Image Denoising using Autoencoders

Denoising noisy imgaes by removing noisy pixels/grains from natural images using Deep learning and autoencoders techniques.

Prof. Uday Kulkarni
School of Computer Science
KLE Technological University
Hubli, India
uday_kulkarni@kletech.ac.in

Sachin Patil
School of Computer Science
KLE Technological University
Hubli, India
sachin.cs2002p@gmail.com

Vikas E
School of Computer Science
KLE Technological University
Hubli, India
biougevikas0@zohomail.com

Rahul Patil
School of Computer Science
KLE Technological University
Hubli, India
rahulrpatil200215@gmail.com

Bodha Kulkarni
School of Computer Science
KLE Technological University
Hubli, India
bodhakulkarni2247@gmail.com

Dr. Meena S. M.
School of Computer Science
KLE Technological University
Hubli, India
msm@kletech.ac.in

Akshay Shanbhag
School of Computer Science
KLE Technological University
Hubli, India
akshayshan2001@gmail.com

***Abstract*—Denoising images is widely used in applications from critical medical systems to software based image enhancement in our cell phones. The natural noise is simulated by adding noise to an image in random pixels. Currently the denoising problem can be solved in either with greedy algorithm or by deep learning techniques. In this paper we are discussing the usage of Autoencoders a deep learning technique to solve the problem of denoising. Autoencoders use down-sampling and up-sampling techniques to remove the unwanted noise from an image.**

By using Encoders we compress the image gradually to remove the tiny details like noise using convolution layers and ReLU as activation function. The compressed images are fed to Decoders, the decoder up-samples the image bit-by-bit till the image reaches the original resolution, while up-sampling the tiny details like human hair or tiny speckles on face are restored without trying to bring back the noisy pixels. To train the proposed model we used FFHQ-Face dataset which consists of 70 thousand images of 128x128 resolution, each image is a face of and unique individual. The output of the neural network is a denoised image with PSNR of above 35, which maximum achieved compared to other pre-trained models.

***Index Terms*—Autoencoders, PSNR, MSE, SGD, ReLU, Sigmoid, Convolution(keywords)**

I. Introduction

Deep learning is one of the machine learning techniques built on artificial neural networks. In fields like computer vision, speech recognition, and natural language processing, deep-learning architectures have been used.

Autoencoder [11] [14] is a type of deep-learning technique for efficient encoding of data. The encoder attempts to down-sample or encode the input vectors without loosing the higher level details. The encoded data can be directly fed to decoder or pass as input to the hidden convolution layers. The output of the hidden layers or encoded data are passed to decoder, the decoder up-samples or decodes the encoded vectors and the outputs are similar to the original vectors. By this way network is trained to ignore/discard unwanted details(noise) [15].

Sometimes when we capture an image using cameras we can absorb that the image contains small grains(dots) of unwanted pixels, they are called noisy images. The noise may be due to moving objects, poor lighting, color-brightness mismatch, or spatial pixel misalignment. There are critical applications such as Medical or bio-metric systems where clean images are a necessity; So denoising plays a vital role for these critical systems [5].

Images can be denoised using techniques such as CNN, Autoencoders, UNET [5], BM3D [1], and other Deep learning techniques. To artificially add noise to clear images there are various random variable generation methods such as Gaussian noise, Poisson noise, salt and pepper, and many more. For our experiment, we will be adding noise instead of noisy images because in this way we can train our model with various noise types. [2]

In this proposed work we are using autoencoders. Using 128 x 128 color images (3 channels) as input for the neural network. The dataset is of hundreds of clear images of different resolutions. Resize the images to 128 x 128 to match the input of the neural network, and store the resized images as the actual output. Add Poisson noise [14] to create training input

Identify applicable funding agency here. If none, delete this.

data and pass through the neural network to train the model to remove the noise. Once the model is trained to remove the noise from the training images we test the model by passing real-world noisy images and verifying the output.

In The second section of this paper, we will be addressing the current work related to this problem. In the third section, we discuss the proposed methodology. In the fourth section, we explain how we implemented the proposed system and the environment for setup and implementation. In the fifth section, we discuss the results of the implemented model along with the outcomes from the existing research.

