to a larger area.

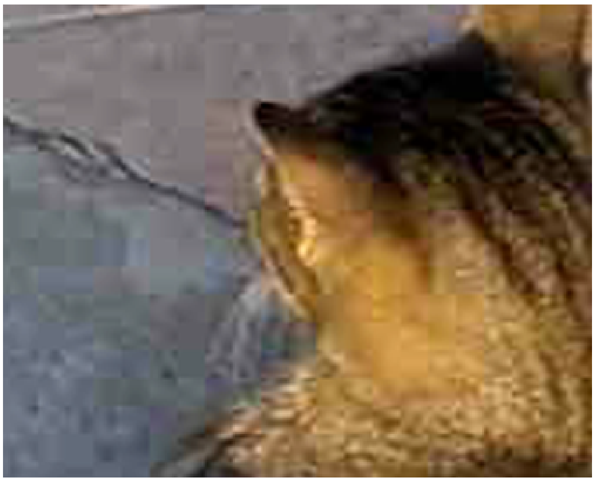

Figure 1.4: *Block coding artifacts:* the 8×8 blocks used internally are clearly visible

Posterization Posterization of an image is the loss of gradual change in tone. Whereas in the original image a color may smoothly go from one color to another, in the compressed image this may be a series of abrupt steps. This can be caused by both chroma component subsampling and quantization of these components.

1.3 Prior work

Automated artifact removal methods exist that aim to remove or reduce these artifacts as a post-processing step. These range from hand-crafted algorithms to algorithms with learned parameters. *TODO: Expand on non-DL approaches*

1.3.1 Deep learning

Every year the *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*[1] [10] is held with as goal to create a system which can automatically determine the content of photographs. In 2010 and 2011 SVM approaches won the classification part of the competition, but ever since 2012 convolutional neural network (CNN) approaches - pioneered in this competition by Krizhevsky, Sutskever and Hinton - have won the competition [6]. These CNN approaches are the state of the art for automated image and speech recognition [4][1]. Machine learning methods which use these connectionist models with many layers (making them *deep*) is often referred to as *deep learning.*

Contrary to regular methods such as SVMs (support vector machines), deep belief networks can automatically learn high level features in a supervised manner [5]. This

[1]http://image-net.org/

is called representation learning. Features learned with automatic methods often outperform hand-crafted feature extractors.

Deep learning methods work by learning and representing features in a hierarchical manner. For image classification for example, in the first layers simple shapes are used as features such as edges or corner. Deeper layers use (possibly non-linear) combinations of the features learned in earlier layers. If for instance a network is trained to recognize faces it may learn features such as eye color and nose shape in one layer, and learn that to best classify a certain person it is the combination of a high activation on the blue eye color, and a lower activation on the other colors.

This representation learning makes it a good candidate for the problem of JPEG compression artifact removal. With the advent of fully convolutional neural networks this can also be applied efficiently to dense prediction tasks like we are dealing with here[7]. Such a network could learn to recognize the artifacts, and reconstruct the image before this artifact was introduced.

1.3.2 Deep learning approaches for artifact removal

TODO: Expand

Dong et al. applied a deep learning model to the problem of artifact removal [3]. They proposed a four layer CNN, AR-CNN, which outperformed existing methods for artifact removal at the time of publication. They note that adding layers comes with great difficulty, as it may not converge. To counter this, they adopt a *easy-to-hard* transfer strategy, where they first train a network to remove artifacts from images compressed with a high quality setting. Then they use the learned weights to initialize a network destined for lower quality setting JPEG images. The same strategy was also used to train deeper networks, where they initialized a five layer network with the weights of an earlier trained four layer network. Svoboda went deeper by adding layers with skip connections making it a residual architecture [13]. This likely makes it easier for the latent clean image to propagate through the network. Technically, however, not a residual function is learned, as opposed to the original ResNet architecture they concatenate the features instead of summing them.

Zhang et al. were able to improve upon these methods considerably [17]. By instead of setting the original image as the training goal, the residual of this image was used as a target. This removes the necessity of a latent image representation in the hidden layers of the network. Also interesting is that this network was not only able to do JPEG artifact removal, but also Gaussian denoising and single image super-resolution.

