Diagnosis of Blood Cells Using Deep Learning

Ahmed J. Khalil and Samy S. Abu-Naser

Deprtment of Information Technology,
Faculty of Engineering and Information Technology,
Al-Azhar University, Gaza, Palestine

***Abstract**: In computer science, Artificial Intelligence (AI), sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals. Computer science defines AI research as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. Deep Learning is a new field of research. One of the branches of Artificial Intelligence Science deals with the creation of theories and algorithms that allow the machine to learn by simulating neurons in the human body. Most in-depth learning research focuses on finding high-level methods. The strippers analyze a large data set using linear and nonlinear transformations. The method of deep learning is used in the detection of several diseases including blood cell diseases and their classification using the radiography of blood cells to help decision makers to know the type of blood cell and its associated diseases and the results will be presented in detail and discussed. This thesis is using python language and deep learning to detect blood cell diseases and their classifications. The proposed deep learning model was trained, validated and the tested. The accuracy of proposed model was 98.00%*

Keywords: Artificial Intelligence, Deep Learning, classification, blood cells

1.1 Introduction

Machine learning is a branch of artificial intelligence (AI) that provides the ability to learn for computers. Interested in designing and developing algorithms and technologies that allow the computer to have the property of "learning". There are two levels of learning: inductive and deductive. Inductive learning derives general rules from big data. Machine science interferes with computer statistics. Mathematical improvement, which focuses on choosing the best alternative among the many alternatives available, provides a lot of tools, theories and applications of machine learning.

Types of problems and tasks for machine learning:

1- Supervised learning is to give examples of inputs and outputs desired by the teacher. The goal is to connect inputs to outputs.
2- Unsupervised learning: The learning algorithm is left to rely on itself in exploring its input structure. The goal is to discover hidden patterns in the data.
3- Reinforcement learning: Program interaction with a dynamic environment. The goal is to achieve a certain goal without a teacher predicting even approaching that goal.

Machine learning examples:

- Use of the decision tree.
- Face Recognition in Facebook
- Google car, self-driving.

Machine learning is a powerful tool in the years to come, helping us solve most of our pressing problems.

1.2 Deep Learning

Deep Learning is a subset of machine learning where artificial neural networks learn algorithms inspired by the synthesis of the human brain itself from large amounts of data. Similar to how we learned from experience, the deep learning algorithm will perform a task frequently each time it is slightly modified to improve the result[1].

We refer to “deep learning” because neural networks have multiple (deep) layers that enable learning. Any problem that requires “thinking” is one that deep learning can learn to solve.

The amount of data we produce every day is staggering - currently estimated at 2.6 billion bytes - a resource that makes deep learning possible.

Because deep learning algorithms require a lot of data to learn from, this increase in data creation is one reason why deep learning capabilities have increased in recent years.

In addition to generating more data, deep learning algorithms take advantage of the most powerful computing power currently available as well as the spread of AI. It harnessed AI as a service for smaller organizations and provided access to AI technology and the identification of AI algorithms required for deep learning without significant initial investment[2].

Deep learning allows machines to solve complex problems even when using a highly diverse, disorganized and interdependent data set. The more algorithms you learn, the better they perform.

1.3 Blood Cells

A blood cell is any cell that is formed during the process of hematopoiesis and is found mainly in the blood. Blood cells are divided into: Red blood cell, white blood cell, blood platelet [3] .

Together, these three types make up 45% of the volume of blood tissue (55% are formed by [blood plasma], the liquid part of the blood).

The volumetric ratio of red blood cells to the total volume of blood (the volume of accumulated erythrocytes) is 45% in males and 40% in females, and is measured by centrifuge or flow cytometry.

Hemoglobin is the main component of red blood cells, a protein that contains iron, and facilitates the transport of oxygen from the lungs to tissues, and carbon dioxide from tissues to the lungs.

Minor blood circulation (pulmonary circulation): is the flow of blood from the heart to the lungs, and then back to the heart[4].

1.4 Problem Statement

Each year more than half a million people develop cancer or leukemia or are born with life-threatening blood disorders such as thalassemia or sickle cell disease. As a group, they represent the most common Non-Communicable Diseases (NCD) in many emerging countries. Most are curable but only a minority of people can get proper care.

Our goal is to ensure a deep learning system using Python that is capable of providing information on the diagnosis of blood cell types to help provide reliable treatment for people with blood disorders.

In this study, a model will be trained using the blood cells dataset to help doctors detect blood cell diseases.

Applied model will use deep learning techniques to increase accuracy and efficiency in data analysis.

1.5 Objectives of the Thesis

The objectives of this thesis are:

- ❖ **Main objective: The implementation of a model to be used to detect** and classify blood cells (Consists of 3 types).
- ❖ **Specific objectives:**

- Increase accuracy in blood cells screening.
- Reduce the cost of screening.
- Use deep learning techniques to detect blood cells.

2.1 Blood Cells Diseases

A blood cell disorder is a condition in which there’s a problem with your red blood cells, white blood cells, or the smaller circulating cells called platelets, which are critical for clot formation.

All three cell types form in the bone marrow, which is the soft tissue inside your bones. Red blood cells transport oxygen to your body’s organs and tissues. White blood cells help your body fight infections. Platelets help your blood to clot. Blood cell disorders impair the formation and function of one or more of these types of blood cells. [1]

Symptoms will vary depending on the type of blood cell disorder. Common symptoms of red blood cell disorders are: fatigue, shortness of breath, trouble concentrating from lack of, oxygenated blood in the brain, muscle weakness a fast heartbeat

Common symptoms of white blood cell disorders are: chronic infections, fatigue, unexplained weight loss, malaise, or a general feeling of being unwell [2].

Common symptoms of platelet disorders are: cuts or sores that don’t heal or are slow to heal, blood that doesn’t clot after an injury or cut, skin that bruises easily, unexplained nosebleeds or bleeding from the gums [3]

There are many types of blood cell disorders that can greatly affect your overall health.

2.2 Convolutional Neural Network

The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers [2].

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.

The final convolution, in turn, often involves backpropagation in order to more accurately weight the end product. Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a sliding dot product or cross-correlation.

This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point [2].

2.2.1 Convolutional Layer

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map [3] .

2.2.2 Pooling Layer

Pooling is basically “downscaling” the image obtained from the previous layers. It can be compared to shrinking an image to reduce its pixel density.

The most popular kind of pooling used is Max Pooling. Suppose you intend to pool by a ratio of 2. This means that the height and width of your image will be half of what it originally was. So you need to compress every 4 pixels (a 2x2 grid) and map it to a new single pixel without loss of “important” data from the missing pixels. Max Pooling is done by taking the largest value of those 4 pixels. So, one new pixel represents 4 old pixels by using the largest value of those 4 pixels. This is done for every group of 4 pixels throughout the whole image [4].

2.2.3 Rectified Linear Unit Layer

Rectified Linear Unit (ReLU) is a type of activation function. Mathematically, it is defined as $y = \max(0, x)$. ... ReLU is the most commonly used activation function in neural networks, especially in CNNs. If you are unsure what activation function to use in your network, ReLU is usually a good first choice [15] .

2.2.4 Dropout Layer

Dropout is a regularization technique patented by Google for reducing overfitting in neural networks by preventing complex co-adaptations on training data.

It is a very efficient way of performing model averaging with neural networks.

The term "dropout" refers to dropping out units (both hidden and visible) in a neural network [5].

2.2.5 Batch Normalization Layer

A batch normalization layer normalizes each input channel across a mini-batch. To speed up training of convolutional neural networks and reduce the sensitivity to network initialization, use batch normalization layers between convolutional layers and nonlinearities, such as ReLU layers.

The layer first normalizes the activations of each channel by subtracting the mini-batch mean and dividing by the mini-batch standard deviation. Then, the layer shifts the input by a learnable offset β and scales it by a learnable scale factor γ.[6]

2.2.6 Fully Connected Layer

Fully connected layers are an essential component of Convolutional Neural Networks (CNNs), which have been proven very successful in recognizing and classifying images for computer vision. The CNN process begins with convolution and pooling, breaking down the image into features, and analyzing them independently.
The result of this process feeds into a fully connected neural network structure that drives the final classification decision.[7]

2.2.7 Softmax

Multi-Class Neural Networks: Softmax. ... Softmax extends this idea into a multi class world.
That is Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. This additional constraint helps training converge more quickly than it otherwise [16]

2.2.8 Backpropagation

Backpropagation algorithms are a family of methods used to efficiently train artificial neural networks (ANNs) following a gradient descent approach that exploits the chain rule. The main feature of backpropagation is its iterative, recursive and efficient method for calculating the weights updates to improve the network until it is able to perform the task for which it is being trained. It is closely related to the Gauss–Newton algorithm.
Backpropagation requires the derivatives of activation functions to be known at network design time.
Automatic differentiation is a technique that can automatically and analytically provide the derivatives to the training algorithm.
In the context of learning, backpropagation is commonly used by the gradient descent optimization algorithm to adjust the weight of neurons by calculating the gradient of the loss function; backpropagation computes the gradient(s), whereas (stochastic) gradient descent uses the gradients for training the model (via optimization).[8]

2.2.9 Adam optimization

Fully known as the Adaptive Moment Estimation Algorithm, but abbreviated Adam, this optimization algorithm was introduced in 2015 by two researchers – Diederik P. Kingma and Jimmy Lei Ba. This algorithm simply estimates moments and uses them to optimize a function. It is essentially a combination of the gradient descent with momentum algorithm and the RMS (Root Mean Square) Prop algorithm.
The Adam algorithm calculates an exponential weighted moving average of the gradient and then squares the calculated gradient. This algorithm has two decay parameters that control the decay rates of these calculated moving averages.[9]

There are several advantages of the Adam Algorithm and some of them are listed below:

1- Easy to implement
2- Quite computationally efficient
3- Requires little memory space
4- Good for non-stationary objectives
5- Works well on problems with noisy or sparse gradients
6- Works well with large data sets and large parameters

