Malaria Detection

DEEP LEARNING PROJECT

MIT Applied Data Science Program

Capstone Project: Deep learning

Delia McNamara

14 August 2024

CONTENTS / AGENDA

→ Executive Summary

→ Problem Definition

→ Solution summary

→ Data Exploration

→ Model Performance Comparison

→ Final Solution Design Proposal

→ Recommendations for implementation

EXECUTIVE SUMMARY

- → Use deep learning to diagnose Malaria infection.
  - ◊ Create convolutional neural network (CNN) model to classify imaged cells as parasitized or uninfected.
  - ◊ Intention to utilize the CNN to overcome global deficit of healthcare access in diagnosing Malaria.
- → 5 CNNs created and compared.
  - ◊ Focus on overall accuracy & recall for parasitized samples.
  - ◊ 3 CNN models with hidden layers including convolutional, dense, normalization, dropout, and flatten layers.
  - ◊ 1 CNN model with data augmentation.
  - ◊ 1 pre-trained CNN model (VGG16).
- → Best model (CNN Model 3) had an accuracy of 97% and a recall for parasitized samples of 98%.
  - ◊ Tried this model with data augmentation.
    - ■ Worse performance with augmentation.
- → Other models ranged from an accuracy of 93.1 to 98.0%
- → CNN Model 3 recommended for use in diagnostics for overall accuracy, best fit, and lowest proportion of false negatives.

PROBLEM DEFINITION

→ 400,00 deaths worldwide caused by Malaria in 2019.

→ Malaria is caused the *Plasmodium* parasite, which can be imaged in cells for diagnosis.

→ 229 million cases of Malaria diagnosed in 2019.

- ◊ Traditional diagnosis requires a lab technician to detect parasite presence in sample via microscope.
    - ■ This labor issue limits global access to proper healthcare options for Malaria diagnosis.
- ◊ Strong need for quick & accurate diagnosis.

→ Objective to build CNN models to predict the presence of *Plasmodium* parasite in imaged tissue samples.

- ◊ Decrease labor needs and increase access to diagnostic tools without compromising patient outcomes.
- ◊ Focus on high accuracy while minimizing false negatives in models.
    - ■ False negatives are patients who have Malaria, but test results indicate they do not have Malaria.
    - ■ Extremely risky for patients – if they test negative, they will likely not be given the treatment they need.
        - Potentially lethal outcomes for these patients.

SOLUTION SUMMARY

→Data preprocessed at import for consistency of dimensions.

- ◊ Imported pre-split into test & train sets.

→Data exploration performed to visualize images & how augmentation may affect them.

- ◊ Images displayed in original RGB format.
- ◊ Average image calculated & visualized.
- ◊ Gaussian blur applied.
- ◊ Images converted to HSV format.

→Labels one-hot encoded for use in neural network.

→CNN Model 1 performed with convolutional, dense, dropout, max pooling, & flatten layers.

→CNN Model 2 performed with similar layers but used Tanh activation rather than ReLU.

→CNN Model 3 performed with similar layers but used Leaky ReLU activation, added normalization, & removed one dropout layer for better fit.

→CNN Model 3 then performed with data augmentation.

- ◊ Used Image Data Generator to augment.
- ◊ Horizontal flip, rotation, & zoom applied.

→Pre-trained VGG16 model performed.

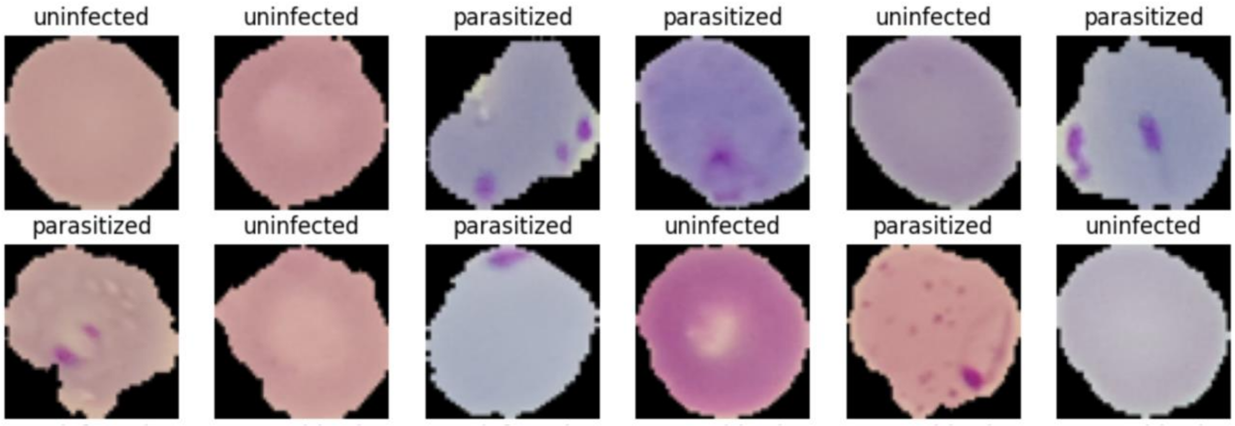
DATA EXPLORATION

DATA OVERVIEW

→ 27,558 images total in dataset.

→ Data labeled, split into train & test sets prior to import.

- ◊ 24,958 train images.
  - ▪ 12,376 images labeled uninfected.
  - ▪ 12,582 images labeled parasitized.
- ◊ 2,600 test images.
  - ▪ 1,300 images labeled uninfected.
  - ▪ 1,300 images labeled parasitized.



→ Images resized to 64 x 64 px at import for uniformity.

- ◊ Prevent model from training on imaging artifacts.

→ Cell images fairly uniform.

- ◊ All have black background.
- ◊ Parasites appear to have a pink to purple hue.

→ Variable hue for cells in images.

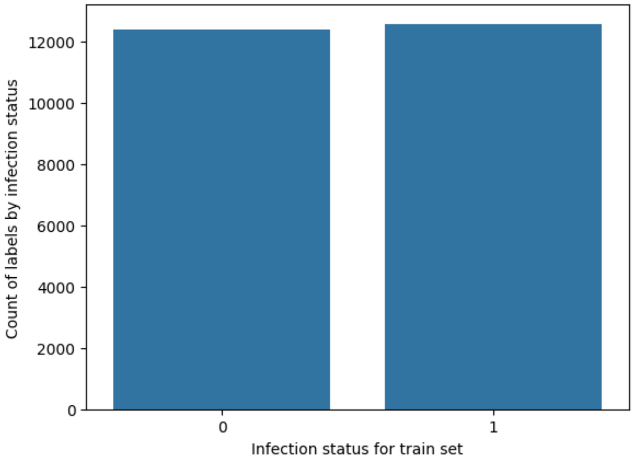
- ◊ Shades of pink, orange, purple, blue.

→ Some uninfected images appear to have a brightness issue with the center of the cell more brightly lit than the perimeter.

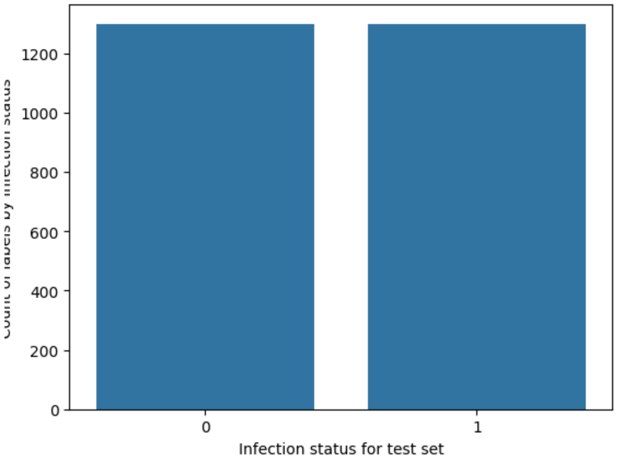
CNN PREPROCESSING

NO IMAGE AUGMENTATION

TRAIN SET



TEST SET



→ Dataset pre-split for model training & validation.

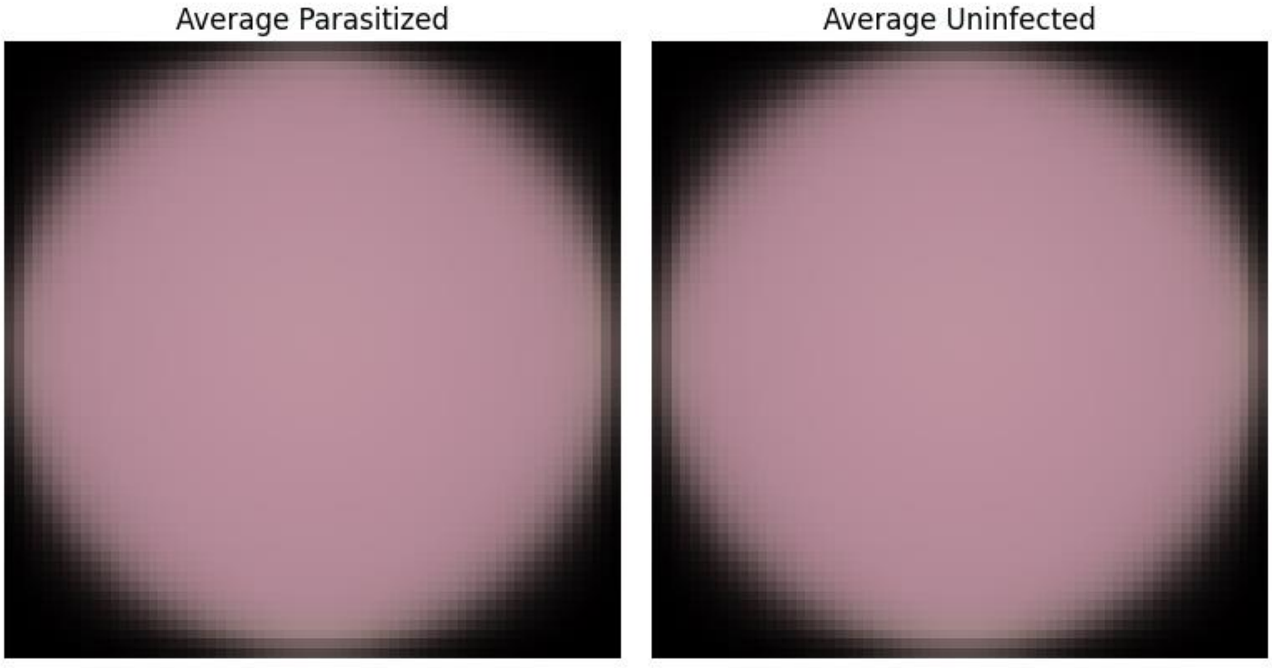
- ◊ 24,958 images to train (90%).
- ◊ 2,600 images to test (10%).
- ◊ 0 represents uninfected, 1 represents parasitized.

