2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT) | 978-1-7281-5791-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICAECT49130.2021.9392554

Image Denoising using Deep Auto-Encoder Network for Production Monitoring in Real-Time

[1]Sujay Jadhav & [2]Pooja Kulkarni

[1]Technische Universität Chemnitz, Chemnitz, Germany
[2]Vishwakarma Institute of Technology (SPPU), Pune, India
E-mail : [1]sujayjadhav.8070@gmail.com & [2]pooja.kulkarni@vit.edu

***Abstract*- Industrial operations today are being smoothed by monitoring the entire process using latest Image Processing (IP) techniques, especially the bottlenecks in the process. The challenges that need to be addressed in order to commercialize and optimize these solutions are equally growing, particularly in image monitoring of the machining and assembly lines in production plants as well as mechanical workshops. One such hindrance in monitoring the jobs on a Lathe machine or a Computer Numerical Control (CNC) machine is the motion blur noise introduced in the images due to the real-time on field vibrations within the setup. In this paper, this problem is addressed with Image denoising in real time frames. Many practical solutions have been addressing the problem of Image denoising or Image reconstruction using different Computer Vision (CV) techniques in the last two decades. Herein, we propose Image restoration of such blurred image using a specific architecture of Deep Autoencoder network with Convolutional Neural layers so as to obtain a fully constructed image without any prior knowledge of the corresponding clean image. For this we have used an averagely deep encoder-decoder neural network to minimize the typical motion blurring noise that gets introduced in the input image captured by the camera setup on the production lines. This particular technique eliminates the need to identify the type of noise, its intensity or any other statistical characteristics. The results evidently show that the reconstructed images are visually convincing, with large improvements retaining the blur image details**

Index Terms—deep learning, image processing, neural networks, auto-encoder, convolution

I. INTRODUCTION

With traditional non-blind image restoration methods of image processing where machine learning is not used, like 'Wiener' filtering, 'Lucy- Richardson' algorithm, 'Deconvolution with Regularized Filter' (DRF) etc., although the results are superior for the selective noise patterns and are relatively obtained with less efforts, the statistics of the noise need to be known in advance. This is not the case in Deep Learning (DL) methods, as a wide range of noises, in real time and of random nature can be processed out and mitigated. Moreover, Wiener filtering is a linear estimation of the original image, which minimizes the overall mean square error in the process of inverse filtering and noise smoothing. It isn't set up to deal with mixture distributions (such as Latent Dirichlet Allocation, K-means, and Non-negative Matrix Factorization can attack) as well as for discrete state systems like Markov chains.

Even with the 'Iterative Blind Deconvolution' (IBD), the main drawback is that the convergence of the iterative process is not guaranteed. But the original image can have effect on the final result. Thus, with these and similar techniques, signal can be processed and restored to a great extent with prior knowledge of both noise and signal, however learning and prediction are the key factors which provide a vital insight in real time image reconstruction. Though costlier to develop and train, DL methods have recently proven to be outperforming most of the conventional methods of signal and image processing.

For the better part of this decade, deep convolutional neural networks have notably outperformed the other traditional methods, to be the state of the art in most visual recognition tasks, [1,4]. Nevertheless, the performance of convolutional networks was limited due to the size of the available training sets and the size of the modeled networks. The breakthrough by Krizhevsky et al. [3] was due to supervised training of a large network with 8 layers and millions of parameters on the 'ImageNet' data set with 1 million training images. Since then, even larger and deeper networks have been trained [5]. Signal reconstruction from corrupted or incomplete measurements is rather difficult and sometimes more critical a job, especially when a clean or statistically informative input data is missing. At any rate, the penalty for false or inaccurate restoration can be high and burdensome in applications like automotive software as well as the specific mass production assembly works. Here, we focus on the critical problem of real time blur noise in image monitoring, introduced due to the unavoidable mechanical vibrations of lathe/CNC machine setup in mechanical workshops as well as large scale production plants. These noises make it arduous to verify the images of the setup and take a toll on job/product quality check with a subsequent need for manual efforts. Be that as it may, human intervention cannot be of a consistent accuracy and speed and by the time the defects on the job have been identified, it is too late. The major hurdle, in continuously monitoring the machinery setup and the jobs is that of the undefined behavior of vibrations due to machine movements like drilling, milling, broaching etc. and the consequent noise introduced in the

real time images.

