```
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint

# for accuracy and loss graph
import matplotlib.pyplot as plt
```

```
keras.__version__
```

```
'2.12.0'
```

```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from tensorflow.python import train
train_data_path="/content/drive/MyDrive/Data/data/train"
validation_data_path="/content/drive/MyDrive/Data/data/val"
```

```
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 5, figsize=(20, 20))
    axes = axes.flatten()
    for img, ax in zip(images_arr, axes):
        ax.imshow(img)
    plt.tight_layout()
    plt.show()
```

```
training_datagen = ImageDataGenerator(rescale=1./255,
                                      rotation_range=40,
                                      width_shift_range=0.2,
                                      height_shift_range=0.2,
                                      shear_range=0.2,
                                      zoom_range=0.2,
                                      horizontal_flip=True,
                                      fill_mode='nearest')

# this is a generator that will read pictures found in
# at train_data_path, and indefinitely generate
# batches of augmented image data
training_data = training_datagen.flow_from_directory(train_data_path, # this is the target directory
                                      target_size=(150, 150), # all images will be resized to 150x150
                                      batch_size=32,
                                      class_mode='binary')
```

```
Found 1951 images belonging to 4 classes.
```

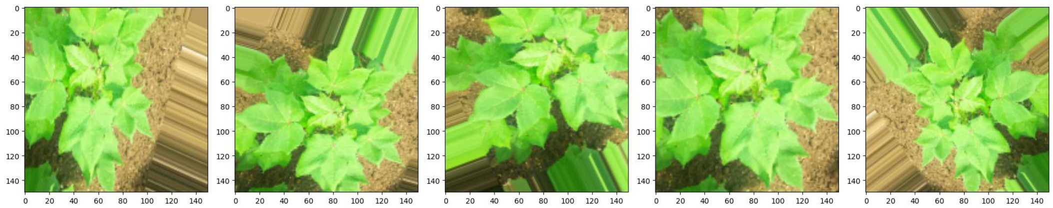
```
training_data.class_indices
```

```
{'diseased cotton leaf': 0,
 'diseased cotton plant': 1,
 'fresh cotton leaf': 2,
 'fresh cotton plant': 3}
```

```
valid_datagen = ImageDataGenerator(rescale=1./255)

# this is a similar generator, for validation data
valid_data = valid_datagen.flow_from_directory(validation_data_path,
                                  target_size=(150,150),
                                  batch_size=32,
                                  class_mode='binary')
```

```
Found 324 images belonging to 4 classes.
```

```
images = [training_data[0][0][0] for i in range(5)]
plotImages(images)
```



```
model_path = '/content/drive/MyDrive/Data/v3_pred_cott_dis.h5'
checkpoint = ModelCheckpoint(model_path, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]
```

```
#Building cnn model
cnn_model = keras.models.Sequential([
                                    keras.layers.Conv2D(filters=32, kernel_size=3, input_shape=[150, 150, 3]),
                                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                                    keras.layers.Conv2D(filters=64, kernel_size=3),
                                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                                    keras.layers.Conv2D(filters=128, kernel_size=3),
                                    keras.layers.MaxPooling2D(pool_size=(2,2)),
                                    keras.layers.Conv2D(filters=256, kernel_size=3),
                                    keras.layers.MaxPooling2D(pool_size=(2,2)),

                                    keras.layers.Dropout(0.5),
                                    keras.layers.Flatten(), # neural network beulding
                                    keras.layers.Dense(units=128, activation='relu'), # input layers
                                    keras.layers.Dropout(0.1),
                                    keras.layers.Dense(units=256, activation='relu'),
                                    keras.layers.Dropout(0.25),
                                    keras.layers.Dense(units=4, activation='softmax') # output layer
])
```

```
# compile cnn model
cnn_model.compile(optimizer = Adam(learning_rate=0.0001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
cnn_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 256)       295168

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 256)        0
 2D)

 dropout (Dropout)           (None, 7, 7, 256)         0

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 128)               1605760

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 256)               33024
```

