Convolutional Neural Network with Inception Blocks for Image Compression Artifact Reduction

Purbaditya Bhattacharya
Department of Signal Processing and Communication
Helmut Schmidt University
Hamburg, Germany
bhattacp@hsu-hh.de

Udo Zölzer
Department of Signal Processing and Communication
Helmut Schmidt University
Hamburg, Germany
zoelzer@hsu-hh.de

***Abstract*—A convolutional neural network is proposed for the reduction of artifacts introduced by JPEG compression. In this work the existing DnCNN architecture has been considered as the basis network and it is extended by introducing inception blocks on top of a normal convolution block consisting of convolution, rectified linear unit and batch-normalization layers. Each inception block contains parallel convolution blocks with various dilation factors, essentially performing Atrous convolution. Compression artifacts result in multiple image neighborhoods or blocks having similar or redundant information. In spatial domain the Atrous convolution introduces a larger receptive field while ignoring redundant neighboring pixels due to the dilation, thus helping in a better reconstruction. The proposed network is trained on a small dataset as used by the basis network. Evaluation of this network on the standard test datasets shows a substantial quantitative and qualitative improvement, particularly for higher compression rates.**

***Index Terms*—Convolutional neural network, compression artifact reduction, image processing**

I. Introduction

Compression standards like JPEG performs fast lossy compression of an image which attempts to reduce its size while retaining its perceptual quality. Such kind of compression is very useful in order to save bandwidth and storage space. However, large compression ratios introduce undesired image artifacts, which make the image perceptually degraded. Such artifacts primarily include block artifacts, undesired blurring and ringing around the edges. Additionally subsequent image processing or computer vision tasks including image super-resolution, deblurring, object detection and segmentation or contour detection among others, perform relatively poorly when executed on a compressed image. Although there are improved compression standards like JPEG2000, WebP or HEVC-MSP, they are relatively slower compared to JPEG encoding and decoding process and are yet to be used widely. Image compression and decompression approaches with the help of machine learning and convolutional neural networks also exist in abundance but are usually computationally expensive. Hence an alternate approach is the use of a convolutional neural network as a post-processing tool to a decompressed image, in order to reduce compression artifacts and improve the image quality.

In the JPEG compression standard the image is usually divided into 8×8 pixel blocks and discrete cosine transformation (DCT) is applied on each block. In the following step the transform coefficients are quantized based on a standard quantization table resulting in a substantial loss of high frequency information thus blurring the image. Different quantizations in adjacent blocks also occur leading to discontinuities along the block edges, hence producing blocking artifacts during image reconstruction. Ringing effects along the object edges is also a result of the coarse quantization of the high-frequency components.

Classical approaches for image deblocking or blocking artifact removal include the application of image filtering and block overlap methods [1], adaptive [2], [3] and optimal filtering [4] methods, filtering methods in spectral or cepstral domain [5], [6], and method based on wavelet transform [7]. Subsequently, data driven learning based approaches based on sparse coding and dictionary learning [8], [9] showed improved performances. Recent approaches tend mostly towards the application of deep learning because of its improved performances in solving many computer vision problems. Convolutional neural networks (CNN) have been developed for many image enhancement problems e.g., denoising [10], [11], [12] and super-resolution of color [13], [14], [15] and infra-red images [16], [17] as well as compression artifact removal [10], [15], [18], [19]. Most of the CNN models learn an end-to-end mapping between a low quality image and the corresponding high-quality desired image. The AR-CNN [18] network is one of the earlier neural network model which proposed a shallow end-to-end model for image deblocking and achieved improved results. Its network architecture consists of layers to extract feature maps of different depths, where the features are initially compressed, enhanced, and then expanded subsequently to reconstruct the final high quality image. The CAS-CNN network [19] has introduced an architecture which extracts and enhances the features from multiple resolutions of the low-quality image or feature maps before recombining them to construct the high-quality image. The aforementioned network also has a multi-stage supervision and is trained multiple times before the final inference. The DnCNN network [10] introduced batch-normalization layers in a deep residual network with consistent feature map depths. The batch-normalization layers along with a gradient clipping approach have helped in a fast minimization and convergence

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

of the training objective, yielding improved image quality. In [15], MemNet is introduced which employs a large number of densely connected gated recurrent units for improved image restoration purposes. A very deep multi-resolution network (MWCNN) has been introduced in [20] consisting of multi-level wavelet decomposition of the feature maps. It has been trained with a large number of images and has shown a very good image restoration performance.

The network proposed in this work assumes the DnCNN network as a basis and replaces a selected number of intermediate layers with inception blocks. The inception block or module has been introduced in [21] which consists of multiple parallel convolution layers with varying filter sizes in order to extract features across different neighborhood sizes or receptive fields. In the proposed network such inception blocks are used with filters having different dilation factors and depths. The dilated filters perform Atrous convolutions and thereby compute the filter responses at various higher resolution without increasing the number of learnable filter parameters. Such an approach results in an improvement of the deblocking and restoration performance when compared to that of the basis network. Multiple experiments have been conducted with the proposed network architecture having varying network depth and feature map depths, and their performances are evaluated.

