Detection and Classification of Skin Cancer Using Deep Convolutional Neural Networks (CNN) via KNIME Analytics Platform Software

Ahmad Mohamad El-fallah Ismail
dr.ahmad_ismail_alrabty@yahoo.com
Department of Electricity and Electronics, College of Engineering, University of Gharian, Libya

ABSTRACT

The use of technologies from many fields, such as mass spectrometry, next-generation sequencing, or image processing, is common in experiments in the life sciences. Complex scripts are frequently used to govern data flow, data transformation, and statistical analysis when passing data between such tools. Such scripts not only tend to be platform dependant, but also tend to expand as the experiment goes on and are rarely clearly documented, which makes the experiment harder to reproduce. Workflow systems like KNIME Analytics Platform, which offers a platform for graphically linking tools and ensures the same results across various operating systems, aim to address these issues. systems that are frequently employed in the biological sciences and describe how they compare and contrast with KNIME. KNIME is an open source program that enables programmers and scientists to share their own extensions with the scientific community. The unified data model of KNIME allows for interoperability, and we describe a few additions from the life sciences that make it easier to explore, analyze, and visualize data. In addition, we mention additional workflow. According to the American Cancer Society, skin cancer is the most prevalent form of malignancy in humans. It is typically identified visually, with first clinical screenings, dermoscopic (skin-related) analysis, a biopsy, and histological examinations as potential follow-up steps. Errors (mutations) in the DNA of skin cells are the cause of skin cancer. The cells proliferate out of control and aggregate into a mass of cancer cells as a result of the mutations. In this paper, convolutional neural networks are used to attempt to categorize photos of skin lesions. The deep neural networks demonstrate enormous potential for classifying images while taking into account the extreme environmental heterogeneity. Due to the current state of technology, it is imperative to use machines rather than people to address the widespread problem of skin cancer. One of the best ways to address skin cancer issues is deep learning. Huge data, virtual reality, augmented reality, and mini-services are all used in the new research area of deep learning in contemporary technology. The advent of powerful arithmetic capabilities enabled deep learning applications using Mobile net (CNN) to revolutionize image classification. The various forms of skin cancer can be categorized using deep learning. Transfer Learning was used during the training on many models. The model's best level of accuracy was over 77.333 %. To guarantee the validity and reproducibility of the aforementioned result, the dataset employed is openly accessible.

Keywords: Artificial Intelligence (AI), Convolutional Neural Networks (CNNs), Deep Learning (DL), Skin Cancer, Image Classification, Knime Analytics Platform Software etc.

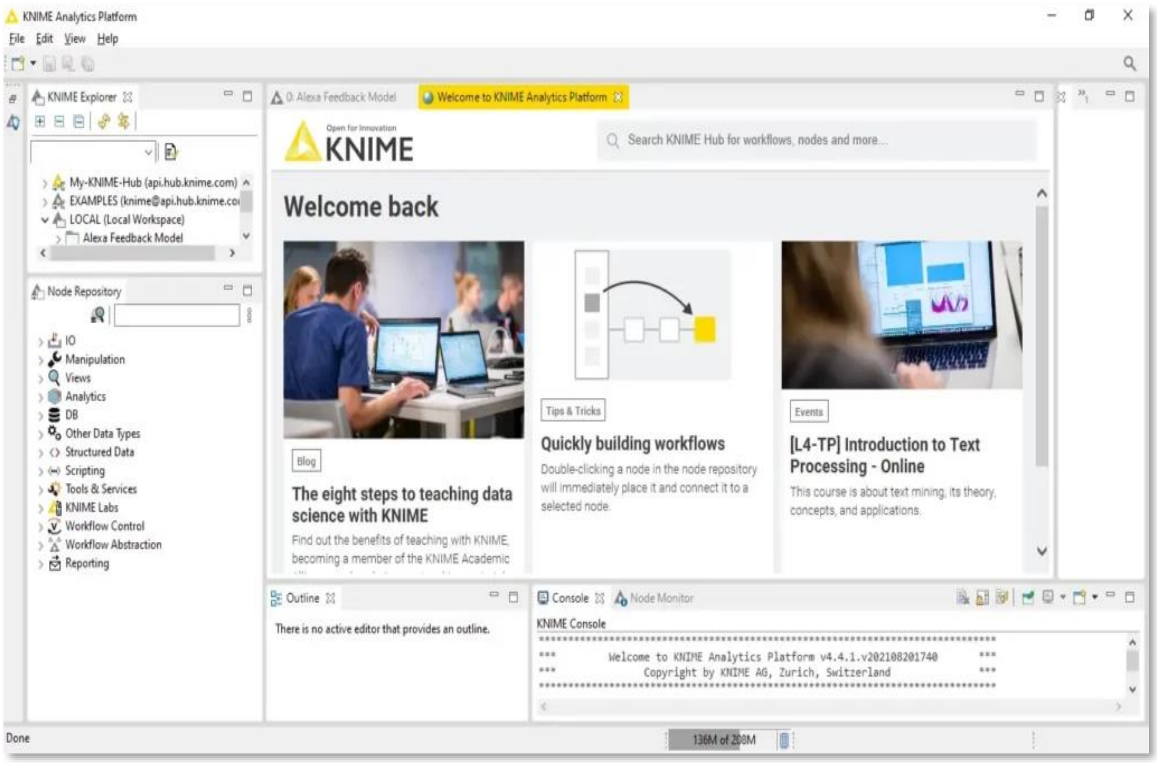
I. INTRODUCTION

Cancer is a disease in which cells in the body grow out of control. When cancer starts in the skin, it is called skin cancer. Skin cancer is the most common cancer in the United States. Some people are at higher risk of skin cancer than others, but anyone can get it. The most preventable cause of skin cancer is overexposure to ultraviolet (UV) light, either from the sun or from artificial sources like tanning beds [1]. About more than a million skin cancer cases occurred in 2018 globally . Skin cancer is one of the fastest-growing diseases in the world. Skin cancer occurs mainly due to the exposure of ultraviolet radiation emitted from the Sun. Considering the limited availability of the resources, early detection of skin cancer is highly important. Accurate diagnosis and feasibility of detection are vital in general for skin cancer prevention policy. Skin cancer detection in early phases is a challenge for even the dermatologist. In recent times, we have witnessed extensive use of deep learning in both supervised and unsupervised learning problems. One of these models is Convolution Neu-ral Networks (CNN) which has outperformed all others for object recognition and object classification tasks. CNNs eliminate the obligation of manually handcrafting features by learning highly discriminative features while being trained end-to-end in a supervised manner. Convolutional Neural Networks have recently been used for the identification of skin cancer lesions. Several CNN models have successfully outperformed trained human professionals in classifying skin cancers. Several methods like transfer learning using large datasets have further improved the accuracy of these models. VGG-16 is a convolutional neural system that is prepared on more than a million pictures from the ImageNet database. The system is 16 layers profound and can arrange pictures into 1000 item classifications, for example, console, mouse, pencil, and numerous creatures. Accordingly, the system has learned rich component portrayals for a wide scope of pictures. The system has a picture info size of 224-by-224. The model accomplishes 92.7% top-5 test precision in ImageNet, which is a dataset of more than 14 million pictures having a place with 1000 classes. In this paper, we have generated a CNN model that analyses the skin pigment lesions and classifies them in using a publicly available dataset by employing various deep learning techniques. We have improved the accuracy of classification by implementing CNN and transfer learning models. We have tested our model using the publicly available HAM-10000 dataset [2]. Deep learning is used successfully in many data science applications, such as image processing, text processing, and fraud detection. KNIME offers an integration to the Keras libraries for deep learning, combining the codeless ease of use of KNIME Analytics Platform with the extensive coverage of deep learning paradigms by the Keras libraries. Though codeless, implementing deep learning networks still requires orientation to distinguish between the learning paradigms, the feedforward multilayer architectures, the networks for sequential data, the encoding required for text data, the convolutional layers for image data, and so on [3]. To create a CNN-based model that accurately detects melanoma, the issue Melanoma is a form of cancer that, if not caught in time, can be fatal. 75% of skin cancer deaths are attributable to it. A system that can analyze photos and notify dermatologists of the existence of melanoma has the potential to greatly minimize the amount of manual work required for diagnosisThe goal of the CNN skin cancer project is to create a deep learning model that is capable of accurately identifying and diagnosing skin cancer. The key to effective therapy for skin cancer, one of the most prevalent cancers, is early detection. An automated system can be developed that can rapidly and accurately detect probable skin malignancies by employing convolutional neural networks (CNNs). This might lower the incidence of missed diagnosis, enhance patient outcomes, and lower the price tag on healthcare related to treating skin cancer. Therefore, the objective of this paper is to study deep learning in order to correctly classify skin cancer types, Therefore, the main objectives to be achieved in this paper are: to locate a dataset for classifying skin cancer using an existing dataset, to create a deep learning model to accurately categorize various forms of skin cancer and to assist physicians and regular users in identifying common symptoms and signs of many types of cancer. For a preliminary evaluation of cancer and the determination of its initiating factors. This paper is organized as follows. Mathematical Modeling of the DC Motor is given in Sec. II. Design Procedure of Compensators using Frequency Response Method (Bode Plot) is given in Sec. III. Analysis of Simulation Results is given in Sec. IV. Conclusion is demonstrated in Sec. V.

II. MATERIALS AND METHODS

1. Knime Analytics Platform Software

KNIME Analytics Platform is a free open source program for creating applications and reports in data science. This software helps you discover what's hidden in your data, find new insights, or anticipate specific events you weren't aware of in the past. Why is KNIME used? KNIME provides the graphical user interface for easily building machine learning models. Contains a set of nodes used for various functions to build a workflow. A number of nodes exist in their repository for specific purposes to build automated learning or workflow models such as data connectivity, data read/browse, etc. The workflow structure is then used to perform analyzes on the data. In addition to that, the workflow system KNIME is an open source software that aims to solve these problems by providing a platform that can be extended easily with new tool integrations, has a strongly-typed data system and allows workflow creators to document
the steps performed by the workflow in detail. [4].



Find out the benefits of teaching with KNIME, becoming a member of the KNIME Academic

Double-clicking a node in the node repository will immediately place it and connect it to a selected node.

This course is about text mining, its theory, concepts, and applications.

There is no active editor that provides an outline.

```
***********************************************************************
***        Welcome to KNIME Analytics Platform v4.4.1.v202108201740      ***
***             Copyright by KNIME AG, Zurich, Switzerland               ***
***********************************************************************
```

Figure 1. The program interface

2. CNN Network Architecture

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing. The term ‘Convolution” in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other. In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image. Basic Architecture There are two main parts to a CNN architecture:

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction.
- The network of feature extraction consists of many pairs of convolutional or pooling layers.
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.
- This CNN model of feature extraction aims to reduce the number of features present in a dataset. It creates new features which summarises the existing features contained in an original set of features. There are many CNN layers as shown in the CNN architecture diagram.

In deep learning, CNNs are the most common networks used with image classification. CNNs
were inspired by the human visual system proposed by Fukushima [5] and LeCun et al. [6]. They are state-of-the-art approaches for pattern recognition, object detection, and many other image applications. In particular, in 2012, the champion of the ImageNet Large Scale Visual Recognition Challenge 2012 competition [7,8] was a deep CNN solution by Krizhevsky et al. [9] which demonstrated the great power of deep CNNs. CNNs are very different from other pattern recognition algorithms in that CNNs combine both feature extraction and classification [10]. Figure 2 shows an example of a simple schematic representation of a basic CNN. This simple network consists of five different layers: an input layer, a convolution layer, a pooling layer, a fully-connected layer, and an output layer. These layers are divided into two parts: feature extraction and classification. Feature extraction consists of an input layer, a convolution layer, and a pooling layer, while classification consists of a fully-connected layer and an output layer. The input layer specifies a fixed size for the input images, which is resized if needed. Then the image is convolved with multiple learned kernels using shared weights by convolution layer. Next, the pooling layer reduces the image size while trying to maintain the contained information. The outputs of the feature extraction are known as feature maps. The classification combines the extracted features in the fully-connected layers. Finally, there exists one output neuron for each object category in the output layer. The output of the classification part is the classification result. In real applications, the feature extraction part contains many convolution and pooling layers, and the classification part also contains many fully connected layers

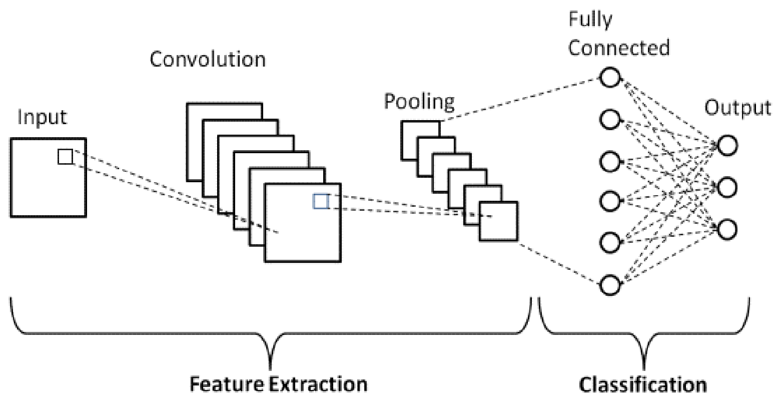


Figure 2. Basic CNN Architecture [11]

With this neural network, the type of skin cancer will be easily classified and discovered by KNIME Software.

2. Images
The skin anatomy has two main layers: epidermis (the outer layer) and dermis (the inner layer).The epidermis is made up of flat, scale-like cells called Squamous cells, and round cells called basal cells. The lower part of the epidermis contains melanocytes. Melanocytes are pigment cells that are found in the lower part of the epidermis and produce melanin, the pigment that gives skin its natural color [12]. When the skin is exposed to the sun, melanocytes produce more pigment, causing the skin to be bronzed, or darken which cause melanoma. Melanoma can be of Benign or Malignant [13] see Figure 3.
The abnormal growths of melanocytes cause malignant melanoma cancer which invades or spreads to other parts of the body without normal controls, Malignant melanoma divided into Superficial Spreading Melanoma (constitutes about 75%of all non-melanomas), Nodular Melanoma (constitutes about 15% of all non-melanomas), Lentigo malignant Melanoma (constitutes about 10%of all non-melanomas) and Acral Lentiginous Melanoma (constitutes about 5% of all melanomas) [14] see Figure 4.