Our approach to resolve this issue involves the current state of- the-art solution using deep learning method. Considering the randomness of the real time noises introduced in such applications, a pre-trained Deep Neural Network (DNN) model can subsequently reduce the chances of missing out vital image details and could easily decipher a cleaner image with the real time image retained. An advantage of DNN methods is that these methods are purely data driven and no hard assumptions about the noise distributions are made. Furthermore, in the fully convolutional networks, the noise is eliminated step by step, i.e., the noise level is reduced after each layer. During this process, the details of the image content may be lost. Nevertheless, in a Convolution-Deconvolution network, the convolution preserves the primary image content, while deconvolution, although in a limited manner, proves to be restoring the lost details [6]. In this case, the details of input image which are vanished due to vibrations while taking the image or due to the motion-blur introduced when the object (job/tool fixed on the lathe machine) "shivers", can easily be predicted and retained by the model based on the exposure to a database of images with similar sets of noise patterns. The database on which the model has been trained, comprises of vertical as well as horizontal blur shift introduced using a small python code snippet, in order to create a fourfold trainable data set of blurry images of the lathe machine, CNC set-up, machining tools, jobs and visually closely resembling images.

II. Dataset Formation

A. Data Augmentation

Data augmentation enables a significant increase in the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks. It proves essential for teaching the network the desired variance and robustness properties, when only few original training samples are available (less than a thousand in our case). Flipping the original data set to get the vertical as well as horizontal counterparts not only would double the data but also minimizes the error facilitating the learning. Furthermore, for this model we are specifically training on the images with motion blur noise i.e. random vertical and horizontal blur effects as introduced by any Simple Harmonic Motion (SHM) or simply the vibrations at the time when images of the setup are being taken. To replicate these real-times images on a large scale (approx. 10000 images), we have used a filter of '1's and '0's, with a kernel window of size 10x10 pixels to traverse it in a vertical or horizontal manner. Per-pixel displacements are then computed using this kernel for all 128x128 pixels of the image to obtain a vertical blur or horizontal blur respectively. The same procedure is carried out with a 15x15 grid kernel, thereby incrementing the total data set count by fourfold.

B. Additive Motion Blur Noise

The input samples for the model are motion blurred (vertically as well as horizontally) with a linear filter of kernel window size 'K' with all zeros (0) but center ((K-1)/2'th row for horizontal motion blur; (K-1)/2'th column for vertical motion blur) pixel values as '1/K'. Adding this 2D filter to our input images, results in sample noisy images as shown below.

The higher the value of K (filter kernel size) the more motionblur is introduced.

Fig. 1: Blurred Input samples (filter kernel size K=10) for training the model.

Fig. 2: Blurred Input samples (filter kernel size K=15) for training the model

III. Architecture And Methodology

Inspired from the U-Net architecture [7], this architecture uses the consecutive convolutional blocks in the first half and equivalent complementary de-convolutional blocks in the later half. The consequent max pooling layer after every two convolutional blocks contributes in extracting the extreme features of the input image thereby reducing the size and computational cost of progressive encoding. Average pooling method smooths out the image and hence the sharp features may not be identified when this pooling method is used. Max pooling preserves the brighter pixels from the image and hence is found to keep any abnormalities (e.g. cracks or dents on jobs in our case) intact. Also, it leads to a faster convergence rate by selecting superior invariant features. Each of the 3x3 convolutions (un-padded) is followed by a Batch Normalization layer (momentum=0.99) and then by a Rectified Linear Unit (ReLU) activation operation. Since, the pixel value of our interest is to be encoded as 1 and 0 otherwise, ReLU always plays a better role than a Sigmoid activation function. Furthermore, Batch Normalization (BN) standardizes the inputs of a layer for

