Use of Machine Learning for Image Processing

Nishita N. Merh[1]

[1]*Student, Madhav Institute of Science and Technology, Gwalior, India*

---***---

Abstract - *This paper discusses about two approaches for image processing. First is conventional digital image processing algorithm and second is the use of convolutional neural network (CNN) for finding image specific parameters. The proposed approach aims to automate the process of finding specific parameters of a given blob image having a normal distribution and using them for its classification.*

Key Words: Convolutional Neural Network (CNN), Digital Image Processing (DIP).

1. INTRODUCTION

In the field of Artificial Intelligence (AI) and Machine Learning (ML), the “data” plays a vital role. This data needs to be processed, analyzed and inferences are drawn and learned. Then only any AI/ML based system can perform optimally. The data may belong to finance, engineering, science, or social fields. For analysis of any kind of data and for getting any inference out of it, a statistical approach is required. This leads to studying the distribution of data. Normal or Gaussian distribution is the fundamental distribution that is used across all the domains ranging from science, technology, medical to finance and social studies.

In this paper we will generate a Gaussian distribution of a blob and find it parameters viz. centre and spread, with conventional digital image processing. The same shall be compared with automated process of finding the parameters with the use of CNN. The results of the processing are presented.

2. METHODOLOGY

The first step done was to generate the required synthetic images or blobs of the normal distribution. First a background image was generated having a 250X250 grid like structure shown in Fig.1.

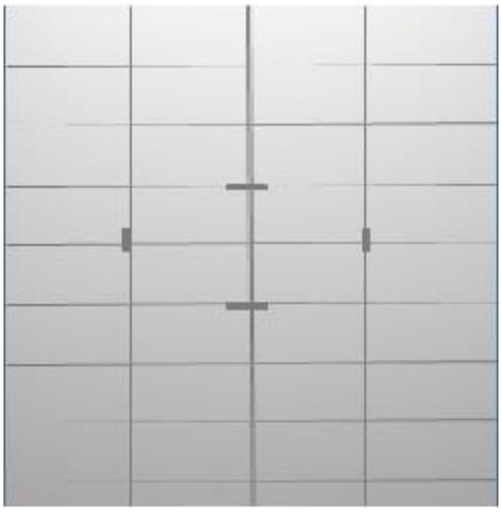


Fig -1: Background Grid

Then a Gaussian image was generated using the equation of a 2D- normal distribution. The 2D Gaussian function used is given as follows [2]:

$$f(x,y) = A\exp\left(-\left(\frac{(x-x_0)^2}{2\sigma_X^2} + \frac{(y-y_0)^2}{2\sigma_Y^2}\right)\right)$$

Where, A is the amplitude, x0 and y0 is the center, and σ_x, σ_y are the x and y spreads of the blob.

The image thus generated is as shown in Fig. 4(a).

2.1 Conventional Digital Image Processing

The steps involved in curve fitting and determination of position and amplitude were:

1. Python library OpenCV (cv2) was used to load the grayscale image. The original image was loaded in grayscale format. [7][9]

2. Cropping was done using NumPy array slicing based on specified target dimensions. The image was cropped to a desired Region of Interest (ROI). [14]

3. Contrast Enhancement: Contrast Limited Adaptive Histogram Equalization (CLAHE) was applied. CLAHE enhanced the contrast of the cropped image, which further

helped in bringing out details in both bright and dark regions.[11]

4. Blob Segmentation: Binary Thresholding (>245) and Contour Finding (cv2.findContours ()) were performed. Thresholding was applied to segment the blob from the background. Contours were then found in the thresholded image, and the largest contour was assumed the blob.[9]

5. Gaussian Curve Fitting: Curve fitting using scipy.optimize.curve_fit was done. Intensity profiles along the x and y-axes were computed. Gaussian curves were fitted to these profiles to estimate the blob's center coordinates and size.[10]

Python libraries used were OpenCV, Numpy, SciPy, Matplotlib.

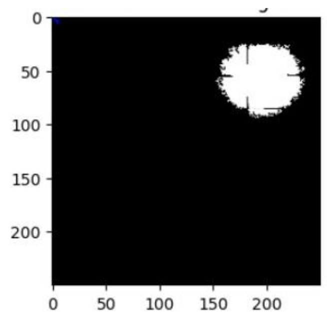
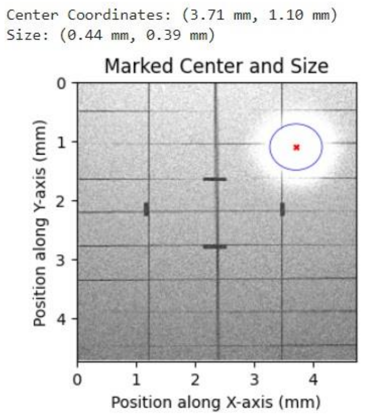


