Mid Project Report (Results)

15/03/2021

Introduction:

Our goal for this project is to be able to detect infected cells with Malaria from a series of blood cells, but more precisely to determine the exact placement of the parasite.

- Data segmentation:

The data is issued from segmenting large images of dimension 2456x2054 pixels in 16-bit grayscale captured from axial LED illumination.

These large images are then segmented in order to get a cell in each image, these will be considered our data that we will later on work with. The segmented images are of dimension 84x84 pixels.

The segmentation however is far from ideal, because some of the cells aren't well captured and centered in the image block, but rather cropped, as we can see from some examples below:

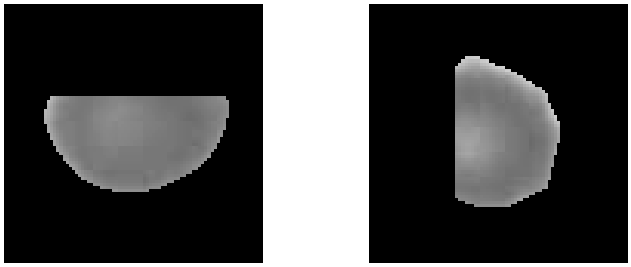


Figure 1 : example of not very well segmented cells

After obtaining the segmented images, these are then labelled by hand as healthy or Unhealth (infected) cells.

In this Project we will be focusing only on the patient CAT01 that has very visibles and large parasites in the infected cells.

- Data Augmentation:

We also need to point out that the initiale dataset is not balanced, meaning the number of healthy cells is far more bigger than those of infected cells, that's where the technique of data augmentation takes place to solve this problem.

What is data augmentation?

CNN are not scale or rotation variant, meaning for instance if we trained our model on a set of images and tried to test and classify the same images but that had been rotated, in this case CNN might not

perform well. That’s why we try to use this technique to generate new samples from already existing training data, this way we can see how the model perform in terms of accuracy and how it overfits the data.

This technique is really interesting because it enables the models to train even on small data sets, because it creates new samples from tweaking the image parameters from rotation, contrast, applying filters and so on.

- Classification:

We will be using the model ResNet for the classification. Using pretrained this model has a goal to extract the features from the parasitized and uninfected cells to aid in improved malaria disease screening. Also to compare how well this model does in terms of accuracy to the two other models that were used handling the same data in previous works (VGGNet and AlexNet).

Characteristics of our model:

- Total params: 23,587,712
- Trainable params: 23,534,592
- Non-trainable params: 53,120

The input images are of size 84x84 pixels, so before feeding the images to the classifier we need to resize them so they’ll be compatible with the input of the classifier. In this case we need to resize the images to be (224,224).

We performed 5-fold cross validation. we get train, validation and test splits for each fold to ensure that none of the input information in the training data leaks into the test data.

- Training data = 4443
- Validation data = 1779
- Testing data = 3244

We procceed by modifying the architecture by adding a global spatial average pooling layer and a fully-connected layer with a dropout ratio of 0.5 to prevent overfitting and help model generalization. We will train only the top layers which are randomly initialized, freeze all the convolutional layers to prevent large gradient updates wrecking the learned weights.

We store the best weights of the model after training, so we can use the best ones to later on use these for more accurate results in the testing phase.

We’re going to do 30 epochs.