2.3 Network Architectures

2.3.1 VGG

It is designed on the core idea that deeper networks are better networks.
This architecture from Simonyan and their co-authors was the runner-up in the ImageNet challenge in 2014.
Though they provide a higher level of accuracy, they have an inherently larger number of parameters (~140M) and use a lot more memory than AlexNet. Visual Geometry Group (VGG) has smaller filters than AlexNet, where each filter is of size 3 x 3 but with a lower stride of one, which effectively captures the same receptive field as a 7 x 7 filter with four strides. It has typically 16-19 layers depending on the particular VGG configuration. The VGG CNN architecture figure illustrates this architecture [11]

2.3.2 ResNet

ResNet is currently the state-of-the-art architecture for large-scale image recognition. One of the themes in common with previous architectures is that the deeper the network is, the better the performance. However, with increasing depth of the network, the problem of vanishing gradients is also amplified since each layer successively computes its gradient with respect to the gradient from the previous layer. The larger the number of layers, the smaller the gradients become, eventually vanishing to 0. To avoid this problem, ResNet introduces a shortening edge, where instead of computing the gradient over $F(x)$, you now compute the gradient over $F(x)+x$, where *x* is the original input to the network. This alleviates the effect of $F(x)$ getting successively smaller. The advantage of this strategy is that now you can create deeper networks with as many as 150 layers, which was not possible before.[10]

2.4 Data Processing

Data Processing is a task of converting data from a given form to a much more usable and desired form i.e. making it more meaningful and informative. Using Machine Learning algorithms, mathematical modeling and statistical knowledge, this entire process can be automated. The output of this complete process can be in any desired form like graphs, videos, charts, tables, images and many more, depending on the task we are performing and the requirements of the machine. This might seem to be simple but when it comes to really big organizations like Twitter, Facebook, Administrative bodies like Paliament, UNESCO and health sector organizations, this entire process needs to be performed in a very structured manner. [18]

2.4.1 Laplacian as focus measure

Shape from focus (SFF) uses focus measure operator for depth measurement from a sequence of images. From the analysis of defocused image, it is observed that the focus measure operator should respond to high frequency variations of image intensity and produce maximum values when the image is perfectly focused. Therefore, an effective focus measure operator must be a high-pass filter. Laplacian is mostly used as focus measure operator in the previous SFF methods. In this paper, generalized Laplacian is used as focus measure operator for better 3D shape recovery of objects.[12]

2.4.2 Normalization

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.[13]

2.4.3 Augmentation

Data augmentation is the creation of altered copies of each instance within a training dataset. Let’s unpack this statement in the context of image classification.

When we feed image data into a neural network, there are some features of the images that we would like the neural network to condense or summarize into a set of numbers or weights. In the case of image classification, these features or signals are the pixels which make up the object in the picture. On the other hand, there are features of the images that we would not like the neural network to incorporate in its summary of the images (the summary is the set of weights). In the case of image classification, these features or noise are the pixels which form the background in the picture.[14]

2.5 Model Evaluation

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and overfitted models. There are two methods of evaluating models in data science, Hold-Out and Cross-Validation. To avoid overfitting, both methods use a test set (not seen by the model) to evaluate model performance.[17]

2.5.1 Performance Estimation

The problem of predictive modeling is to create models that have good performance making predictions on new unseen data.
Therefore it is critically important to use robust techniques to train and evaluate your models on your available training data. The more reliable the estimate of the performance on your model, the further you can push the performance and be confident it will translate to the operational use of your model.

There are a number of model evaluation techniques that you can choose from, and the Weka machine learning workbench offers four of them, as follows:

1- Training Dataset
2-Supplied Test Set
3-Percentage Split
4-Cross Validation

3.1 Previous Studies

Recently, researchers have carried out many researches to use artificial intelligence, expert system and neural networks to improve the diagnosis of blood cell diseases.

Several methods and models have been proposed that have contributed to improving the efficiency of this test.

Generally, the microscopic-based evaluation was the standard method to perform RBC
counting analysis.

Despite its long clinical success, this method requires an expertise to manually classify the cells which is tedious, time-consuming and qualitative process [1].

As consequences, several automatic medical diagnosis systems have been developed to help doctors to diagnose disease particularly in Red Blood (RBC) and White Blood (WBC) Cells of human that provides valuable information to pathologists. Red blood cell composition reveals important diagnostic to diagnose patient’s disease and subsequently facilitate doctors to determine an appropriate treatment to the patient.

The RBC count analysis is performed to evaluate mean size and shape of the cells. Various diseases such as anemia, leukemia, malnutrition, chronic inflammation, a renal tumor and organs overloaded with iron can be indicated from abnormally high or low counts in blood cells [1][2].

In this thesis a computer-aided systems is proposed to automate the process of detection and identification of RBC from blood smear image. Initially RBCs region are extracted from the background by using global threshold method applied on green channel color image. Next, noise and holes in the RBCs are abolished by utilizing morphological filter and connected component labeling. Following that, information from the RBCs’ are extracted based on its geometrical properties. Eventually, the RBCs were classified as normal/abnormal by using Artificial Neural Network (ANN) classifier.

The proposed method has been tested on blood cell images and demonstrates a reliable and effective system for classifying normal and abnormal RBC [3].

Tests related to blood cells are examined for the patients as a starting point of diagnosis and information obtained about their abnormalities give doctors a preliminary idea about the illnesses. This thesis issues generation of a CV application that would be used as an assistant of doctors who have domain expertise. The thesis issues segmentation of blood cells, classification of red and white blood cells containing 6 types such as erythrocyte, lymphocyte, platelets, neutrophil, monocytes and eosinophils using the segmentation results. It also discusses about a method for detection of abnormalities on red blood cells (erythrocyte) [4].

There are different areas in medicine where expert systems are used in blood cells diagnosis.

It has been designed and implemented to solve the health condition of human stability.

There are two basic systems of experts used from time to time in health care to diagnose blood diseases and prescribe treatments. The two main systems are: MYCIN and CADUCEUS MYCIN is an early expert system that uses artificial intelligence to identify bacteria that cause acute infections, such as bacteremia and meningitis, and to recommend antibiotics, while adjusting the dose to the patient's body weight.

Among these diverse areas includes an expert system for Eye, Hypertension, Pregnancy, Leukemia and several other Human diseases. Sam S. Abu [4] in his research work for eye expert system, he used CLIPS language in his research thereby serving as a tunnel to the inner workings of the body [4]. Also, J Gudu [5] in his research for expert system to diagnosis and treat hypertension in pregnancy stated that the diagnostic and treatment expert system for hypertension in pregnancy has so far remained at the testing phase of its life cycle and is yet to be implemented [5].

More so, Azar developed an expert system for diagnosing leukemia, his programs simulate the pattern of thinking and the manner of how human operates and causes the operation of expert systems to be close to operations of human or an expert [6]. Leukemia is very common and serious cancer starts in blood tissue such as the bone marrow [6].

It causes large numbers of abnormal blood cells to be produced and enter the blood. Speed is always effective in diagnosis and treatment of Leukemia and recovery of patients, but sometimes there is no access to specialists for patients. Because of this reason, designing a system with specialist knowledge, that offers the diagnosis and appropriate treatment to patients, and providing timely treatment of patients [6] was brought into the scene.

The authors in this study [20] proposed a model for assisting in diagnosing white blood cells diseases. They divided these diseases into two categories, each category includes similar symptoms diseases that may cause confusion in diagnosing. Based on the doctor's selection, one of two approaches was implemented. Each approach was applied on one of the two diseases category by computing different features. Finally, They used Random Forest classifier for final decision. Their proposed approach aimed to early discover white blood cells cancer, reduce the misdiagnosis cases in addition to improve the system learning methodology. Moreover, they allowed the experts only to have the final tuning on the result obtained from the system. Their proposed approach achieved an accuracy of 93% in the first category and 95% in the second category[20].

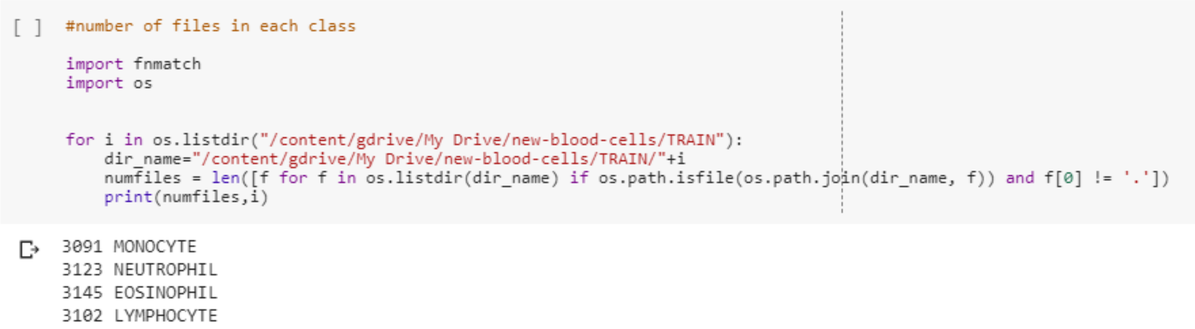
The authrors in their study [21] proposed a system for Iron deficiency anemia (IDA) which is a nutritional disorder that impacts over one billion people worldwide; IDA can be diagnosed by detection of red blood cells (RBCs) that are characteristically small (microcytic) and deficient in hemoglobin (hypochromic). The author proposed a low-cost and rapid method to diagnose IDA using aqueous multiphase systems (AMPS)—thermodynamically stable mixtures of biocompatible polymers and salt that spontaneously form discrete layers having sharp steps in density. AMPS are preloaded into a microhematocrit tube and used with a drop of blood from a fingerstick. After only two minutes in a low-cost centrifuge, the tests (n = 152) were read by eye with a sensitivity of 84% (72–93%) and a specificity of 78% (68–86%), corresponding to an area under the curve (AUC) of 0.89. The AMPS test outperforms diagnosis by hemoglobin alone (AUC = 0.73) and is comparable to methods used in clinics like reticulocyte hemoglobin concentration (AUC = 0.91). Standard machine learning tools were used to analyze images of the resulting tests captured by a standard desktop scanner to 1) slightly improve diagnosis of IDA—sensitivity of 90% (83–96%) and a specificity of 77% (64– 87%), and 2) predict several important red blood cell parameters, such as mean corpuscular hemoglobin concentration. These results suggest that the use of AMPS combined with machine learning provides an approach to developing point-of-care hematology.[21]
The counting and classification of blood cells allows the evaluation and diagnosis of a vast number of diseases. Through the analysis of white blood cells (WBCs) the ALL - Acute Lymphocytic Leukemia, a blood cancer that can be fatal if left untreated, can be detected. Nowadays the morphological analysis of blood cells is performed manually by skilled operators. This involves numerous drawbacks, such as slowness of the analysis and a non-standard accuracy, dependent on the operator skills. In literature there are only few examples of automated systems in order to analyze and classify the blood cells, most of which only partial.