II. Proposed Model

The CNN model is an end-to-end structure containing a cascade of alternating inception blocks and normal convolution blocks, where these 2 blocks always appear as a pair until the penultimate network layer. Initially the RGB images from the dataset are converted to YCbCr images and the Y channel, which is the luminance component, is extracted. The Y channel is then resampled by multiple down and upsampling factors to enlarge the amount of training images. The low quality images are generated by compressing them with different factors by the JPEG compression algorithm. In the next step overlapping patches of size 48×48 are cropped from the low and high quality images. The low quality patches are used as inputs to the model while the original patches are used as the ground truth examples. The following section describes the proposed network architecture and the training of the model.

A. Network architecture

As mentioned earlier, the proposed network is a cascaded structure primarily consisting of convolution, activation and batch-normalization layers. The network architecture is illustrated in Fig. 1. It is a cascade of several inception and convolution blocks, the convolution block being similar to the one used in DnCNN. The convolution block consists of a convolution layer (Conv) having a filter of dimension $3\times3\times D_{inp}$, where D_{inp} will denote the depth of the input feature maps or input image. The number of such filter tensors in the convolution layer determines the depth of the output feature map. Each coefficient in the filter tensor is randomly initialized in order to get a decorrelated set of filters and they are gradually adapted during the optimization process. The convolutional layer usually performs a cross-correlation operation between the filters and the input feature maps. This layer is followed by the batch-normalization layer (BN) which is given by

$$y_i = \gamma \left(\frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \right) + \beta, \tag{1}$$

where x_i and y_i represent an input and output element of the layer respectively, μ_B and σ_B^2 represent the mini-batch mean and variance respectively, and γ and β represent two learnable parameters. This layer initially normalizes the input feature maps over a mini-batch in order to reduce the internal covariance shift [22] between input feature maps from different layers. The layer is also parameterized by a scaling factor and a shifting factor which are also adapted during the optimization process. Batch-normalization makes the training method more stable allowing higher learning rates and with a proper initialization it helps the network converge faster. The batch-normalization layer is followed by a rectified linear unit (ReLU) which is given by

$$y_i = \max(0, x_i), \tag{2}$$

where x_i and y_i denote an input and output element of the ReLU layer respectively. This activation layer makes the output feature map sparse resulting in faster computation and prevents the vanishing gradient problem which usually occurs when exponential or quadratic activation functions with large saturation regions are used instead.

The inception layer contains parallel convolution blocks having different filter settings. As shown in Fig. 1 it contains 5 parallel convolutional blocks. The filters used in the convolutional blocks have the same number of learnable coefficients but have different dilation factors, resulting in dilated filters. A dilated filter upsamples the filter by inserting zeros in between the coefficients thus increasing the filter's receptive field. The dilated filters perform Atrous convolution which has been used before in stationary or non-subsampled wavelet transforms as an alternative to discrete wavelet transform, where the passband features are downsampled instead. Therefore this operation is an alternative for a downsampling operation of features. Hence, the proposed network draws similarities with the previously mentioned networks like CAS-CNN [19], and MWCNN [20], which perform feature map downsampling. In a compressed image there are certain regions or neighborhoods where the information is very redundant and a small filter kernel cannot extract and construct sufficiently good features from such neighborhoods. Hence dilated filters help in aggregating multi-scale context which benefits the final reconstruction of the image. The first convolution block inside the inception block uses a standard 3×3 filter in its convolution layer and produces an output feature tensor with the same depth D_{in} as the input feature tensor. The other convolution blocks use 3×3 filters with dilation factors of 1, 2, 4, and 6 respectively while the depth of the output feature maps from those convolution blocks are $\frac{D_{in}}{2}$, $\frac{D_{in}}{4}$, $\frac{D_{in}}{8}$, and $\frac{D_{in}}{8}$

