Grenze International Journal of Engineering and Technology, June Issue

Image De-noising using Deep Convolution Neural Networks

Samyuktha Inala[1], N. Aleshwari[2], N. Bhanu prasad[3] and Dr. G.N. Swamy[4]
[1-4]VR Siddhartha Engineering College /EIE, Vijayawada, India
Email: samyukthainala@gmail.com, {nedadhavollualeshwari , nakkabhanuprasad7 , gavini_s }@gmail.com

***Abstract*— The sources of noise present substantial obstacles for image denoising. Gaussian, impulse, and speckle noise are particularly complex sources of noise in imaging. Convolutional neural networks (CNN) have gained popularity in picture denoising tasks. Several CNN denoising approaches have been investigated. These approaches evaluated using various datasets. In this research, we present an in-depth investigation of various CNN approaches for speckle noise removal in picture denoising. Different CNN image denoising algorithms were classified and analyzed. The study investigated popular datasets for testing CNN image denoising methods. A few works on CNN image denoising were chosen for evaluation and analysis. CNN approaches' motivations and concepts were described. We proposed a CNN-based picture denoising review.**

***Index Terms*— de-noising, CNN (convolutional neural networks), Speckle noise and Machine Learning.**

I. INTRODUCTION

Convolutional neural networks have been around for decades [1], and deep CNNs have recently exploded in popularity, thanks in part to their performance in image categorization [2], [3]. They've also been used successfully in other computer vision fields like object detection [4], [7], face recognition [6], and pedestrian detection [5]. Several factors are critical in this progress:
(i) Efficient training implementation on modern powerful GPUs [9],
(ii) The proposal of the rectified linear unit (ReLU), which allows for much faster convergence while maintaining good quality [9],
(iii) Easy access to a large amount of data (such as ImageNet [8]) for training larger models. These advancements enhance our strategy as well.

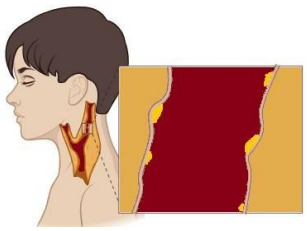

Fig 1: Carotid Artery

Grenze ID: 01.GIJET.10.2.197

“Fig 1” illustrates the carotid arteries which are main blood vessels in the neck that transport oxygenated blood from the heart to the brain. Normally, the lining of these arteries is smooth, allowing unhindered flow through them. Plaque, which is made up of cholesterol, calcium, fatty cells, and fibrous tissue, can accumulate in the vessel wall over time, much like rust does in a pipe. Plaque build up causes the carotid arteries to become unhealthy, making them harder and narrower, which can lead to stroke or death. The degree of artery narrowing and whether there have been "warning signs" of an oncoming neurological catastrophe determine the risk of this result.

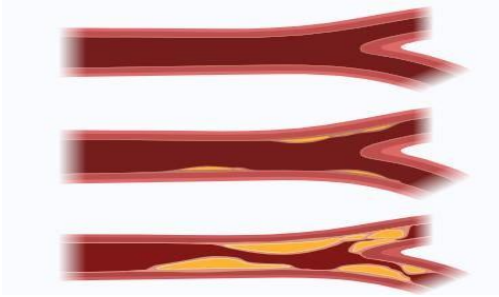

Fig 2: Plaque Formation

In “Fig 2” , Carotid Artery Plaque was removed. Plaque in an artery is prone to splitting under blood flow stress, and the body responds by generating a clot on the broken plaque surface. This clot may be so large that it completely clogs the artery. Alternatively, it may become dislodged and migrate downstream to a smaller artery within the brain matter, with or without a component of the plaque. If it obstructs this artery, oxygen delivery beyond it is cut off. Brain cells in that area quit operating. The neurological dysfunction is only brief if the body is effective in breaking up the blockage and restoring forward flow via the artery. If the condition lasts shorter than 24 hours, it is referred to as a Transient Ischaemia Attack (or TIA). If the condition lasts fewer than 24 hours. If the body is unable to restore flow rapidly and the shortage lasts more than 24 hours, a permanent Stroke (or CVA) is stated to have occurred.

