International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-10, Oct 2019 E-ISSN: 2347-2693

Denoising Dirty Document using Autoencoder

Mohammad Imran[1*], T. Sita Mahalakshmi[2], M.D. Venkata Prasad[3], V. Kumar Kopparty[4]

[1]Computer Science and Engineering, Neil Gogte Institute of Technology (NGIT), Affiliated to Osmania University, Survey No-35, Peerzadiguda Road, Kachawanisingaram, Uppal, Hyderabad, India
[2]Department of Computer Science and Engineering, GITAM Institute of Technology, Andhra Pradesh, India
[3]Research Scholar (Regd No: *1260316406*), Dept. of Computer Science and Engineering, GITAM Deemed to be University, Visakhapatnam, Andhra Pradesh, India
[4]Research Scholar (Regd No: *41900148*), Dept. of Computer Science and Engineering, LPU (Lovely Professional University), Jalandhar - Delhi G.T. Road, Phagwara, Punjab, India

**Corresponding Author: drimran.ngit@gmail.com*

DOI: https://doi.org/10.26438/ijcse/v7i10.2126 | Available online at: www.ijcseonline.org

Accepted: 10/Oct/2019, Published: 31/Oct/2019

Abstract -An autoencoder is an unsupervised machine learning algorithm [12] that applies back propagation, setting the target values to be equal to the inputs. Deep autoencoders are used to reduce the size of our inputs into a minor representation. If anyone needs the original data, they can reconstruct it from the compressed data.The input seen by the autoencoder is not the raw input but a stochastically corrupted version. A denoising autoencoder is thus trained to reconstruct the original document from the noisy version.In the implementation of Deep autoencoders we have trained the algorithm with noisy and cleaned document images; we generated a model which helps us in removing noise or unnecessary interruption from the documents. Document denoising can be achieved with the deep learning model which automatically learns the discriminative features necessary for classification of input images.

Keywords—document denoising,deep autoencoder,supervised learning, deep learning ,classification,cleaned and noisy images

I. INTRODUCTION

Autoencoder can be broken in to three parts encoder, decoder, latent space, encoder of the network compresses or down samples the input into a fewer number of bits. When the decoder is able to reconstruct the input exactly as it was fed to the encoder, you can say that the encoder is able to produce the best encodings for the input with which the decoder is able to reconstruct well!

MOTIVATION

Many of the recent deep learning models rely on extracting complex features from data. The goal is to transform the input from its raw format, to another representation calculated by the neural network.

This representation contains features that describe hidden unique characteristics about the input.

There are variety of autoencoders, such as the convolutional autoencoder [13], denoising autoencoder, variational autoencoder and sparse autoencoder.The goal of image restoration techniques [1] is to restore the original image from a noisy observation of it and generates the output by removing any noise or unnecessary interruption.

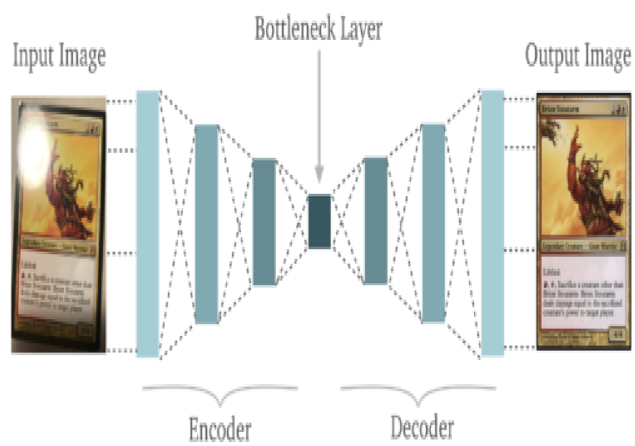


Fig: 1 Feature variation

Dimensionality reduction can be achieved using deep autoencoders,the reconstructed image is the same as our input but with reduced dimensions. It helps in providing the similar image with a reduced pixel value.

Fig: 2 Dimensionality Reduction

Document Denoising is the most prominent and effective technique. The common ideas of these approaches is to transfer image signals to an alternative domain where they can be more easily separated from the noise [2, 3]. In this paper, we use Autoencoder [4] to achieve image denoising.