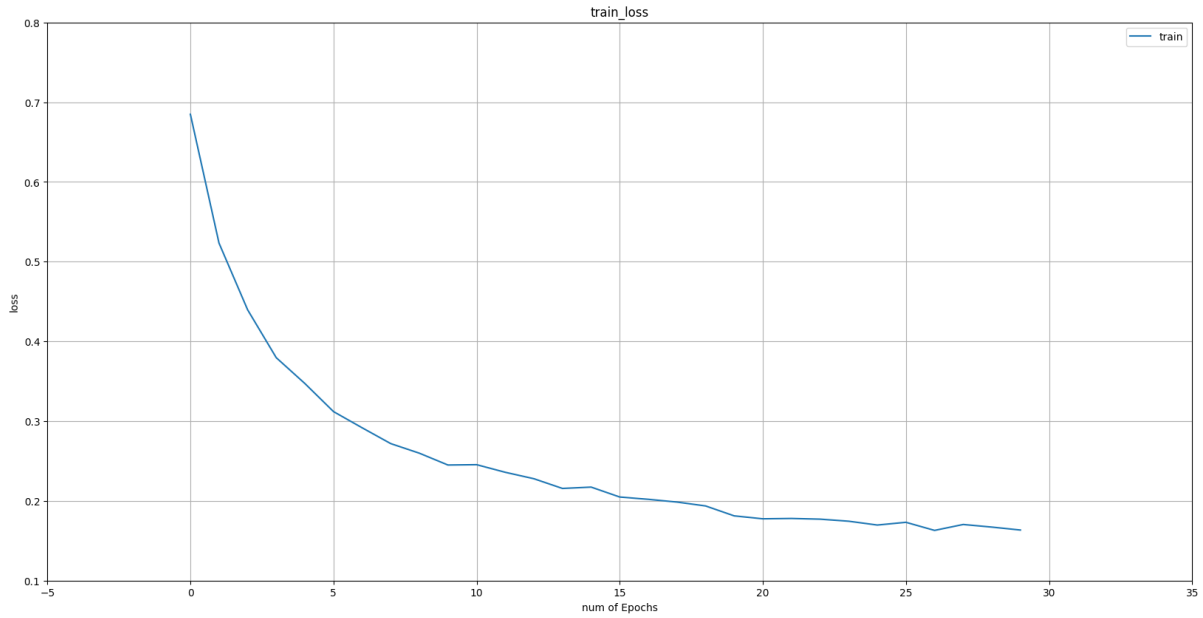


Figure 2: training loss wrt num of Epochs

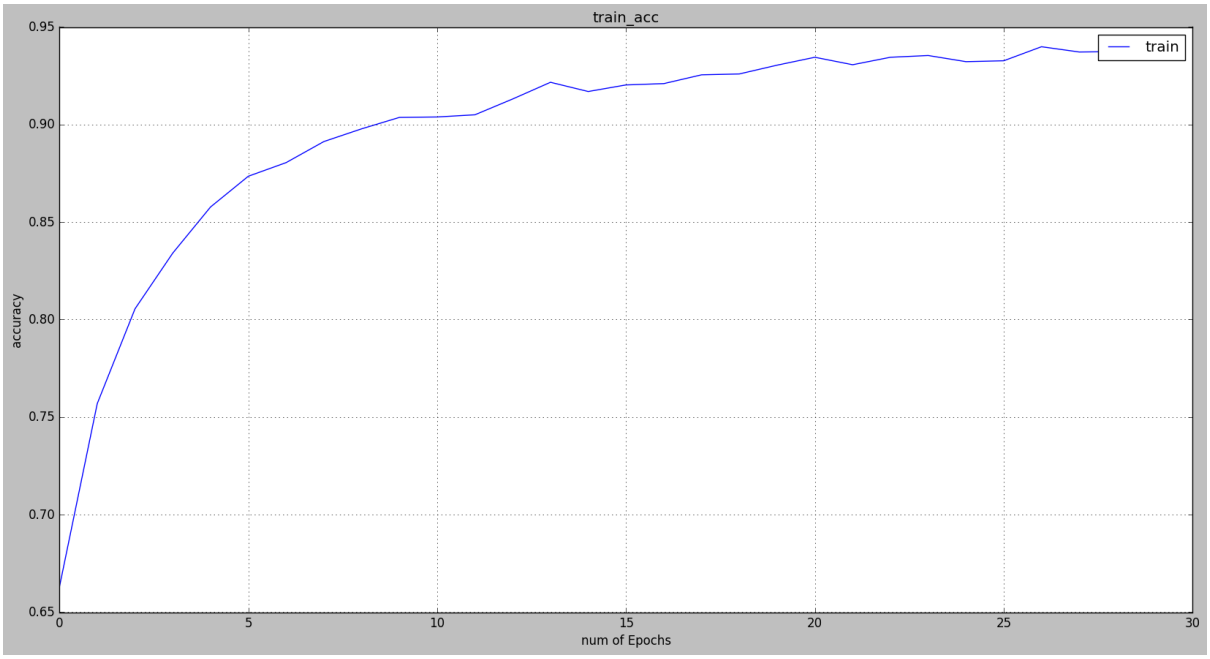


Figure 3: training accuracy wrt num of Epochs

Epochs	loss	accuracy
10	0.2450	0.9037
20	0.1812	0.9305
30	0.1634	0.9408

The trainig took an average of 615s per Epochs.

The whole training was done in 19542.6s.

```
Transform targets to keras compatible format.
(3244, 224, 224, 3) (3244, 2)
------------------------------
Predicting on the test data...
------------------------------
102/102 [==============================] - 481s 5s/step
Test_Accuracy =  0.09802712700369914
```

The testing that was done using a smaller dataset:

```
------------------------------
Creating test images...
------------------------------
healthy 165
Done: 0/165 images
Done: 100/165 images
infected 14
179
Loading done.
Transform targets to keras compatible format.
(179, 224, 224, 3) (179, 2)
------------------------------
Predicting on the test data...
------------------------------
6/6 [==============================] - 25s 4s/step
Test_Accuracy =  0.9217877094972067
```

- ROC curve:

Let us now compute the performance metrics for the pretrained Resnet model with the test data. The performance metrics involve computing the ROC-AUC values, cross-entropy loss score, average precision score, prediction probabilities and storing these values and plotting the ROC curves.

