iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** I **Month of publication:** January 2020

DOI: http://doi.org/10.22214/ijraset.2020.1109

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 8 Issue I, Jan 2020- Available at www.ijraset.com

Detecting Malaria from Segmented Cell Images of Thin Blood Smear Dataset using Keras from Tensorflow

Mohammed Azam Sayeed[1], Sakeena Fiza[2], Noor Ayesha[3], Mohammed Akram Sayeed[4]
[1]*Department of Computer sciences, SET Jain University Bangalore -562112*
[2]*UG Student at UC Irvine*

Abstract: This paper presents generalized methodology to be able to detect malaria from Segmented Cell images of Thin blood Smear dataset. Paper attempts to Simplify the concept of Convolution neural network and successfully applies CNN for training and validating the dataset using keras classes from the tensorflow framework.
Keywords: Keras, Tensorflow, malaria, Paratized, Uninfected, blood smear,Neurons, Perceptrons, Visual cortex, Layers of CNN, Convolution, filter, ReLU, Pooling, Fully Connected Layer, Image Data Generator, Data Preparation, Sequential, avoid overfitting, loading model .

I. IMPORTANCE OF DETECTION OF MALARIA FROM BLOOD SMEAR

"Eradicating Malaria." Science Progress defines Malaria as "a protozoan disease transmitted by the bite of infected female Anopheles mosquitoes. It is the most important of the parasitic diseases of humans, with transmissions in 107 countries affecting close to three billion people and causing one to two million deaths each year," (Berman 3). Data analysis has shown cases and deaths associated with malaria are concentrated in the least developed countries and within these countries, malaria is prominent amongst the poorest. This dire fact is evidenced in "Clinical and Epidemiological Profiles of Severe Malaria in Children from Delhi, India." Journal of Health, Population and Nutrition which states, "As per the World Malaria Report 2009, there were 1.53 million confirmed cases of malaria from India, of which 0.77 million were due to plasmodium falciparum." (Kaushik 113). A bird's-eye view of the population in rural and vulnerable countries highlights the obstacles responsible for poor management of disease, parasite resistance to drugs, and incomplete treatments.

The citizens belonging to low-income households are burdened by the economic penuries of health care and the lack of doctors accessible to them. Hence, the renewed interest in malaria research is based on the inhumane toll this disease takes on vulnerable populations. Microscopic examination of a blood film remains the gold standard for malaria diagnosis in clinical practice. Effective rapid diagnostic tests with sensitivity and specificity above 97 percent for falciparum malaria are widely available. But most are not as sensitive as an experienced microscopist and don't give any information about the density of the parasites, which is important for prognosis and treatment decisions. This serves to be disadvantageous as the diagnosis can be misinterpreted, the "Investigation and Treatment of Imported Malaria in Non-Endemic Countries." in BMJ: British Medical Journal states how this can happen, "The symptoms and signs of malaria are non-specific and overlap with many other common infections. Influenza is a common misdiagnosis, and case reports show that malaria has also been misdiagnosed as gastroenteritis, hepatitis, and lower respiratory tract infection." (Whitty 32). This paper will present an improved surveillance method with the use of modern electronic reporting systems with feedback to aid in the identification of malaria and ease the burden for microscopists in resource-constrained regions and improve diagnostic accuracy.

Dr. David Nabarro, the first executive director for the Roll Back Malaria initiative at the World Health Organization states, "Malaria is taking costly bites out of Africa…. It is feasting on the health and development of African children and it is draining the life out of African economies". The parasitic disease carries with a tiresome economic burden on workers and employers, as it reduces their productivity and efficiency massively due to sickness and loss of labor. The economic cost of malaria is high and the investment in malaria control remains one of the only ways to combat it. Here, we discuss an automated system for efficient and accurate identification of malaria using innovative technology and cost-effective measures. The use of a mobile application on a smartphone attached to a conventional light microscope could serve as the answer to eradicating malaria in vulnerable populations that lack access to proper care and reduce the economic burden caused by this crippling disease. Casey W. Pirnstill and Gerard L. Coté in their paper Malaria Diagnosis Using a Mobile Phone Polarized Microscope highlight the effectiveness of the

implementation of this portable device, "In general, mobile phones offer an ideal platform for creating a field-based, modular polarized microscope. Currently, over 6 billion cell-phone subscriptions exist worldwide (accounting for approximately 75% of the world having access to mobile phone networks) with the vast majority of these users (~5 billion) located in developing countries. Utilizing the existing mobile infrastructure allows for a significant reduction in cost and size of mobile-based designs as compared to traditional microscopes" [2]. The use of a prototype application on blood smear images from conventional light microscope could serve to be a helpful tool for medical professionals as a low-cost and high-efficiency diagnosis method, particularly in remote-areas with low-resources around the world. This could serve as the answer to eradicating malaria in vulnerable populations that lack access to proper care and reduce the economic burden caused by this crippling disease. Rapid detection of malaria in the lab testing phase itself can speed up the process and saves the lives of malaria-infected patients and dedicated medical expertise health professionals can start specific targeted treatments for malaria.