→ Good balance between uninfected & parasitized samples.

- ◊ Model will not be biased by the label split.

→ Train & test set values normalized & converted to float data type.

- ◊ Imported with range of pixel values from 0 to 255.
- ◊ Normalized to values of 0 to 1.
- ◊ Facilitates network learning by ensuring pixel values are within a similar range.

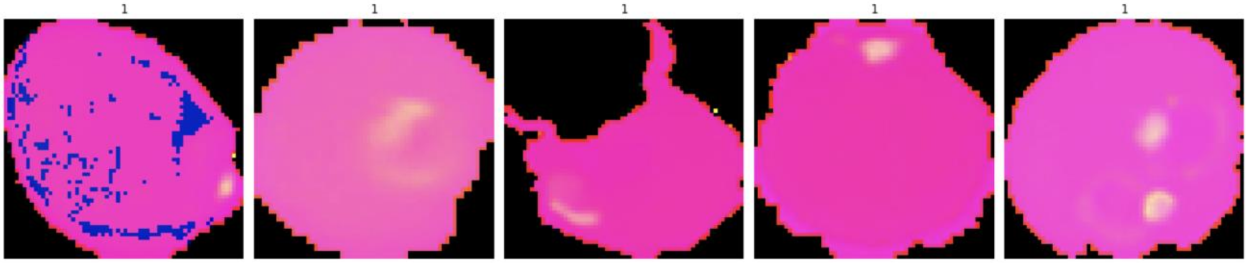


→ Calculated mean image for Parasitized & Uninfected.

→ Nearly indistinguishable.

- ◊ Good for model to learn to distinguish between the two based on the appearance of the parasite and not imaging artifacts or bias.
    - ■ E.g. all uninfected samples imaged on a white background, all infected samples imaged on a black background.

WITH IMAGE AUGMENTATION



→ Converted train & test images from RGB to HSV.

→ Visualize options for data augmentation.

→ Images more saturated with less discernable value ranges.

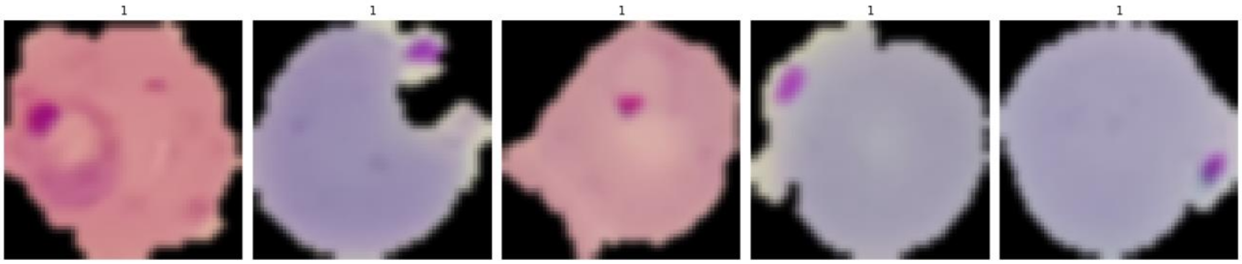
→ Consistent cell color between images.

- ◊ Better than original RGB.

→ Parasites appear to have a yellow hue.

→ One image contains many blue pixels.

- ◊ Some anomalies in this image format.



→ Gaussian blur applied to further visualize options for data augmentation.

→ Images less sharp.

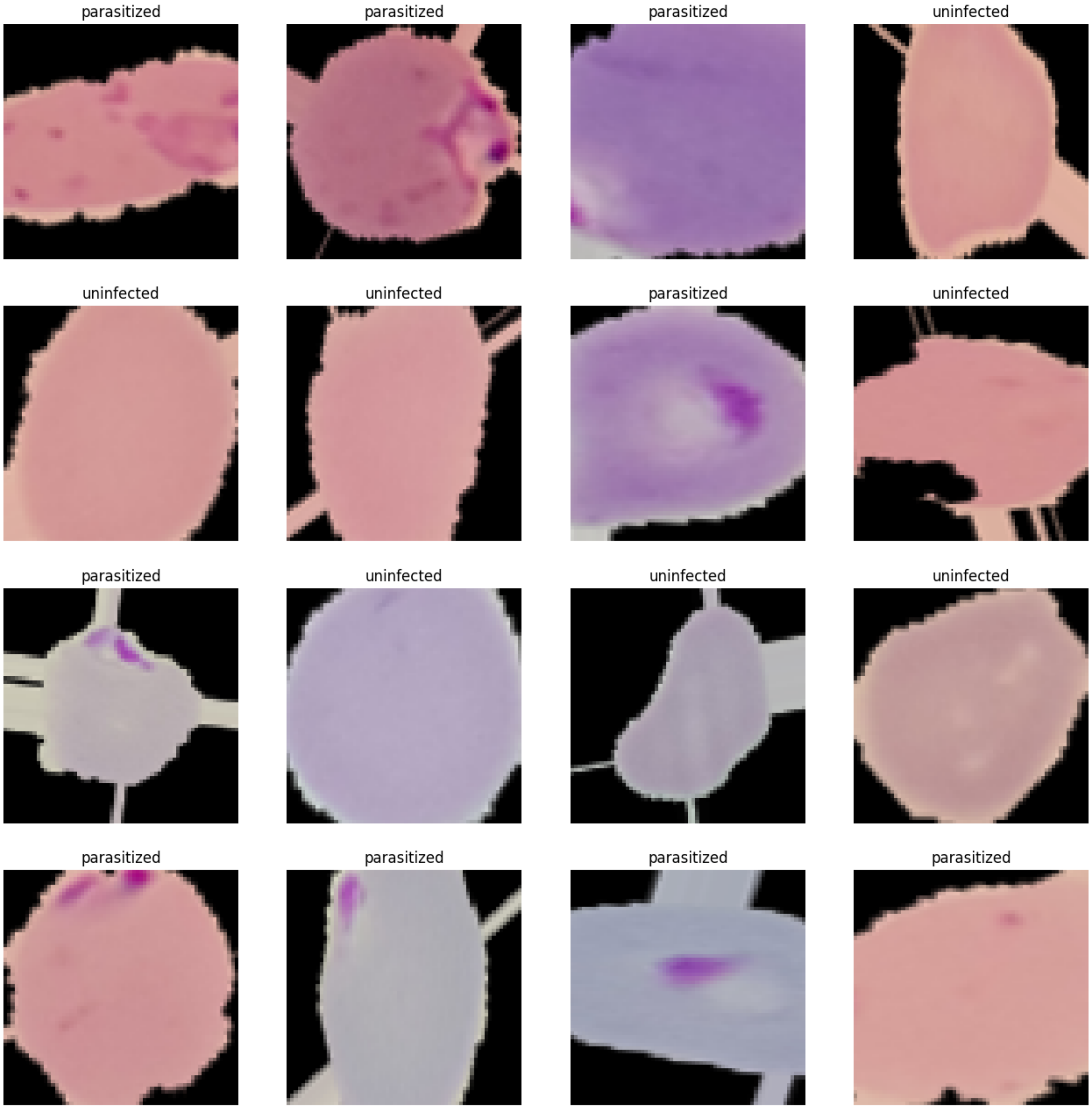
→ Parasites still visibly pink to purple.

- ◊ Edges are less distinguishable.

→ Cell colors still variable between shades of pink, orange, blue, purple.

→ Likely not a good use of augmentation.

- ◊ Blurring is better suited for reducing features in overly complex images – these images are simple.



→ Image Data Generator used to augment images.

→ Horizontal flip, rotation, & zoom applied.

→ Add variability for the model to learn.