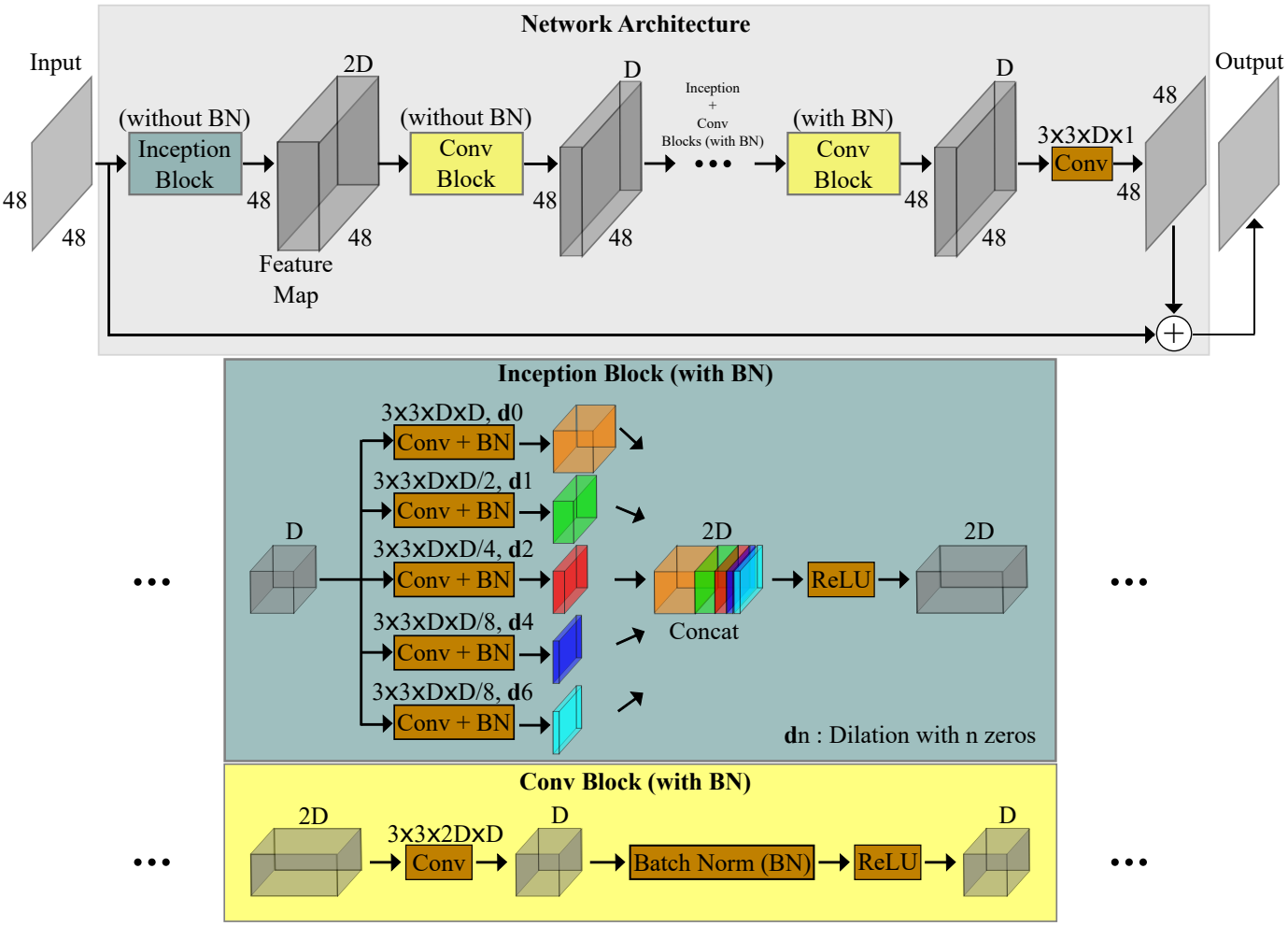


Fig. 1: Architecture of the proposed CNN ($Arch1$) with inception blocks, containing filter-banks of different dilation factors, and normal convolution blocks.

respectively. It is noteworthy to mention that, the dilation factor, in this context, refers to the amount of zeros inserted between two consecutive coefficients. Each convolution layer is followed by a batch-normalization layer. The feature maps from the parallel convolution blocks are concatenated to create a feature tensor of depth $2\times D_{in}$ which is an input to the ReLU layer.

Initially an image patch serves as the input to the first inception block which does not contain any batch-normalization layer and the generated feature maps are used as the input to a convolution block, also without any batch-normalization layer. This is followed by the cascaded inception-convolution block pairs with batch-normalization, as described previously. After the last convolution block layer there is only one convolution layer which reconstructs a residual image patch. The input image patch is added to this residual patch via a skip connection, thus yielding a residual network. The predicted image patch acts as an input to the loss layer where the error is calculated with respect to the ground truth patch. Experiments have been conducted with multiple inception-convolution block pairs as long as improvements in test images could be quantitatively and qualitatively observed.

B. Objective Function

The objective of the training process is to minimize the mean squared error (MSE) between the high quality ground truth patches and the estimated patches. The CNN is trained iteratively to minimize the loss function in order to find an optimal solution for the free or trainable parameters of the convolutional and batch-normalization layers. The objective function can be formulated as

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \;\; E_{y,\hat{y}}\left[L_2(y,\hat{y})\right], \tag{3}$$

where

$$\hat{y} = F(\theta, x), \tag{4}$$

$$L_2(y,\hat{y}) = \frac{1}{N}||y-\hat{y}||_2^2. \tag{5}$$

In the above set of equations, θ^* denotes the optimal solution of the free or trainable parameters denoted by θ, and

$E_{y,\hat{y}}[\cdot]$ denotes the expected value of the cost function. $F(\theta, x)$ denotes the CNN function parameterized by θ with x being its input and $\hat{y}$ being the predicted output. L_2 denotes the MSE function, and y denotes the ground truth. In addition to the MSE function, a L_2 regularization of the prediction is also done with a very small influence factor. Apart from the L_2 loss, L_1 loss and a combination of both losses are also experimented without any noticeable improvement in the final results.

