

Image Scraping and Classification Project

Submitted by:

Priyanka Yadav

# ACKNOWLEDGMENT

I would like to thank FlipRobo Technologies to provide me this opportunity to perform this analysis. I would also like to thank my mentor Swati Mahaseth for helping me throughout this project.

# Import libraries.

*# Core*

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

import glob import random import os import cv2

*#tesor fow & keras*

import tensorflow as tf from tensorflow import keras

from keras.regularizers import l2 from sklearn.utils import shuffle

from sklearn.model\_selection import train\_test\_split from tensorflow.keras.utils import to\_categorical from keras.models import load\_model

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Flatten,

Dense,BatchNormalization,Dropout,Input from keras.models import Sequential, Model

from keras.layers import Conv2D,GlobalMaxPooling2D from tensorflow.keras.applications import Xception,VGG16,InceptionResNetV2

from tensorflow.keras.preprocessing.image import ImageDataGenerator

*#cnn*

from tensorflow.keras import datasets, layers, models

from keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D, concatenate, Conv2DTranspose, BatchNormalization, Dropout, Lambda from keras.engine.base\_layer import Layer

from sklearn.metrics import classification\_report,confusion\_matrix from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score from sklearn.metrics import recall\_score

from sklearn.metrics import precision\_score

seed = 42 np.random.seed =seed

image\_path = './'

# Helper Function

*#this function used to get code simple # get label Name*

**def** get\_Label(number):

labels = {0:'saree', 1:'jeans', 2:'trouser'}

**return** labels[number]

*#plot predction function*

**def** plot\_predection(model\_name): plt.figure(figsize=(20,15)) plt.suptitle("Predection Images", fontsize=20) images = []

path =image\_path+'/'+'test/' count = 0 *#val\_images,val\_labels*

**for** i,files **in** enumerate(os.listdir(path)) : img = plt.imread(path+files)

img = cv2.resize(img,(128,128)) plt.imshow(img,cmap=plt.cm.binary) img = np.expand\_dims(img, axis=0) feature = model\_name.predict(img)

predection = np.argmax(feature, axis=1) plt.subplot(5,7,i+1)

plt.xticks([])

plt.yticks([]) plt.grid(False)

plt.xlabel("Predicted"+get\_Label(int(predection))) count += 1

**if** count == 34 :

**break**

# Data Preperation

### Load Labels

* collect all image label and add it in data frame. Load Images

### Train images

* Test images

### Validation images

*# 0 for saree ,1 for jeans ,2 for trouser*

all\_categories = []

images = os.listdir(image\_path+'/'+'train')

**for** file **in** images:

**if** file.split('.')[0] == 'saree': all\_categories.append(0)

**elif** file.split('.')[0] == 'jeans': all\_categories.append(1)

#### else:

all\_categories.append(2)

*#creat data fram to save each image with its label*

df =pd.DataFrame({ 'image': images,

'category':all\_categories

})

df.head(5)

image category

1. saree.0.jpg 0
2. saree.2.jpg 0
3. saree.1.jpg 0
4. saree.10.jpg 0
5. saree.3.jpg 0

df['category'].value\_counts() 0 200

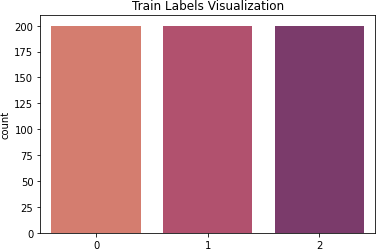
1 200

2 200

Name: category, dtype: int64

# Virsialization Train Images

plt.title('Train Labels Visualization') sns.countplot(x=all\_categories,palette='flare') plt.show()



### Frome Here We Find Data is Totaly Balanced.

plt.figure(figsize=(20,15)) plt.suptitle("Train Images", fontsize=20) path =image\_path+'/'+'train/'