During acquisition and transmission, digital images [10] are frequently damaged with noise, decreasing performance in later tasks such as image identification and medical diagnosis. Many noise reduction techniques have been developed to increase the accuracy of these tasks when using corrupted images. Development of deep learning architecture which is fully automatic in the elimination of speckle noise without losing the edge information and improved contrast to noise ratio.

However, the majority of these techniques are either specifically designed for a specific sort of noise or make assumptions about the statistical characteristics of the corrupting noise. To demonstrate compression artifacts, we will use Di- com file compression as an example.The JPEG compression technique divides an image into 8x8 pixel blocks and performs block discrete cosine transformation (DCT) on each one separately. The DCT coefficients are then quantized to save storage space.

So far, deep learning has shown amazing results on both high-level and low-level vision challenges. Dong et al.'s SRCNN demonstrates the immense potential of an end-to-end DCN in image super-resolution. As seen in Figure 1(a), this approach will produce a complex mix of various artifacts.

II. What Is De-Noising?

The process of de-noising involves the reduction or elimination of noise from an image. It is essential to acknowledge that complete noise elimination may not always be achievable. This section begins with a list of the most prevalent noise models, followed by some denoising approaches.

III. Noise Sources

The principal sources of noise in digital photographs arise during the acquisition phase, resulting from factors such as inadequate photon collection, sensor temperature variations, and other related considerations. Additionally, noise can be introduced during the transmission stage, manifesting as echoes and atmospheric distortions in wireless communication. In other circumstances, noise is also used to mimic flaws in the mathematical model of picture production, which, like any physical model, is bound to diverge from reality!

Because noise is a random phenomenon, it is represented by a probability density that represents the noise's intensity distribution. In the following, we designate the observed (and noisy) image by y, the noise by b, and the non-noisy image by x.

A. Additive Gaussian white noise

Each pixel of the observation is modelled by the sum of a pixel from the noiseless image and a pixel from the noise using additive white Gaussian noise:

$$\forall m,n=y(m,n)=x(m,n)+b(m,n)$$

Where each pixel be (m, n), observation y, noise image x and pixel noise b.

$$b(m,n) \sim N(0,\sigma^2)$$

This model is straightforward and makes calculations easier. It finds application in diverse fields, including photography.

B. Poisson Noise

Poisson noise (also known as shot noise) simulates photon acquisition at a photo site. The quantity of photons is inherently unpredictable and is contingent upon the characteristics of light. The associated Poisson process is characterized by a mean equivalent to the illumination level.

$$\forall m,n \quad y(m,n) \sim P(x(m,n))$$

C. Salt-and-pepper noise

Salt-and-pepper noise, also called less poetically impulse noise, models saturated or dead pixels (due to photo sites malfunction or saturation).

$\forall m,n \quad y(m,n)=\{\blacksquare(x\_[\![min \quad]\!] \quad [\![\quad \text{with probability } P\_min]\!] @x\_max \quad [\![\quad \text{with probability } P\_max]\!] @x(m,n) \quad 1- P\_min- P\_max .)\boxplus$

Where x-min and x-max are the intensity minimum and maximum.