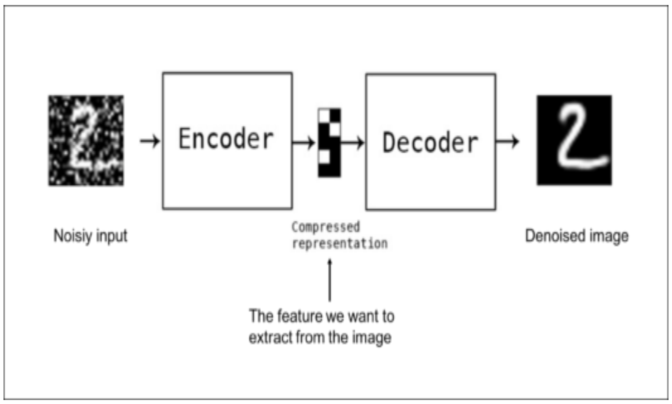


Fig: 3 Denoising Image

Watermark removal. It is also used for removing watermarks from images or to remove any object while filming a video or a movie.

Fig: 4 Watermark Removals

Architecture of Autoencoders [9]

With the prosper development of neural networks, image denoising by neural networks [5] has been a hot topic, an autoencoder consist of three layers:

1. Encoder 2.Code 3.Decoder

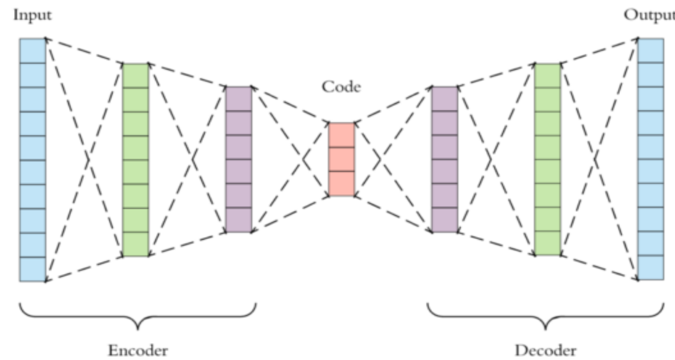


Fig: 5 Architecture of Autoencoders

ENCODER: This part of the network compresses the input into a latent space representation. The encoder layer encodes the input image as a compressed representation in a reduced dimension. The compressed image is the distorted version of the original image.

CODE: This part of the network represents the compressed input which is fed to the decoder.

DECODER: This layer decodes the encoded image back to the original dimension. The decoded image is a lossy reconstruction of the original image and it is reconstructed from the latent space representation.

II. RELATED WORK

The layer between the encoder and decoder, i.e., the code is also known as Bottleneck. This is a well-designed approach to decide which aspects of observed data are relevant information and what aspects can be discarded. It does this by balancing two criteria. Compactness of representation, measured as the compressibility. It retains some behaviourally relevant variables from the input.

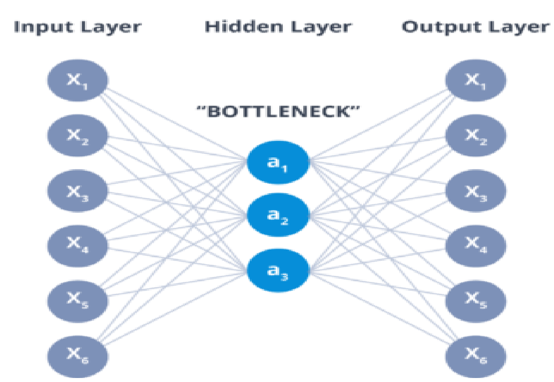


Fig: 6 Layer between the encoder and decoder

$$Y_i = g(f(x_i)) \approx x_i$$

The image shows how a denoising autoencoder may be used to generate correct input from corrupted input. Handwritten digit images are commonly used in optical character recognition and machine learning research [6][7]