II. CONCEPT OF CONVOLUTED NEURAL NETWORK

A. *Intuition of Using Convoluted Neural Network*

1) *Biological Inspiration of Neural Network:* [6] states that "The human brain contains approximately 100 billion neurons. To interpret this number, let us assume we have 100 billion one dollar bills, where each bill is only 0.11 mm thick. If we stack all these one dollar bills on top of each other, the resulting stack will be 10922.0 km high. This illustrates the scale and magnitude of the human brain. A biological neuron is a nerve cell which processes information [Jain et al., 1996]. Each neuron is surrounded by a membrane and has a nucleus which contains genes. It has specialized projections which manage the input and output to the nerve cell. These projections are termed dendrites and axons. We describe these and other key aspects of the biological neuron below.

a) *Dendrites:* Dendrites are fibers which act as receptive lines and bring information (activations) to the cell body from other neurons. They are the inputs of the neuron.

b) *Axons:* Axons are fibers which act as transmission lines and take information away from the cell body to other neurons. They act as outputs of the neuron.

c) *Cell Body:* The cell body (also called the soma) receives the incoming information through dendrites, processes it and sends it, to other neurons via axons. Synapses:The specialized connections between axons and dendrites which allow the communication of signals are called synapses. The communication takes place by an electro-chemical 40

d) *Neural Networks Basics:* Process where the neurotransmitters (chemicals) are released at the synapse and are diffused across the synaptic gap to transmit information. There is a total of approximately 1 quadrillion (1015) synapses in the human brain [Changeux and Ricoeur, 2002]. Connections: Neurons are densely inter-connected with each other. On average, each neuron receives inputs from approximately 105 synapses.

e) *Neuron Firing:* A neuron receives signals from connected neurons via dendrites. The cell body sums up the received signals and the neuron fires if the combined input signal exceeds a threshold. By neuron firing, we mean that it generates an output which is sent out through axons. If the combined input is below the threshold, no response signal is generated by the neuron (i.e., the neuron does not fire). The thresholding function which decides whether a neuron fires or not is called activation function."

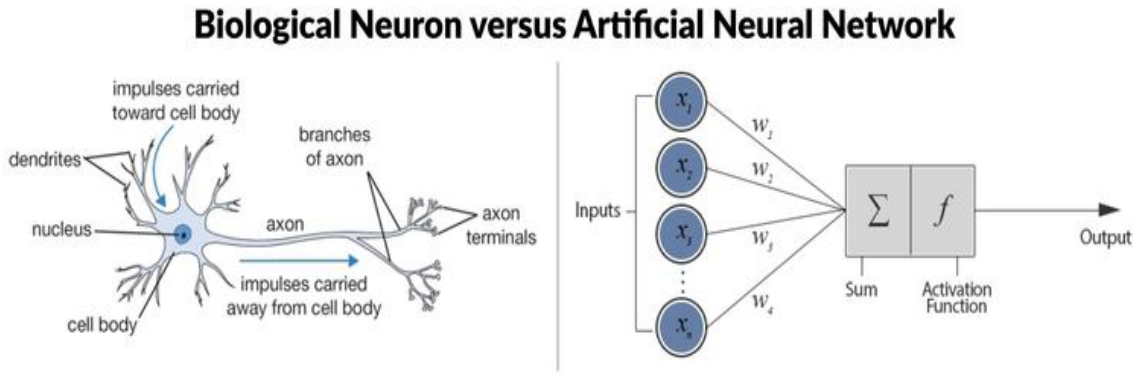


Fig. 1 : [6] A biological neuron (left) and a computational model (right) which is used to develop artificial neural networks.

2) *Perceptron:* basic computational model of Neuron The most simple neural network is the "perceptron", which, in its simplest form, consists of a single neuron. There are six components to artificial neurons. From left to right of Fig 1(right), these are:

a) Input node is associated with a numerical value, which can be any real number.

b) Connection that departs from the input node has a weight associated with it, and this can also be any real number. given as y=f(w1∗x1+w2∗x2+...wD∗xD). This result will be the input for a transfer or activation function. In the simplest but trivial case, this transfer function would be an identity function, f(x)=x or y=x. In this case, x is the weighted sum of the input nodes and the connections. However, just like a biological neuron only fires when a certain threshold is exceeded, the artificial neuron will also only fire when the sum of the inputs exceeds a threshold like the sigmoid function. The weights are adjusted based on the forward propagation technique elaborated in [6].

c) Output node is associated with the function (such as the sigmoid function) of the weighted sum of the input nodes.

d) Bias is an additional parameter in perceptron which you can consider as the weight associated with an additional input node that is permanently set to 1. The bias value is critical because it allows you to shift the activation function to the left or right, which can make a determine the success of your learning. Bias are adjusted based on backward propagation technique elaborated in [6].

3) *Reading Image by a Computer:* The machine reads the image not as collectively but as corresponding pixel values of each separate channel represented as matrixes of 3 channels. The RGB color image is separated into its corresponding 3 channels Red, green and blue component. Each of the components has own respective pixel values which are read by the machine. As per Fig 2. The total size of the image is 260x194x3 pixels where there are 260 rows and 194 columns and 3 individual channels which collectively form the RGB color image.

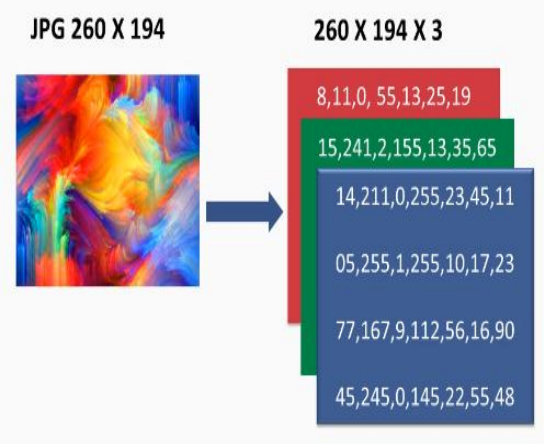


Fig. 2 Image read by computer

Fully Connected network build by multi layer perceptrons is not a feasible approach for Image classification because as per Fig 2 The Fully connected multi perceptron model will have 260*194*132= 151,320 number of weights in the first hidden layer itself. As depicted in Fig 3. Thus directly using fully connected ANN will result in managing large number of parameters and neurons which may result in overfitting problem.

