Capstone project Malaria detection

Erick Martínez

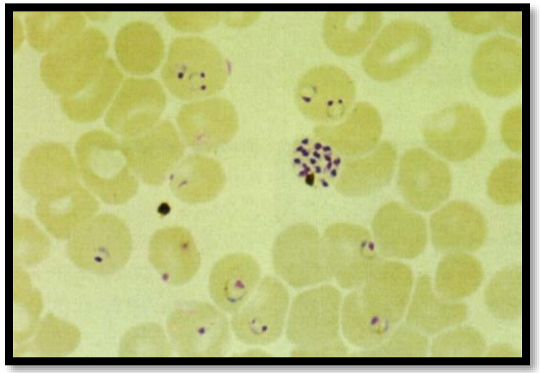
Content:

- Problem definition
- Data exploration
- Proposed model solution
- Model solution results
- Partial conclusions
- Final model
- Conclusions

Problem definition

Malaria disease caused by **Plasmodium parasites**

Transmitted by female Anopheles **mosquitoes**



Plasmodium falciparum, under the microscope

The parasites begin damaging red blood cells (RBCs), which can result in respiratory distress and other complications.

Almost **50%** of the world's **population** is in danger from malaria

From 229 million malaria cases were **400,000** malaria-related **deaths** (2019)

Children are the **most vulnerable** population group 67% of all malaria deaths (2019).

Great Learning
Kevin M. & Catherine W., What is malaria?, World Healt (1998)

Problem definition

One of the **most important** strategies for the reduction of malaria deaths is **early detection**, since **late treatment** can cause complications and could even be **fatal**.

PROBLEM TO SOLVE

Traditional diagnosis:

- Requires **careful inspection** by an **experienced professional** to discriminate between healthy and infected red blood cells
- **Tedious** process
- **Time-consuming** process
- The diagnostic **accuracy** can be **adversely impacted** by inter-observer variability

PROPOSED SOLUTION

Develop an **automated system** able to **help with the early and accurate detection of malaria.**

Convolutional neural networks (**CNN**) models tend to have a better performance at higher amounts of data

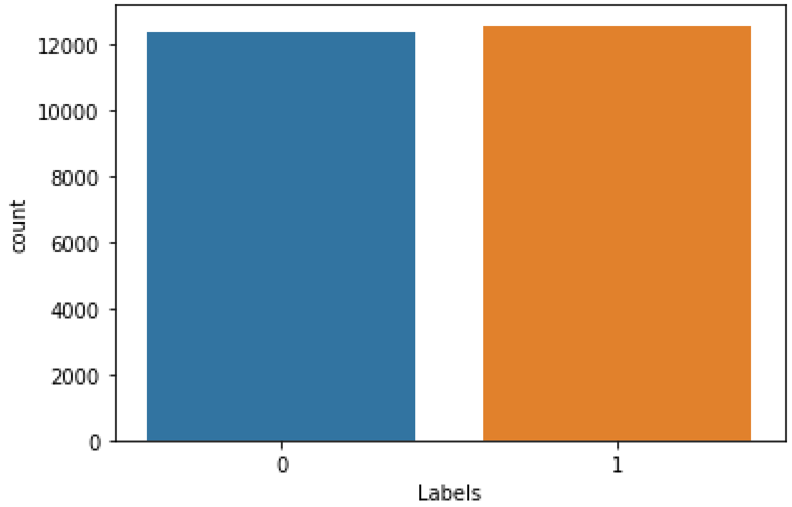
Objective:

Build an efficient computer vision **model** to detect malaria.
Identify whether the images of blood cells are **infected** with malaria **or not**.
Classify the images as **parasitized** or **uninfected** respectively.

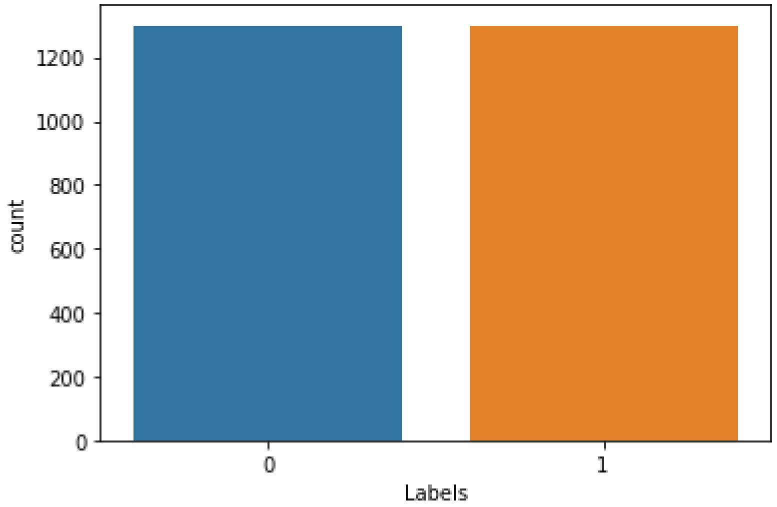
Data exploration

Total of **27558** colored **images** of red blood cells

Train set (**24958** images)



