ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse49942020.pdf
https://doi.org/10.30534/ijatcse/2020/49942020

# Minimizing Time Complexity by Using Modified Strassen's Matrix Multiplication Algorithm on Convolutional Layers

**Siddharth Narula, Akshit Mittal, Harsimran Jit Singh**
Bachelors of Technology in Department of Computer Science and Engineering,
Guru Tegh Bahadur Institute of Technology
(Affiliated to Guru Gobind Singh Indraprastha University), New Delhi, Delhi, India

**ABSTRACT**

Digital image processing refers to the processing of digital images by a computer. An image is composed of a finite number of elements each having a particular location and value. For image recognition in python, there are many layers called convolutional layers. Each of these layers requires multiplication with one another to generate a result. This is generally done by methods like Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA), which were time-consuming. Using our modified Strassen's matrix multiplication algorithm on the image data yields a better result as it can provide the same accuracy by training more images in lesser time. By using our modified algorithm, we were able to reduce the time by 37.408 %. This study helps in providing a better approach to image recognition than the general approach maintaining the same accuracy rate but reducing the time to achieve it.

**Key words :** Keras, OpenCV, ReLU, ResNet, Strassen's Matrix, TensorFlow, TFLearn

## 1. INTRODUCTION

"'Image Processing' or 'Digital Image Processing' is the technique that uses a computer and other digital technologies to process an image using specific algorithms. Digital Image Processing has many advantages over the previously used technology that is Analog Image Processing. Digital Image Processing allows a wide range of algorithms that can be used on the input data and avoid problems that occurred in Analog Image Processing like the build-up of noise and distortion during the processing" [1]. Such an algorithm is Strassen Algorithm, an algorithm for matrix-multiplication that uses a simple technique of breaking the matrix into small parts and then following the algorithm to obtain a result as a product of two matrices. In contrast, we applied this algorithm to develop a way to use this mechanism to break the image and compare it with the test data.

In Section 2, we have discussed Strassen's Algorithm in detail. Section 3 has general information about image processing. In section 4, there are details about the essential python libraries that are Keras and Tensorflow TFLearn Open CV. Section 5 contains information about layers of the neural network. A general overview of the design and linking between the algorithm and the application is present in section 6. The experimental result is present in section 7. Ultimately we have concluded in section 8.

## 2. STRASSEN'S ALGORITHM

"Strassen's algorithm published in year '1969' proved that multiplication of a matrix by the general method that had a time complexity of n3 was not optimal" [2]. It presented a way to reduce the time complexity to n 2.807. It was a slight decrease in time complexity but was remarkable since matrix multiplication has many applications like graphic scaling, translations, rotations. Strassen's algorithm achieves a lower complexity by using seven multiplications instead of 8 for a 2x2 matrix, which was further reduced by us by substituting a 3x3 matrix using 23 multiplications instead of 27.

Strassen's Algorithm for a 2x2 matrix

$$\begin{bmatrix} a[0][0] & a[0][1] \\ a[1][0] & a[1][1] \end{bmatrix} \begin{bmatrix} b[0][0] & b[0][1] \\ b[1][0] & b[1][1] \end{bmatrix} \begin{bmatrix} c[0][0] & c[0][1] \\ c[1][0] & c[1][1] \end{bmatrix}$$

```
p=(a[0][0]+a[1][1])*(b[0][0]+b[1][1])
q=(a[1][0]+a[1][1])*b[0][0]
r=a[0][0]*(b[0][1]-b[1][1])
s=a[1][1]*(b[1][0]-b[0][0])
t=(a[0][0]+a[0][1])*b[1][1]
u=(a[1][0]-a[0][0])*(b[0][0]+b[0][1])
v=(a[0][1]-a[1][1])*(b[1][0]+b[1][1])
```

We can see from here that we require only seven multiplications p, q, r, s, t, u, and v since seven multiplications are required, thus reducing the number by 1. We obtain our result matrix C by performing

```
c[0][0]=p+s-t+v
c[0][1]=r+t
c[1][0]=q+s
c[1][1]=p-q+r+u
```