each mini-batch, which stabilizes the learning process through normalization of the input layer by re-centering and re-scaling. In this model, we have used BN layers after every alternate convolution as well as deconvolution layer (BN layer could have been used after each convolution/deconvolution layer, but this also increases computational time). These alternate normalizations of the convoluted matrices considerably reduce the number of training epochs for our model (from >40 epochs to as less as 25 epochs), required to minimize the training loss to an equivalent value. The kernel size for convolution and deconvolution is set to 3 × 3, which has shown decent statistical performance. The input and output of the network are images of the same size w × h × c, where w, h and c are width, height and number of channels. For all the images, we have used c = 1. Although, it is straightforward to apply to images with a greater number of channels (c=3). The size of input image can be arbitrary since our network is essentially a pixel-wise prediction. With every consequent 2D-convolutional layer, a specific encoding of image is performed and which is where the concatenation of memorized values from initial convolutional layers with the later to-be-deconvoluted layers proves to be a boost in retaining important image details. Moreover, deeper networks often suffer from gradients vanishing and become hard to train.

IV. Parameter Training

The output image function is computed by a pixel-wise ReLU over the final feature map combined with the Mean Squared Error (MSE) loss function. The ReLU function is defined as:

$$f(x) = x^{+} = \max(0, x) \tag{1}$$

The MSE loss function is then responsible to penalize at each position the deviation of f(x) from 1, using the Mean Squared Error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2}$$

Both Stochastic Gradient Descent (SGD) as well as Adam optimizer can be used for calculating the weights throughout the traverse for all epochs. Although, SGD along with a certain momentum is believed to dodge local minima better than Adam, in our case both these optimizers showed similar declining pattern for loss. But adaptive learning optimizer is found to converge considerably faster while training this particular model. The initial learning rate is set to be 0.0001 and is decayed down with an exponential rate by using weighting parameters beta1= 0.9 and beta2= 0.999 per step. Regularization is one of the methods to balance the Bias-Variance trade-off of the deep learning model. In all the convolutional layers throughout, we have used 'L2' (with Ridge regression weight decay) regularizer to avoid the problem of overfitting due to high variance, by penalizing the stubborn weights when they try to 'over-learn' the complexity of model, than what is actually needed.

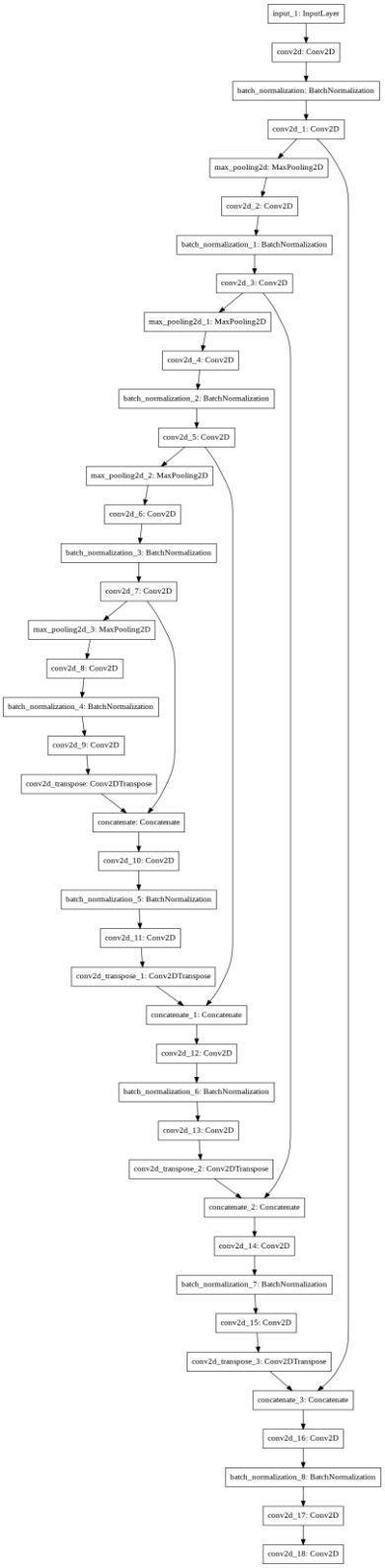

Fig. 3: Flowchart representation of the 4 pair Encoder-Decoder Deep Learning model. Both the first input layer and the last reconstructed output layer being an image array of the size (128x128x1).