Test set (**2600** images)



0 = Uninfected
1= Parasitized

A **distribution of** approximately **50/50 %** can be observed between images of uninfected and parasitized cells in both sets of images.

represents

Good balance to develop a **CNN model** with good performance

Proposed model solution

CNN Base model structure;

1. Convolution + Relu
   Pooling + Dropout
2. Convolution + Relu
   Pooling + Dropout
3. Convolution + Relu
   Pooling + Dropout

Feature learning

4. Flatten
   Dense+ Dropout
   Softmax

Classification

Total of **1,058,786** trainable params

Confusion matrix and **definitions** of variables to be monitored

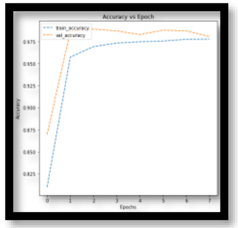
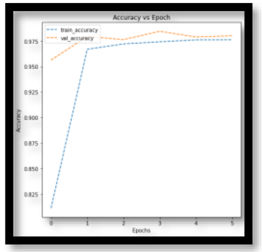
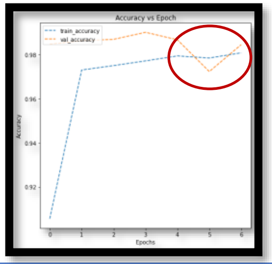
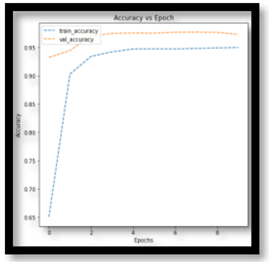
Parasitized	Uninfected
TP	FP
FN	TN

Recall = TP/(TP+FN)

Accuaracy = (TP+TN)/(TP+FP+FN+TN)

It is **desirable to reduced** as much as possible to an acceptable precision value because it is a public health problem.

Model solution results

		Recall	Accuracy	" Stability " evolution of accuracy as a function of epochs	
Base model		0,98	0,98	**Non significative fluctuating** behavior	
Model 1 (relu and Dropout)	= Base model → CONV2D (100 neurons), CONV2D (50 neurons)	0,98	0,97	**Non significative fluctuating** behavior	
Model 2	= Model 1 → "BatchNormalization", "LeakyRelu"	0,98	0,98	**Discarded** to avoid problems to converge **Significative fluctuating** behavior	
Model 3	= Model 1 + Data augmentation	0,98	0,98	**Non significative fluctuating** behavior	

Partial conclusions

The results suggest that the **increment** of the **number of layers** does **not favor** the monitoring variables especially **recall**

The incorporation of **"LeakyRelu"** and **"BatchNormalization"** as a substitute for “relu” and “Dropout” does **not favor** model stability.

Final model proposal

- **Remove the Dense layer** from the base model
- **Incorporate Gaussian Blurring** methodology

Note: is well-known that the **Gaussian Blurring** methodology can help make our machine learning **models more resilient to the harsh realities** they will encounter in real-world situations

Final model

CNN Final model structure;

1. Convolution + Relu
   Pooling + Dropout
2. Convolution + Relu
   Pooling + Dropout
3. Convolution + Relu
   Pooling + Dropout

Feature learning

Flatten

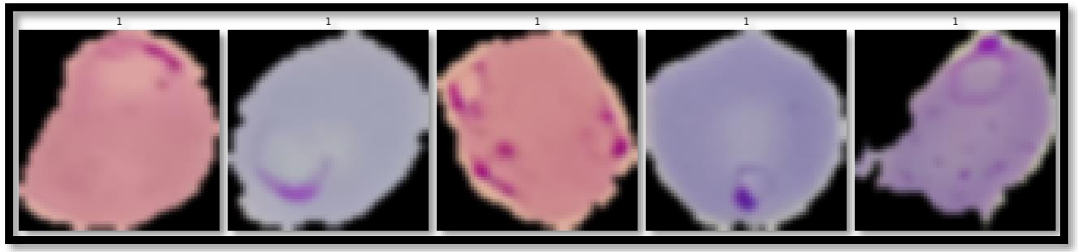
Softmax

Classification

Total of **12,770** trainable params

- **Significant reduction** in the trainable parameters
- **Reduction** in **computational requirements** and machine **time to converge**.