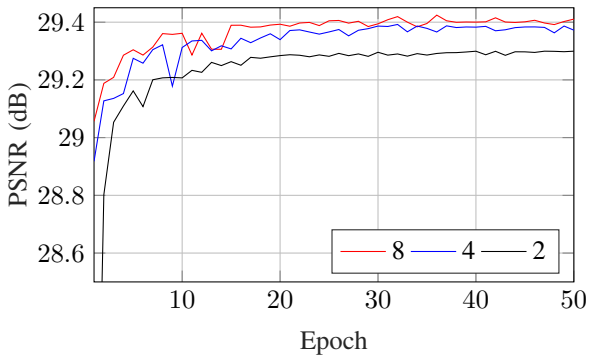
C. Hyperparameter Setup and Experiments

The network is trained separately for compression ratios of 10, 20, and 30 to generate three models. A batch of 128 image patches is used per iteration and the image patches are flipped randomly during data augmentation. The network parameters are initialized with the initialization given by [23]. In order to update the weights, the adaptive momentum (Adam) optimization [24] is used, with its default hyper-parameter settings. An initial learning rate of 0.001 is selected during training and the learning rate is divided by a factor of 10 every 20 epochs. Gradient clipping [25] is also done during training for fast convergence and the absolute value of the clipping threshold is set to 0.005. The networks are trained for 60 epochs, although it usually converges and becomes stable after 40 epochs. It has been observed that a higher learning rate usually results in a poorer performance compared to the used learning rate. A weight decay of 0.0001 is used for parameter regularization. Due to memory constraints, 2 sub-batches have been used to process the overall batch of 128 examples per iteration. The learning rate factor for the bias is lower than the weight in the first and the last convolution layers and no bias has been trained in the intermediate blocks.

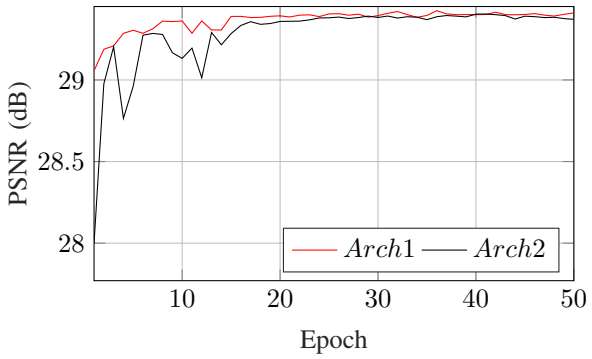
Different network configurations are experimented with during the work. The experiments are primarily done by varying the network depth, i.e, the number of inception-convolution block pairs with batch-normalization as well as the number of filters in the convolution layers. Networks are built with 2, 4, and 8 pair of inception-convolution block pairs to study the impact of the network depth on the CNN outcome. Their performances on test datasets are shown in Table. I shows the qualitative results on the benchmark dataset Classic5 and LIVE1 [26] for different network depths of the CNN $Arch1$ as shown in Fig. 1. The tabulated values indicate an average improvement of 0.06 dB when the number of inception and convolution block pair increases from 2 to 4. Further extension of the network shows a reduction in the average quantitative improvement which may be attributed to the overfitting of the network to insufficient augmentation of the small dataset. Referring to Fig. 1, the depth of the output feature tensors from the inception block is usually twice as large (2×D) as the depth of the input feature tensor (D). We refer to this network as $Arch1$.

TABLE I: Average PSNR (dB) and SSIM values for a quality factor of 10 on the Classic5 and LIVE1 dataset tested with CNNs ($Arch1$) of different depths.

	Classic5		LIVE1	
No. of inception and conv block (with BN) pairs	PSNR (dB)	SSIM	PSNR (dB)	SSIM
2	29.46	0.805	29.28	0.814
4	29.53	0.808	29.33	0.816
8	29.56	0.809	29.35	0.817



(a) Performance over Network ($Arch1$) Depth



(b) Performance over Feature Depth

Fig. 2: Performance of the CNN on combined dataset of Classic5 and LIVE1 for (a) for 2, 4, and 8 inception-convolution block pairs (with BN) for quality factor 10 and (b) for 8 inception-convolution block pairs (with BN) but with different feature map depths, where $Arch1$ refers to the CNN with depth 128 and $Arch2$ refers to the CNN with depth 64 after each inception block.

Another similar network is also built where the output feature map depth after the inception block remains the same as the input feature map depth (D) and we refer to it as $Arch2$. In this network the feature map depth in the hidden layers remain the same. This is done by modifying the convolution layers inside the inception blocks, where the number of filters in each convolution layer is halved. The original and this modified network are also evaluated and their performances are compared. Fig. 2 shows the average network performance per epoch, on test images from benchmark datasets Classic5 [5] and the LIVE1 [26] , in terms of peak-signal-to-noise ratio or PSNR (dB), for the first 50 epochs. It can be noted from the plots that PSNR rapidly increases in the first few epochs

TABLE II: Average PSNR (dB) / PSNR-B (dB) [30] / SSIM (rounded to 3 decimals) for quality factors 10, 20, and 30 on Classic5 and LIVE1 datasets by different CNNs.

