Nat. Volatiles & Essent. Oils, 2021; 8(4): 11088-11097

NVEO
Natural Volatiles &
Essential Oils

Image Compression Using Neural Networks

S. YAGNASREE[1] , A. SUBRAMANYAM[2] , M. ANAND[3]

[1,2,3]Assistant Professor, GCET, Hyderabad, India

Abstract: A large fraction of Internet traffic is now driven by requests from mobile devices with relatively small screens and often stringent bandwidth requirements. Due to these factors, it has become the norm for modern graphics-heavy websites to transmit low-resolution, low-byte count image previews (thumbnails) as part of the initial page load process to improve apparent page responsiveness. Increasing thumbnail compression beyond the capabilities of existing codecs is therefore a current research focus, as any byte savings will significantly enhance the experience of mobile device users. Toward this end, a general framework is proposed for image compression and a novel architecture based on convolutional and de-convolutional LSTM neural networks. This paper presents a set of full-resolution lossy image compression methods based on neural networks. Each of the architectures described can provide variable compression rates during deployment without requiring retraining of the network: each network need only be trained once. Proposed work is compared to previous work, showing improvements of 4.3%–8.8% AUC (area under the rate-distortion curve), depending on the perceptual metric used.

Keywords: Lossy image compression, Recurrent Neural Network-LSTM, Convolutional neural network, Video compression

I. Introduction:

Image compression is necessary so as to enable saving large amounts of images in a limited storage area. Most images captured today are for human consumption. Human vision is sensitive towards some features of an image. e.g. Low-frequency components of an image are easily noticed while high-frequency components are not. This fact is used in lossy image compression algorithms. Hence, lossy image compression algorithms focus on the removal of such features from the image. In this paper a new lossy image compression framework is proposed which could provide better image compression ratio while maintaining the quality of the images.

Many techniques such as Run-Length Encoding (RLE), Discrete Cosine Transform (DCT), etc. are used traditionally for image compression. These deterministic image compression algorithms rely mainly on image filters, discrete transformations and quantization. Because of Moore's law, handheld devices and personal computers now have much higher processing power than they had at any time in the past. Thishas allowed the development of modern image compression algorithms. Many image compression frameworks have now been proposed, based on deep neural networks. Parts and ideas from several of those frameworks are used to develop a new deep neural network architecture which is an improvement upon existing architectures in terms of efficiency and image quality metrics such as SSIM, PSNR.

II. Related work:

Jiang, et al. considered an end to end image compression network in which fully convolutional neural network based encoder is used for image compression [1]. Proposed architecture closely follows this model. Hence on several instances, this architecture have been used as our baseline model and compared our performance with this architecture. In this architecture, a smaller image is constructed by encoder. This image is a smaller replica of the original image. The decoder, in this instance calledre-constructor, is designed to generate an original image back from its smaller replica. Average PSNR and SSIM metrics obtained were 28 and 0.56 respectively after implementation. We achieved better results than this model. Another convolutional neural network based architecture proposed by Cavigelli, efficient in suppressing the artifacts which are introduced during image compression process. This architecture makes efficient use of several skip connections to train the model. Henceforth skip connections are used to train our model faster. Space separable operations such as pointwise and depth wise operations are used at some levels of our neural network to reduce the training as well as inference time required to train our model.

We have also designed a loss function such a way that image generated from compression module has a very low variance in pixel values. This limits the pixel values the resultant compresses image can have. The arithmetic encoder is used to process this compressed image. Since the compressed image nowhas a very small number of distinct values, an arithmetic encoder can use lossless data compression algorithms such as entropy coding or RLE to further compress the results obtained from compression module.

Hyper-parameter optimization was performed on this network to calculate hyper- parameters such as number or layers, learning rate, coefficients for loss function etc. We used hyper-band scheduler for this purpose. We could improve our results using values obtained hyper-parameter search. The recently developed algorithms like WebP and High-Efficiency Image File Format (HEIF) use more complex encoding structures. Even though these compression techniques require more computation power than the traditional algorithms like JPEG, but resulted in much smaller file size while maintaining a similar quality of an image.