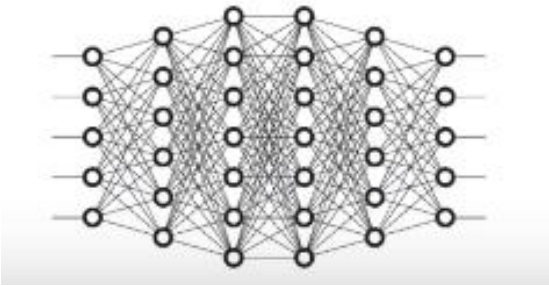


Fig. 3 Fully Connected Neural Network constructed

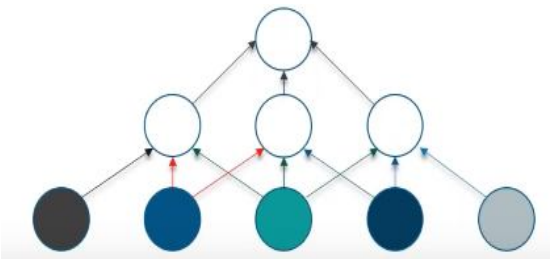


Fig. 4 Convoluted neural network

In case of Convoluted Neural Network the neurons in a layer will be connected to a small region of the layer before it, instead of all Neurons in a fully connected manner which will drastically enhance image segmentation of images.

4) Biological Inspiration Of Convolution Neural Network

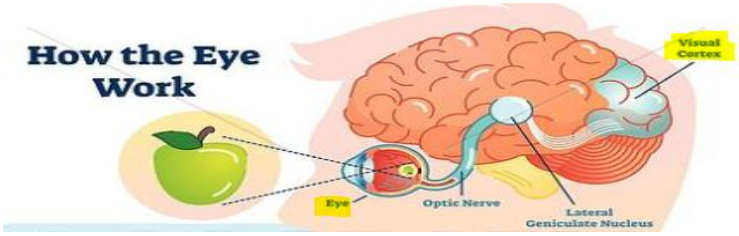


Fig. 5 Visual cortex

CNN is a type of feed forward artificial neural network in which the connectivity pattern between its neuron is inspired by the organization of visual cortex. The visual cortex has a small region of cells that are sensitive to specific regions of the visual field. Some individual neural cells are fired to the presence of edges/orientations and colors. Therefore different neurons will be fired in case of viewing apple than banana for an instance.

[6]CNNs are one of the most popular categories of neural networks, especially for high-dimensional data (e.g., images and videos). CNNs operate in a way that is very similar to standard neural networks. The CNN learns to map a given image to its corresponding category by detecting a number of abstract feature representations, ranging from simple to more complex ones. These discriminative features are then used within the network to predict the correct category of an input image.

Difference of ANN pipeline to CNN is the hierarchy of useful feature representations and its integration of the classification and feature extraction stages in a single pipeline which is trainable in an end-to-end manner. This reduces the need for manual design and expert human intervention.

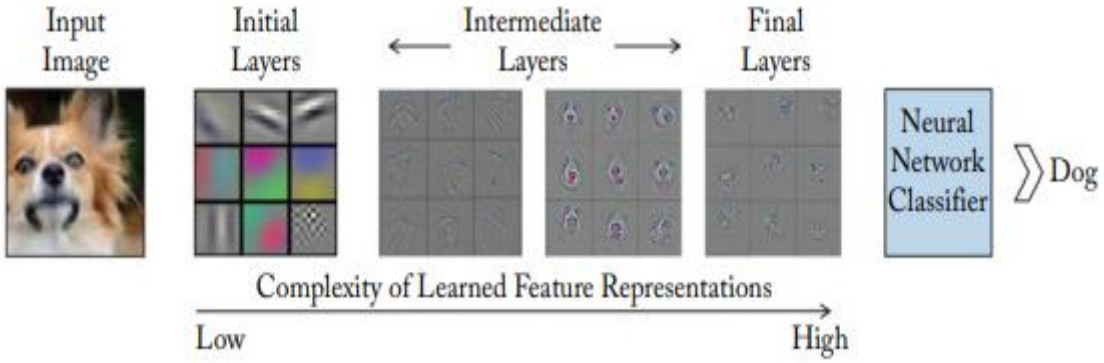


Fig. 6 A CNN learns low-level features in the initial layers, followed by more complex intermediate and high-level feature representations which are used for a classification task. The feature visualizations are adopted from Zeiler and Fergus [2014].

B. Layers of CNN

CNN compares the images piece by piece. The pieces that it looks for is called features. By finding rough feature matches, in roughly the same position in two images, CNN is a lot better at seeing similarity than whole image matching schemes.

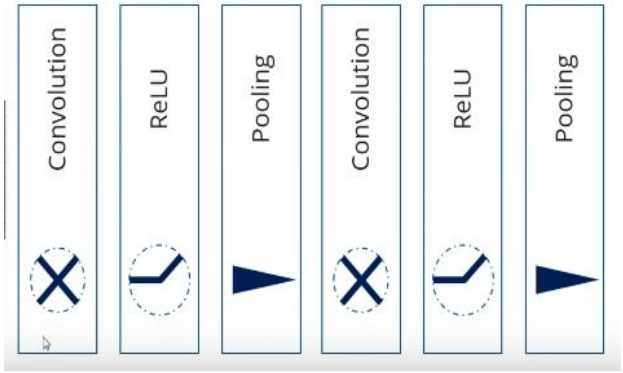


Fig. 7 Stacking multiple layers in CNN

The CNN have the following layers –

1) *Convolution:* A convolutional layer is the most important component of a CNN. It comprises a set of filters (also called convolutional kernels) which are convolved with a given input to generate an output feature map.