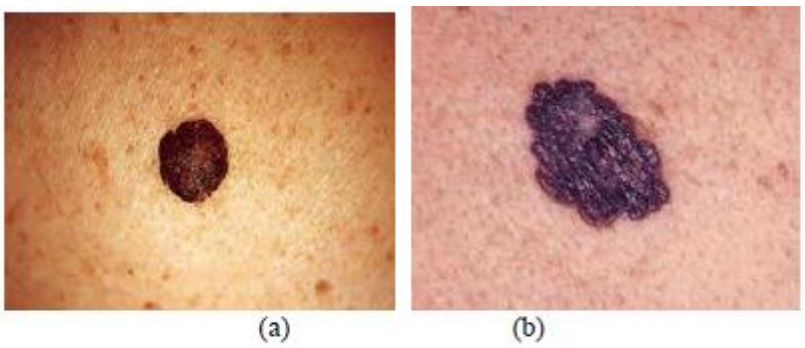


Figure 3. Melanoma. (a) Benign. (b) Malignant

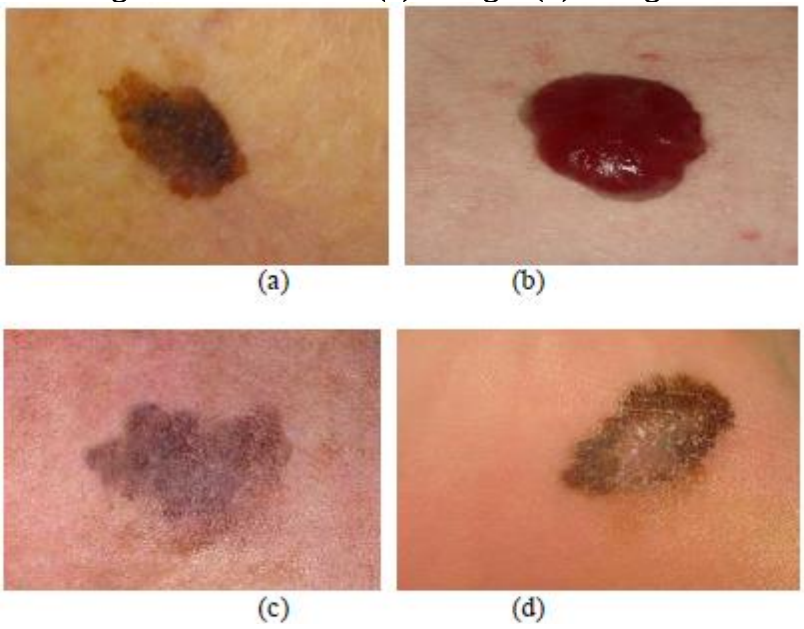


Figure 4. Malignant Melanoma Types. (a) Superficial Spreading Melanoma. (b) Nodular Melanoma (c) Lentigo malignant Melanoma. (d) Acral Lentiginous Melanoma.

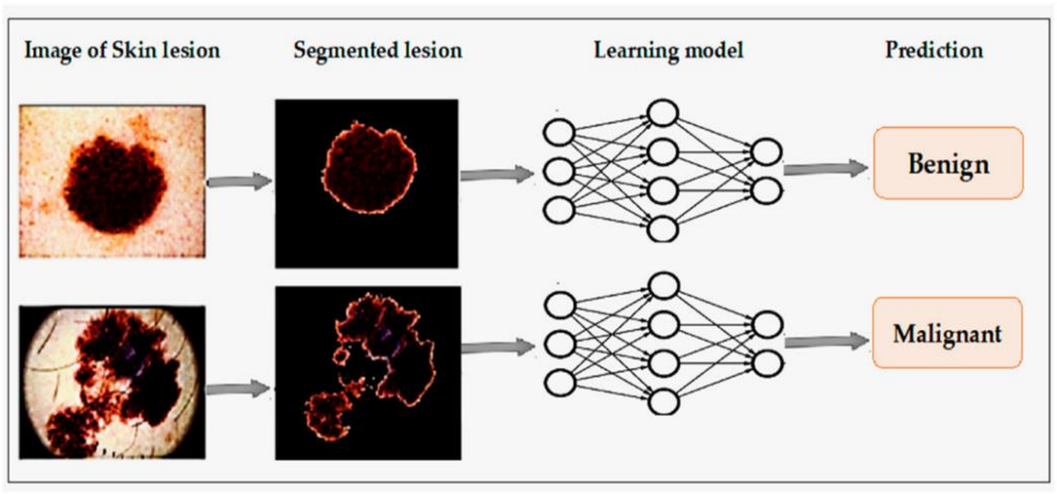
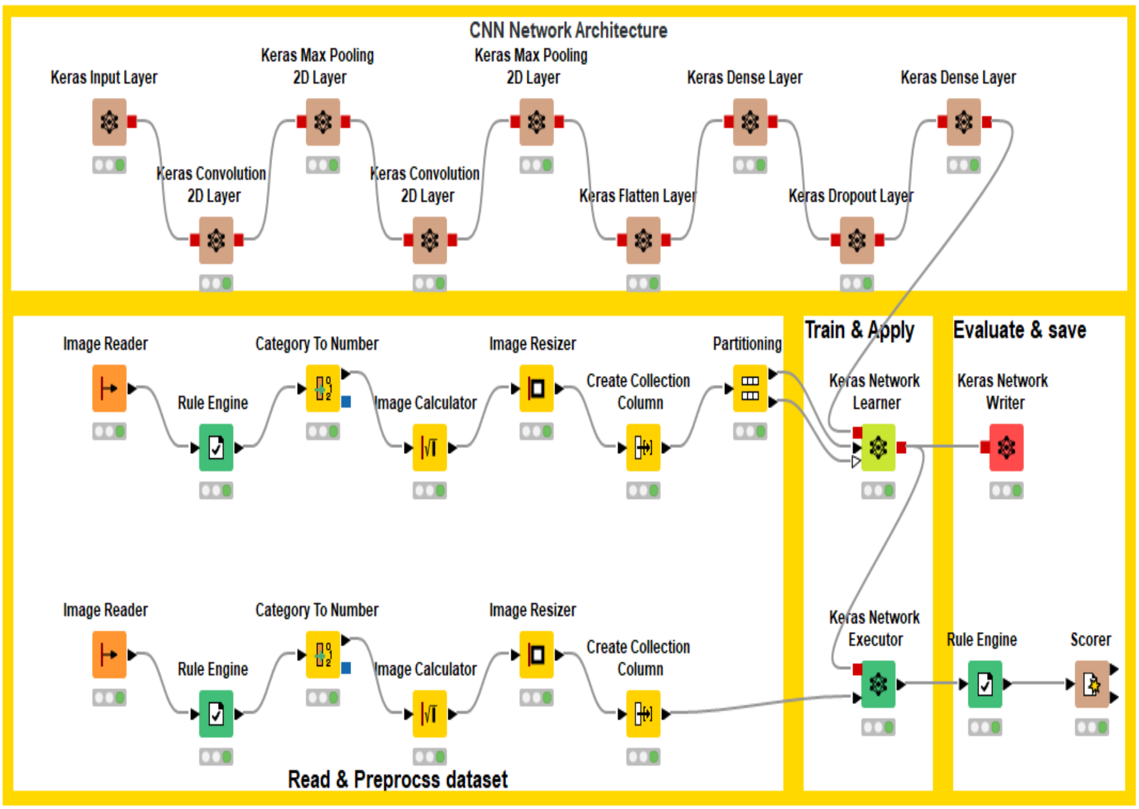


Figure 2. Classification Disease type [15]

4. Method

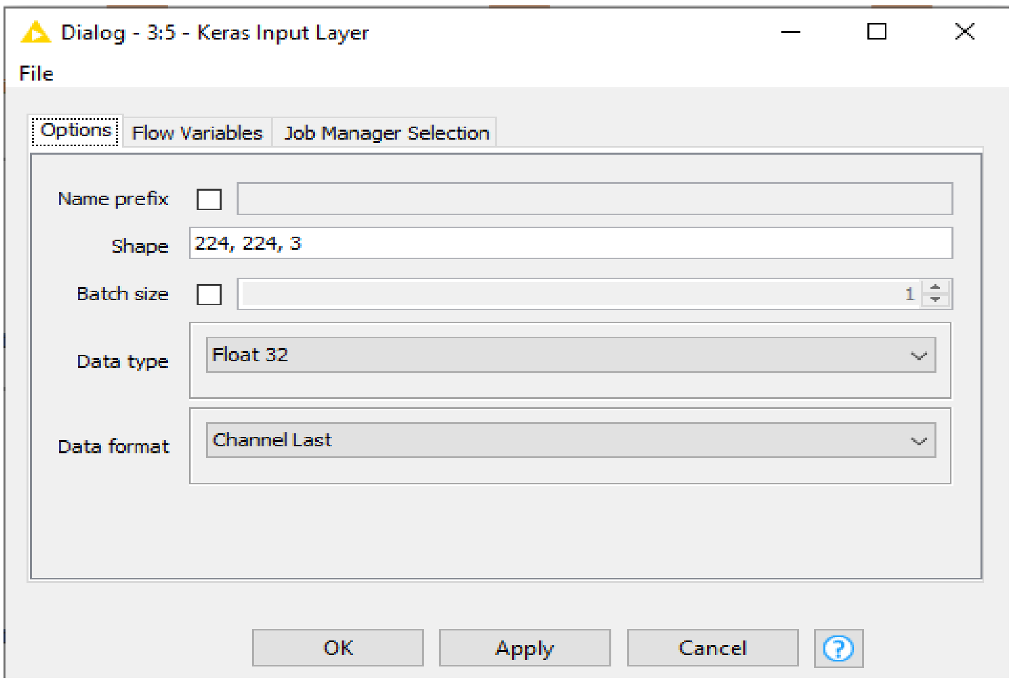
Figure 3. KERAS line for cancer detection



4.1 Keras Input Layer

It is the first layer in the Keras CNN Network Architecture model. They are used to create a Keras tensor instance so that it can be used as input to the model. This layer takes raw data (images), usually in the form of heterogeneous matrices, and creates a tensor representation of the data. This tensor is then passed through the rest of the layers in the form.

Procedures: Shape (224,224,3).



Dialog - 3:5 - Keras Input Layer

File

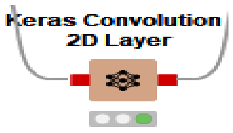
Options | Flow Variables | Job Manager Selection

Name prefix	☐
Shape	224, 224, 3
Batch size	☐ 1
Data type	Float 32
Data format	Channel Last

OK | Apply | Cancel

Figure 4. Keras input layer features

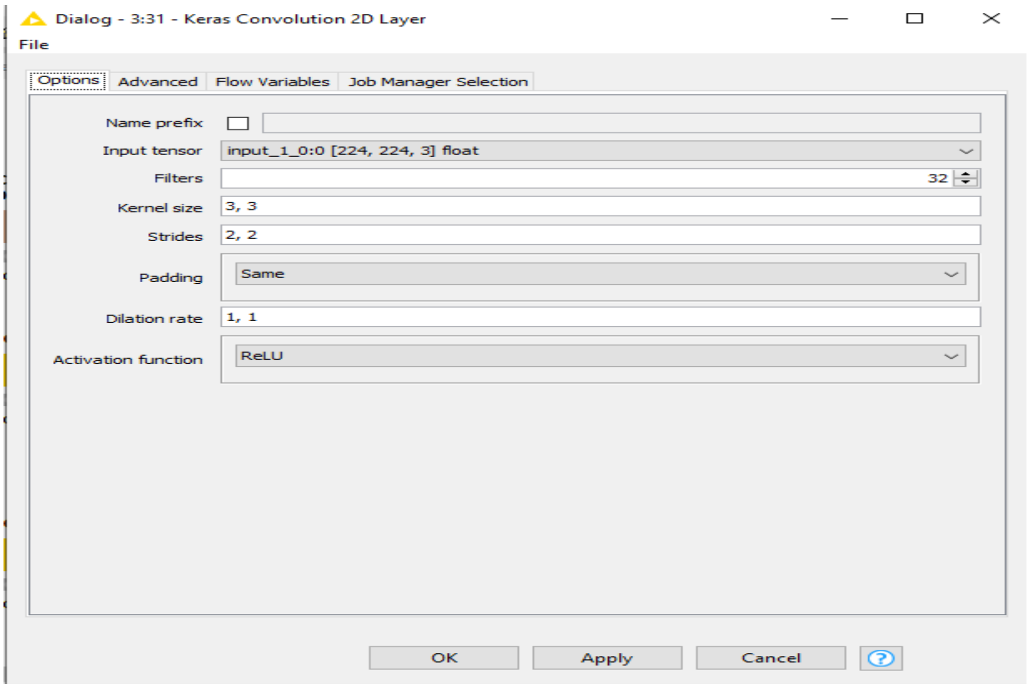
4.2 Keras Convolution 2D layer



This layer creates a convolution kernel that is convolved with the layer input over two dimensions. The convolution kernel produced by this layer is convolved with the layer input to form a tensor of outputs. An output bias vector is constructed and appended if use bias is set to true. Also applied to the outputs if activation is set to a value other than None is the last clause. The Keras Convolution 2D layers is connected to a Keras Input Layer in order to apply a 2D convolution to the previous input, which is a set of images. This layer is used to detect features in the input and extract them as a set of filters. A layer consists of several parameters, such as the number of filters, filter size, pitch, and padding. The output of this layer is a feature map that can be used for further processing.

Procedures:

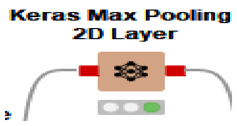
- Filter: 32.
- Kernel size: [3*3].
- Strides: [2,2].
- Padding : same .
- Activation Function : ReLU.


Dialog - 3:31 - Keras Convolution 2D Layer
File
Options | Advanced | Flow Variables | Job Manager Selection

Name prefix	☐
Input tensor	input_1_0:0 [224, 224, 3] float
Filters	32
Kernel size	3, 3
Strides	2, 2
Padding	Same
Dilation rate	1, 1
Activation function	ReLU

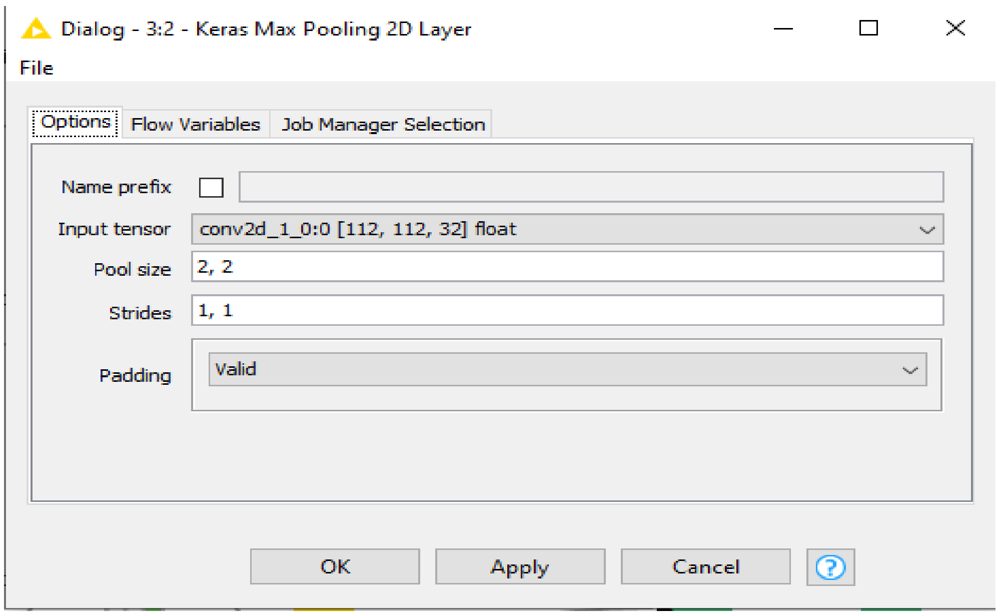
Figure 5. Keras convolution 2D layer features

4.3 Keras Max pooling 2D Layer



This layer applies max pooling in two dimensions . Maximum pooling for 2D spatial data. The input is downsampled along its spatial dimensions (height and width) by taking the maximum value for each input channel over an input window with a size determined by pool size. Steps are taken to move the window along **each dimension. Keras Max Pooling 2D Layer is connected to Keras Convolution 2D Layer, and this layer is used for pooling to perform maximum pooling of images. This is to reduce the size of the input by dividing it into rectangular aggregation regions and taking the maximum value from each region.**
Procedures:

- Pool size:[2*2].
- Strides:[1,1].
- Padding : Valid.



