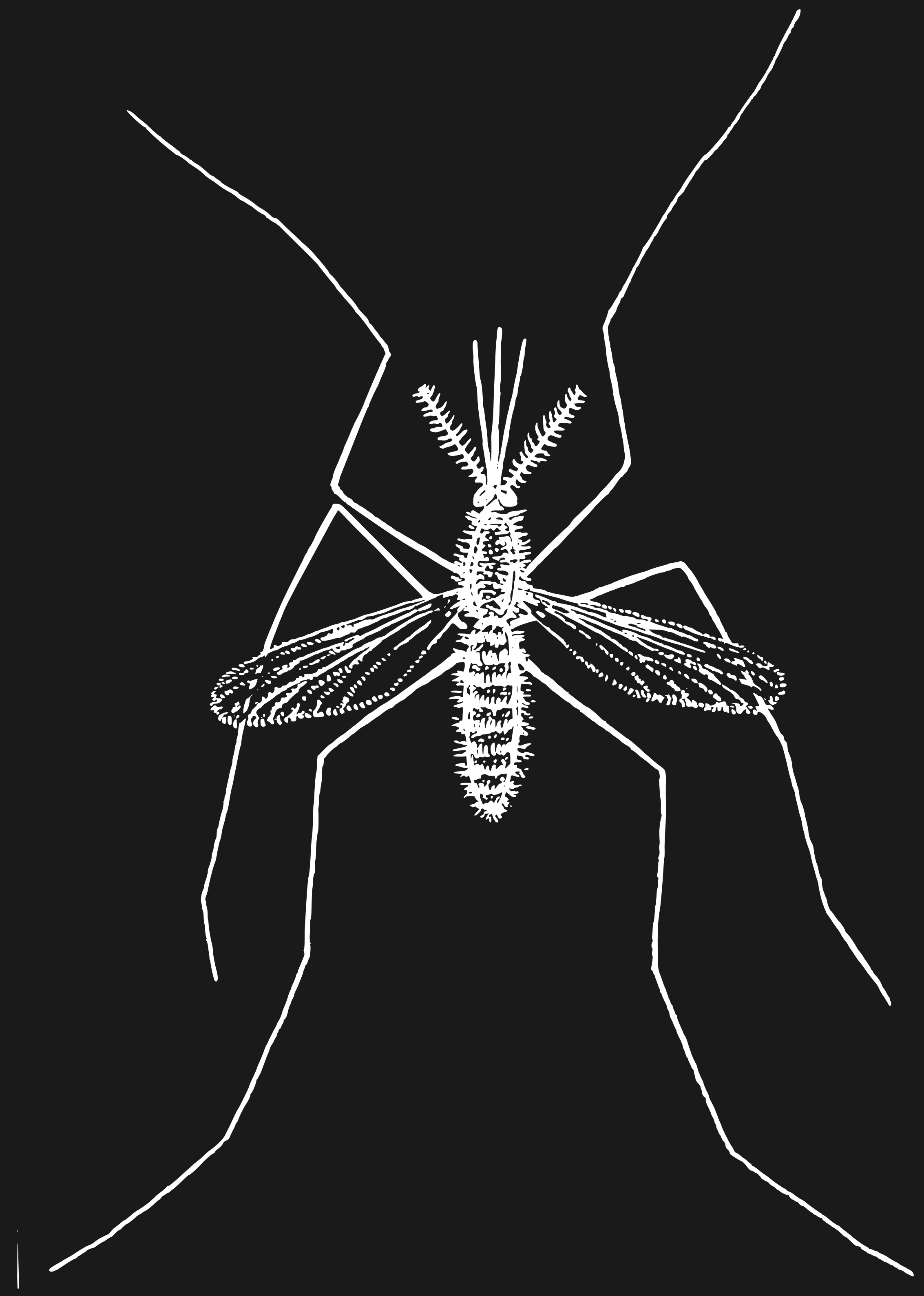
Malaria Blood Smear Classifier

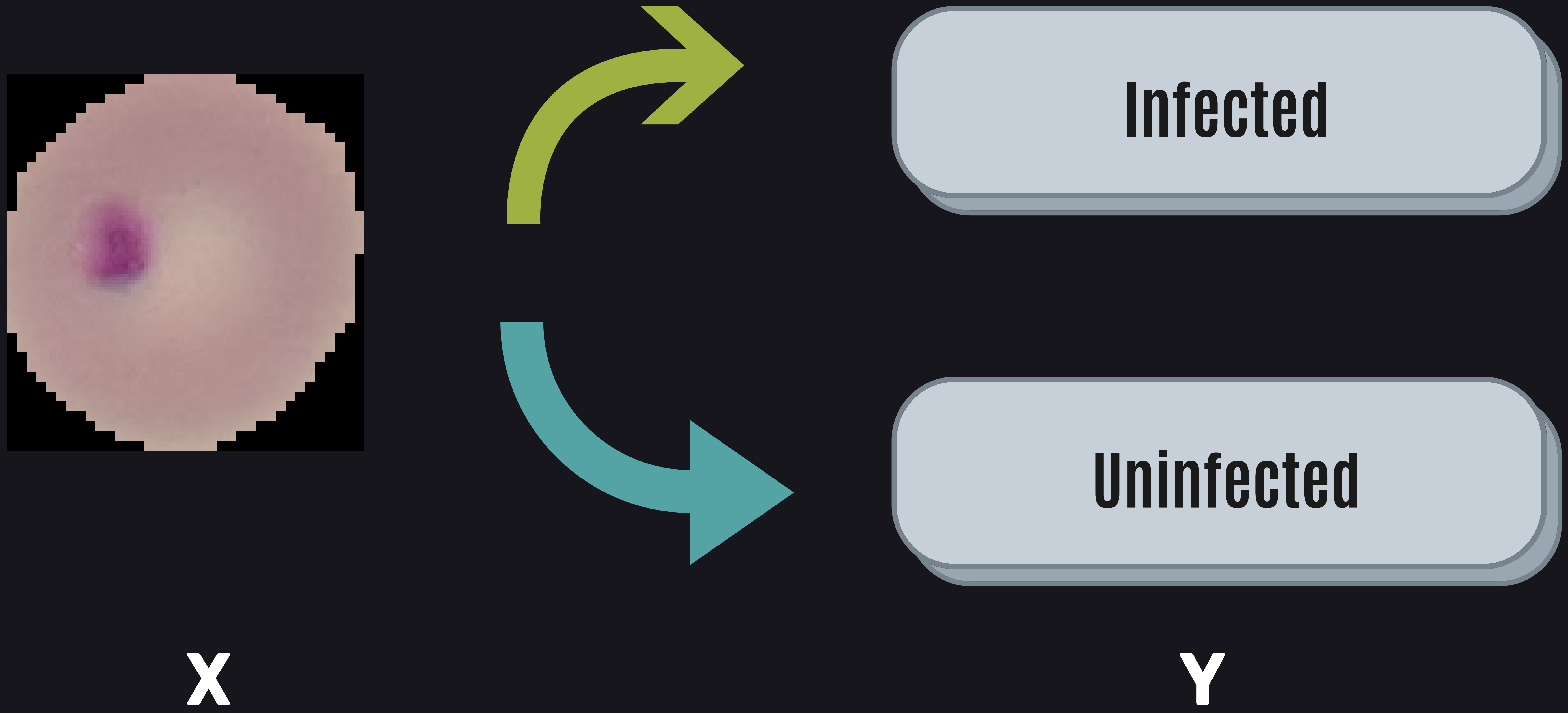
Content

Malaria

Malaria is a mosquito-borne infectious disease that affects humans and other animals. Malaria causes symptoms that typically include fever, tiredness, vomiting, and headaches.



Project Purpose:
Detect if a cell is infected with Malaria



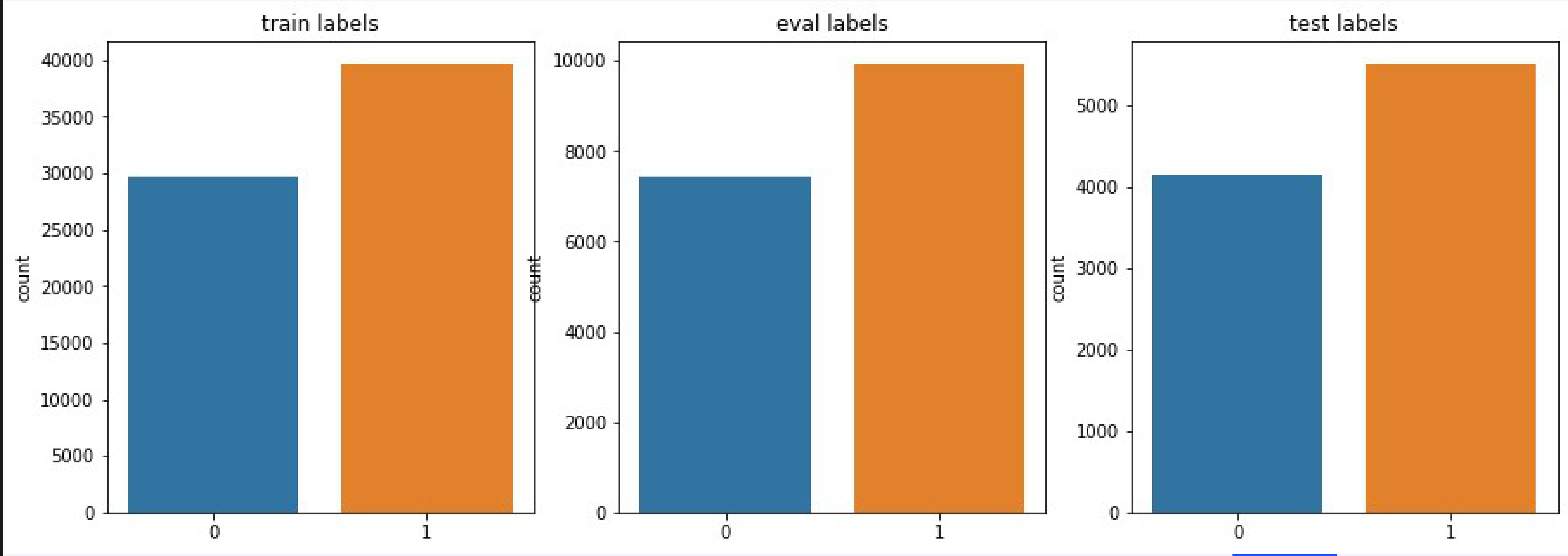
Dataset

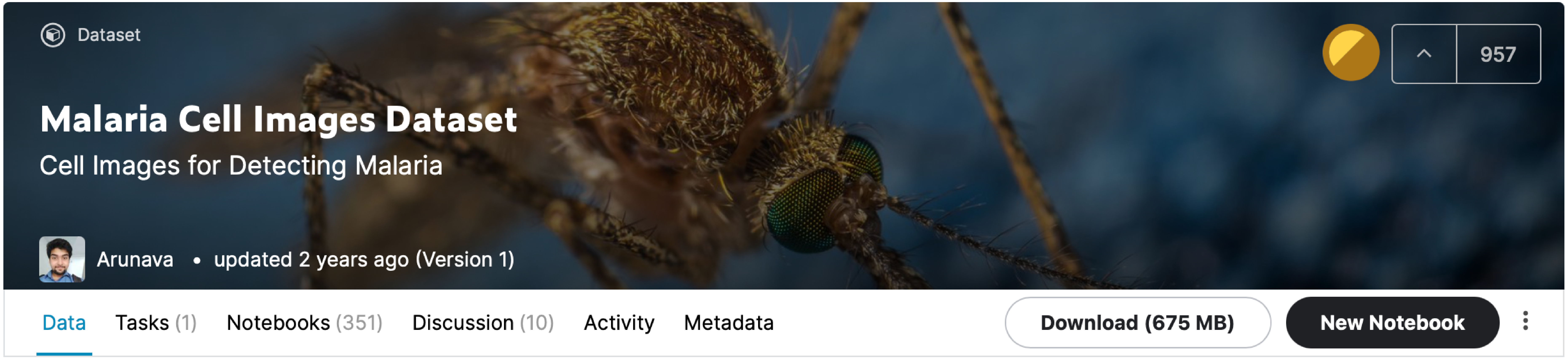
Malaria Cell Images

Include infected and uninfected pictures with Cells and Label folders

Description

96453 piece of data
train data shape (77162, 50, 50, 3)
eval data shape (9645, 50, 50, 3)
test data shape (9646, 50, 50, 3)