This paper presents a complete and fully automatic method for WBCs identification and classification from microscopic images. The proposed method firstly individuates WBCs from which, subsequently, are extracted morphological features necessary for the final stage of classification. The whole work has been developed using MATLAB environment.[22]

This review focuses on how image processing and machine learning can be useful for the morphological characterization and automatic recognition of cell images captured from peripheral blood smears. The basics of the 3 core elements (segmentation, quantitative features, and classification) are outlined, and recent literature is discussed. Although red blood cells are a significant part of this context, this study focuses on malignant lymphoid cells and blast cells. There is no doubt that these technologies may help the cytologist to perform efficient, objective, and fast morphological analysis of blood cells. They may also help in the interpretation of some morphological features and may serve as learning and survey tools. Although research is still needed, it is important to define screening strategies to exploit the potential of image-based automatic recognition systems integrated in the daily routine of laboratories along with other analysis methodologies.[23]

4.1 Dataset

The data set in this research is a huge collection of images up to approximately 15000 images and these images of the four types of white blood cells and they : EOSINOPHIL, MONOCYTE, NEUTROPHIL, LYMPHOCYTE The number of images in each type of blood cells about 3000 images , Uploaded from the famous site kaggel ().



```
[ ]  #number of files in each class

     import fnmatch
     import os


     for i in os.listdir("/content/gdrive/My Drive/new-blood-cells/TRAIN"):
         dir_name="/content/gdrive/My Drive/new-blood-cells/TRAIN/"+i
         numfiles = len([f for f in os.listdir(dir_name) if os.path.isfile(os.path.join(dir_name, f)) and f[0] != '.'])
         print(numfiles,i)
```

```
3091 MONOCYTE
3123 NEUTROPHIL
3145 EOSINOPHIL
3102 LYMPHOCYTE
```

4.2 Language and tool used

The research team used the third generation of Python language, which is a high-level languages, characterized by simple writing and reading, easy to learn, and is a free and open source languages that allows the use of many libraries, including library keras ,shutil, fnmatch, os .

The research team used several tools, the most important of which is Google Colab to write python codes, a research tool for teaching and searching for a learning machine, an easy to use and does not require any preparation for use, Google Colab is characterized by speed in performance because it has very fast processors of type (GPU).

Google Colab is a free-to-use research project that can store and read all notebooks directly from Google Drive

4.3 Preprocessing

Preprocessing refers to all conversions on raw data before they are entered into the machine learning algorithm or deep learning. For example, researchers encountered multiple problems in images, since different image sizes are not uniform, and some images have huge sizes [(512,254) (700,1024) (256,256)],And some corrupted images can't be read.

These problems exacerbate the memory and the nature of the output has become necessary to standardize images, Researchers have standardized images by using Fast Image Resizer program or using Python to standardize images. He started to standardize from size (512,512) and then to size (256,256), but the size of the images is still large on memory.

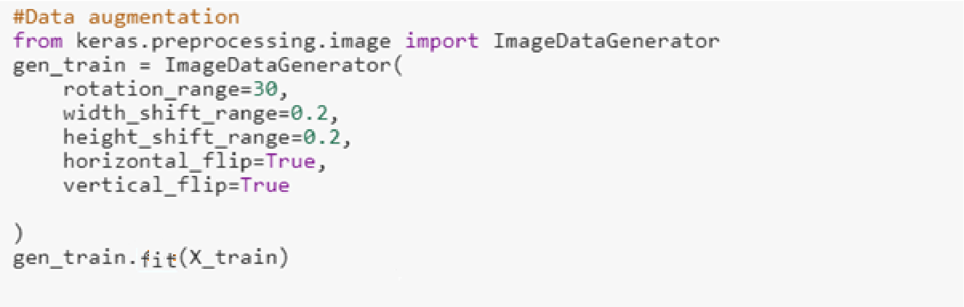
In the end, the researchers standardized the size of the images to 128 by,128 pixels.

4.4 Data augmentation

Data augmentation is the creation of modified versions of each case within the training dataset.

When the amount of images is insufficient or the process of learning encountered the problem of overfitting "A well-known issue for data scientists… Overfitting is a modeling error which occurs when a function is too closely fit to a limited set of data points."[19]

overfitting is to reach the stage that the system is unable to learn, then we use the feature of data magnification, a feature provided by the Kares library, which produces new images from the old images and then use the new images instead From old images, it is produced by rotation the image horizontally and vertically , extending the image , and changing the width and height of the image, and a number of images are produced from one image.

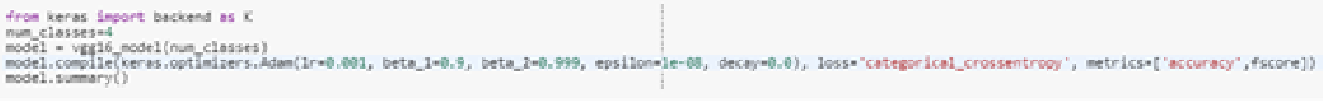


```
#Data augmentation
from keras.preprocessing.image import ImageDataGenerator
gen_train = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    vertical_flip=True

)
gen_train.fit(X_train)
```

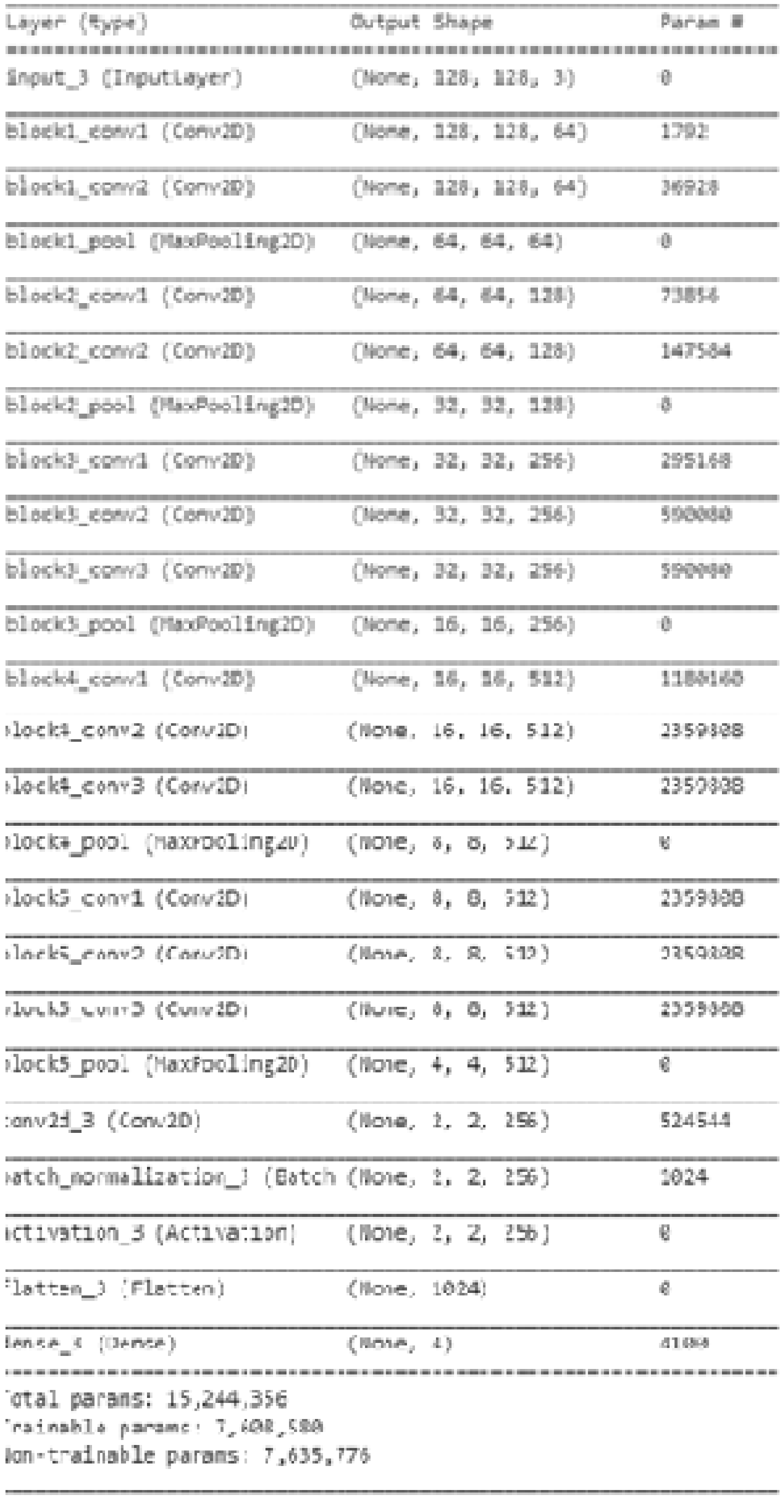
4.5 Network Architecture

The research team built an integrated model from start to finish, and this model was trained in various ways, but the best accuracy reached is 82%.