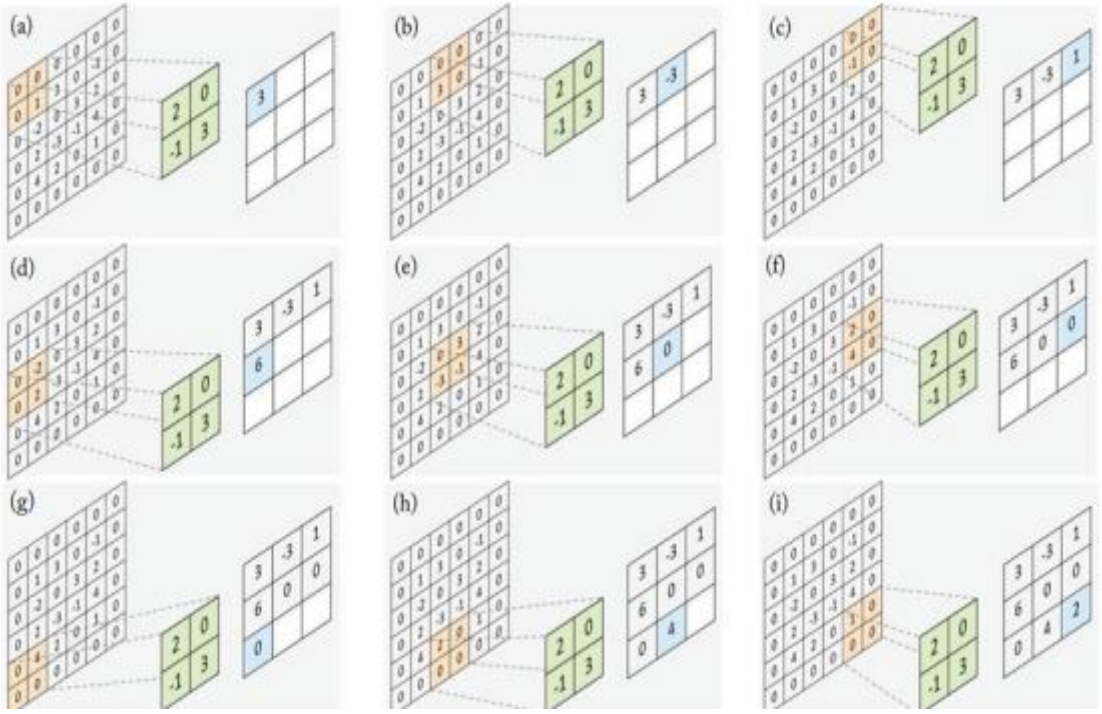


Fig. 8 Example from [6] applying filter in convolution layer to image pixels

The operation of a convolution layer is illustrated in the figure 8 show the computations performed at each step, as the filter is slided onto the input feature map to compute the corresponding value in the output feature map. The 2x2 filter (shown in green) is multiplied with the same sized region (shown in orange) within a 4 4 input feature map and the resulting mean values are corresponding entry (shown in blue) in the output feature map at each convolution step.

2) *ReLU Layer:* In this layer, it removes every negative values from the filtered image and replace it with zeroes. This layer ensures the values don't sum up to zero. ReLU transform function only activates a node only if the input is above a certain threshold. If the input is below zero, the ouput is zero and when the input rises above a certain threshold it has a linear relationship with dependent variables.

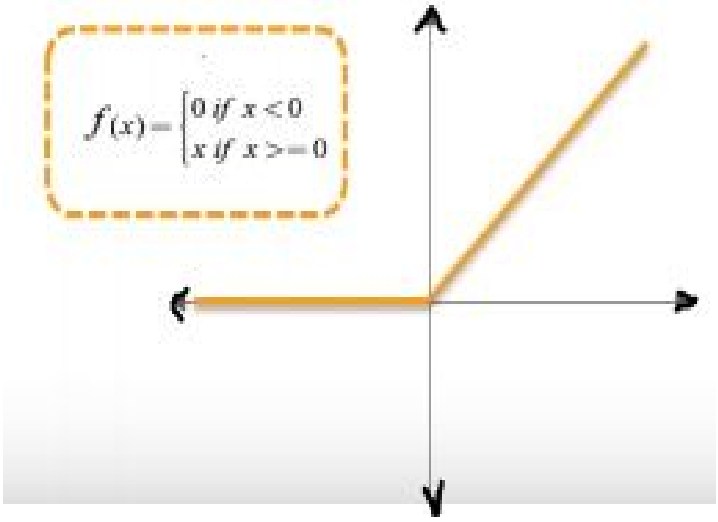


Fig. 9 reLU transform Function

3) *Pooling:* In this Layer we shrink the image stack into a smaller size. A pooling layer operates on blocks of the input feature map and combines the feature activations. This combination operation is defined by a pooling function such as the average or the max function. Similar to the convolution layer, we need to specify the size of the pooled region and the stride. Figure 4.8 shows the max pooling operation, where the maximum activation is chosen from the selected block of values. This window is slided across the input feature maps with a step size defined by the stride (a in the case of Fig. 10).