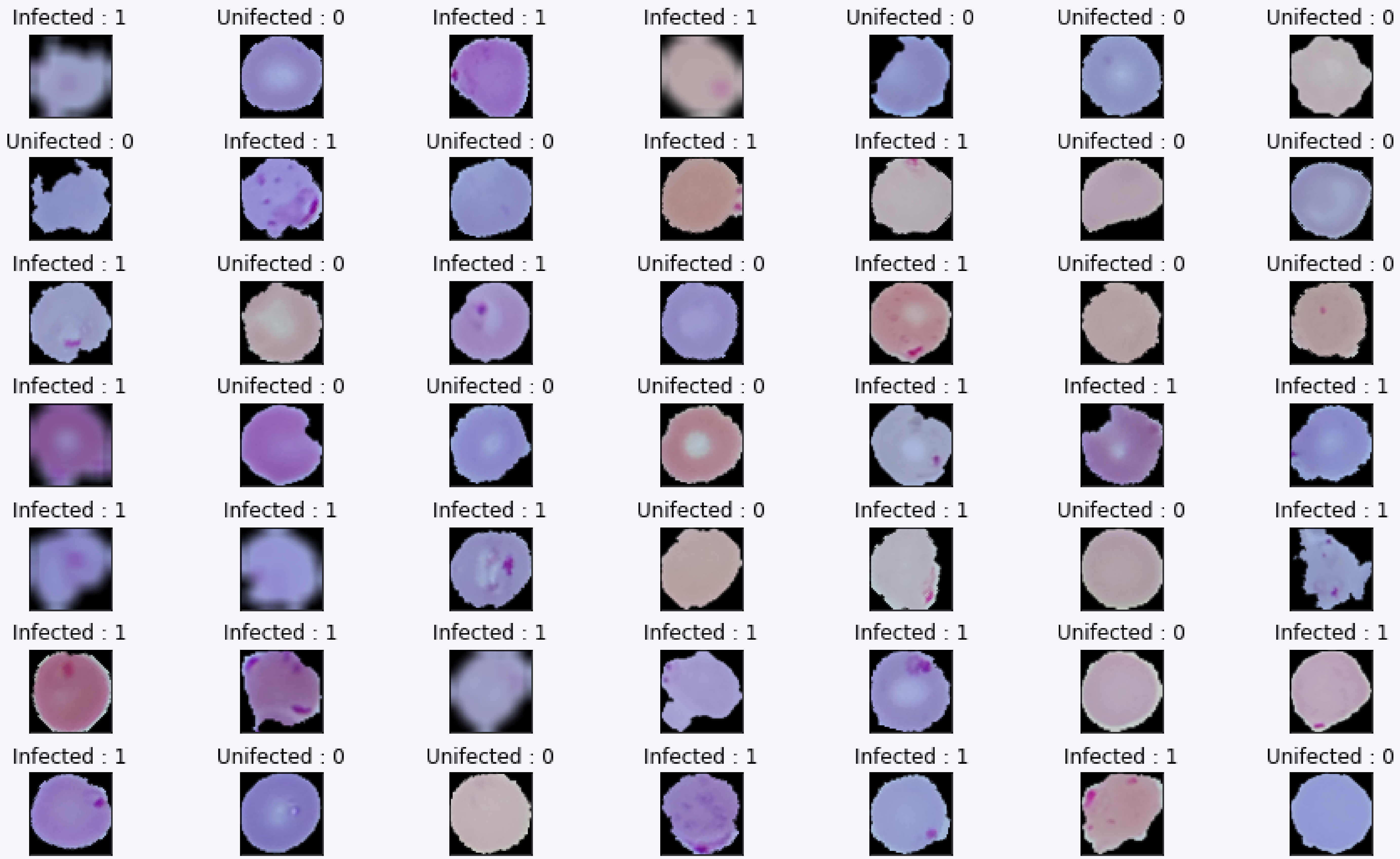
Dataset

Malaria Cell Images Dataset

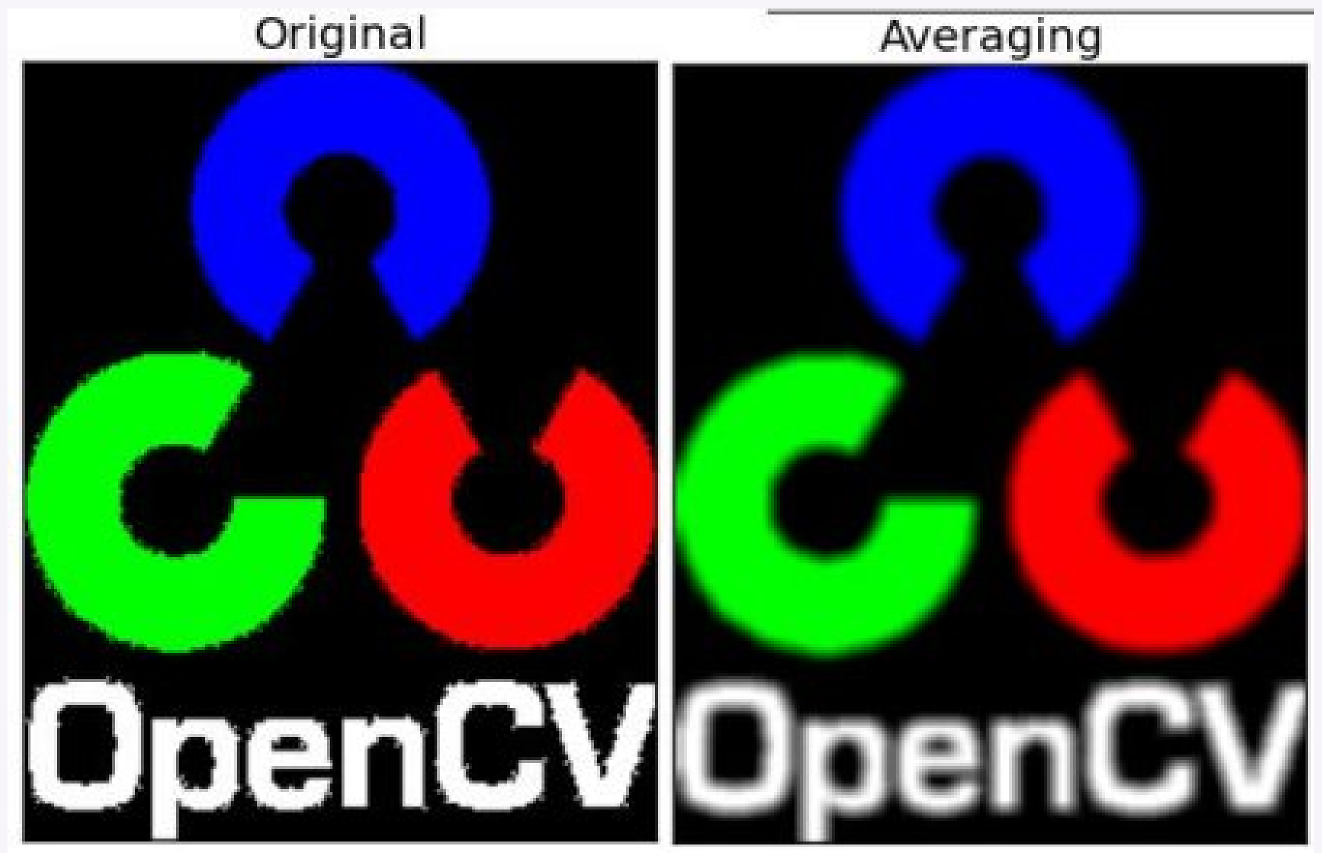
Cell Images for Detecting Malaria

Arunava • updated 2 years ago (Version 1)

Data Preview



How we deal with pictures...



- **cv2.blur(影像平滑模糊化)**

  平均濾波 Averaging:
  簡單地計算內核區域下所有像素的平均值，並用該平均值取代中心元素

- **Resize**

  將每張照片尺寸大小調為相同
  width = 50
  height = 50

Environment & Tools used

Google Colab

Python development environment that runs in the browser using Google Cloud

GPU

Tensorflow with GPU

Module

tensorflow
matplotlib
sklearn
seaborn

First , find best parameters...

1. **filter**

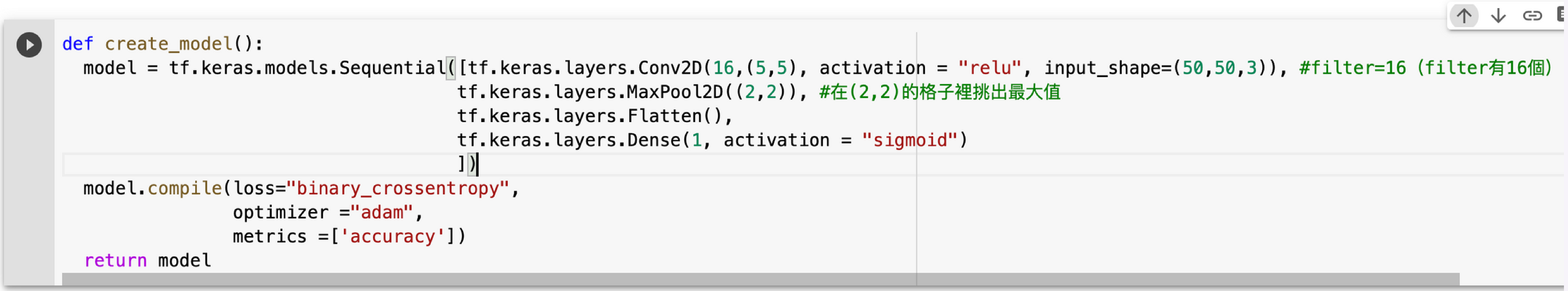
   Tentative 5/16/28

2. **kernal size**

   Tentative (2, 2), (3, 3), (4, 4), (5, 5)