Strassen's Algorithm for a 3x3 matrix

$$\begin{bmatrix} a[0][0] & a[0][1] & a[0][2] \\ a[1][0] & a[1][1] & a[1][2] \\ a[2][0] & a[2][1] & a[2][2] \end{bmatrix} \begin{bmatrix} b[0][0] & b[0][1] & b[0][2] \\ b[1][0] & b[1][1] & b[1][2] \\ b[2][0] & b[2][1] & b[2][2] \end{bmatrix}$$

```
[ c[0][0]  c[0][1]  c[0][2] ]
[ c[1][0]  c[1][1]  c[1][2] ]
[ c[2][0]  c[2][1]  c[2][2] ]
```

We can see from here that we require only 23 multiplications Mi 1<i<23, since 23 multiplications are performed, thus reducing the number of multiplications by 4. We obtain our result matrix C by performing

```
m1=(a[0][0]+a[0][1]+a[0][2]-a[1][0]-a[1][1]
    -a[2][1]-a[2][2])*b[1][1]
m2=(a[0][0]-a[1][0])*(-b[0][1]+b[1][1])
m3=a[1][1]*(-b[0][0]+b[0][1]+b[1][0]-b[1][1]
    -b[1][2]-b[2][0]+b[2][2])
m4=(-a[0][0]+a[1][0]+a[1][1])*(-b[0][0]+b[0][1])
m5=(a[1][0]+a[1][1])*(-b[0][0]+b[0][1])
m6=a[0][0]*b[0][0]
m7=(-a[0][0]+a[2][0]+a[2][1])*(b[0][0]-b[0][2]
    +b[1][2])
m8=(-a[0][0]+a[2][0])*(b[0][2]-b[1][2])
m9=(a[2][0]+a[2][1])*(-b[0][0]+b[0][2])
m10=(a[0][0]+a[0][1]+a[0][2]-a[1][1]-a[1][2]
    -a[2][0]-a[2][1])*b[1][2]
m11=a[3][2]*(-b[0][0]+b[0][2]+b[1][0]-b[1][1]
    -b[1][2]-b[2][0]+b[2][1])
m12=(-a[0][2]+a[2][1]+a[2][2])*(b[1][1]+b[2][0]
    -b[2][1]
m13=(a[0][2]-a[2][2])*(b[1][1]-b[2][1])
m14=a[0][2]*b[2][0]
m15=(a[2][1]+a[2][2])*(-b[2][0]+b[2][1])
m16=(-a[0][2]+a[1][1]+a[1][2])*(b[1][2]+b[2][0]
    -b[2][2])
m17=(a[0][2]-a[1][2])*(b[1][2]-b[2][2])
m18=(a[1][1]+a[1][2])*(-b[2][0]+b[2][2])
m19=a[0][1]*b[1][0]
m20=a[1][2]*b[2][1]
m21=a[1][0]*b[0][2]
m22=a[2][0]*b[0][1]
m23=a[2][2]*b[2][2]
```

We recursively keep on dividing the matrix into smaller parts until we get a matrix of size 2x2 or 3x3.

```
c[0][0]=m6+m14+m19
c[0][1]=m1+m4+m5+m6+m12+m14+m15
c[0][2]=m6+m7+m9+m10+m14+m16+m18
c[1][0]=m2+m3+m4+m6+m14+m16+m17
c[1][1]=m2+m4+m5+m6+m20
c[1][2]=m14+m16+m17+m18+m21
c[2][0]=m6+m7+m8+m11+m12+m13+m14
c[2][1]=m12+m13+m14+m15+m22
c[2][2]=m6+m7+m8+m9+m23
```

Originally the Strassen's algorithm was meant to perform on a 2x2 matrix, as the matrices have to be divided into four equal parts recursively. The division stops once we reach a matrix of size 2x2. But here the division ends once we obtained a 2x2 matrix or a 3x3 matrix and the remaining part in still divided further to get a 2x2 or 3x3 matrix from the more substantial order.

## 3. IMAGE PROCESSING

"An image defined as a two-dimensional function f(x, y), where x and y are coordinates of the plane, and the amplitude of function f at any pair of coordinates (x, y) is generally known as the intensity of the image at that point. When x, y, and the amplitude values of f are all finite (discrete quantities), it is called a digital image, and the field of generating this is known as digital image processing or image processing" [3]. "Hence a digital image is composed of a finite number of elements. Each of the components has a particular location and value" [4]. "These are small as compared with the image and are known as pixels. A digital image is a representation of a two-dimensional image as a finite set of digital values called picture elements or pixels" [5]. Pixel values typically represent gray levels, colors, heights, opacities etc. Consider each pixel as a single index of a large matrix; hence the collection of these pixels together forms a matrix. "A digital image represents a matrix of numerical values. These values depict the data associated with the pixels—the intensity of different pixels, average to a single amount, expressing itself in a matrix format" [6].