$$loss = l2 * reduce\_sum((x^2)) \tag{3}$$

The L2 regularization penalty is computed in order to prefer solutions with smaller norms characterized by an unknownvector 'w' such that f(x)=w. x, keeping in consideration all the weights.

$$loss = \min_{w} \sum_{i=1}^{n} V(\hat{X}_i \cdot w, \hat{Y}_i) + \lambda \|w\|_2^2 \tag{4}$$

This is also called Tikhonov regularization, one of the most common forms of regularization. Therefore, the loss function in (4).

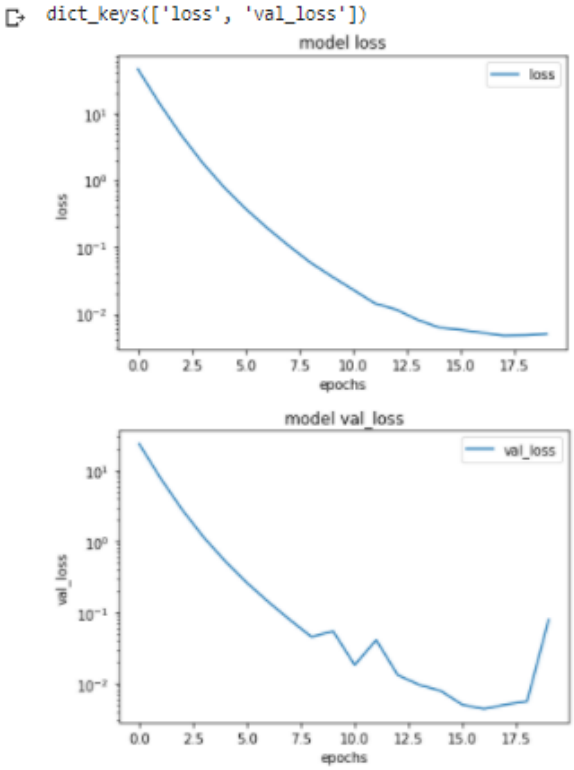

Fig. 4: Training and validation loss without any regularization.

The validation loss is more stable in the later part, and thereby minimizing the variance/overfitting of model. Convergence to a local minima is dodged.

Considering the de facto in such literature, we use all grayscale images for denoising. The training data set comprises of approximately 400 images taken on field by smartphones as well as 400 google images resembling the machining, tooling and all similar sample images of CNC and lathe machine jobs.

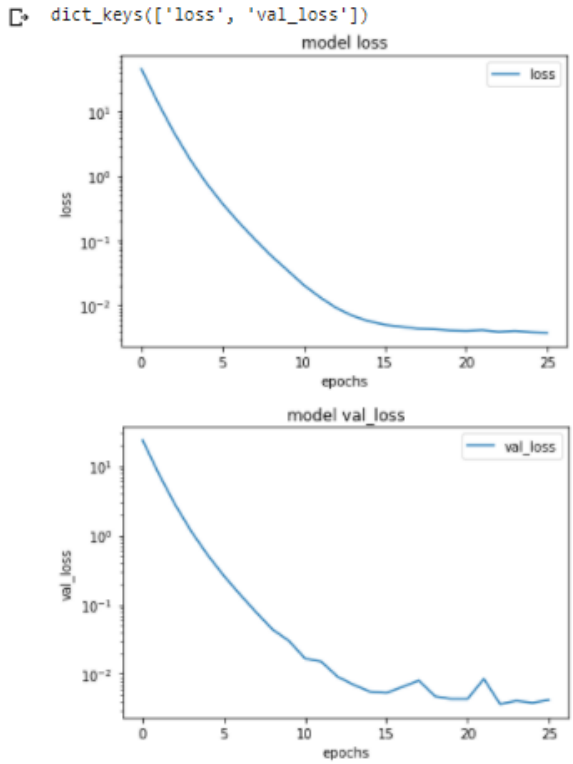

Fig. 5: Training and Validation loss with L2 regularization.

