Jad Hamdan

MAIS 202 Project Deliverable 2

Disclaimer:
The dataset I had chosen initially to build a handwritten to LaTeX mathematical expression converter was in inkML, a complicated format comprised of the strokes done to draw each individual symbol. As days went by, I had trouble preprocessing the data and the deliverable's deadline was rapidly approaching. I then found and rushed to a version of the dataset that was already converted to PNG images, only to find out that it contained repeated and corrupted photos. Because of that, I decided to switch datasets to one that was simpler on the pre-processing side, due to timing constraints.

1. Problem Statement:

My new project is a model that, upon receiving an image of a cell as input, predicts whether or not said cell is parasitized with Malaria.

2. Data Preprocessing:

The dataset I switched to [1] proved to be very easy to work with. Simple in terms of structure, the dataset contained two folders: one with images of infected cells and another with pictures of their healthy counterparts.

I then proceeded to manually divide those images into three subcategories (training, validation and testing data) with 12267, 1000 and 512 samples respectively. I then divided each of these further, with a folder containing infected cells and another containing the uninfected ones. The data is not labelled, but this way of dividing it into folders allows me to use Keras' flow_from_directory function to automatically label by folder.

As for the images themselves, I proceeded to normalize them and resize them all to size 64x64 to avoid size mismatch and dimension errors while training. The aspect ratio is what mattered the most to me, with the number 64 being randomly chosen.

3. Machine Learning model

I had previously mentioned that I was going to use a Convolutional Neural Network for my classification problem, as this type of network has an aptitude for feature extraction. Seeing as my new project is also an image classification project, I decided to stick to that choice.

- I used Keras (with TensorFlow for backend) to build and train my network. I chose this high level API to be able to experiment without worrying about details that could end up costing me hours of debugging.
- I ended up using a very simple 2D CNN. As indicated by its name, the net is made of two Convolution layers with a kernel of size 3 and two Max Pooling layers of size 2.