The recognition system or program takes the input as intensities and location of each of these different pixels in the image. "This information, the system learns to map out a relationship between the subsequent images that are input to it as a part of the learning process" [6]. After the training process is complete, the system performs on test data. Generally, training is done on a large amount of data and testing on twenty to thirty percent of this test data. The performance of the model is determined by how well it is to tell that the test data provided will yield the correct output.

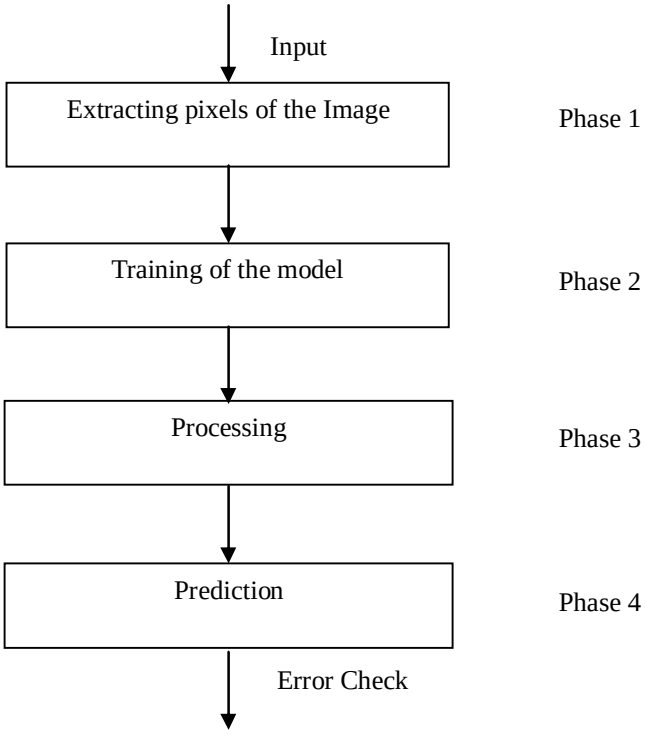


**Figure 1:** Phases of Image Processing

This process of image recognition model is the same as the process of machine learning modeling. Modeling processes for image recognition start from

Phase: 1- Extracting pixel features from an image
From an image, we extract a significant number of characteristics called features each pixel represented by a number or a set of numbers. A bit/color depicts the colors present in each pixel collectively these combine to form color depth or bit depth of an image, indicating the maximum number of colors used to create an image. Consider an example of the black and white or greyscale image, and each pixel will have a value between 0 to 255.

Phase: 2- Training of the model
Once each image is converted into pixels with known labels of the image, it is used to train the model; if more images are used for training for each category, the model can be better trained to tell an image which category it belongs.

Phase 3: Training the model so it can categorize images
The complete network is considered here as a filter. The input is the images and outputs the type of each image. The aim here is to get the image to match the set that is entered in phase 2. More the number of images that are present in the network better, the data would be trained; hence we can expect an error-free or result with fewer errors.

Phase 4: Prediction of a new image
Once the training of a model is done, it can be used to recognize (or predict) an unknown image. The new image follows the same procedure by going through the process of breaking into pixels, each with its different properties, and comparing it with the training dataset to get an output.

## 4. PYTHON LIBRARIES

TensorFlow: "TensorFlow offers multiple levels of abstraction, so it helps to choose the right one for the user's needs. Building and training models using high-level Keras API makes getting started with TensorFlow and machine learning accessible" [7]. TensorFlow provides training of the model irrespective of the platform or language.

