

**--------------------------------------------------------**

**Centum - Data analytics Training Project completion report**

**--------------------------------------------------------**

**Project Title: Image processing**

**Denoising of medical image using CNN**

**Project Lead: Nilesh Singh**

**Project team members: Atul Tiwari**

**Bhavesh Sharma**

**Jitendra Singh**

**Gaurav**

**Batch: MCA**

**Date: 31/01/2021**

# Section 1: Project description

**Image processing** can be defined as the technical analysis of an **image** by using complex algorithms. Here, **image** is used as the input, where the useful information returns as the output.

* Medical Image de-noising means the removal of unwanted noises from the medical images so that disease prediction can become more precise and treatment can do efficiently.
* Image Denoising  is one of the most important process in image processing.
* Most Real Life Medical Images provide different types of noise distortions as a challenge in  image denoising.
* Now-a-days interest of the radiologists is towards medical data mining..
* In medical Imaging, denoising is very important for analysis of image, diagnosis and treatment of diseases. Currently, image diagnosis methods are based on Deep Learning.
* Application od Medical Image denoising:

1. Preprocessing of Lung Images for Enhancing the Quality and Performance

2. Denoising techniques for medical ultrasound and for magnetic resonance images

3. Practical applications will often simplify the theory using heuristics, when this leads

to algorithms lower complexity or higher flexibility.

To achieve our goal, we will use one of the famous machine learning algorithms out there which is used for Image Classification i.e. Convolutional Neural Network(or CNN).

For the dataset we will use the kaggle dataset of lung images.

• train dataset

• test dataset

Now after getting the data set, we need to preprocess the data a bit and provide labels to each of the image given there during training the data set. To do so we can see that name of each image of training data set is either start with “Test” or “Train” so we will use that to our advantage then we use one hot encoder for machine to understand the labels.

**Section 2: Libraries Required**

# • TFLearn – Deep learning library featuring a higher-level API for TensorFlow used to create layers of our CNN

# • tqdm – Instantly make your loops show a smart progress meter, just for simple designing sake

# • numpy – To process the image matrices • open-cv – To process the image like converting them to grayscale and etc.

# • os – To access the file system to read the image from the train and test directory from our machines

# • random – To shuffle the data to overcome the biasing • matplotlib – To display the result of our predictive outcome.

# • tensorflow – Just to use the tensorboard to compare the loss and adam curve our result data or obtained log.

# • Seaborn- statistical data visualization. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics

# TRAIN\_DIR and TEST\_DIR should be set according to the user convenience and play with the basic hyperparameters like epoch, learning rate, etc to improve the accuracy. os.mkdir("Images/train") os.mkdir("Images/test")

**Problem statement:**

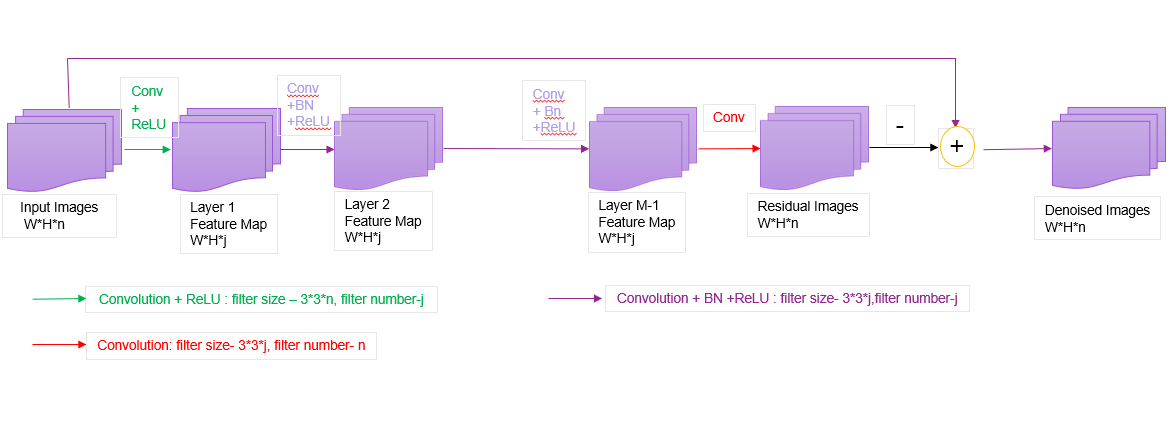
After all survey and conclusion we came with the idea that the

* The increasing number of patient data in medical images imposes a research challenge for the scientific treatment for diagnosing detecting and prediction of the diseases
* Medical images like X-RAY, CT, MR, PET and SPECT have minute information about heart, brain, nerves etc. These images are corrupted by noise during capturing and transmission
* The image interpretation process becomes very tough.
* The Observation through literature survey is that the accuracy rate of existing method is poor so improvement is required to make them more consistent.