Therefore, was used a pre-trained model to classify items , It classifies about 1000 items and the name of this model is vgg16.
The vgg16 model consists of five layers and a classifier, The original form classifier was deleted and the fifth layer was deleted, note that the first, second, third and fourth layer of the vgg16 model is frozen and have not been modified.
finally, a classifier was then added For blood cells ,And then retrain the new model of blood cell classification.



```
from keras import backend as K
num_classes=4
model = vgg16_model(num_classes)
model.compile(keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0), loss='categorical_crossentropy', metrics=['accuracy',fscore])
model.summary()
```

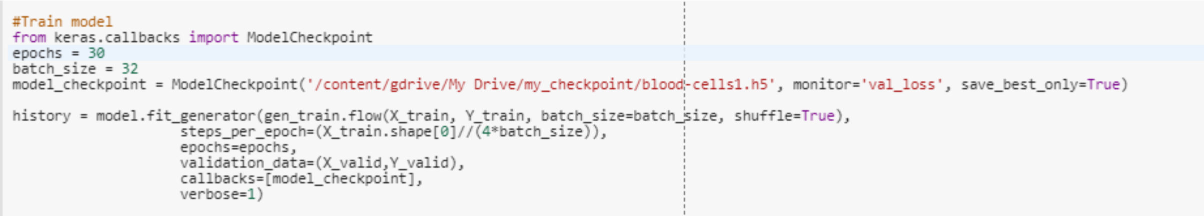


```
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 128, 128, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 128, 128, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 128, 128, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 64, 64, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 64, 64, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 64, 64, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 32, 32, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 32, 32, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 16, 16, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 16, 16, 512)       1180160
lock4_conv2 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
lock4_conv3 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
lock4_pool (MaxPooling2D)   (None, 8, 8, 512)         0
_________________________________________________________________
lock5_conv1 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
lock5_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
lock5_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
lock5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
onv2d_3 (Conv2D)            (None, 2, 2, 256)         524544
_________________________________________________________________
atch_normalization_3 (Batch (None, 2, 2, 256)         1024
_________________________________________________________________
ctivation_3 (Activation)    (None, 2, 2, 256)         0
_________________________________________________________________
latten_3 (Flatten)          (None, 1024)              0
_________________________________________________________________
ense_4 (Dense)              (None, 4)                 4100
=================================================================
otal params: 15,244,356
rainable params: 7,608,580
on-trainable params: 7,635,776
_________________________________________________________________
```

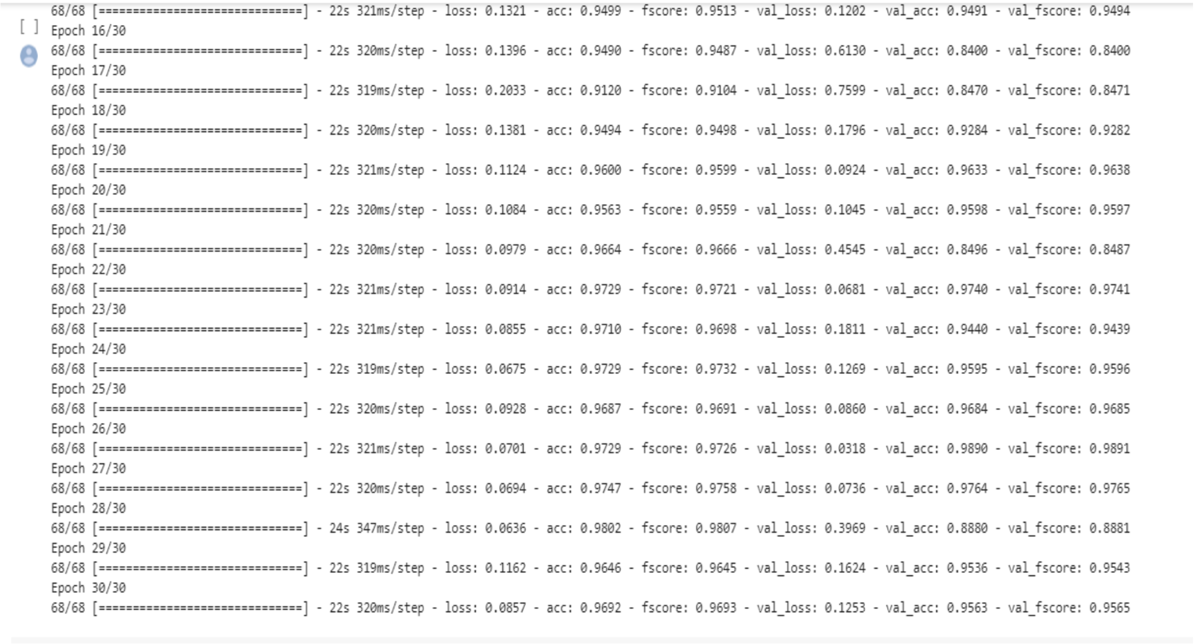
4.6 Training and Validation the Model

In the training and verification process, the research team used fit genartion model to train the blood cell model.in the first time the model was trained 30 epochs , but the training accuracy was nearly 96% and the val_fscore was nearly 97% .
The model was trained several times to improve accuracy.

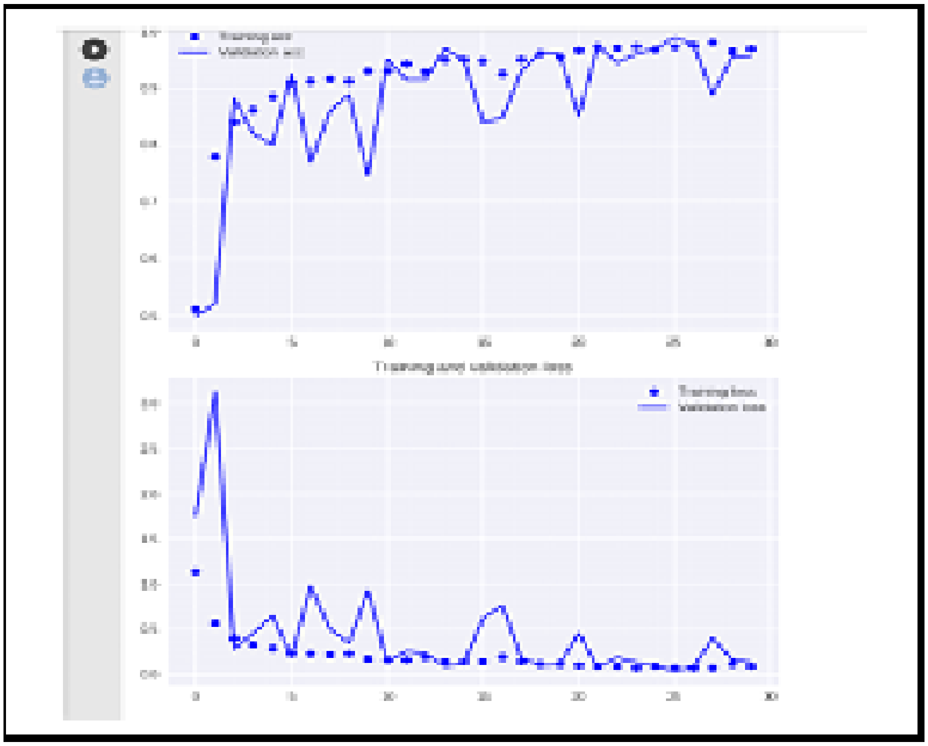
last time it was re-trained 10 epochs , until it reached an accuracy of 99% ,and an f_score reached of 98%.

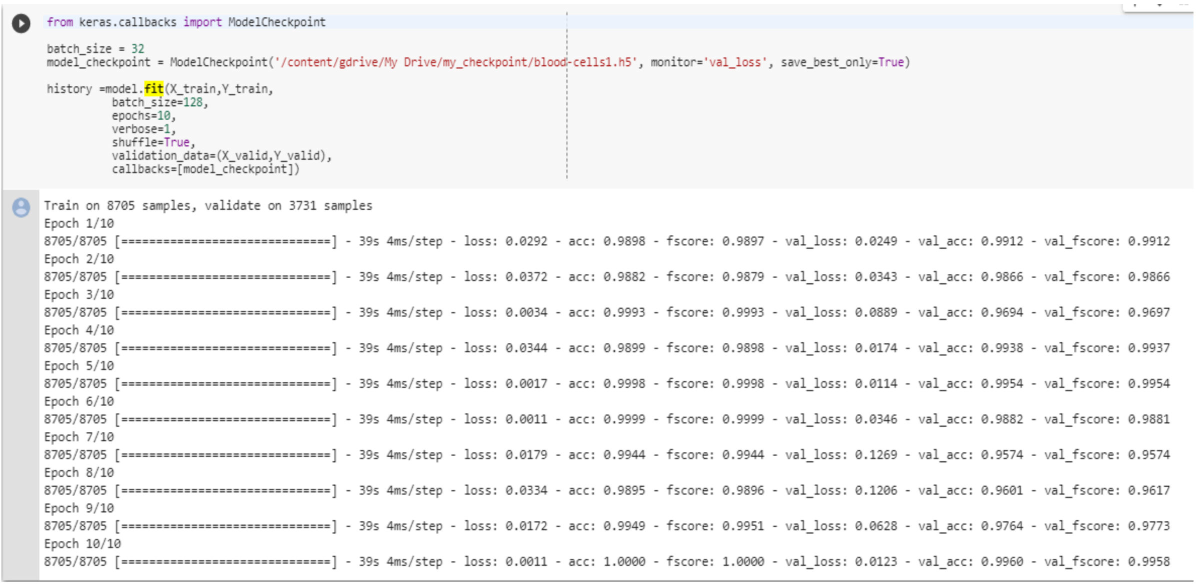


```
#Train model
from keras.callbacks import ModelCheckpoint
epochs = 30
batch_size = 32
model_checkpoint = ModelCheckpoint('/content/gdrive/My Drive/my_checkpoint/blood-cells1.h5', monitor='val_loss', save_best_only=True)

history = model.fit_generator(gen_train.flow(X_train, Y_train, batch_size=batch_size, shuffle=True),
                    steps_per_epoch=(X_train.shape[0]//(4*batch_size)),
                    epochs=epochs,
                    validation_data=(X_valid,Y_valid),
                    callbacks=[model_checkpoint],
                    verbose=1)
```