These original images are augmented to form an eight-fold larger batch of images for training. The entire data set is split with 0.1 test size using test-train-split (Scikit-learn library) functionality a validation set of 0.2 fraction.

V. Experiments And Evaluation

In this development experiment, due to monotonous and limited amount of data with average computing power we have computed all the input Images with 128x128 frame size and in gray scale (i.e. channel count=1). The average signal to noiseratio (PSNR) for 20 vertically motion blurred test samples with kernel-size K=10 is 29.532 dB(Fig.6 and 7), and average PSNR for kernel K=15 is 24.602 dB(Fig.8 and 9).

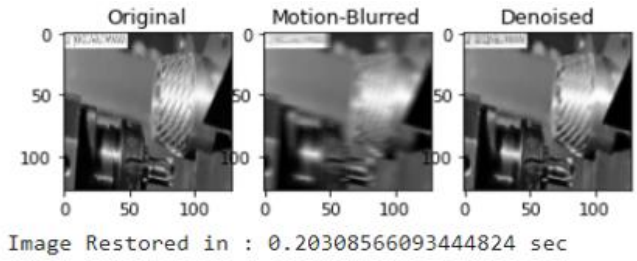

Fig. 6: [Test Image] : (K=10), source [Internet].

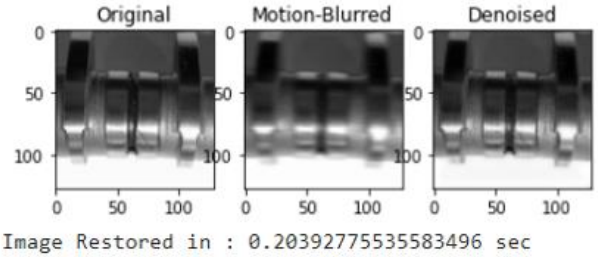

Fig. 7: [Test Image] : (K=10), source [Internet].

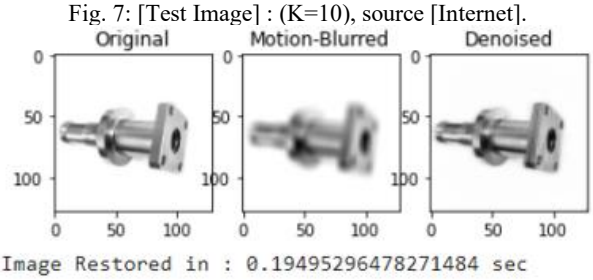

Fig. 8: [Test Image] : (K=15), source [Internet].

The application perspective of this model is enhanced by one of the major advantages of deploying the pre-trained model on hundreds of variant images of the same setup, with saved parameters and predicting the restoring image in realtime (with a speed >2 frames per second). The proposed approach is implemented using Tensorflow [8]. This particular model takes approximately 7000 seconds (roughly 2 hours) for training a small data set of 6192(8x774 original) input images. All the training and testing are performed on 'Nvidia K80' GPU from the google Colaboratory and averagely takes

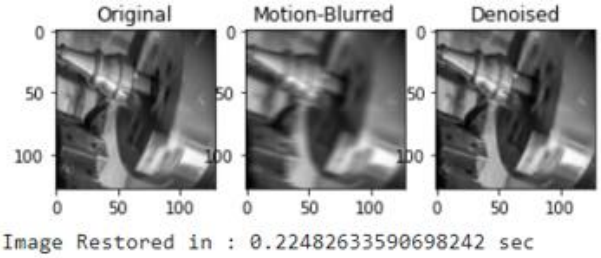

Fig. 9: [Test Image] : (K=15), source [Internet].