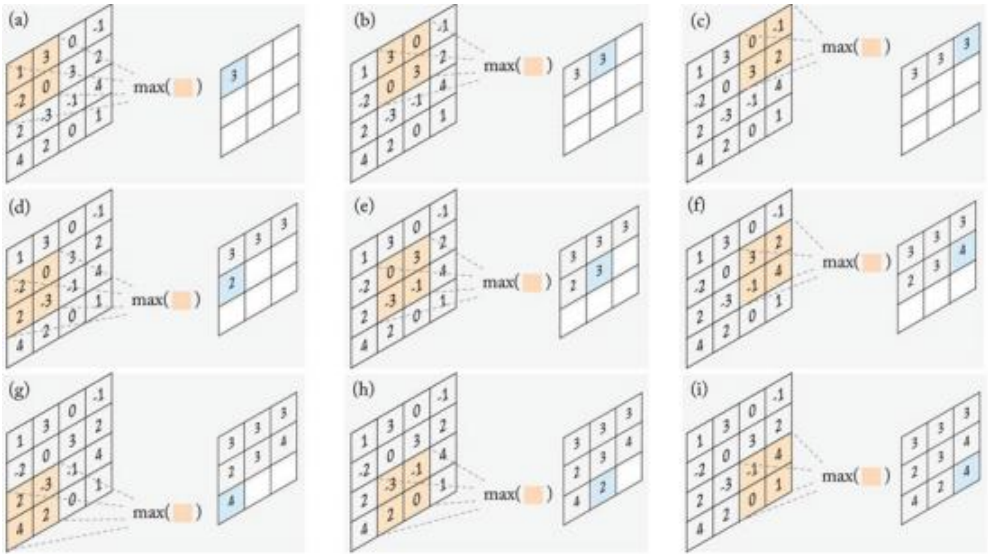


Fig. 10 Pooling with filter and max function

4) *Fully Connected Layer:* This is the final layer where the actual classification happens, here we take our filtered and shrinked images and put them into a single list. This single list will be used by the machine to understand similarities between images and eventually classifies the images into labels based on similarities.

III.PROPOSED METHODOLOGY

The Generalized Methodology can be briefly depicted as Fig 11. We are using tf.keras, is TensorFlow's implementation of the Keras API specification. This is a high-level API to build and train models that includes first-class support for TensorFlow-specific functionality, such as eager execution, tf.data pipelines, and Estimators. tf.keras makes TensorFlow easier to use without sacrificing flexibility and performance.

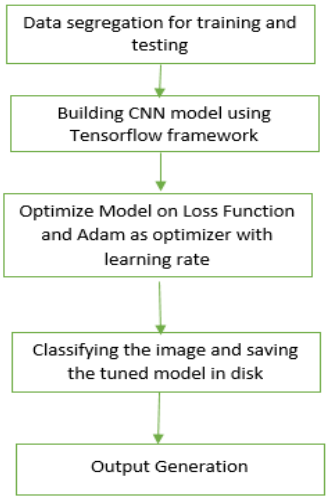


Fig. 11 Generalized Methodology for image classification

TensorFlow framework can be installed using the below command in anaconda command prompt
>> $ conda install tensorflow

A. Data Set

https://lhncbc.nlm.nih.gov/publication/pub9932

[7]"This page hosts a repository of segmented cells from the thin blood smear slide images from the Malaria Screener research activity. To reduce the burden for microscopists in resource-constrained regions and improve diagnostic accuracy, researchers at the Lister Hill National Center for Biomedical Communications (LHNCBC), part of National Library of Medicine (NLM), have developed a mobile application that runs on a standard Android smartphone attached to a conventional light microscope. Giemsa-stained thin blood smear slides from 150 P. falciparum-infected and 50 healthy patients were collected and photographed at Chittagong Medical College Hospital, Bangladesh. The smartphone's built-in camera acquired images of slides for each microscopic field of view. The images were manually annotated by an expert slide reader at the Mahidol-Oxford Tropical Medicine Research Unit in Bangkok, Thailand. The de-identified images and annotations are archived at NLM (IRB#12972)."

The dataset repositiory is open sourced for other researchers in field of health and malaria related, [7]The dataset contains a total of 27,558 cell images with equal instances of parasitized and uninfected cells. An instance of how the patient-ID is encoded into the cell name is shown herewith: "P1" denotes the patient-ID for the cell labeled "C33P1thinF_IMG_20150619_114756a_cell_179.png". We have also included the CSV files containing the Patient-ID to cell mappings for the parasitized/Malaria infected and uninfected classes. The CSV file for the parasitized class contains 151 patient-ID entries. The slide images for the parasitized patient-ID "C47P8thinOriginal" are read from two different microscope models (Olympus and Motif). The CSV file for the uninfected class contains 201 entries since the normal cells from the infected patients' slides also make it to the normal cell category (151+50 = 201).