II. Related Work

The authors of the paper "A Novel Image Denoising method using CNN" [2] proposed a residual method of CNN by constructing symmetric and dilated residual networks with Leaky ReLU. They replaced SGD with Batch Normalization to solve internal co-variate shifting and reduce the vanishing gradient. Batch normalization was used before each layer of the network and combined the normalization, scaling, and shift operation to solve the co-variate transformations.

They used Leaky ReLU as an activation function so the neuron with negative values thresholds are updated and preserve the useful information of the image during the training phase. The authors initialized the weights of the network by using the Gaussian distribution function.

The proposed CNN was made up of 10 layers consisting of Dilated convolution, Batch Normalization. Leaky ReLU and a Convolution layer. Each layer of the network was made up of various kernels of sizes 3x3, 5x5, 7x7, 9x9, 11x11, and also penalty layer after layer 1. By deploying this model they achieved a PSNR of 33 and SSIM of 0.86.

The authors of "Facial Image Denoising Using Convolutional Autoencoder Network" [6] proposed a convolutional autoencoder approach for image denoising based on the ORL dataset. They resize the image to 128 by 128 before using autoencoders to save computing resources. The photos are then divided into 75% for training and 25% for testing.

Using NumPy's random normalisation function, they introduced noise to the photos with a noise factor of 0.2 using three distinct noise methods: Gaussian noise, salt and pepper noise, and Poisson noise. After adding noise, they are made up of two 2D convolutional layers for encoding and two 2D convolutional layers for decoding.

In the paper "Image noise reduction by denoising autoencoder" [7] they used Denoising Autoencoders (DAE) to solve the slow noise reduction, and dimension reduction, which will be fed as input to hidden layers. Recurrent neural networks (RNN) are used with DAE in order to denoise continuous streams of images. Initially, 3 models were created with Input and output shapes 512x512x3x128. Later the final model was designed with the same input and output layers and the middle layer contains 128x128x32, 64x64x128, and 128x128x16.

The authors of the paper "Residual learning of deep CNN for image denoising" [1] suggested a deep CNN model called DnCNN(Denoising Convolutional Neural networks). Residual learning and batch normalization are used with SGD and Adam optimizer. The model uses modified VGG architecture with the Kernel size of Convolutional layers set to 3x3 and pooling layers are removed.

For training and testing images of size 180x180 are used, Images of three different noise levels are used σ = 15, 25, and 50. Images are taken from Berkeley BSD68 dataset.

A single model is able to perform all three tasks of Gaussian denoising, super-resolution, and image de-blocking.

The authors of "Denoising high-resolution multi-spectral images using deep learning approach" [10] discussed a stacked autoencoder method for image denoising using 17 high-resolution multi-spectral images. Each multi-spectral image consists of 4 bands i.e blue, green, red, and near-infrared bands.

The noise in these photos is a mixture of Gaussian and Poisson noise. 12 out of 17 photos from each image were picked for training from 100,000 random patches of size 26 x 26. Divided the entire image into 8 horizontal sections and randomly picked 12,500 patches from each portion to guarantee that the identical patches did not represent the same place in the image.

The author of the work "Improved Denoising Auto-encoders for Image Denoising" [11] introduces unsupervised Denoising Auto-encoders and builds an improved auto-encoders training loss function based on Method noise reduction and residual entropy maximisation.

The model is trained on the benchmark MNIST dataset which has 10 classes with 60000 images with a size of 28x28. Gaussian noise and salt pepper noise are added to test denoising for different noise types.

The convolutional encoders have kernel size 3x3 and pooling layers of 2x2 and decoders with convolutional layers of size 3x3 and sampling layers of size 2x2. The model performs when compared to other state-of-the-art methods.

III. Methodology

A. Adding noise to images

Initially, we collected clear images from various sources of various sizes. Then resize the images to 128 x 128 using the wand library in python, and store the resized images in a separate directory as clear images.