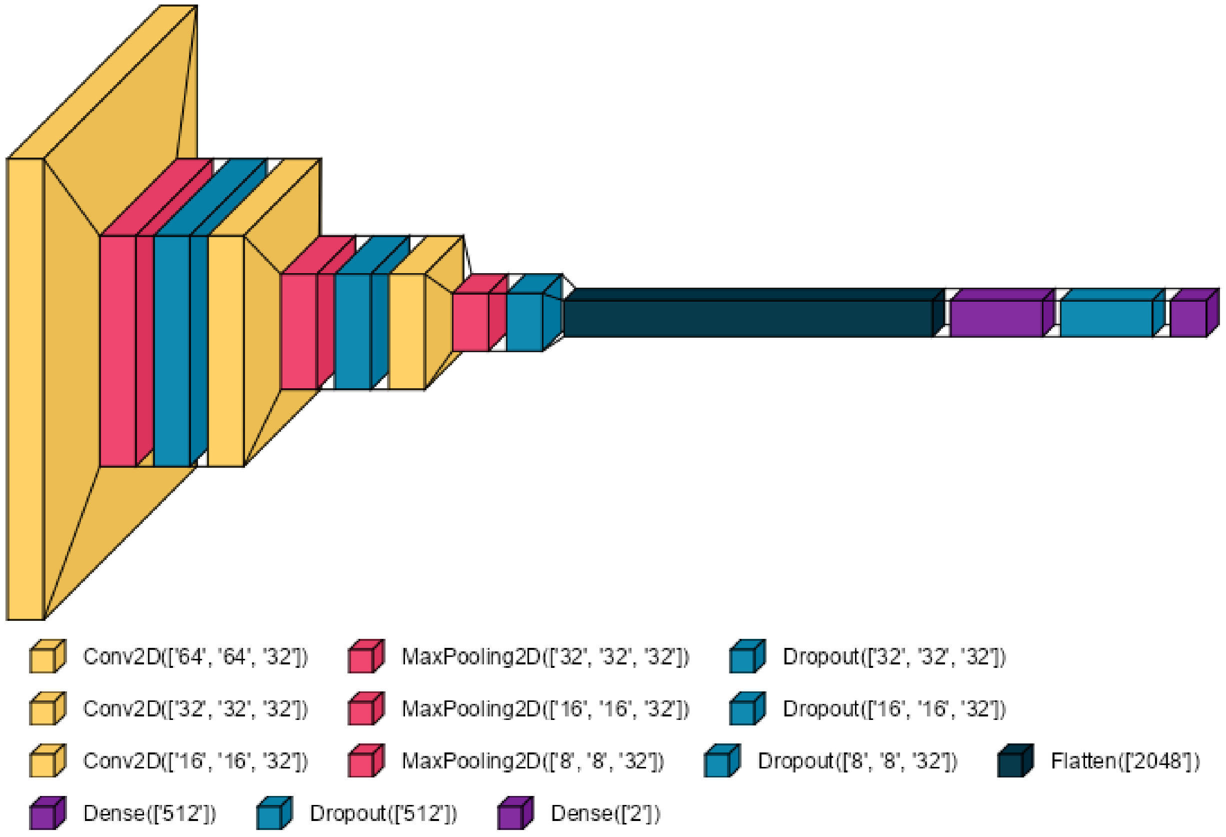
LABEL PREPROCESSING

→ One-hot encoded train & test set labels.

- ◊ 2 labels for each.
- ◊ Allows a dataset originally labeled with categorical labels to be used in neural network.

MODEL PERFORMANCE COMPARISON

CNN MODEL 1


Conv2D(['64', '64', '32'])
MaxPooling2D(['32', '32', '32'])
Dropout(['32', '32', '32'])
Conv2D(['32', '32', '32'])
MaxPooling2D(['16', '16', '32'])
Dropout(['16', '16', '32'])
Conv2D(['16', '16', '32'])
MaxPooling2D(['8', '8', '32'])
Dropout(['8', '8', '32'])
Flatten(['2048'])
Dense(['512'])
Dropout(['512'])
Dense(['2'])

→ CNN model with:

- 3 convolutional layers.
    - Output size same as input.
    - Kernel size 3 x 3.
    - ReLU activation
- 3 max-pooling layers.
    - Pool size 2 x 2.
- 4 Dropout layers
- 1 flatten layer.
- 2 dense layers.
- 1 output layer.
    - 2 nodes.
    - Softmax activation.

→ Compiled with:

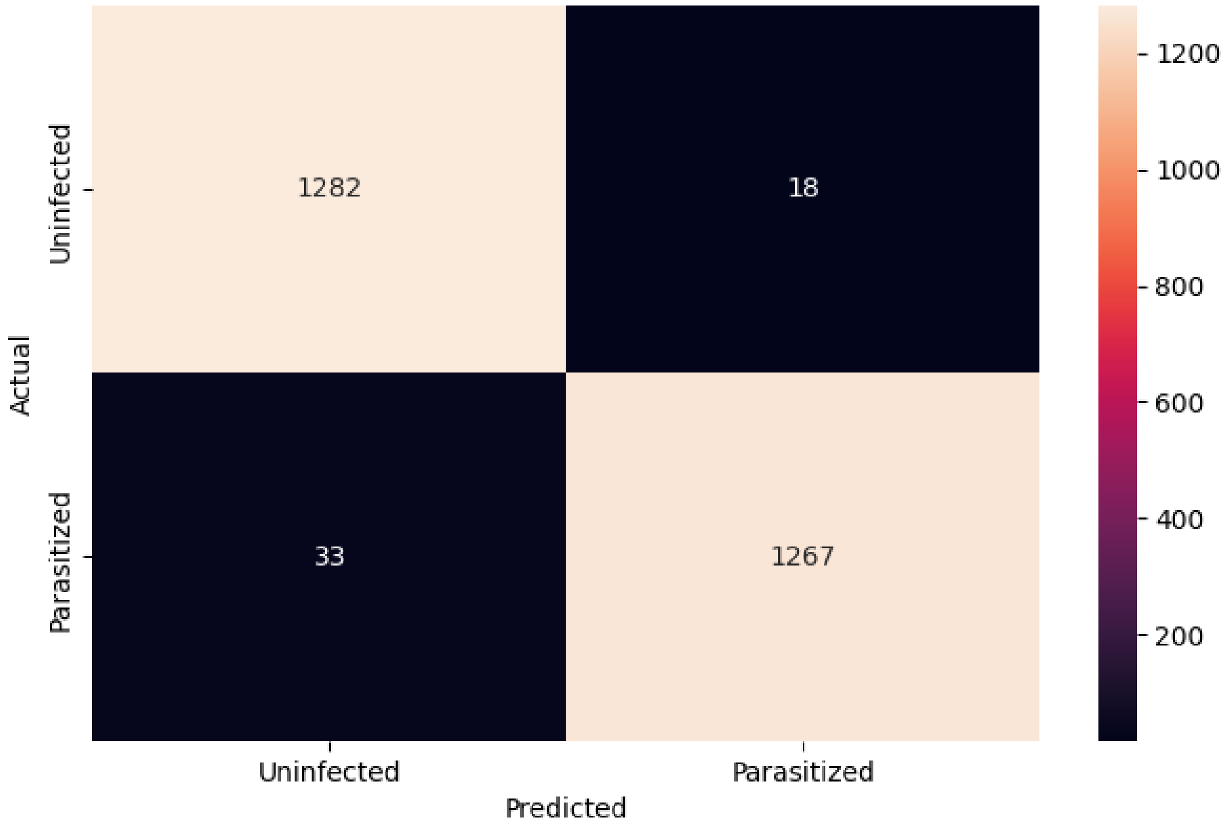
- Loss: categorical cross-entropy.
- Optimizer: Adam.
    - Learning rate: 0.001.

→ Fitted with:

- Callback for early stopping with validation loss monitor to prevent further epochs if loss is no longer decreasing.
- Validation split: 20%.
- Batch size: 32.
- Called with 20 epochs.
    - Ran 8 epochs.

→ Model performance on test data:

- Test accuracy 98.0%



```
              precision   recall  f1-score   support

           0       0.97      0.99      0.98      1300

           1       0.99      0.97      0.98      1300

    accuracy                           0.98      2600

   macro avg       0.98      0.98      0.98      2600

weighted avg       0.98      0.98      0.98      2600
```

→ 33 false negatives, 18 false positives.

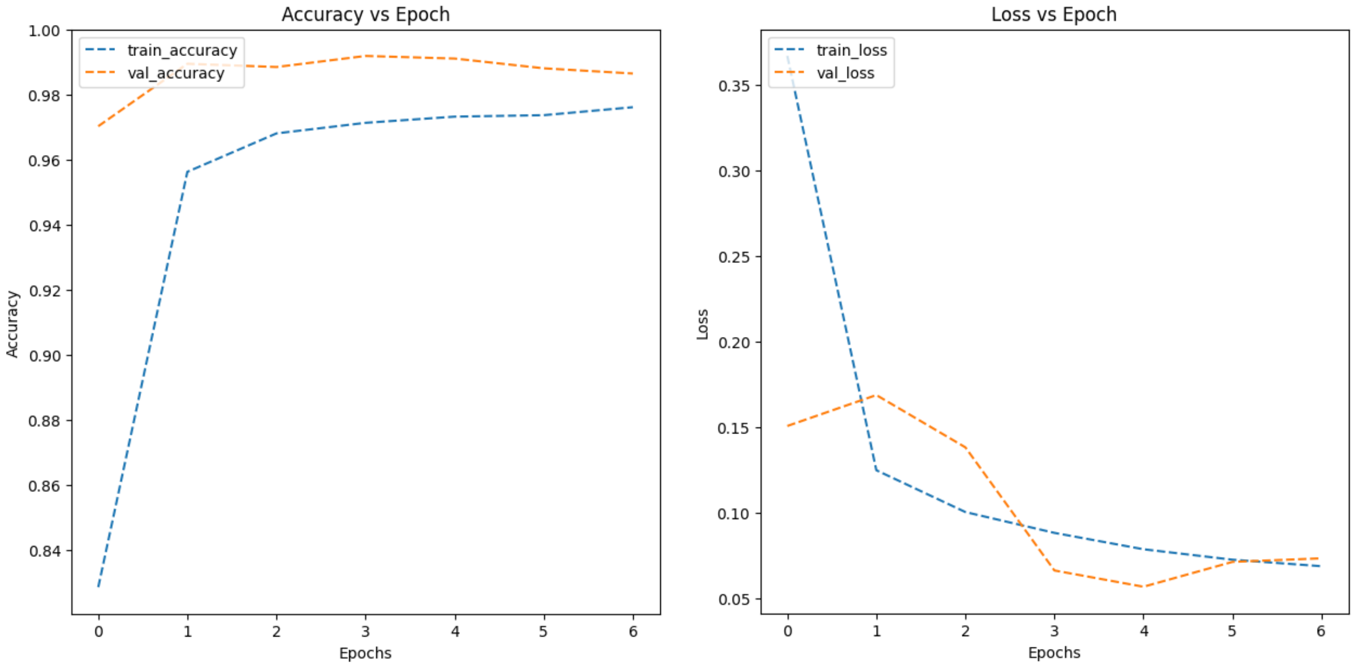
- ◊ More images classified as uninfected when actually parasitized.
- ◊ Seek to minimize this issue – false negatives are weighed heavily when the patient outcome can be lethal.

→ Good precision at 97% for uninfected & 99% for parasitized.

- ◊ Few false positives.

→ Good recall at 99% for uninfected, OK at 97% for parasitized.

- ◊ More important metric to avoid lethality to patients.
- ◊ Good result, but can be improved.



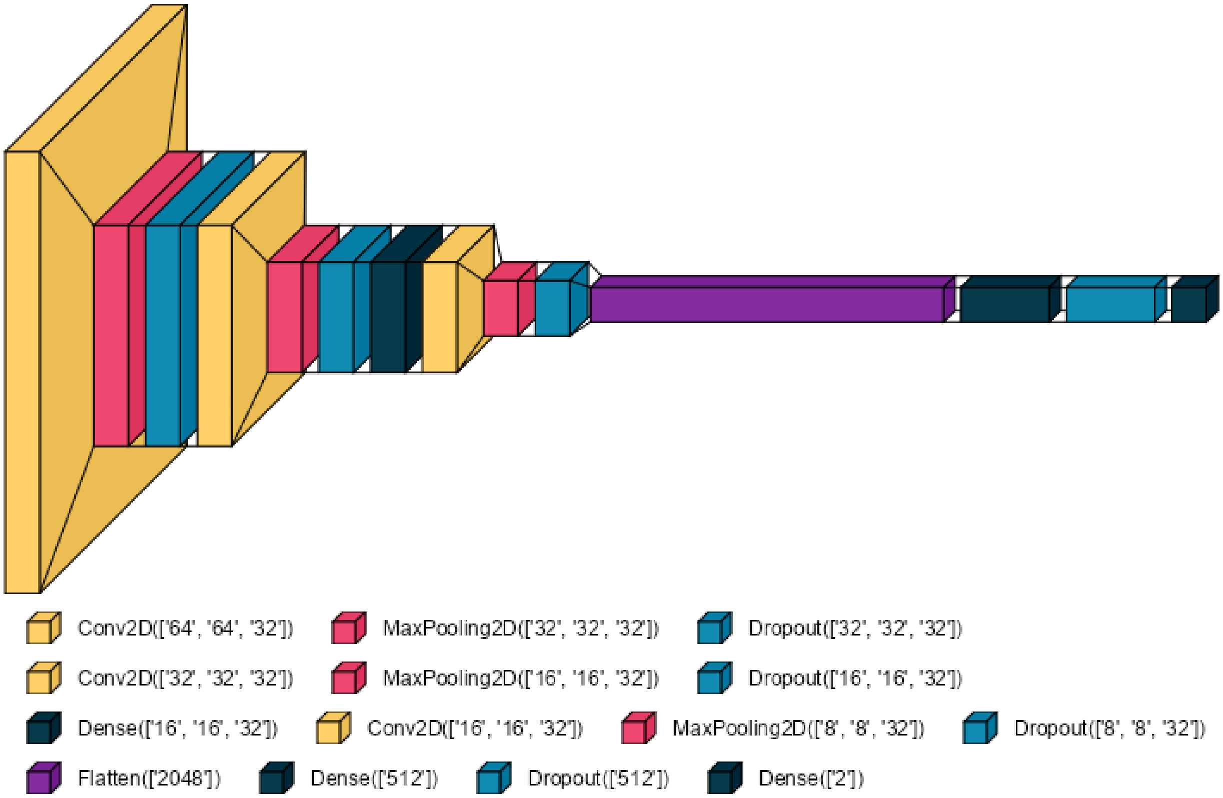
→ **Validation accuracy higher overall than test accuracy.**

◊ **May indicate some underfitting.**

→ **Validation loss stabilizes with training loss by epoch 5.**

◊ **Underfitting is not too bad.**

CNN MODEL 2


Conv2D(['64', '64', '32'])
MaxPooling2D(['32', '32', '32'])
Dropout(['32', '32', '32'])
Conv2D(['32', '32', '32'])
MaxPooling2D(['16', '16', '32'])
Dropout(['16', '16', '32'])
Dense(['16', '16', '32'])
Conv2D(['16', '16', '32'])
MaxPooling2D(['8', '8', '32'])
Dropout(['8', '8', '32'])
Flatten(['2048'])
Dense(['512'])
Dropout(['512'])
Dense(['2'])

→ CNN model with:

- ◊ 3 convolutional layers.
  - o Output size same as input.
  - o Kernel size 3 x 3.
  - o 2 ReLU activation.
  - o 1 tanh activation.
- ◊ 3 max-pooling layers.
  - o Pool size 2 x 2.
- ◊ 4 Dropout layers
- ◊ 1 flatten layer.
- ◊ 2 dense layers.
  - o Tanh activation.
- ◊ 1 output layer.
  - o 2 nodes.
  - o Softmax activation.

→ Compiled with:

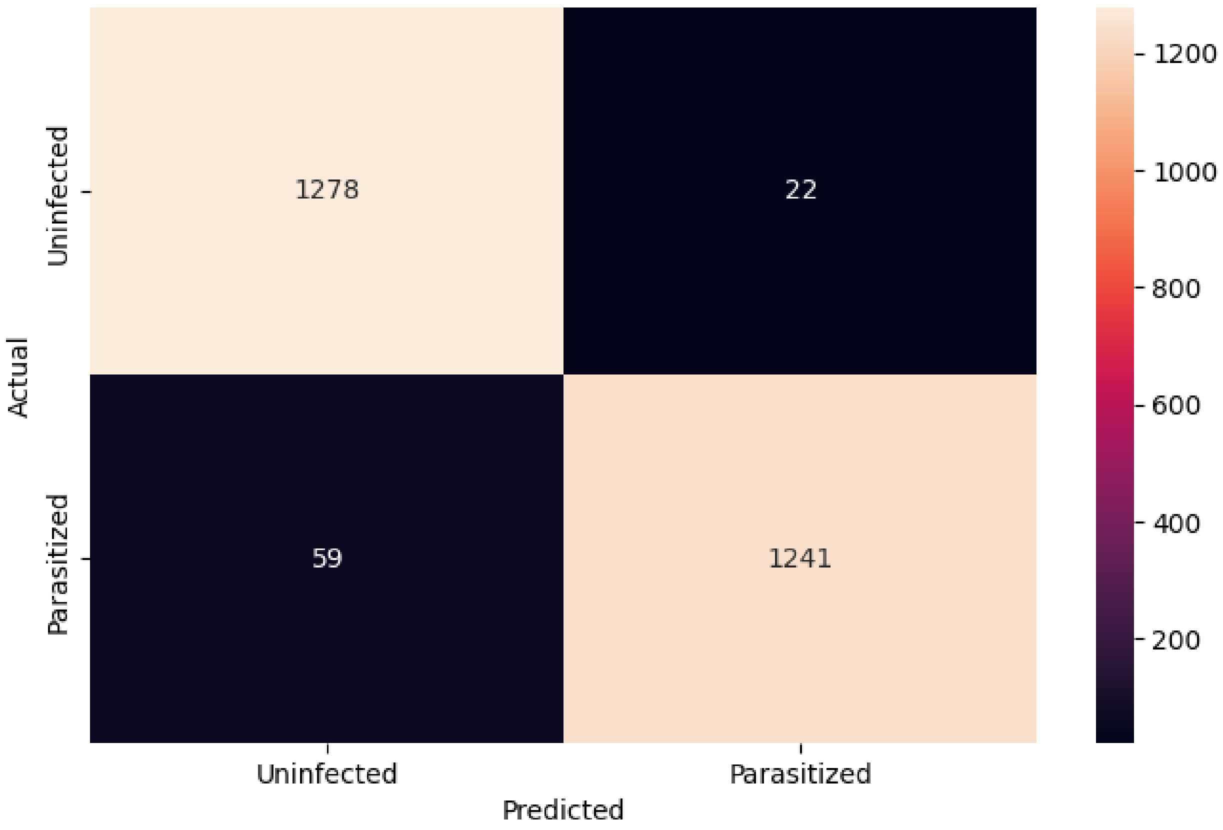
- ◊ Loss: categorical cross-entropy.
- ◊ Optimizer: Adam.
  - o Learning rate: 0.001.

→ Fitted with:

- ◊ Callback for early stopping with validation loss monitor to prevent further epochs if loss is no longer decreasing.
- ◊ Validation split: 20%.
- ◊ Batch size: 32.
- ◊ Called with 20 epochs.
  - o Ran 6 epochs.

→ Model performance on test data:

- ◊ Test accuracy 96.9%



```
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      1300

           1       0.98      0.95      0.97      1300

    accuracy                           0.97      2600

   macro avg       0.97      0.97      0.97      2600

weighted avg       0.97      0.97      0.97      2600
```

→ 59 false negatives, 22 false positives.

◊ More images classified as uninfected when actually parasitized.

◊ Seek to minimize this issue – false negatives are weighed heavily when the patient outcome can be lethal.

→ OK precision at 96% for uninfected & 98% for parasitized.

◊ Few false positives.

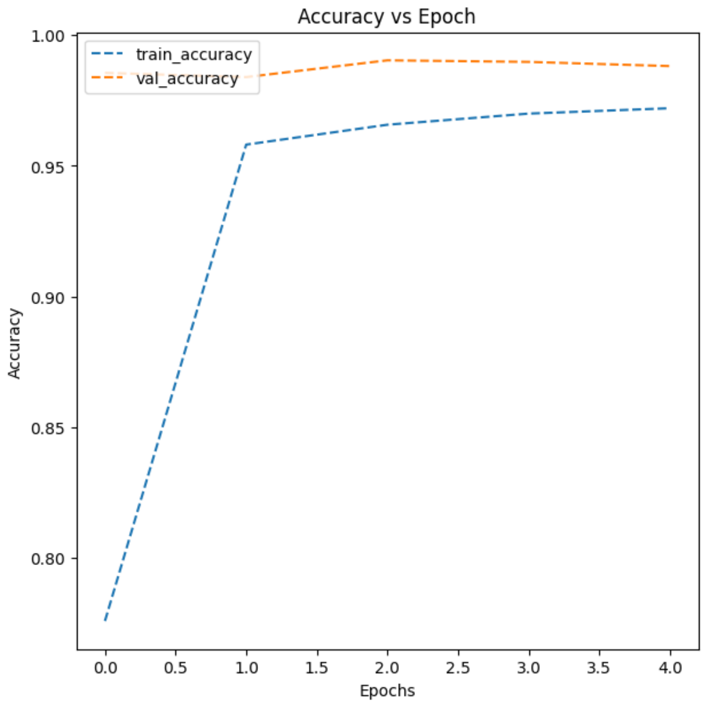
→ Fair recall at 98% for uninfected, poor recall at 95% for parasitized.

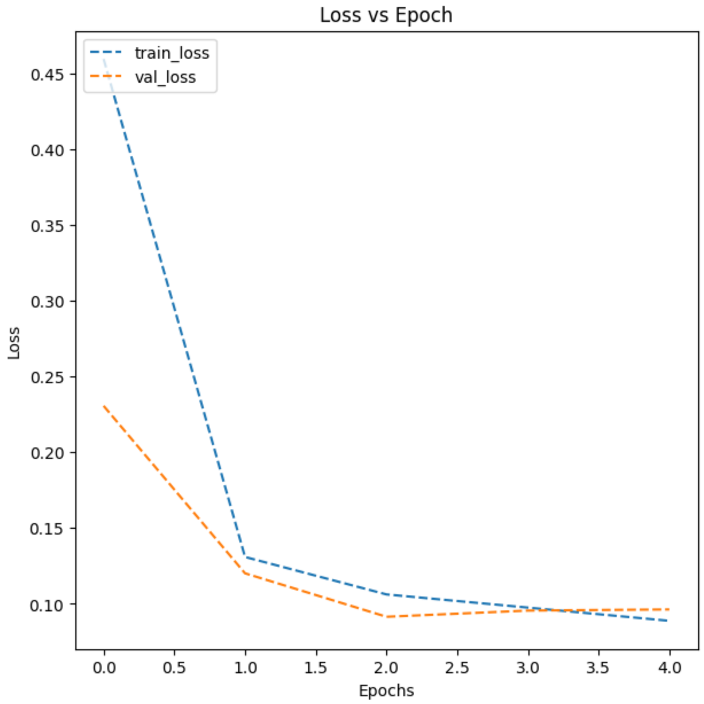
◊ More important metric to avoid lethality to patients.

◊ Poor result for parasitized samples.

→ CNN Model 1 fewer false negatives.

◊ Better option for patient outcomes.





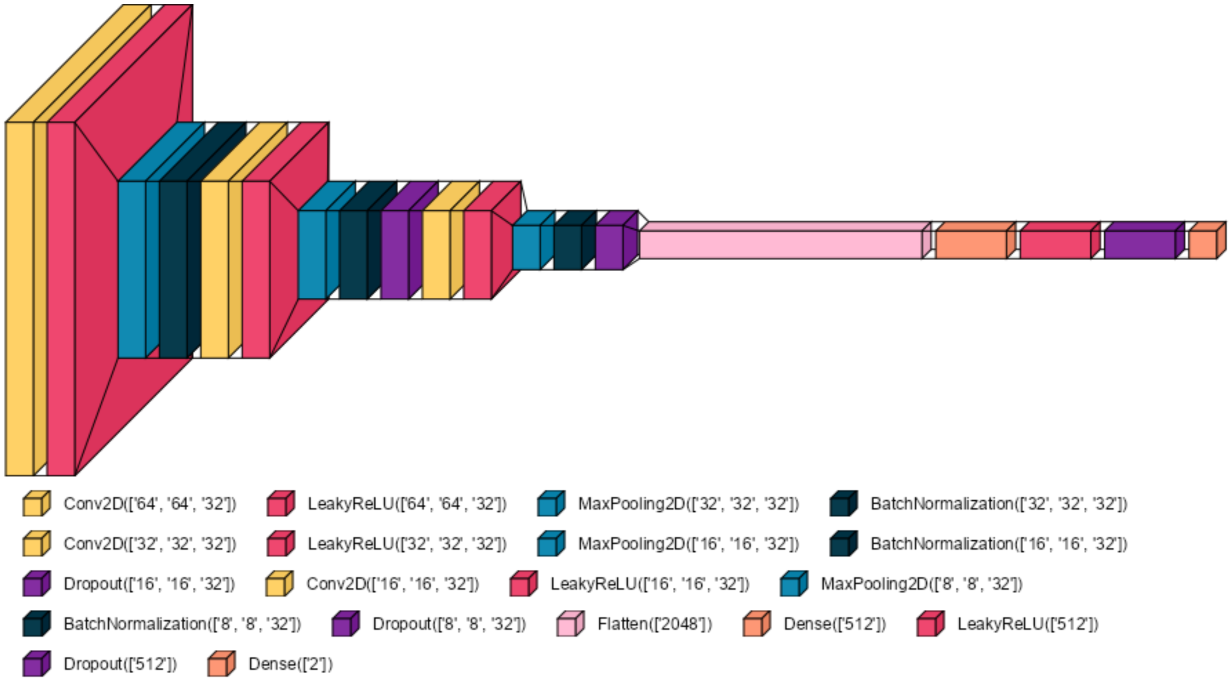
→ Validation accuracy higher overall than training accuracy.

- ◊ May indicate some underfitting.

→ Validation loss stabilizes with training loss by epoch 3.

- ◊ Underfitting is not too bad.

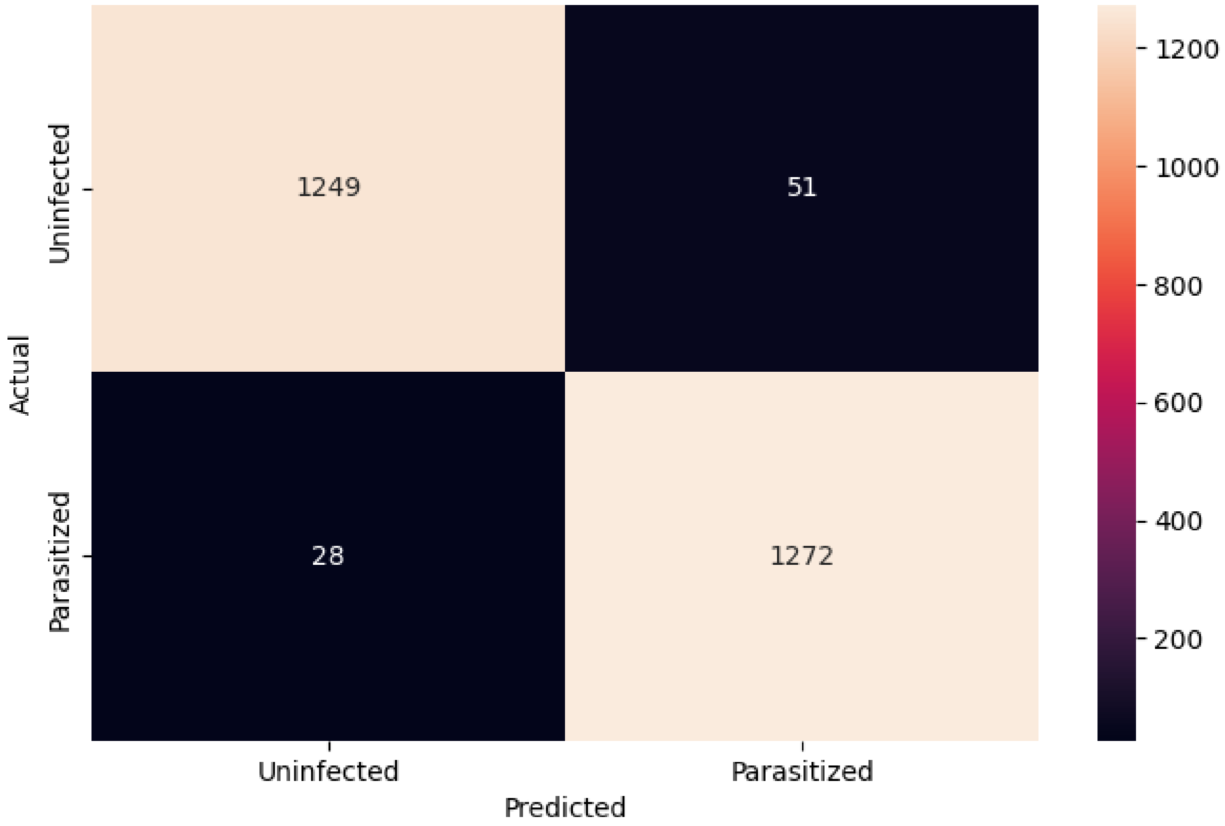
CNN MODEL 3


Conv2D(['64', '64', '32'])
LeakyReLU(['64', '64', '32'])
MaxPooling2D(['32', '32', '32'])
BatchNormalization(['32', '32', '32'])
Conv2D(['32', '32', '32'])
LeakyReLU(['32', '32', '32'])
MaxPooling2D(['16', '16', '32'])
BatchNormalization(['16', '16', '32'])
Dropout(['16', '16', '32'])
Conv2D(['16', '16', '32'])
LeakyReLU(['16', '16', '32'])
MaxPooling2D(['8', '8', '32'])
BatchNormalization(['8', '8', '32'])
Dropout(['8', '8', '32'])
Flatten(['2048'])
Dense(['512'])
LeakyReLU(['512'])
Dropout(['512'])
Dense(['2'])

- → CNN model with:
  - ◊ 3 convolutional layers.
    - o Output size same as input.
    - o Kernel size 3 x 3.
    - o Leaky ReLU activation.
  - ◊ 3 max-pooling layers.
    - o Pool size 2 x 2.
  - ◊ 3 Dropout layers
  - ◊ 1 flatten layer.
  - ◊ 2 dense layers.
    - o Leaky ReLU activation.
  - ◊ 1 output layer.
    - o 2 nodes.
    - o Softmax activation.

- → Compiled with:
  - ◊ Loss: categorical cross-entropy.
  - ◊ Optimizer: Adam.
    - o Learning rate: 0.001.
- → Fitted with:
  - ◊ Callback for early stopping with validation loss monitor to prevent further epochs if loss is no longer decreasing.
  - ◊ Validation split: 20%.
  - ◊ Batch size: 32.
  - ◊ Called with 20 epochs.
    - o Ran 3 epochs.