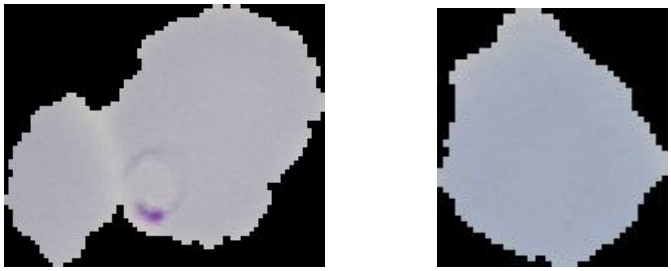


Fig. 12 Sample Images Malaria infected(left) and Uninfected(right)

B. Implementation of CNN using Tensorflow framework

1) Importing Packages: The Import Tensorflow and the Keras classes needed to construct our model.

```
In [2]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
        from tensorflow.keras.preprocessing.image import ImageDataGenerator

        import os
        import numpy as np
        import matplotlib.pyplot as plt
```

Fig. 13 Importing Keras classes and Tensorflow

2) Data Segregation of Images in file System: The open source repositiory has 12,279 images for parasitized patients(malaria infected blood smear images) and 12,279 images for uninfected patients. We have arranged the file structure containing the dataset as Fig 11 which will be used for label encoding or setting labels for training and validation phase of model. Also we are using 2000 images for training and 1000 for validation randomly selected from original dataset. According to the file structure we will have two labels Parasitized and Uninfected.

```
The dataset has the following directory structure:

Parasitized_and_Uninfected_filtered
|__ train
     |_______ Parasitized
              |_____1000 images
     |_______ Uninfected
              |_____1000 images
|__ validation
     |_______ Parasitized
              |___500 images
     |_______ Uninfected
              |___500 images
```

Fig. 14 File Structure using for label encoding

```
In [7]: PATH = os.path.join('C:\\Users\\Malaria dataset', 'Parasitized_and_Uninfected_filtered')
        train_dir = os.path.join(PATH, 'train')
        validation_dir = os.path.join(PATH, 'validation')
        train_Parasitized_dir = os.path.join(train_dir, 'Parasitized')  # directory with our training Parasitized pictures
        train_Uninfected_dir = os.path.join(train_dir, 'Uninfected')  # directory with our training Uninfected pictures
        validation_Parasitized_dir = os.path.join(validation_dir, 'Parasitized')  # directory with our validation Parasitized pictures
        validation_Uninfected_dir = os.path.join(validation_dir, 'Uninfected')  # directory with our validation Uninfected pictures
```

Fig. 15 segregation of images into corresponding dir

3) *Data Preparation using Image Data Generator Class*: We are considering values for Data preparation in step 4 as: batch_size = 128, epochs = 15, IMG_HEIGHT = 150, IMG_WIDTH = 150. We have to Format the images into appropriately pre-processed floating point tensors before feeding to the network:

a) Read images from the disk.

b) Decode contents of these images and convert it into proper grid format as per their RGB content.

c) Convert them into floating point tensors.

d) Rescale the tensors from values between 0 and 255 to values between 0 and 1, as neural networks prefer to deal with small input values.

All these tasks can be done with the ImageDataGenerator class provided by tf.keras. It can read images from disk and preprocess them into proper tensors. It will also set up generators that convert these images into batches of tensors—helpful when training the network. Additionally we apply horizontal flip, zoom augmentation and rotation to avoid overfitting problem in model.

```
In [36]: image_gen_train = ImageDataGenerator(
                     rescale=1./255,
                     rotation_range=45,
                     width_shift_range=.15,
                     height_shift_range=.15,
                     horizontal_flip=True,
                     zoom_range=0.1
                     )

In [38]: train_data_gen = image_gen_train.flow_from_directory(batch_size=batch_size,
                                                    directory=train_dir,
                                                    shuffle=True,
                                                    target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                    class_mode='binary')

         Found 2000 images belonging to 2 classes.

In [40]: image_gen_val = ImageDataGenerator(rescale=1./255)

In [41]: val_data_gen = image_gen_val.flow_from_directory(batch_size=batch_size,
                                                directory=validation_dir,
                                                target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                class_mode='binary')

         Found 1000 images belonging to 2 classes.
```

Fig. 16 data preparations

Fig. 17 Datapreparation on images using ImageDataGenerator class

4) *Creating the Model:* The model consists of three convolution blocks with a max pool layer in each of them. There's a fully connected layer with 512 units on top of it that is activated by a reLU activation function. The model outputs class probabilities based on binary classification by the sigmoid activation function. We are using Keras, to assemble layers to build models. A model is (usually) a graph of layers. The most common type of model is a stack of layers: the tf.keras.Sequential model. We choose the ADAM optimizer and binary cross entropy loss function. To view training and validation accuracy for each training epoch, pass the metrics argument. Additionally we are applying dropout to randomly set 20% of the neurons to zero during each training epoch. This helps to avoid overfitting on the training dataset.

```
In [42]: model_new = Sequential([
             Conv2D(16, 3, padding='same', activation='relu',
                    input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
             MaxPooling2D(),
             Dropout(0.2),
             Conv2D(32, 3, padding='same', activation='relu'),
             MaxPooling2D(),
             Conv2D(64, 3, padding='same', activation='relu'),
             MaxPooling2D(),
             Dropout(0.2),
             Flatten(),
             Dense(512, activation='relu'),
             Dense(1, activation='sigmoid')
         ])

         model_new.compile(optimizer='adam',
                           loss='binary_crossentropy',
                           metrics=['accuracy'])
```