Gaussian Blurring technique

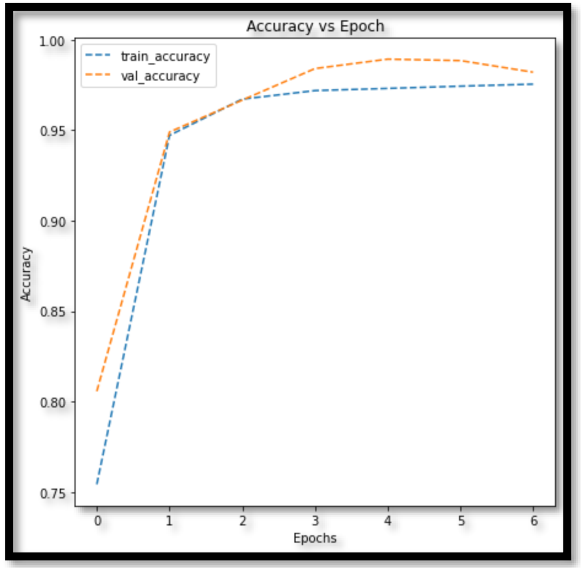


Results;

Recall = 0,99

Accuracy = 0,98

" Stability “



Conclusions

PROPOSED BUSINESS SOLUTION

- Alternatives in disease detection strategies are needed
- Investing in the neural networks branch for efficient and accurate detection of the disease.
- The application of similar models (transfer learning) to the one studied in this work could be functional especially in the branch of imaging used in the health area, which is very diverse and has a very important impact on vital aspects of society such as public health and its economy.

EXECUTING BUSINESS SOLUTION

- It is proposed that for the implementation of the studied model a transition plan should be carried out, not to displace the professional detection personnel, but rather that, the detection personnel together with data science professionals monitor the studied model for its validation in the field.
- Conduct information campaigns in the target population regarding the advantages of the proposed model, in order to overcome the delay in the implementation of the new malaria detection strategy, because there is a natural resistance to the adoption of new strategies and technologies in the general population.

EXECUTIVE SUMMARY

- CNN model can detect malaria through effective identification and classification of blood cell images
- Its proper implementation would help vulnerable groups such as children under 5 years of age, pregnant women, and HIV/AIDS patients.
- The CNN model studied for malaria detection is not a tedious, time-consuming process and the diagnostic accuracy does not depend too much on inter-observer variability. Therefore, a reduction in the economic costs of disease detection and an increase in the accessibility of diagnosis are expected.

Thank you

Appendix

Parasitized	Uninfected
TP	FP
FN	TN

		Parasitized	Uninfected
Precision	=	TP/(TP+FP)	TN/(TN+FN)
Recall	=	TP/(TP+FN)	TN/(TN+FP)

Accuaracy = (TP+TN)/(TP+FP+FN+TN)