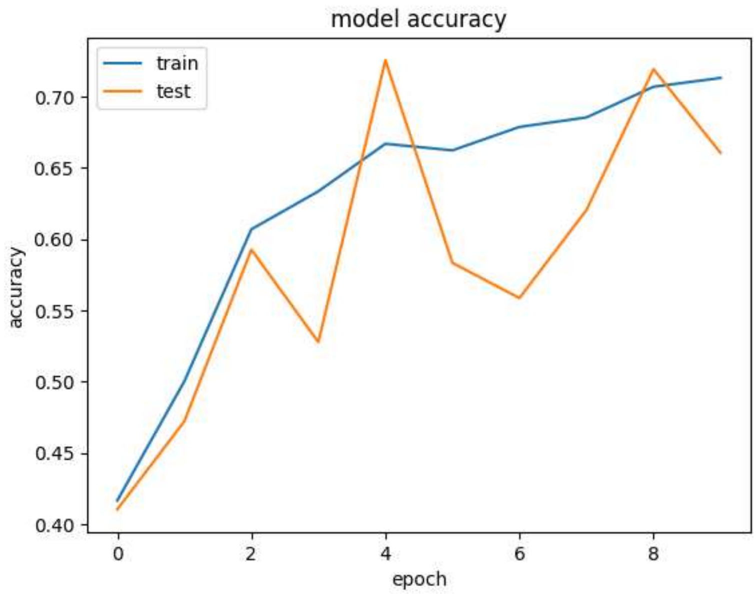
```
 dropout_2 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 4)                 1028

=================================================================
Total params: 2,028,228
Trainable params: 2,028,228
Non-trainable params: 0
_________________________________________________________________
```