Keras: Is a widely-used open source neural network library. It is capable of running on top of TensorFlow, Microsoft Cognitive, and many more. "It is an API designed for human beings, not machines. Keras always follows best practices for reducing cognitive load by offering consistent and straightforward API's" [4]. "It minimizes the number of user actions required for everyday use cases and provides clear and actionable error messages. It also has extensive documentation and developer guides".[8]

TFLearn: "TFLearn is a deep learning library built on top of TensorFlow. It is designed to provide a high-level API for TensorFlow. Using this helps to ease and speed up the experimentations on neural network training" [9]. TFLearn does not have a pre-built API to train a network; instead, it integrates a set of functions that can easily handle any neural network training depending on the number of inputs, outputs, and optimizers.

OpenCV: Open Source Computer Vision Library or OpenCV has many optimized algorithms that can be used for the detection and reorganization of objects. “OpenCV is based on the Inception Resnet engineering and there are commonly 19 layers”[10]. It is capable of classifying human actions in videos. Using OpenCV reading an image is secure, this image is stored in the form of a matrix, and operations on this matrix are performed. The final matrix now generated can be saved and stored as an image in a file.

## 5. LAYERS IN NEURAL NETWORK

In neural networks, layers have an essential role, there are many layers in the neural network, each having a specific task and purpose. Depending on these layers, the complete architecture of the neural network is defined. Keras, as defined in section III, has many layers present in it. These layers are used for adding different properties to the image.

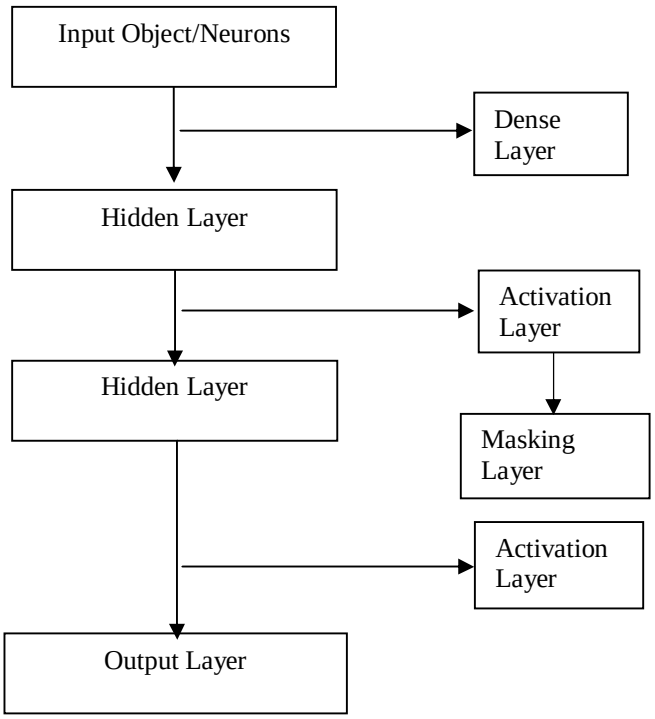


**Figure 2:** Different layers of Neural Networks and their linking with each other

“Input Object: The function Input() is used to instantiate Keras tensor, a TensorFlow symbolic object which helps build a simple Keras model with only input as an argument. “As the radiographs are of different sizes, so before feeding images into the network, we normalized each image to have the same mean and standard deviation of images in the Image Net training set”[11].

Dense Layer: This layer defines the number of neurons needed in the model. Where a neuron is a collection of a single small unit of artificial intelligence called the perceptron” [12].

Activation Layer: In this layer, multiple functions are used to reassemble the neurons form the Dense Layer. It includes many features such as

ReLU Activation Function (Rectified Linear Unit) is an activation function that defines the positive part of the argument where x is the neuron Figure 3.
For f(x)=x if x>0
0 if x<0

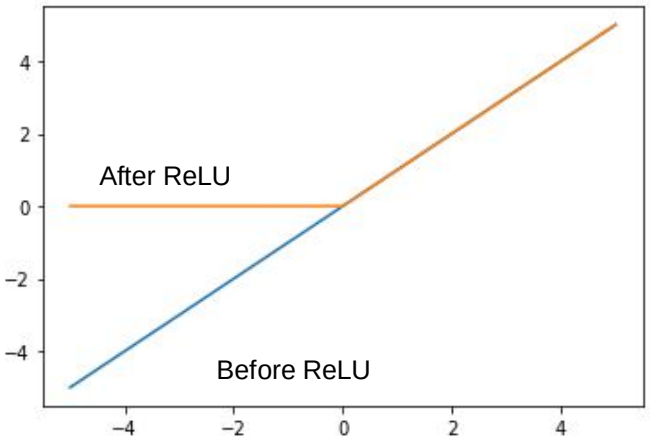


**Figure 3** : Rectified Linear Unit Activation Function (ReLU)

"Softmax Activation Function (Normalized Exponential Function) it takes the input as a vector z of K real number. It normalizes it into a probability distribution function PDF with K probabilities proportional to the input numbers" [13].