Before the advent of Deep Neural Networks, techniques such as Run-length encoding, Entropy encoding, Differential Pulse-Code Modulation (DPCM) were used for image compression. In run-length encoding series of bits of 0s and 1s are replaced by a bit symbol followed by a count of the number of bits. Entropy encoding method on the other hand works, on a higher level of image representation. In this technique, quantized pixel values are replaced by symbols. Length of these symbols is determined on the basis of frequency of occurrence. Huffman Coding, Arithmetic Coding, and Range Coding techniques are some examples of entropy encoding techniques. Run-length encoding and Entropy encoding techniques are lossless data compression techniques. Data is lost when quantization results in the lower granularity of values. However, it is necessary to convert analog values to its corresponding digital values. Differential Pulse-Code Modulation (DPCM) is another technique used to convert an analog signal into a digital signal. In this technique, the difference between sampled

values from the analog signal and predicted values is quantized and encoded. Since pixel values are predicted based on previous values, the compression factor of DPCM is higher than quantization techniques.

An Image in WebP format is represented by 32 bit format. In this format, alpha channel is added along with 'R', 'G', 'B' values which represent the opacity value. Lossy WebP architecture uses predictive encoding technique in which, the value of a pixel is predicted using the value of neighboring pixels. Lossless WebP compression technique uses a variety of lossless transformation techniques such as color de-correlation transform, Subtract Green Transform and color cache encoding in order to provide better lossless performance than earlier techniques. HEIF is a video and single image compression format in which images are stored in the form of thumbnails in several containers and the final image is built using those representations. HEIF format supports 16-bit color as opposed to an 8-bit color used by JPEG. HEIF format supports block sizes of 8 × 8 to 16 × 16 pixels. Pixel value in each block is predicted using the data in another block. This format uses Context-Adaptive Binary Arithmetic

Recurrent Neural Networks (RNNs) are typically used for sequential time series data predictions. RNNs have been used for image compression in an architecture proposed by Toderici, et al [8]. This architecture was developed small network bandwidth for handheld devices. In this architecture, the output image is refined and improved successively as more data is obtained from the network. Even though this network performs better compared Vanilla CNN based approaches, every iteration of a network requires a minimum of eleven layers of RNN convolutional layers and hence, it can be a more complicated model to train. This model is more useful in the instances where compressed image data is received while image is being constructed.

III. Proposed Methodology:

This paper discusses about a model that is interluding with one of RNN's network that is LSTM and how the output will be reproduced depending on the functional features of each block of the network. Our compression networks are comprised of an encoding network 'E', a binarizer 'B' and a decoding network 'D', where D and E contain recurrent network components. The input images are first encoded, and thentransformed into binary codes that can be stored or transmitted to the decoder. The decoder network creates an estimate of the original input image based on the received binary code. This procedure is repeated with the residual error, the difference between the original image and the reconstruction from the decoder. All the details mentioned will be explored.

Figure.1 shows the architecture of a single iteration of our model. While the network weights are shared between iterations, the states in the recurrent components are propagated to the next iteration. Therefore residuals are encoded and decoded in different contexts in different iterations. Note that the binarizer B is stateless in our system.

We can compactly represent a single iteration of our networks as follows:

$$b_t = B\big(E_t(r_{t-1})\big), \quad \hat{x}_t = D_t(b_t) + \Upsilon x_{t-1},$$