counter =0

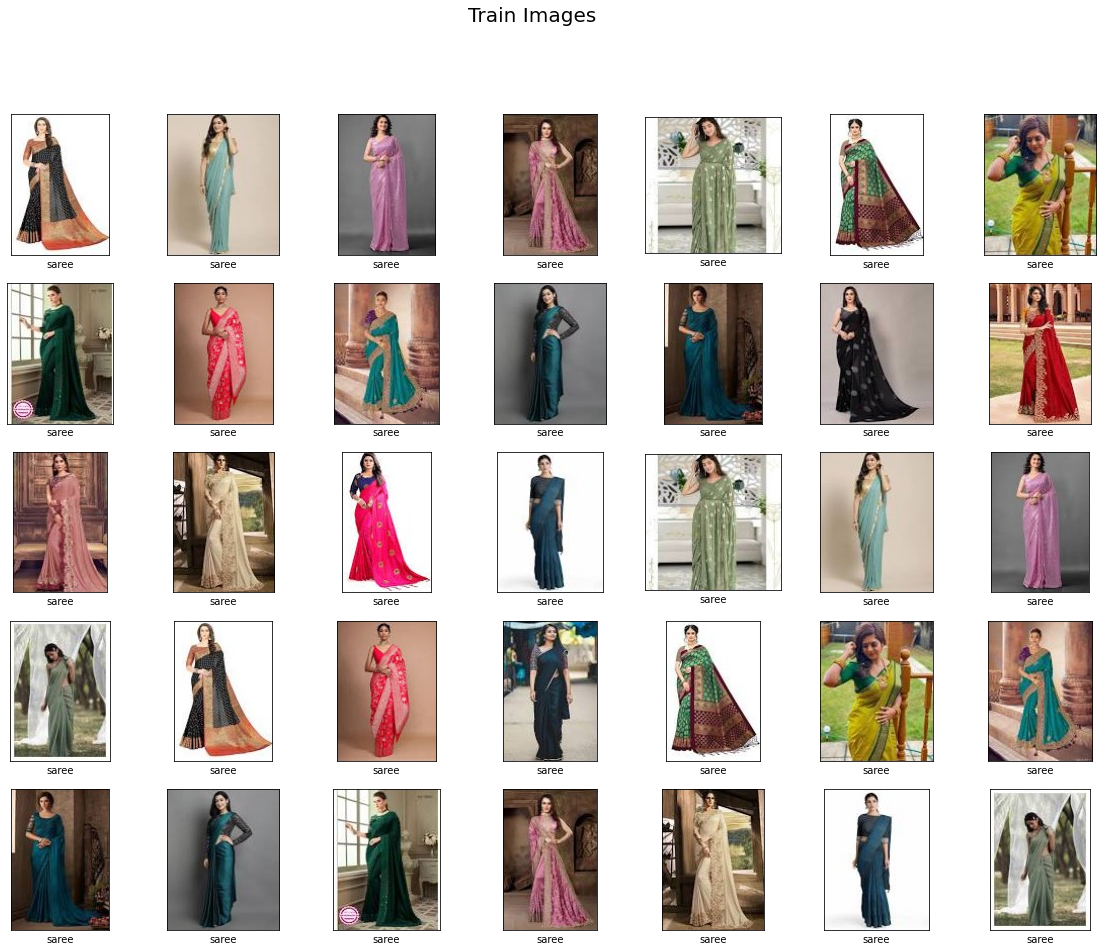
**for** i,img **in** enumerate(os.listdir(path)) : plt.subplot(5,7,i+1)

full\_image= plt.imread(path+img) plt.xticks([])

plt.yticks([]) plt.grid(False)

plt.xlabel(get\_Label(all\_categories[i])) plt.imshow(full\_image, cmap=plt.cm.binary) **if** i == 34:

#### break



*#conver label to categorical in stead of numerical value to use feature of image generator*

df['category'] = df['category'].replace({1:'jeans',0:'saree',2:'trouser'}) df.head(10)

|  |  |  |
| --- | --- | --- |
|  | image | category |
| 0 | saree.0.jpg | saree |
| 1 | saree.2.jpg | saree |
| 2 | saree.1.jpg | saree |
| 3 | saree.10.jpg | saree |
| 4 | saree.3.jpg | saree |
| 5 | saree.6.jpg | saree |
| 6 | saree.8.jpg | saree |
| 7 | saree.9.jpg | saree |
| 8 | saree.4.jpg | saree |
| 9 | saree.7.jpg | saree |

df['category'].value\_counts()

saree 200

jeans 200

trouser 200

Name: category, dtype: int64

# Data Preperation

### Data speliting

1. re index to spleated data to order data

### get shape of data

*#splid data to train and test*

x\_train,x\_val =

train\_test\_split(df ,random\_state=42,shuffle=True,test\_size=0.1) x\_train['category'].value\_counts()

jeans 187

trouser 180

saree 173

Name: category, dtype: int64 x\_val['category'].value\_counts()

saree 27

trouser 20

jeans 13

Name: category, dtype: int64

*# reindex Data Frame*

x\_train = x\_train.reset\_index(drop=True) x\_val = x\_val.reset\_index(drop=True)

print('Train Images shape is : ',x\_train.shape) print('Validation Images shape is : ',x\_val.shape)