Name prefix ☐

Input tensor: conv2d_1_0:0 [112, 112, 32] float

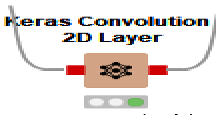
Pool size: 2, 2

Strides: 1, 1

Padding: Valid

Figure 6. Keras max poolimg 2D layer features

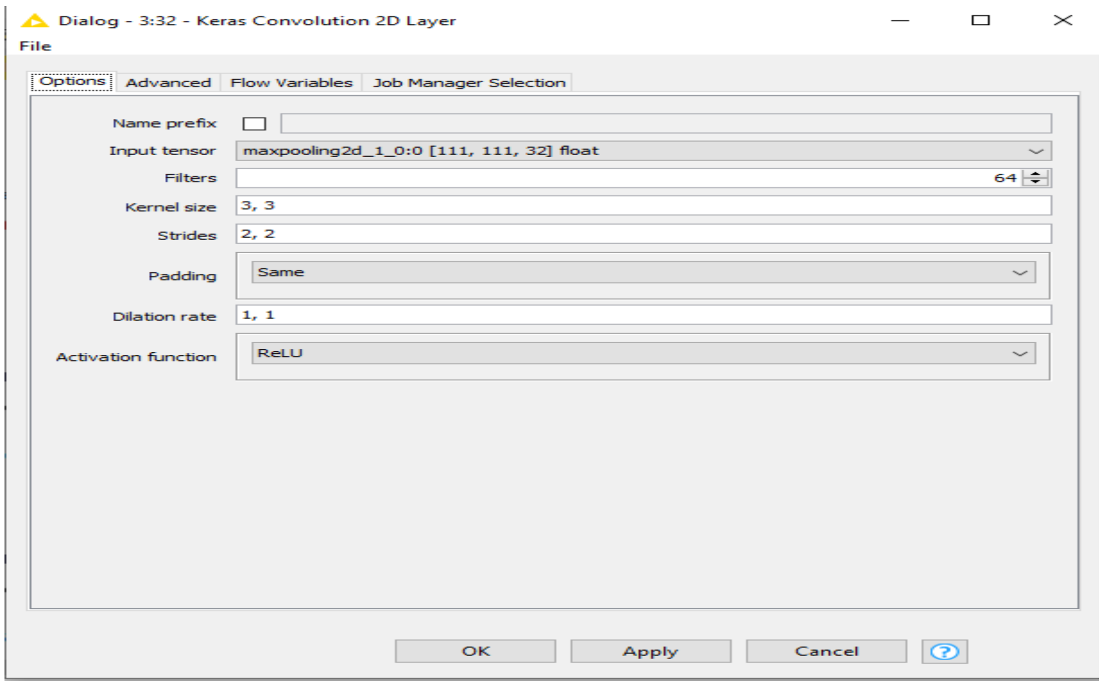
4.4 Keras Convolution 2D Layer



The Keras Convolution 2D layers is connected to a Keras Max pooling 2D layer in order to apply a 2D convolution to the previous entries, which are a set of images. We use this layer to detect features in the input and extract them as a set of filters. The layer consists of several parameters, such as the number of filters, filter size, slope, and padding. The output of this layer is a feature map that can be used for further processing.

Procedures:

- Filter :64.
- Kernel size [3*3].
- Strides [2,2].
- Padding :same.
- Activation Function : ReLU.

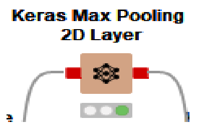

Dialog - 3:32 - Keras Convolution 2D Layer
File
Options | Advanced | Flow Variables | Job Manager Selection

Name prefix: ☐
Input tensor: maxpooling2d_1_0:0 [111, 111, 32] float
Filters: 64
Kernel size: 3, 3
Strides: 2, 2
Padding: Same
Dilation rate: 1, 1
Activation function: ReLU

OK | Apply | Cancel

Figure 7. Keras convolution 2D layer features

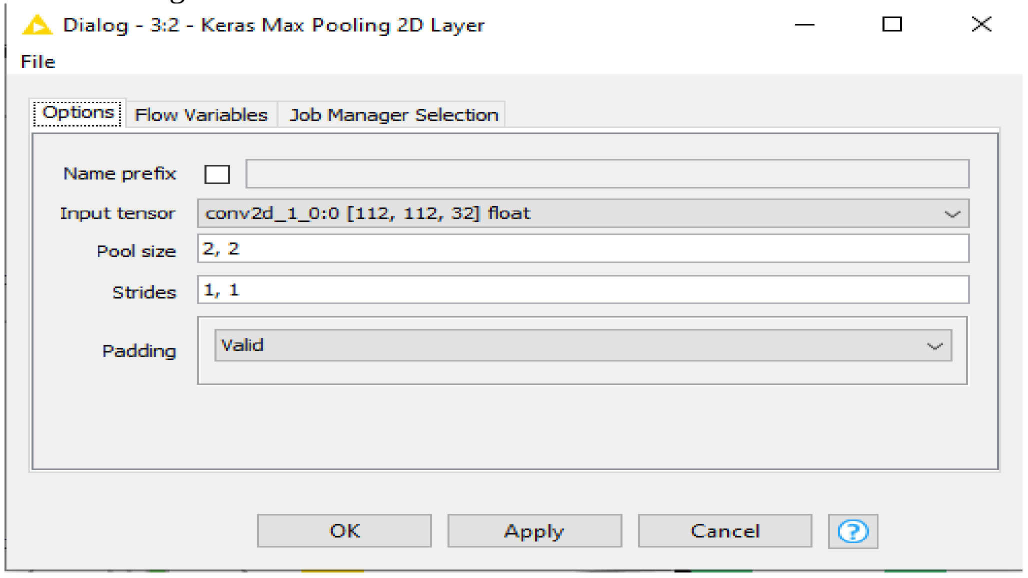
4.5 Keras Max pooling 2D Layer



This layer applies maximum pooling in two dimensions. Maximum 2D spatial data aggregation. Inputs are scaled down along their spatial dimensions (height and width) by taking the maximum value of each input channel through an input window with a size determined by the pool size. Steps are taken to move the window along each dimension.

Procedures:

- Pool size:[2*2].
- Strides:[1,1].
- Padding : Valid.

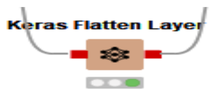

Dialog - 3:2 - Keras Max Pooling 2D Layer
File
Options | Flow Variables | Job Manager Selection

Name prefix: ☐
Input tensor: conv2d_1_0:0 [112, 112, 32] float
Pool size: 2, 2
Strides: 1, 1
Padding: Valid

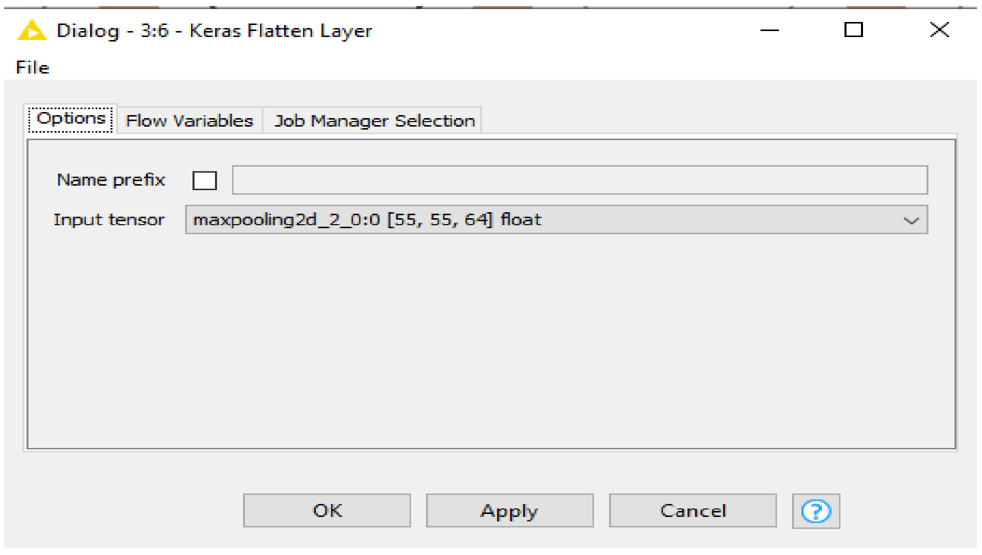
OK | Apply | Cancel

Figure 10. Keras max poolimg 2D layer features

4.6 Keras Flatten Layer



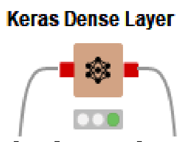
Flatten is used to flatten the input. For example, if flatten is applied to layer having input shape as (batch_size, 2,2), then the output shape of the layer will be (batch_size, 4). The Keras Flatten Layer is connected to the Keras Max Pooling 2D Layer, which we use to flatten the inputs into a one-dimensional array. That is, the input is converted into a single vector, which is then used as the input for a fully connected layer.



Name prefix ☐
Input tensor: maxpooling2d_2_0:0 [55, 55, 64] float

Figure 11. Keras flatten layer features

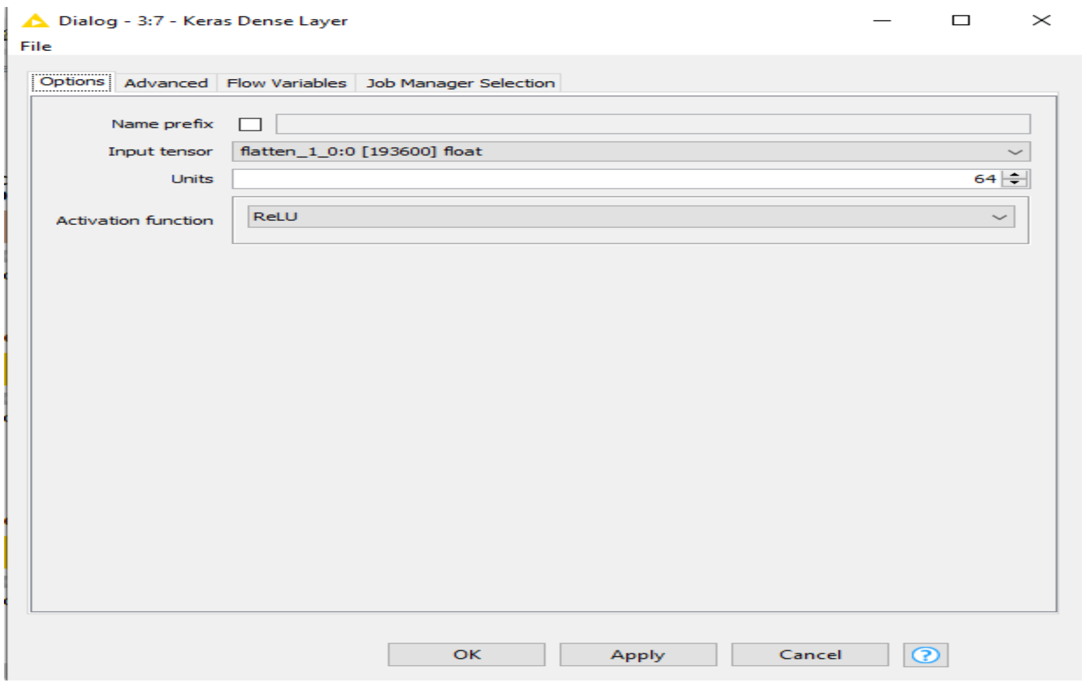
4.7 Keras Dense Layer



Keras Dense layer is the layer that contains all the neurons that are deeply connected within themselves. This means that every neuron in the dense layer takes the input from all the other neurons of the previous layer. It is possible to add as many dense layers as required. It is one of the most commonly used layers. Keras Dense layers plugged into a Keras Flatten Layer, which we used to learn nonlinear combinations of input and output features. The layer contains weights learned during the training process, which allows it to make predictions based on the input data.

Procedures:

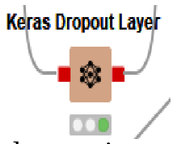
- Units :64.
- Activation Function :ReLU.


Dialog - 3:7 - Keras Dense Layer
File
Options | Advanced | Flow Variables | Job Manager Selection

Name prefix: ☐
Input tensor: flatten_1_0:0 [193600] float
Units: 64
Activation function: ReLU

Figure 12. Keras dense layer features

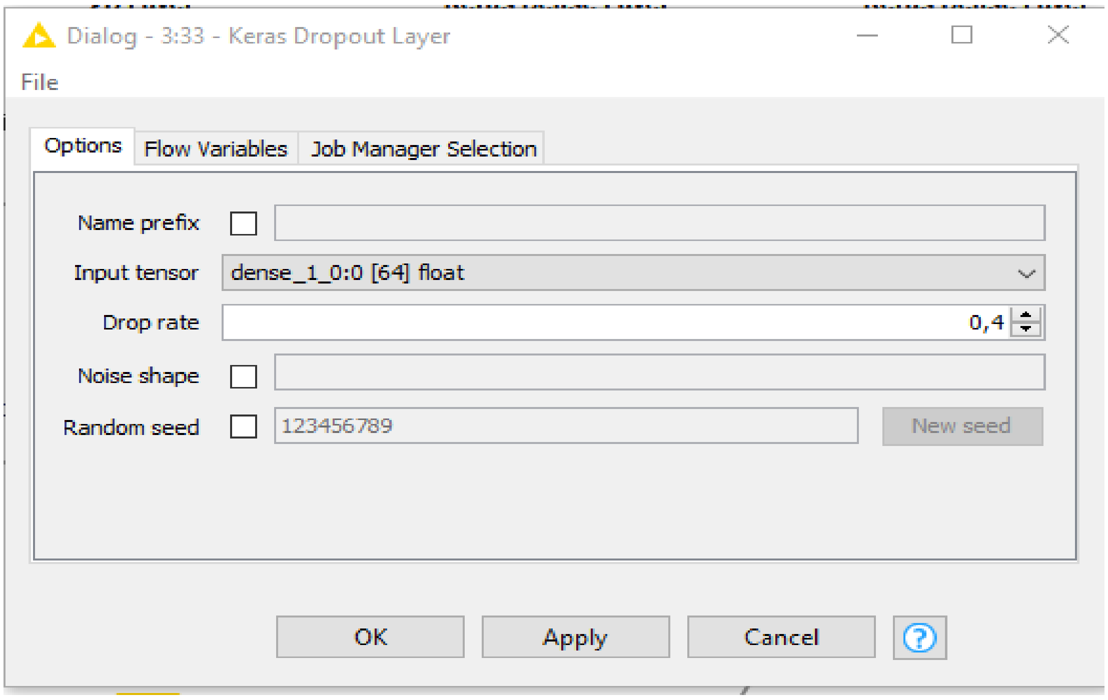
4.8 Keras Dropout Layer



The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.The Keras Dropout layer is connected to the Keras Dense Layer, which is a regularization technique used to reduce overfitting in neural networks. It randomly sets the input units to 0 with a certain probability (called the dropout rate) on each update during training, which helps prevent neurons from co-adapting too much.