$$r_t = x - \hat{x}_t, \quad r_0 = x, \ \hat{x}_0 = 0$$

A. Block diagram:

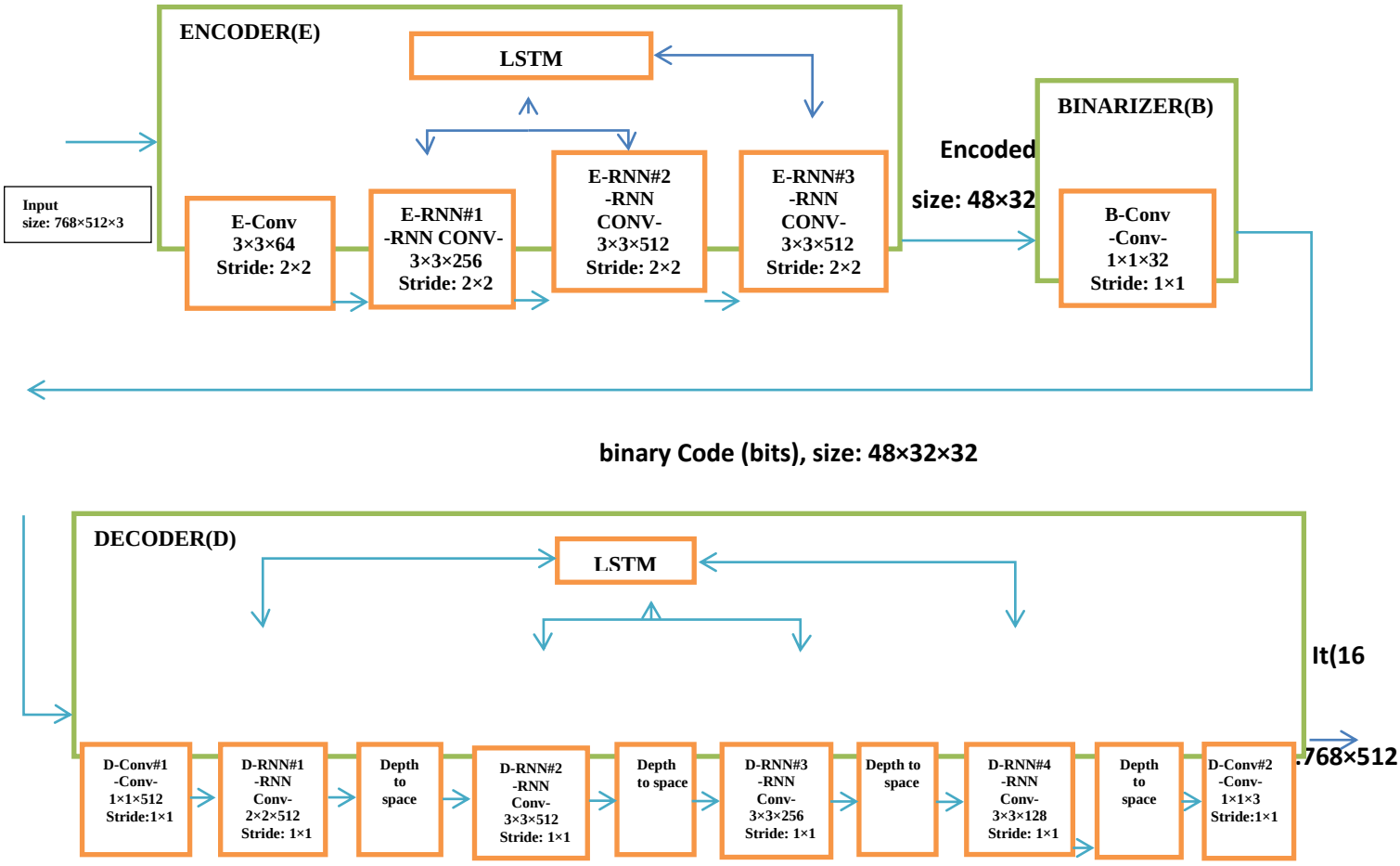

Figure.1 Block diagram

Where, D_t and E_t represent the decoder and encoder with their states at iteration t respectively, b_t is the progressive binary representation; $x\hat{}_t$ is the progressive reconstruction of the original image x with γ = 0 for "one-shot" reconstruction or 1 for additive reconstruction and r_t is the residual between x and the reconstruction $x\hat{}_t$. In every iteration, B will produce a binarized bit stream $b_t\{1, 1\}m$; where m is the number of bits produced after every iteration.

After k iterations, the network produces m, k bits in total. Since our models are fully convolutional, m is a linear function of input size. For image patches of 768x512, m = 128. The recurrent units used to create the encoder and decoder, include two convolutional kernels:

i. The input vector which comes into the unit from the previous layer.

ii. The state vector which provides the recurrent nature of the unit.

Convolution on the state vector and its kernel is referred to as the “hidden convolution” and the “hidden kernel”. All convolutional kernels allow full mixing across depth. For example, the unit D-RNN#3 has 256 convolutional kernels that operate on the input vector, each with 3x3 spatial extent and full input-depth extent (128 in this example, since the depth of D-RNN#2 is reduced by a factor of four as it goes the “Depth-to-Space” unit). The spatial extents of the hidden kernels are all 1x1, except for in units D-RNN#3 and D-RNN#4 where the hidden kernels are 3x3. The larger hidden kernels consistently resulted in improved compression curves compared to the 1×1 hidden kernels.

During training, a L1 loss is calculated on the weighted residuals generated at each iteration, so our total loss for the network is:

$$\beta \sum_{t} |r_t|$$

Combination of recurrent unit variants and reconstruction frameworks for our compression systems are explored and compared these compression results to the results from the de-convolutional network.