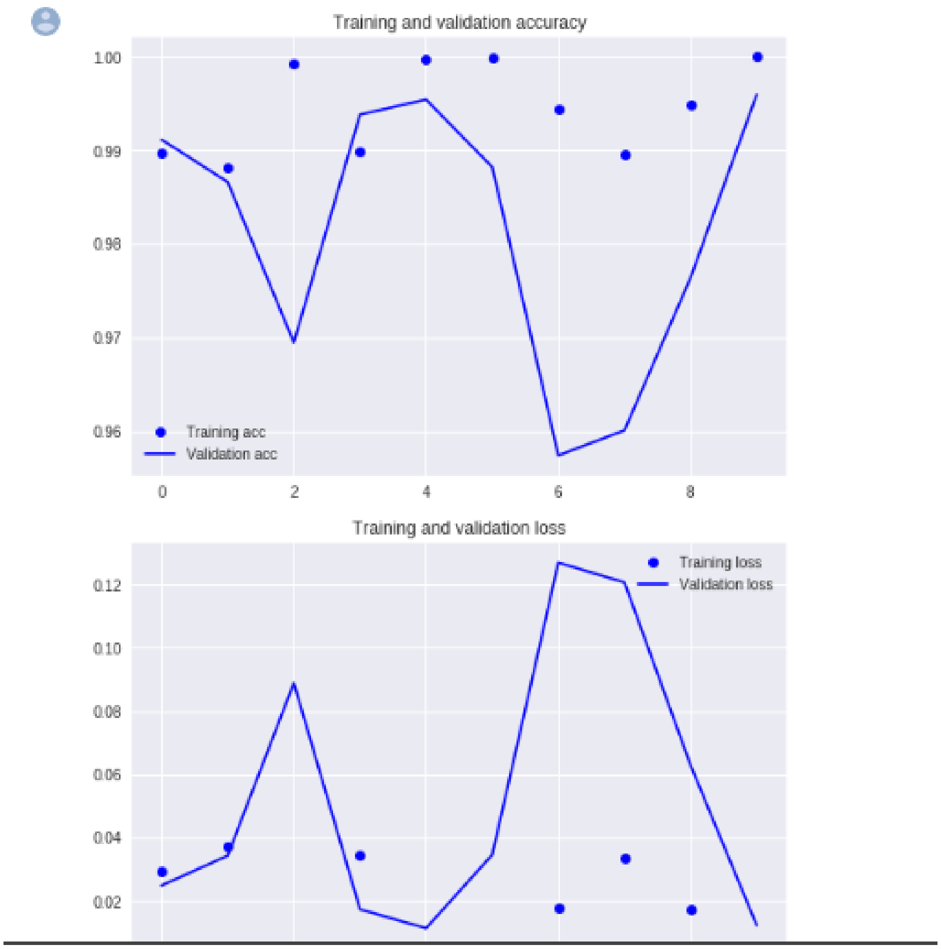
```
68/68 [==============================] - 22s 321ms/step - loss: 0.1321 - acc: 0.9499 - fscore: 0.9513 - val_loss: 0.1202 - val_acc: 0.9491 - val_fscore: 0.9494
Epoch 16/30
68/68 [==============================] - 22s 320ms/step - loss: 0.1396 - acc: 0.9490 - fscore: 0.9487 - val_loss: 0.6130 - val_acc: 0.8400 - val_fscore: 0.8400
Epoch 17/30
68/68 [==============================] - 22s 319ms/step - loss: 0.2033 - acc: 0.9120 - fscore: 0.9104 - val_loss: 0.7599 - val_acc: 0.8470 - val_fscore: 0.8471
Epoch 18/30
68/68 [==============================] - 22s 320ms/step - loss: 0.1381 - acc: 0.9494 - fscore: 0.9498 - val_loss: 0.1796 - val_acc: 0.9284 - val_fscore: 0.9282
Epoch 19/30
68/68 [==============================] - 22s 321ms/step - loss: 0.1124 - acc: 0.9600 - fscore: 0.9599 - val_loss: 0.0924 - val_acc: 0.9633 - val_fscore: 0.9638
Epoch 20/30
68/68 [==============================] - 22s 320ms/step - loss: 0.1084 - acc: 0.9563 - fscore: 0.9559 - val_loss: 0.1045 - val_acc: 0.9598 - val_fscore: 0.9597
Epoch 21/30
68/68 [==============================] - 22s 320ms/step - loss: 0.0979 - acc: 0.9664 - fscore: 0.9666 - val_loss: 0.4545 - val_acc: 0.8496 - val_fscore: 0.8487
Epoch 22/30
68/68 [==============================] - 22s 321ms/step - loss: 0.0914 - acc: 0.9729 - fscore: 0.9721 - val_loss: 0.0681 - val_acc: 0.9740 - val_fscore: 0.9741
Epoch 23/30
68/68 [==============================] - 22s 321ms/step - loss: 0.0855 - acc: 0.9710 - fscore: 0.9698 - val_loss: 0.1811 - val_acc: 0.9440 - val_fscore: 0.9439
Epoch 24/30
68/68 [==============================] - 22s 319ms/step - loss: 0.0675 - acc: 0.9729 - fscore: 0.9732 - val_loss: 0.1269 - val_acc: 0.9595 - val_fscore: 0.9596
Epoch 25/30
68/68 [==============================] - 22s 320ms/step - loss: 0.0928 - acc: 0.9687 - fscore: 0.9691 - val_loss: 0.0860 - val_acc: 0.9684 - val_fscore: 0.9685
Epoch 26/30
68/68 [==============================] - 22s 321ms/step - loss: 0.0701 - acc: 0.9729 - fscore: 0.9726 - val_loss: 0.0318 - val_acc: 0.9890 - val_fscore: 0.9891
Epoch 27/30
68/68 [==============================] - 22s 320ms/step - loss: 0.0694 - acc: 0.9747 - fscore: 0.9758 - val_loss: 0.0736 - val_acc: 0.9764 - val_fscore: 0.9765
Epoch 28/30
68/68 [==============================] - 24s 347ms/step - loss: 0.0636 - acc: 0.9802 - fscore: 0.9807 - val_loss: 0.3969 - val_acc: 0.8880 - val_fscore: 0.8881
Epoch 29/30
68/68 [==============================] - 22s 319ms/step - loss: 0.1162 - acc: 0.9646 - fscore: 0.9645 - val_loss: 0.1624 - val_acc: 0.9536 - val_fscore: 0.9543
Epoch 30/30
68/68 [==============================] - 22s 320ms/step - loss: 0.0857 - acc: 0.9692 - fscore: 0.9693 - val_loss: 0.1253 - val_acc: 0.9563 - val_fscore: 0.9565
```





```
from keras.callbacks import ModelCheckpoint

batch_size = 32
model_checkpoint = ModelCheckpoint('/content/gdrive/My Drive/my_checkpoint/blood-cells1.h5', monitor='val_loss', save_best_only=True)

history =model.fit(X_train,Y_train,
          batch_size=128,
          epochs=10,
          verbose=1,
          shuffle=True,
          validation_data=(X_valid,Y_valid),
          callbacks=[model_checkpoint])
```

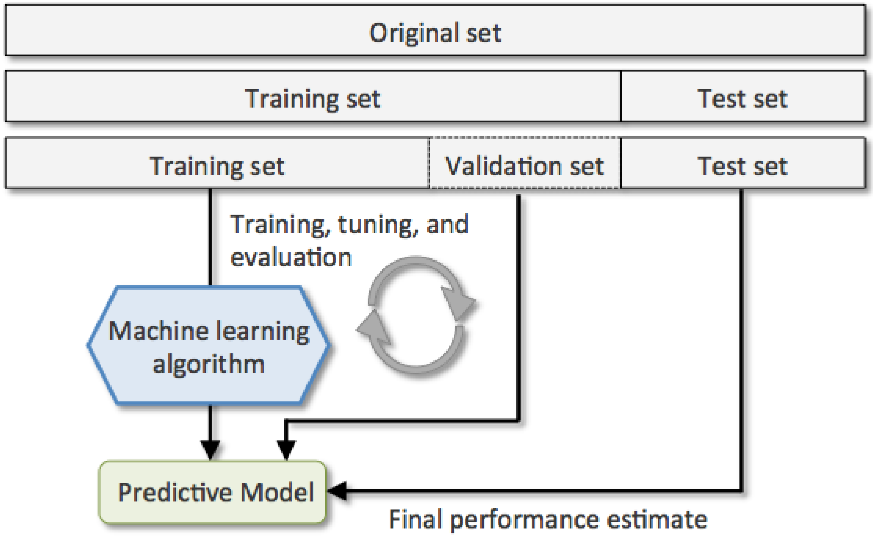
```
Train on 8705 samples, validate on 3731 samples
Epoch 1/10
8705/8705 [==============================] - 39s 4ms/step - loss: 0.0292 - acc: 0.9898 - fscore: 0.9897 - val_loss: 0.0249 - val_acc: 0.9912 - val_fscore: 0.9912
Epoch 2/10
8705/8705 [==============================] - 39s 4ms/step - loss: 0.0372 - acc: 0.9882 - fscore: 0.9879 - val_loss: 0.0343 - val_acc: 0.9866 - val_fscore: 0.9866
Epoch 3/10
8705/8705 [==============================] - 39s 4ms/step - loss: 0.0034 - acc: 0.9993 - fscore: 0.9993 - val_loss: 0.0889 - val_acc: 0.9694 - val_fscore: 0.9697
Epoch 4/10
8705/8705 [==============================] - 39s 4ms/step - loss: 0.0344 - acc: 0.9899 - fscore: 0.9898 - val_loss: 0.0174 - val_acc: 0.9938 - val_fscore: 0.9937
Epoch 5/10
8705/8705 [==============================] - 39s 4ms/step - loss: 0.0017 - acc: 0.9998 - fscore: 0.9998 - val_loss: 0.0114 - val_acc: 0.9954 - val_fscore: 0.9954
Epoch 6/10
8705/8705 [==============================] - 39s 4ms/step - loss: 0.0011 - acc: 0.9999 - fscore: 0.9999 - val_loss: 0.0346 - val_acc: 0.9882 - val_fscore: 0.9881
Epoch 7/10
8705/8705 [==============================] - 39s 4ms/step - loss: 0.0179 - acc: 0.9944 - fscore: 0.9944 - val_loss: 0.1269 - val_acc: 0.9574 - val_fscore: 0.9574
Epoch 8/10
8705/8705 [==============================] - 39s 4ms/step - loss: 0.0334 - acc: 0.9895 - fscore: 0.9896 - val_loss: 0.1206 - val_acc: 0.9601 - val_fscore: 0.9617
Epoch 9/10
8705/8705 [==============================] - 39s 4ms/step - loss: 0.0172 - acc: 0.9949 - fscore: 0.9951 - val_loss: 0.0628 - val_acc: 0.9764 - val_fscore: 0.9773
Epoch 10/10
8705/8705 [==============================] - 39s 4ms/step - loss: 0.0011 - acc: 1.0000 - fscore: 1.0000 - val_loss: 0.0123 - val_acc: 0.9960 - val_fscore: 0.9958
```