Dataset	Quality	JPEG	ARCNN	DnCNN	CNN (Ours)
Classic5	10	27.82 / 25.21 / 0.760	29.03 / 28.76 / 0.793	29.40 / 29.14 / 0.803	29.59 / 29.21 / 0.810
	20	30.12 / 27.50 / 0.834	31.15 / 30.60 / 0.852	31.63 / 31.20 / 0.861	31.83 / 31.38 / 0.865
	30	31.48 / 28.94 / 0.867	32.51 / 31.99 / 0.881	32.91 / 32.39 / 0.886	33.12 / 32.48 / 0.890
LIVE1	10	27.77 / 25.36 / 0.773	28.98 / 28.74 / 0.810	29.20 / 28.96 / 0.813	29.39 / 29.06 / 0.819
	20	30.07 / 27.61 / 0.851	31.29 / 30.85 / 0.873	31.59 / 31.16 / 0.880	31.81 / 31.36 / 0.885
	30	31.41 / 28.97 / 0.899	32.68 / 32.26 / 0.904	32.98 / 32.46 / 0.909	33.20 / 32.64 / 0.913

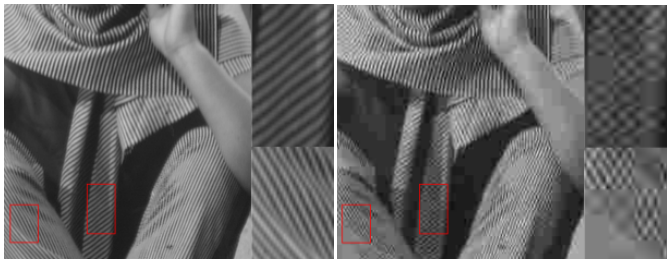
and then vary slowly. It can also be seen in Fig. 2 (a), that after 4 inception-convolution block pairs the improvement in test performance is negligible and hence the PSNR curves are very close to one another. Table I also shows a small improvement from 4 inception-convolution block pairs to 8. Fig. 2 (b) also shows a marginal improvement in performance for $Arch1$ compared to $Arch2$. Experiments are also done with different combination of dilation factors as well. The choice of reducing the number of filters for higher dilation factors is taken because larger number of filters led to higher computational load without any significant improvement in reconstruction quality. Dilation factors of more than 6 did not lead to any further improvement.

III. Evaluation

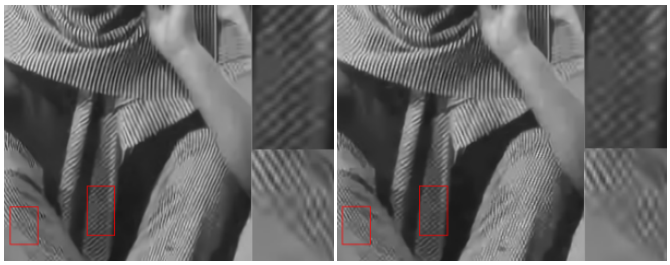
The proposed network is trained with a dataset originally referred in [27] and used by DnCNN. It is a part of the Berkley segmentation database [28] and contains 400 images of size 180 px $\times$ 180 px. The dataset is initially augmented and multiple patches from these images are extracted with a certain overlap in order to create the final training database. The low quality compressed images are produced by the MATLAB JPEG coder. The training is performed in the MatConvNet [29] environment on a machine with Nvidia Titan Xp graphical processing unit. The final network consists of 12 inception-convolution block pairs, since no significant improvement is observed in the performance with a deeper architecture. For testing the performance of the network Classic5 and the LIVE1 datasets are used. The Classic5 dataset contains a set of 5 grayscale images widely used as benchmark images for various computer vision and image processing applications. LIVE1 dataset contains 29 color images, from which the Y-channel is extracted to do the evaluation. For qualitative evaluation, PSNR and the structural similarity index metric (SSIM) is used. Along with these standard metrics a third metric introduced in [30], called PSNR-B is also used for evaluation. This objective metric introduces a factor which is proportional to any kind of blocking artifacts within an image and is more aligned towards subjective perception. For image compression the standard Matlab JPEG codec is used where different compression ratios are achieved by selecting an image quality factor metric between 0 and 100. Here, a quality factor of 0 indicates maximum compression and a quality factor of 100 indicates no compression. As an example, a quality factor of 10 yields nearly 94 % average compression in terms of size or data saving, and a quality factor of 20 results in nearly 90 % average compression or data saving on the LIVE1 dataset. In this work, experiments are performed for quality factors of 10, 20, and 30 respectively.