IV. Explanation Of De-Noising

1. The necessary libraries are imported: OS for file operations, numpy for numerical operations, pydicom for reading medical image files in DICOM format, and keras for defining and training a deep learning model.
2. The function load_images_from_folder is defined to read and preprocess the medical image files from a given folder. For each file in the folder, it reads the file using pydicom. dcmread, extracts the pixel array, converts it to a float32 type, adds an extra dimension to the array, and normalizes the pixel values to be between 0 and 1. The function yields a NumPy array containing all the images present in the specified folder.
3. Two folders containing medical images are defined: no_tumor_folder for images without tumors and tumor_folder for images with tumors.
4. The function load_images_from_folder is invoked twice, once for each folder, to load and preprocess the images.
5. The process involves concatenating the two arrays of images using the np.concatenate function to form a unified array encompassing all images, subsequently randomizing their order using np.random.shuffle.
6. The dataset is partitioned into training and testing sets, with the initial 80 images designated for training and the remaining images allocated for testing.
7. The CNN model is defined using Sequential from keras. models and Conv2D, MaxPooling2D, UpSampling2D, and Batch Normalization layers from keras layers. The model has an encoder-decoder structure, with multiple convolutional layers and max pooling layers in the encoder to reduce the spatial dimensionality of the input images, and multiple up sampling layers and convolutional layers in the decoder to reconstruct the original input images. The activation function used in all convolutional layers is ReLU, except for the last layer, which uses the sigmoid activation function to output values between 0 and 1 for binary image segmentation. Batch normalization is added after each convolutional layer to improve the training process by normalizing the inputs to each layer.
8. The compilation of the model involves the utilization of the Adam optimizer with a learning rate set at 0.001,along with a binary cross-entropy loss function.

9. The model undergoes training through the fit method applied to the training data for 100 epochs, employing a batch size of 16 and incorporating a validation split of 0.1. This facilitates the monitoring of performance on a reduced segment of the training data throughout the training process.
10. The evaluation of the model involves utilizing the evaluate method on the testing data to compute the loss.
11. Denoising of the test images is executed by transmitting them through the trained model via the predict method.
12. The original and denoised images are visualized using Matplotlib's pyplot. Each test image generates a figure with two subplots: one for the original image and another for the denoised version. The imshow method is employed for image display, with the cmap parameter configured to 'gray' for grayscale representation.

V. What Is Classification?

The Classification algorithm, belonging to the domain of Supervised Learning techniques, is employed to ascertain the category of novel observations through the utilization of training data. In the context of Classification, the program assimilates information from the provided dataset or observations, subsequently categorizing new observations into various classes or groups.

VI. Classification Of Tumour And Non-Tumour

Here's a step-by-step explanation of what the code does:
1. Imports necessary libraries including OS, random, pydicom, numpy, tensorflow, and matplotlib.
2. Introduces a function named load_dicom, designed to read a DICOM file, normalize its pixel values to a range of 0 to 1, and reshape it into a 3D tensor to align with the input shape requirements of the Convolutional Neural Network (CNN).
3. Sets up the paths to the directories containing the DICOM files of tumor and no tumor images.
4. Loads in the DICOM files of tumor and no tumor images, using the load_dicom function and storing them as numpy arrays.
5. Concatenates the no tumor and tumor image arrays into a single array X and creates a corresponding array y of labels (0 for no tumor and 1 for tumor).
6. Defines a function train_val_split to split the dataset into training and validation sets. This function takes in X and y, and an optional val_ratio parameter (default 0.2) to specify the validation set size as a proportion of the full dataset. The function shuffles the indices of X and y randomly, then splits them at the index corresponding to the specified val_ratio.
7. Calls train_val_split to split X and y into training and validation sets.
8. Specifies a Convolutional Neural Network (CNN) model through the incorporation of Keras layers. The architecture comprises three 2D convolutional layers with Rectified Linear Unit (ReLU) activation, two max pooling layers, a flatten layer, and two dense layers — one activated with ReLU and the other with sigmoid. The model is then compiled using the Adam optimizer with binary cross-entropy loss and accuracy metric.
9. Prints a summary of the model's architecture.
10. Specifies the batch size and the number of epochs for the training process.
11. Establishes an Image Data Generator to apply data augmentation techniques to the training data, encompassing rotation, shifting, shearing, zooming, and flipping. The method employs nearest-neighbor filling to address any gaps in pixel values.
12. Trains the model using the augmented training data and assesses its performance on the validation set. The history object is employed to retain the training and validation loss, as well as accuracy, at each epoch.
13. Plots the training and validation accuracy and loss using matplotlib.
14. Evaluates the model on the validation set and prints the validation loss and accuracy.
15. Defines a function predict image to take in a DICOMfile path, load the image using load dicom, and make aprediction using the trained model. If the predicted probability is greater than 0.5, the function prints "Tumor" otherwise it prints "No Tumor".
16. Calls predict image on a specific DICOM file path, to demonstrate the function's usage.