Train Images shape is : (540, 2) Validation Images shape is : (60, 2)

# Buliding CNN Model

## i will use Data augmentation,as it is a set of techniques to artificially increase the amount of data by generating new data points from existing data. This includes making small changes to data or using deep learning models to generate new data points.

batch\_size = 64

*#create image generator for images*

image\_gen = ImageDataGenerator(

rescale = 1./255, shear\_range = 0.2,

zoom\_range = 0.5, height\_shift\_range=0.2, width\_shift\_range=0.2, fill\_mode='nearest',

horizontal\_flip=True, rotation\_range = 20,

)

*# Create Image Data Generator for Test/Validation Set*

test\_data\_gen = ImageDataGenerator(

rescale = 1./255, shear\_range = 0.2,

zoom\_range = 0.5, height\_shift\_range=0.2, width\_shift\_range=0.2, fill\_mode='nearest',

horizontal\_flip=True, rotation\_range = 20,)

train = image\_gen.flow\_from\_dataframe( x\_train, image\_path+'/'+'train/', x\_col='image', y\_col='category', target\_size=(128,128), class\_mode='categorical', shuffle=True, batch\_size=batch\_size

)

validate = test\_data\_gen.flow\_from\_dataframe( x\_val,

image\_path+'/'+'train/', x\_col='image', y\_col='category', target\_size=(128,128), class\_mode='categorical',

shuffle=True, batch\_size=batch\_size

)

Found 540 validated image filenames belonging to 3 classes. Found 60 validated image filenames belonging to 3 classes.

*# get shape of train data*

**for** train\_img , train\_label **in** train : print('image shape ',train\_img.shape) print('label shape ',train\_label.shape) **break**

image shape (64, 128, 128, 3)

label shape (64, 3)

*# get shape of train data*

**for** v\_img , v\_label **in** validate : print('image shape ',v\_img.shape) print('label shape ',v\_label.shape) **break**

image shape (60, 128, 128, 3)

label shape (60, 3)

*#start bulding CNN Model* cnn\_model = Sequential() cnn\_model = models.Sequential()

cnn\_model.add(layers.Conv2D(64,(3,3),padding ='Same',activation = 'relu',input\_shape=(128,128,3))) cnn\_model.add(layers.MaxPooling2D(2,2)) cnn\_model.add(layers.Conv2D(64,(3,3) ,padding

='same',activation='relu')) cnn\_model.add(layers.MaxPooling2D(2,2)) cnn\_model.add(layers.Conv2D(128,(3,3),padding

='same',activation='relu')) cnn\_model.add(layers.MaxPooling2D(2,2)) cnn\_model.add(layers.Conv2D(128,(3,3) ,padding

='same',activation='relu')) cnn\_model.add(layers.MaxPooling2D(2,2)) cnn\_model.add(layers.Conv2D(256,(3,3) ,padding

='same',activation='relu')) cnn\_model.add(layers.MaxPooling2D(2,2)) cnn\_model.add(BatchNormalization()) cnn\_model.add(layers.Conv2D(256,(3,3) ,padding

='same',activation='relu')) cnn\_model.add(layers.MaxPooling2D(2,2)) cnn\_model.add(BatchNormalization()) cnn\_model.summary()

*# cnn\_model.add(Dropout(0.2))*

Model: "sequential\_3"

Layer (type) Output Shape Param #

=================================================================

|  |  |  |
| --- | --- | --- |
| conv2d\_6 (Conv2D) | (None, 128, 128, 64) | 1792 |
| max\_pooling2d\_6 (MaxPooling 2D) | (None, 64, 64, 64) | 0 |
| conv2d\_7 (Conv2D) | (None, 64, 64, 64) | 36928 |
| max\_pooling2d\_7 (MaxPooling 2D) | (None, 32, 32, 64) | 0 |
| conv2d\_8 (Conv2D) | (None, 32, 32, 128) | 73856 |
| max\_pooling2d\_8 (MaxPooling 2D) | (None, 16, 16, 128) | 0 |
| conv2d\_9 (Conv2D) | (None, 16, 16, 128) | 147584 |
| max\_pooling2d\_9 (MaxPooling 2D) | (None, 8, 8, 128) | 0 |
| conv2d\_10 (Conv2D) | (None, 8, 8, 256) | 295168 |
| max\_pooling2d\_10 (MaxPoolin g2D) | (None, 4, 4, 256) | 0 |
| batch\_normalization\_4 (Batc hNormalization) | (None, 4, 4, 256) | 1024 |
| conv2d\_11 (Conv2D) | (None, 4, 4, 256) | 590080 |
| max\_pooling2d\_11 (MaxPoolin g2D) | (None, 2, 2, 256) | 0 |
| batch\_normalization\_5 (Batc hNormalization) | (None, 2, 2, 256) | 1024 |