Masking Layer: This layer helps to mask a sequence of values to skip timestamps. It uses mask values to achieve this task by having a more significant amount of masking. It is not affected by the input layer as well as by the hidden layer.

Output Layer: It provides labels that help in the prediction of the input image or object.

## 6. DESIGNING THE SOLUTION

While studying about Strassen's Algorithm for Matrix Multiplication in our third year of Bachelor of Technology in Computer Science, We were inclined towards the algorithm and to reduce the time complexity of the same. The work on this direction started in December when we came with a method to reduce the time complexity of Strassen's algorithm by implementing a matrix of order 3x3 to substitute the existing one that had an order of 2x2 which made it easy to divide the matrix if it had odd rows or columns. Strassen's algorithm was suitable for a square matrix; hence it was an easy way to implement on images since an image is considered as a matrix only, so it became convenient for us to use it. Multiplication of a matrix was the issue that we were facing since the algorithm could not give the desired result, but using the same for dividing the image into parts and then comparing it with the original image yielding the desired output in a short period.
"To implement the following, we took a car dataset of 100 images" [14]. The images are read through cv2.imread() function of OpenCV. Then in the general approach, the training of the following dataset is done through the YOLO object detection library or Topic Modeling and Linear Discriminant Analysis (LDA) or ResNet short for Residual Networks or Inception that is the core concept of a sparsely connected architecture. Here to simplify the LDA algorithm, we used Strassen's algorithm for matrix multiplication. Images are being processed as a matrix.

"We calculated that an array is formed called the stride of a collection. Astride of a display can be called increment, pitch, or step size that is the number of memory locations between the beginnings of the successive array elements" [6]. For, an image array is calculated for each row, so we had obtained a matrix instead of the variety called stride matrix. To train the model, this stride matrix is given a maximum memory location value, and it is multiplied by another form called the Filter, which has elements that lie between -1 and 1 only. All these multiplications were done by the usual method in LDA and hence required more time to run. However, as our Strassen's matrix algorithm did the multiplication of these matrices, we obtained a less time complexity solution than the general approach, reducing the time from 30.286 seconds to 23.720 seconds. Since this step is repeated several times in a recursive manner where it stops until the weights assigned to each matrix stops the process. So instead of taking 30.286 seconds each time, it took 23.720 seconds each time. While processing the image, this filter matrix is multiplied by hidden layers, which, if done by the general method, will take $T(n)=n^3$. However, since Strassen's algorithm also did this multiplication for matrix multiplication, the new time complexity would be $T(n)=n^{2.81}$ . When Strassen's algorithm was applied, the difference of the general method to our method came out to be 17.312 seconds forming the matrix of the convolution layer.

## 7. RESULT AND DISCUSSION

Using different algorithms on the same dataset of car images, we came up with the following graphs showing the relationship between the Number of Images taken on the Y-axis and Time in Seconds taken on the X-axis.

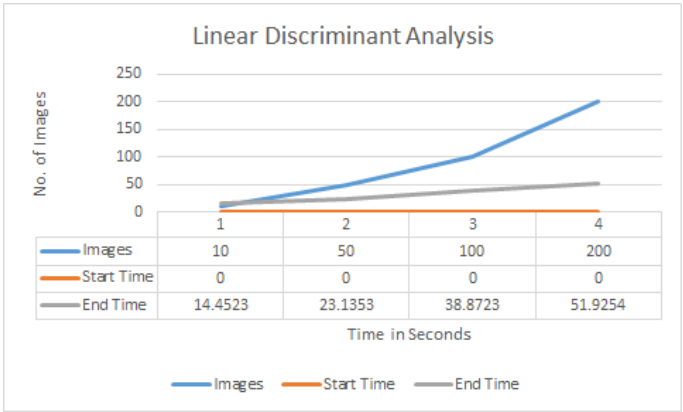


**Figure 4:** Linear Discriminant Analysis (LDA) on the images of the dataset

"LDA is a methodology employed in pattern recognition and machine learning to search out a linear combination of options that characterizes or separates two or many categories of objects or events"[15]. By applying LDA (Linear Discriminant Analysis) Figure 4, the following result was obtained that showed as the number of images was increased, the time complexity of the algorithm also increased to a more significant extent. If the test images are made two hundred,

then the time that the LDA algorithm is 51 seconds for more pictures will be equal to a minute or, in some large cases, will be more than a minute.

