```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam, SGD
from keras.datasets import cifar100
from keras.layers import Conv2D, MaxPooling2D,Flatten
import keras
import cv2
import glob
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import keras
from keras.models import *
from keras.layers import *

from keras.callbacks import EarlyStopping
```

```
# train
directory = '/content/drive/MyDrive/Colab Notebooks/Data/train'

data_list = []
for filename in os.listdir(directory):
    for img in os.listdir(directory+"/"+filename):
        data_list.append({'directory':directory+"/"+filename+"/"+img, 'class':filename})

data_df = pd.concat([pd.DataFrame(data_list)], ignore_index=True)# train
directory = '/content/drive/MyDrive/Colab Notebooks/Data/train'

data_list = []
for filename in os.listdir(directory):
    for img in os.listdir(directory+"/"+filename):
        data_list.append({'directory':directory+"/"+filename+"/"+img, 'class':filename})

data_df = pd.concat([pd.DataFrame(data_list)], ignore_index=True)
```

```
data_df = data_df.sample(frac = 1, random_state=7)
print(data_df.head(5))

test_img_path = data_df.iloc[2][0]
test_img = cv2.imread(test_img_path)
print(test_img_path.split("/")[-2], test_img_path.split("/")[-1])
print(test_img.shape)
plt.imshow(test_img)
```

```
                                            directory  \
293  /content/drive/MyDrive/Colab Notebooks/Data/tr...
501  /content/drive/MyDrive/Colab Notebooks/Data/tr...
318  /content/drive/MyDrive/Colab Notebooks/Data/tr...
329  /content/drive/MyDrive/Colab Notebooks/Data/tr...
370  /content/drive/MyDrive/Colab Notebooks/Data/tr...

                                              class
293       adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib
501                                          normal
318  squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa
329  squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa
370  squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa
squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa 000069 (2).png
(314, 400, 3)
<matplotlib.image.AxesImage at 0x7d8464fc6440>
```


```
batch_size = 64
size = (224,224,3)
img_width = img_hight = size[0]
random_state = 7

classes = list(data_df['class'].unique())
classes
```

```
['adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib',
 'normal',
 'squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa',
 'large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa']
```

```
from keras.preprocessing.image import ImageDataGenerator

train_gen = ImageDataGenerator(rescale=1./255)
train_data = train_gen.flow_from_directory(
    '/content/drive/MyDrive/Colab Notebooks/Data/train',target_size=(img_hight, img_hight),
    batch_size=batch_size, class_mode='categorical',
    classes=classes, seed=42,shuffle=True,subset='training')

test_gen = ImageDataGenerator(rescale=1./255)
test_data = train_gen.flow_from_directory(
    '/content/drive/MyDrive/Colab Notebooks/Data/test',target_size=(img_hight, img_hight),
    batch_size=batch_size, class_mode='categorical',
    classes=classes, seed=42,shuffle=True)

valid_gen = ImageDataGenerator(rescale=1./255)
valid_data = train_gen.flow_from_directory(
    '/content/drive/MyDrive/Colab Notebooks/Data/valid',target_size=(img_hight, img_hight),
    batch_size=batch_size, class_mode='categorical',
    classes=classes, seed=42,shuffle=True)
```

```
Found 613 images belonging to 4 classes.
Found 54 images belonging to 4 classes.
Found 72 images belonging to 4 classes.