=================================================================

Total params: 1,147,456

Trainable params: 1,146,432

Non-trainable params: 1,024

cnn\_model.add(layers.Flatten()) cnn\_model.add(layers.Dense(1024, activation='relu')) cnn\_model.add(BatchNormalization())

*# cnn\_model.add(Dropout(0.7))*

cnn\_model.add(layers.Dense(512, activation='relu')) cnn\_model.add(BatchNormalization())

*# cnn\_model.add(Dropout(0.3))* cnn\_model.add(layers.Dense(3, activation ='softmax')) cnn\_model.summary()

Model: "sequential\_3"

Layer (type) Output Shape Param #

=================================================================

|  |  |  |  |
| --- | --- | --- | --- |
| conv2d\_6 (Conv2D) | (None, 128, 128, 64) | | 1792 |
| max\_pooling2d\_6 (MaxPooling 2D) | (None, 64, 64, 64) | | 0 |
| conv2d\_7 (Conv2D) | (None, 64, 64, 64) | | 36928 |
| max\_pooling2d\_7 (MaxPooling 2D) | (None, 32, 32, 64) | | 0 |
| conv2d\_8 (Conv2D) | (None, 32, 32, 128) | | 73856 |
| max\_pooling2d\_8 (MaxPooling 2D) | (None, 16, 16, 128) | | 0 |
| conv2d\_9 (Conv2D) | (None, 16, 16, 128) | | 147584 |
| max\_pooling2d\_9 (MaxPooling 2D) | (None, 8, 8, 128) | | 0 |
| conv2d\_10 (Conv2D) | (None, 8, 8, 256) | | 295168 |
| max\_pooling2d\_10 (MaxPoolin g2D) | (None, 4, 4, 256) | | 0 |
| batch\_normalization\_4 (Batc hNormalization) | (None, 4, 4, 256) | | 1024 |
| conv2d\_11 (Conv2D) | (None, 4, 4, 256) | | 590080 |
| max\_pooling2d\_11 (MaxPoolin (None, 2, 2, g2D) | | 256) | 0 |
| batch\_normalization\_5 (Batc (None, 2, 2, | | 256) | 1024 |
| hNormalization) |  |  | |
| flatten\_1 (Flatten) | (None, 1024) | 0 | |
| dense\_3 (Dense) | (None, 1024) | 1049600 | |

batch\_normalization\_6 (Batc (None, 1024) 4096

hNormalization)

dense\_4 (Dense) (None, 512) 524800

batch\_normalization\_7 (Batc (None, 512) 2048

hNormalization)

dense\_5 (Dense) (None, 3) 1539

=================================================================

Total params: 2,729,539

Trainable params: 2,725,443

Non-trainable params: 4,096

cnn\_model.compile(optimizer='adam',loss='categorical\_crossentropy',met rics=['accuracy'],)

## Defining Callbacks

* + A callback is an object that can perform actions at various stages of training (e.g. at the start or end of an epoch, before or after a single batch, etc)

## Reduce Learning Rate on Plateau

### Is used to reduce the learning rate when a metric has stopped improving.

from tensorflow.keras.callbacks import EarlyStopping,ReduceLROnPlateau early = EarlyStopping(monitor="loss", mode="min",min\_delta = 0,

patience = 10,

verbose = 1, restore\_best\_weights = True)

learning\_rate\_reduction = ReduceLROnPlateau(monitor='loss', patience = 2, verbose=1,factor=0.3, min\_lr=0.000001)

callbacks\_list = [ early, learning\_rate\_reduction]

*# Training model*

n\_training\_samples = len(train) n\_validation\_samples = len(validate) history = cnn\_model.fit(

train, epochs=60,

validation\_data=validate, validation\_steps=n\_validation\_samples//batch\_size, *# steps\_per\_epoch =n\_training\_samples//batch\_size,* shuffle = True,

callbacks=callbacks\_list

)