Fig. 18 Sample Sequential Model used for Image classification

5) *Training and Validating the Model:* Use the fit_generator method of the ImageDataGenerator class to train the network. As in Fig 12 we have achieved an accuracy of accuracy: 0.8168 , Note the accuracy can further be improvised by parameter tuning and using larger number of images in train dataset from the original repository

```
In [44]: history = model_new.fit_generator(
             train_data_gen,
             steps_per_epoch=total_train // batch_size,
             epochs=epochs,
             validation_data=val_data_gen,
             validation_steps=total_val // batch_size
         )

WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 15 steps, validate for 7 steps
Epoch 1/15
15/15 [==============================] - 86s 6s/step - loss: 0.8753 - accuracy: 0.5016 - val_loss: 0.7017 - val_accuracy: 0.500
0
Epoch 2/15
15/15 [==============================] - 83s 6s/step - loss: 0.6587 - accuracy: 0.5230 - val_loss: 0.7090 - val_accuracy: 0.463
2
Epoch 3/15
15/15 [==============================] - 70s 5s/step - loss: 0.6059 - accuracy: 0.6480 - val_loss: 0.8279 - val_accuracy: 0.415
2
Epoch 4/15
15/15 [==============================] - 69s 5s/step - loss: 0.5867 - accuracy: 0.6571 - val_loss: 0.8344 - val_accuracy: 0.354
9
Epoch 5/15
15/15 [==============================] - 72s 5s/step - loss: 0.5712 - accuracy: 0.6859 - val_loss: 0.7804 - val_accuracy: 0.462
1
Epoch 6/15
15/15 [==============================] - 84s 6s/step - loss: 0.5690 - accuracy: 0.6832 - val_loss: 0.8049 - val_accuracy: 0.387
3
Epoch 7/15
15/15 [==============================] - 93s 6s/step - loss: 0.5532 - accuracy: 0.6987 - val_loss: 0.7898 - val_accuracy: 0.426
3
Epoch 8/15
15/15 [==============================] - 94s 6s/step - loss: 0.5276 - accuracy: 0.7174 - val_loss: 0.8269 - val_accuracy: 0.466
5
Epoch 9/15
15/15 [==============================] - 114s 8s/step - loss: 0.5200 - accuracy: 0.7329 - val_loss: 0.8604 - val_accuracy: 0.43
08
Epoch 10/15
15/15 [==============================] - 78s 5s/step - loss: 0.5225 - accuracy: 0.7217 - val_loss: 0.8183 - val_accuracy: 0.465
4
Epoch 11/15
15/15 [==============================] - 79s 5s/step - loss: 0.5106 - accuracy: 0.7458 - val_loss: 0.8082 - val_accuracy: 0.507
8
Epoch 12/15
15/15 [==============================] - 75s 5s/step - loss: 0.4777 - accuracy: 0.7847 - val_loss: 0.8720 - val_accuracy: 0.523
4
Epoch 13/15
15/15 [==============================] - 79s 5s/step - loss: 0.4582 - accuracy: 0.7858 - val_loss: 0.8720 - val_accuracy: 0.511
2
Epoch 14/15
15/15 [==============================] - 83s 6s/step - loss: 0.4499 - accuracy: 0.7997 - val_loss: 0.8261 - val_accuracy: 0.535
7
Epoch 15/15
15/15 [==============================] - 113s 8s/step - loss: 0.4208 - accuracy: 0.8168 - val_loss: 0.8325 - val_accuracy: 0.52
34
```

Fig. 19 Training the model for 15 steps and validate for 7 steps

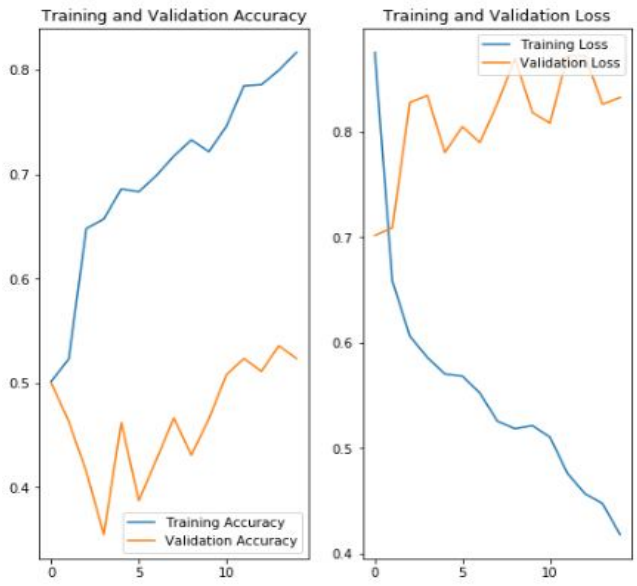


Fig. 20 Graphical representation of validation step

6) Saving The Model In Disk For Loading When Required

```
In [43]: # Load and evaluate a saved model
from numpy import loadtxt
from tensorflow.keras.models import load_model
model = load_model('model.h5')
# summarize model.
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 150, 150, 16)      448
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 75, 75, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 75, 75, 16)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 75, 75, 32)        4640
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 37, 37, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 37, 37, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 18, 18, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 18, 18, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 20736)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               10617344
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 10,641,441
Trainable params: 10,641,441
Non-trainable params: 0
_________________________________________________________________
```