3. **batch size**

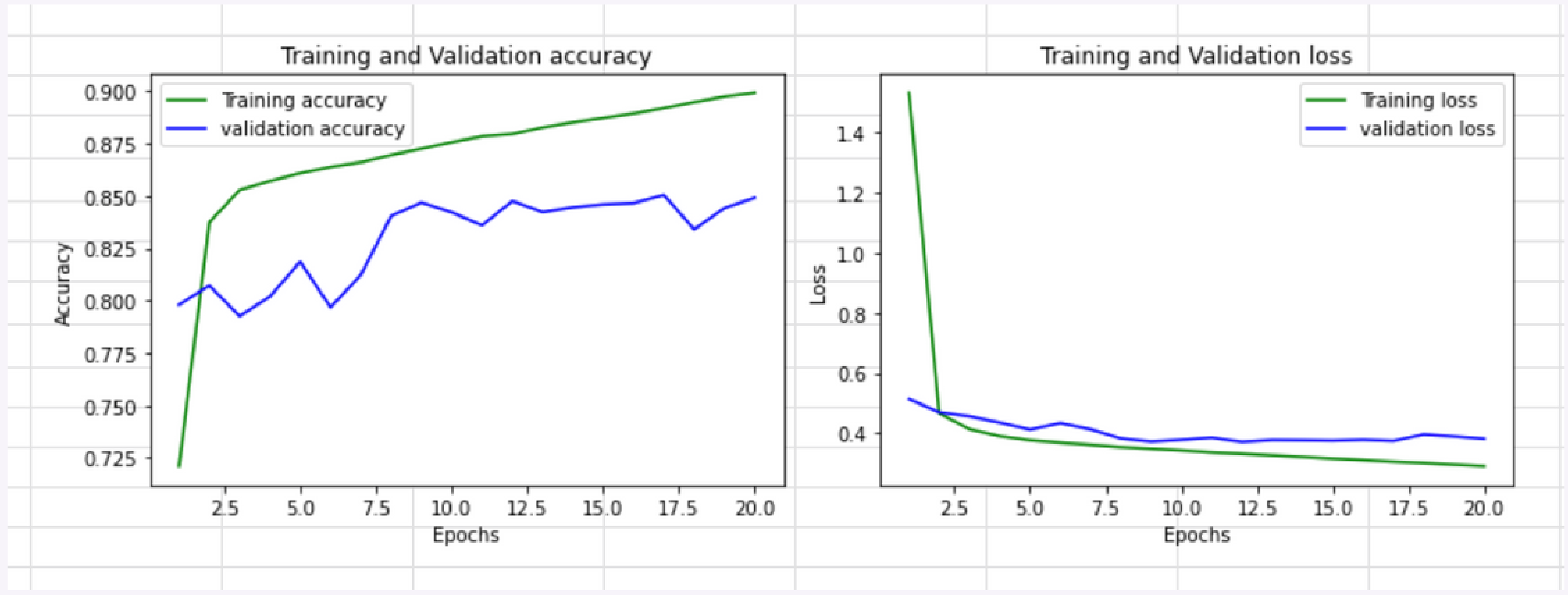
   Tentative 32/64



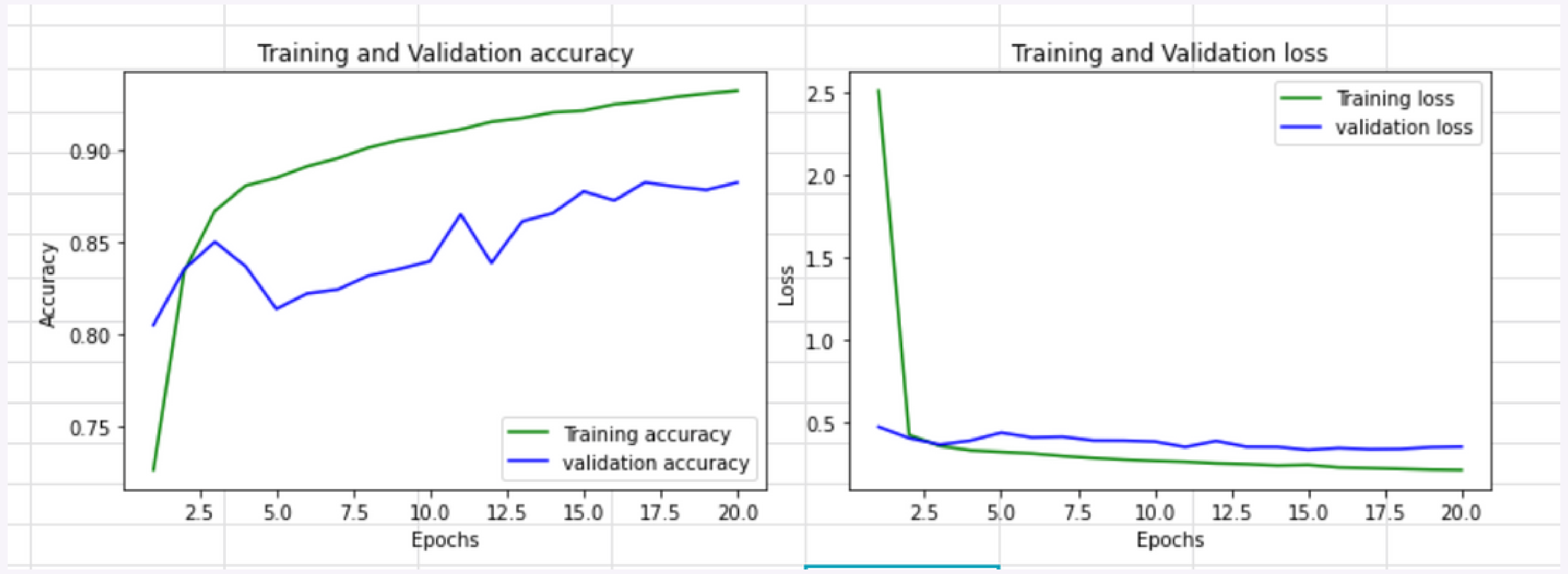
```python
def create_model():
  model = tf.keras.models.Sequential([tf.keras.layers.Conv2D(16,(5,5), activation = "relu", input_shape=(50,50,3)), #filter=16 (filter有16個)
                                      tf.keras.layers.MaxPool2D((2,2)), #在(2,2)的格子裡挑出最大值
                                      tf.keras.layers.Flatten(),
                                      tf.keras.layers.Dense(1, activation = "sigmoid")
                                      ])
  model.compile(loss="binary_crossentropy",
                optimizer ="adam",
                metrics =['accuracy'])
  return model
```