The test results are compared with the basis DnCNN, which is the primary objective of the proposed work, and with ARCNN, which has the hourglass model of feature depth transitions as in $Arch1$ described before. However, the test results are close or comparable to some state-of-the-art CNNs. It is also necessary to mention that the evaluation is performed exactly as it is done by the basis DnCNN network. This evaluation probably differs marginally in some networks including the network in [19], based on their published results. Table II summarizes the performance of our method for 3 quality factors and they indicate an improved performance over DnCNN. Indeed, in Table. I it can be observed that the shallowest network already performs better in terms of quantitative metrics when compared to the basis network and the results gradually improve with increasing depth. The average PSNR and average SSIM improves roughly by 1.7 dB and 0.05 respectively w.r.t JPEG , over the two benchmark datasets for the quality factor (QF) of 10. For QF=20 and QF=30, the corresponding improvements in PSNR are also about 1.7 dB, and the improvements in SSIM are about 0.035 and 0.02 respectively. Similar improvements occur with PSNR-B metric as well.

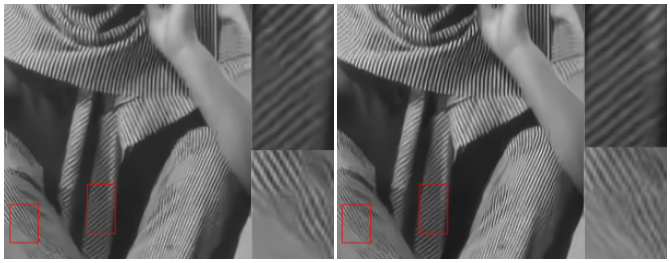
Some example images are selected from the test dataset and the cropped sections of the images are shown in this section. Fig. 3 shows the enhancement results for a quality factor of 10 on an example image from the Classic5 dataset. The image is cropped and presented for a better visualization of the details. It can be observed in the images that the reconstruction done by DnCNN and ARCNN has block artifacts in certain areas of high texture while the result from our CNN show that the same regions are relatively smoother in comparison. It is seen that our method is able to reconstruct the patterns on the scarf and the pants better than the other methods and remove certain artifacts. It can also be observed that the result of the deep network (12 inception-convolution blocks) is relatively better



(a) Ground Truth (PSNR/SSIM) (b) JPEG (25.79/.762)

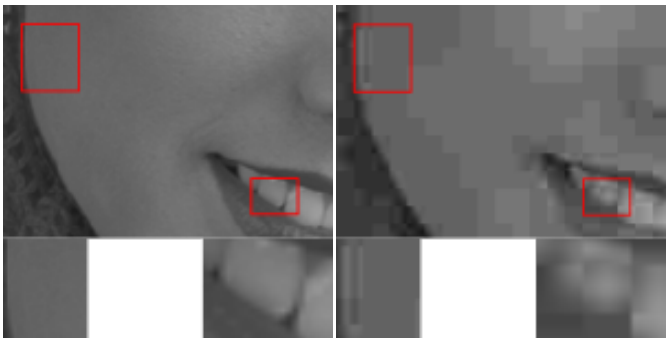


(c) DnCNN (27.59/0.816) (d) ARCNN (26.92/0.797)

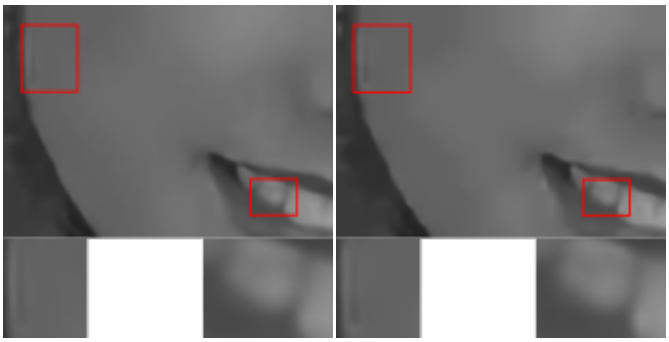


(e) CNN (shallow) (27.73/0.826) (f) CNN (deep) (27.79/0.828)

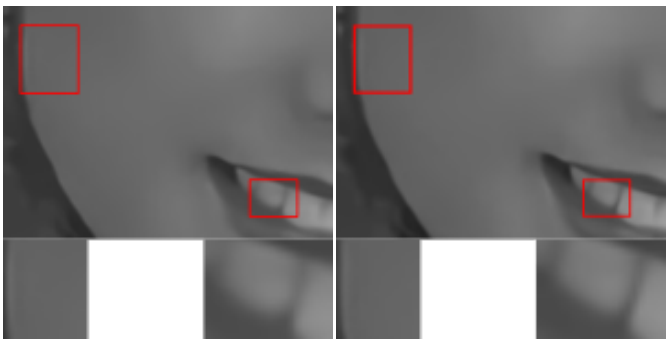
Fig. 3: An example of an original image, the JPEG image, and the enhancement results by different CNNs for a quality factor of 10 (best viewed on a screen).