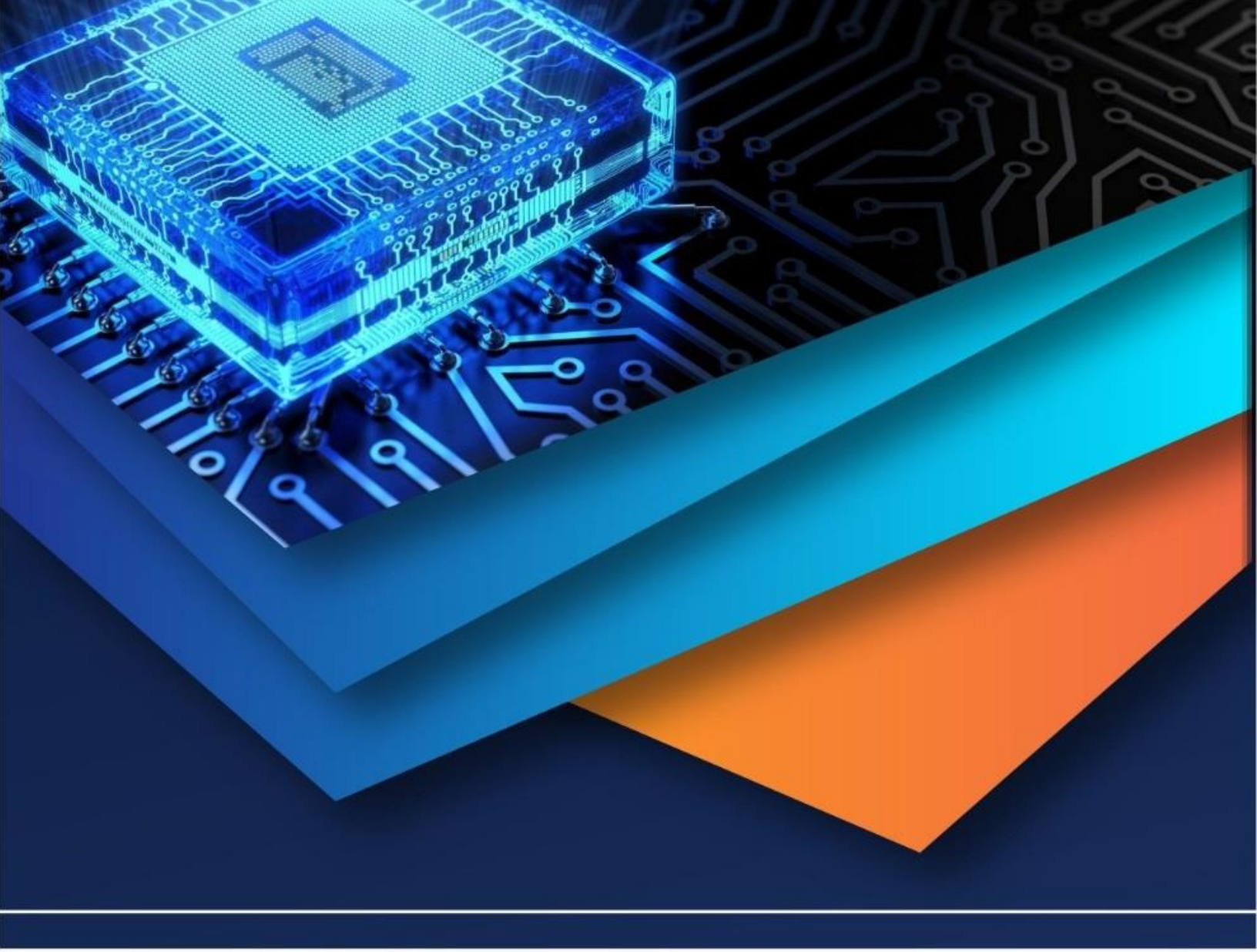
Fig. 21 model parameters are saved for future loading

IV. APPLICATIONS OF METHODOLOGY

The proposed system is capable of Rapid detection of malaria in blood smear images This could serve as the answer to eradicating malaria in vulnerable populations that lack access to proper care and reduce the economic burden caused by this crippling disease.

REFERENCES

[1] BREMAN, JOEL G. "Eradicating Malaria." Science Progress (1933-), vol. 92, no. 1, 2009, pp. 1–38. JSTOR, www.jstor.org/stable/43423234. Accessed 21 Jan. 2020.
[2] Whitty, Christopher J M, et al. "Investigation and Treatment of Imported Malaria in Non-Endemic Countries." BMJ: British Medical Journal, vol. 346, no. 7909, 2013, pp. 31–34. JSTOR, www.jstor.org/stable/23494742. Accessed 21 Jan. 2020.
[3] Ribera, Joan Muela, and Susanna Hausmann-Muela. "The Straw That Breaks the Camel's Back Redirecting Health-Seeking Behavior Studies on Malaria and Vulnerability." Medical Anthropology Quarterly, vol. 25, no. 1, 2011, pp. 103–121., www.jstor.org/stable/23012066. Accessed 21 Jan. 2020.
[4] Kaushik, Jaya Shankar, et al. "Clinical and Epidemiological Profiles of Severe Malaria in Children from Delhi, India." Journal of Health, Population and Nutrition, vol. 30, no. 1, 2012, pp. 113–116. JSTOR, www.jstor.org/stable/23500113. Accessed 21 Jan. 2020.
[5] Packard, Randall M. "'Roll Back Malaria, Roll in Development'? Reassessing the Economic Burden of Malaria." Population and Development Review, vol. 35, no. 1, 2009, pp. 53–87. JSTOR, www.jstor.org/stable/25487642. Accessed 22 Jan. 2020.
[6] Khan, Salman & Rahmani, Hossein & Shah, Syed & Bennamoun, Mohammed. (2018). A Guide to Convolutional Neural Networks for Computer Vision. Synthesis Lectures on Computer Vision. 8. 1-207. 10.2200/S00822ED1V01Y201712COV015.
[7] Rajaraman S, Antani SK, Poostchi M, Silamut K, Hossain MA, Maude, RJ, Jaeger S, Thoma GR. (2018) Pre-trained convolutional neural networks as feature extractors toward improved Malaria parasite detection in thin blood smear images. PeerJ6:e4568 https://doi.org/10.7717/peerj.4568



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY