Seeing Malaria using Deep Learning

Blake Wallace

Capstone Technical Report

Final Submission

Jan 21, 2022

Executive Summary

What are the key takeaways? What are the key next steps?

The primary objective of this project was to learn about the various layers used in the architecture of a convolutional neural network (CNN). To accomplish this task, a CNN model has been created to evaluate photos of red blood cells and determine if they are infected with the Plasmodium parasite that causes malaria. During the process various iterations of convolutional, max pooling, batch normalization, dropout, flattening, and fully connected layers were used to attempt to optimize the model performance.

A simple base model was first trained. Then subsequent models were constructed as modifications to this base model. During the process there were two alternative base models used. One was a simple model constructed in a standard way. The other was constructed in a way similar to a custom model proposed by Rahman [1].

The performance of the predictions of a model was assessed using the model accuracy. The goal was to minimize any errors from the model.

The two models considered as potential solutions were chosen because of their slightly higher overall range of performance measures during the model exploration phase. At that time the "best" performing model was one of the more complicated models in that it had more than 2.5 times the number of parameters to train relative to the second option. This made the second model more appealing because of its quick training times.

The final solution was chosen on several bases. First, it is considered lightweight for having less unknown parameters to train then the other options. Second, it is considered good-fitting as the model was training with high training and validation accuracies, and the validation accuracy was slightly lower than the training. Third, the range of model accuracies was decently high in that all models that successfully trained had an accuracy greater than 95.2%.

Problem and Solution Summary

What problem was being solved? What are the key points that describe the final proposed solution design? Why is this a 'valid' solution that is likely to solve the problem?

The problem being solved is to empower a computer to determine if a red blood cell has the Plasmodium parasite that causes Malaria by feeding it a photo of the cell. In practice human intervention is being used to make this determination.

There were two model designs that were explored during this phase. Ultimately, the second model performed better than the first. But, this was not the case at the completion of the model selection process during Milestone 2.

The first is a bit more complex of a model having 4.8M parameters to train. But, it performed with the highest minimum accuracy model during the model selection phase. Its overall accuracy during training fell between 95.11% and 95.49%.

The second model design is simple in that it has less than 1.9M parameters to learn. This was on the smaller side, compared to the other trained models. This solution has consistently performed with accuracy greater than 95%. During the final model selection process, the amount of patience and number of epochs was increased. Also, one change has been made in the architecture, namely four of the activation functions that were previously ReLU have been changed to Leaky ReLU.

The final proposed model architecture is:

- Conv2D(filters=32, kernel_size=(3, 3), padding="same", input_shape=(64, 64, 3)) with LeakyReLU(0.1)
- Conv2D(filters=32, kernel_size=2, padding='same', activation='relu')
- MaxPool2D(pool_size=(2, 2))

- Dropout(0.3)
- Conv2D(filters=64, kernel_size=(3, 3), padding='same') with LeakyReLU(0.1)
- Conv2D(filters=64, kernel_size=2, padding='same', activation='relu')
- MaxPool2D(pool_size=(2, 2))
- Dropout(0.3)
- Conv2D(filters=128, kernel_size=(3, 3), padding='same') with LeakyReLU(0.1)
- Conv2D(filters=128, kernel_size=2, padding='same', activation='relu')
- MaxPool2D(pool_size=(2, 2))
- Dropout(0.3)
- Conv2D(filters=256, kernel_size=(3, 3), padding='same') with LeakyReLU(0.1)
- Conv2D(filters=256, kernel_size=2, padding='same', activation='relu')
- MaxPool2D(pool_size=(2, 2))
- Dropout(0.30)
- Flatten()
- Dense(256)
- Dropout(0.5)
- Dense(64)
- BatchNormalization()
- Dropout(0.5)
- Dense(2, activation='softmax')

Initially, all of the convolution layers had kernel size of 3x3 and the ReLU activation function. However, after training dozens of models it was deemed better to make some changes to the kernel sizes and the activation functions that are reflected in the above model. Note that after the flattening layer, which puts the data into a single vector for continued processing, all of the "Dense" layers have no activation. Models were trained using both ReLU and Leaky ReLU at these layers, but none of them performed as well.

Table 1 below has information about 30 iterations of this trained model. Note that three different values for the patience of the model were used, 40, 20, and 10. Patience is a number that determines how long the model will continue to train without seeing an improvement in the metric that is being monitored.

Notice that there are three iterations that trained with an accuracy of 0.5. In these model iterations the model simply predicted all of the unseen data as part of the positive class. Also, note that there was one iteration of the model where the model score was above 96%. Otherwise, all of the models scored within the 95% range. The actual range of the middle 24 models is 95.23% to 95.99%.

Table 1: Model Accuracy

Patience*	**Epochs****	**Accuracy*****
40	59/300	0.9592
40	56/300	0.9599
40	79/300	0.9561
40	79/300	0.9526
40	46/300	0.5
40	58/300	0.9546
40	56/300	0.9584
40	46/300	0.5
40	46/300	0.5
40	79/300	0.9592
20	29/100	0.9576
20	40/100	0.9599
20	42/100	0.9565
20	28/100	0.9599
20	42/100	0.9553
20	42/100	0.9575
20	42/100	0.9553
20	52/100	0.9546
20	42/100	0.9538
20	44/100	0.9569
10	11/75	0.9573
10	17/75	0.9573
10	26/75	0.9569
10	11/75	0.9557
10	32/75	0.9526
10	17/75	0.9573
10	19/75	0.9546
10	30/75	0.9603
10	32/75	0.9523
10	17/75	0.9565

*This is the number of epochs allowed to run without seeing an improvement in the validation loss.
**The number of epochs used in training / the total number of epochs allowed during training.
***Testing Accuracy used to determine the performance of the model on unseen data.

Many of the models were good fitting. By this it is meant that during training the validation accuracy was slightly below the training accuracy, and that both were high. Take Figure 1 for example. This is a graph of the accuracy during the training steps. It corresponds to the bottom most model in the table with patience of 40.

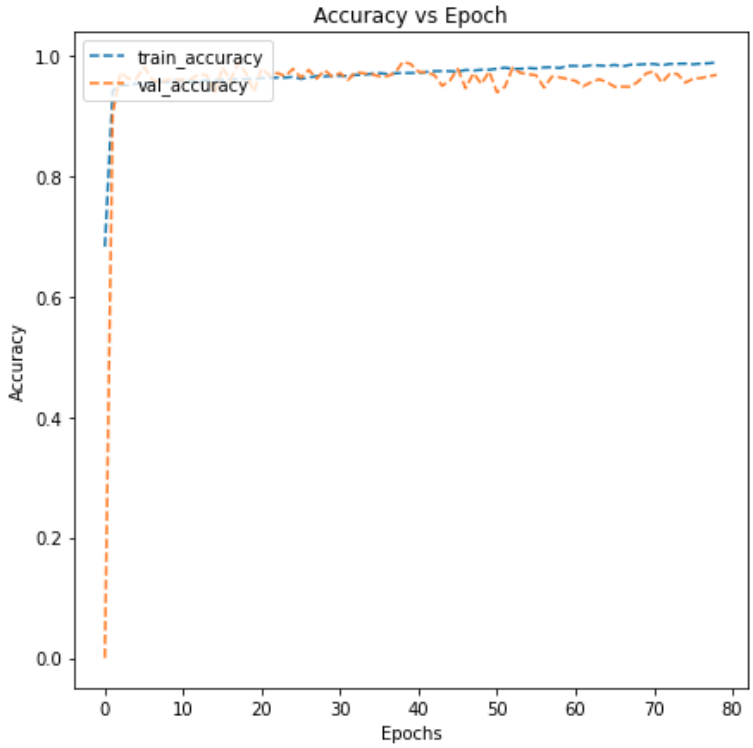

Figure 1: The final training accuracy is 0.9878. The final validation accuracy is 0.9671. The model accuracy on unseen data is 0.9592.

As a contrast to Figure 1, Figure 2 shows a model with the Leaky ReLU activation function.