Procedures:

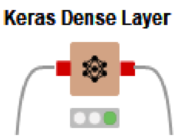
- Drop rate :0,4.
- There is no Noise.


Dialog - 3:33 – Keras Dropout Layer
File
Options | Flow Variables | Job Manager Selection

Name prefix	☐
Input tensor	dense_1_0:0 [64] float
Drop rate	0,4
Noise shape	☐
Random seed	☐ 123456789

Figure 13. Keras dropout layer feature

4.9 Keras Dense Layer



A densely connected layer that connects each unit of the layer input with each output unit of this layer. Corresponds to the Keras Dense Layer. The Keras Dense layers are connected to the Keras Dropout layer, which it is be used to learn nonlinear combinations of input and output features. The layer contains weights that were learned during the training process, allowing it to make predictions based on the input data, and with that, the disease convolutional neural network have been finished creating.

Procedures :

- Units :2.
- Activation Function : softmax.

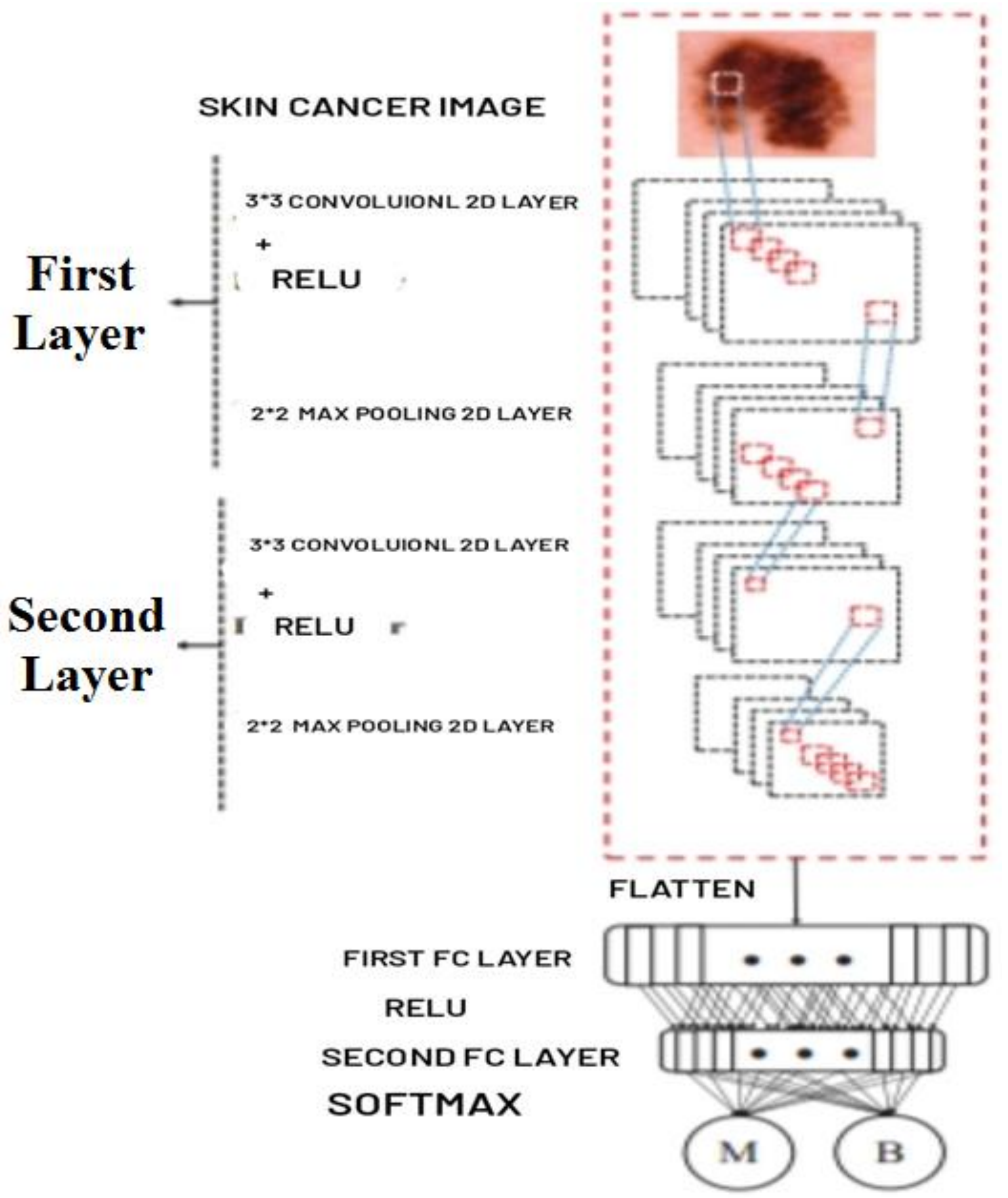


Figure 14. CNN which is designed to train a model for classifying the skin cancer.

IV. ANALYSIS OF SIMULATION RESULTS

Simulation is the imitation of the operation of a process or system in the real world over time. Simulations require the use of models; The model represents the main characteristics or behaviors of the chosen system or process, while the simulation represents the evolution of the model over time. Often, computers are used to perform simulations. This is what will be done in this chapter, in which the method of operating the program will be explain and presenting the results obtained.

- The System: windows 10.
- Programming Language: Python.

In this project, 600 test images were used, 2200 images were used for training, and they were divided into 20% validation images for training, and 80% for training. Where the user images from the Kaggle dataset are of two categories:

- Benign tumor.
- Malignant tumor.

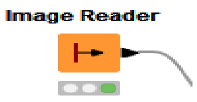
Those images are trained using CNN model with different optimizer methods like: ReLU function, drop rate: 0.4, No noise, filters, soft max, etc. The performance criteria measured in this study are:

- Accuracy
- Wrong classified
- Error Cohen's kappa(x)

After training with 20 repetitions (epoch), Batch Size: 16, Accuracy comparison and suggested performance loss can be observed.

1. The Training

2.1 Image Reader



It reads images from various file formats supported by the Bio-Formats library and imports them into the internal KNIME image format. Image readers are used to view and edit digital images in matrix form, so the network training was worked with about 1100 images of benign tumors and 1100 images of malignant tumors.

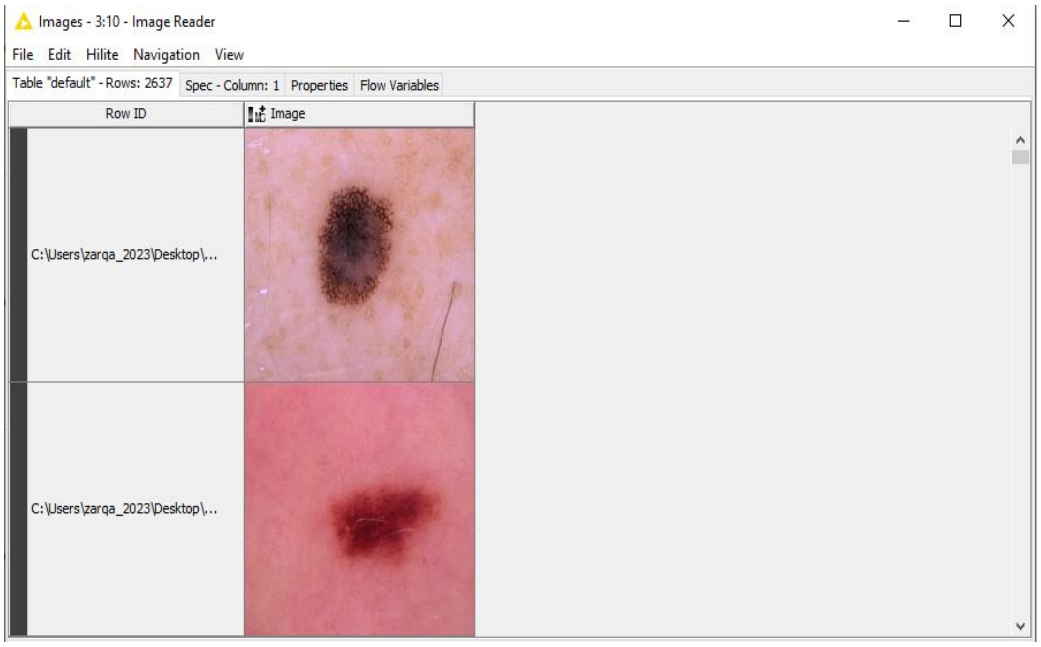

Images - 3:10 - Image Reader
File Edit Hilite Navigation View
Table "default" - Rows: 2637
Spec - Column: 1
Properties
Flow Variables
Row ID
Image
C:\Users\zarqa_2023\Desktop\...
C:\Users\zarqa_2023\Desktop\...

Figure 15. Image reader

2.2 Rule Engine

A rule engine is a software system that automates the process of making decisions based on a set of rules. where a condition for translating the images in the new matrix is specified, so that for each satisfactory group, a specific value is encoded denoting it (by appending class coins).

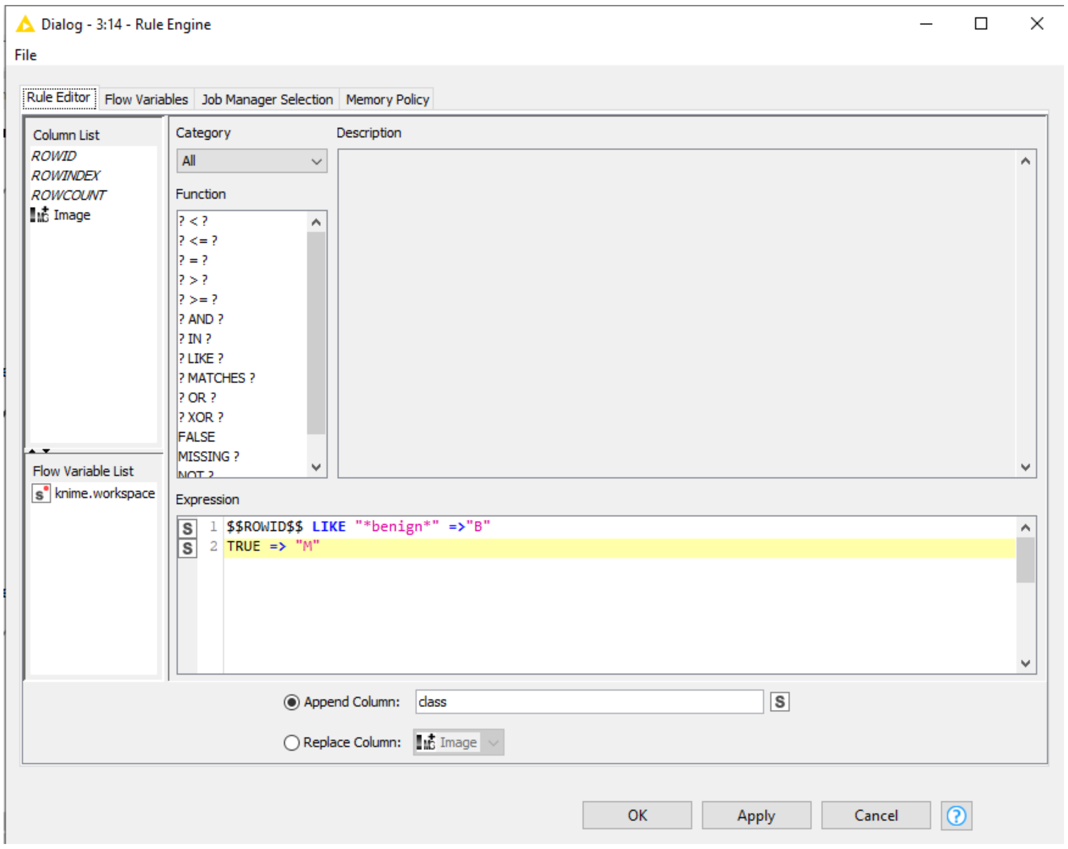
Procedures :

The condition:

$$ROWID$$ LIKE "*benign*" =>"B"
TRUE => M

Where:

M: Malicious.
B: Benign.

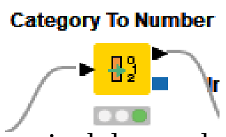


```
$$ROWID$$ LIKE "*benign*" =>"B"
TRUE => "M"
```

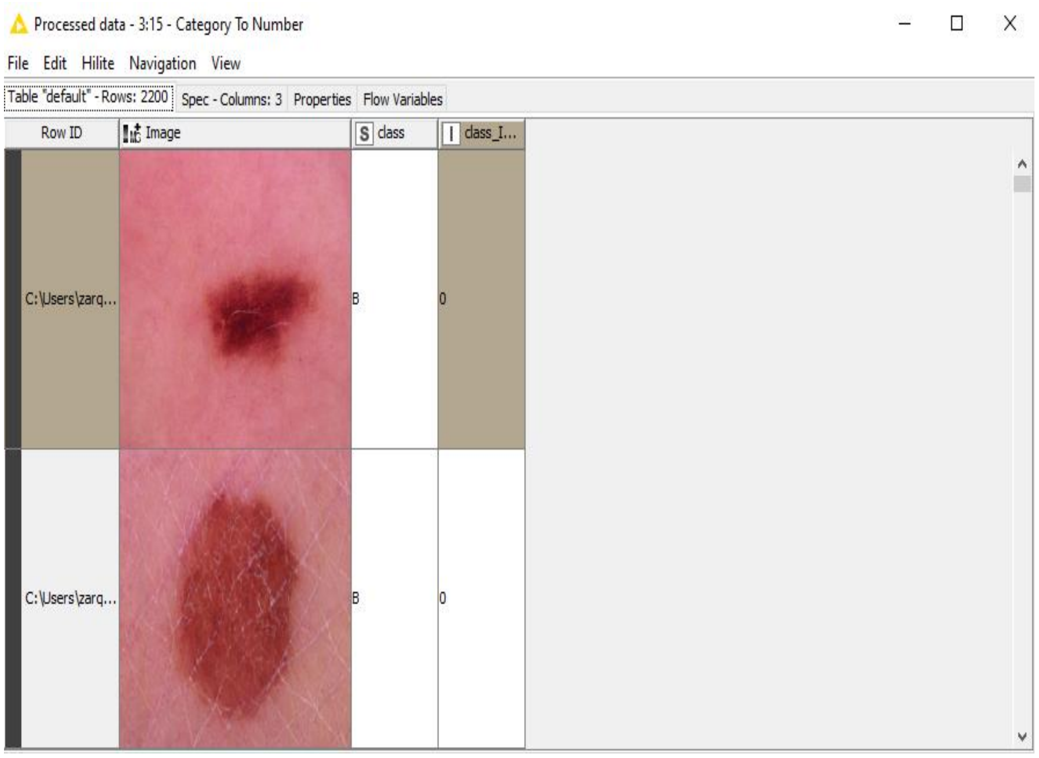
(●) Append Column: class

Figure 16. **Condition to clarify the type of disease**

2.3 Category to Number



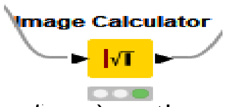
This node takes columns with nominal data and maps every category to an integer. For convenience, multiple columns can process with this node. However, these columns are processed separately as if you would use a single Category to Number node for every column. The process of assigning integer values to categories or labels for indexing is known as Category to Number. Numeric values are easier to work with than categorical labels. For example, a data set might contain a column of sex (B, M) classifications that can be converted to numerical values (0, 1).