B. Image Quality Metrics:

Image quality metrics are useful for us to measure how well is an architecture has performed. These can also be used to define loss in neural networks. Here, we will review some reference image quality metrics.

Mean Square Error (MSE) : MSE calculates the addition of squared differences between pixel values of two images. Here, M and N is a size of image1 and image2 respectively. I(x,y) is a pixel value at position x,y. This is the simplest image quality metric to understand. However, This metric is not always a good metric to access image compression quality since it does not take into account the range of variations in pixel values in an image and high, low-frequency components in an image. These factors, however, affect human perception towards quality.

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}[I(x,y) - I'(x,y)]^2$$

Peak Signal to Noise Ratio (PSNR): PSNR is a measure of peak error between two images. This method is used to calculate the quality of compression method where higher PSNR value represents better quality of compression of an image.

$$PSNR = 10\log_{10}\frac{R^2}{MSE}$$

Where R represents maximum fluctuation in input image pixel values

Structural Similarity Index (SSIM): SSIM calculates quality degradation in an image due to image processing tasks such as compression. SSIM is considered a better metric to access degradation of images because it takes into account visible structures of an image. SSIM is calculated using a combination of variance and covariance terms

between two images.

Dataset augmentation: Data augmentation is used introduced to regularization in deep neural networks. Augmentation in images increases the effective dataset size and hence helps the neural network learn important features about images. We applied data augmentation techniques such as random cropping, random scaling, random flips and rotation of images and random changes in colors of an image.

C. Network Design

Our proposed architecture consists of four parts - Image Compression Module (ICM), Arithmetic Encoder, Arithmetic Decoder and Image Reconstruction Module (IRM).

Arithmetic Encoder and Decoder

Arithmetic encoder and decoder are made up of python implementation of Huffman coding. In Huffman coding, 0-255 values of pixels are treated as separate symbols, the frequency table is calculated, and this symbol is replaced by a sequence of bits. The number of bits used to represent a given symbol is inversely proportional to the frequency of that symbol. Huffman coding was implemented using reference arithmetic coding library in python. Since original data can be completely retrieved in the decoder part, it is a lossless compression algorithm and its introduction or removal does not affect other neural network-based modules. For the purpose of neural network training, we did not use an arithmetic coder or decoder so as to avoid unnecessary computational overhead.

Image compression module (ICM)

The image compression module consists of a series of CNN layers. These CNN layers are used to learn features of the images which can be useful for further image reconstruction tasks. Series of convolutional neural networks identify latent features in images and helps to develop series of feature maps which could hold information useful for identifying critical components in an image. These components, includes overall structure of an image as well as some salient features such as edges and corners which cannot be regenerated by reconstruction layer unless they are provided as an input. Thus, this module acts as a filter through which only few critical components are passed to an intermediate image. We have performed hyper-parameter optimization on this component where we tried three, five and ten layers. However, performance of overall network did not improve with more layers in this module. Hence, we have used a three layer CNN module as shown in Figure.2

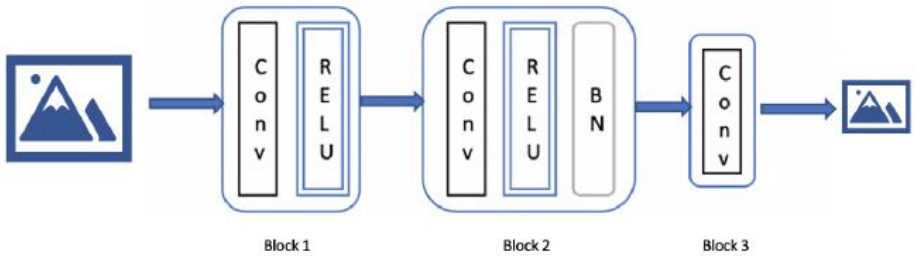

Figure. 2 Image Compression Module

Image reconstruction module (IRM)

Reconstruction module is tasked at regenerating an image such that it is very similar to the original image. The reconstruction module has two responsibilities - Resize an image to the original size and improve the quality of the resized image. Since this module is tasked with regeneration of an image from a minimal information passed on by compression module, this module requires more layers to hold information on how to reconstruct an image. This information is held in feature maps of convolutional layers. The first few layers are responsible in reconstruction of basic shapes such as line, points, corners etc. while further layers adds more information about the image such as facial expressions. The size of kernel used in this network is maintained at 3 × 3 since all these features are local to a region and are less likely to have any impact on other parts of an image.