```

```
def Create_model(Image_shape, block1=True, block2=True, block3=True,
                 block4=True, block5=True, regularizer=keras.regularizers.l2(0.0001),
                 Dropout_ratio=0.25):

    # * Create the model
    model = keras.Sequential()

    # * configure the inputshape
    model.add(keras.Input(shape=Image_shape))

    # * Add the first block
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu',
              trainable=block1))
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu',
              trainable=block1))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(BatchNormalization())

    # * Add the second block
    model.add(Conv2D(128, (3, 3), padding='same', activation='relu',
              trainable=block2))
    model.add(Conv2D(128, (3, 3), padding='same', activation='relu',
              trainable=block2))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(BatchNormalization())

    # * Add the third block
    model.add(Conv2D(256, (3, 3), padding='same', activation='relu',
              trainable=block3))
    model.add(Conv2D(256, (3, 3), padding='same', activation='relu',
              trainable=block3))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(BatchNormalization())
       # * Add the fourth block
    model.add(Conv2D(512, (3, 3), padding='same', activation='relu',
              trainable=block4))
    model.add(Conv2D(512, (3, 3), padding='same', activation='relu',
              trainable=block4))
    model.add(Conv2D(512, (3, 3), padding='same', activation='relu',
              trainable=block4))

    #* flatten + Fc layer
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(Dropout_ratio))

    # * Output layer
    #model.add(Dense(3, activation='linear'))
    model.add(Dense(4, activation='softmax'))
    print('Done')
    return model
model = Create_model(size)
```

```
Done
```

```
model.compile(Adam(), keras.losses.CategoricalCrossentropy(),metrics='accuracy')
```

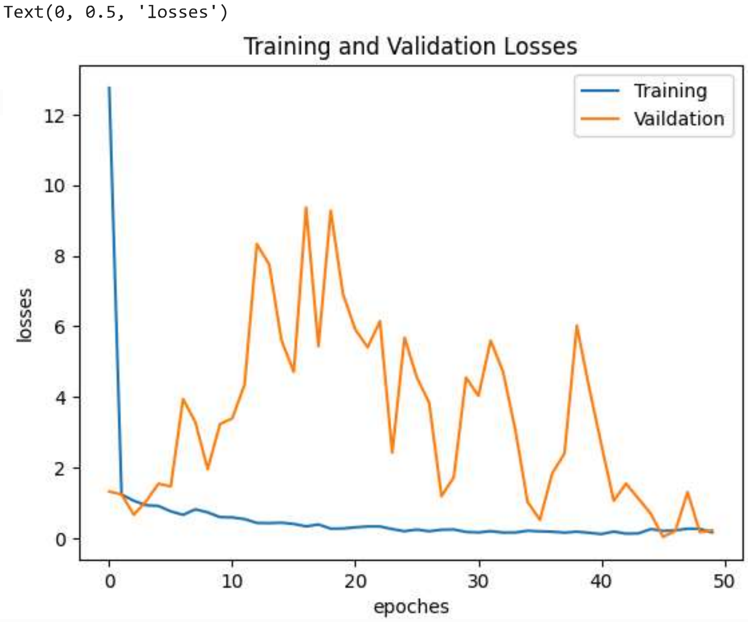
```
results=model.fit(
    train_data,
    validation_data= test_data,
    epochs=50
  )
```

```
Epoch 29/50
```

```
Epoch 29/50
10/10 [==============================] - 8s 759ms/step - loss: 0.2543 - accuracy: 0.8858 - val_loss: 1.7306 - val_accuracy: 0.5185
Epoch 30/50
10/10 [==============================] - 8s 762ms/step - loss: 0.1837 - accuracy: 0.9282 - val_loss: 4.5557 - val_accuracy: 0.5370
Epoch 31/50
10/10 [==============================] - 8s 761ms/step - loss: 0.1699 - accuracy: 0.9201 - val_loss: 4.0326 - val_accuracy: 0.3148
Epoch 32/50
10/10 [==============================] - 8s 802ms/step - loss: 0.2069 - accuracy: 0.9119 - val_loss: 5.6001 - val_accuracy: 0.1481
Epoch 33/50
10/10 [==============================] - 8s 814ms/step - loss: 0.1664 - accuracy: 0.9445 - val_loss: 4.7236 - val_accuracy: 0.1111
Epoch 34/50
10/10 [==============================] - 8s 781ms/step - loss: 0.1684 - accuracy: 0.9266 - val_loss: 3.0770 - val_accuracy: 0.1481
Epoch 35/50
10/10 [==============================] - 8s 773ms/step - loss: 0.2183 - accuracy: 0.9038 - val_loss: 1.0337 - val_accuracy: 0.7222
Epoch 36/50
10/10 [==============================] - 8s 771ms/step - loss: 0.2015 - accuracy: 0.9168 - val_loss: 0.5253 - val_accuracy: 0.7778
Epoch 37/50
10/10 [==============================] - 8s 748ms/step - loss: 0.1914 - accuracy: 0.9086 - val_loss: 1.8404 - val_accuracy: 0.5741