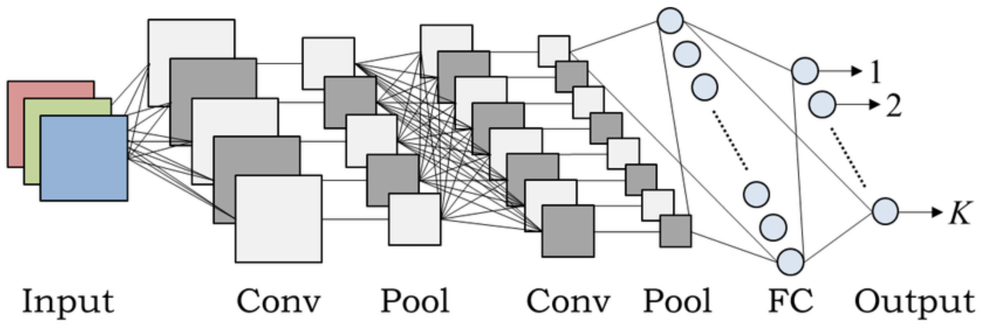


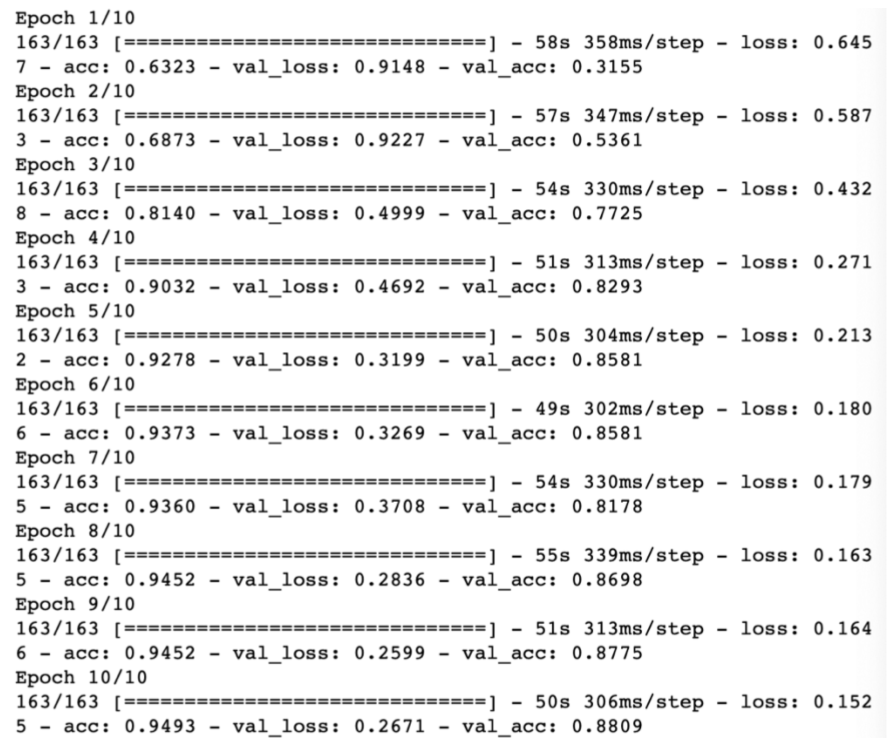
Fig 1: 2D CNN Architecture

[1] https://ceb.nlm.nih.gov/repositories/malaria-datasets/

- I didn't use any regularization techniques not optimization tricks yet. I initially set my hyperparameters randomly and experimented with different combinations until reaching the results I have now.
- My model seems to be slightly overfitting, seeing as my training accuracy is extremely high (close to 95%). To fix this, I plan on trying some regularization techniques and perhaps changing the size of my training set.

4. Preliminary Results:

The model produced some pretty impressive results after 10 Epochs:



```
Epoch 1/10
163/163 [==============================] - 58s 358ms/step - loss: 0.645
7 - acc: 0.6323 - val_loss: 0.9148 - val_acc: 0.3155
Epoch 2/10
163/163 [==============================] - 57s 347ms/step - loss: 0.587
3 - acc: 0.6873 - val_loss: 0.9227 - val_acc: 0.5361
Epoch 3/10
163/163 [==============================] - 54s 330ms/step - loss: 0.432
8 - acc: 0.8140 - val_loss: 0.4999 - val_acc: 0.7725
Epoch 4/10
163/163 [==============================] - 51s 313ms/step - loss: 0.271
3 - acc: 0.9032 - val_loss: 0.4692 - val_acc: 0.8293
Epoch 5/10
163/163 [==============================] - 50s 304ms/step - loss: 0.213
2 - acc: 0.9278 - val_loss: 0.3199 - val_acc: 0.8581
Epoch 6/10
163/163 [==============================] - 49s 302ms/step - loss: 0.180
6 - acc: 0.9373 - val_loss: 0.3269 - val_acc: 0.8581
Epoch 7/10
163/163 [==============================] - 54s 330ms/step - loss: 0.179
5 - acc: 0.9360 - val_loss: 0.3708 - val_acc: 0.8178
Epoch 8/10
163/163 [==============================] - 55s 339ms/step - loss: 0.163
5 - acc: 0.9452 - val_loss: 0.2836 - val_acc: 0.8698
Epoch 9/10
163/163 [==============================] - 51s 313ms/step - loss: 0.164
6 - acc: 0.9452 - val_loss: 0.2599 - val_acc: 0.8775
Epoch 10/10
163/163 [==============================] - 50s 306ms/step - loss: 0.152
5 - acc: 0.9493 - val_loss: 0.2671 - val_acc: 0.8809
```

Fig 2: Testing and validation accuracies after 10 epochs

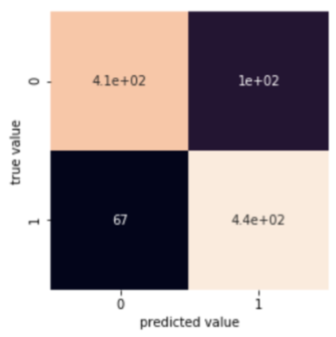


Fig 3: Binary Classifier's confusion matrix

With a testing accuracy of 93%, the net's simple architecture seems to be quite effective. A confusion matrix (fig 3) shows us that amongst the incorrect predictions, false positives (indicating that at uninfected cell is in fact parasitized) are most common. This will be taken into consideration when trying to improve the model and its performance.

5. Next Steps:

This Keras model manages to achieve a testing accuracy above 90%. However, its implementation was very simple seeing as both the problem and model were simple. I'd like to tackle a bigger challenge, now that reading week is approaching and I ti
With reading week approaching, I'll have more time to work on the project and decide what to next with it. This could mean giving my initial dataset a second chance, altering my CNN's architecture or attempting to solve this very same problem from scratch using only numpy.

[1] https://ceb.nlm.nih.gov/repositories/malaria-datasets/