filter : loss is stable

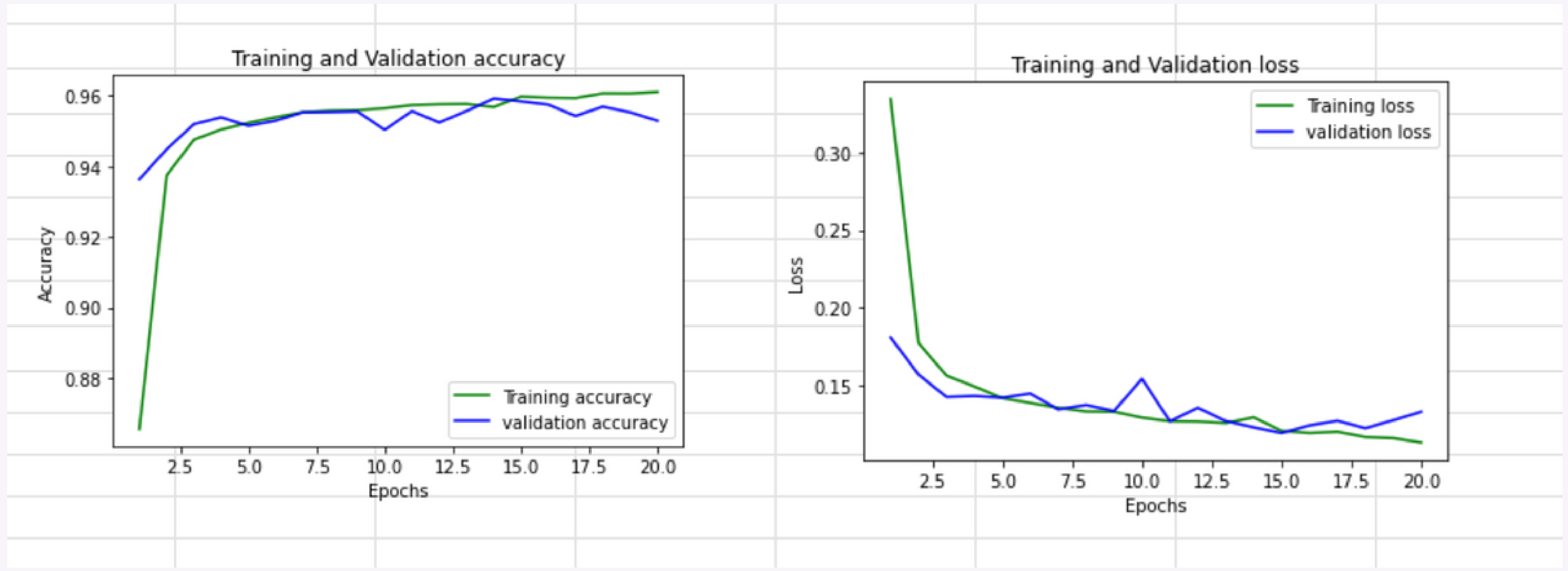
5



16

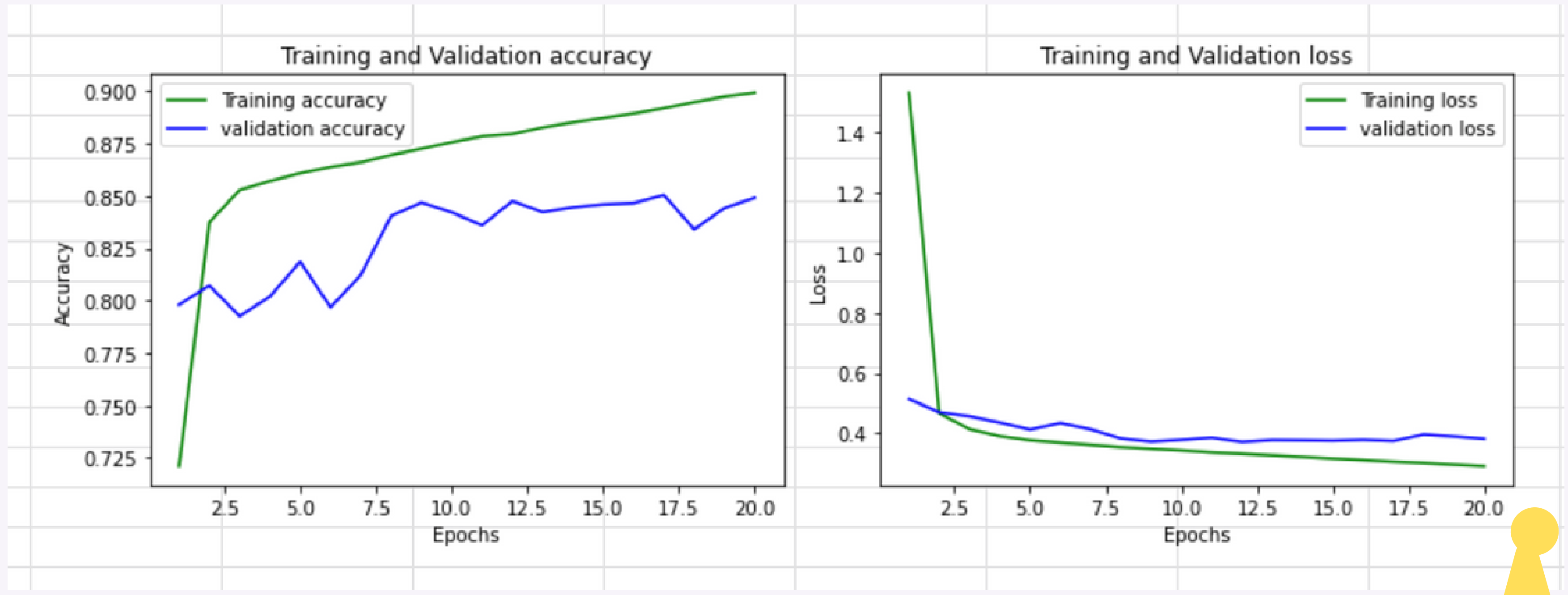


28

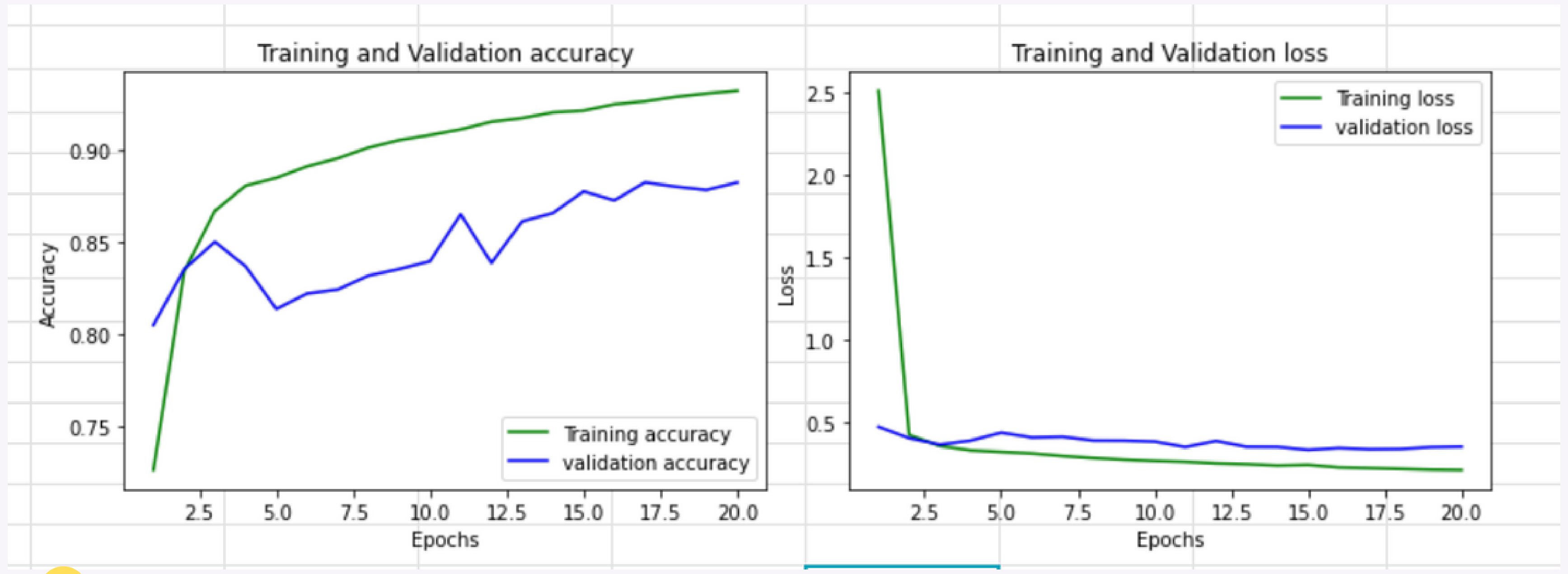


filter : the more filter, the better accuracy

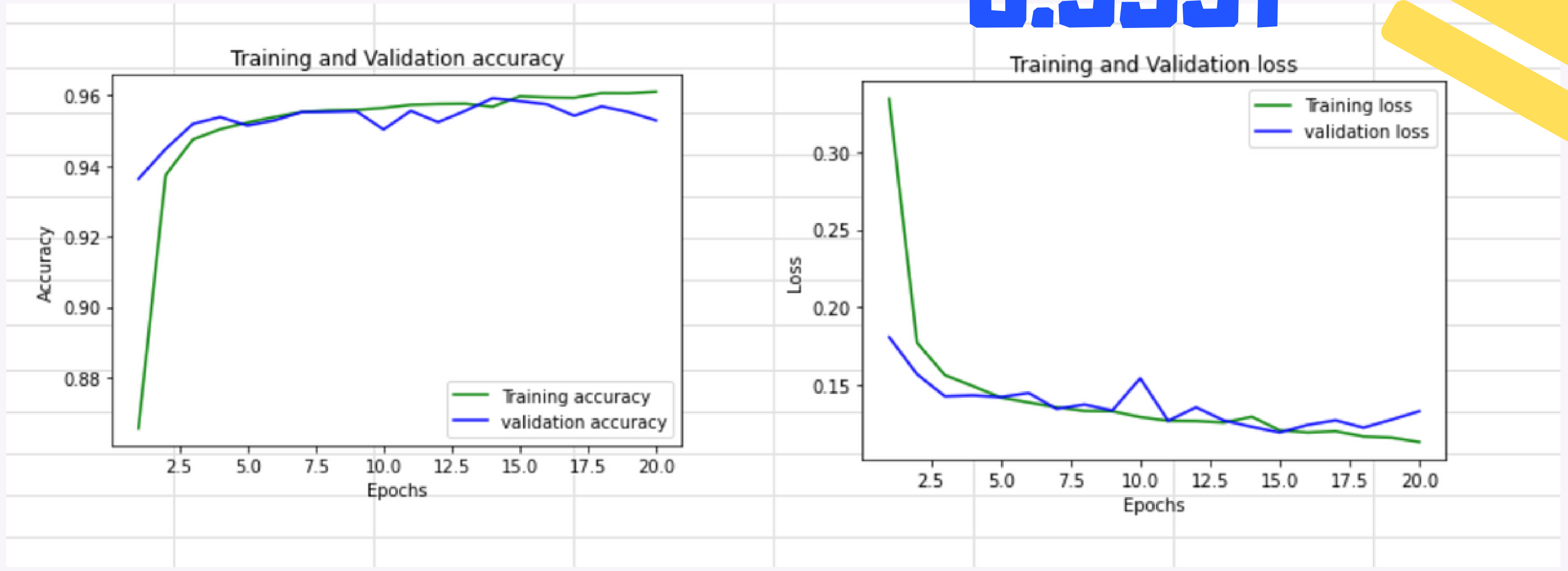
5 **0.8515**



16 **0.8851**

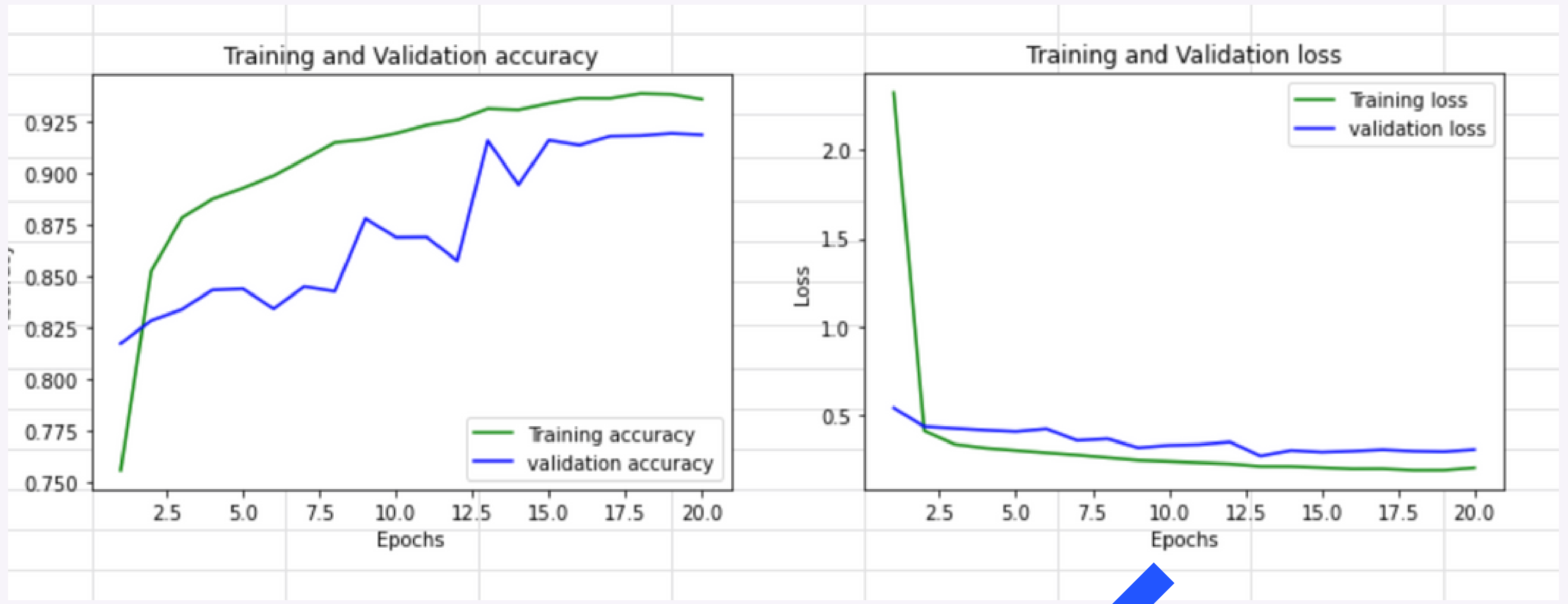


28 **0.9591**

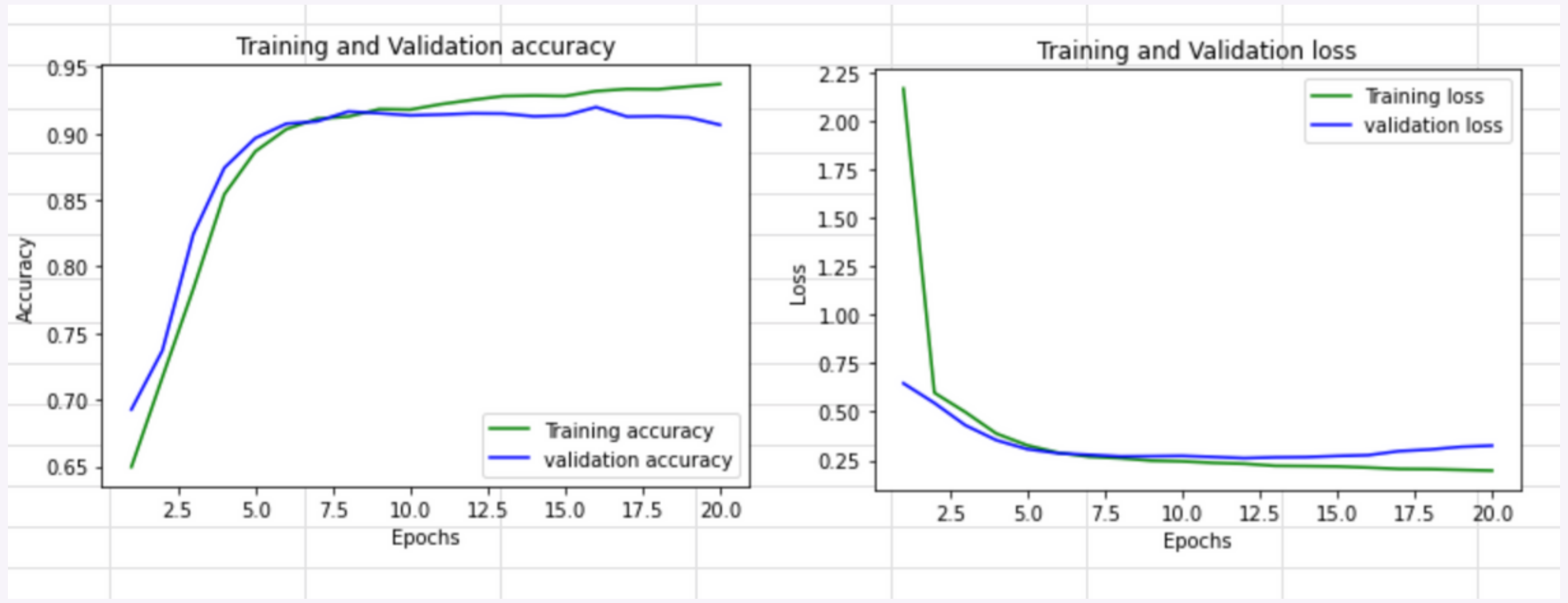


kernal size : (3, 3) (4, 4) better !