Processed data - 3:15 - Category To Number
File Edit Hilite Navigation View
Table "default" - Rows: 2200 | Spec - Columns: 3 | Properties | Flow Variables

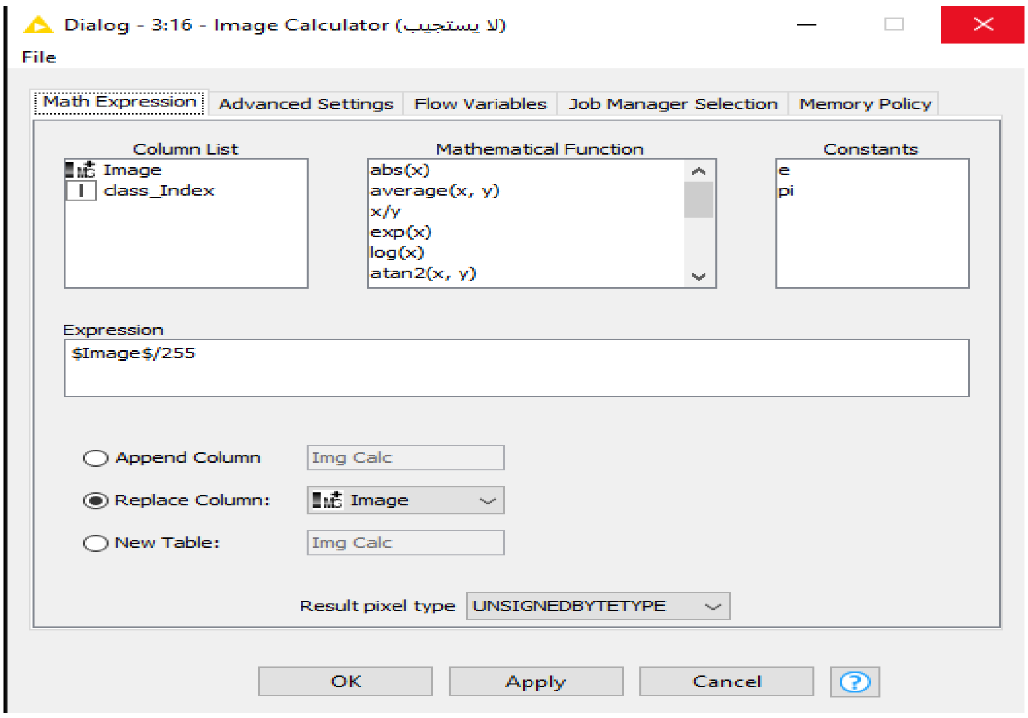
Row ID	Image	class	class_I...
C:\Users\zarq...		B	0
C:\Users\zarq...		B	0

Figure 17. Converting the type of each disease into numerical values (0,1)

2.3.1 Image Calculator



This node evaluates a (free-form) mathematical expression based on the images in a row. The computed results can be either appended as new column or be used to replace an input column. Available variables are the values in the corresponding row of the table (left list in the dialog). Commonly used functions are shown in the list "Mathematical Functions". Image Calculator is a software program that performs the normalization of images.
Procedures: $Image$/255


Dialog - 3:16 - Image Calculator (لا يستجيب)
File
Math Expression | Advanced Settings | Flow Variables | Job Manager Selection | Memory Policy

Column List	Mathematical Function	Constants
Image	abs(x)	e
class_Index	average(x, y)	pi
	x/y	
	exp(x)	
	log(x)	
	atan2(x, y)	

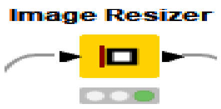
Expression
```
$Image$/255
```

Append Column: Img Calc
[✓ Replace Column:] Image
New Table: Img Calc

Result pixel type: UNSIGNEDBYTETYPE

Figure 18. Image calculator

2.3.2 Image Resizer



Resizes the image in each dimension. The resulting values at each position in the image are set according to the resizing strategy. Take in consideration that the input values for each dimension can be interpreted in several ways. Image Resizer is free software that allows users to quickly and easily resize digital images. So all the images are made to be at the same size.

Procedures : Resize to X =224,Y=224,Z=1,Channel=3.

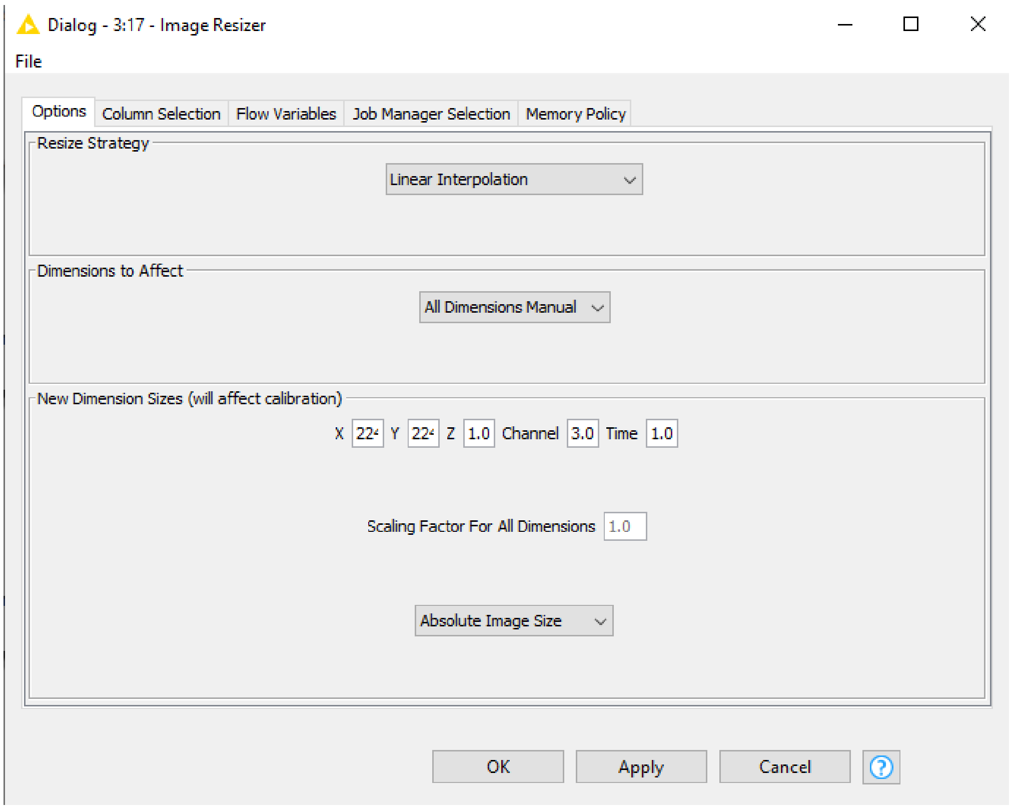
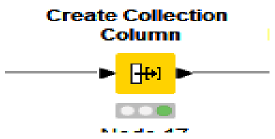
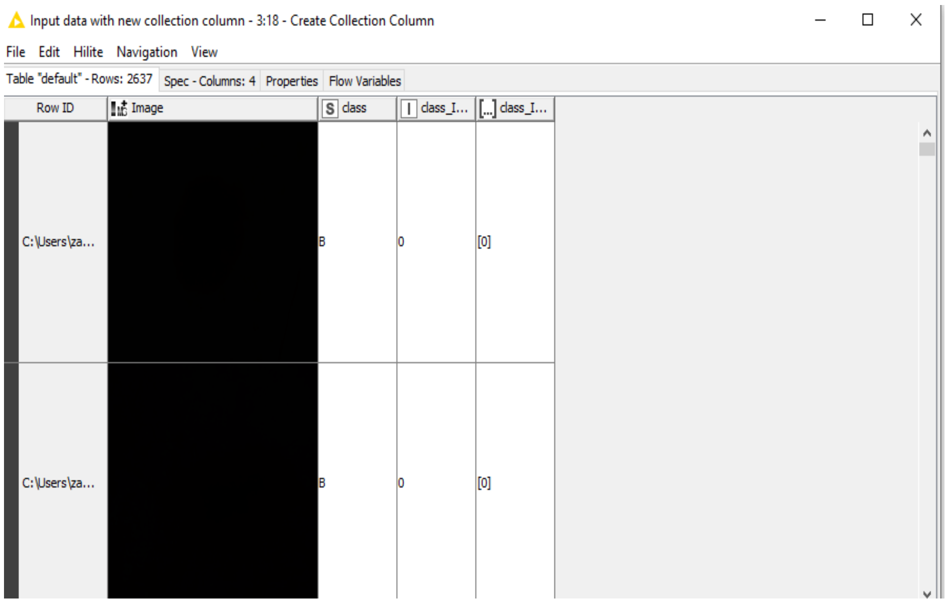

Dialog - 3:17 - Image Resizer
File
Options | Column Selection | Flow Variables | Job Manager Selection | Memory Policy
Resize Strategy: Linear Interpolation
Dimensions to Affect: All Dimensions Manual
New Dimension Sizes (will affect calibration)
X 224 Y 224 Z 1.0 Channel 3.0 Time 1.0
Scaling Factor For All Dimensions 1.0
Absolute Image Size

Figure 19. Image Resizer

2.3.3 Create Collection Column



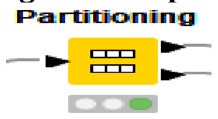
This node aggregates multiple columns in a separate new column, whose type is a "collection type". The cells in the new column are typed collections of cells; that is, the content can be safely be split into the original column content. The reverse operation is available in the "Split Collection Column" node. A group column is a type of database table column that stores multiple values in a single row. So the group (category index) is changed with numeric values (0 and 1) into two groups, and with that, a group cell is created with label index.


Input data with new collection column - 3:18 - Create Collection Column
File Edit Hilite Navigation View
Table "default" - Rows: 2637 | Spec - Columns: 4 | Properties | Flow Variables

Row ID	Image	class	class_I...	class_I...
C:\Users\za...		B	0	[0]
C:\Users\za...		B	0	[0]

Figure 20. Input data with new collection

2.3.4 Partitioning



The input table is split into two partitions (i.e., row-wise), e.g. train and test data. The two partitions are available at the two output ports. The input table is divided into two parts: training and test data. So that each of the two sections is dedicated to training, a collie on 80% training cells is created with a label indicator for 20% testing. So far, all of the preceding steps have been connected serially, and Partitioning to Keras Network Learner has been linked.

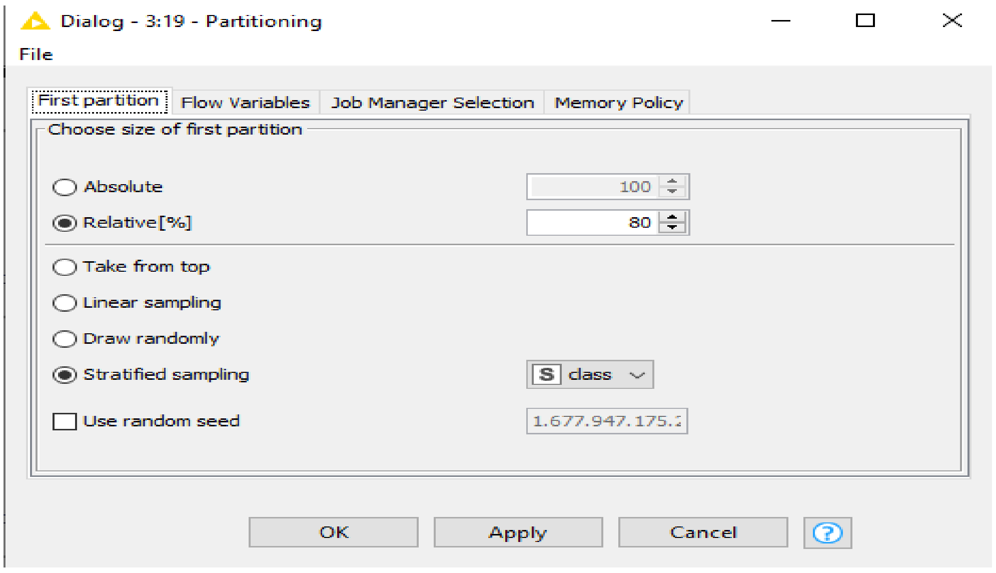
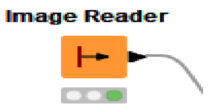

Dialog - 3:19 - Partitioning
File
First partition | Flow Variables | Job Manager Selection | Memory Policy
Choose size of first partition
( ) Absolute 100
(●) Relative[%] 80
( ) Take from top
( ) Linear sampling
( ) Draw randomly
(●) Stratified sampling S class
☐ Use random seed 1.677.947.175.2
OK Apply Cancel

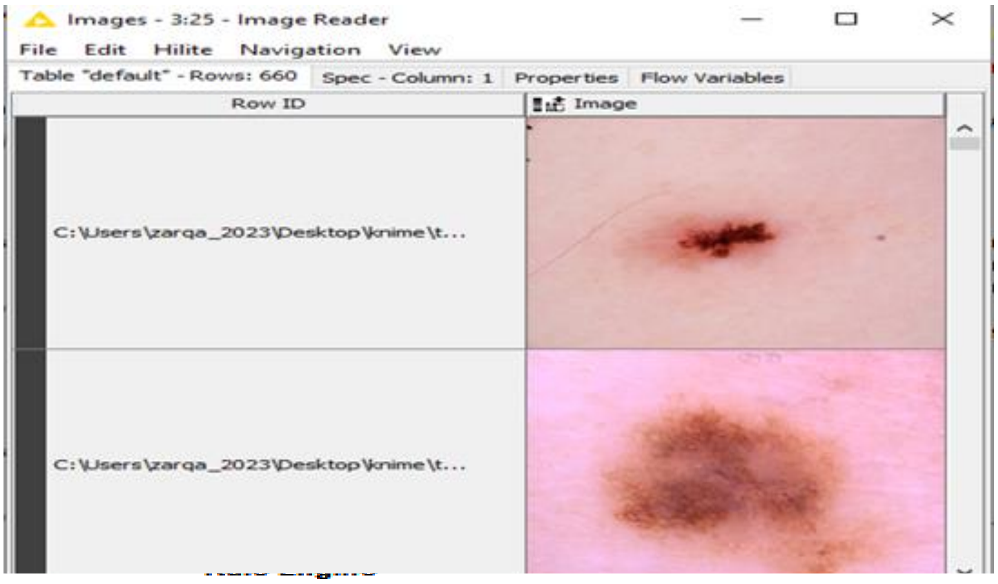
Figure 21. Partition Property

2.4 The Testing

2.4.1 Image Reader



It reads images from various file formats supported by the Bio-Formats library and imports them into the internal KNIME image format. Image readers are used to view and edit digital images in matrix form, so about 300 images of benign tumors and 300 images of malignant tumors are used for the purpose of network training.