**Problem solutions:**

* Making of Machine Learning Model that is based on Convolutional Neural Network which will contain all the filters required to denoise MRI or USI Image.
* This model will have same error rate efficiency like those of data mining technique which radiologists were interested in.
* The filters namely Weiner Filter, Gaussian Filter, Median Filter which are found to remove most common noises Salt and Pepper, Poisson, Speckle, Blurred, Gaussian will be used to remove noises from MRI images in Grey Scale and RGB Scale.
* The Machine learning model will contain different denoising algorithms altogether.
* Unsupervised learning that synthesizes training from specific noise models.

**Proposed Methodologies:**

****

**FEASIBILITY STUDY:**

* Feasibility Study The expanding number and quality of the images
* Devitalizes to overpower radiologist’s abilities to translate them.
* In numerous genuine radiologic rehearses, mechanized and smart images investigation and strategy, for example, processing, segmentation and detection in addition to the use of intelligent algorithm in case of cancer problem in broad area and demand in market.
* Nonlinear methods have been the state of art denoising methods. The methods differ from each other on the type of noise used. All types of noise for example impulse, Gaussian white noise and speckle have been kept under considerations with their associated denoising methods.

We have focused on several methods of image denoising in nonlinear domain, although the linear methods are simple, effective and easy to implement but they are limited with high noise densities and complex noise models

**Section 2: Program Execution & results summary:**

Algorithm:

1. **Import Libraries**
2. **Load train and test Dataset**
3. **Identify features and response variable(s) and value(s) must be numeric numpy arrays**
4. **Make directory for test and training of data**

os.mkdir("Images/train") os.mkdir("Images/test")

## Use 60% of the total dataset for Training and 40% as Test Dataset and write the data in their respective directories :

if(i<round(0.6\*len(files))): cv2.imwrite("Images/train/"+str(i)+".png",x)

else:

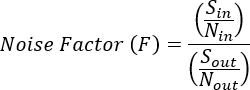
cv2.imwrite("Images/test/"+str(i)+".png",x)

## Get the image and Rescale it by 255 pixels

1. **Reshape the image till 64\*64 matrices (64,64,1)**

x = np.reshape(x,(64,64,1)) //np here is Numpy library

## Add noise by using noise factor

img+= noise\_factor\*np.random.normal(mu,sigma,size=img.shape) img = np.clip(img,0.,1.)

return img

## We are using poissons noise to train our model :

def poisson\_noise(self,img):

img+= numpy.random.poisson(img).astype(float) img = np.clip(img,0.,1.)

return img

1. **Read in each input, perform preprocessing and get labels**

img = self.getImage(self.path+input\_path) img = self.rescale(img)

img = img.astype(np.float)

## Return a tuple of (input, output) to feed the network second argument is p in pa per for gaussian noise

batch\_x = np.array( batch\_input ) batch\_y = np.array( batch\_output ) yield batch\_x, batch\_y

## (batch x : Input variables\_values\_training ) (batch y: Target variables\_values\_training )

1. **Important) split all images in folder "Images" in 60-**

**40 percent as 60% images in "Images/train" folder and 40% in "Imag es/test" folder**

train = DataGen("Images/train/",16,(64,64))

train\_gen = train.generate() # training set generator test = DataGen("Images/test/",10,(64,64))

test\_gen = test.generate() # test set generator

## Now use epoch end to train only when the value loss is less then

**0.23 (if value less is more stop the training of data)**

1. **Use keras methods and Maxpool 2d for convolutional methods to create a window clipping for testing of the data set .**

**(The window used here is of (3,3)of(64,64,1) where the denoising will be done)**

1. **let it run upto 10 epochs** plt.plot(hist.history["loss"],label="Training Loss") plt.plot(hist.history["val\_loss"],label="Validation Loss") **(If val\_loss>0.23 the epoch end will be called)**

## Test the data gen set

1. **Upscale again all (3,3) into (64,64,1) after Denoising .**

z.shape

img1 = r[0][:,:,0]

img2 = r[1][:,:,0]

img3 = r[2][:,:,0]

img4 = r[3][:,:,0]

img5 = r[4][:,:,0]

den\_img1 = z[0][:,:,0]\*255.

den\_img2 = z[1][:,:,0]\*255.

den\_img3 = z[2][:,:,0]\*255.

den\_img4 = z[3][:,:,0]\*255.

den\_img5 = z[4][:,:,0]\*255.