We convert all the images to PyTorch tensors which helps for easy computation of the images. We add the noise to clear images by creating another tensor of the same shape as the input image. Create the noise tensor using the **torch.poisson** function,where Poisson function is

$$P(x) = \frac{e^{-}\lambda * \lambda^{x}}{x!} \tag{1}$$

Where x is the input image, λ is the mean of image pixel values and e is the Euler's constant. We add the noise to the image by storing the Poisson-generated values (P_n) in one tensor and the image data in another tensor (I). The), resulting tensor is an image with Poisson noise (I_{P_n}).

$$I_{P_n} = I + P_n \tag{2}$$

B. Autoencoders

Unsupervised machine learning algorithms such as autoencoders attempt to encode input images and then attempt to decode them using a set number of bits from the latent space representation. Encoder and decoder are the two primary components, respectively. The encoder converts the input images into a hidden representation, while the decoder uses the encoder's codes to recreate the original input images.

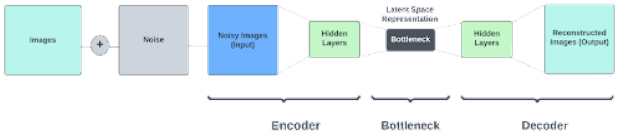

Fig. 1. Autoencoder network

The images in form of tensors are fed to the Encoder where the image is compressed gradually logarithmic ($\log_2 n$) till image it reaches desired size; the small details like noise and tiny details are lost after down-sampling then the output vector tensors of Encoders is fed as input to the Decoder. The decoder decompressed the images by tensor up-sampling till the size matches the original images. While up-sampling the neural network layers of Decoder tries to reconstruct images and bring back the lost details without bringing back the noisy pixels.

C. Neural Network layers

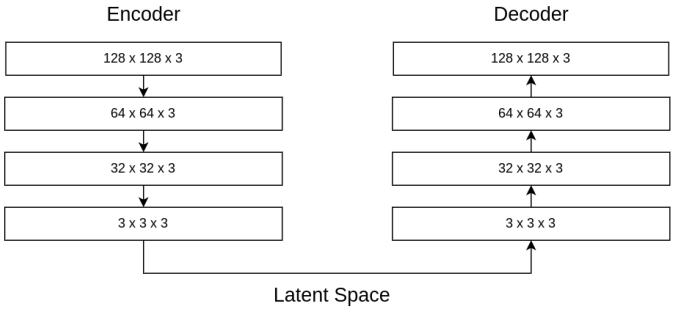

Fig. 2. Layers inside the Autoencoders

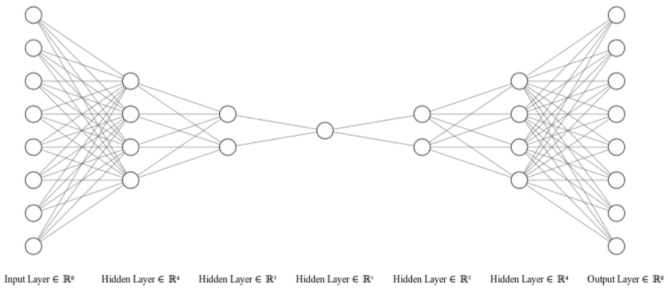

Fig. 3. Representation of layers inside autoencoders [12]

1) Encoder: We are using the input images of 128 x 128, 3 channels(RGB) which is compressed by degrees. First the 128x128 is compressed to 64x64 with ReLU as activation function and max pool of (2x2). The same is done to further compress the image to 32x32 and then 3x3. All the down-sampling is done using defined Convolution 2D function.

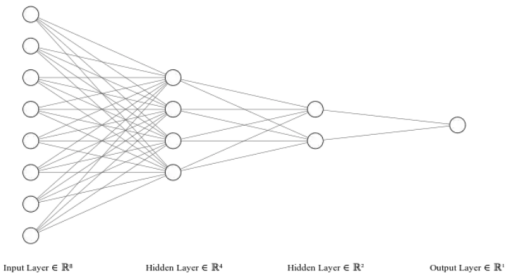

Fig. 4. Encoder layers [12]

2) Latent space: After the tensors are down-sampled by the Encoder there's a latent space between Encoder and Decoder where the Tensors size is 3x3x3. This 3x3x3 tensors are fed to The Decoder for further processing.

3) Decoder: The input of 3x3x3 tensors are fed to Decoder which is initially up-sampled from 3x3 to 32x32 with Sigmoid as activation function. Then its up-sampled to 64x64 later 128x128 but this time using ReLU as activation function. The up-sampling is done by Convolution Transpose.