250 milliseconds to predict one denoised image using learned knowledge. This allows us to denoise the captured image in real-time, using the pre-saved model weights from training all the possible relevant images of the CNC/lathe machine setup itself. We have used OpenCV (Fig.10 and 11) to run and monitor the setup with a web-cam at a speed of approximately 30 fps (frames per second). The average frequency of the trail function to denoise the samples from these set of images is 3 images per second, which is considerably better than a human inspection with comparatively same efficiency and better consistency. Moreover, the future advancements would enable this model to run on lesser RAM and averagely fast CPUs, in dedicated a embedded device, without a need for external server.

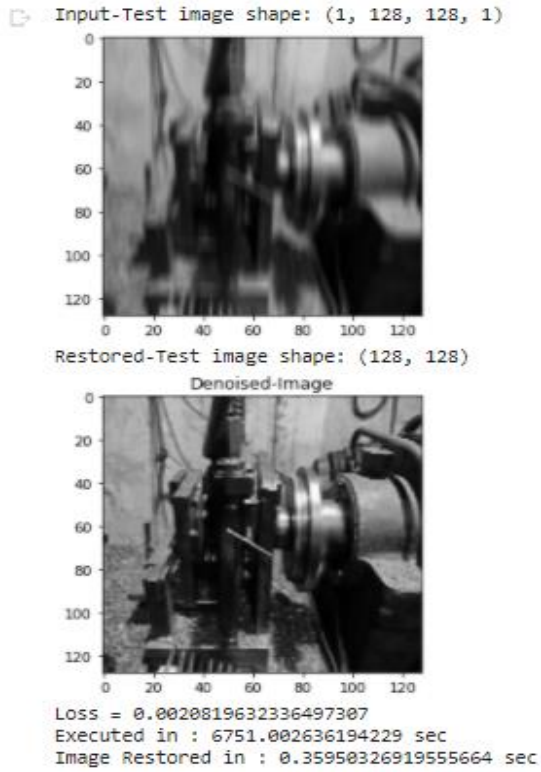

Fig. 10: [Test Image] : source - Real time workshop image by laptop (Dell Inspiron 3542, 0.92MP webcam; Cropped image size:128x128x1).

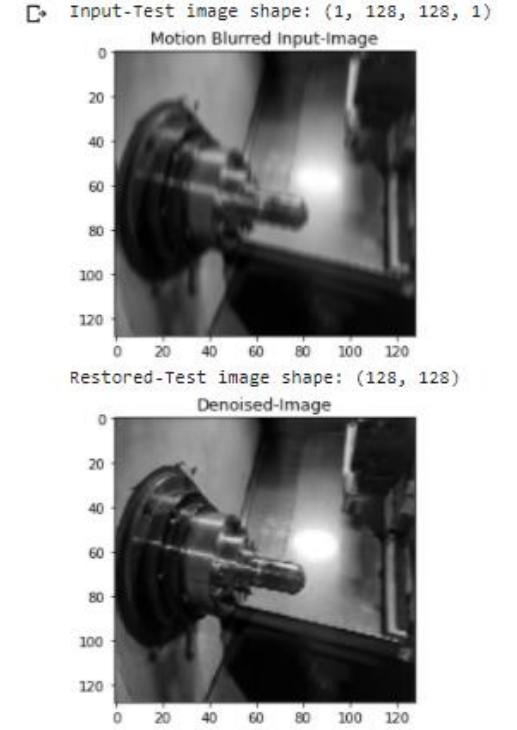

Fig. 11: [Test Image] : source - Real time workshop image by laptop (Dell Inspiron 3542, 0.92MP webcam; Cropped image size:128x128x1).

Undoubtedly, with more eclectic but relevant data set and better computing performance (or optionally using Discrete Cosine Transform (DCT) transform to condense the computational complexity as mentioned in 'future-scope' discussions), the predicted denoised image would be of even better restoration capability as well as higher resolution.

VI. Conclusion

The results evidently show that the denoised images using model-predicted pixel values with MSE loss minimization, are visually convincing, with large improvements locating the lost or blur image details. Thus, the vagueness of original real-time blurred image is not eliminated entirely, but is surely reduced in order to obtain few more image specifics. While this model has been proposed through numerous trial and error processes, there should be an improved way of tuning the model structure and hyper parameters so as to obtain a higher Signal to Noise Ratio(SNR) for future images. Deriving a method to design suitable model complexity for each problem is needed. The future scope of this article extends towards achieving greater accuracy with research focused on minimum loss of image details while abstraction in convoluting the image arrays and retaining these details during deconvolutional layers, towards the other half of the model.