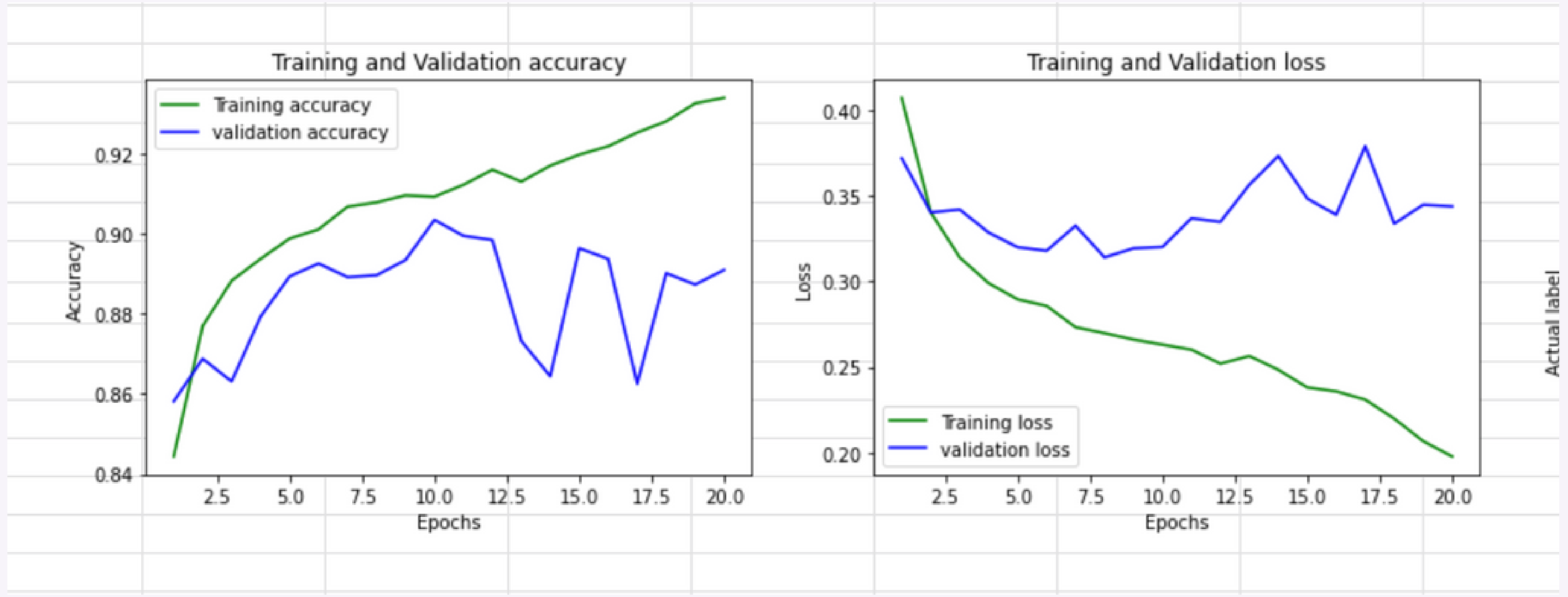
(2, 2)



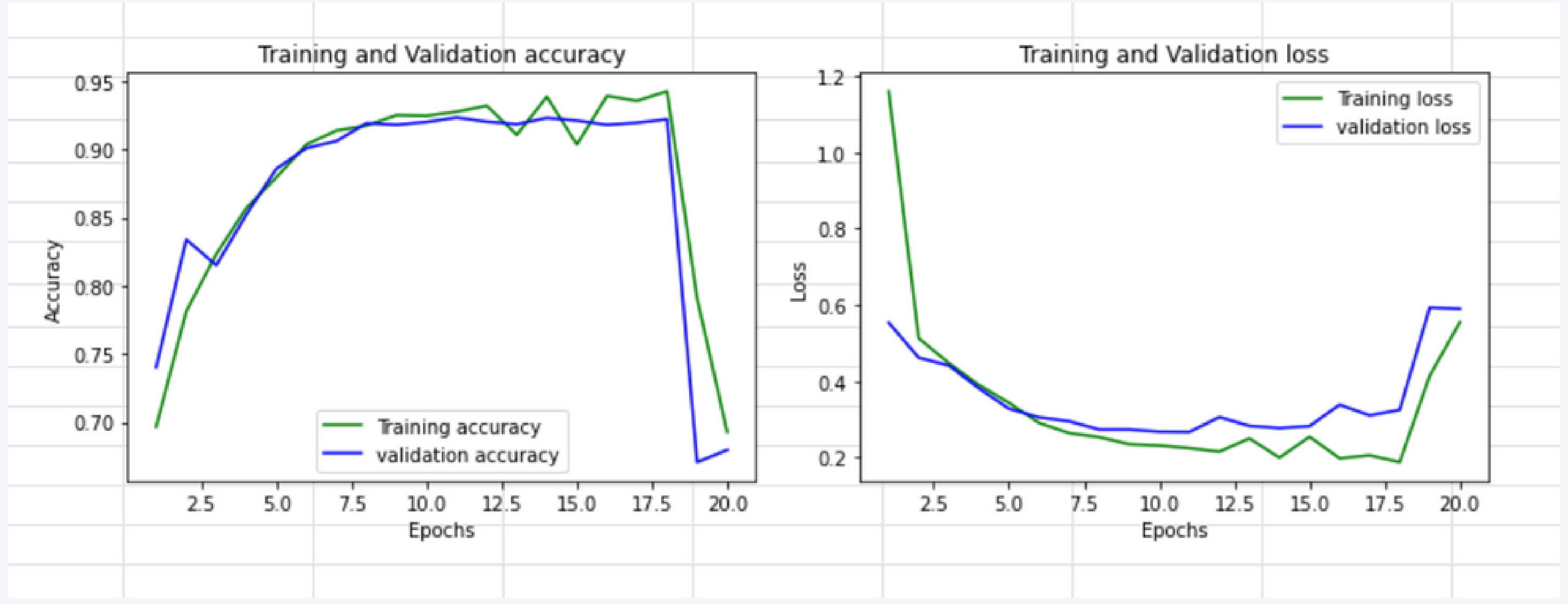
(3, 3) ✓



(4, 4) ✓

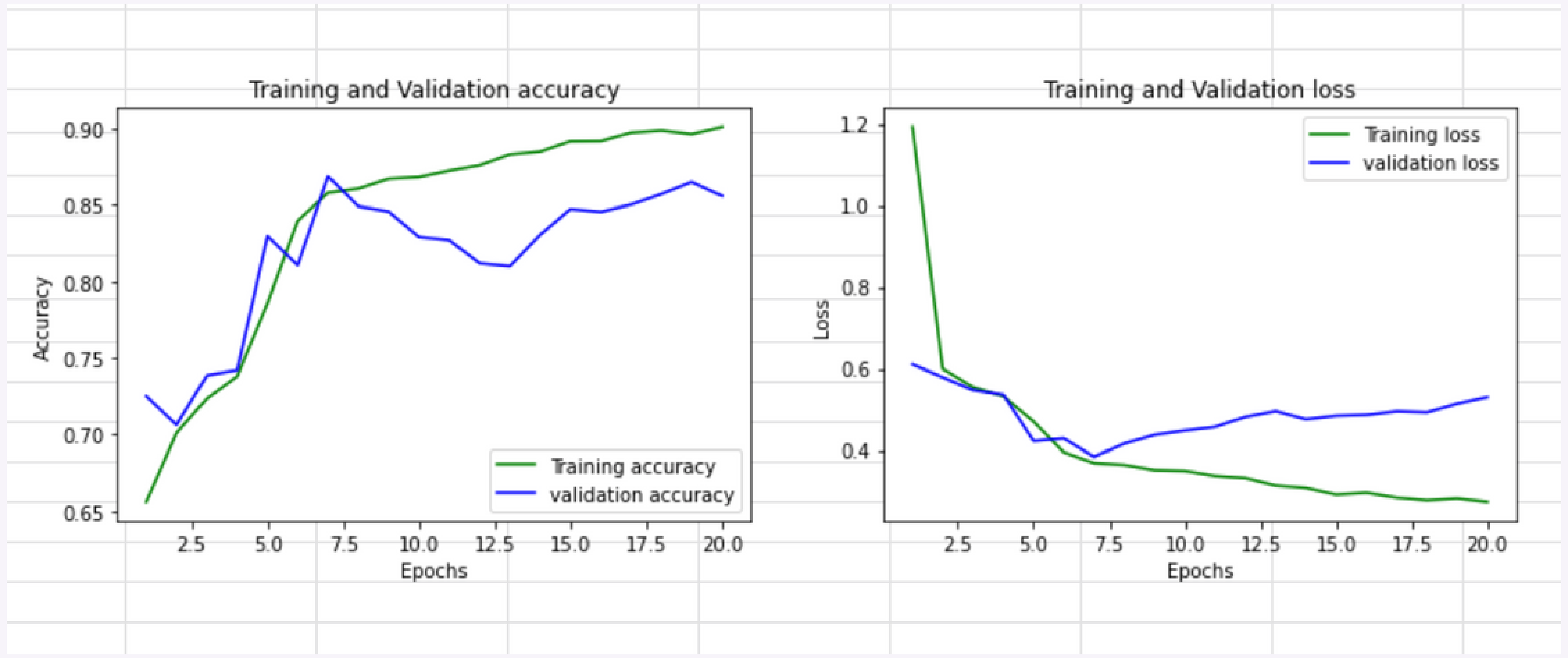


(5, 5)

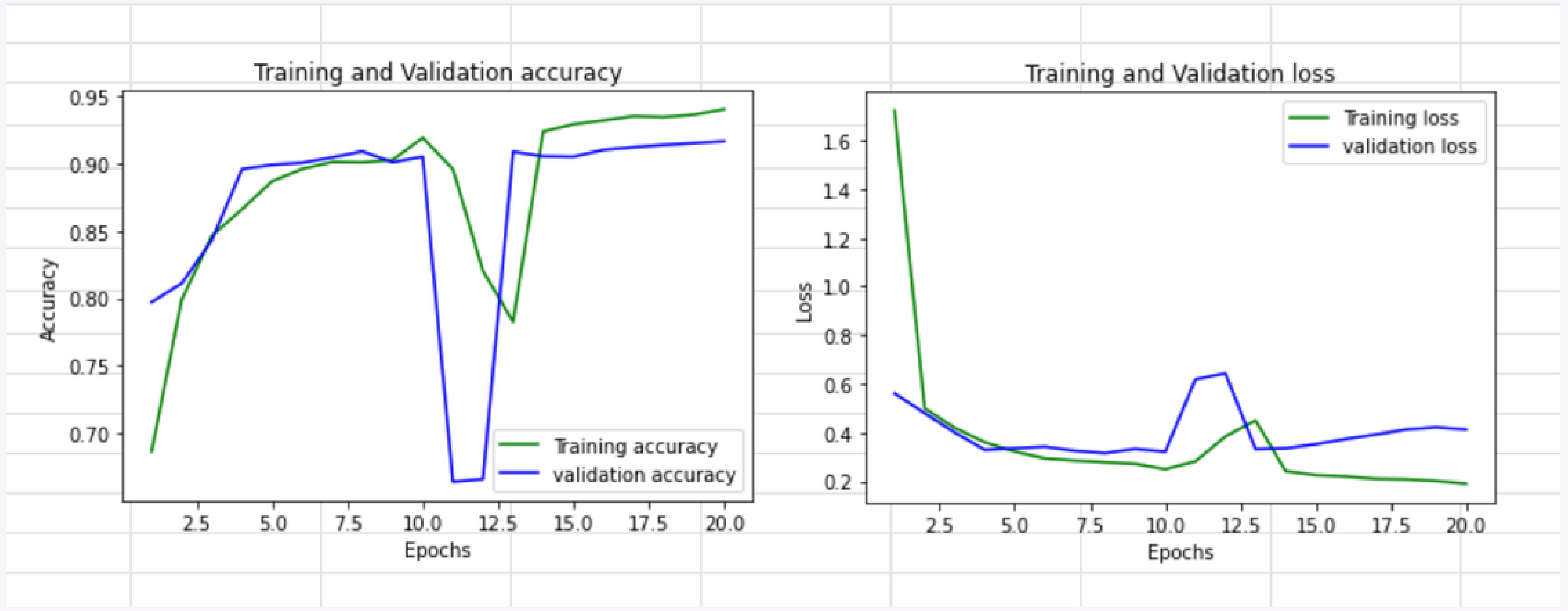


batch size : 32 weird...