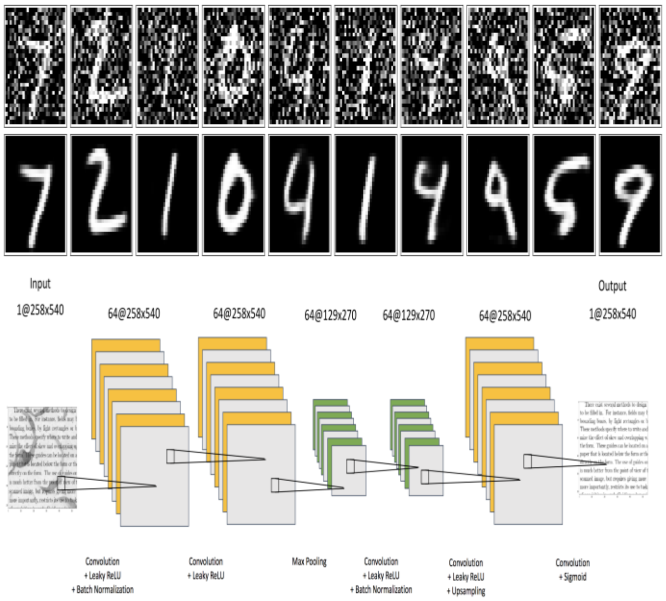


Fig: 7 Algorithm of Denoising autoencoder

As I've mentioned before, autoencoders like the ones we have built so far aren't too useful in practive. However, they can be used to denoise images quite successfully just by training the network on noisy images. We can create the noisy images ourselves by adding Gaussian noise to the training images [8], then clipping the values to be between 0 and 1. We'll use noisy images as input and the original, clean images as targets. Here's an example of the noisy images I generated and the denoised images.

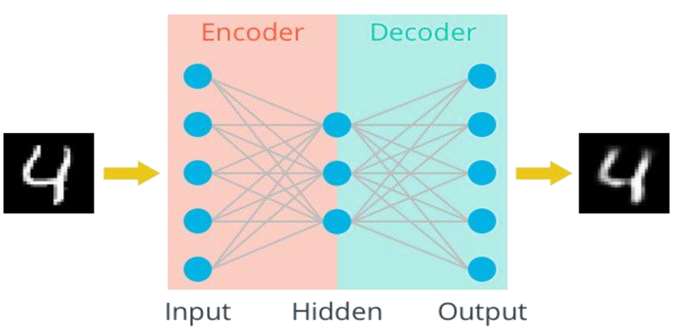


Fig: 8 Simple autoencoder

III.RESULTS AND DISCUSSION

We use keras API [11] which uses tensorflow as backend we loaded the following libraries using keras.layers we load Input[9], Dense, Conv2D, Maxpooling2D[15], and UpSampling2D from keras.models we import Model[10]

We load the following dataset which are having noisy and cleaned images we divide the dataset in to two parts train_fpath and train_cleaned_fpath with these we generate a model using autoencoder.

1. Steps for loading dirty document dataset

```
train_fpath = "../input/denoising/denoising-dirty-documents/train/train/"
train_cleaned_fpath = "../input/denoising/denoising-dirty-documents/train_cleaned/train_cleaned/"
test_fpath = "../input/denoising/denoising-dirty-documents/test/test/"

print (os.listdir (train_fpath))
```

2.Data exploration

Dataset consisting of three directories out of which two are train and one is test directory

```
print ("No. of files in train folder = ",len(os.listdir
       (train_fpath)))
print ("\n No. of files in train_cleaned folder = ",
       len(os.listdir(train_cleaned_fpath)))
print ("\n No. of files in test folder = ",len(os.listdir
       (test_fpath)))
```

```
['27.png', '126.png', '35.png', '117.png', '203.png', '98.png', '167.png', '69.png',
'83.png', '119.png', '141.png', '156.png', '87.png', '56.png', '162.png', '18.png', '1
7.png', '113.png', '72.png', '176.png', '39.png', '9.png', '47.png', '152.png', '66.pn
g', '92.png', '144.png', '189.png', '45.png', '6.png', '143.png', '174.png', '29.png',
'53.png', '188.png', '77.png', '54.png', '60.png', '159.png', '93.png', '89.png', '20
7.png', '170.png', '11.png', '5.png', '132.png', '84.png', '68.png', '111.png', '125.p
ng', '185.png', '90.png', '122.png', '146.png', '95.png', '215.png', '200.png', '171.p
ng', '62.png', '213.png', '42.png', '78.png', '191.png', '96.png', '105.png', '149.pn
g', '164.png', '20.png', '138.png', '194.png', '86.png', '107.png', '110.png', '120.pn
g', '30.png', '74.png', '81.png', '206.png', '197.png', '23.png', '173.png', '99.png',
'150.png', '32.png', '131.png', '137.png', '129.png', '153.png', '177.png', '180.png',
'212.png', '14.png', '48.png', '198.png', '161.png', '26.png', '101.png', '75.png', '2
1.png', '140.png', '44.png', '192.png', '63.png', '116.png', '135.png', '102.png', '11
4.png', '24.png', '182.png', '179.png', '155.png', '128.png', '50.png', '209.png', '3
6.png', '134.png', '8.png', '65.png', '3.png', '57.png', '186.png', '183.png', '104.pn
g', '216.png', '201.png', '2.png', '123.png', '147.png', '41.png', '15.png', '158.pn
g', '51.png', '12.png', '71.png', '59.png', '168.png', '80.png', '108.png', '195.png',
'38.png', '165.png', '204.png', '33.png', '210.png']
```

```
No. of files in train folder =  144

No. of files in train_cleaned folder =  144

No. of files in test folder =  72
```

