```
# install the dependencies
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.6/dist-packages (1.5.
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: python-slugify in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from k
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.6/dist-p
```

```
from google.colab import files
files.upload()
```

Choose Files kaggle.json

- **kaggle.json**(application/json) - 70 bytes, last modified: 5/17/2020 - 100% done

```
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"solveyourerror","key":"ba2ddc59efb623337c4f7d945d370cb
```

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
# change the permission
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d iarunava/cell-images-for-detecting-malaria
```

```
Downloading cell-images-for-detecting-malaria.zip to /content
100% 672M/675M [00:06<00:00, 116MB/s]
100% 675M/675M [00:06<00:00, 107MB/s]
```

```
from zipfile import ZipFile
file_name = "/content/cell-images-for-detecting-malaria.zip"
with ZipFile(file_name,'r') as zip:
  zip.extractall()
  print('Done')
```

```
Done
```

```
!pip install tensorflow-gpu==2.0.0-rc0
```

```
Collecting tensorflow-gpu==2.0.0-rc0
  Downloading https://files.pythonhosted.org/packages/6a/12/8c64cc62149cc21c70c550185
     |████████████████████████████████| 380.5MB 37kB/s
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: keras-applications>=1.0.8 in /usr/local/lib/python3.6/
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/local/lib/python3.6
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: gast>=0.2.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: astor>=0.6.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages
Collecting tb-nightly<1.15.0a20190807,>=1.15.0a20190806
  Downloading https://files.pythonhosted.org/packages/bc/88/24b5fb7280e74c7cf65bde47c
     |████████████████████████████████| 4.3MB 42.5MB/s
Requirement already satisfied: numpy<2.0,>=1.16.0 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: protobuf>=3.6.1 in /usr/local/lib/python3.6/dist-packa
Collecting tf-estimator-nightly<1.14.0.dev2019080602,>=1.14.0.dev2019080601
  Downloading https://files.pythonhosted.org/packages/21/28/f2a27a62943d5f041e4a6fd40
     |████████████████████████████████| 501kB 43.7MB/s
Requirement already satisfied: google-pasta>=0.1.6 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (from k
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.6/dist-pac
```

```
1 import tensorflow as tf
2 from tensorflow.keras import Sequential
3 from tensorflow.keras.layers import Flatten, Dense, Conv2D, MaxPool2D, Dropo
4
5 from tensorflow.keras.preprocessing.image import ImageDataGenerator
6
7 print(tf.__version__)
```

```
2.0.0-rc0
```

```
1 import numpy as np
2 import matplotlib.pyplot as plt
```

```
1 img_width = 64
2 img_height = 64
```

```
1 datagen = ImageDataGenerator(rescale=1/255.0, validation_split=0.2)
```

```
1 train_data_generator = datagen.flow_from_directory(directory='/content/cell_
2                                                    target_size = (img_width,
3                                                    class_mode = 'binary',
4                                                    batch_size = 16,
5                                                    subset = 'training'
6 )
```

```
Found 22048 images belonging to 2 classes.
```

```
validation_data_generator = datagen.flow_from_directory(directory='/content/
                                                        target_size = (img_width,
                                                        class_mode = 'binary',
                                                        batch_size = 16,
                                                        subset = 'validation'
)
```

```
Found 5510 images belonging to 2 classes.
```

```
train_data_generator.labels
```

```
array([0, 0, 0, ..., 1, 1, 1], dtype=int32)
```

CNN Model Building

```
model = Sequential()

model.add(Conv2D(16, (3,3), input_shape = (img_width, img_height, 3), activa
model.add(MaxPool2D(2,2))
model.add(Dropout(0.2))

model.add(Conv2D(32, (3,3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(1, activation='sigmoid'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 16)        448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 31, 31, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
```

```
# data optimization is adam
model.compile(optimizer='adam', loss='binary_crossentropy', metrics = ['accu
```

```
_________________________________________________________________
```

```
# train
history = model.fit_generator(generator=train_data_generator,
                              steps_per_epoch = len(train_data_generator),
                              epochs = 5,
                              validation_data = validation_data_generator,
                              validation_steps = len(validation_data_generat
```

```
rom /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/math_grad.py:13
lating:
 which has the same broadcast rule as np.where
==========================] - 164s 119ms/step - loss: 0.4860 - accuracy: 0.7667 - val_loss