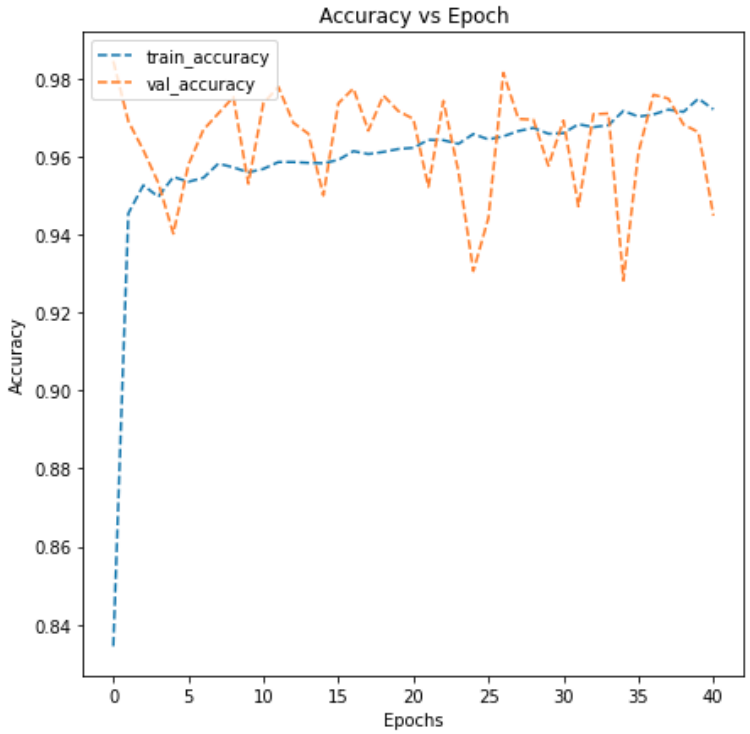

Figure 2: The final training accuracy is 0.9722. The final validation accuracy is 0.9449. The model accuracy on unseen data is 0.9515.

As can be seen in the above graphs, the example without Leaky ReLU was more consistent in its growth over time during the training process.

Recommendations for Implementation

What are some key recommendations to implement the solutions? What are the key actionables for stakeholders? What is the expected benefit and/or costs? What are the key risks and challenges? What further analysis needs to be done or what other associated problems need to be solved?

Key risks and challenges to actually implementing this model do exist. During the model construction phase the overall accuracy was used to evaluate its performance. Ultimately, the goal was to minimize errors in the model predictions. In practice it is potentially more valuable to emphasize the importance of not telling a patient they are well when they are sick, as opposed to telling a patient they are sick when they are not. In the ladder case further medical testing would be needed to validate the finding. What the actual costs are associated with these two alternative errors probably resides in the question of local access to medical facilities where proper testing can be performed to validate the findings from the model. This is something any stakeholder should consider carefully before putting this model into production.

For further analysis it could be beneficial to consider different metrics separate from accuracy. For example, to consider minimizing errors where the model determines a blood cell has no parasite when in fact it does. It could be possible to train a model that makes this prediction well without sacrificing to much in the accumulation of other types of errors. More testing is needed to answer this question.

Another consideration for further analysis, note that the primary objective of this project was to learn about the various model layers that are commonly used to create a Convolutional Neural Network. Towards this aim a base model was modified by adding to or changing various aspects of its architecture, and then comparing the model accuracy to the accuracy of the base model. The problem with this method was that almost all of the models had an accuracy between 95.1% and 95.9% regardless of the changes made. So, while the approach provided a metric of comparison, it is believed that the model comparison to the particular unseen data was biased towards the particular set of testing data. This caused some difficulty in actually determining an optimum model architecture, because making changes did not have much impact on the overall model accuracy. At the current stage in the process it is believed that there are two potential ways to fix this problem. First, find more data. The data set that is being used is quite small, making it hard for the model to learn all the important various aspects of the data during training. An alternative approach to addressing this issue would be to consider creating a third holdout

set, which can be used to actually make comparisons between the different models. The final model performance can then be determined from the separate set of unseen data.

References

[1] Aimon Rahman et al. *Improving Malaria Parasite Detection from Red Blood Cell using Deep Convolutional Neural Networks*. 2019. arXiv: 1907.10418 [eess.IV].