3. Load noisy images

```
def load_images(fpath):
    images = [ ]
    for image in os.listdir(fpath):
        #print(fpath+image)
        if image!='train' and image!='train_cleaned'
                                        and image!='test':
            img = cv2.imread(fpath+image)
            img = cv2.cvtColor(img, cv2.COLOR_BG
                    R2RGB)

            img_array = Image.fromarray(img, "RGB")

            resized_img = img_array.resize((252,252))

            images.append(np.array(resized_img))
    return images
train_images = load_images(train_fpath)
train_images = np.array(train_images)
print("No. of images loaded = ",len(train_images),"\nShape
of the images loaded = ",train_images[0].shape)
```

```
No. of images loaded =  144
Shape of the images loaded =  (252, 252, 3)
```

4.Load clean images

```
train_cleaned_images = load_images
                        (train_cleaned_fpath)
train_cleaned_images = np.array
                        (train_cleaned_images)
print("No. of images loaded = ",
                len(train_cleaned_images),"\n
Shape of the images loaded = ",
                train_cleaned_images[0].shape)
```

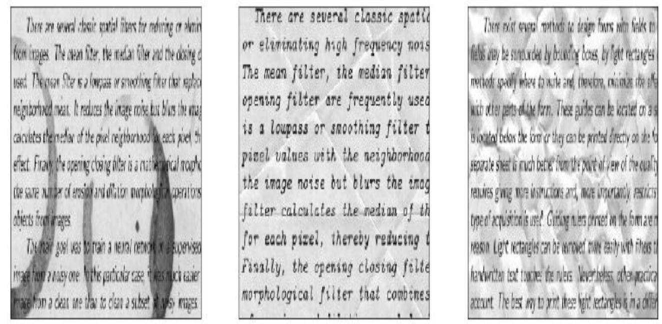
```
No. of images loaded =  144
Shape of the images loaded =  (252, 252, 3)
```

5. Load noisy test images

```
test_images = load_images(test_fpath)
test_images = np.array(test_images)
print("No. of images loaded = ",len(test_images),"
\nShape of the images loaded = ",test_images[0].
shape)
```

```
No. of images loaded =  72
Shape of the images loaded =  (252, 252, 3)
```

```
Displaying noisy training images
```



6. Data normalization [16]

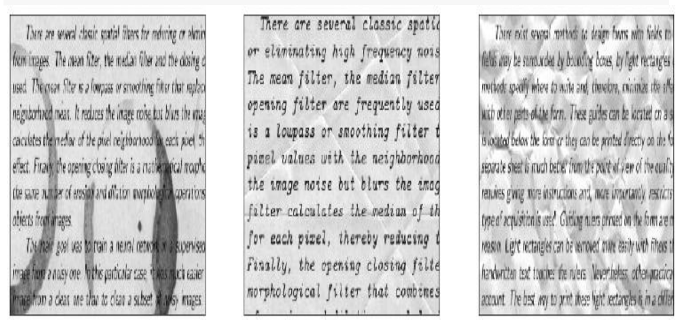
```
train_images = train_images.astype (np.float32)
train_cleaned_images = train_cleaned_images.astype
(np.float32)
test_images = test_images.astype(np.float32)

train_images = train_images/255
train_cleaned_images = train_cleaned_images/255
test_images = test_images/255
print(train_images[0].shape, train_cleaned_images[0].shape,
test_images[0].shape)
```

```
(252, 252, 3) (252, 252, 3) (252, 252, 3)
```

7. Displaying noisy training images after normalization

```
print("Displaying noisy training images
        after normalization")
display_images(train_images)
```



8. Define Deep autoencoder

You are provided two sets of images, train and test. These images contain various styles of text, to which synthetic noise has been added to simulate real-world, messy artifacts[20]. The training set includes the test without the noise (train_cleaned).