1.3.3 Sparse coding approaches

TODO

Table 1.1: *LIVE1 dataset validation metrics of prior methods for JPEG artifact removal. To do: Finish this, or leave it out.. Decisions..*

Method	PSNR	PSNR-B	SSIM
Quality factor 10			
JPEG	27.77	0.7905	25.33
SA-DCT	28.65	0.8093	28.01
AR-CNN AR-CNN [3]	28.98	0.8217	28.70
L4 Residual [13]	29.08	0.824	28.71
CAS-CNN MS loss [2]	29.36	0.830	28.92
CAS-CNN w/ loss FT [2]	29.44	0.833	29.19
Quality factor 20			
JPEG	30.07	0.8683	27.57
SA-DCT	30.81	0.8781	29.82
AR-CNN [3]	31.29	0.8871	30.76
L4 Residual [13]	31.42	0.890	30.83
L8 Residual [13]	31.51	0.891	30.92
CAS-CNN MS loss [2]	31.67	0.894	30.84
CAS-CNN w/ loss FT [2]	31.70	0.895	30.88

1.4 Exploiting JPEG redundancies further with color

Most, if not all, prior work on JPEG compression only consider grayscale images. As the Y, Cb and Cr components of the image are processed in the same way it is argued that these methods also apply color images by simply applying it to each of these components independently. While it is true that this works, the components are not processed in exactly the same way. First of all, for the chroma components often a quantization matrix with higher values is used (which means it is more lossy). Secondly, it is very common that the chroma components are first downscaled (e.g. with a 2x2 factor) prior to splitting into the 8×8 pixel blocks. To apply methods only trained on grayscale images, one should thus first downscale the color components to match the 8x8 DCT blocks.

JPEG compression artifact reduction mostly relies on the redundancies of adjacent 8×8 blocks of pixels, especially those that operate (partially) in the DCT domain. The DCT blocks of the chroma components, however, describe a 16×16 block in the original image. These redundancies are thus at a larger spatial scale than the Y component's block redundancies. As the luma and chrominance are likely very highly correlated on a spatially local scale, the larger scale information of the chroma channels can be used to improve the luma channel, and vice versa.

Also, due to the more aggressive quantization of the chroma channels, the possible interval of original value of one of the DCT coefficients is large, especially for the high frequency components. This is finer for the luma channel, this information can be used to better infer the original coefficients.

1.5 Evaluative measures

To determine the efficacy of a method for compression artifact removal several quantitative methods are available. Generally it is a distance measure between the restored image and the original image prior to compression.

Mean squared error (MSE) MSE is the simplest measure, often used as the loss function for methods that rely on optimization. It is simply the square of the distance per pixel averaged over all pixels in the dataset. It is given by

$$MSE = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} (x_i - y_i)^2,$$

with N the total amount of pixels, y a predicted pixel, and x the pixel prior to compression.

Peak signal-to-noise ratio (PSNR) Peak signal-to-noise ratio is given by the the ratio of noise to the maximum possible power of the signal, which in this case is the image. It is most often expressed on the logarithmic decibel scale. PSNR in dB is given by

$$PSNR = 10 \cdot log_{10}(\frac{DEPTH^2}{MSE}),$$

with $DEPTH$ the maximum pixel value of the image (255 for 8 bit images).

Structural similarity (SSIM) The Structural Similarity index was developed as an improvement to MSE and PSNR with the goal of being more consistent with how humans perceive images[15]. The index consists of a weighed combination of comparison measures between samples of luminance, constrast and structure across the whole image. These samples are taken with a sliding window approach (generally with an 8×8 window).

Peak signal-to-noise ratio including blocking effects (PSNR-B) PSNR-B is an adaptation of the PSNR introduced by Yim et al. to account for the blocking effect often introduced by image or movie compression algorithms [16]. They introduce the mean squared error including blocking effects *MSE-B*, which includes a blocking effect factor *BEF* which is determined from the pixel values on the boundary of the blocks (8×8 in the JPEG case):

$$MSE\text{-}B = MSE + BEF,$$

the PSNR-B then becomes

$$PSNR\text{-}B = 10 \cdot log_{10}(\frac{DEPTH^2}{MSE\text{-}B}).$$

1.6 Datasets

A large amount of different datasets exist for benchmarking and training of image processing, segmentation and classification techniques. For the compression artifact reduction task any image is potentially suitable as ground truth. Ideally, however, the image used as ground truth (prior to JPEG compression) is uncompressed or compressed with a lossless compression algorithm. Below is a list of datasets used in prior work, and other datasets identified as usable in this context. The majority of these datasets are images that have been compressed with a lossy algorithm, albeit it with a high quality factor. *TODO: some accompanying text on each of these.*

LIVE1 dataset [12] 29 grayscale images.

Berkeley Segmentation Dataset (BSD500) [8] Training set of 200 images, validation set of 100 images and a test set of 100 images.

Places365-Standard dataset [18] Training set of 1,803,460 images, validation set of 36,500 images and test set of 328,500 images.

Uncompressed Colour Image Dataset (UCID) [11] 1338 truly uncompressed images taken with the same camera.