- → Model performance on test data:
  - ◊ Test accuracy 97.0%



```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      1300

           1       0.96      0.98      0.97      1300

    accuracy                           0.97      2600

   macro avg       0.97      0.97      0.97      2600

weighted avg       0.97      0.97      0.97      2600
```

→ 28 false negatives, 51 false positives.

◊ Fewer images classified as uninfected when actually parasitized.

→ Fair precision at 98% for uninfected & 96% for parasitized.

◊ Few false negatives, more false positives than previous models.

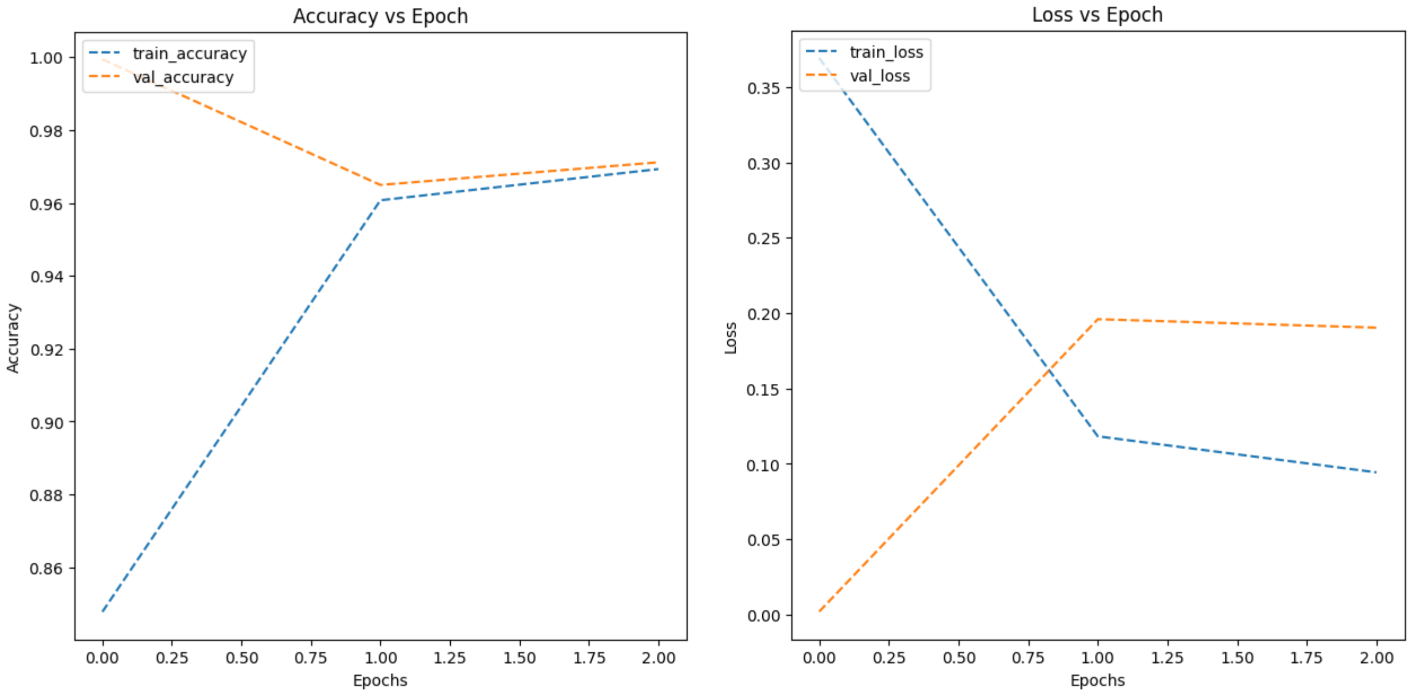
→ Poor recall at 96% for uninfected, fair recall at 98% for parasitized.

◊ More important metric to avoid lethality to patients.

◊ Better result for parasitized samples.

→ This model fewest false negatives so far.

◊ Better option for patient outcomes.



→ **Validation & training accuracy values are similar after 1 epoch.**

◊ **Good model fit very quickly.**

→ **Validation loss higher than training loss after 1 epoch.**

◊ **Some overfitting may be occurring.**

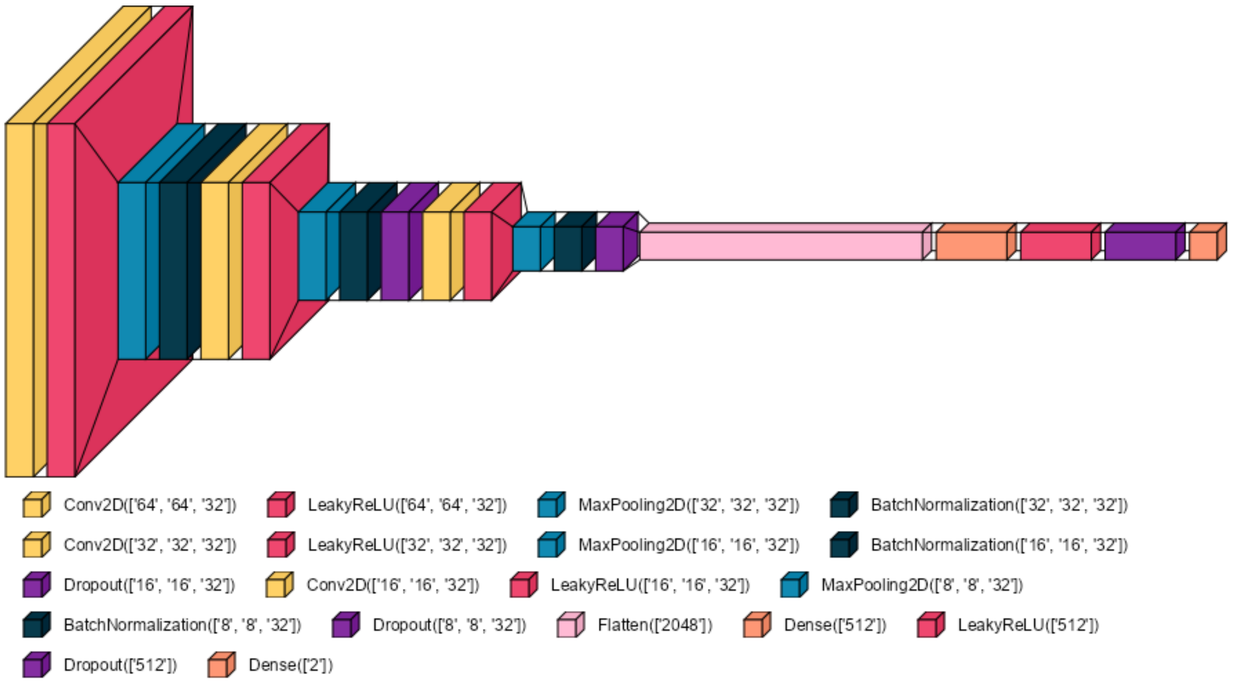
→ **Fewer epochs.**

◊ **Favorable for computation, but may be preventing loss from decreasing.**

→ **Model accuracy is highest for this model out of all performed.**

CNN MODEL 3
WITH DATA AUGMENTATION

→ **Data augmentation with Image Data Generator noted previously.**


Conv2D(['64', '64', '32'])
LeakyReLU(['64', '64', '32'])
MaxPooling2D(['32', '32', '32'])
BatchNormalization(['32', '32', '32'])
Conv2D(['32', '32', '32'])
LeakyReLU(['32', '32', '32'])
MaxPooling2D(['16', '16', '32'])
BatchNormalization(['16', '16', '32'])
Dropout(['16', '16', '32'])
Conv2D(['16', '16', '32'])
LeakyReLU(['16', '16', '32'])
MaxPooling2D(['8', '8', '32'])
BatchNormalization(['8', '8', '32'])
Dropout(['8', '8', '32'])
Flatten(['2048'])
Dense(['512'])
LeakyReLU(['512'])
Dropout(['512'])
Dense(['2'])

→ CNN model with:

- ◊ 3 convolutional layers.
  - o Output size same as input.
  - o Kernel size 3 x 3.
  - o Leaky ReLU activation.
- ◊ 3 max-pooling layers.
  - o Pool size 2 x 2.
- ◊ 3 Dropout layers
- ◊ 1 flatten layer.
- ◊ 2 dense layers.
  - o Leaky ReLU activation.
- ◊ 1 output layer.
  - o 2 nodes.
  - o Softmax activation.