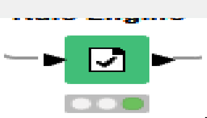

Images - 3:25 - Image Reader

File Edit Hilite Navigation View

Table "default" - Rows: 660 | Spec - Column: 1 | Properties | Flow Variables

Row ID	Image
C:\Users\zarqa_2023\Desktop\knime\t...	
C:\Users\zarqa_2023\Desktop\knime\t...	

2.4.2 Rule Engine



This node takes a list of user-defined rules and tries to match them to each row in the input table. If a rule matches, its outcome value is added into a new column. The first matching rule in order of definition determines the outcome. A rule engine is a software system that automates the process of making decisions based on a set of rules. where a condition for translating the images in the new matrix is specified so that for each satisfactory group, a specific value is encoded for denoting it (by appending class coins).

Procedures :

The condition:

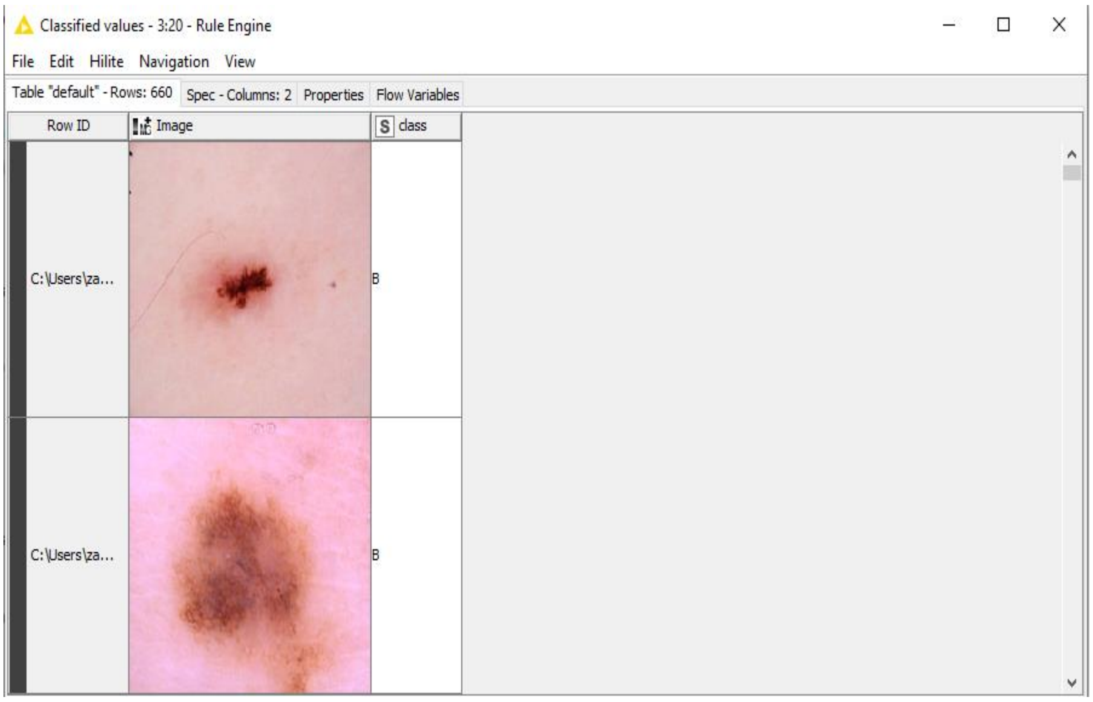
$$ROWID$$ LIKE "*benign*" =>"B"

TRUE => M

Where:

M: malicious.

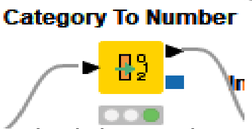
B: benign.


Classified values - 3:20 - Rule Engine
File Edit Hilite Navigation View
Table "default" - Rows: 660 | Spec - Columns: 2 | Properties | Flow Variables

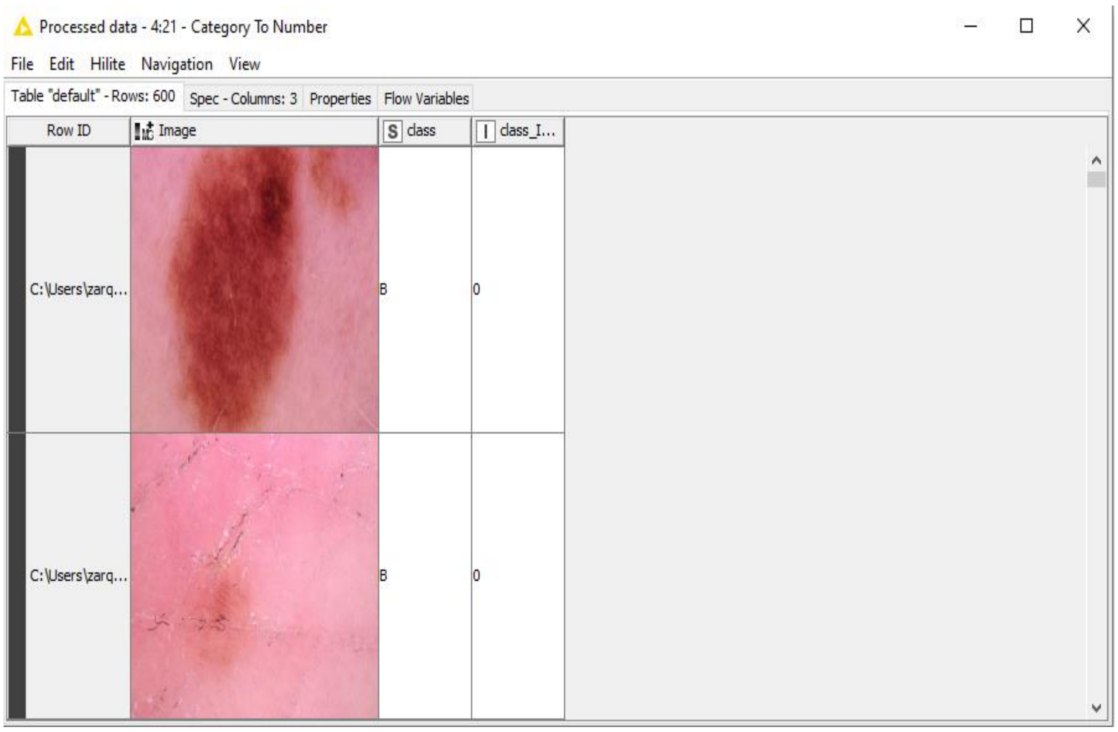
Row ID	Image	class
C:\Users\za...		B
C:\Users\za...		B

Figure 23. Explanation of conditions on pictures

2.4.3 Category To Number



This node takes columns with nominal data and maps every category to an integer. For convenience, multiple columns can process with this node. However, these columns are processed separately as if you would use a single Category To Number node for every column. The process of assigning integer values to categories or labels for indexing is known as Category to Number. Numeric values are easier to work with than categorical labels. For example, a data set might contain a column of sex (B, M) classifications that can be converted to numerical values (0, 1).



Processed data - 4:21 - Category To Number

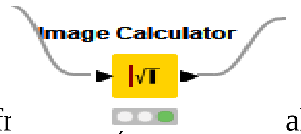
File Edit Hilite Navigation View

Table "default" - Rows: 600 | Spec - Columns: 3 | Properties | Flow Variables

Row ID	Image	class	class_I...
C:\Users\zarq...		B	0
C:\Users\zarq...		B	0

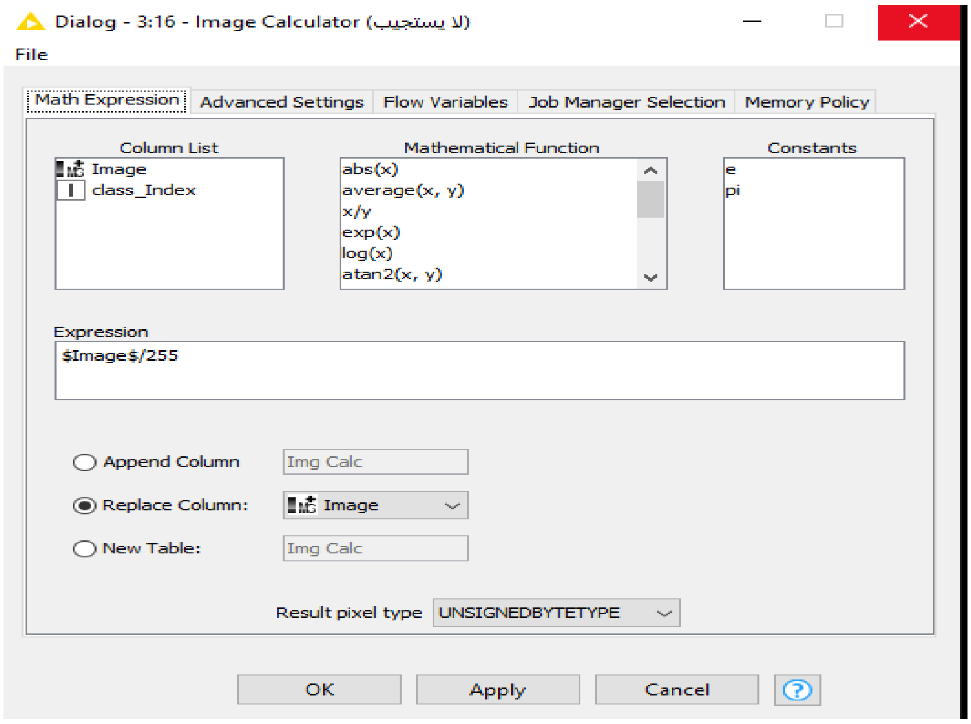
Figure 24. Converting the type of each disease into numerical values (0,1)

2.4.4 Image Calculator



This node evaluates a (fr al expression based on the images in a row. The computed results can be either appended as new column or be used to replace an input column. Available variables are the values in the corresponding row of the table (left list in the dialog). Commonly used functions are shown in the list "Mathematical Functions". Image Calculator is a software program that performs the normalization of images.

Procedures : $Image$/255.



Column List: Image, class_Index

Mathematical Function: abs(x), average(x, y), x/y, exp(x), log(x), atan2(x, y)

Constants: e, pi

Expression: $Image$/255

Append Column: Img Calc
[✓ Replace Column:] Image
New Table: Img Calc

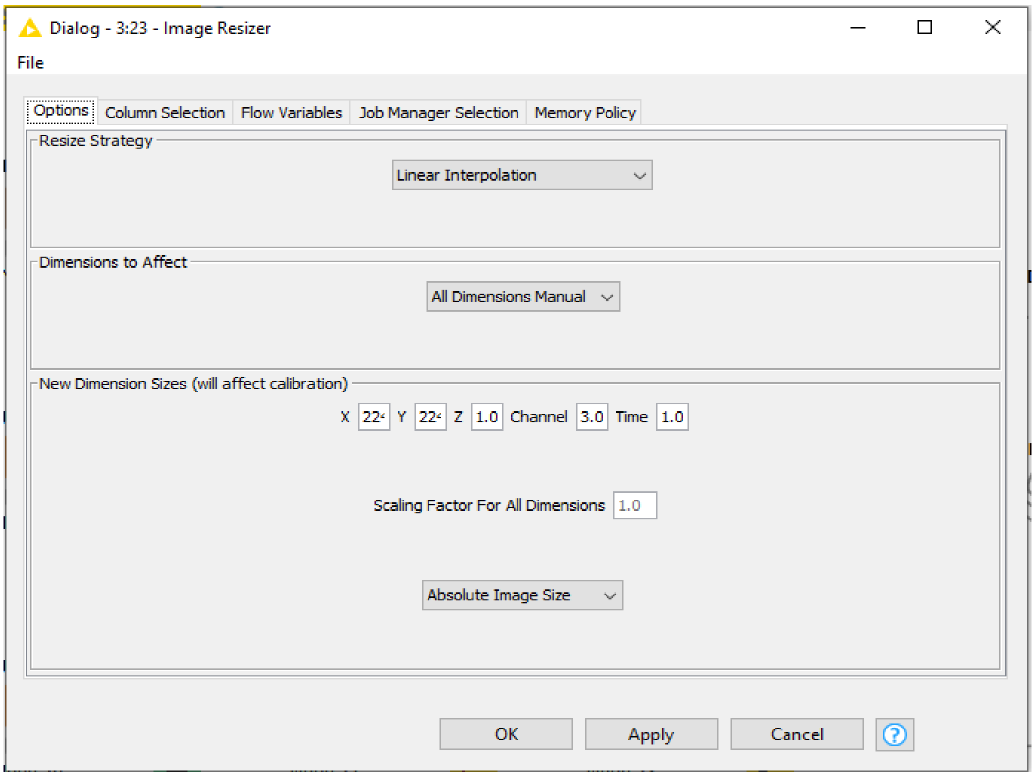
Result pixel type: UNSIGNEDBYTETYPE

Figure 25. Image calculator

2.4.5 Image Resizer

Resizes the image in each dimension. The resulting values at each position in the image are set according to the resizing strategy. Take in cinsidiration that, the input values for each dimension can be interpreted in several ways. Image Resizer is free software that allows users to quickly and easily resize digital images. So all the images are we made to be at the same size.

Procedures : X =224,Y=224,Z=1,Channel=3.


Dialog - 3:23 - Image Resizer
File
Options | Column Selection | Flow Variables | Job Manager Selection | Memory Policy

Resize Strategy: Linear Interpolation

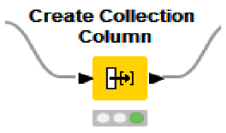
Dimensions to Affect: All Dimensions Manual

New Dimension Sizes (will affect calibration)
X 224 Y 224 Z 1.0 Channel 3.0 Time 1.0

Scaling Factor For All Dimensions 1.0

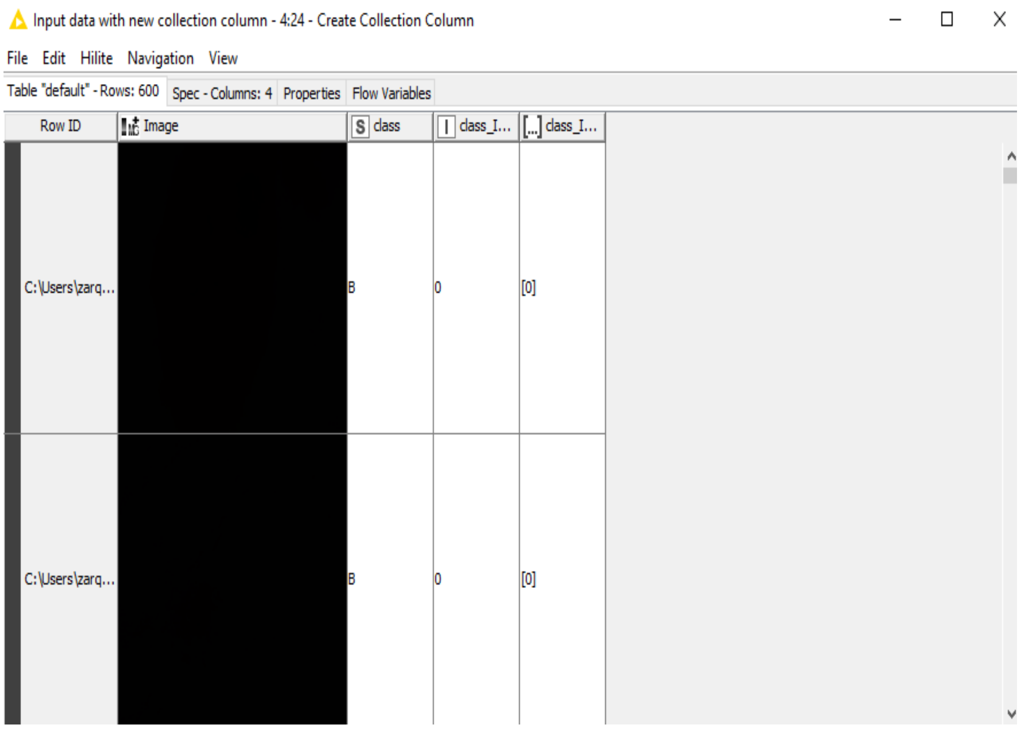
Absolute Image Size

Figure 26. Image Resizer



2.4.6 Create Collection Column

This node aggregates multiple columns in a separate new column, whose type is a "collection type". The cells in the new column are typed collections of cells; that is, the content can be safely be split into the original column content. The reverse operation is available in the "Split Collection Column" node. A group column is a type of database table column that stores multiple values in a single row. So the group (class index) is changed with numeric values (0 and 1) into two groups, and with this, a group cell is created with label index. So it will be connected it to the Keras Network Executor.