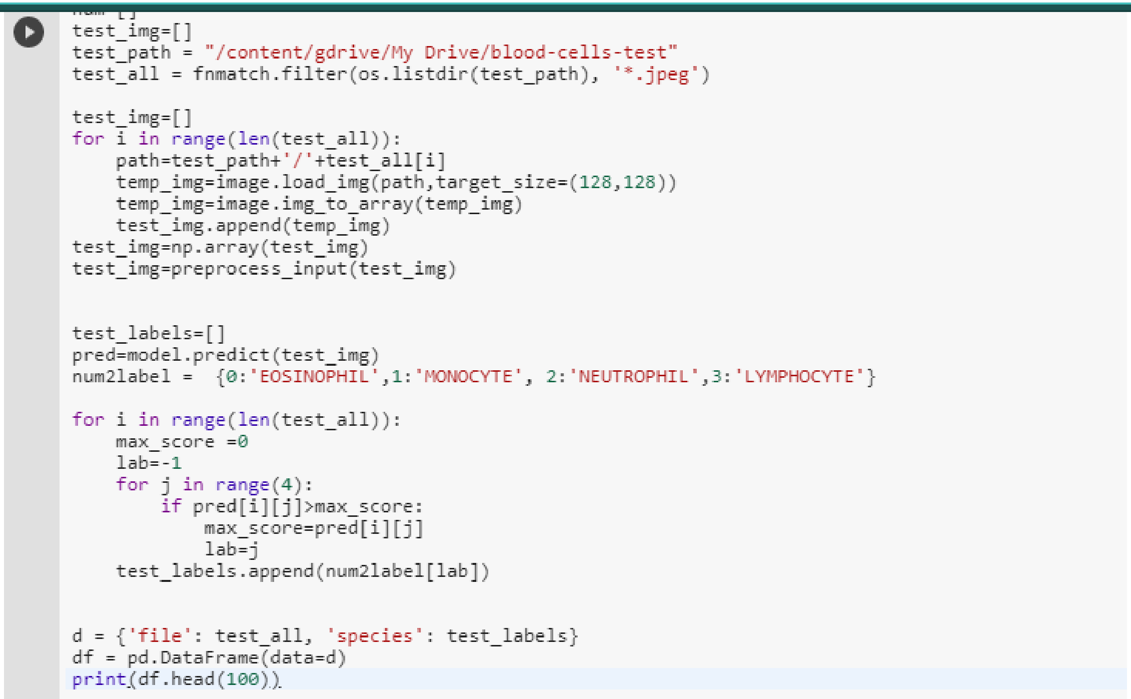


5.1 Data Set for testing the model

The data used to build the final model usually comes from multiple datasets. In particular, three data sets are commonly used in different stages of the creation of the model.

The first group is a training dataset is a dataset of examples used for learning that is to fit the parameters (e.g., weights) of, for example, a classifier.
The second group is a validation dataset is a dataset of examples used to tune the hyperparameters (i.e. the architecture) of a classifier. It is sometimes also called the development set or the "dev set".
The third group is a test dataset is a dataset that is independent of the training dataset, but that follows the same probability distribution as the training dataset. If a model fit to the training dataset also fits the test dataset well, minimal overfitting has taken place (see figure below). A better fitting of the training dataset as opposed to the test dataset usually points to overfitting.
A test set is therefore a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier.[20][21]
The research team used a blood cell data set of approximately 1,500 blood cell images.
Initially, the research team then divided the dataset into three sections: Training, Verification and testing.
Since the training section contains 60% of the images, the verification section contains 30% of the pictures and the testing section contains 10% of the images of blood cells.


Original set
Training set
Test set
Training set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Predictive Model
Final performance estimate

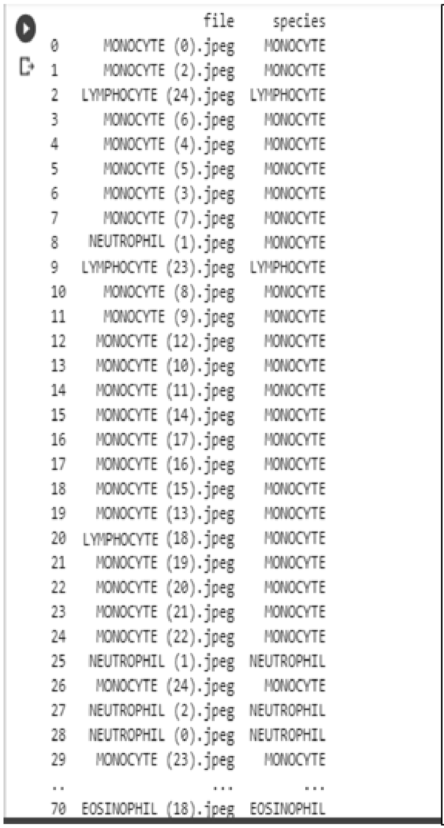


```
num=[]
test_img=[]
test_path = "/content/gdrive/My Drive/blood-cells-test"
test_all = fnmatch.filter(os.listdir(test_path), '*.jpeg')

test_img=[]
for i in range(len(test_all)):
    path=test_path+'/'+test_all[i]
    temp_img=image.load_img(path,target_size=(128,128))
    temp_img=image.img_to_array(temp_img)
    test_img.append(temp_img)
test_img=np.array(test_img)
test_img=preprocess_input(test_img)


test_labels=[]
pred=model.predict(test_img)
num2label =  {0:'EOSINOPHIL',1:'MONOCYTE', 2:'NEUTROPHIL',3:'LYMPHOCYTE'}

for i in range(len(test_all)):
    max_score =0
    lab=-1
    for j in range(4):
        if pred[i][j]>max_score:
            max_score=pred[i][j]
            lab=j
    test_labels.append(num2label[lab])


d = {'file': test_all, 'species': test_labels}
df = pd.DataFrame(data=d)
print(df.head(100))
```

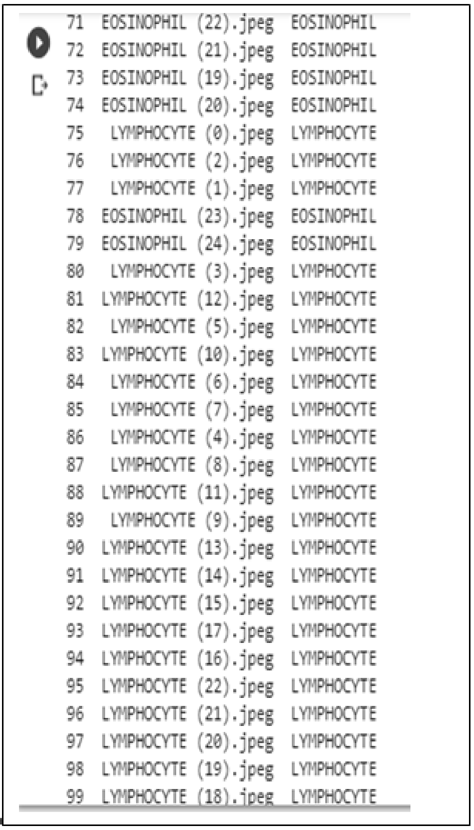
5.2 Testing the model

The data set for the testing consists of approximately 1300 images of the four types of blood cells , and all of These images are of JPEG type .