→ Compiled with:

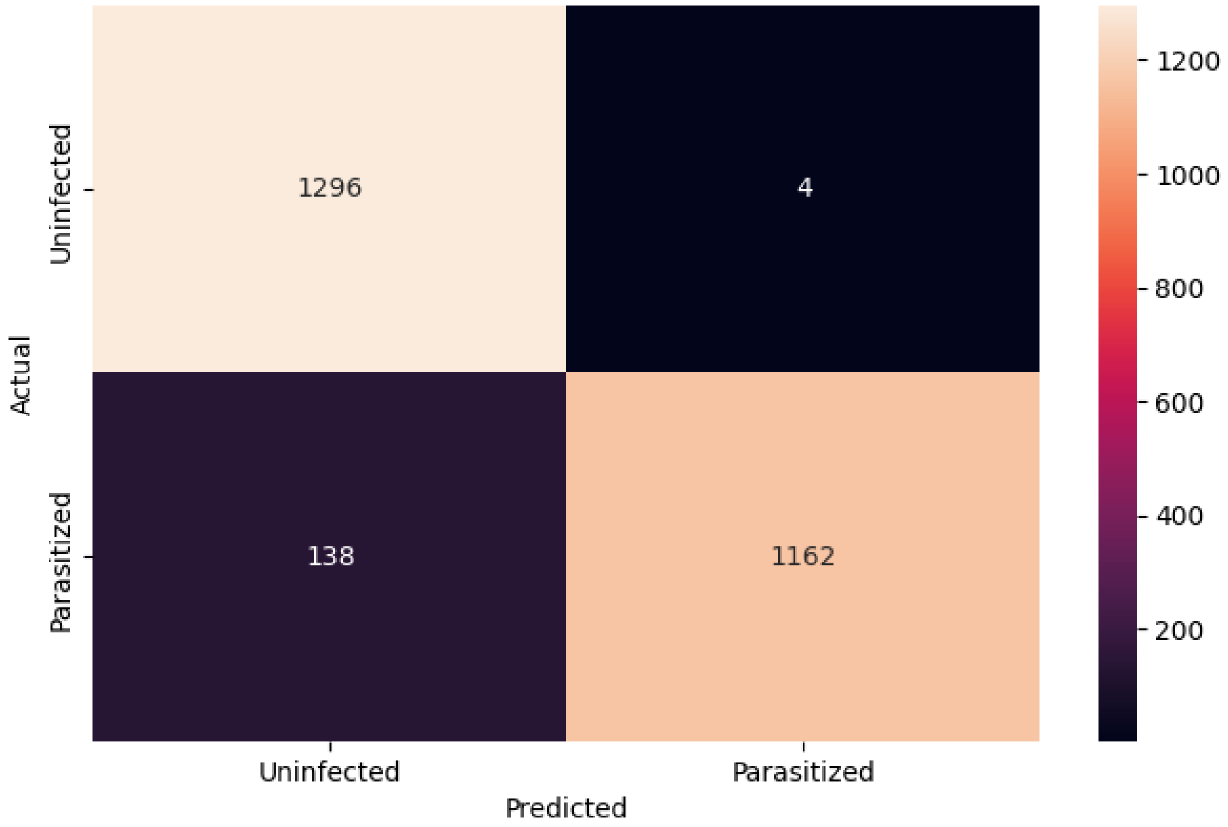
- ◊ Loss: categorical cross-entropy.
- ◊ Optimizer: Adam.
  - o Learning rate: 0.001.

→ Fitted with:

- ◊ Callback for early stopping with validation loss monitor to prevent further epochs if loss is no longer decreasing.
- ◊ Validation split: 20%.
- ◊ Batch size: 32.
- ◊ Called with 20 epochs.
  - o Ran 7 epochs.

→ Model performance on test data:

- ◊ Test accuracy 94.5%



```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      1300

           1       1.00      0.89      0.94      1300

    accuracy                           0.95      2600

   macro avg       0.95      0.95      0.95      2600

weighted avg       0.95      0.95      0.95      2600
```

→ 138 false negatives, 4 false positives.

◊ More images classified as uninfected when actually parasitized.

◊ Seek to minimize this issue – false negatives are weighed heavily when the patient outcome can be lethal.

→ Very poor precision at 90% for uninfected, great at 100% for parasitized.

◊ Very few false positives for uninfected.

→ Very good recall at 100% for uninfected, very poor recall at 89% for parasitized.

◊ More important metric to avoid lethality to patients.

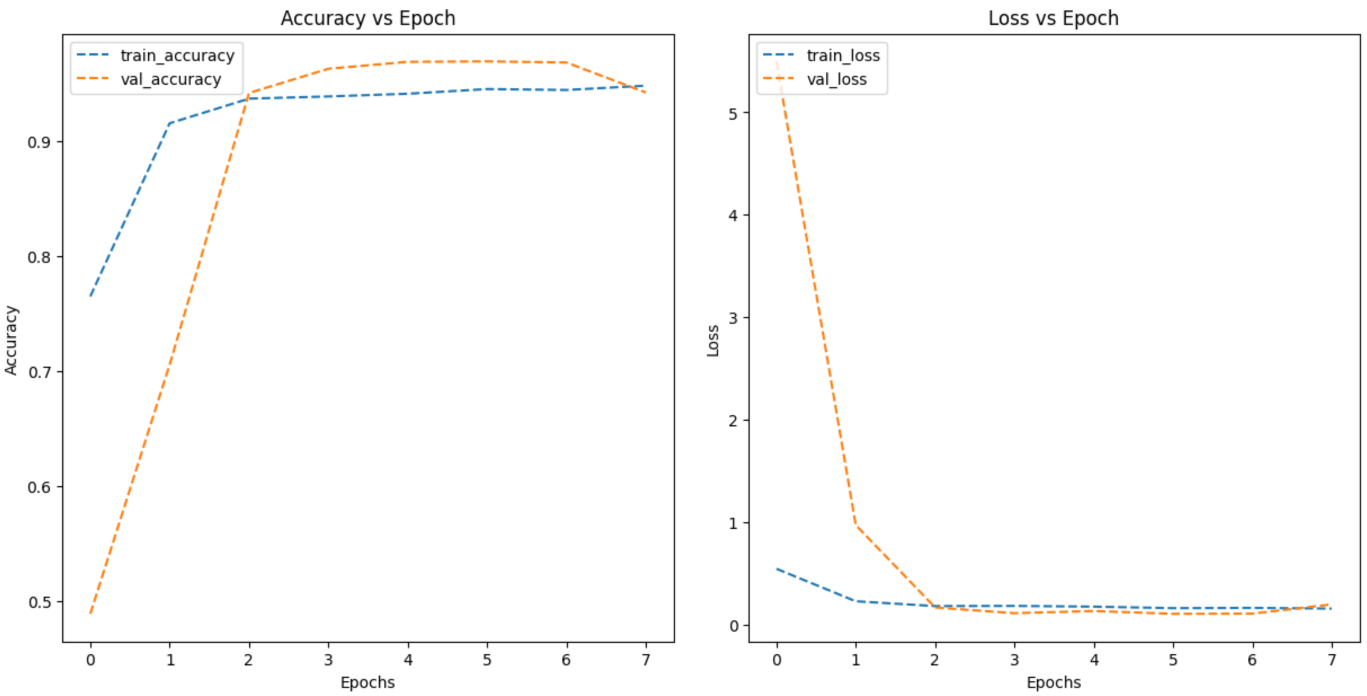
◊ Poor result for parasitized samples.

→ Poor F1 scores – precision & recall are not well-balanced.

→ Too many false negatives to use this model.

→ CNN Model 3 fewer false negatives.

◊ Better option for patient outcomes.



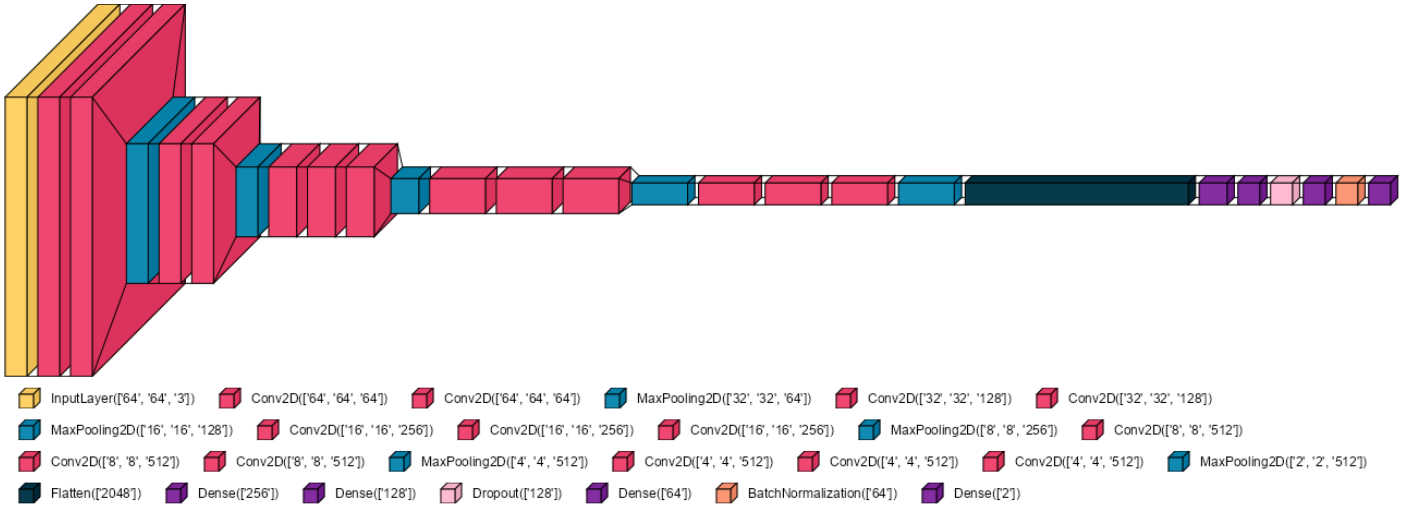
→ **Training & validation accuracies are similar after 2 epochs, diverge slightly, then reconverge after 7 epochs.**

  ◊ **Little overfitting.**

→ **Training & validation loss converge quickly by epoch 2.**

→ **The fit is good, but accuracy is not very high.**

PRE-TRAINED MODEL
VGG16


InputLayer(['64', '64', '3'])
Conv2D(['64', '64', '64'])
Conv2D(['64', '64', '64'])
MaxPooling2D(['32', '32', '64'])
Conv2D(['32', '32', '128'])
Conv2D(['32', '32', '128'])
MaxPooling2D(['16', '16', '128'])
Conv2D(['16', '16', '256'])
Conv2D(['16', '16', '256'])
Conv2D(['16', '16', '256'])
MaxPooling2D(['8', '8', '256'])
Conv2D(['8', '8', '512'])
Conv2D(['8', '8', '512'])
Conv2D(['8', '8', '512'])
MaxPooling2D(['4', '4', '512'])
Conv2D(['4', '4', '512'])
Conv2D(['4', '4', '512'])
Conv2D(['4', '4', '512'])
MaxPooling2D(['2', '2', '512'])
Flatten(['2048'])
Dense(['256'])
Dense(['128'])
Dropout(['128'])
Dense(['64'])
BatchNormalization(['64'])
Dense(['2'])

→ CNN model with:

→ Pre-defined layers:

  - ◊ 13 convolutional layers.
  - ◊ 5 max-pooling layers.