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate (TPR):

$$TPR = \frac{TP}{TP + FN}$$

- False Positive Rate (FPR):

$$FPR = \frac{FP}{FP + TN}$$

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.

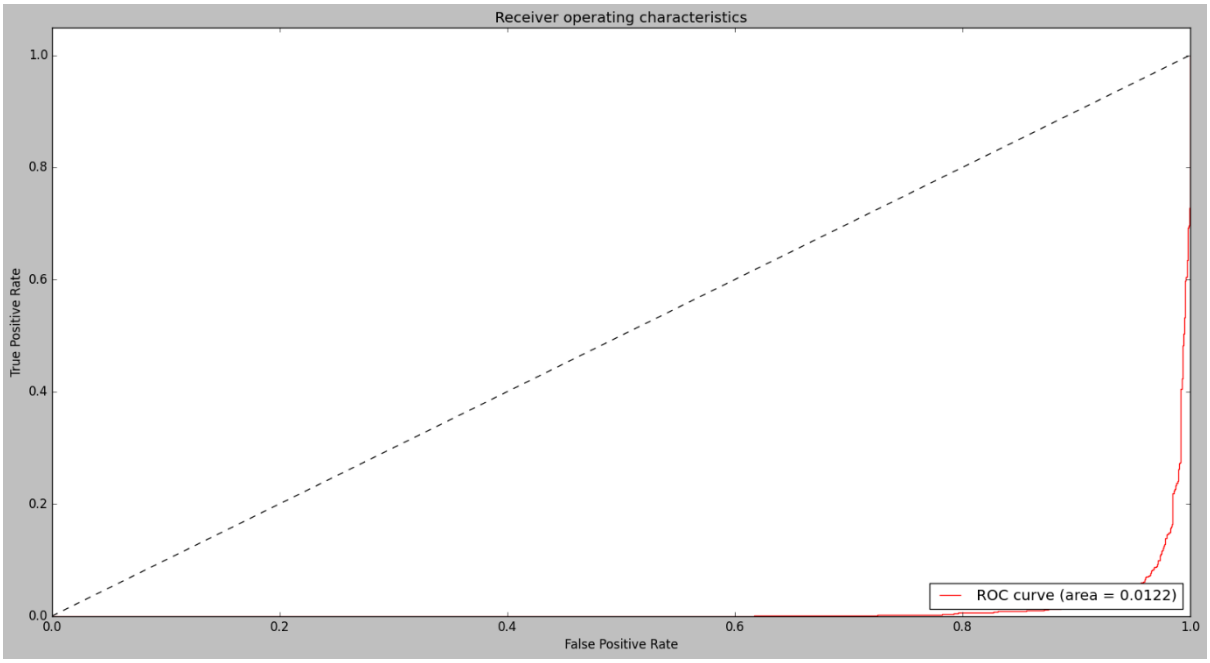


Figure 4: Area under the ROC Curve

Area under the ROC Curve (AUC) ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

AUC is desirable for the following two reasons:

- AUC is scale-invariant. It measures how well predictions are ranked, rather than their absolute values.

- AUC is classification-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

We get the following values:

- Score: 3.3395
- prec_score: 0.3077

- Confusin Matrix:

Now we plot the confusion matrix of the model's performance.