VII. Future Scope

With possible applications in automotive domain, where indefinite motion blur and vibration noise is encountered, this particular model can find its use on platforms with lesser memory and reasonable GPU specifications (like smartphones and 'GoPro' action-cameras), unlike most deep neural networks which require very high-performance GPUs to train on. Furthermore, with optimization of prediction code functionality the image restoration in real time can be accelerated. Probably up to even >10 denoised frames per second, in a real time setup.

While the practical maneuvers can be advanced using better hardware as well as software versions (using well accepted pre-defined packages for data augmentation), the theoreticalapproach can also be modified and optimized with the use of DCT for the input image data set in our model. Most of the signal information is concentrated in a few low-frequency components of the DCT which aids in reducing the spatial and spectral redundancy between the neighboring pixels. Also, the high frequency component in a DCT transform of our sample input image (motion blurred due to vibration), will be the pixel values that are spontaneously changed due to the noise. The main challenge in introducing this step would be that of retaining the original pixels while performing the Inverse-Discrete Cosine Transform (IDCT), in the second half of our model. This being the major area of work for thefurther improvement of this article, it would prove to bereducing computing cost as well as training time for this very model bolstering the entire model to be more suitable for fieldapplication in real-time.

References

[1] H. Noh, S. Hong, and B. Han. "Learning deconvolution network for semantic segmentation," in Proc. IEEE Int. Conf. Comp. Vis., pages 1520–1528, 2015.

[2] S. Hong, H. Noh, and B. Han. "Decoupled deep neural network for semi-supervised semantic segmentation," In Proc. Advances in Neural Inf. Process. Syst., 2015.

[3] Krizhevsky, A., Sutskever, I., Hinton, G.E.: "Imagenet classification with deep convolutional neural networks," In: NIPS. pp. 1106–1114 (2012).

[4] Girshick, R., Donahue, J., Darrell, T., Malik, J.: "Rich feature hierarchies for accurate object detection and semantic segmentation," In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014).

[5] Simonyan, K., Zisserman, A.: "Very deep convolutional networks for large-scale image recognition" (2014), arXiv:1409.1556 [cs.CV]

[6] Mao, Xiao-Jiao, Shen, Chunhua, and Yang, Yu-Bin. "Image restoration using convolutional auto-encoders with symmetric skip connections," In Proc. NIPS, 2016.

[7] Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. "U-net: Convolutional networks for biomedical image segmentation," MICCAI, 9351:234–241, 2015.

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scalemachine learning," in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, pp. 265–283.

[9] Shuang Zhang , Member, IEEE, Ada Zhen, Member, IEEE, and Robert L. Stevenson, Member, IEEE, "Deep Motion Blur Removal Using Noisy/Blurry Image Pairs", arXiv:1911.08541v2 [cs.CV] 25 Nov 2019.

[10] Jaakko Lehtinen1 2 Jacob Munkberg1 Jon Hasselgren1 Samuli Laine1 Tero Karras1 Miika Aittala3 Timo Aila1., "Noise2Noise: Learning Image Restoration without Clean Data", arXiv:1803.04189v3 [cs.CV] 29 Oct 2018.

[11] Ashish Bora, Eric Price, Alexandros G. Dimakis. AmbientGAN: "Generative models from lossy measurements," ICLR, 2018.

[12] Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. "Generative Adversarial Networks," In NIPS, 2014.

[13] Xiao-Jiao Mao , Chunhua Shen , Yu-Bin Yang, "Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections", arXiv:1603.090, 56v2 [cs.CV] 1 Sep 2016.

[14] Meyer Scetbon, Michael Elad, Fellow, IEEE, and Peyman Milanfar, Fellow, IEEE, "Deep K-SVD Denoising", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. X, NO. Y, AUGUST 2019. arXiv:1909.13164v1 [cs.LG] 28 Sep 2019

[15] D. A. Fish, A. M. Brinicombe, and E. R. Pike, "Blind deconvolution by means of the Richardson–Lucy algorithm", J. Opt. Soc. Am. A/Vol. 12, No. 1/January 1995.

