**Data Preprocessing**

The data was checked initially to ensure that all the images are in a proper extension.

Then before going to the model, they were resized, normalized, and augmented.

**Reflection**

*This project is on Kaggle, the project is designed to classify the cells that are parasites and the uninfected ones, the project started by cleaning the dataset from any corrupted images and rescaling them. By building a CNN model from scratch and trial of many architectures till we reach the one that did not give us just a random classification error.*

*The most difficult aspect of the project is the runtime of the project according to the availability of GPU in the development environment or not.*

*After applying many architectures, I chose the one that gives me the most satisfactory result for my case.*

*In any image classification project, there is no single architecture that could be used to any dataset with a guaranteed result, one should iterate till he reach to the most satisfactory model for his use-case*

**Improvement**

* The result in this model could be definitely improved by applying all the datasets available, also we could try different model architectures to get the best result in not such a big time.
* The usage of GPU in the computation could reduce the runtime dramatically.
* Using different pre-trained models could give a better result
* Try the usage of transfer learning method

The final classification layer was replaced by another dense layer with 1 neuron instead of 1000.

The model was trained for 30 epochs, optimized by Stochastic Gradient Descent with the learning rate = 0.04 to reduce the Cross-entropy loss function. The performance during training was validated based on the loss value on the validation set. On the test set, the model achieved 79% of accuracy and a loss of 0.647.

1. The solution model was compared to a benchmark CNN with:
2. - 4 convolutional layers, each of them followed by ReLU and 2D Max Pooling,
3. - a flattening layer,
4. - a drop-out layer,
5. - a fully-connected layer,
6. - ReLU,
7. - drop-out
8. - and another fully-connected layer.
9. The architecture was designed to follow the common structure of CNN classifiers and VGG in particular: the first layers are convolutional, the number of features increases in higher layers. The feature extractor is followed by flattening of the feature tensor and the classifier: 2 dense layers with activation functions. The kernel size of (3, 3) is the most popular, having several features like the power of 2 is also a standard.
10. The benchmark model was trained for 20 epochs, optimized by Stochastic Gradient Descent with the learning rate = 0.05 to reduce the Cross-entropy loss function. The performance during training was validated based on the loss value on the validation set. On the test set, the model achieved 18% of accuracy and a loss of 3.565.