Fig -2 a): Image extracted after processing



```
Center Coordinates: (3.71 mm, 1.10 mm)
Size: (0.44 mm, 0.39 mm)
```

Fig -2 b): Results after applying DIP

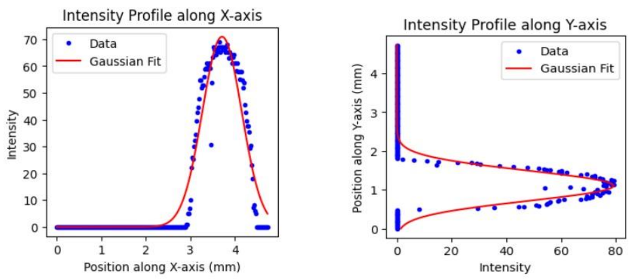


Fig -3: Gaussian fit of blob in x and y directions

The same process was applied to multiple number of images. Though the number increased but the images were of similar type. In order to increase the variety and avoid generating similar images, Gaussian noise as well as salt and pepper noise was introduced in the images as shown in Fig 4(b) and Fig. 4 (c).

However, there were two issues faced in determining the results. First, it was a tedious manual processing which consumed lot of time and secondly the results so obtained needed to be verified for the correctness. The results thus obtained were accurate for only few random images and inaccurate for the remaining set of images.

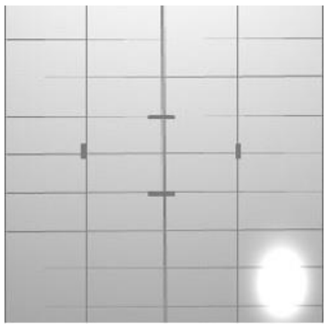


Fig -4 a): Image without noise

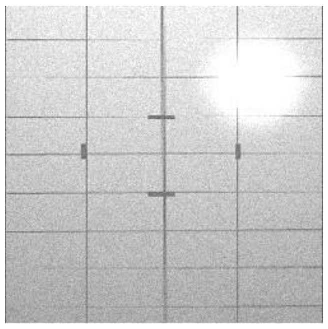


Fig -4 b): Image with gaussian noise

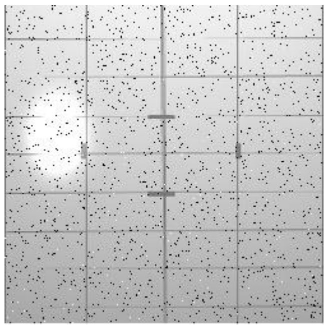


Fig -4 c): Image with salt and pepper noise

2.2 Need for Deep learning

With the use of conventional image processing algorithms, achieving desired results accurately proved difficult. Thus, shifting to a more sophisticated approach involving machine learning, specifically CNN was thought of. CNN is a regularized type of feed-forward neural network that learns feature engineering by itself via filters (or kernel) optimization [3]. Vanishing gradients and exploding gradients, seen during backpropagation in earlier neural networks, are prevented by using regularized weights over fewer connections [3]. Recognizing CNNs' proficiency in learning spatial hierarchies of features from images [4], we made a comprehensive dataset comprising synthetic Gaussian blob profiles and corresponding parameters.

To remove the limitation of manual calculation of the image parameters viz. center point, σx and σy, a convolutional neural network was designed.

A CNN model was constructed with architecture as shown in Table 1. This net was constructed using the PyTorch [8] framework and aims at predicting Gaussian parameters from synthetic images generated with known Gaussian distributions. The implementation involves several key libraries and functions. Firstly, OpenCV (cv2) was utilized for image processing tasks, while numpy served for numerical computations and manipulation. Python library matplotlib.pyplot [12] was used for data and interactions with the operating system were managed with os. Additionally, pandas [13] was employed for data manipulation and analysis, while the core deep learning functionalities were handled by torch and its submodules such as torch.nn for defining neural network architectures and torch.utils.data for data handling. Furthermore, sklearn.model_selection was used for splitting the data into training and testing sets. We utilized nn.Sequential module of PyTorch to construct the CNN layers, including convolutional layers, activation functions (ReLU), pooling layers (Max Pooling), and fully connected layers (Linear). We used Adam optimizer to optimize the results.