Epoch 1/60

9/9 [==============================] - 34s 4s/step - loss: 1.2073 -

accuracy: 0.6444 - lr: 0.0010 Epoch 2/60

9/9 [==============================] - 29s 3s/step - loss: 0.5689 -

accuracy: 0.7741 - lr: 0.0010 Epoch 3/60

9/9 [==============================] - 29s 3s/step - loss: 0.4743 -

accuracy: 0.8204 - lr: 0.0010 Epoch 4/60

9/9 [==============================] - 29s 3s/step - loss: 0.3617 -

accuracy: 0.8593 - lr: 0.0010 Epoch 5/60

9/9 [==============================] - 29s 3s/step - loss: 0.4050 -

accuracy: 0.8463 - lr: 0.0010 Epoch 6/60

9/9 [==============================] - 30s 3s/step - loss: 0.3134 -

accuracy: 0.8944 - lr: 0.0010 Epoch 7/60

9/9 [==============================] - 29s 3s/step - loss: 0.2262 -

accuracy: 0.9000 - lr: 0.0010 Epoch 8/60

9/9 [==============================] - 29s 3s/step - loss: 0.1601 -

accuracy: 0.9370 - lr: 0.0010 Epoch 9/60

9/9 [==============================] - 29s 3s/step - loss: 0.2468 -

accuracy: 0.9111 - lr: 0.0010 Epoch 10/60

9/9 [==============================] - 29s 3s/step - loss: 0.1391 -

accuracy: 0.9556 - lr: 0.0010 Epoch 11/60

9/9 [==============================] - 29s 3s/step - loss: 0.1742 -

accuracy: 0.9352 - lr: 0.0010 Epoch 12/60

9/9 [==============================] - ETA: 0s - loss: 0.1721 -

accuracy: 0.9278

Epoch 12: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.

9/9 [==============================] - 29s 3s/step - loss: 0.1721 -

accuracy: 0.9278 - lr: 0.0010 Epoch 13/60

9/9 [==============================] - 29s 3s/step - loss: 0.1375 -

accuracy: 0.9426 - lr: 3.0000e-04 Epoch 14/60

9/9 [==============================] - 29s 3s/step - loss: 0.0924 -

accuracy: 0.9704 - lr: 3.0000e-04 Epoch 15/60

9/9 [==============================] - 29s 3s/step - loss: 0.0850 -

accuracy: 0.9722 - lr: 3.0000e-04 Epoch 16/60

9/9 [==============================] - 29s 3s/step - loss: 0.0653 -

accuracy: 0.9759 - lr: 3.0000e-04

Epoch 17/60

9/9 [==============================] - 29s 3s/step - loss: 0.0806 -

accuracy: 0.9685 - lr: 3.0000e-04 Epoch 18/60

9/9 [==============================] - ETA: 0s - loss: 0.0961 -

accuracy: 0.9685

Epoch 18: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.

9/9 [==============================] - 29s 3s/step - loss: 0.0961 -

accuracy: 0.9685 - lr: 3.0000e-04 Epoch 19/60

9/9 [==============================] - 29s 3s/step - loss: 0.0679 -

accuracy: 0.9796 - lr: 9.0000e-05 Epoch 20/60

9/9 [==============================] - ETA: 0s - loss: 0.0822 -

accuracy: 0.9722

Epoch 20: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.

9/9 [==============================] - 29s 3s/step - loss: 0.0822 -

accuracy: 0.9722 - lr: 9.0000e-05 Epoch 21/60

9/9 [==============================] - 29s 3s/step - loss: 0.0487 -

accuracy: 0.9796 - lr: 2.7000e-05 Epoch 22/60

9/9 [==============================] - 29s 3s/step - loss: 0.0478 -

accuracy: 0.9833 - lr: 2.7000e-05 Epoch 23/60

9/9 [==============================] - 29s 3s/step - loss: 0.0447 -

accuracy: 0.9889 - lr: 2.7000e-05 Epoch 24/60

9/9 [==============================] - 28s 3s/step - loss: 0.0527 -

accuracy: 0.9796 - lr: 2.7000e-05 Epoch 25/60

9/9 [==============================] - ETA: 0s - loss: 0.0519 -

accuracy: 0.9778

Epoch 25: ReduceLROnPlateau reducing learning rate to 8.100000013655517e-06.