To resize the compressed image to its original size SRGAN is used. This network returns the image of size four times bigger than the actual image. This image can be scaled down to the desired size using the interpolation technique before it is fed to the reconstruction module. SRGAN is known to generate an image with much better quality as compared to simple interpolation techniques. While training this network, interpolation technique is for upscaling instead of GAN due to the overhead GAN might have caused during the training phase. Moreover, this allowed us to use bad quality images to train the network for image enhancement. Using a bicubic interpolation instead of SRGAN alleviated some need for data augmentation to training reconstruction network.

SRGAN was trained separately in its original proposed shape on the ImageNet dataset. The IRM consists of five blocks each containing one convolutional layer and an optional batch normalization or RELU layer.

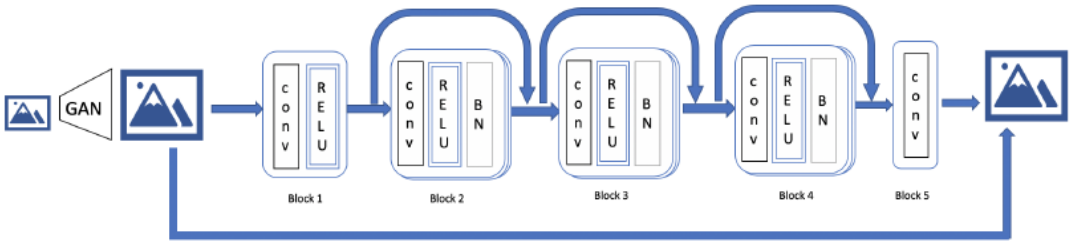

Figure. 3 Image Reconstruction Module

IV. Results

Training Output:

Our neural network is trained on STL10, CIFAR10, COCO and CLIC image datasets for varying number of epochs and network sizes, got the best results out of 50 epochs of COCO dataset. It required more than 110 hours of training on Google Cloud instance with Nvidia Tesla P 100 GPU. Image size of 200x200 is used for this training. SRGAN and arithmetic encoder and decoders were not added to the network during training time.

Figure.4 Training start

```
Start training... step per epoch: 24
Epoch number: 1
2021-05-18 13:18:45.217587: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)
2021-05-18 13:18:55.160411: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudnn.so.8
2021-05-18 13:18:56.112004: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcublas.so.11
2021-05-18 13:18:56.343566: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcublasLt.so.11
Epoch number: 1 step 10 loss 3.0840246468782424
Epoch number: 1 step 20 loss 2.5051344275474547
Epoch number: 1 ms-ssim 0.5524077933582733
Epoch number: 2
Epoch number: 2 step 10 loss 3.870975947380066
Epoch number: 2 step 20 loss 3.49526823759079
Epoch number: 2 ms-ssim 0.24216148985792388
Epoch number: 3
Epoch number: 3 step 10 loss 2.6993813157081603
Epoch number: 3 step 20 loss 2.6924786806106566
Epoch number: 3 ms-ssim 0.3704710859755013
Epoch number: 4
Epoch number: 4 step 10 loss 2.499044680595398
Epoch number: 4 step 20 loss 3.172033655643463
Epoch number: 4 ms-ssim 0.43302197429060774
Epoch number: 5
Epoch number: 5 step 10 loss 3.602510231733322
Epoch number: 5 step 20 loss 2.7561867117881773
Epoch number: 5 ms-ssim 0.43926087039057843
```

```
Epoch number: 94
Epoch number: 94 step 10 loss 1.8065174490213394
Epoch number: 94 step 20 loss 1.635795819759369
Epoch number: 94 ms-ssim 0.45629867752051934
Epoch number: 95
Epoch number: 95 step 10 loss 1.9116175949573517
Epoch number: 95 step 20 loss 2.078054940700531
Epoch number: 95 ms-ssim 0.25949855617468837
Epoch number: 96
Epoch number: 96 step 10 loss 1.8082870244979858
Epoch number: 96 step 20 loss 1.695320588350296
Epoch number: 96 ms-ssim 0.5344237967921661
Epoch number: 97
Epoch number: 97 step 10 loss 1.9166377544403077
Epoch number: 97 step 20 loss 1.8278205037117004
Epoch number: 97 ms-ssim 0.7317611475288933
Epoch number: 98
Epoch number: 98 step 10 loss 1.7333704769611358
Epoch number: 98 step 20 loss 1.7619138062000275
Epoch number: 98 ms-ssim 0.7242398039289074
Epoch number: 99
Epoch number: 99 step 10 loss 1.8579369068145752
Epoch number: 99 step 20 loss 1.7530948758125304
Epoch number: 99 ms-ssim 0.7021067134075223
Epoch number: 100
Epoch number: 100 step 10 loss 1.626701247692108
Epoch number: 100 step 20 loss 1.1877119600772859
Epoch number: 100 ms-ssim 0.6799705897351449
Done
```

```
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Epoch number
Done
```