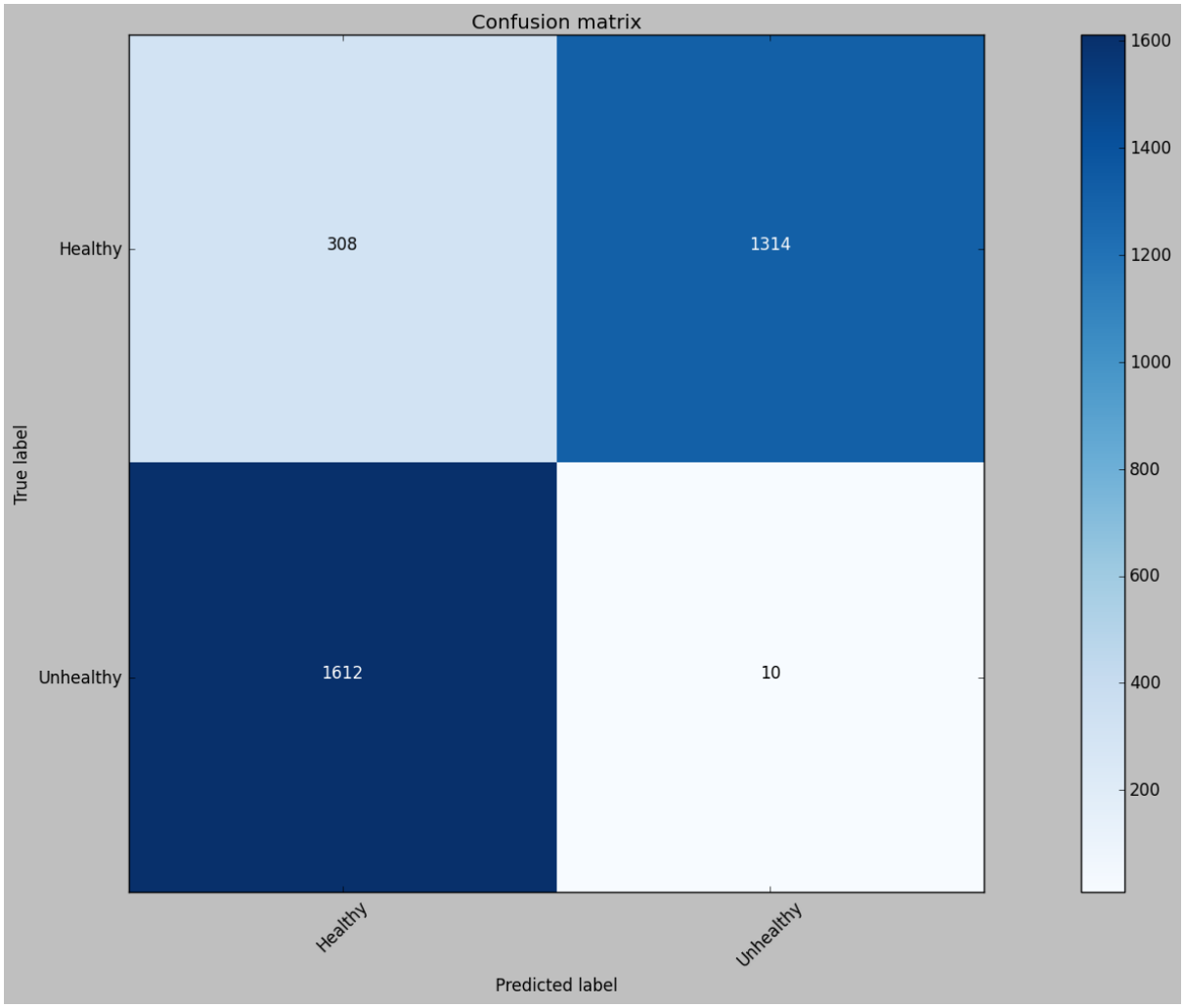


Figure 5: Confusion matrix of the model’s performance

This confusion matrix does indeed highlight the problem of missclassfying the cells.

	precision	recall	f1-score	support
Healthy	0.16	0.19	0.17	1622
Unhealthy	0.01	0.01	0.01	1622
accuracy			0.10	3244
macro avg	0.08	0.10	0.09	3244
weighted avg	0.08	0.10	0.09	3244

```
[[ 308 1314]
 [1612   10]]
Confusion matrix, without normalization
[[ 308 1314]
 [1612   10]]
```

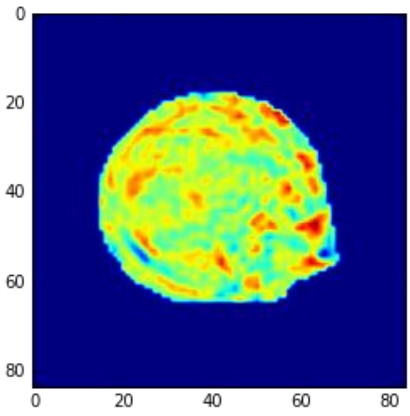
Grad-CAM:

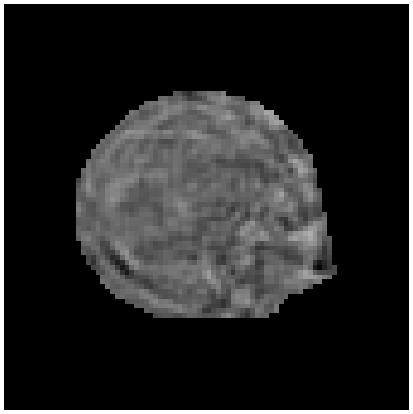
Grad-CAM or Gradient -weighted class activation mapping is a localization approach, it's a technique used for identifying discriminative regions used by a restricted class of image classification CNNs which do not contain fully connected layers.

Why is it used? To explain why model predicted what they predicted, especially in the medical field.

These visualization techniques answer two important questions: Why did the network think this is a parasitic cell? Where is the parasite seen in the cell image? In particular, it's interesting to note that the areas showing the parasites are strongly activated: this is probably how the network can tell the difference between normal and abnormal (parasitic) cells.

Testing different codes: (not finalized yet)





The error using different codes is the same:

```
494   if src_graph is None:

RuntimeError: tf.gradients is not supported when eager execution is enabled. Use tf.GradientTape instead.
```