Table -1: Architecture of CNN

Layer (type)	Output Shape	Param #
Conv2 -1	[-1, 8, 250, 250]	80
ReLU-2	[-1, 8, 250, 250]	0
MaxPool2d-3	[-1, 8, 125, 125]	0
Conv2d-4	[-1, 16, 125, 125]	1,168
ReLU-5	[-1, 16, 125, 125]	0
MaxPool2d-6	[-1, 16, 62, 62]	0
Flatten-7	[-1, 61504]	0
Linear-8	[-1, 256]	15,745,280
ReLU-9	[-1, 256]	0
Linear-10	[-1, 64]	16,448
ReLU-11	[-1,64]	0
Linear-12	[-1,4]	260

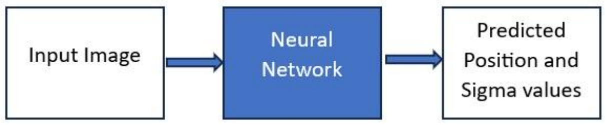


Fig -5: Basic Architecture of CNN

3. RESULTS AND CONCLUSIONS

The neural network was trained for a data set of 900 images and test set of 100 images was taken. A loss function is a measure of how accurately the CNN model can predict the expected outcome. The common loss functions [5] are Mean Squared Error (MSE) and Mean Absolute Error (MAE).

The MSE is given by [6]:

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2$$

Where N is number of samples and Y and Y^ are actual and predicted values. It measures the variance of samples.
The MAE is the average of absolute difference between actual and predicted values of sample and is given by:[6]

$$\mathrm{MAE} = \frac{1}{m}\sum_{i=1}^{m}|X_i - Y_i|$$

Where N is number of samples and Y and Y^ are actual and predicted values.

The Root Mean Squared error gives the standard deviation of the difference of actual and predicted values [6].
It is given by:

$$\mathrm{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2}$$

The following table shows the errors obtained by the model:

Table -2: Results of the model

Model Verification Results	**Value**
Mean Squared Error (MSE)	16.7207
Mean Absolute Error (MAE)	3.0847
R-squared (R2)	0.8105

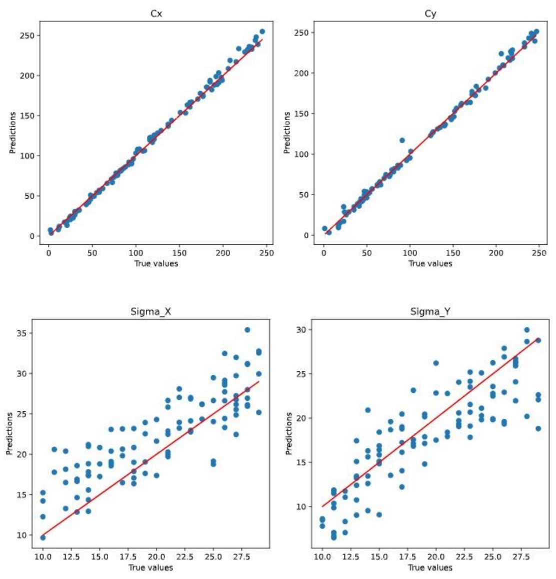


Fig -6: Scatter plot of actual and predicted values of center position and spread of the blob.

The model performance needs to be improved. This can be done by using a bigger data set. In addition, the network layers need to be optimized by changing the architecture of the neural network. Alternative optimizers can be tried out. Pre-trained models can be used for more accurate results. The model can then be tested in real time to verify the results.

REFERENCES

[1] F. M. Dickey and S. C. Holswade, Laser Beam Shaping: Theory and Techniques, Marcel Dekker, New York (2000).

[2] "GLAD optical software commands manual, Entry on GAUSSIAN command" (PDF). Applied Optics Research. 2016-12-15.

[3] Venkatesan, Ragav; Li, Baoxin (2017-10-23). Convolutional Neural Networks in Visual Computing: A Concise Guide. CRC Press. ISBN 978-1-351-65032-8. Archived from the original on 2023-10-16. Retrieved 2020-12-13.

[4] Jonathan Janke, Mauro Castelli, Aleš Popovič, Analysis of the proficiency of fully connected neural networks in the process of classifying digital images. Benchmark of different classification algorithms on high-level image features from convolutional layers, Expert Systems with Applications, Volume 135, 2019, Pages 12-38, ISSN 0957-4174,

[5] J. Qi, J. Du, S. M. Siniscalchi, X. Ma and C. -H. Lee, "On Mean Absolute Error for Deep Neural Network Based Vector-to-Vector Regression," in IEEE Signal Processing Letters, vol. 27, pp. 1485-1489, 2020, doi: 10.1109/LSP.2020.3016837.

[6] Chicco D, Warrens MJ, Jurman G. 2021. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. PeerJ Computer Science 7:e623 https://doi.org/10.7717/peerj-cs.623 https://medium.com/analytics-vidhya

[7] Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org

[8] https://pytorch.org/

[9] https://opencv.org/

[10] https://scipy.org/

[11] G. R. Vidhya and H. Ramesh, "Effectiveness of contrast limited adaptive histogram equalization technique on multispectral satellite imagery", Proc. Int. Conf. Video Image Process., pp. 234-239, Dec. 2017.

[12] https://matplotlib.org/

[13] https://pandas.pydata.org/

[14] https://numpy.org/