DEEP LEARNING PROJECT REPORT

* 1)I had taken Malaria Cell Images Dataset. I had used three deep learning approaches

which I learnt in class.

* Multi-Layer Perceptron
* Convolutional Neural Networks
* Data Augmentation

In my dataset I had performed classification because we need to detect the malaria cells is infected or not in humans by using images. Cell images are classified into two types.

* Parasitized
* Uninfected
* 2)Reference 1: <https://www.kaggle.com/abhijit96/malaria-cell-image-usingkeras>

In my reference1 he had loaded the data and shuffle the data. With that data he assigned x\_train, x\_test, y\_test, y\_train. He had defined a for loop to process the images. From that samples he passed those samples to CNN layers. He had taken only 3 filters and 1 dense layer from that data he computed accuracy and plotted the data.

Reference 2 : The Effectiveness of Data Augmentation in Image Classification using Deep Learning by [Luis Perez](https://arxiv.org/search/cs?searchtype=author&query=Perez%2C+L), [Jason Wang](https://arxiv.org/search/cs?searchtype=author&query=Wang%2C+J)

In my reference 2 I had gone through What is Data Augmentation? How to use Data Augmentation? What are the attributes required to compute Data Augmentation?

* 3)I want to know how the accuracies are obtained by using same data process for Multi-Layer perceptron, and convolution neural network. I got low accuracies when I process data to multi-layer perceptron. My accuracies are improved by using the same data process for convolution neural networks. I used different approach to solve my dataset that is data augmentation.
* 4)I had implemented some approaches to solve my dataset which are applicable to my dataset. I had learned what is convolution, kernels, maxpooling, types of optimizers, Activation Functions.

DEEP LEARNING PROJECT

Malaria Cell Images Dataset(In Depth)

DIFFERENT DEEP LEARNING TECHNIQUES TO SOLVE MY DATASET:

* Multi-Layer Perceptron
* Convolution Neural Network
* Data Augmentation

Multi-layer perceptron:

Import Libraries:

I began by importing NumPy, glob, os and cv2. I decided to use keras with TensorFlow as backend to implement Multilayer Perceptron. So, I imported layers, models, optimizers.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Import Dataset:

In Kaggle, all data files are located inside the input folder which is one level up from where the notebook is located. The images are inside the cell images folder. Thus, I set up the data directory as “path” to point to that location. To store the features, I used the “parasitized” and “uninfected” . For this project, I set each image size to be 60x60.

A screenshot of a cell phone

Description automatically generated

Next step was to import the data. The parasitized (infected) cell images are inside the “parasitized folder” and uninfected files images are inside the “uninfected folder”.

A screenshot of a social media post

Description automatically generated

Split Data:

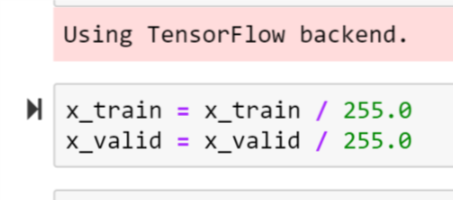
Data is classified into x\_train, x\_valid, y\_train, y\_valid with the help of x and y.

A screenshot of a cell phone

Description automatically generated

Data Normalization :

To normalize x\_train, x\_test we use Data Normalization for getting better accuracy.



To categorical Function :

Converts a class vector (y\_train, y\_valid) to binary class matrix and stored the data in two new variables (y1\_train, y1\_valid)

A screenshot of a cell phone

Description automatically generated

Multi-Layer Perceptron Model:

A screenshot of a social media post

Description automatically generated

Accuracy:

A screenshot of a cell phone

Description automatically generated

I am getting low accuracy when executing multi-layer perceptron . To get the higher accuracy we will use convolution neural networks.

We will use the same preprocessing data to execute convolution neural network

A screenshot of a social media post

Description automatically generated

Convolution neural network model

A screenshot of a social media post

Description automatically generated

**Convolution2D**

**filter:**The first parameter defines the output shape of the layer. In this case, we use 32,64,128,128.

**kernel size:**It defines the size of the window we want to use that will traverse along the image. I set it as 7x7,5x5,3x3,3x3.

* **input\_shape:**It is used to define the input size of each image. In this project, I am using images of size 60,60 and the images are colored i.e. they are composed of RED, BLUE and GREEN. The channels are thus 3. So, the parameter input\_shape will be (64x64x3). We need to define input\_shape only for the first layer.
* **activation:** The activation function is defined in this parameter. I used relu which is Rectified Linear Unit as the activation function.

#### MaxPool2D

* **pool\_size:**It defines the matrix size which defines the number of pixel values that will be converted to 1 value. I used the value as 2x2 so an image of size 60x60 will be converted to 30x30.

#### Batch Normalization

It normalizes the output from the previous activation function.

#### Dropout

It selects some of the values at random to be set as 0.25 so as to prevent overfitting in the model and I used only the rate parameter.

Flatten

It flattens the complete n-dimensional matrix to a single array. So, if its size was 60x60x3, it will be converted to an array and it acts as the input for the dense layer ahead.

#### Dense

It defines a densely connected neural network layer and I defined the following parameters:

* **activation:**It defines the activation function which I set as relu except for the last layer. For the last dense layer, I set the activation as sigmoid.
* **units:** It defines the number of neurons in the given layer. I created three layers with neuron count as 512 and 2 respectively.

**Compile :**

last step is to compile the model. The optimizer is Adam and this being a categorical problem, I used the loss as categorial\_crossentropy and evaluation metric as accuracy.

Train and Valid Accuracy:

A screenshot of a cell phone

Description automatically generated

Plotting values of Train and Valid Accuracy:

A screenshot of a cell phone

Description automatically generated

Plotting values of Train and valid Loss:

A screenshot of a map

Description automatically generated