we create an algorithm to clean the images in the test set.

```
input_img = Input(shape=(252, 252, 3))

x = Conv2D(32, (3, 3), activation='relu',
        padding='same')(input_img) [14]
x  = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(32, (3, 3), activation='relu',
        padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')
(x)
x = Conv2D(32, (3, 3), activation='relu',
padding='same') (encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(3, (3, 3), activation='sigmoid',
padding='same') (x)

autoencoder = Model(input_img, decoded) [17]
autoencoder.compile(optimizer='sgd', [18] ,
loss='binary_crossentropy')
autoencoder.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 252, 252, 3)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 252, 252, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 126, 126, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 126, 126, 32)      9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 63, 63, 32)        9248
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 126, 126, 32)      0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 126, 126, 32)      9248
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 252, 252, 32)      0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 252, 252, 3)       867
=================================================================
Total params: 29,507
Trainable params: 29,507
Non-trainable params: 0
```

Now let's train autoencoder for 400 epochs:[19]

```
Epoch 1/400
144/144 [==============================] - 5s 31ms/step - loss: 0.6621
Epoch 2/400
144/144 [==============================] - 1s 4ms/step - loss: 0.5485
Epoch 3/400
144/144 [==============================] - 1s 4ms/step - loss: 0.3896
Epoch 4/400
144/144 [==============================] - 1s 4ms/step - loss: 0.3584
Epoch 5/400
144/144 [==============================] - 1s 4ms/step - loss: 0.3623
Epoch 6/400
144/144 [==============================] - 1s 4ms/step - loss: 0.3714
Epoch 7/400
144/144 [==============================] - 1s 4ms/step - loss: 0.3712
Epoch 8/400
144/144 [==============================] - 1s 4ms/step - loss: 0.3704
Epoch 9/400
144/144 [==============================] - 1s 4ms/step - loss: 0.3671
Epoch 10/400
144/144 [==============================] - 1s 4ms/step - loss: 0.3676
Epoch 11/400
144/144 [==============================] - 1s 4ms/step - loss: 0.3659
Epoch 12/400
144/144 [==============================] - 1s 4ms/step - loss: 0.3613
```

autoencoder.fit(train_images, train_cleaned_images,epochs=400, batch_size=100, shuffle=True)

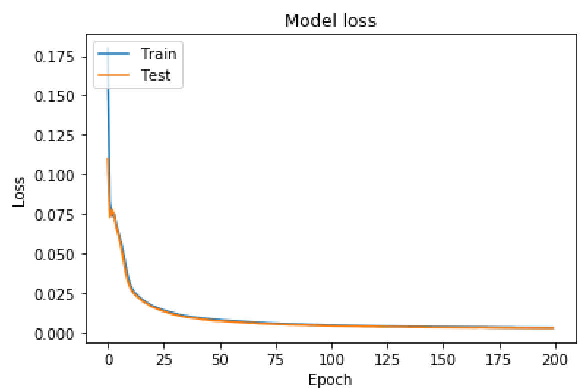


Fig : 9 Model loss

IV.CONCLUSION

The main purpose of this paper is to implement autoencoder for denoising dirty document to generate reconstructed image from the latent space,After 400 epochs, the autoencoder seems to reach a stable train/test loss value of about 0.2065. We can try to visualize the reconstructed inputs and the encoded representations[21]. We will use Matplotlib to display clean images predicted by the autoencoder for the given test images.

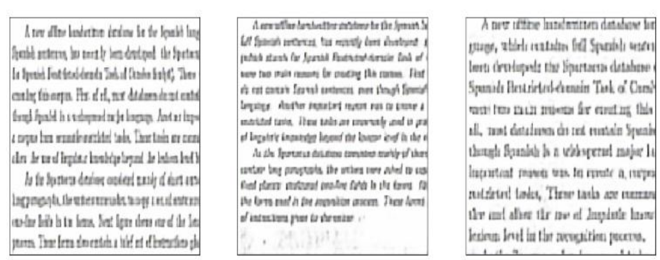


Fig:10 clean image after using auto encoder

REFERENCES

[1]. Xie, J., Xu, L., Chen, E.: Image denoising and in painting with deep neural networks. In: NIPS. (2012)

[2]. J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. Image Processing, IEEE Transactions on, 12(11):13381351, 2003.

[3]. F. Luisier, T. Blu, and M. Unser. A new SURE approach to image denoising: Interscale orthonormal wavelet thresholding. IEEE Transactions on Image Processing, 16(3):593606, 2007.

[4]. K. Matsumoto et al.,"Learning classifier system with deep autoencoder," 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, 2016,pp. 4739- 4746.

[5] A. Krizhevsky, I. Sutskever and G. Hinton,"ImageNet classification with deep convolutional neural networks", Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017.