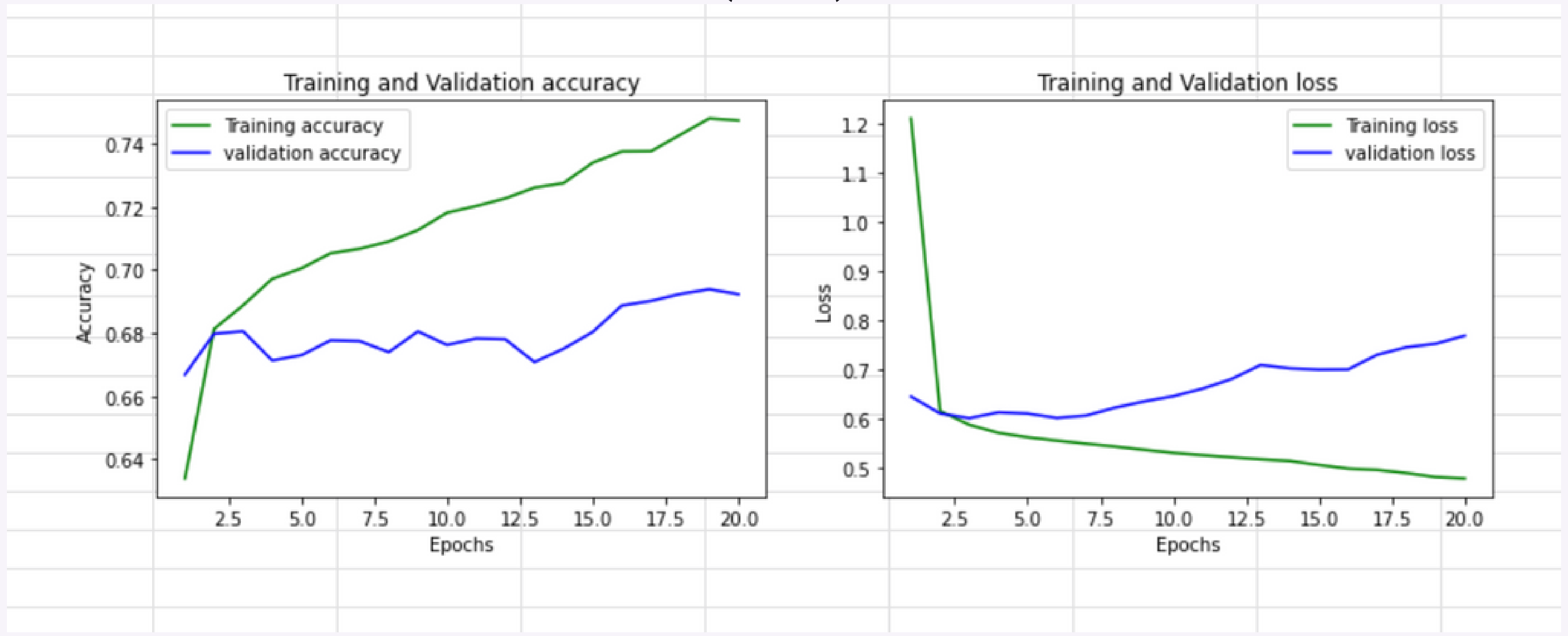
(3, 3)



(4, 4)

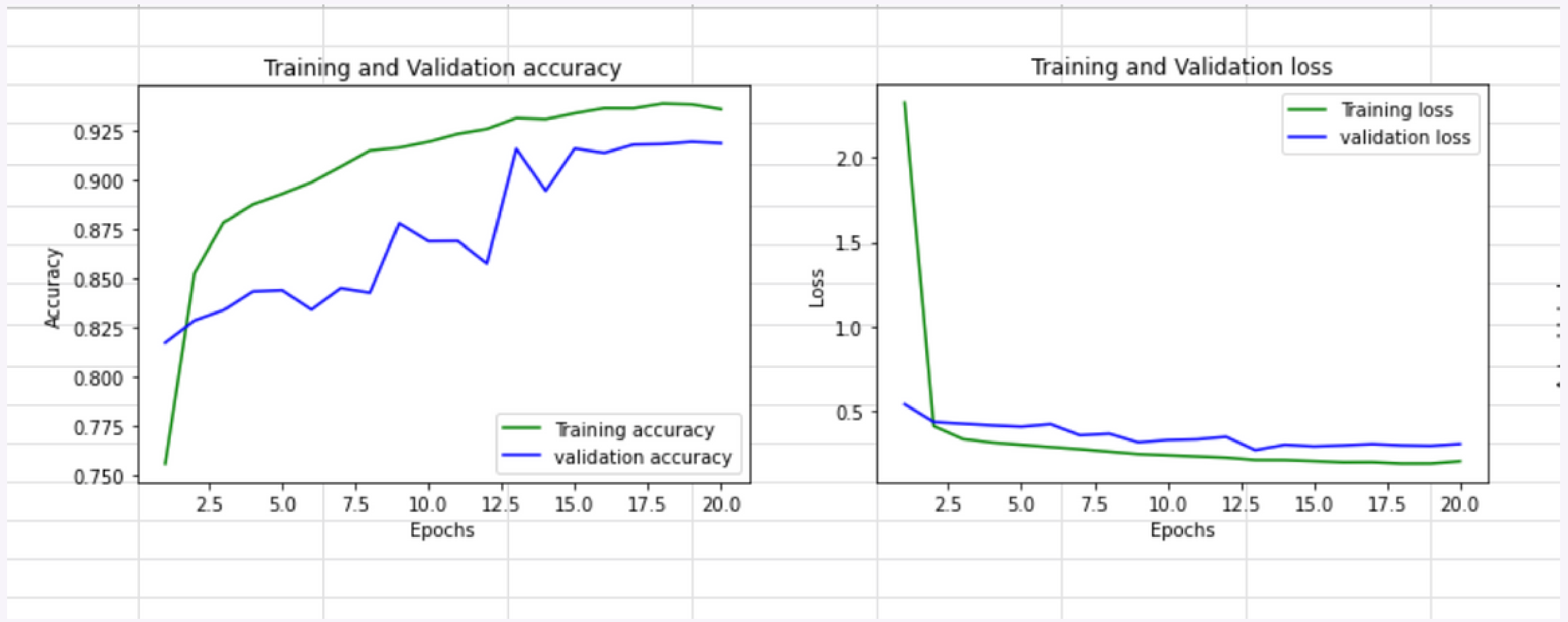


(5, 5)

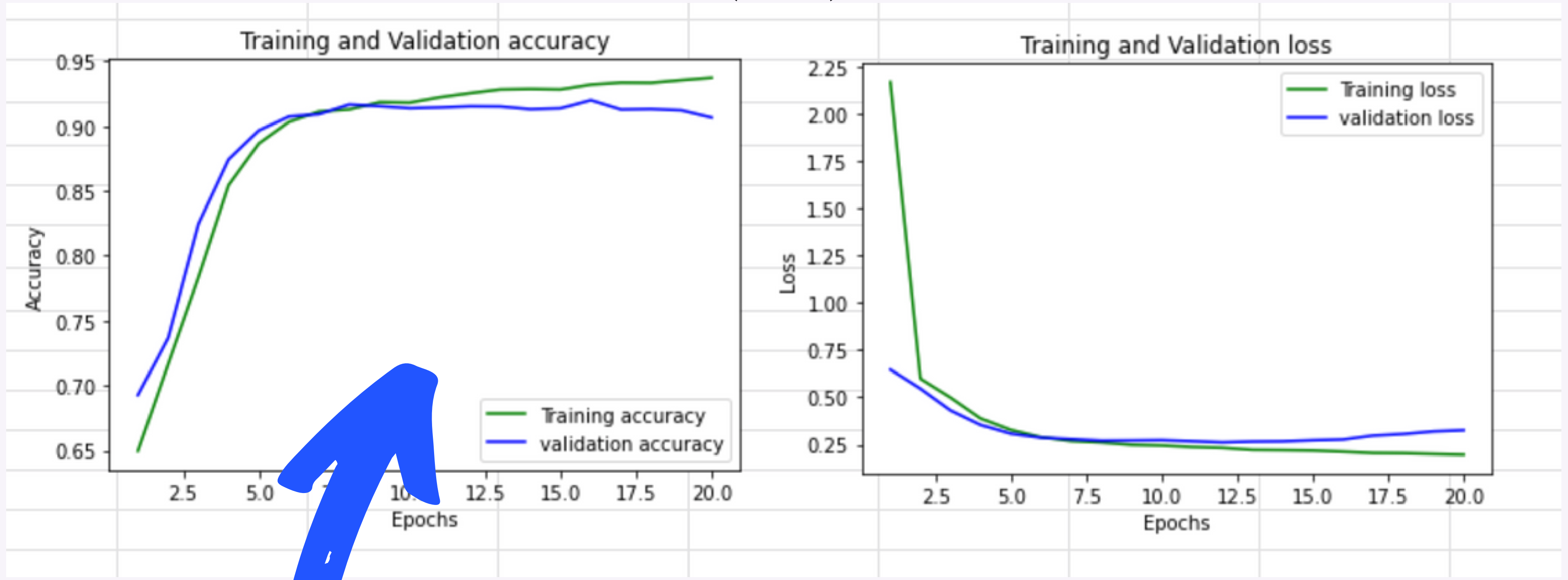


batch size : 64 with less validation

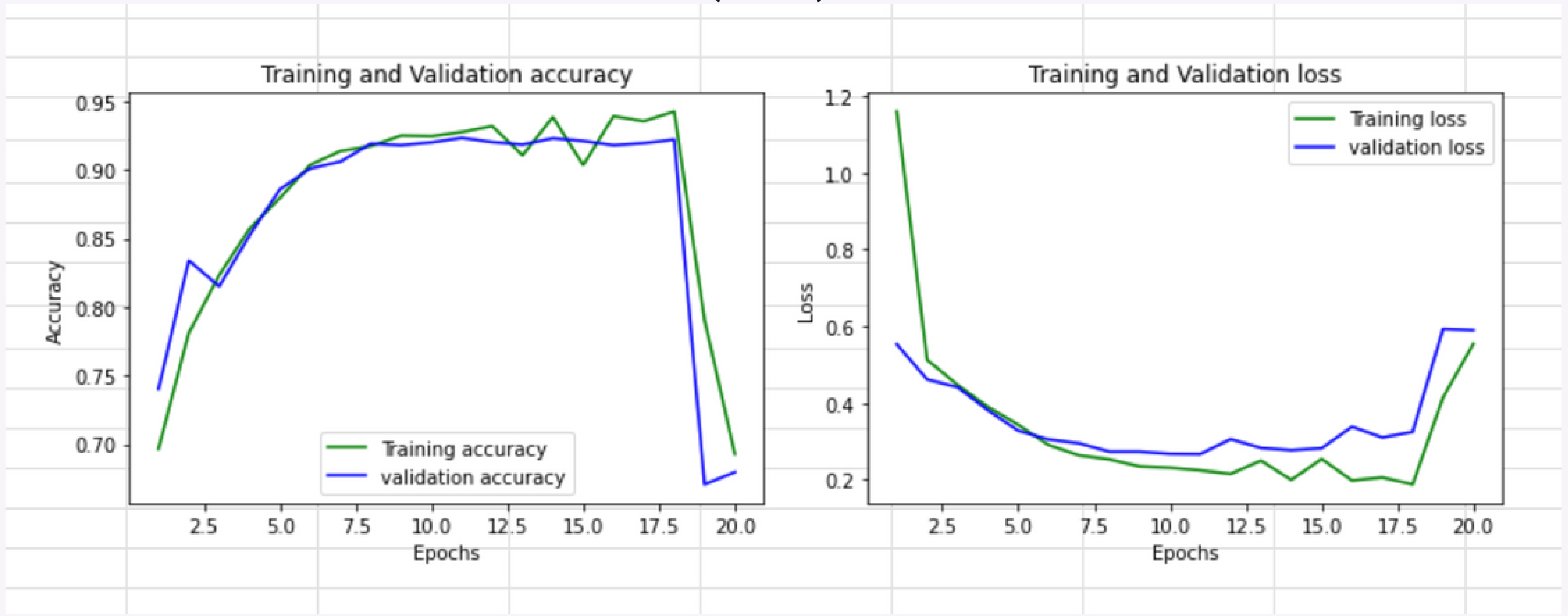
(3, 3)



(4, 4)



(5, 5)