VII. Functions Used In Programming

A. RELU Activation Function

Illustration of a linear graph depicting a ReLU activation function, with an example of ReLU (depicted in blue) and its variant, Softplus (depicted in green) provided in Fig 3.

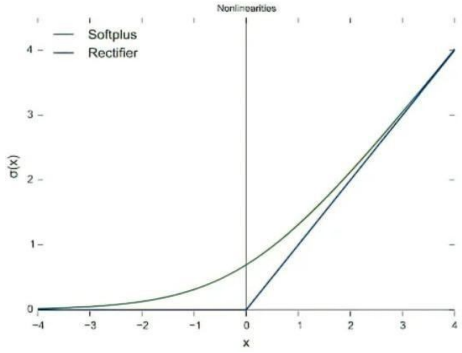

Fig 3: ReLU Activation graph

The depicted diagram features a blue line symbolizing the Rectified Linear Unit (ReLU), while the green line represents a modification of ReLU known as Softplus. The other variants of ReLU include leaky ReLU, exponential linear unit (ELU) and Sigmoid linear unit (SiLU), etc., which are used to improve performances in some tasks.

This article focuses solely on the rectified linear unit (ReLU) as it remains the default and widely employed activation function for a majority of deep learning tasks. Variants of ReLU are generally reserved for specific purposes where they may offer a slight advantage over the standard ReLU.

B. Sigmoid Function

In machine learning, the term sigmoid function is normally used to refer specifically to the logistic function, also called the logistic sigmoid function.

In “Fig 4” , every sigmoid function possesses the characteristic of mapping the complete number line onto a limited range, typically spanning from 0 to 1 or -1 to 1. Consequently, one application of a sigmoid function is the conversion of a real value into a representation that can be interpreted as a probability.

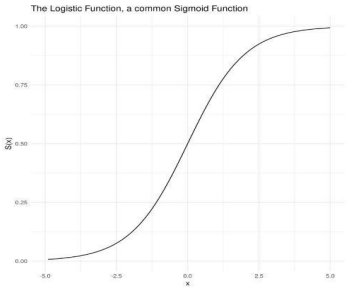

Fig 4: Sigmoid Function wave

VIII. IMPLEMENTATION

A. For Denoising

During the process of implementation of denoising program using CNN (Convolutional Neural Network) we used 128 neurons connected to previous volume. To compile the model, we used Adam-optimizer which is for image recognition and created 5 hidden layers for execution and 1 output layer. Epoch limit is 100.Validation Loss = <40%.

B. For Classification

During the process of implementation of denoising program using CNN (Convolutional Neural Network) we used 64 Neurons. For the compilation of model, we used Adam- optimizer (refer table 1 below) and created 2 hidden layers for the execution and 1 output layer. We will load DICOM file from the drive and classify the image as tumor or non- tumor. Validation Accuracy = 97.24%, Validation loss is 0.1 % [Fig 6 and Fig 7].

TABLE I: SEQUENTIAL OUTPUT

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 510, 510, 32)	320
max_pooling2d (MaxPooling2D)	(None, 255, 255, 32)	0
conv2d_1 (Conv2D)	(None, 253, 253, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 126, 126, 64)	0
conv2d_2 (Conv2D)	(None, 124, 124, 64)	36928
flatten (Flatten)	(None, 984064)	0
dense (Dense)	(None, 64)	62980160
dense_1 (Dense)	(None, 1)	65

Total params: 63,035,969
Trainable params: 63,035,969
Non-trainable params: 0

IX. RESULTS AND DISCUSSION

The research was conducted with the support of Google Colab. Utilizing an excessive number of hidden layers leads to overfitting, resulting in premature termination of execution. Therefore, it is crucial for the number of hidden layers to align proportionally with the size of the datasets.