Input data with new collection column - 4:24 - Create Collection Column

File Edit Hilite Navigation View

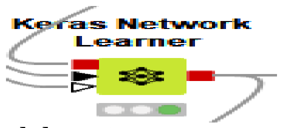
Table "default" - Rows: 600 | Spec - Columns: 4 | Properties | Flow Variables

Row ID	Image	class	class_I...	class_I...
C:\Users\zarq...		B	0	[0]
C:\Users\zarq...		B	0	[0]

Figure 27. Input data with new collection

2.5 The Training and Applying

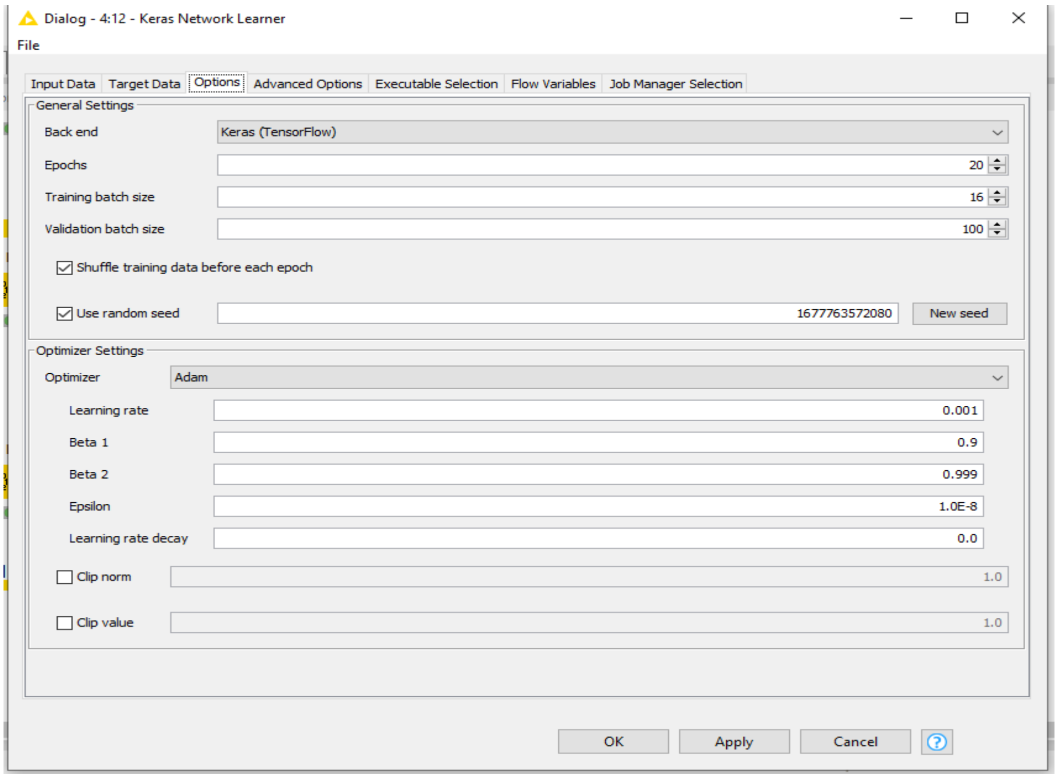
2.5.1 Keras Network Learner



This node performs supervised learning on a Keras deep learning network. Keras Network Learner works as a network learner; it allows to quickly and easily create deep learning models and also it allows to define own layers and models using simple APIs so that a powerful neural network can quickly build without having to write complex code to train the network.

Procedures :

- Epochs:20.
- Batch Size:16.



General Settings

Back end	Keras (TensorFlow)
Epochs	20
Training batch size	16
Validation batch size	100
☑ Shuffle training data before each epoch	
☑ Use random seed	1677763572080

Optimizer Settings

Optimizer	Adam
Learning rate	0.001
Beta 1	0.9
Beta 2	0.999
Epsilon	1.0E-8
Learning rate decay	0.0
☐ Clip norm	1.0
☐ Clip value	1.0

Figure 28. Keras Network Learner (options)

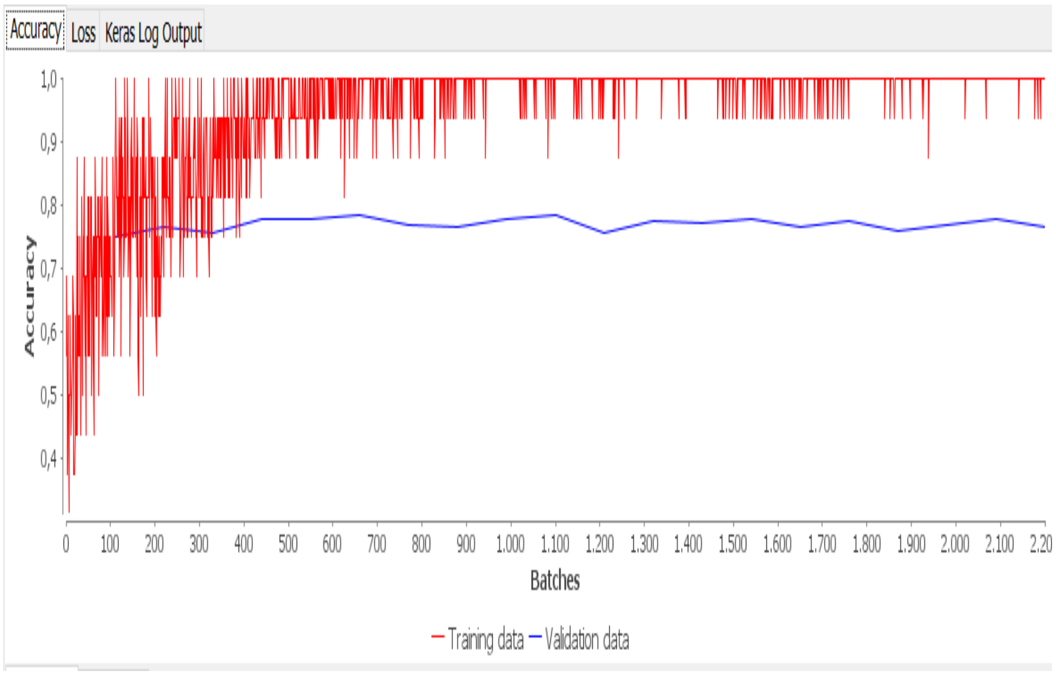
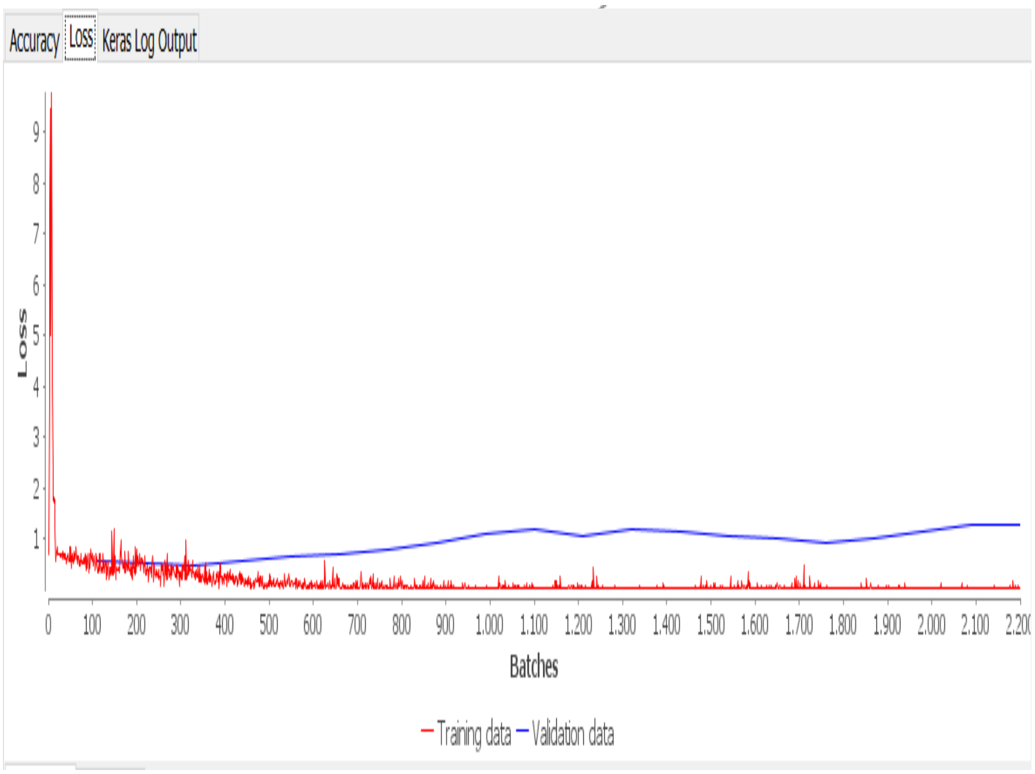


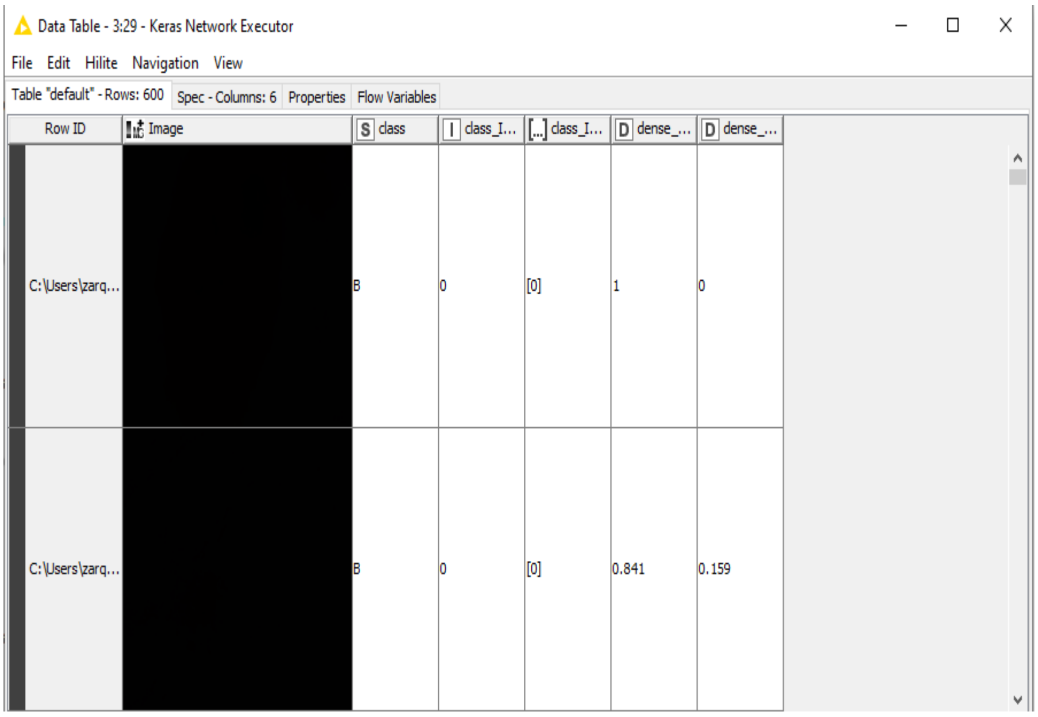
Figure 29. Learning Monitor (Accuracy)

Figure 30. Learning Monitor (loss)



2.5.2 Keras Network Executor

This node executes a Keras deep learning network on a compatible external back end that can be selected by the user. A configuration is made of the check for the test images in a specific way so that two outputs work so that it gives a value to both of them to know to which they belong.

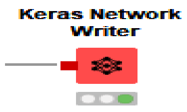

Data Table - 3:29 - Keras Network Executor
File Edit Hilite Navigation View
Table "default" - Rows: 600
Spec - Columns: 6
Properties
Flow Variables

Row ID	Image	class	class_I...	class_I...	dense_...	dense_...
C:\Users\zarq...		B	0	[0]	1	0
C:\Users\zarq...		B	0	[0]	0.841	0.159

Figure 31. Data Table

2.6 Evaluating and Save

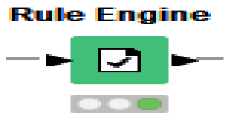
1.6.1 Keras Network Writer



Writes a Keras network to a file and We will save the entire model in a file.

For example: The trained model should be saved as a.h5 file.