And then it was an experiment ,this image after the completion of the training and verification process the result of the testing process about 98%, also was shown for the first 100 images of different types of blood cells to check the ratio .



```
                    file     species
0      MONOCYTE (0).jpeg    MONOCYTE
1      MONOCYTE (2).jpeg    MONOCYTE
2    LYMPHOCYTE (24).jpeg  LYMPHOCYTE
3      MONOCYTE (6).jpeg    MONOCYTE
4      MONOCYTE (4).jpeg    MONOCYTE
5      MONOCYTE (5).jpeg    MONOCYTE
6      MONOCYTE (3).jpeg    MONOCYTE
7      MONOCYTE (7).jpeg    MONOCYTE
8    NEUTROPHIL (1).jpeg    MONOCYTE
9    LYMPHOCYTE (23).jpeg  LYMPHOCYTE
10     MONOCYTE (8).jpeg    MONOCYTE
11     MONOCYTE (9).jpeg    MONOCYTE
12    MONOCYTE (12).jpeg    MONOCYTE
13    MONOCYTE (10).jpeg    MONOCYTE
14    MONOCYTE (11).jpeg    MONOCYTE
15    MONOCYTE (14).jpeg    MONOCYTE
16    MONOCYTE (17).jpeg    MONOCYTE
17    MONOCYTE (16).jpeg    MONOCYTE
18    MONOCYTE (15).jpeg    MONOCYTE
19    MONOCYTE (13).jpeg    MONOCYTE
20   LYMPHOCYTE (18).jpeg    MONOCYTE
21    MONOCYTE (19).jpeg    MONOCYTE
22    MONOCYTE (20).jpeg    MONOCYTE
23    MONOCYTE (21).jpeg    MONOCYTE
24    MONOCYTE (22).jpeg    MONOCYTE
25   NEUTROPHIL (1).jpeg  NEUTROPHIL
26    MONOCYTE (24).jpeg    MONOCYTE
27   NEUTROPHIL (2).jpeg  NEUTROPHIL
28   NEUTROPHIL (0).jpeg  NEUTROPHIL
29    MONOCYTE (23).jpeg    MONOCYTE
..                   ...         ...
70   EOSINOPHIL (18).jpeg  EOSINOPHIL
```



```
71  EOSINOPHIL (22).jpeg  EOSINOPHIL
72  EOSINOPHIL (21).jpeg  EOSINOPHIL
73  EOSINOPHIL (19).jpeg  EOSINOPHIL
74  EOSINOPHIL (20).jpeg  EOSINOPHIL
75    LYMPHOCYTE (0).jpeg  LYMPHOCYTE
76    LYMPHOCYTE (2).jpeg  LYMPHOCYTE
77    LYMPHOCYTE (1).jpeg  LYMPHOCYTE
78  EOSINOPHIL (23).jpeg  EOSINOPHIL
79  EOSINOPHIL (24).jpeg  EOSINOPHIL
80    LYMPHOCYTE (3).jpeg  LYMPHOCYTE
81  LYMPHOCYTE (12).jpeg  LYMPHOCYTE
82    LYMPHOCYTE (5).jpeg  LYMPHOCYTE
83  LYMPHOCYTE (10).jpeg  LYMPHOCYTE
84    LYMPHOCYTE (6).jpeg  LYMPHOCYTE
85    LYMPHOCYTE (7).jpeg  LYMPHOCYTE
86    LYMPHOCYTE (4).jpeg  LYMPHOCYTE
87    LYMPHOCYTE (8).jpeg  LYMPHOCYTE
88  LYMPHOCYTE (11).jpeg  LYMPHOCYTE
89    LYMPHOCYTE (9).jpeg  LYMPHOCYTE
90  LYMPHOCYTE (13).jpeg  LYMPHOCYTE
91  LYMPHOCYTE (14).jpeg  LYMPHOCYTE
92  LYMPHOCYTE (15).jpeg  LYMPHOCYTE
93  LYMPHOCYTE (17).jpeg  LYMPHOCYTE
94  LYMPHOCYTE (16).jpeg  LYMPHOCYTE
95  LYMPHOCYTE (22).jpeg  LYMPHOCYTE
96  LYMPHOCYTE (21).jpeg  LYMPHOCYTE
97  LYMPHOCYTE (20).jpeg  LYMPHOCYTE
98  LYMPHOCYTE (19).jpeg  LYMPHOCYTE
99  LYMPHOCYTE (18).jpeg  LYMPHOCYTE
```

5.3Result and Discussion

The result and discussion of this research are that the research team used two experiments to build a model for diagnosing blood cells to help doctors make decisions.
The first experiment was when the research team built a complete model from start to finish,
However, the research team had a problem with accuracy of not more than 82%.
The second experiment was when the research team used the trained model vgg16 and modified it.
By deleting the fifth layer and adding the classifier of blood cells the research teams could reach 99% accuracy in training, 97% accuracy in the validation, and 98% in testing.
Note that, using a pre-trained model in the current problem was better in terms of accuracy than building a complete model from start to finish.

6.1 Conclusion

The conclusion of this research work is that many doctors and specialists today have a problem in the diagnosis of blood cells and the inability to distinguish between types of blood cells
So the research team decided to help doctors diagnose blood cells, by building a deep learning model to solve this problem, this model was written in the third language version of Python listed on the Google Colab platform.
The research team used two methods to build this model.
The first method is to build a model from start to the end, but the accuracy was no more than 82%.
The second method is to use a pre-trained model called vgg16, it was amended and added a Classifier for blood cells. the researchers arrived after using this model to accuracy in training nearly 98% and accuracy in verification nearly 99% and in the testing to 98%. These results were better and could be used to diagnose blood cells.

6.2 Future Work

Future work can involve enhancing the performance of the model as follows:
The ability to develop this system to work through the application via mobile phones, this idea was included while doing in this research work, but time did not help us.