1.7 Relevance and impact

To save on bandwidth and storage costs, images are often compressed using a lossy compression algorithm. The compression ratios of these lossy methods is often much better than those of lossless compression algorithms. This compression may result in a noticable degradation in image quality.

This compression often happens unknowningly for the average user. An image attached to a Twitter tweet, uploaded to other social media, or saved in a cloud storage service (such as Dropbox) may be compressed without the user knowing.

Also, sometimes digital cameras do not even support the option of storing a lossless version of the original image.

Removing the artifacts introduced by lossy compression can allow for restoring the higher quality original image in these cases.

CHAPTER 2

Methodology

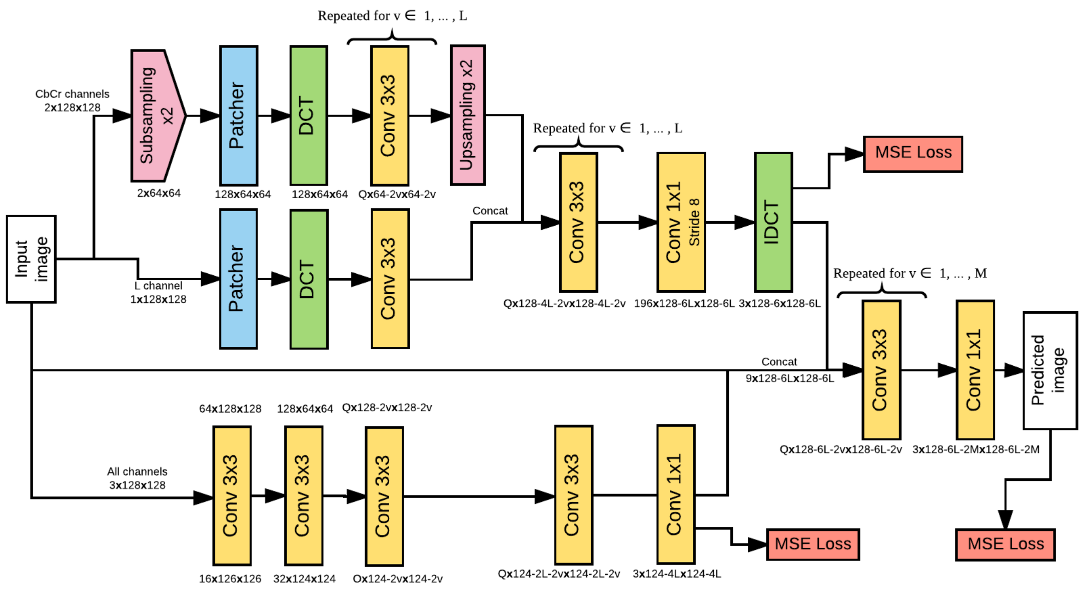

Figure 2.1: *DDC-Net:* Proposed architecture which can exploit additional redundancies due to subsampling of the chroma components in JPEG compression. Every convolution layer is implicitly followed by a ReLu activation function.

2.0.1 DCT domain branch

The first part of the DCT domain part of the network starts by splitting the Y (luma) component from the CbCr (chroma) components. The chroma component branch first features a downsampling of two in both x and y dimensions, this is done to match the 8×8 coding blocks used in JPEG. Then in both the Y and the CbCr branch L convolution layers with a Q 3×3 filters are stacked designed to exploit the redunancies present in the neighboring 8×8 JPEG coding blocks.

Prior to merging these two streams, the chroma component branch's feature maps are upscaled by a factor of two to match the luma branch spatially. This is followed by another series of L convolutions. The main idea is that the chroma branch's pixels describe a 16x16 area, whereas in the luma branch this is a finer 8x8 pixels, fusing the information of both may improve both.

2.1 Data Augmentation

Artificially increasing the amount of data by adding variations of the original data can help train a model that generalizes well. Augmentation consists of applying (random) pertubations to the samples in ways that do not change the targets of these samples. For the problem of JPEG compression artifact removal we can simply determine the new targets as long as we perform these augmentations prior to generating the compressed version.

Expanding the dataset through augmentation has a regularizing effect which helps prevent overfitting. Especially effective are augmentations that are also realistic examples of real world data. As such, we perform the following augmentation methods which we believe yield realistic new samples:

- **Flips** The images are randomly mirrored in the X and/or Y direction, both with a 0.5 probability.
- **Rotations** The images are randomly rotated 0, 90, 180, or 270 degrees with equal probability.
- **Zooming** The images are randomly magnified by a factor between 0.9 and 1.1 with uniform probability.
- **HSV jittering** HSV is a colorspace that represents a color image in three channels; a *hue* (color), *saturation* (vibrance) and a *value* (brightness) channel. We randomly jitter the hue, saturation and vibrance of an image by multiplying it with a random value between -1.075 and 1.075. The values of all pixels are then clipped between 0 and 1.