f1-score = 2*Precision*Recall/Precision+Recall

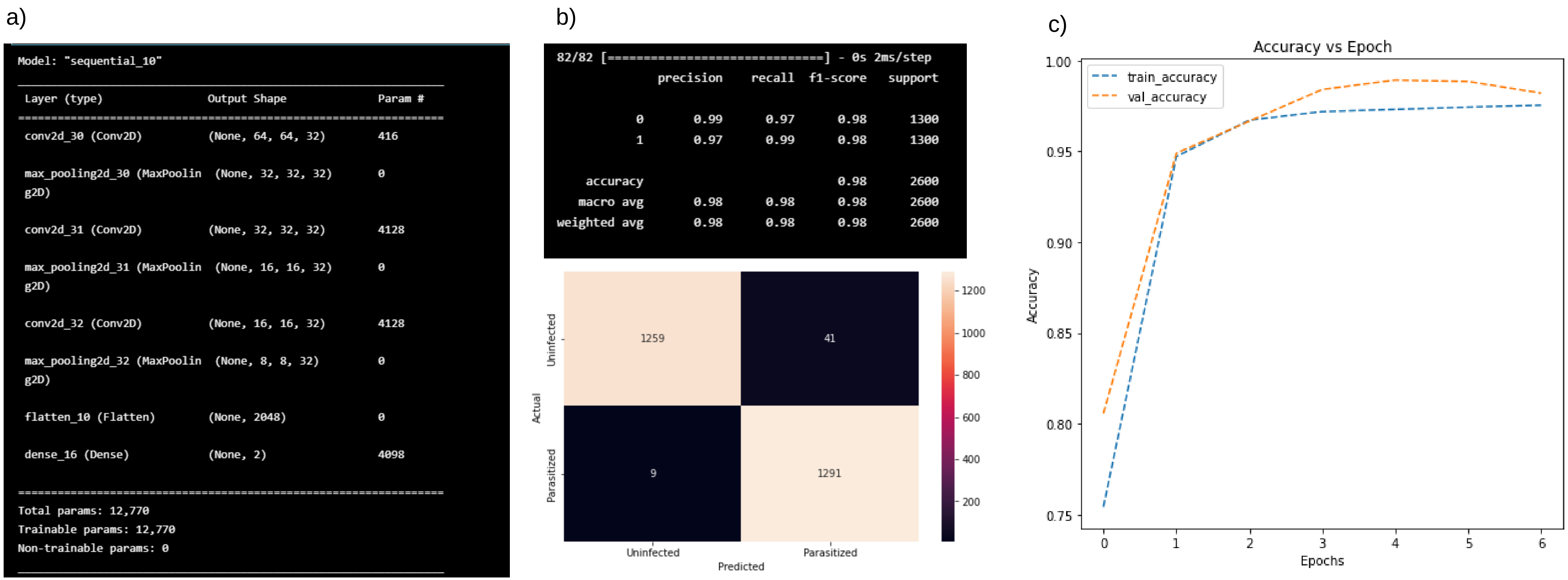


Figure 13. Structure (Layers) (a)), general report of performance (b)) and evolution of accuracy as a function of epochs (c)) for validation and train data relative to the final model

Gaussian Blurring on train data

```
import cv2
gbx = []  # To hold the blurred images

for i in np.arange(0, 24958, 1):

    b = cv2.GaussianBlur(train_images[i], (5, 5), 0)

    gbx.append(b)

gbx = np.array(gbx)

gbx.shape
```

[11]

```
(24958, 64, 64, 3)
```

Building the model

```
# Creating sequential model
model3 = Sequential()

# Build the model here

model3.add(Conv2D(filters = 32, kernel_size = 2, padding = "same", activation = "relu", input_shape = (64, 64, 3)))

model3.add(MaxPooling2D(pool_size = 2))

model3.add(Conv2D(filters = 32, kernel_size = 2, padding = "same", activation = "relu"))

model3.add(MaxPooling2D(pool_size = 2))

model3.add(Conv2D(filters = 32, kernel_size = 2, padding = "same", activation = "relu"))

model3.add(MaxPooling2D(pool_size = 2))

model3.add(Flatten())

model3.add(Dense(2, activation = "softmax"))

# Use this as the optimizer
adam = optimizers.Adam(learning_rate = 0.001)

model3.compile(loss = "binary_crossentropy", optimizer = adam, metrics = ['accuracy'])

model3.summary()
```

[Sin titulo]

61]

Using Callbacks

```
callbacks = [EarlyStopping(monitor = 'val_loss', patience = 2),
             ModelCheckpoint('.mdl_wts.hdf5', monitor = 'val_loss', save_best_only = True)]
```

[62]

Fit and train our Model

```
# Fit the model with min batch size as 32 can tune batch size to some factor of 2^power ]
history = model3.fit(gbx, train_labels, batch_size = 32, callbacks = callbacks, validation_split = 0.2, epochs = 20, verbose = 1)
```

[63]

```
Epoch 1/20
624/624 [==============================] - 5s 7ms/step - loss: 0.5223 - accuracy: 0.7543 - val_loss: 0.5719 - val_accuracy: 0.8057
Epoch 2/20
624/624 [==============================] - 4s 6ms/step - loss: 0.1608 - accuracy: 0.9471 - val_loss: 0.2243 - val_accuracy: 0.9489
Epoch 3/20
624/624 [==============================] - 4s 7ms/step - loss: 0.0943 - accuracy: 0.9671 - val_loss: 0.1478 - val_accuracy: 0.9665
Epoch 4/20
624/624 [==============================] - 4s 6ms/step - loss: 0.0822 - accuracy: 0.9718 - val_loss: 0.0967 - val_accuracy: 0.9840
Epoch 5/20
624/624 [==============================] - 4s 6ms/step - loss: 0.0767 - accuracy: 0.9730 - val_loss: 0.0684 - val_accuracy: 0.9892
Epoch 6/20
624/624 [==============================] - 4s 6ms/step - loss: 0.0737 - accuracy: 0.9743 - val_loss: 0.0690 - val_accuracy: 0.9884
Epoch 7/20
624/624 [==============================] - 4s 6ms/step - loss: 0.0711 - accuracy: 0.9754 - val_loss: 0.0788 - val_accuracy: 0.9820
```

Evaluating the model on test data

```
accuracy = model3.evaluate(test_images, test_labels, verbose = 1)
print('\n', 'Test_Accuracy:-', accuracy[1])
```

```
82/82 [==============================] - 0s 4ms/step - loss: 0.0865 - accuracy: 0.9808

 Test_Accuracy:- 0.9807692170143127
```