→ Added classification layers:

  - ◊ 1 Dropout layer.
  - ◊ 1 flatten layer.
  - ◊ 2 dense layers.
    - ○ ReLU activation.
  - ◊ 1 normalization layer.
  - ◊ 1 output layer.
    - ○ 2 nodes.
    - ○ Softmax activation.

→ Compiled with:

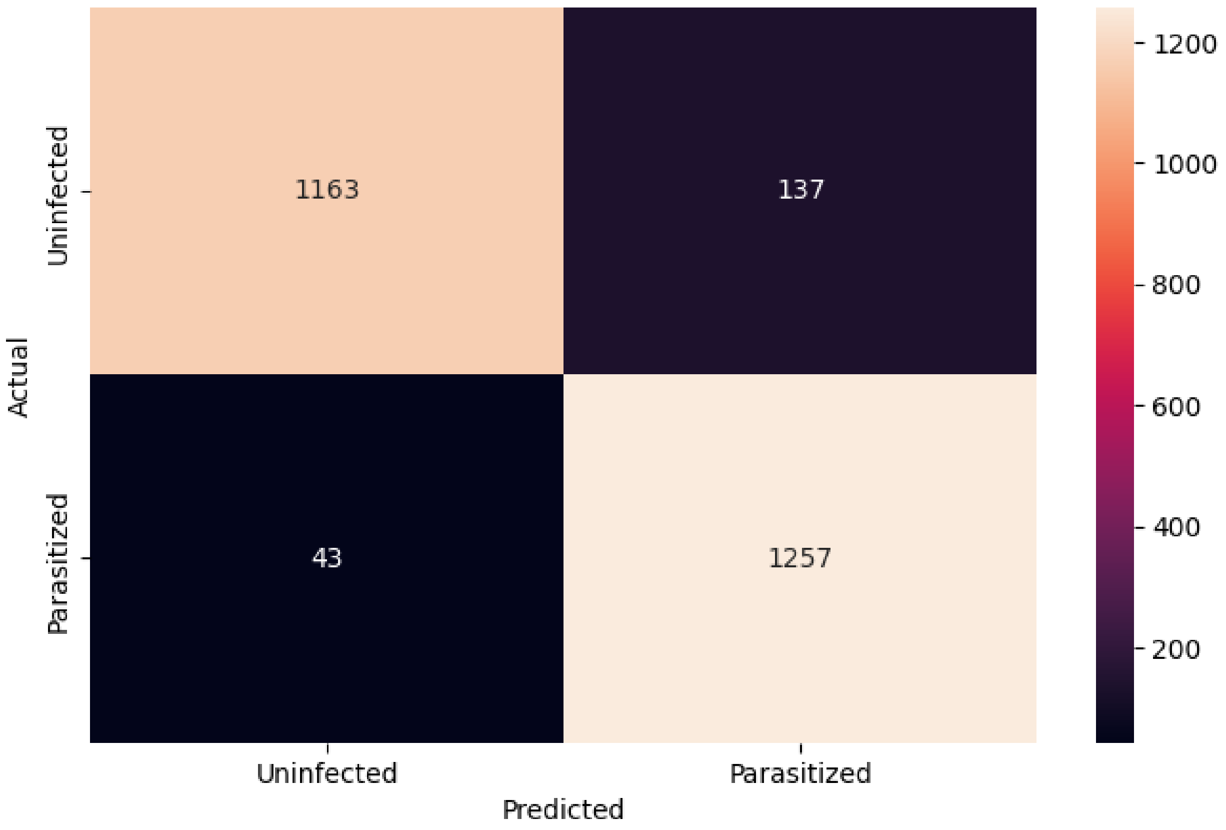
  - ◊ Loss: categorical cross-entropy.
  - ◊ Optimizer: Adam.
    - ○ Learning rate: 0.001.

→ Fitted with:

  - ◊ Callback for early stopping with validation loss monitor to prevent further epochs if loss is no longer decreasing.
  - ◊ Validation split: 20%.
  - ◊ Batch size: 32.
  - ◊ Called with 10 epochs.
    - ○ Ran 5 epochs.

→ Model performance on test data:

  - ◊ Test accuracy 93.1%



```
                precision   recall  f1-score   support

           0       0.96      0.89      0.93      1300

           1       0.90      0.97      0.93      1300

    accuracy                           0.93      2600

   macro avg       0.93      0.93      0.93      2600

weighted avg       0.93      0.93      0.93      2600
```

→ 43 false negatives, 137 false positives.

- ◊ More images classified as parasitized when actually uninfected.

→ Poor precision at 96% for uninfected, very poor at 90% for parasitized.

- ◊ Many false positives for uninfected.

→ Very poor recall at 89% for uninfected, fair recall at 97% for parasitized.

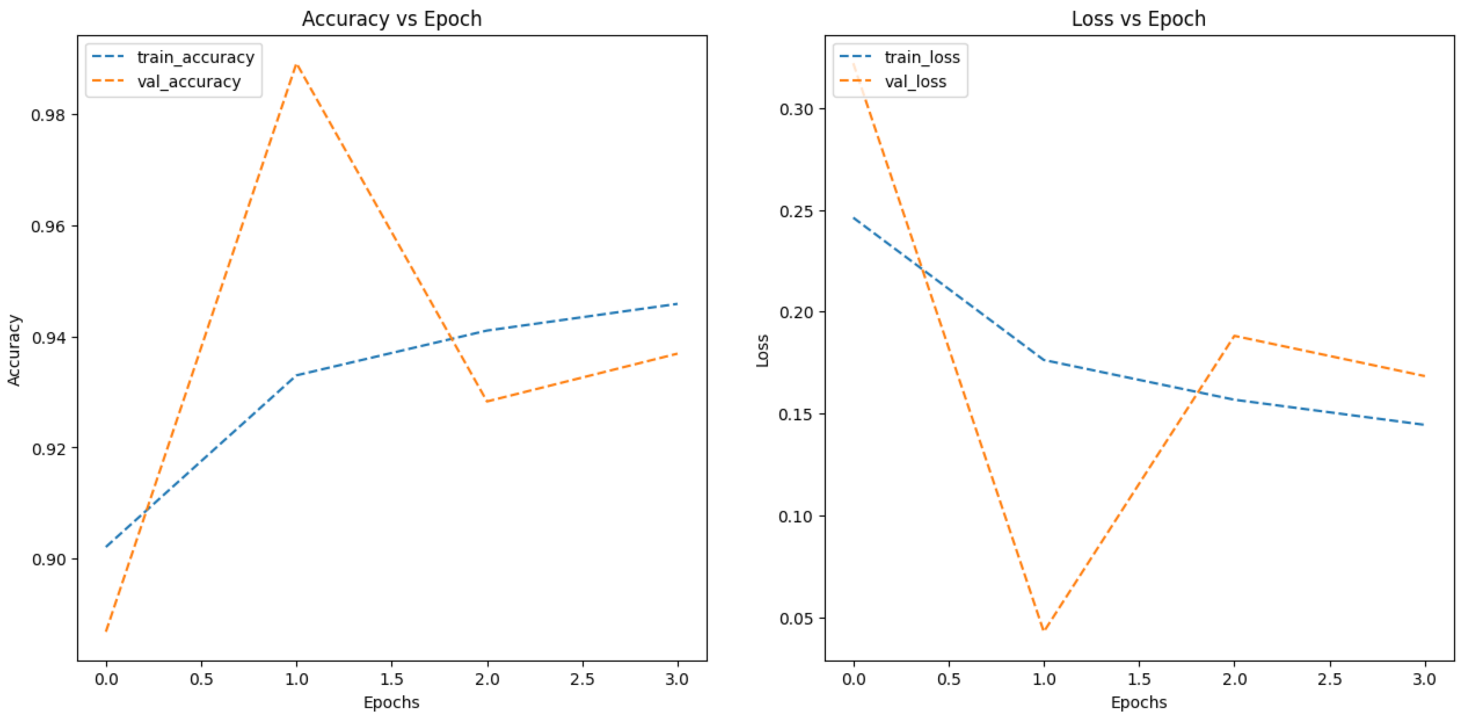
- ◊ More important metric to avoid lethality to patients.

→ Poor F1 scores – precision & recall are not well-balanced.

→ Fewer false negatives in this model compared to others.

→ CNN Model 3 fewer false negatives.

- ◊ Better option for patient outcomes.



→ Training & validation accuracies diverge greatly at epoch 1, become more similar at epoch 2 but training remains higher.

◊ Indicates some overfitting.

→ Training & validation loss diverge greatly at epoch 1, become more similar at epoch 2 but validation remains higher.

◊ Indicates overfitting.

→ Overall, one of the poorer fitting models in comparison to the others.

FINAL SOLUTION DESIGN PROPOSAL

→ CNN Model 3 with no data augmentation displayed the best performance for key metrics.

- ◊ Fewest false negatives.
  - ■ Extremely important to avoid lethal outcome for patients due to misclassification.
- ◊ Few false positives, but more than false negatives.
  - ■ Ideal to minimize, but not at the expense of minimizing false negatives.
- ◊ Great model accuracy at 97%
  - ■ Overall predictive power is good.
- ◊ Validation loss indicated some potential overfitting, but accuracy convergence between training & validation sets was contradictory.

RECOMMENDATIONS FOR IMPLEMENTATION

→ Recommend using CNN Model 3 as a tool for Malaria diagnosis.

→ CNN Model 3 displayed the best balance of fit and accuracy.

- ◊ CNN Model 1 had a higher overall accuracy at 98%.
    - ■ Underfitting in training accuracy for this model indicates it may not be able to extrapolate to new images well.
    - ■ Fit is important for resiliency in the model to be able to handle images that vary in quality – an issue likely to be seen in areas with limited access to diagnostic resources.
- ◊ Accuracy of 97% was still one of the highest of all models.

→ CNN Model 3 displayed the lowest proportion of false negatives

- ◊ One of the key metrics being used to evaluate these models.
    - ■ Must avoid diagnosing patients as uninfected if they are parasitized to ensure they are treated for the parasite.