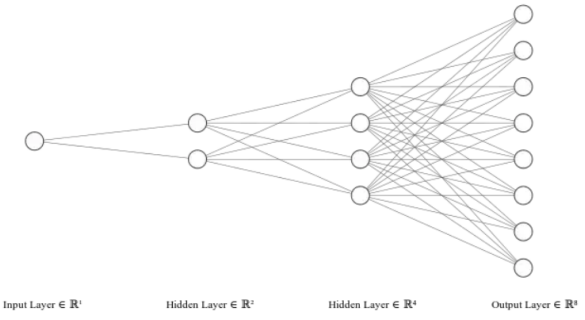

Fig. 5. Decoder layers [12]

D. Result evaluation parameters

1) PSNR: **Peak Signal to Noise Ratio** is unit for measuring the amount of noise in an image. Its value is number of clear pixels per each noisy pixel in an image. below is an equation to find PSNR of any image.

$$PSNR = 10 * \log_{10}\left(\frac{MAX_i^2}{MSE}\right) \tag{3}$$

2) MSE: **Mean Square Error** is the average squared difference between the estimated value and the actual value also known as loss.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{4}$$

Where n is number of data points, Y_i is actual value and $\hat{Y}_i$ is the predicted value.

IV. Setup and Experiment

A. System Specification

The proposed model was trained on a standard NVIDIA DGX-1 node with the specifications below.

Operating System	Ubuntu 20.04 LTS x86_64
CPU	Intel Xeon E5-2698 v4
Memory	512GB
GPU	NVIDIA Tesla V100
GPU VRAM	32GB

TABLE I
Hardware Specifications

Python	Version 3.8
PyTorch	Stable Build 1.13.0
Jupyter Notebook	v6.4.12

TABLE II
Environment Specification

B. Dataset

We used FFHQ-Face Dataset which has 70,000 color images in **128px x 128px** resolution. Each image is of a person's face. This dataset is generally used for image recognition and for Deep learning models. We used this dataset to train our proposed model. We split the dataset as 80% for training (56,000 images) and 20% for validation (14,000 images).

C. Hyper Parameters

1) Learning Rate α: We Initially set the learning rate of $5e^{-3}$ as learnt from literature survey [3]. The growth of accuracy was not satisfactory as compared to the results. Later we tried various other values for α eventually we fixed to $\alpha = 1e^{-3}$.

2) Batch size: Instead of playing with various batch sizes we analyzed the hardware of the setup and fixed the batch size to 16 images.

3) Epochs: Initially we created over own dataset of only 200 images so for the purpose of testing we over fit the model with 1000 epochs. Later when we switched to the desired dataset we initially set the epoch to 25, then gradually increase till 40. After the comparison of results we realized that 25 epochs is perfect fit for our experiment.

D. Adding noise

After loading clean images as tensors create a noise Tensor where the values of the tensor are generated by Poisson random value generating function. The Poisson tensor is added to image tensors to create images with Poisson noise.2

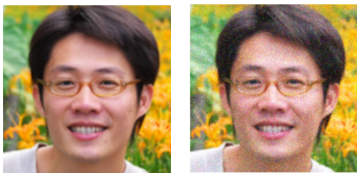

Fig. 6. Clear vs Noisy images

V. Results and Discussion

After successfully experimenting with the implementation of the proposed model here are the results of the experiment.

A. Growth of PSNR for every every epoch

As the denoising models are measured using PSNR our major focus during the experiment was focused on increasing the PSNR, bellow is a graph showing growth of PSNR as number of epochs increase.

The PSNR value varies between 35 and 36 till 40 epochs, but after that the PSNR slowly starts going up but the model starts over-fitting to the training dataset, which would result in bad prediction for unexpected images.