(a) Ground Truth (PSNR/SSIM) (b) JPEG (30.49/.769)



(c) DnCNN (31.79/0.811) (d) ARCNN (31.70/0.808)



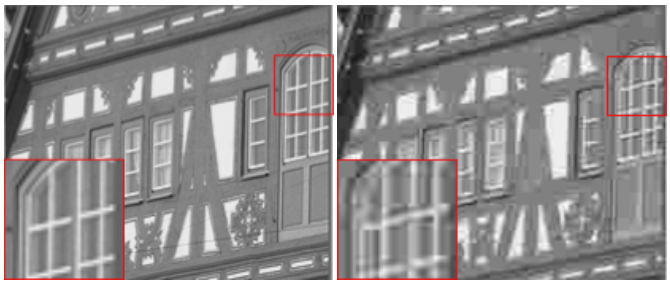
(e) CNN (shallow) (31.99/0.814) (f) CNN (deep) (32.07/0.817)

Fig. 4: An example of an original image, the JPEG image, and the enhancement results by different CNNs for a quality factor of 10 (best viewed on a screen).

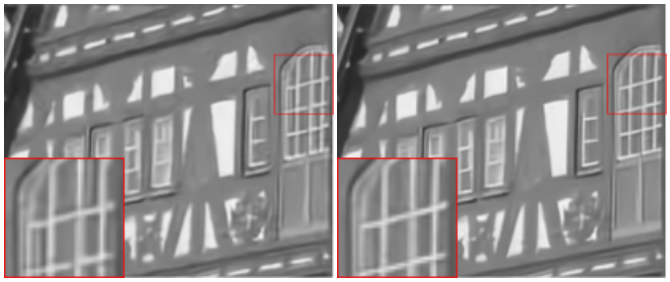
than the shallow network (4 inception-convolution blocks).

Similar improvements can also be observed in the example from the LIVE1 dataset shown in Fig. 4, by our method where the image is less blurry or sharper compared to the other methods. It also gets rid of a few small artifacts which are still present in the results of the other methods. It is however noticeable that certain regions of the image are not well reconstructed because of large blocks in the JPEG input image having no information at all.

Fig. 5 shows the results for a quality factor of 10 on the cropped section of another image from the LIVE1 dataset. This is an example where the relative improvements are very difficult to perceive in spite of the quantitative improvement. The image has many patterns on windows and edges, some of which are smoothly constructed by our method as well as DnCNN. However, there is a marginal improvement in sharpness compared to the DnCNN output. It can be observed further, that many such patterns on the reconstructed images do not resemble the ones in the original image. Fig. 6 shows a comparison between the reconstructed images for quality factors (QF) 10, 20, and 30. From the example images across the datasets one can conclude that the network is able to produce visually better results with the additional parallel convolution blocks with different dilation factors and the objective measures support the inference. The enhancement results usually appear comparatively sharp and detailed for factors of 20 and 30, while some artifacts still remain in the images for a quality factor of 10. It has also been observed during the work that deblocking and enhancement performance improve in terms of visual quality and artifacts, if the network depth is increased but the blurriness does not improve sufficiently. The small dataset used by the basis network is a limitation towards increasing network depth or width, while use of the MSE loss can also prevent the improvement of blurring effect.



(a) Ground Truth (PSNR/SSIM) (b) JPEG (25.08/.763)



(c) DnCNN (26.75/0.807) (d) CNN (deep) (26.99/0.817)

Fig. 5: An example of an original image, the JPEG image, and the enhancement results by different CNNs for a quality factor of 10 (best viewed on a screen).

(a) Ground Truth (PSNR/SSIM) (b) CNN (QF10) (28.48/.853)

(c) CNN (QF20) (31.08/0.907) (d) CNN (QF30) (32.58/0.930)

Fig. 6: An example of an original image and the enhancement results by our CNN for quality factors (QF) 10, 20, and 30 (best viewed on a screen).

IV. Summary and Conclusion

In this work we propose a convolutional neural network containing inception blocks with the goal of reducing JPEG compression artifacts. The network is a cascaded structure containing an inception block having multiple parallel convolution and batch-normalization layers, and a convolution block, similar to the one used in the DnCNN architecture. Each convolution layer inside the inception block contains dilated filters with various dilation factors generating feature maps of varying depths. The dilated filters increase their receptive fields by different amount and aggregates multi-scale features. Additionally, experiments are conducted with different network depths and number of filter tensors to study their influence on the overall reconstruction quality. Evaluation of the proposed network on the benchmark datasets exhibit an improvement in artifact suppression and reconstruction quality in comparison to the basis DnCNN network. The performance also improves with increasing network depth and number of filters. The small dataset and the redundancy introduced by the augmentation methods seem to limit performance of the proposed network. Therefore, a larger dataset can be used in order to improve network training. The quadratic loss function used for such kind of problems usually results in a smooth output image and suppressed high frequency content. Hence, additional regularizers for edge enhancement or edge focused objective functions can be used for improving the qualitative results. The CNN architecture can be extended by introducing short and long skip connections in between different modules, since densely connected networks have shown improved performances in multiple regression based computer vision tasks. In addition to the previous modifications, the number of inception blocks can also be reduced by replacing them with a deeper inception block yielding an alternative architecture for the same task or adding attention modules inside the inception blocks. Finally, other low level computer vision tasks like impulsive or salt-and-pepper noise reduction and deblurring or super-resolution can be addressed by the network since multi-scale feature aggregation can improve the performance of those applications as well.