Epoch 38/50
10/10 [==============================] - 8s 785ms/step - loss: 0.1651 - accuracy: 0.9282 - val_loss: 2.4086 - val_accuracy: 0.5741
Epoch 39/50
10/10 [==============================] - 8s 762ms/step - loss: 0.1914 - accuracy: 0.9103 - val_loss: 6.0211 - val_accuracy: 0.2963
Epoch 40/50
10/10 [==============================] - 8s 794ms/step - loss: 0.1598 - accuracy: 0.9315 - val_loss: 4.2702 - val_accuracy: 0.2963
Epoch 41/50
10/10 [==============================] - 8s 813ms/step - loss: 0.1252 - accuracy: 0.9413 - val_loss: 2.6457 - val_accuracy: 0.6852
Epoch 42/50
10/10 [==============================] - 8s 761ms/step - loss: 0.1937 - accuracy: 0.9250 - val_loss: 1.0659 - val_accuracy: 0.7037
Epoch 43/50
10/10 [==============================] - 8s 808ms/step - loss: 0.1369 - accuracy: 0.9429 - val_loss: 1.5515 - val_accuracy: 0.5741
Epoch 44/50
10/10 [==============================] - 8s 812ms/step - loss: 0.1431 - accuracy: 0.9282 - val_loss: 1.1304 - val_accuracy: 0.6667
Epoch 45/50
10/10 [==============================] - 8s 761ms/step - loss: 0.2645 - accuracy: 0.9315 - val_loss: 0.7067 - val_accuracy: 0.7222
Epoch 46/50
10/10 [==============================] - 8s 773ms/step - loss: 0.2139 - accuracy: 0.9168 - val_loss: 0.0455 - val_accuracy: 1.0000
Epoch 47/50
10/10 [==============================] - 8s 759ms/step - loss: 0.2288 - accuracy: 0.9282 - val_loss: 0.2101 - val_accuracy: 0.9815
Epoch 48/50
10/10 [==============================] - 8s 802ms/step - loss: 0.2774 - accuracy: 0.9021 - val_loss: 1.3082 - val_accuracy: 0.9074
Epoch 49/50
10/10 [==============================] - 8s 808ms/step - loss: 0.2719 - accuracy: 0.8972 - val_loss: 0.1775 - val_accuracy: 0.9815
Epoch 50/50
10/10 [==============================] - 8s 775ms/step - loss: 0.1705 - accuracy: 0.9184 - val_loss: 0.2339 - val_accuracy: 0.9074
```

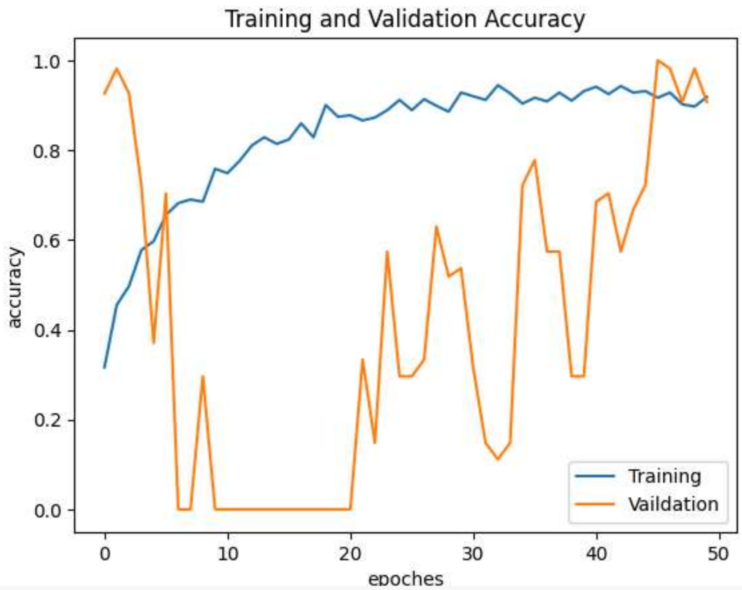
```
plt.plot(results.history['loss'])
plt.plot(results.history['val_loss'])
plt.legend(['Training','Vaildation'])
plt.title('Training and Validation Losses')
plt.xlabel('epoches')
plt.ylabel('losses')
```

```
Text(0, 0.5, 'losses')
```



```
plt.plot(results.history['accuracy'])
plt.plot(results.history['val_accuracy'])
plt.legend(['Training','Vaildation'])
plt.title('Training and Validation Accuracy')
plt.xlabel('epoches')
plt.ylabel('accuracy')
```

```
Text(0, 0.5, 'accuracy')
```




```
y_pred= model.predict(test_data)
import numpy as np

y=np.round(y_pred).flatten()

y_pred1=np.argmax(y_pred,axis=1)
```

```
1/1 [==============================] - 1s 611ms/step
```

```
from sklearn.metrics import accuracy_score,  confusion_matrix
confusion_matrix = confusion_matrix(test_data.labels,y_pred1)
print(confusion_matrix)

accuracy_score(test_data.labels,y_pred1)*100
```

```
[[ 0  0  0]
 [ 1 49  4]
 [ 0  0  0]]
90.74074074074075
```

```
from sklearn.metrics import classification_report
report=classification_report(test_data.labels,y_pred1)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.91      0.95        54
           2       0.00      0.00      0.00         0

    accuracy                           0.91        54
   macro avg       0.33      0.30      0.32        54
weighted avg       1.00      0.91      0.95        54

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defi
  _warn_prf(average, modifier, msg_start, len(result))
```