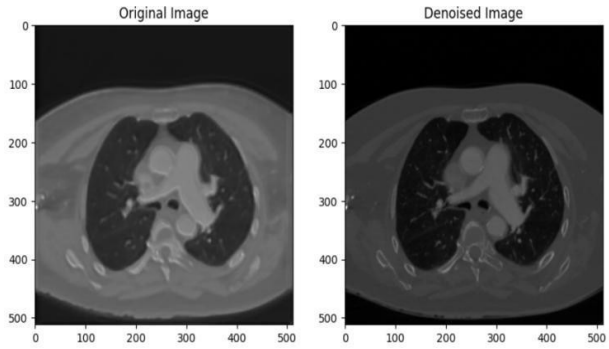

Fig 5: Output of de-noising

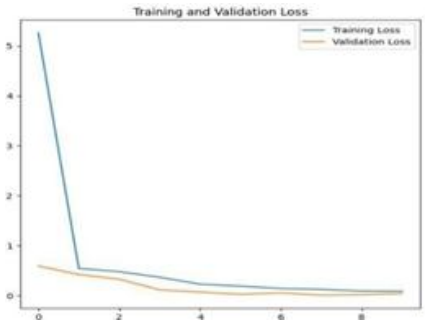

Fig 6:Training and Validation Loss

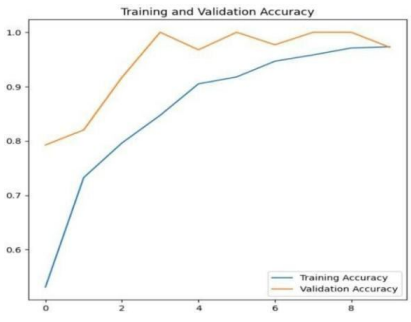

Fig 7: Training and Validation Accuracy

X. CONCLUSION

The application of deep learning methods in picture denoising is experiencing a growing trend. This paper presents an overview of diverse strategies to enhance reader comprehension. Within this section, we explore potential avenues for future research in picture denoising and underscore several unresolved issues. Deep learning-based image denoising approaches are very good at improving denoising performance and efficiency, and moreover performing difficult denoising jobs. The following are some solutions for enhancing denoising performance:

1) Improving the de-imaged output's correctness.
2) As much as feasible, reduce the validation loss of edge information.

REFERENCES

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard,W.Hubbard, and L.D.Jackel, "Backpropagation applied to hand written zip code recognition," Neural Comput., vol. 1, no. 4, pp.541–551, 1989.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in Proc. Eur.Conf. Comput. Vis., 2014, pp. 346–361.

[3] Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. Adv. Neu ral Inf. Process. Syst., 2012, pp. 1097–1105.

[4] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S.Yang, Z. Wang, C.-C. Loy, and X. Tang, "Deepid-net: Deformable deep convolutional neural networks for object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recogn., 2015, pp. 2403–2412.

[5] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in Proc. IEEE Int. Conf. Comput. Vis., 2013, pp. 2056- 2063.

[6] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face repre sentation by joint identification- verification," in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 1988–1996.

[7] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based R CNNs for fine-grained category detection," in Proc. Eur. Conf. Comput. Vis., 2014, pp. 834–849.

[8] IEEE Trans. Image Process., vol. 9, no. 11, pp. 636–650, Apr. 2000. J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei- Fei,"ImageNet: A large-scale hierarchical image database," in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2009, pp. 248–255.

[9] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[10] KC. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In ECCV, pages 184–199.