References

1. Al Barsh, Y. I., et al. (2020). "MPG Prediction Using Artificial Neural Network." International Journal of Academic Information Systems Research (IJAISR) 4(11): 7-16.
2. Alajrami, E., et al. (2019). "Blood Donation Prediction using Artificial Neural Network." International Journal of Academic Engineering Research (IJAER) 3(10): 1-7.
3. Alajrami, E., et al. (2020). "Handwritten Signature Verification using Deep Learning." International Journal of Academic Multidisciplinary Research (IJAMR) 3(12): 39-44.
4. Al-Araj, R. S. A., et al. (2020). "Classification of Animal Species Using Neural Network." International Journal of Academic Engineering Research (IJAER) 4(10): 23-31.
5. Al-Atrash, Y. E., et al. (2020). "Modeling Cognitive Development of the Balance Scale Task Using ANN." International Journal of Academic Information Systems Research (IJAISR) 4(9): 74-81.
6. Alghoul, A., et al. (2018). "Email Classification Using Artificial Neural Network." International Journal of Academic Engineering Research (IJAER) 2(11): 8-14.
7. Abu Nada, A. M., et al. (2020). "Age and Gender Prediction and Validation Through Single User Images Using CNN." International Journal of Academic Engineering Research (IJAER) 4(8): 21-24.
8. Abu Nada, A. M., et al. (2020). "Arabic Text Summarization Using AraBERT Model Using Extractive Text Summarization Approach." International Journal of Academic Information Systems Research (IJAISR) 4(8): 6-9.
9. Abu-Saqer, M. M., et al. (2020). "Type of Grapefruit Classification Using Deep Learning." International Journal of Academic Information Systems Research (IJAISR) 4(1): 1-5.
10. Afana, M., et al. (2018). "Artificial Neural Network for Forecasting Car Mileage per Gallon in the City." International Journal of Advanced Science and Technology 124: 51-59.
11. Al-Araj, R. S. A., et al. (2020). "Classification of Animal Species Using Neural Network." International Journal of Academic Engineering Research (IJAER) 4(10): 23-31.
12. Al-Atrash, Y. E., et al. (2020). "Modeling Cognitive Development of the Balance Scale Task Using ANN." International Journal of Academic Information Systems Research (IJAISR) 4(9): 74-81.
13. Alghoul, A., et al. (2018). "Email Classification Using Artificial Neural Network." International Journal of Academic Engineering Research (IJAER) 2(11): 8-14.
14. Al-Kahlout, M. M., et al. (2020). "Neural Network Approach to Predict Forest Fires using Meteorological Data." International Journal of Academic Engineering Research (IJAER) 4(9): 68-72.
15. Alkronz, E. S., et al. (2019). "Prediction of Whether Mushroom is Edible or Poisonous Using Back-propagation Neural Network." International Journal of Academic and Applied Research (IJAAR) 3(2): 1-8.
16. Al-Madhoun, O. S. E.-D., et al. (2020). "Low Birth Weight Prediction Using JNN." International Journal of Academic Health and Medical Research (IJAHMR) 4(11): 8-14.
17. Al-Massri, R., et al. (2018). "Classification Prediction of SBRCTs Cancers Using Artificial Neural Network." International Journal of Academic Engineering Research (IJAER) 2(11): 1-7.
18. Al-Mobayed, A. A., et al. (2020). "Artificial Neural Network for Predicting Car Performance Using JNN." International Journal of Engineering and Information Systems (IJEAIS) 4(9): 139-145.
19. Al-Mubayyed, O. M., et al. (2019). "Predicting Overall Car Performance Using Artificial Neural Network." International Journal of Academic and Applied Research (IJAAR) 3(1): 1-5.
20. David Adel, John Mounir, Mahmoud El-Shafey, Youssef Alaa Eldin, Noha El Masry, Ashraf AbdelRaouf, Ihab S. Abd Elhamid, "Oral Epithelial Dysplasia Computer Aided Diagnostic Approach", Computer Engineering and Systems (ICCES) 2018 13th International Conference on, pp. 313-318, 2018.
21. Jonathan W. Hennek, Ashok A. Kumar, Alex B. Wiltschko,, Matthew R. Patton, Si Yi Ryan Lee, Carlo Brugnara, Ryan P. Adams, and George M. Whitesides, Diagnosis of iron deficiency anemia using density-based fractionation of red blood cells, Lab Chip, 2016,16, 3929-3939
22. Lorenzo Putzu, Cecilia Di Ruberto, White Blood Cells Identification and Classification from Leukemic Blood Image, IWBBIO 2013. Proceedings Granada, 18-20 March, 2013.
23. Rodellar, S. Alférez, A. Acevedo, A. Molina, Image processing and machine learning in the morphological analysis of blood cells. Special Issue: International Society for Laboratory Hematology 2018 Education Issue. Volume40, IssueS1, May 2018, Pages 46-53
24. Alshawwa, I. A., et al. (2020). "Analyzing Types of Cherry Using Deep Learning." International Journal of Academic Engineering Research (IJAER) 4(1): 1-5.
25. Al-Shawwa, M., et al. (2018). "Predicting Temperature and Humidity in the Surrounding Environment Using Artificial Neural Network." International Journal of Academic Pedagogical Research (IJAPR) 2(9): 1-6.
26. Ashqar, B. A., et al. (2019). "Plant Seedlings Classification Using Deep Learning." International Journal of Academic Information Systems Research (IJAISR) 3(1): 7-14.
27. Bakr, M. A. H. A., et al. (2020). "Breast Cancer Prediction using JNN." International Journal of Academic Information Systems Research (IJAISR) 4(10): 1-8.
28. Barhoom, A. M., et al. (2019). "Predicting Titanic Survivors using Artificial Neural Network." International Journal of Academic Engineering Research (IJAER) 3(9): 8-12.
29. Belbeisi, H. Z., et al. (2020). "Effect of Oxygen Consumption of Thylakoid Membranes (Chloroplasts) From Spinach after Inhibition Using JNN." International Journal of Academic Health and Medical Research (IJAHMR) 4(11): 1-7.
30. Dalffa, M. A., et al. (2019). "Tic-Tac-Toe Learning Using Artificial Neural Networks." International Journal of Engineering and Information Systems (IJEAIS) 3(2): 9-19.
31. Dawood, K. J., et al. (2020). "Artificial Neural Network for Mushroom Prediction." International Journal of Academic Information Systems Research (IJAISR) 4(10): 9-17.
32. Dheir, I. M., et al. (2020). "Classifying Nuts Types Using Convolutional Neural Network." International Journal of Academic Information Systems Research (IJAISR) 3(12): 12-18.
33. El-Khatib, M. J., et al. (2019). "Glass Classification Using Artificial Neural Network." International Journal of Academic Pedagogical Research (IJAPR) 3(2): 25-31.
34. El-Mahelawi, J. K., et al. (2020). "Tumor Classification Using Artificial Neural Networks." International Journal of Academic Engineering Research (IJAER) 4(11): 8-15.
35. El-Mashharawi, H. Q., et al. (2020). "Grape Type Classification Using Deep Learning." International Journal of Academic Engineering Research (IJAER) 3(12): 41-45.
36. Elzamly, A., et al. (2015). "Classification of Software Risks with Discriminant Analysis Techniques in Software planning Development Process." International Journal of Advanced Science and Technology 81: 35-48.
37. Elzamly, A., et al. (2015). "Predicting Software Analysis Process Risks Using Linear Stepwise Discriminant Analysis: Statistical Methods." Int. J. Adv. Inf. Sci. Technol 38(38): 108-115.
38. Elzamly, A., et al. (2017). "Predicting Critical Cloud Computing Security Issues using Artificial Neural Network (ANNs) Algorithms in Banking Organizations." International Journal of Information Technology and Electrical Engineering 6(2): 40-45.
39. Habib, N. S., et al. (2020). "Presence of Amphibian Species Prediction Using Features Obtained from GIS and Satellite Images." International Journal of Academic and Applied Research (IJAAR) 4(11): 13-22.
40. Harz, H. H., et al. (2020). "Artificial Neural Network for Predicting Diabetes Using JNN." International Journal of Academic Engineering Research (IJAER) 4(10): 14-22.
41. Hassanein, R. A. A., et al. (2020). "Artificial Neural Network for Predicting Workplace Absenteeism." International Journal of Academic Engineering Research (IJAER) 4(9): 62-67.
42. Heriz, H. H., et al. (2018). "English Alphabet Prediction Using Artificial Neural Networks." International Journal of Academic Pedagogical Research (IJAPR) 2(11): 8-14.
43. Jaber, A. S., et al. (2020). "Evolving Efficient Classification Patterns in Lymphography Using EasyNN." International Journal of Academic Information Systems Research (IJAISR) 4(9): 66-73.
44. Kashf, D. W. A., et al. (2018). "Predicting DNA Lung Cancer using Artificial Neural Network." International Journal of Academic Pedagogical Research (IJAPR) 2(10): 6-13.
45. Khalil, A. J., et al. (2019). "Energy Efficiency Predicting using Artificial Neural Network." International Journal of Academic Pedagogical Research (IJAPR) 3(9): 1-8.
46. Kweik, O. M. A., et al. (2020). "Artificial Neural Network for Lung Cancer Detection." International Journal of Academic Engineering Research (IJAER) 4(11): 1-7.
47. Maghari, A. M., et al. (2020). "Books' Rating Prediction Using Just Neural Network." International Journal of Engineering and Information Systems (IJEAIS) 4(10): 17-22.
48. Mettleq, A. S. A., et al. (2020). "Mango Classification Using Deep Learning." International Journal of Academic Engineering Research (IJAER) 3(12): 22-29.
49. Metwally, N. F., et al. (2018). "Diagnosis of Hepatitis Virus Using Artificial Neural Network." International Journal of Academic Pedagogical Research (IJAPR) 2(11): 1-7.
50. Mohammed, G. R., et al. (2020). "Predicting the Age of Abalone from Physical Measurements Using Artificial Neural Network." International Journal of Academic and Applied Research (IJAAR) 4(11): 7-12.
51. Musleh, M. M., et al. (2019). "Predicting Liver Patients using Artificial Neural Network." International Journal of Academic Information Systems Research (IJAISR) 3(10): 1-11.
52. Oriban, A. J. A., et al. (2020). "Antibiotic Susceptibility Prediction Using JNN." International Journal of Academic Information Systems Research (IJAISR) 4(11): 1-6.
53. Qwaider, S. R., et al. (2020). "Artificial Neural Network Prediction of the Academic Warning of Students in the Faculty of Engineering and Information Technology in Al-Azhar University-Gaza." International Journal of Academic Information Systems Research (IJAISR) 4(8): 16-22.
54. Sadek, R. M., et al. (2019). "Parkinson's Disease Prediction Using Artificial Neural Network." International Journal of Academic Health and Medical Research (IJAHMR) 3(1): 1-8.
55. Salah, M., et al. (2018). "Predicting Medical Expenses Using Artificial Neural Network." International Journal of Engineering and Information Systems (IJEAIS) 2(20): 11-17.
56. Salman, F. M., et al. (2020). "COVID-19 Detection using Artificial Intelligence." International Journal of Academic Engineering Research (IJAER) 4(3): 18-25.
57. Samra, M. N. A., et al. (2020). "ANN Model for Predicting Protein Localization Sites in Cells." International Journal of Academic and Applied Research (IJAAR) 4(9): 43-50.
58. Shawarib, M. Z. A., et al. (2020). "Breast Cancer Diagnosis and Survival Prediction Using JNN." International Journal of Engineering and Information Systems (IJEAIS) 4(10): 23-30.
59. Zaqout, I., et al. (2015). "Predicting Student Performance Using Artificial Neural Network: in the Faculty of Engineering and Information Technology." International Journal of Hybrid Information Technology 8(2): 221-228.
60. Saleh, A. et al. (2020). "Brain Tumor Classification Using Deep Learning." 2020 International Conference on Assistive and Rehabilitation Technologies (iCareTech). IEEE, 2020.
61. Almadhoun, H. et al. (2021). "Classification of Alzheimer's Disease Using Traditional Classifiers with Pre-Trained CNN." International Journal of Academic Health and Medical Research (IJAHMR) 5(4):17-21.
62. El-Habil, B. et al. (2022). "Cantaloupe Classification Using Deep Learning." International Journal of Academic Engineering Research (IJAER) 5(12): 7-17.
63. Alkahlout, M. A. et al. (2022). "Classification of Fruits Using Deep Learning." International Journal of Academic Engineering Research (IJAER) 5(12): 56-63.
64. Alfarra, A. H. et al. (2022). "Classification of Pineapple Using Deep Learning." International Journal of Academic Information Systems Research (IJAISR) 5(12): 37-41.
65. Al-Masawabe, M. M. et al. (2022). "Papaya maturity Classification Using Deep Convolutional Neural Networks." International Journal of Engineering and Information Systems (IJEAIS) 5(12): 60-67.
66. Abu-Jamie, T. N., et al. (2022). "Six Fruits Classification Using Deep Learning." International Journal of Academic Information Systems Research (IJAISR) 6(1): 1-8.
67. Aish, M. A., et al. (2022). "Classification of pepper Using Deep Learning." International Journal of Academic Engineering Research (IJAER) 6(1): 24-31.
68. Alsaqqa, A. H., et al. (2022). "Using Deep Learning to Classify Different types of Vitamin." International Journal of Academic Engineering Research (IJAER) 6(1): 1-6.
69. MAlnajjar, M. K. and S. S. Abu-Naser (2022). "Heart Sounds Analysis and Classification for Cardiovascular Diseases Diagnosis using Deep Learning." International Journal of Academic Engineering Research (IJAER) 6(1): 7-23.
70. Aldammagh, Z., Abdeljawad, R., & Obaid, T. (2021). Predicting Mobile Banking Adoption: An Integration of TAM and TPB with Trust and Perceived Risk. Financial Internet Quarterly, 17(3), 35-46.
71. Obaid, T. (2021). Predicting Mobile Banking Adoption: An Integration of TAM and TPB with Trust and Perceived Risk. Available at SSRN 3761669.
72. Jouda, H., Abu Jarad, A., Obaid, T., Abu Mdallalah, S., & Awaja, A. (2020). Mobile Banking Adoption: Decomposed Theory of Planned Behavior with Perceived Trust. Available at SSRN 3660403.
73. Obaid, T., Abdaljawad, R., & Mdallalahc, S. A. (2020). Factors Driving E-Learning Adoption In Palestine: An Integration of Technology Acceptance Model And IS Success Model. Available at SSRN 3686490.
74. Obaid, T. F., & Eneizan, B. M. (2016). Transfer of training and post-training on job performance in Middle Eastern countries. Review of Public Administration and Management, 400(3786), 1-11.
75. Obaid, T. F., Zainon, M. S., Eneizan, P. D. B. M., & Wahab, K. A. TRANSFER OF TRAINING AND POST-TRAINING ON JOB PERFORMANCE IN MIDDLE EASTERN COUNTRIES.