2.2 Implementation Details

All methods were implemented using the publicy available PyTorch library for tensor computation and deep neural networks, which saw its beta release in January 2017. This library allows for GPU acceleration with CUDA and cuDNN primitives. All models were trained on a system with 16GB of RAM, an Intel i5-4670K at 3.6GHz, and an Nvidia Gefore GTX1080 graphics card.

CHAPTER 3 Schedule

TODO: Obviously this won't be in the final version :-) This chapter describes a rough planning for the full thesis project. It is hard to foresee the time required for the individual goals of the project which makes it likely that time lines will shift.

December-January	Proposal, background study	Kicking off the project, writing the project proposal. Final version by January.
January	Framework, explorative experiments, write background	Create environment for testing and running experiments. Finish background chapter for thesis, detailing the inner working of JPEG, overview of artifacts, and existing (non-deep learning) approaches for artifact removal (5 to 7 pages?).
February	Gather datasets, implement architectures	Collect datasets to evaluate the project's system. If feeling ambitious, also generate random vector images. Apply the network architectures described in section 2.1, as well as tweaking their hyperparameters. Aside from tweaking the parameters, different pre and postprocessing methods such as data augmentation will be explored to further increase the performance of the networks. Finish methods chapter of thesis (5 pages?).
March-March	Further tweaking, evaluate different initialization and sampling strategies	Proper initialization is very important for this problem. Evaluate some different initialization strategies as described in literature, and continue tweaking architectures. Collect first results and describe these. Create shell for results chapter (around 3 pages?)
Early April	Run final experiments, gather and describe results.	
April	Buffer and finalizing the thesis	Buffer for previous steps, room for stretch goals, writing the final version of the thesis. Late April or start of May: thesis presentation and defense, concluding the project.

Bibliography

[1] Ossama Abdel-Hamid et al. “Convolutional neural networks for speech recognition”. In: *Audio, Speech, and Language Processing, IEEE/ACM Transactions on* 22.10 (2014), pages 1533–1545.

[2] Lukas Cavigelli, Pascal Hager, and Luca Benini. “CAS-CNN: A Deep Convolutional Neural Network for Image Compression Artifact Suppression”. In: *CoRR* abs/1611.07233 (2016). URL: http://arxiv.org/abs/1611.07233.

[3] Chao Dong et al. “Compression artifacts reduction by a deep convolutional network”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pages 576–584.

[4] Yanming Guo et al. “Deep learning for visual understanding: A review”. In: *Neurocomputing* (2015).

[5] Yoshua Bengio Ian Goodfellow and Aaron Courville. “Deep Learning”. Book in preparation for MIT Press. 2016. URL: http://www.deeplearningbook.org.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pages 1097–1105.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pages 3431–3440.

[8] D. Martin et al. “A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics”. In: *Proc. 8th Int’l Conf. Computer Vision*. Volume 2. July 2001, pages 416–423.

[9] M. Narasimha and A. Peterson. “On the Computation of the Discrete Cosine Transform”. In: *IEEE Transactions on Communications* 26.6 (1978), pages 934–936. ISSN: 0090-6778. DOI: 10.1109/TCOM.1978.1094144.

[10] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pages 211–252. DOI: 10.1007/s11263-015-0816-y.

[11] G. Schaefer and M. Stich. "UCID: an uncompressed color image database". In: *Storage and Retrieval Methods and Applications for Multimedia 2004*. Edited by M. M. Yeung, R. W. Lienhart, and C.-S. Li. Volume 5307. December 2003, pages 472–480. DOI: `10.1117/12.525375`.

[12] H. R. Sheikh et al. *LIVE Image Quality Assessment Database Release 2*. April 2014.

[13] Pavel Svoboda et al. "Compression Artifacts Removal Using Convolutional Neural Networks". In: *CoRR* abs/1605.00366 (2016). URL: `http://arxiv.org/abs/1605.00366`.

[14] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pages 1–9.

[15] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pages 600–612.

[16] Changhoon Yim and Alan Conrad Bovik. "Quality assessment of deblocked images". In: *IEEE Transactions on Image Processing* 20.1 (2011), pages 88–98.

[17] Kai Zhang et al. "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising". In: *CoRR* abs/1608.03981 (2016). URL: `http://arxiv.org/abs/1608.03981`.

[18] Bolei Zhou et al. "Places: An Image Database for Deep Scene Understanding". In: *CoRR* abs/1610.02055 (2016). URL: `http://arxiv.org/abs/1610.02055`.