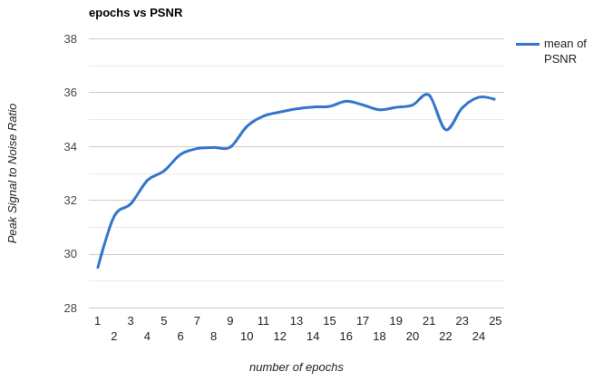

Fig. 7. Epochs vs PSNR

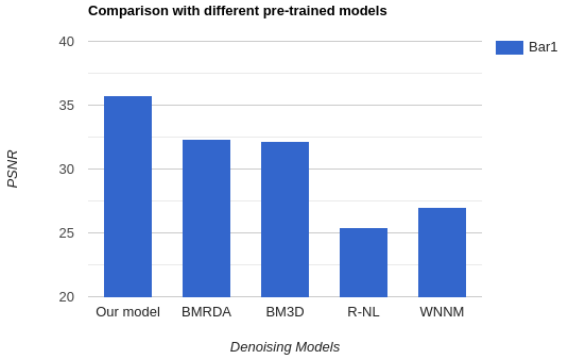

Fig. 9. Comparison with different denoising models

B. Reducing error(MSE) by each epoch

As with any self-supervised models the error reduction is the major focus, so we decided to go with learning rate of $1e^{-3}$ to handle the loss handling. We use Mean Square Error(MSE) to measure the error/loss between iterations. The above graph represents the decrease in loss as number of epochs increase.

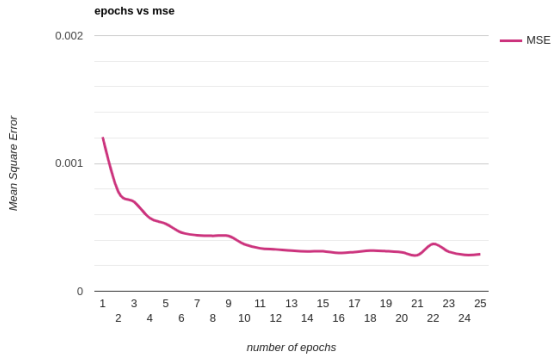

Fig. 8. Epochs vs MSE

C. Comparison with existing model

There already exist many pre-trained models for solving the denoising problem, but most of the models yield PSNR which is less than our proposed architecture. And most of the current experiments which have higher PSNR are trained to only work with a 28x28 image dataset, which is MNIST dataset; open-source dataset which is build into various python libraries. Bellow is a graph representing PSNR of top pre-trained models for image denoising compared to our proposed model.

D. Resulting images

The images from the dataset were split to 20% for validation(14,000 images) and these validation images are passed through the trained model for testing the reliability of the training, The mean PSNR of the validation was same as that of the training results. Above are few images where we can see that the denoised images are remarkably similar to that of the ground truth.

Fig. 10. Caption

VI. Conclusion

This paper has discussed the methods for solving the image denoising problem using deep learning technology. We proposed an autoencoder architecture to implement the denoising model, by implementing various types of convolution layers; at end we constructed a neural network of 8 layers for encoder and 7 layers for decoder with combination of Convolution layers and Transpose layers with ReLU or Sigmoid as activation functions and Max-pool for down sampling. The end results are clean images with **PSNR of 35** or above which is way higher than the predefined models and other existing solutions.

VII. Acknowledgment

The completion of this research would be incomplete without the guidance from Prof. Uday N Kulkarni who was nothing but constant source of knowledge and enthusiasm and whose valuable suggestions were very helpful for working on this project. We take this opportunity to also thank our Head Dr. Meena S. M. for her tremendous source of inspiration. We would like to thank the staff and university for providing the resources and the knowledge for us to successfully complete this project.