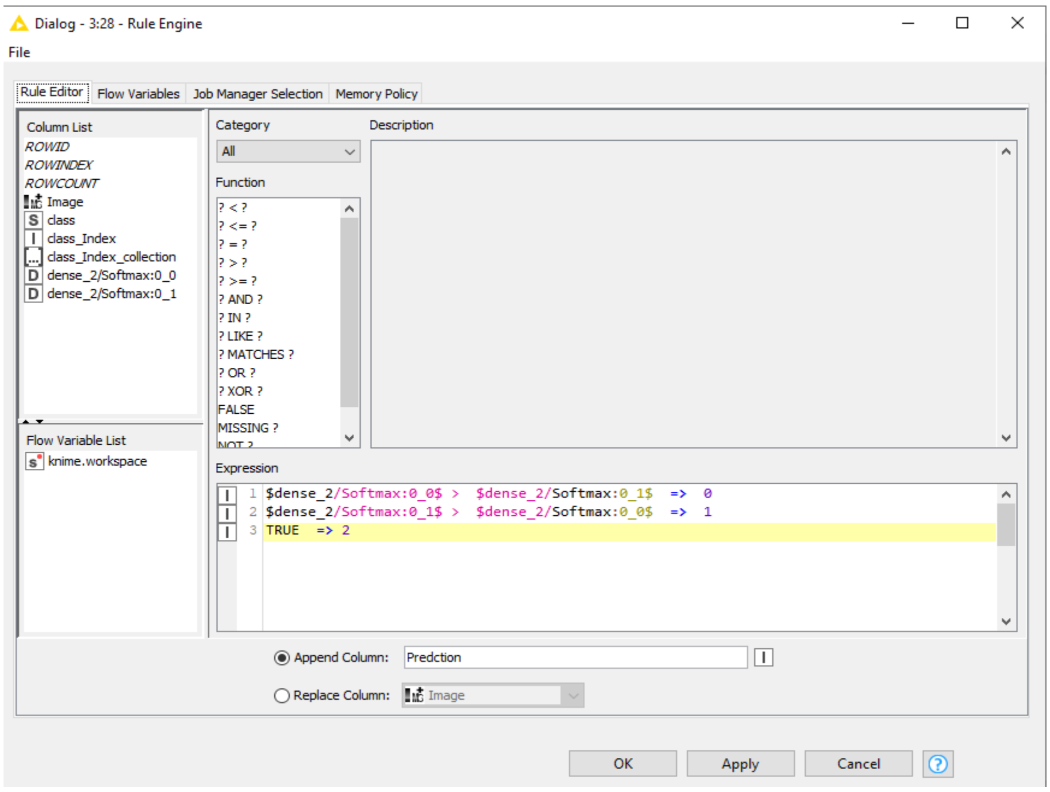
1.6.2 Rule Engine



This node takes a list of user-defined rules and tries to match them to each row in the input table. If a rule matches, its outcome value is added into a new column. The first matching rule in order of definition determines the outcome. It uses rules to evaluate the data and determine the appropriate action to take (add class columns).

The condition:

$dense_2/Softmax:0_0$ > $dense_2/Softmax:0_1$ => 0
$dense_2/Softmax:0_1$ > $dense_2/Softmax:0_0$ => 1
TRUE => 2

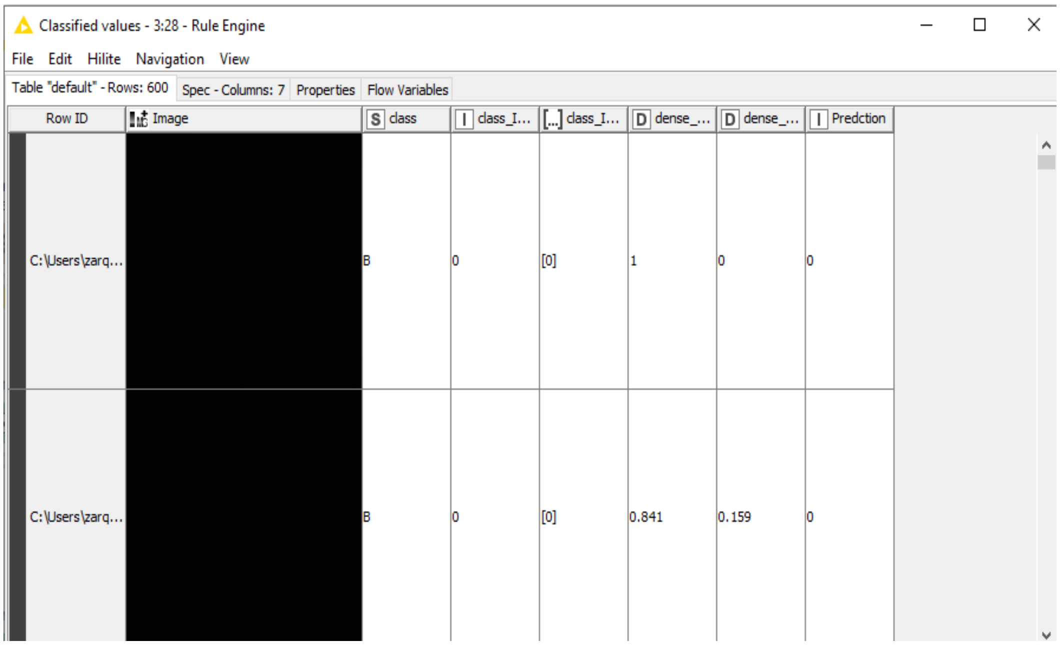


```
$dense_2/Softmax:0_0$ > $dense_2/Softmax:0_1$ => 0
$dense_2/Softmax:0_1$ > $dense_2/Softmax:0_0$ => 1
TRUE => 2
```

(◉) Append Column: Predction

( ) Replace Column: Image

Figure 32. uses rules to evaluate the data and determine the appropriate action

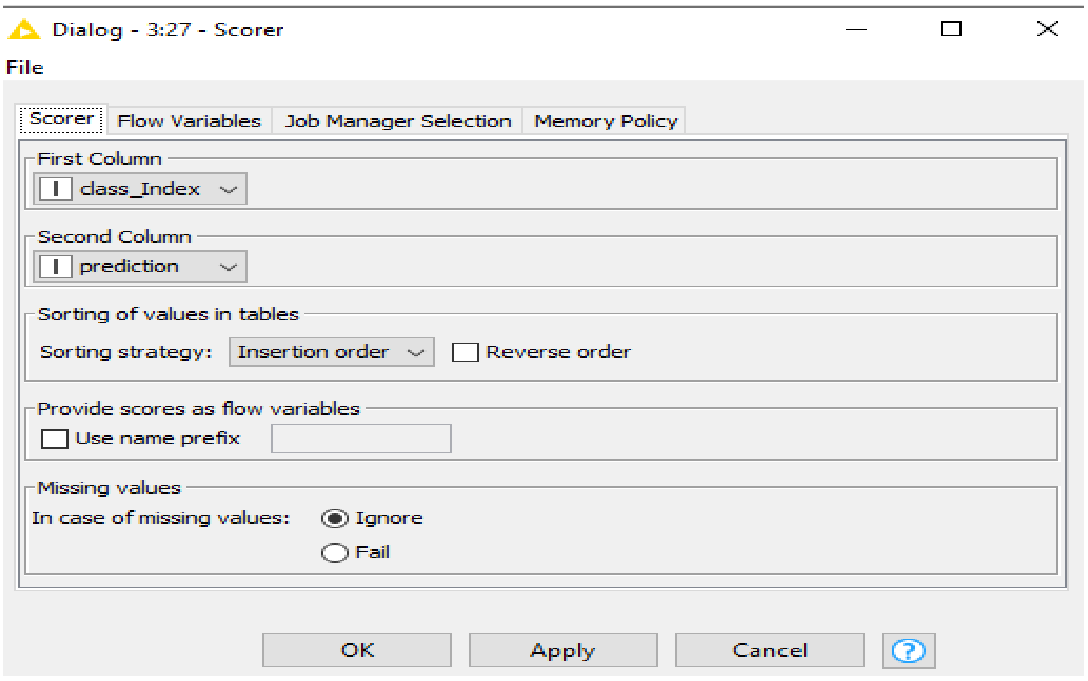


Row ID	Image	class	class_I...	class_I...	dense_...	dense_...	Predction
C:\Users\zarq...		B	0	[0]	1	0	0
C:\Users\zarq...		B	0	[0]	0.841	0.159	0

Figure 33. Explanation of conditions on pictures

1.6.3 Scorer

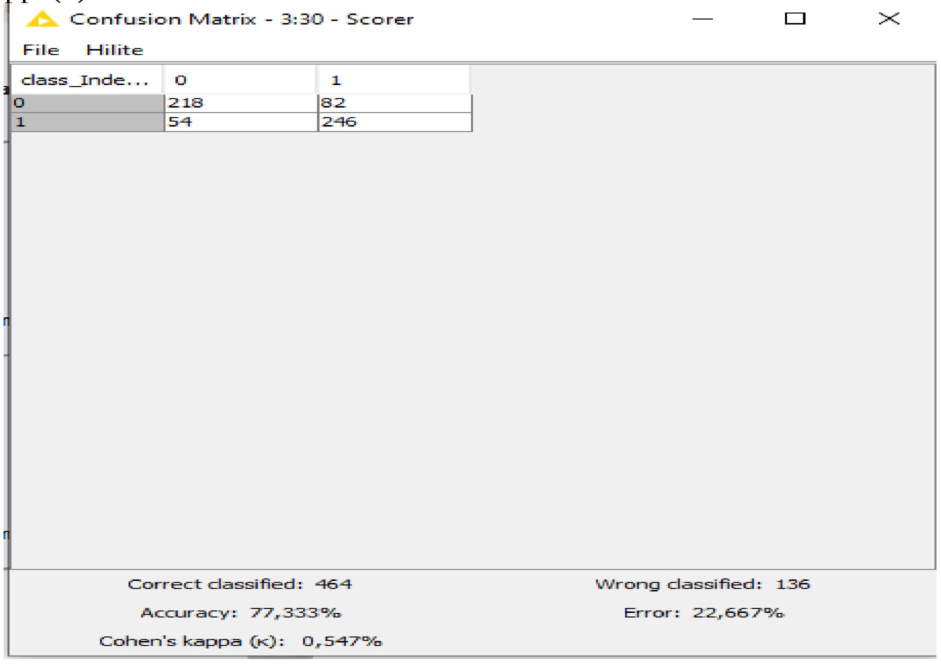
Two columns are compared by their attribute-value pairs and the display of the confusion matrix, i.e. the number of rows in which an attribute matches its taxonomy. In addition, it is possible to highlight the cells of this matrix to identify the initial rows. The dialog allows to select two columns for comparison; The values from the first specified column are represented by the rows of the confusion matrix and the values from the second column by the columns of the confusion matrix. The node output is the confusion matrix with the number of matches in each cell. Additionally, the second output reports a number of accuracy stats. score (sentence).


Dialog - 3:27 - Scorer

First Column: class_Index
Second Column: prediction
Sorting of values in tables
Sorting strategy: Insertion order ☐ Reverse order
Provide scores as flow variables
☐ Use name prefix
Missing values
In case of missing values: [✓ Ignore] Fail

Figure 34. Compare Data Result

With this training, we obtained an accuracy of 77.333%, and an error rate of 22.667, Cohen's kappa(x) 0.547%.


Confusion Matrix - 3:30 - Scorer

class_Inde...	0	1
0	218	82
1	54	246

```
Correct classified: 464          Wrong classified: 136
Accuracy: 77,333%                Error: 22,667%
Cohen's kappa (κ): 0,547%
```

Figure 35. The end result that was done

V. Conclusion

In this paper, it is noted that Artificial Intelligence (AI) using a deep learning model is significantly more effective at predicting the melanoma disease, which suggests that deep learning characteristics are more discriminating and may be better at predicting disease to aid radiologists in accurately and quickly diagnosing patients with skin cancer. In this paper, a classification model is developed for melanoma image recognition utilizing the KNIME Analytics Platform Software based on the integration of CNN and ML characteristics. This model enhances the network architecture and enhances performance results for the binary classification job of differentiating between unhealthy and infected photos. Malignant melanoma must be identified early on in order to be properly diagnosed and treated; however, this can be difficult to do with just the naked eye. As a result, it usually only gets detected after it has advanced a certain amount. This work demonstrated that the extraction and classification of lesion regions using a training strategy based on two deep learning models can aid in the early identification of malignant melanoma. The results of the training process for the deep learning models showed that, with the exception of a few photos where the lesion areas were unlikely to be identified using dermoscopy, sufficient segmentation performance was attained in most cases. The comparison of tests and evaluation metrics reveals that the model that is usedIn this project , significantly increases recognition accuracy when compared to both deep learning algorithm models and traditional machine learning algorithms, and prediction accuracy can reach 77.333%. The proposed CNN model may successfully identify and categorize photos of skin cancer, according to experiments.

REFERENCES

[1] Ferlay, J., et al. "Estimating the global cancer incidence and mortality in 2018: GLOBOCAN sources and methods." International journal of cancer 144.8 (2019): 1941-1953.
[2] https://forum.knime.com/t/online-course-l4-dl-introduction-to-deep-learning-online/32536.
[3] https://www.cdc.gov/cancer/skin/basic_info/index.htm
[4] Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B., 2007. KNIME: the Konstanz information miner. Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007). Springer.
[5]. Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol. Cybern. 1980, 36, 193–202, doi:10.1007/BF00344251.
[6] LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. Proc. IEEE 1998, 86, 2278–2324, doi:10.1109/5.726791.
[7] ImageNet Large Scale Visual Recognition Competition (ILSVRC). Available online: http://www.image-net.org/challenges/LSVRC/ (accessed on 10 October 2019).
[8] Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; Li, F.F. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09), Miami Beach, FL, USA, 20–25 June 2009.
[9] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the Neural Information Processing Systems Conference, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.

[10] Hertel, L.; Barth, E.; Käster, T.; Martinetz, T. Deep convolutional neural networks as generic feature extractors, In Proceeding of the 2015 International Joint Conference on Neural Networks, Killarney, Ireland, 12–16 July 2015.
[11] Phung, V.H.; Rhee, E.J. A Deep Learning Approach for Classification of Cloud Image Patches on Small Datasets. J. Inf. Commun. Converg. Eng. 2018, 16, 173–178, doi:10.6109/jicce.2018.16.3.173.
[12] National Cancer Institute, "Melanoma skin cancer ", U.S. Department of Health and Human Services, National Institutes of Health, 2010.
[13] Aswin.R.B, J. Abdul Jaleel, Sibi Salim, "Implementation of ANN Classifier using MATLAB for Skin Cancer Detection", College of Engineering and Technology during December, 2013 at Trivandrum, Kerala, India.
[14] Bareqa Salah, Mohammad Alshraideh, Rasha Beidas and Ferial Hayajneh, "Skin Cancer Recognition by Using a Neuro-Fuzzy System'', The University of Jordan, Amman 11942, Jordan, 2011.
[15] Kalouche, S. Vision-Based Classification of Skin Cancer Using Deep Learning. 2016. Available online: https://www.semanticscholar.org/paper/Vision-Based-Classification-of-Skin-Cancer-using-Kalouche/b57ba909756462d812dc20fca157b3972bc1f533 (accessed on 10 January 2021).

AUTHOR BIOGRAPHY

Ahmad M. EL-Fallah Ismail, born on June 5th 1982 at Gharian (Libya), Received the Bachelor of Engineering (BEng) degree in Electrical and Electronic Enginering (Control Systems) from Al-Jabal Al-Gharbi University, Gharian (libya) in 2006, the Diploma degree in Electrical and Electronic Engineering (Control Systems) from Faculty of engineering, Alfatah University, Tripoli (libya) in 2008, the M.Tech. degree in Electrical and Electronic Enginering. (Control Systems) from Sam Higginbottom University of Agriculture, Technology and Sciences (SHUATS), Allahabad (India) in 2011, and the Ph.D. degree in Electrical and Electronic Enginering (Control Systems) from Sam Higginbottom University of Agriculture, Technology and Sciences (SHUATS), Allahabad (India) in 2015, the field of interest is Artificial Intelligence and Evolutionary Algorithms. E-mail: dr.ahmad_ismail_alrabty@yahoo.com or Ahmad.Ismail@gu.edu.ly.