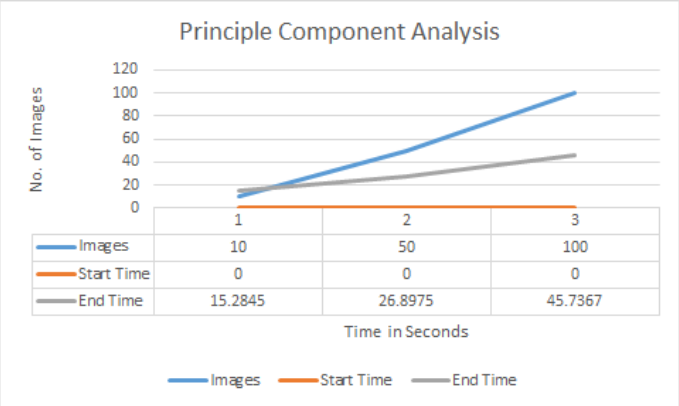


**Figure 5:** Principal Component Analysis (PCA) on the images of the dataset

"Traditional methods were dependent on features, like descriptors, combined with machine learning techniques, such as principal component analysis (PCA)"[16].Using PCA in Figure 5, we inferred that as the number of test images reached 100, the time required to run this algorithm was 46 seconds somewhat equal to a minute. Comparing this to the LDA algorithm, we can see that PCA needed more time, and as the number of images will increase, it will be a too tedious task to use this algorithm.

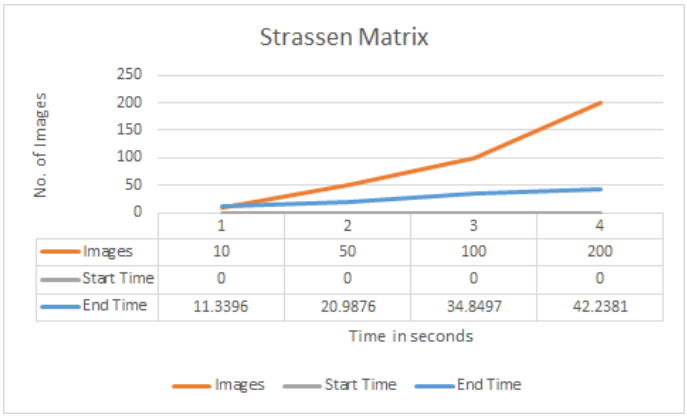


**Figure 6** Strassen's Matrix Multiplication Algorithm on the images of the dataset

By using the Strassen's Matrix multiplication algorithm that we modified stated in section II on the same dataset **Figure 6**, we can see that the time required for the hundred test images is 34 seconds, which is a notable decrease as noted. If we increase the number of copies to two hundred, then only 8 seconds of increase is present that shows that using this method for a large number of images, the growth is subsequently less.

## 8. CONCLUSION

The study's main objective is to reduce the time complexity of digital image processing from each step, starting from image reading until training the image. It was achieved by using Strassen's matrix multiplication algorithm, which resulted in significantly less time than the previous methods used, having reduced the steps of multiplication from n3. The comparative analysis plot shows that as the number of images exceeds from hundred, we have a drastic rise in the time required by LDA (Linear Discriminant Analysis) and PCA (Principal Discriminant Analysis). However, for Strassen's Algorithm, the increase is less.