==========================] - 163s 118ms/step - loss: 0.2736 - accuracy: 0.9195 - val_loss

==========================] - 163s 119ms/step - loss: 0.2463 - accuracy: 0.9321 - val_loss

==========================] - 165s 120ms/step - loss: 0.1964 - accuracy: 0.9365 - val_loss

==========================] - 168s 122ms/step - loss: 0.1882 - accuracy: 0.9371 - val_loss
```

```
history.history
```

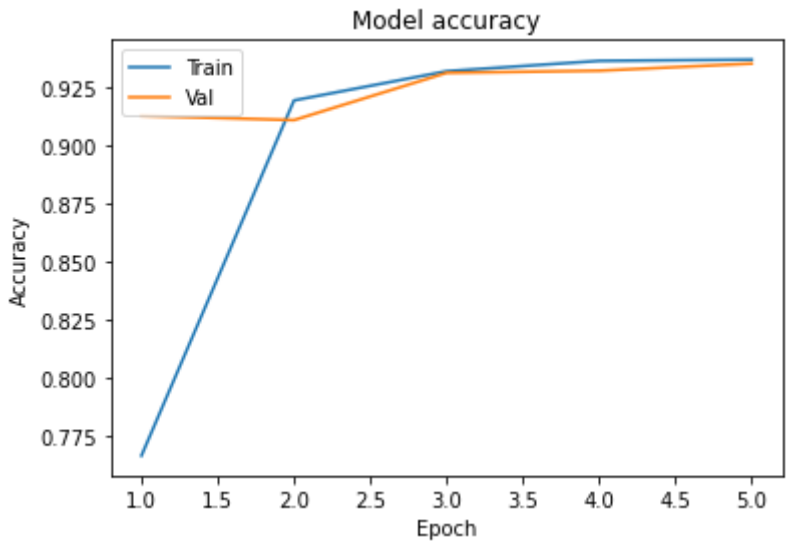
```
{'accuracy': [0.7667362, 0.9195392, 0.9321027, 0.9365475, 0.9370918],
 'loss': [0.48602462820276204,
  0.27357066065310914,
  0.24625902730263283,
  0.19637806036108119,
  0.1881721192722159],
 'val_accuracy': [0.91288567, 0.91107076, 0.93139744, 0.9323049, 0.9353902],
 'val_loss': [0.2681370893250341,
  0.27332649928504144,
  0.22460641115903854,
  0.18770437418226746,
  0.1834789704841872]}
```

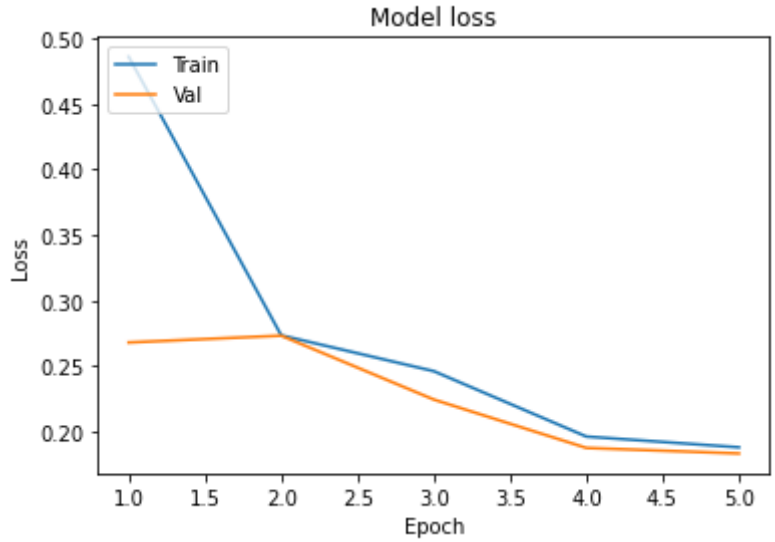
```
def plot_learningCurve(history, epoch):
  # Plot training & validation accuracy values
```

```
# Plot training & validation accuracy values
epoch_range = range(1, epoch+1)
plt.plot(epoch_range, history.history['accuracy'])
plt.plot(epoch_range, history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(epoch_range, history.history['loss'])
plt.plot(epoch_range, history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```

```
plot_learningCurve(history, 5)
```





Confusion Matrix

1