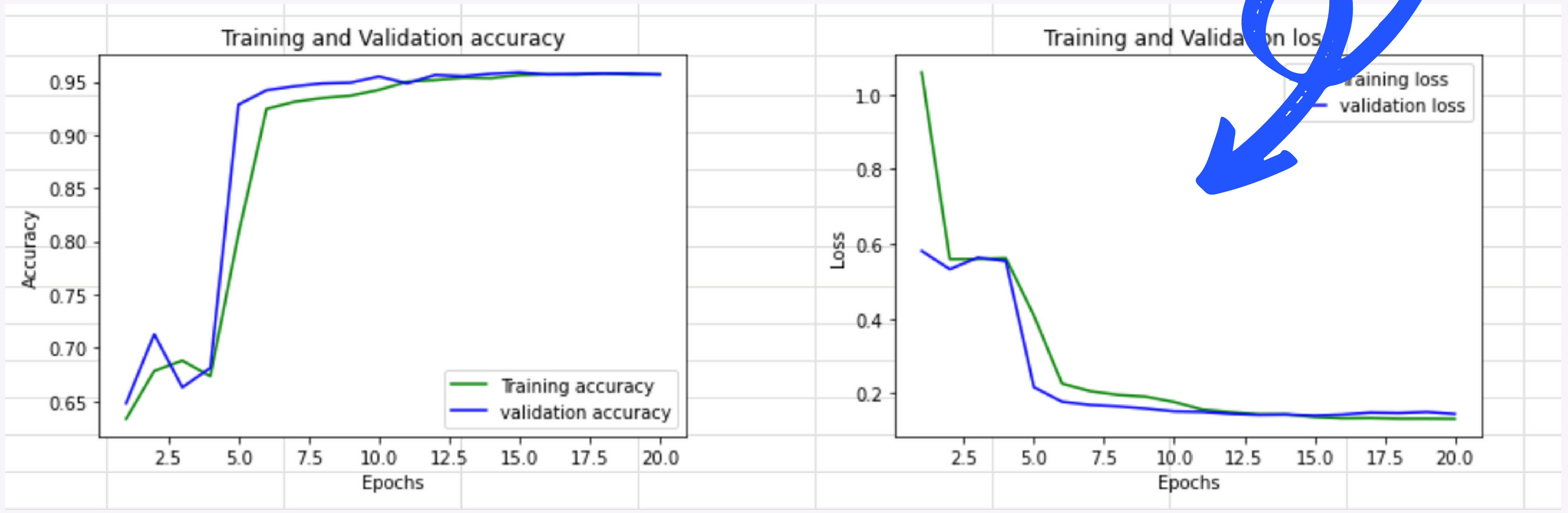
training and validation line is closer

To improve accuracy, add more convolution layer !

good accuracy, but with strange line

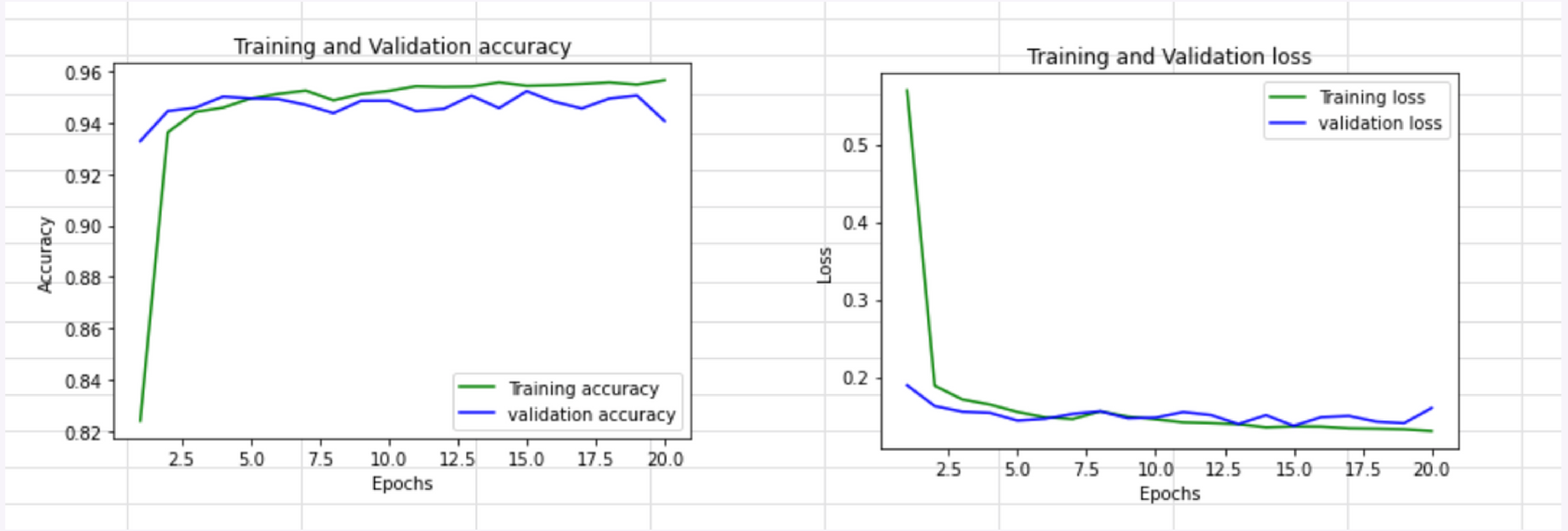
0.9573

two convolution, two pool

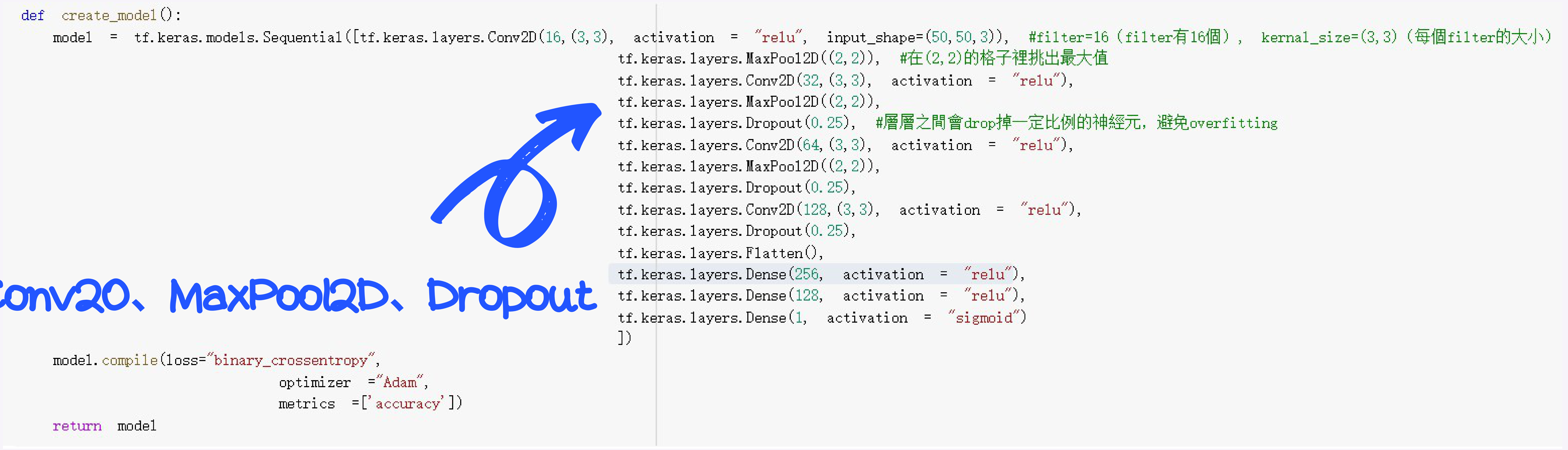


0.9417

three convolution, three pool



This is how we add more convolution layer !



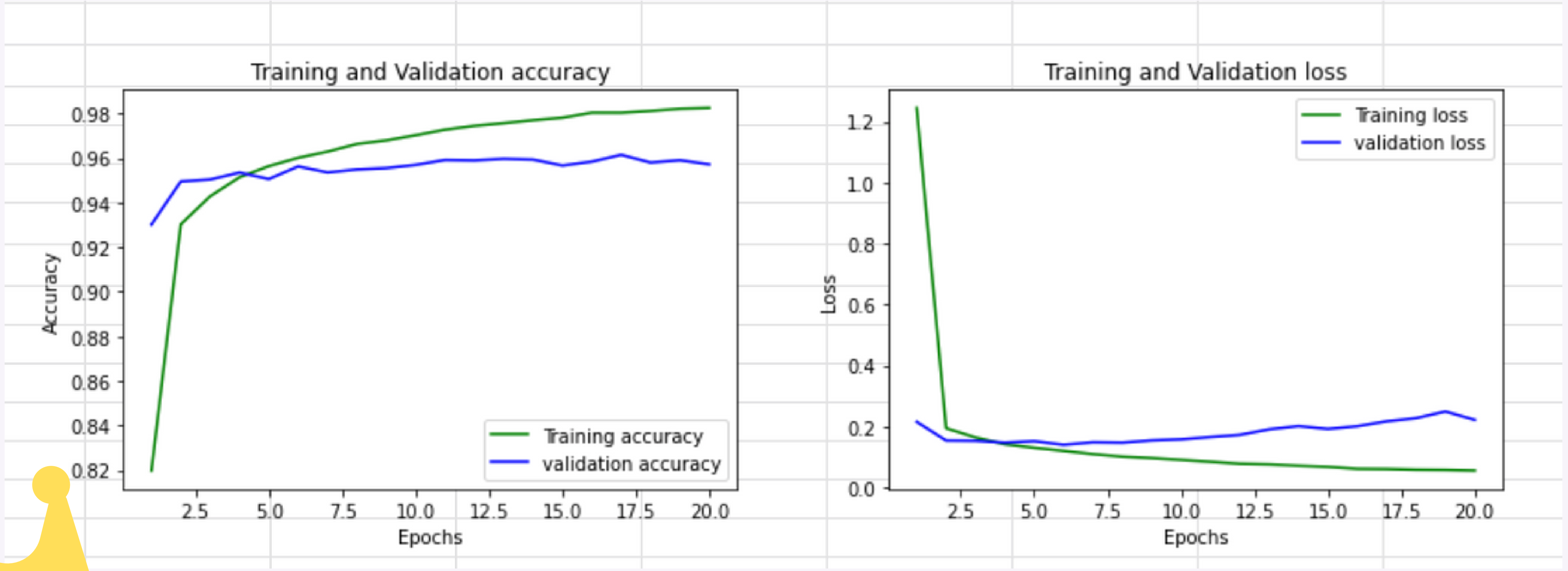
```
def  create_model():
    model  =  tf.keras.models.Sequential([tf.keras.layers.Conv2D(16,(3,3),  activation  =  "relu",  input_shape=(50,50,3)),  #filter=16 (filter有16個) ,  kernal_size=(3,3) (每個filter的大小)
                                        tf.keras.layers.MaxPool12D((2,2)),  #在(2,2)的格子裡挑出最大值
                                        tf.keras.layers.Conv2D(32,(3,3),  activation  =  "relu"),
                                        tf.keras.layers.MaxPool12D((2,2)),
                                        tf.keras.layers.Dropout(0.25),  #層層之間會drop掉一定比例的神經元, 避免overfitting
                                        tf.keras.layers.Conv2D(64,(3,3),  activation  =  "relu"),
                                        tf.keras.layers.MaxPool12D((2,2)),
                                        tf.keras.layers.Dropout(0.25),
                                        tf.keras.layers.Conv2D(128,(3,3),  activation  =  "relu"),
                                        tf.keras.layers.Dropout(0.25),
                                        tf.keras.layers.Flatten(),
                                        tf.keras.layers.Dense(256,  activation  =  "relu"),
                                        tf.keras.layers.Dense(128,  activation  =  "relu"),
                                        tf.keras.layers.Dense(1,  activation  =  "sigmoid")
                                        ])

    model.compile(loss="binary_crossentropy",
                  optimizer  ="Adam",
                  metrics  =['accuracy'])
    return  model
```