Figure.5

Training end

V. Conclusion

The proposed framework is a mix of SRGAN and end-to-end image compression network. In this architecture, the bicubic interpolation layer is replaced with SRGAN and have introduced some skip connections as proposed in image compression artifact suppression network to improve the performance. The SSIM and PSNR metrics obtained with our new framework in some cases beats recently proposed deep neural networks. The time required for the execution of deep learning framework is largely proportional to the number of neural network layers. Our networks only need to be trained once (not per-image), regardless of input image dimensions and the desired compression

rate. On a large-scale benchmark of 32x32 thumbnails, our LSTM based approaches provide better visual quality than (header less) JPEG, JPEG2000 and WebP, with a storage size that is reduced by 10% or more.

References:

[1] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao, ‘‘An end-to-end compression framework based on convolutional neural networks,’’ IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 10, pp. 3007--3018, 2018.

[2] Adnan khashman, Kamil dimililer, “Image Compression using Neural Networks and Haar Wavelet”, WSEAS TRANSACTIONS on SIGNAL PROCESSING, Issue 5, Volume 4, May 2008.

[3] Antem Gorodilov, Dmitriy Gavrilov, Dmitriy Schelkunov, “Neural networks for image and video compression”, 2018 International Conference on Artificial Intelligence Applications and Innovations (IC-AIAI),

[4] White paper Cisco public, "Cisco Visual Networking Index: Forecast and Methodology, 2016–2021," 15 09 2017

[5] N. Francisco, N. Rodrigues, E. Silva и S. Fariaab, “A generic post- deblocking filter for block based image compression,” Signal Processing: Image Communication, т. 27, № 9, p. 985–997, 2012.

[6] C.-H. Yeh, L.-W. Kang, Y.-W. Chiou, C.-W. Lin и S.-J. J. Fan , “Self- learning-based post-processing for image/video deblocking via sparse representation,” Journal of Visual Communication and Image Representation, т. 25, № 5, pp. 891-903, 2014.

[7] K. Takahashi , T. Naemura и M. Tanaka, “Rate-distortion analysis of super-resolution image/video decoding,” Proc. IEEE Int. Conf. Image Process., pp. 1629-1632, 2011.

[8] G. Toderici, S. M. O'Malley, S. H. Jin , D. Vincent, D. Minnen, S. Baluja, M. Covell и R. Sukthankar, “Variable rate image compression with recurrent neural networks,” arXiv preprint arXiv:1511.06085, 2015.

[9] G. Toderici, D. Vincent, N. Johnston, J. . H. Sung , D. Minnen, J. Shor и M. Covell, “Full resolution image compression with recurrent neural networks,” arXiv preprint arXiv:1608.05148, 2016..

[10] K. He, X. Zhang, S. Ren и J. Sun, “Deep residual learning for image recognition,” arXiv preprint arXiv:1512.03385, 2015.

[11] J. Duchi, E. Hazan и Y. Singer , “Adaptive subgradient methods for online learning and stochastic optimization,” Journal of Machine Learning Research, т. 12, № Jul, p. 2121–2159, 2011.

[12] J. Kim, J. L. Kwon и K. L. Mu , “Accurate image super-resolution using very deep convolutional networks,” arXiv preprint arXiv:1511.04587, 2015.

[13] J. Feng, T. Wen, L. Shaohui , R. Jie , G. Xun и Z. Debin , “An End-to- End Compression Framework Based on Convolutional Neural Networks,” Submitted to IEEE transactions on circuits and systems for video technology. arXiv:1708.00838v1, 2017.

[14] Y. Li, D. Liu, H. Li, L. Li, F. Wu и H. Zhang, “Convolutional Neural Network-Based Block Up-Sampling for Intra Frame Coding,” IEEE Transactions on Circuits and Systems for Video Technology, т. 28, № 9, pp. 2316-2330,

2018.

[15] “Reference software for ITU-T H.264 advanced video coding,” Telecomutication standartization setor of ITU, 2016.