1. **predicted result:**

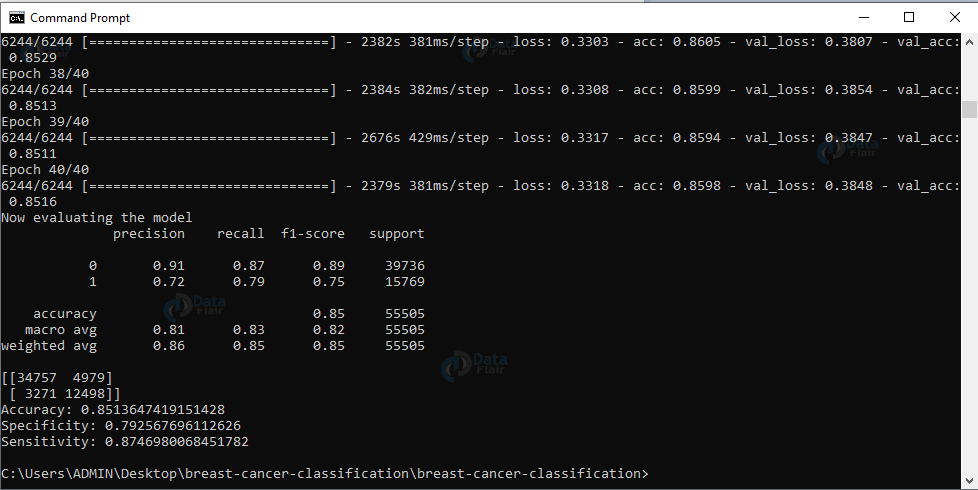
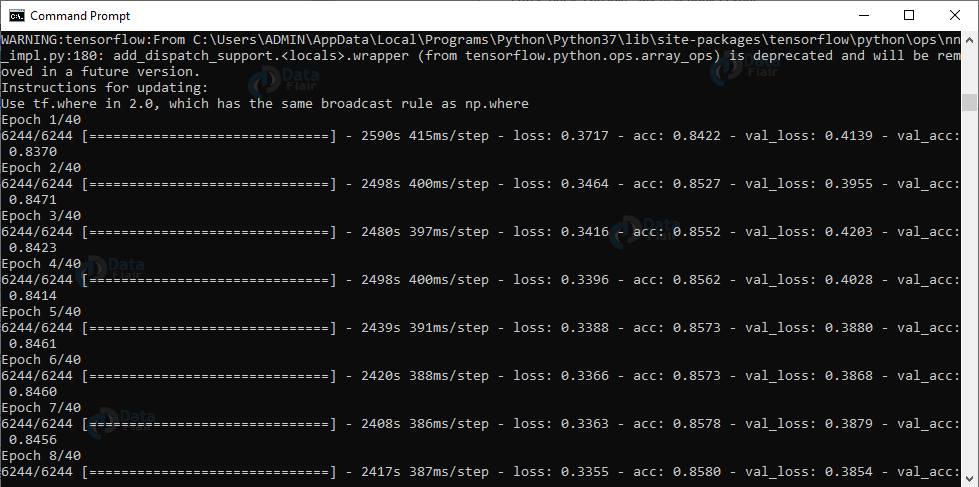
autoencoder.predict(r)

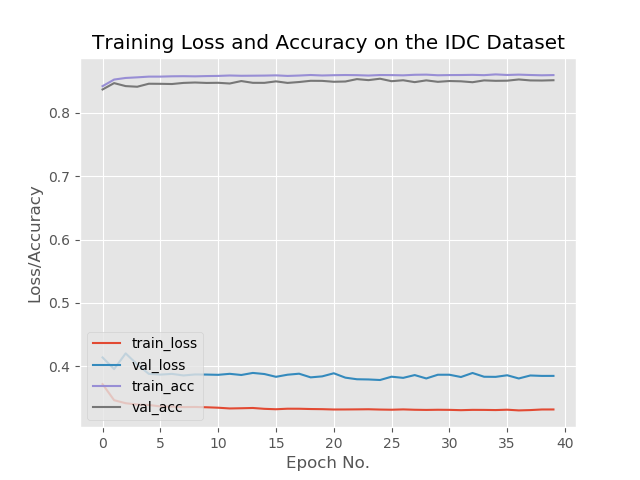
1. **Output:**

ssim\_sum += ssim(originalSet[i], noisySet[i],data\_range=originalSet[ i].max() - noisySet[i].min(), multichannel=True)

return 1.0\*ssim\_sum/originalSet.shape[0] get\_ssim\_result(r, z)

## END





**Future Scope:**

* AI-enabled technologies are plying a bigger role in analyzing such images with accuracy. Actually, these medical images are used to train the machines through computer vision and then a capable AI machine scan and analyze the images with possible malady detected.
* Image processing and artificial intelligence will involve spoken commands, anticipating the information requirements of governments, translating languages, recognizing and tracking people and things, diagnosing medical conditions, performing surgery, reprogramming defects in human DNA, and automatic driving all forms of transport.
* With increasing power and sophistication of modern computing, the concept of computation can go beyond the present limits and in future, image processing technology will advance and the visual system of man can be replicated.

**Conclusion:**

* Nonlinear methods have been the state of art denoising methods. The methods differ from each other on the type of noise used. All types of noise for example impulse, Gaussian white noise and speckle have been kept under considerations with their associated denoising methods.
* Every method has its own performance measures in its problem domain which may not work and fulfill the requirements in other problem domain. However, overall methods associated with wavelet domain have achieved great performance due to their noise adaptive and sparseness.
* Median based methods have been outstanding for image restoration because of nonlinearity but comparatively spatial domain computations are complex and time consuming.
* Future work include examining larger amounts of data for the denoiser to converge towards a better optimum solution.