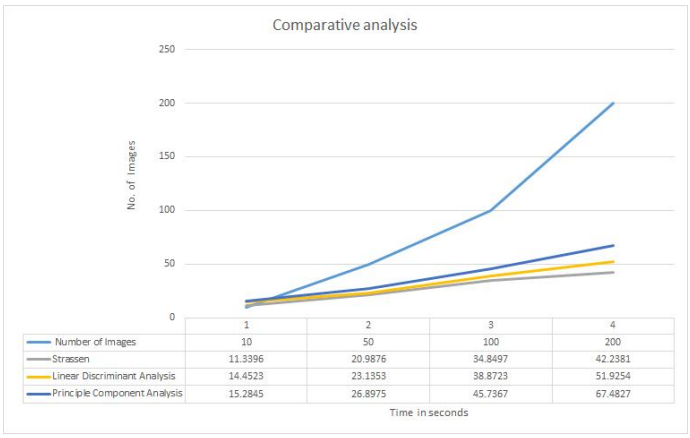


**Figure 7:** A comparative of all the algorithms to show the time required by them.

From the above analysis in Figure 7 of the algorithms, we can conclude that for a large number of images, Strassen's Algorithm is the most suitable algorithm for training image datasets as it required the least time.

## REFRENCES

[1] Gonzalez Rafael C and Woods Richard E. (Second Edition) **Digital Image Processing.** Published by Prentice Hall; 2nd edition (January 15, 2002)

[2] Cormen Thomas H., Leiserson Charleas E., Rivest Ronald L., Stein Clifford, (Second Edition) **Introduction to Algorithms.** Published by The MIT Press Cambridge, Massachusetts London, England McGraw-Hill.

[3] **Digital Image Processing** Available at: https://www.ukessays.com/essays/engineering/

[4] **Overview of modern computer vision tools** Available at: https://svitla.com/blog/overview-of-modern-computer-vision-tools

[5] R.Sathya , A.Asha , K.Muralikrishna ,Bakyalakshmi.V "**Bespoke Measurement based on Convolutional Neural Network using OpenCV**" *International Journal of Advanced Trends in Computer Science and Engineering* [IJATCSE], Vol 9, No.3, 2020, pp.2735-2748, https://doi.org/10.30534/ijatcse/2020/39932020

[6] **Multivariable polynomial matrix formultation** Available at: https://www.aut.ac.nz/research

[7] Why TensorFlow Available at https://www.charusat.ac.in/research.php

[8] **Keras layers API, Keras API reference / Keras layers API** Available at: https://keras.io/

[9] **Convolution 2D, TFLearn/Docs/Layers/Convolutional Layers** Available at:

https://www.predictiveanalyticstoday.com/top-artificial-neural-network-software/

[10] P.Pandiaraja ,G. Muralidharan , V. Vigneshwaran , A. Vijay Prasanth , S. Santhosh Sivan "**Attendance Automation by Facial Recognition Using OpenCV**" *International Journal of Advanced Trends in Computer Science and Engineering* [IJATCSE], Vol 9, No.2, 2020, pp. 925-929, https://doi.org/10.30534/ijatcse/2020/03922020

[11] Shubhangi Tirpude , Naman Vidyabhanu , Hashir Sheikh Shoeb Pathan , Zeeshan Ali syed, Shivam Singh "**Abnormal X-Ray Detection System using Convolution Neural Network"** *International Journal of Advanced Trends in Computer Science and Engineering* [IJATCSE], Vol 9, No.1, 2020, pp.828-832, https://doi.org/10.30534/ijatcse/2020/119912020

[12] Kriesel David, (2005) **A Brief Introduction to Neural Networks** Available at http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf

[13] **Softmax, Layer Activation functions** Available at: https://keras.io/api/layers/activations/#softmax-function

[14] [Dataset of Cars] Roy Prasun**, Natural Images** (August 2018) Available at: https://www.kaggle.com/prasunroy/natural-images

[15] Jay Robert B. Del Rosario **"Development of a Face Recognition System Using Deep Convolutional Neural Network in a Multi-view Vision Environment"** *International Journal of Advanced Trends in Computer Science and Engineering* [IJATCSE], Vol 8, No.3, 2019, pp.369-374 https://doi.org/10.30534/ijatcse/2019/06832019

[16] Sukhada Chokkadi , Sannidhan MS , Sudeepa K B , Abhir Bhandary **"A Study on various state of the art of the Art Face Recognition System using Deep Learning Techniques"** *International Journal of Advanced Trends in Computer Science and Engineering* [IJATCSE], Vol 8, No.4, 2019, pp.1590-1600 https://doi.org/10.30534/ijatcse/2019/84842019