Malaria Detection Using Convolutional Neural Network

**Project Goals and Objectives:**

This system aims to reduce the malaria-infected cases through:

* Produce faster diagnostic method.
* Produce system that can be used by everyone.
* A reliable malaria results. Our project will be useful especially in regions were its hard to find the expert/system that provide fast diagnostic result.

**Preparing data :**

I’m going to use kaggal dataset and work with approximately 27k image of malaria cell

(total of infected and uninfected) reference to who works on

“Improving Malaria Parasite Detection from Red Blood Cell using Deep Convolutional Neural Networks”

i will divide the images into 3 parts (70%train, 15%test and 15%validate).

**System Analysis:**

CNN has two stages which are:

* Feature extraction

Can automatically learn hierarchical representation from data by extracting features from the input data.

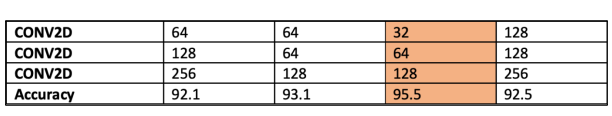
* Classification

The Backpropagation (BP) is one of the most widely used algorithms for supervised training networks.

**Software Requirements:**

* Python (For the coding).
* Anaconda Server (For python coding).
* Kaggal (For malaria dataset).

**Modeling:**

extracted features using Conv2D with kernel sized (3,3) and choosing the most affected filter numbers, here in the table as you see there are all our filter numbers used to achieve the higher accuracy. Starting with (64,128,256) which give us 92.1%, (64,64,128) with 93,1% accuracy, (32,64,128) with 95.5% and (128,128,256) which gave us 92.5%, so the most affected filter number is (32,64,128).

**System architecture :**

is composed of three Convolutional layers where a Max-pooling layer follows each layer.

For performing the convolutional operation in each layer, the kernel size is defined as (3 \* 3) with the same padding and 1-pixel stride, kernel is placed in the top-left corner of the image. The pixel values covered by the kernel are multiplied with the corresponding kernel values and the products are summated.

The result is placed in the new image at the point corresponding to the center of the kernel, and it has two fully connected layers. The conv layer and max pooling will repeat three times.

When the user inserts the chip which contain the blood, the input image is moved to the CNN.

Firstly, the input layer will expect an image of size (250\*250) pixels and send it to convolution layer, it is the core building block of a CNN where the feature extraction is done and filters are applied to the original image then it will move to the max pooling which will reduces the number of parameters and down sampling the spatial dimensions of the input image, we will use the most common setting used with max-pooling which is (2\*2) ,notice that any larger than (2\*2) pooling is then too lossy and aggressive. This usually leads to worse performance.

Max-pooling layer follows each layer using the equation S=max(0,M) for performing the convolutional operation in each layer. Here, S is the output after applying non-linearity on matrix M.

To sample down the features map, max-pooling layer shown in Z=maxh,wi,j=1(Mi,j). Z is the output matrix containing maximum value of each patch from input matrix M.

The result of this process feeds into a fully connected layer that drives the final classification decision.

In this case have first fully connected layer, which takes the inputs from the feature analysis and applies weights to predict the correct label, proper weight initialization enhances the learning process. All inputs units have a separable weight to each output unit. For “n” inputs and “m” outputs, the number of weights is “n\*m”. Additionally, this layer has the bias for each output node, so “(n+1)\*m” parameters, and the second FC is fully connected output layer which gives the final probabilities for each label, and it will feed the result into the SoftMax classifier.

Finally, SoftMax turn logits (numeric output of the last linear layer of a multi-class classification neural network) into probabilities by taking the exponents of each output and then normalize each number by the sum of those exponents so the entire output vector adds up to one all probabilities should add up to one, and gives the result which is either (+ or - ).