References

[1] H. Reeve and J. Lim, "Reduction of blocking effect in image coding," ICASSP '83. IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol.8, pp. 1212–1215, April 1983.

[2] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking filter," IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 614–619, July 2003.

[3] J. Wang, J. Kim, and J. Jeong, "An new deblocking algorithm for DCT coded images using adaptive spatial filters," 2009 IEEE International Conference on Network Infrastructure and Digital Content, pp. 813–818, November 2009.

[4] K. Daehee, and H. Yo-Sung, "A Method for Blocking Effect Reduction Based on Optimal Filtering," Advances in Multimedia Information Processing - PCM 2004, Springer Berlin Heidelberg, pp. 135–142, 2005.

[5] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images," IEEE Transactions on Image Processing, vol. 16, no. 5, pp. 1395–1411, May 2007.

[6] N. I. Cho, "Reduction of blocking artifacts by cepstral filtering," Signal Processing, vol. 81, pp. 633–642, 2001.

[7] A. W-Liew and H. Yan, "Blocking artifacts suppression in block-coded images using overcomplete wavelet representation," IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no. 4, pp. 450–461, April 2004.

[8] Y. Chiou, C. Yeh, L. Kang, C. Lin, and S. Fan-Jiang, "Efficient image/video deblocking via sparse representation," 2012 Visual Communications and Image Processing, pp. 1-6, November 2012.

[9] H. Chang, M. K. Ng, and T. Zeng, "Reducing Artifacts in JPEG Decompression Via a Learned Dictionary," IEEE Transactions on Signal Processing, vol. 62, no. 3, pp. 718-728, February 2014.

[10] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: residual learning of deep CNN for image denoising," in IEEE Transactions on Image Processing, vol. 26, no. 7, pp. 3142–3155, July 2017.

[11] X. Mao, C. Shen, and Y.-B. Yang, "Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections," Advances in Neural Information Processing Systems 29, Curran Associates, Inc., pp. 2802–2810, 2016.

[12] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, "Non-Local Recurrent Network for Image Restoration," Advances in Neural Information Processing Systems 31, Curran Associates, Inc. pp. 1673–1682, June 2018.

[13] J. Kim, J. K. Lee and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, pp. 1646-1654, 2016.

[14] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, pp. 1132-1140, 2017.

[15] Y. Tai, J. Yang, X. Liu, and C. Xu, "MemNet: A Persistent Memory Network for Image Restoration," 2017 IEEE International Conference on Computer Vision (ICCV), pp. 4549–4557, 2017.

[16] P. Bhattacharya, J. Riechen, and U. Zölzer, "Infrared Image Enhancement in Maritime Environment with Convolutional Neural Networks," Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP), SciTePress, vol. 4, pp. 37-46, 2018.

[17] Y. W. K. Zoetgnande, J. Dillenseger, and J. Alirezaie, "Edge focused super-resolution of thermal images," 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8, July 2019.

[18] C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression Artifacts Reduction by a Deep Convolutional Network," 2015 IEEE International Conference on Computer Vision (ICCV), pp. 576-584, December 2015.

[19] L. Cavigelli, P. Hager, and L. Benini, "CAS-CNN: A deep convolutional neural network for image compression artifact suppression," 2017 International Joint Conference on Neural Networks (IJCNN), pp. 752-759 , May 2017.

[20] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level Wavelet-CNN for Image Restoration," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 886-895, June 2018.

[21] C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9, June 2015.

[22] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Proceedings of the 32nd International Conference on International Conference on Machine Learning, Lille, France, vol. 37, pp. 448-456, 2015.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1026-1034, December 2015.

[24] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," CoRR, 2014.

[25] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," Proceedings of the 30th International Conference on Machine Learning, Atlanta, USA, vol. 28, pp. 1310–1318, 2013.

[26] H. R. Sheikh, Z. Wang, L. Cormack, and A. C. Bovik, "LIVE image quality assessment database release 2," , http://live.ece.utexas.edu/research/quality, 2005.

[27] Y. Chen and T. Pock, "Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1256–1272, June 2017.

[28] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in Proc. 8th International Conference of Computer Vision, vol. 2, pp. 416–423, July 2001.

[29] A. Vedaldi and K. Lenc, "MatConvNet –convolutional neural networks for MATLAB," Proceeding of the ACM Int. Conf. on Multimedia, 2015.

[30] C. Yim and A. C. Bovik, "Quality Assessment of Deblocked Images," IEEE Transactions on Image Processing, vol. 20, no. 1, pp. 88–98, January 2011.