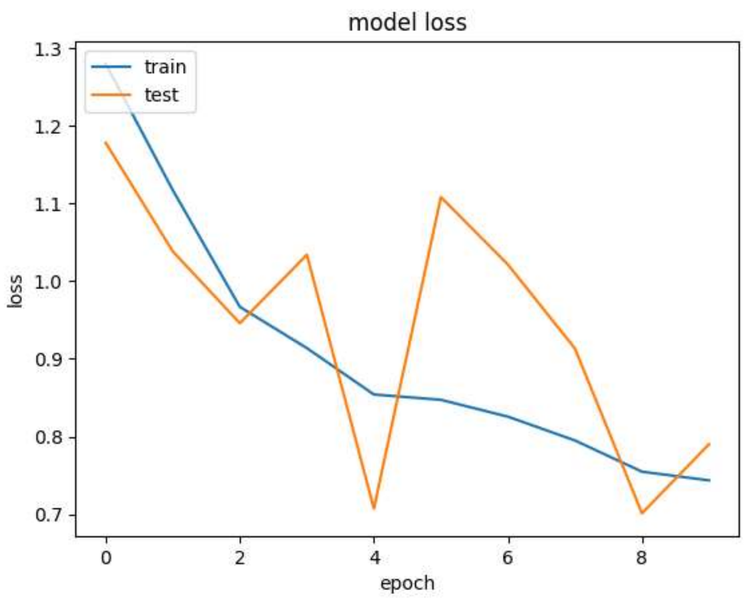
```
# train cnn model
history = cnn_model.fit(training_data,
                          epochs=10,
                          verbose=1,
                          validation_data= valid_data,
                          callbacks=callbacks_list) # time start 16.06
```

```
Epoch 1/10
61/61 [==============================] - ETA: 0s - loss: 1.2799 - accuracy: 0.4167
Epoch 1: val_accuracy improved from -inf to 0.41049, saving model to /content/drive/MyDrive/Data/v3_pred_cott_dis.h5
61/61 [==============================] - 531s 9s/step - loss: 1.2799 - accuracy: 0.4167 - val_loss: 1.1781 - val_accuracy: 0.4105
Epoch 2/10
61/61 [==============================] - ETA: 0s - loss: 1.1173 - accuracy: 0.5003
Epoch 2: val_accuracy improved from 0.41049 to 0.47222, saving model to /content/drive/MyDrive/Data/v3_pred_cott_dis.h5
61/61 [==============================] - 151s 2s/step - loss: 1.1173 - accuracy: 0.5003 - val_loss: 1.0383 - val_accuracy: 0.4722
Epoch 3/10
61/61 [==============================] - ETA: 0s - loss: 0.9670 - accuracy: 0.6069
Epoch 3: val_accuracy improved from 0.47222 to 0.59259, saving model to /content/drive/MyDrive/Data/v3_pred_cott_dis.h5
61/61 [==============================] - 150s 2s/step - loss: 0.9670 - accuracy: 0.6069 - val_loss: 0.9460 - val_accuracy: 0.5926
Epoch 4/10
61/61 [==============================] - ETA: 0s - loss: 0.9139 - accuracy: 0.6335
Epoch 4: val_accuracy did not improve from 0.59259
61/61 [==============================] - 150s 2s/step - loss: 0.9139 - accuracy: 0.6335 - val_loss: 1.0341 - val_accuracy: 0.5278
Epoch 5/10
61/61 [==============================] - ETA: 0s - loss: 0.8542 - accuracy: 0.6668
Epoch 5: val_accuracy improved from 0.59259 to 0.72531, saving model to /content/drive/MyDrive/Data/v3_pred_cott_dis.h5
61/61 [==============================] - 155s 3s/step - loss: 0.8542 - accuracy: 0.6668 - val_loss: 0.7072 - val_accuracy: 0.7253
Epoch 6/10
61/61 [==============================] - ETA: 0s - loss: 0.8473 - accuracy: 0.6622
Epoch 6: val_accuracy did not improve from 0.72531
61/61 [==============================] - 151s 2s/step - loss: 0.8473 - accuracy: 0.6622 - val_loss: 1.1083 - val_accuracy: 0.5833
Epoch 7/10
61/61 [==============================] - ETA: 0s - loss: 0.8256 - accuracy: 0.6786
Epoch 7: val_accuracy did not improve from 0.72531
61/61 [==============================] - 150s 2s/step - loss: 0.8256 - accuracy: 0.6786 - val_loss: 1.0215 - val_accuracy: 0.5586
Epoch 8/10
61/61 [==============================] - ETA: 0s - loss: 0.7949 - accuracy: 0.6853
Epoch 8: val_accuracy did not improve from 0.72531
61/61 [==============================] - 153s 3s/step - loss: 0.7949 - accuracy: 0.6853 - val_loss: 0.9133 - val_accuracy: 0.6204
Epoch 9/10
61/61 [==============================] - ETA: 0s - loss: 0.7547 - accuracy: 0.7068
Epoch 9: val_accuracy did not improve from 0.72531
61/61 [==============================] - 157s 3s/step - loss: 0.7547 - accuracy: 0.7068 - val_loss: 0.7012 - val_accuracy: 0.7191
Epoch 10/10
61/61 [==============================] - ETA: 0s - loss: 0.7435 - accuracy: 0.7130
Epoch 10: val_accuracy did not improve from 0.72531
61/61 [==============================] - 156s 3s/step - loss: 0.7435 - accuracy: 0.7130 - val_loss: 0.7901 - val_accuracy: 0.6605
```

```
model_path2 = '/content/drive/MyDrive/Data/v3_pred_cott_dis.h5'
cnn_model.save(model_path2)
```

```
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```






```
history.history
```

```
{'loss': [1.2798739671707153,
  1.1172654628753662,
  0.9669752717018127,
  0.9139034748077393,
  0.8541598320007324,
  0.8472960591316223,
  0.825603187084198,
  0.7949424982070923,
  0.7547361254692078,
  0.7435344457626343],
 'accuracy': [0.41670939326286316,
  0.5002562999725342,
  0.6068682670593262,
  0.6335212588310242,
  0.6668375134468079,
  0.6622244715690613,
  0.6786263585090637,
  0.6852896213531494,
  0.7068170309066772,
  0.7129676938056946],
 'val_loss': [1.1781359910964966,
  1.0383433103561401,
  0.9460245966911316,
  1.0341404676437378,
  0.707227349281311,
  1.1083322763442993,
  1.0214567184448242,
  0.9133340120315552,
  0.7011666893959045,
```

```
  0.7900735139846802],
 'val_accuracy': [0.4104938209056854,
  0.4722222089767456,
  0.5925925970077515,
  0.5277777910232544,
  0.7253086566925049,
  0.5833333134651184,
  0.5586419701576233,
  0.620370388031059,
  0.7191358208656311,
  0.6604938507080078]}
```