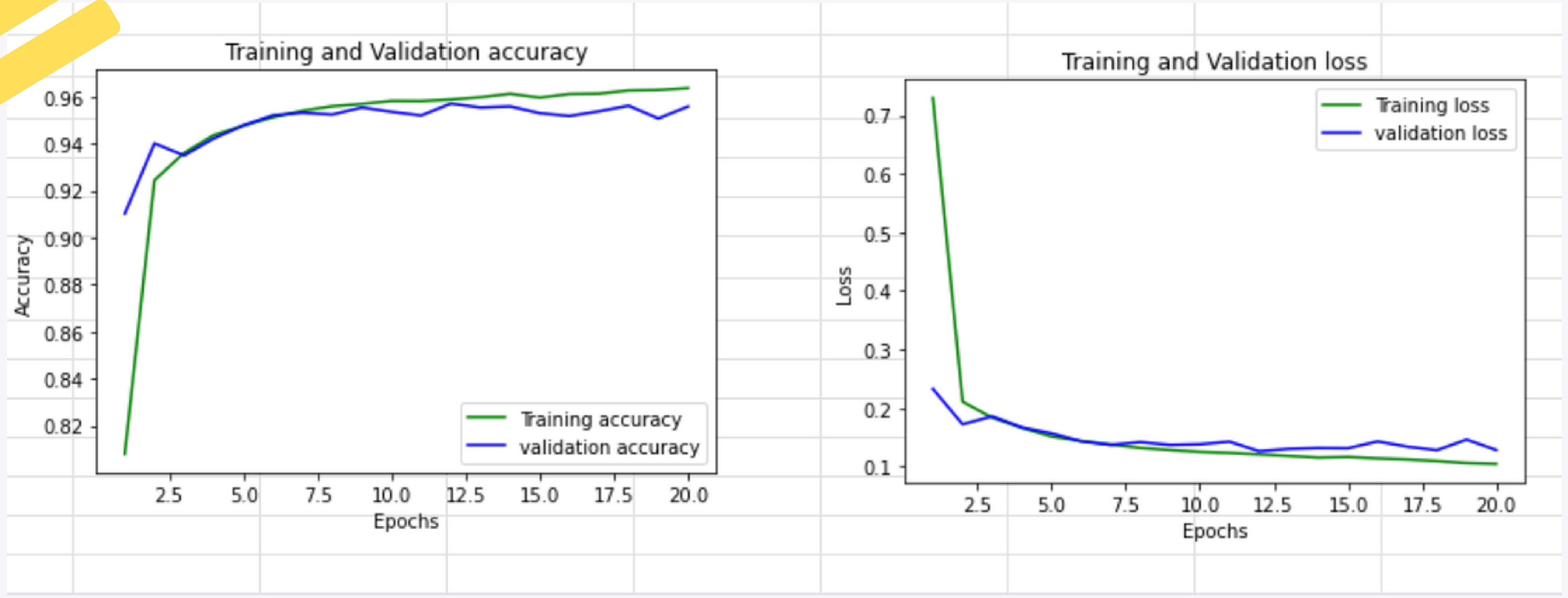
Conv2D、MaxPool2D、Dropout

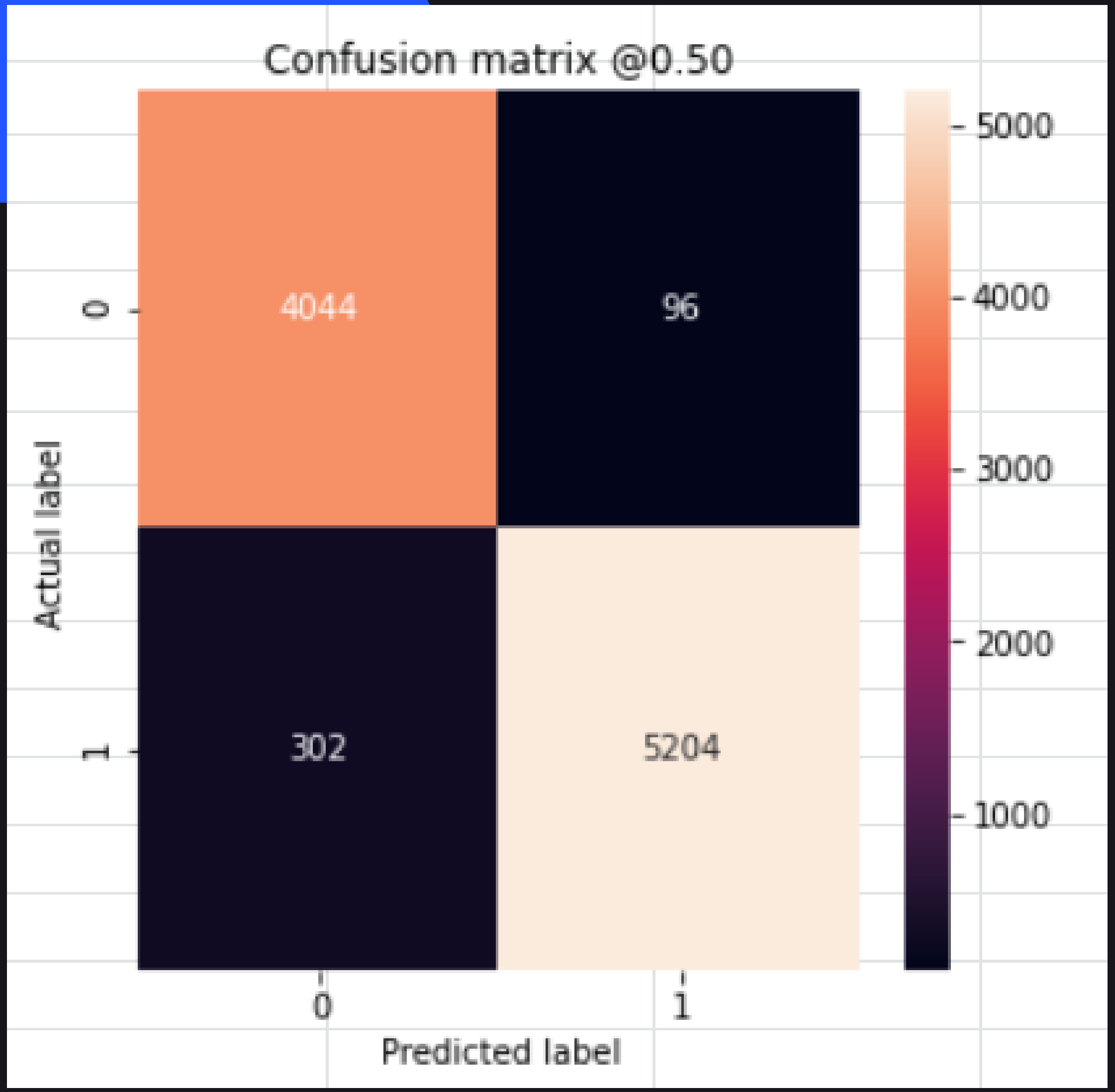
Add some activation function to improve

0.9584



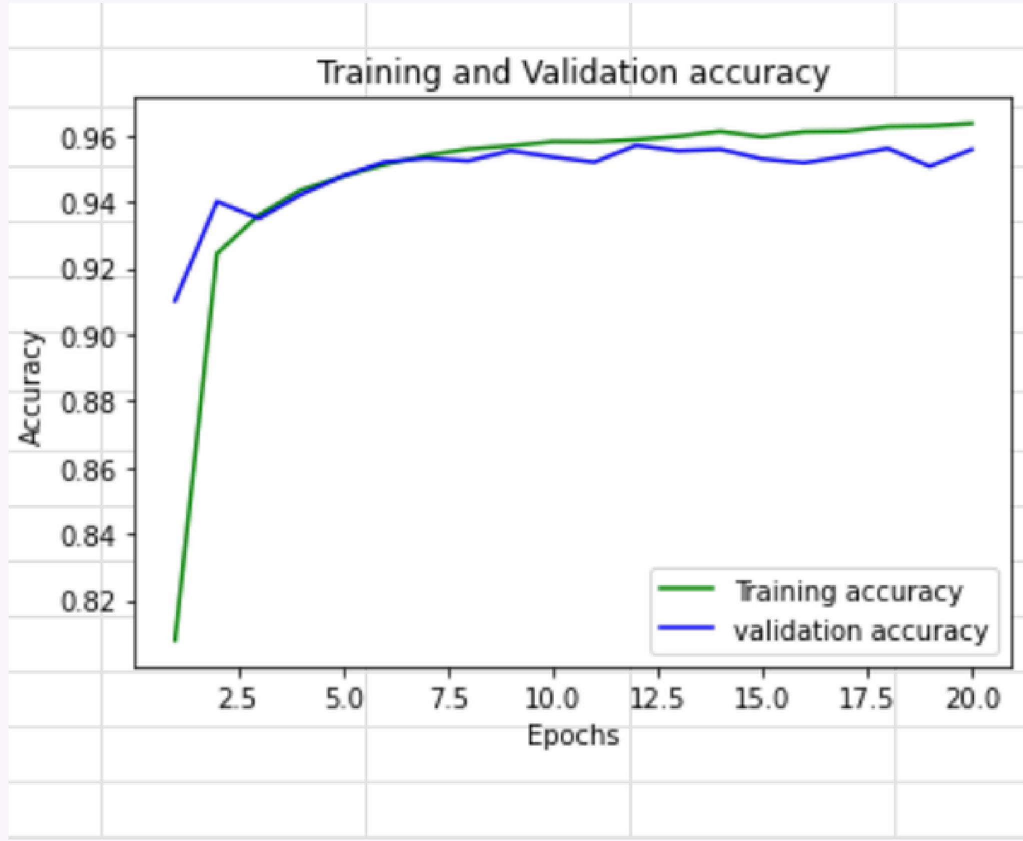
0.9587

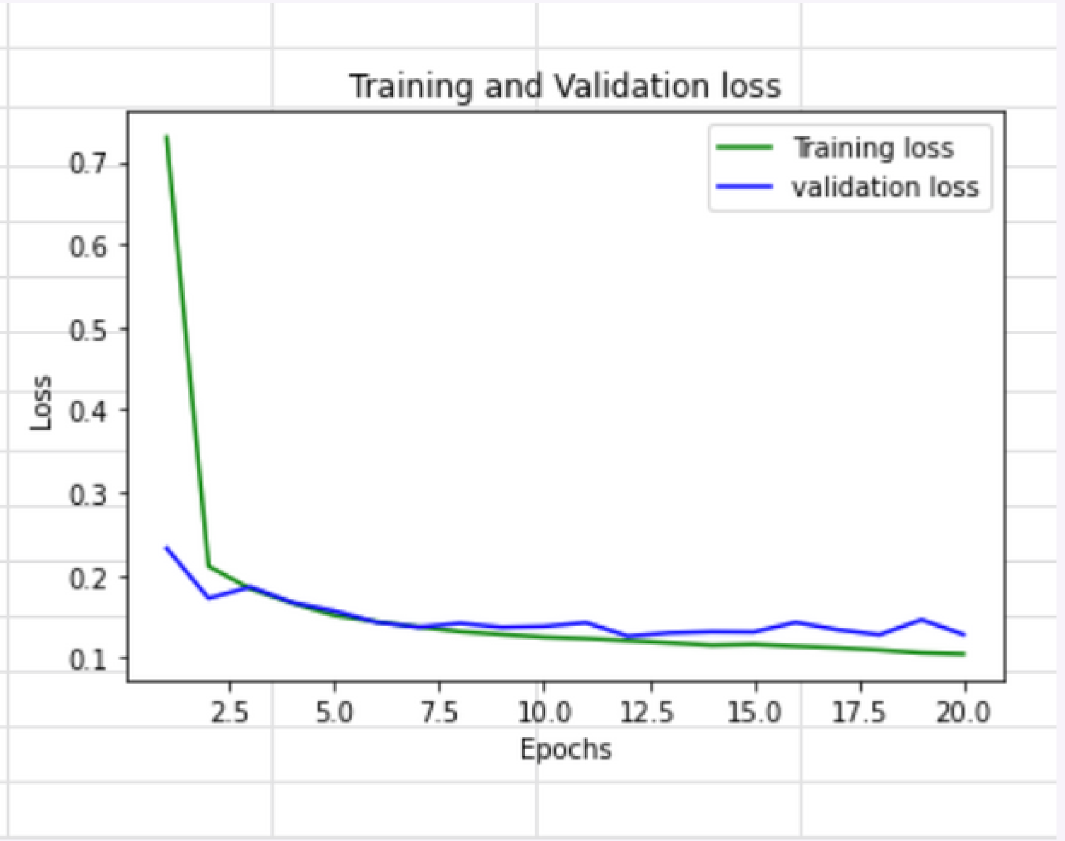




Parameters

Convolution: 16, (3, 3)/32, (3,3)/64, (3,3)
Activation: relu
Pooling: (2, 2)
L1-AF: relu L2-AF: relu L3-AF: sigmoid
Optimizer: Adam
epohs: 20
batch_size: 64
Accuracy: 0.9587





If you want to see our test progress, here is our test data excel.

https://reurl.cc/5olxLM

Thank You For Listening!