[6] Semeion Research Center of Sciences of Communication, via Sersale 117, 00128 Rome, Italy Tattile Via Gaetano Donizetti, 1-3-5, 25030 Mairano (Brescia), Italy.

[7] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," in IEEE Signal Processing Magazine, vol. 29, no.6, pp.141-142, Nov.2012.

[8] J. Schmidhuber, "Deep learning in neural networks: An overview", Neural Networks, vol. 61, pp. 85-117, 2015.

[9] "All About Autoencoders", Pythonmachinelearning.pro, 2018.

[10] "Image recovery Theory and application", Automatica, vol. 24, no. 5, pp. 726-727, 1988.

[11] "Building Autoencoders in Keras", Blog.keras.io, 2018. [12] M. Celebi and K. Aydin, Unsupervised learning algorithms.

[13] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks", Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017.

[14] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In ICML, 2010

[15] "PyTorch", Pytorch.org, 2018.

[16] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778. DOI: 10.1109/CVPR.2016.90

[17] T. D. Gedeon and D. Harris, "Progressive image compression," [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, Baltimore, MD,1992, pp. 403-407 vol.4.

[18] L. Bottou. Large-scale machine learning with stochastic gradient descent. COMPSTAT, 2010.

[19] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. ICLR, 2015.

[20] A. V. Lugt, "Signal detection by complex spatial filtering," in IEEE Transactions on Information Theory, vol. 10, no. 2, pp. 139-145, Apr 1964.

[21] E. Kaur and N. Singh, "Image Denoising Techniques: A Review", Rroij.com, 2018.

AUTHORS PROFILE

Dr.Mohammad Imran received his B.Tech (CSE) in 2006 and M.Tech (CSE) in 2008 from JNTU, Hyderabad, His Research interests include Big Data Analytics, Deep learning, Artificial Intelligence, Class Imbalance Learning, Ensemble learning, Machine Learning and Data mining. He completed his Ph.D (CSE) in March 2019 in the department of Computer Science and Engineering, Rayalaseema University, Kurnool-518007, Andhra Pradesh. He has published more than 13 research papers in reputed international journals including Scopus Indexed (SCI & Web of Science) and conferences including IEEE and it's also available online .He is currently working as an Associate Professor in Department of CSE, Neil Gogte Institute of Technology (NGIT),Affiliated to Osmania University, Survey No-35 Peerzadiguda Road, Kachawanisingaram,Uppal, Hyderabad,Telangana, India.

Dr. Tummala Sita Mahalakshmi is working as a Professor in the Department of Computer Science and Engineering, GITAM University. She has published more than 15 research papers in reputed international journals including Thomson Reuters (SCI & Web of Science) and conferences including IEEE and it's also available online. Her main research work focuses on Cryptography Algorithms, Network Security, Cloud Security and Privacy, Big Data Analytics, Data Mining. She has 20 years of teaching experience.

Mr. Maradana Durga Venkata Prasad received his B.TECH (Computer Science and Information Technology) in 2008 from JNTU, Hyderabad and M.Tech. (Software Engineering) in 2010 from Jawaharlal Nehru Technological University, Kakinada, He is a Research Scholar with Regd No:1260316406 in the department of Computer Science and Engineering, Gandhi Institute Of Technology And Management (GITAM) Deemed to be University,Visakhapatnam,Andhra Pradesh, INDIA His Research interests include Clustering in Data Mining ,Big Data Analytics, Artificial Intelligence, Class Imbalance Learning, Ensemble learning, Machine Learning and Data mining.He is currently working as an Assistant Professor in Department of Information Technology, Muffakham Jah College of Engineering and Technology, Banjara Hills, Hyderabad-500034,Telangana,INDIA. He is also an industrial trainee where he teaches programming languages. He is the author of several research papers in the area of Software Engineering.

Mr. Vinay Kumar Kopparty received his B.TECH (Computer Science and Information Technology) in 2008 from JNTU, Kakinada and M.Tech. (Computer Science and Engineering) in 2012 from Jawaharlal Nehru Technological University, Kakinada. He is a Research Scholar with (Regd No: 41900148), Department of Computer Science and Engineering, LPU (Lovely Professional University), Jalandhar - Delhi G.T. Road, Phagwara, Punjab (India)- 144411.His Research interests include Clustering in Data Mining, Big Data Analytics, Artificial Intelligence, Class Imbalance Learning, Ensemble learning, Machine Learning and Data mining.He is currently working as an Assistant Professor in Department of IT, JBREC, Moinabad, Hyderabad, Telangana.