References

[1] Kai Zhang, Wangmeng Zuo, Yunjin Chen, DeyuMeng, Lei Zhang "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising." Hong Kong Polytechnic University, Hong Kong, vol. 26, Issue 7, July 2017. - in press

[2] Yizhen Meng, Jun Zhung "A Novel Gray Image Denoising Method Using Convolutional Neural Network" Tangshan Normal University, Tangshan, China, vol. 10, April 2022. - in press

[3] Wei Shan, Peng Liu, Lin Mu, Caihong Cao, Guojin He "Hyperspectral Image Denoising with Dual Deep CNN" vol. 7, November 2019. - in press

[4] Bo Juang, Yao Lu, Jiahuan Wang, Guangming Lu, David Zhang "Deep Image Denoising with Adaptive Priors" Harbin Institute of Technology, Shenzhen, China, vol. 32, Issue 8, August 2022. - in press

[5] Milan Tripathi "Facial image denoising using AutoEncoder and UNET" Tribhuvan University, Lalitpur, Nepal, vol. 2, July 2021. - in press

[6] Naing Min Tun, Alexander I. Gavrilov, Nyan Linn Tun "Facial Image Denoising using convolutional Autoencoder Network" Bauman Moscow State Technical University, Moscow, Russia, vol. 1 June 2020. - in press

[7] Lev Yasenko, Yaroslav Klyatchenko, Oksana Tarsenko-Klyatchenko "Image noise reduction by denoising autoencoders" National Technical Univesity of Ukraine, Kyiv, Ukraine, June 2020. - in press

[8] Lovedeep Gondara "Medical Image Denoising Using Convolutional Denoising Autoencoders" Simon Farser University, December 2016. - in press

[9] Sharath Solomon "Image Denoising using Deep Learning" August 2021. - unpublished

[10] U. Ojha and A. Garg, "Denoising High Resolution Multispectral Images Using Deep Learning Approach," 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), 2016, pp. 871-875, doi: 10.1109/ICMLA.2016.0156.

[11] Q. Xiang and X. Pang, "Improved Denoising Auto-Encoders for Image Denoising," 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2018, pp. 1-9, doi: 10.1109/CISP-BMEI.2018.8633143.

[12] Abien Fred Agarap "Implementing an Autoencoder in TensorFlow 2.0", March, 2019.

[13] Santiago L. Valdarrama "Convolutional autoencoder for image denoising", March, 2021

[14] Michel Kana, Ph.D "Autoencoder For Denoising Images". April, 2021

[15] Arnaghosh "Auto-encoder on torch - trying out the various AEs". January, 2021

[16] Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders. arXiv.

[17] Baldi, Pierre "Autoencoders, Unsupervised Learning, and Deep Architectures", Proceedings of ICML Workshop on Unsupervised and Transfer Learning, July, 2012

[18] Buades, A. and Coll, B. and Morel, J. M. "A Review of Image Denoising Algorithms, with a New One" 2005. doi:10.1137/040616024

[19] Mukesh C. Motwani, Mukesh C. Gadiya, Rakhi C. Motwani "Survey of Image Denoising Techniques "

[20] G. Y. Chen B. Kegl "Image denoising with complex ridgelets", https://doi.org/10.1016/j.patcog.2006.04.039

[21] Han Xiao, Kashif Rasul, Roland Vollgraf "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms", Aug 2017. arXiv:1708.07747

[22] Kulkarni. Uday, S. M. Meena, Sunil V. Gurlahosur, Gopal Bhogar "Quantization Friendly MobileNet (QF-MobileNet) architecture for vision based applications on embedded platforms", 2020. - in press

[23] U. Kulkarni, S. M. Meena, S. V. Gurlahosur and U. Mudengudi, "Classification of Cultural Heritage Sites Using Transfer Learning.", 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM), Conference on Multimedia Big Data (BigMM), 2019, pp. 391-397, doi: 10.1109/BigMM.2019.00020. - in press

[24] U. Kulkarni, S. M. Meena, P. Joshua, K. Rodrigues and S. V. Gurlahosur, "Integrated Crowdsourcing Framework Using Deep Learning for Digitalization of Indian Heritage Infrastructure," 2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM), New Delhi, India, 2020, pp. 200-208, doi: 10.1109/BigMM50055.2020.00036. - in press

[25] Sreekanth, P., Kulkarni, U., Shetty, S., & Meena, S. M. (2019). Head pose estimation using transfer learning. Paper presented at the Proceedings of the 2018 International Conference on Recent Trends in Advanced Computing, ICRTAC-CPS 2018, 73-79. doi:10.1109/ICRTAC.2018.8679209 - in press