[16] S. Meivel, A.Nivetha, M.Ramkumar, S.Mohana priya, R.Poongodi, "UAV- Real Time Video Stabilization Using OPENCV Technical Analysis", March 2007International Journal of Innovative Research in Computer and Communication Engineering 5(3):4734.

[17] Jiaya Jia and Chi-Keung Tang Vision and Graphics Group, Computer Science Department Hong Kong University of Science and Technology, "Image Repairing: Robust Image Synthesis by Adaptive ND Tensor Voting", in Proceedings IEEE CVPR 2003.

APPENDIX

Model summary		
Layer (type)	**Output Shape**	**Param #**
input_1 (InputLayer)	(None, 128, 128, 1)	0
conv2d (Conv2D)	(None, 128, 128, 16)	160
batch_normalization	(None, 128, 128, 16)	64
conv2d_1 (Conv2D)	(None, 128, 128, 16)	2320
max_pooling2d	(None, 64 , 64, 16)	0
conv2d_2 (Conv2D)	(None, 64 , 64, 32)	4640
batch_normalization_1	(None, 64 , 64, 32)	128
conv2d_3 (Conv2D)	(None, 64 , 64, 32)	9248
max_pooling2d_1	(None, 32 , 32, 32)	0
conv2d_4 (Conv2D)	(None, 32 , 32, 64)	18496
batch_normalization_2	(None, 32 , 32, 64)	256
conv2d_5 (Conv2D)	(None, 32 , 32, 64)	36928
max_pooling2d_2	(None, 16 , 16, 64)	0
conv2d_6 (Conv2D)	(None, 16, 16, 128)	73856
batch_normalization_3	(None, 16, 16, 128)	512
conv2d_7 (Conv2D)	(None, 16, 16, 128)	147584
max_pooling2d_3	(None, 8, 8, 128)	0
conv2d_8 (Conv2D)	(None, 8, 8, 256)	295168
batch_normalization_4	(None, 8, 8, 256)	1024
conv2d_9 (Conv2D)	(None, 8, 8, 256)	590080
conv2d_transpose (Conv2DTranspose)	(None, 16, 16, 128)	131200
concatenate (Concatenate)	(None, 16, 16, 256)	0
conv2d_10 (Conv2D)	(None, 16, 16, 128)	295040
batch_normalization_5	(None, 16, 16, 128)	512
conv2d_11 (Conv2D)	(None, 16, 16, 128)	147584
conv2d_transpose_1 (Conv2DTranspose)	(None, 32, 32, 64)	32832
concatenate_1 (Concatenate)	(None, 32, 32, 128)	0
conv2d_12 (Conv2D)	(None, 32, 32, 64)	73792
batch_normalization_6	(None, 32, 32, 64)	256
conv2d_13 (Conv2D)	(None, 32, 32, 64)	36928
conv2d_transpose_2 (Conv2DTranspose)	(None, 32, 32, 64)	8224
concatenate_2 (Concatenate)	(None, 64, 64, 64)	0
conv2d_14 (Conv2D)	(None, 32, 32, 64)	18464
batch_normalization_7	(None, 32, 32, 64)	128
conv2d_15 (Conv2D)	(None, 32, 32, 64)	9248
conv2d_transpose_3 (Conv2DTranspose)	(None, 128, 128, 16)	2064
concatenate_3 (Concatenate)	(None, 128, 128, 32)	0
conv2d_16 (Conv2D)	(None, 128, 128, 16)	4624
batch_normalization_8	(None, 128, 128, 16)	64
conv2d_17 (Conv2D)	(None, 128, 128, 16)	2320
conv2d_18 (Conv2D)	(None, 128, 128, 1)	17
Total params: 1,943,761 **Trainable params: 1,942,289** **Non-trainable params: 1,472**		

Fig. 12: Image Denoising Auto encoder- model summary.