9/9 [==============================] - 29s 3s/step - loss: 0.0519 -

accuracy: 0.9778 - lr: 2.7000e-05 Epoch 26/60

9/9 [==============================] - 29s 3s/step - loss: 0.0357 -

accuracy: 0.9944 - lr: 8.1000e-06 Epoch 27/60

9/9 [==============================] - 29s 3s/step - loss: 0.0872 -

accuracy: 0.9796 - lr: 8.1000e-06 Epoch 28/60

9/9 [==============================] - ETA: 0s - loss: 0.0449 -

accuracy: 0.9870

Epoch 28: ReduceLROnPlateau reducing learning rate to 2.429999949526973e-06.

9/9 [==============================] - 29s 3s/step - loss: 0.0449 -

accuracy: 0.9870 - lr: 8.1000e-06 Epoch 29/60

9/9 [==============================] - 29s 3s/step - loss: 0.0402 -

accuracy: 0.9833 - lr: 2.4300e-06 Epoch 30/60

9/9 [==============================] - ETA: 0s - loss: 0.0666 -

accuracy: 0.9815

Epoch 30: ReduceLROnPlateau reducing learning rate to 1e-06.

9/9 [==============================] - 29s 3s/step - loss: 0.0666 -

accuracy: 0.9815 - lr: 2.4300e-06 Epoch 31/60

9/9 [==============================] - 29s 3s/step - loss: 0.0281 -

accuracy: 0.9926 - lr: 1.0000e-06 Epoch 32/60

9/9 [==============================] - 29s 3s/step - loss: 0.0496 -

accuracy: 0.9852 - lr: 1.0000e-06 Epoch 33/60

9/9 [==============================] - 29s 3s/step - loss: 0.0368 -

accuracy: 0.9889 - lr: 1.0000e-06 Epoch 34/60

9/9 [==============================] - 29s 3s/step - loss: 0.0380 -

accuracy: 0.9926 - lr: 1.0000e-06 Epoch 35/60

9/9 [==============================] - 29s 3s/step - loss: 0.0413 -

accuracy: 0.9870 - lr: 1.0000e-06 Epoch 36/60

9/9 [==============================] - 29s 3s/step - loss: 0.0515 -

accuracy: 0.9852 - lr: 1.0000e-06 Epoch 37/60

9/9 [==============================] - 29s 3s/step - loss: 0.0316 -

accuracy: 0.9926 - lr: 1.0000e-06 Epoch 38/60

9/9 [==============================] - 29s 3s/step - loss: 0.0501 -

accuracy: 0.9796 - lr: 1.0000e-06 Epoch 39/60

9/9 [==============================] - 29s 3s/step - loss: 0.0683 -

accuracy: 0.9833 - lr: 1.0000e-06 Epoch 40/60

9/9 [==============================] - 33s 3s/step - loss: 0.0488 -

accuracy: 0.9815 - lr: 1.0000e-06 Epoch 41/60

9/9 [==============================] - ETA: 0s - loss: 0.0540 -

accuracy: 0.9778Restoring model weights from the end of the best epoch: 31.

9/9 [==============================] - 29s 3s/step - loss: 0.0540 -

accuracy: 0.9778 - lr: 1.0000e-06 Epoch 41: early stopping

score, acc = cnn\_model.evaluate(validate,batch\_size=batch\_size)

print('Test score:', score) print('Test accuracy:', acc)

1/1 [==============================] - 1s 1s/step - loss: 0.7665 -

accuracy: 0.5833

Test score: 0.766460657119751

Test accuracy: 0.5833333134651184

cnn\_prediction = cnn\_model.predict(validate) cnn\_prediction

array([[6.0740672e-03, 4.3969736e-01, 5.5422860e-01], [4.7305265e-01, 3.0753916e-04, 5.2663976e-01], [9.9928409e-01, 3.8402650e-05, 6.7763252e-04], [7.4064070e-01, 1.6429554e-01, 9.5063761e-02], [9.9933052e-01, 1.6761982e-05, 6.5274647e-04], [5.5768162e-01, 2.5943507e-04, 4.4205898e-01], [1.1244741e-03, 8.4340709e-01, 1.5546840e-01], [2.2699898e-02, 5.5012858e-01, 4.2717159e-01], [3.2665182e-03, 9.4743294e-01, 4.9300544e-02], [9.6781865e-02, 3.0162296e-04, 9.0291649e-01], [2.9386247e-03, 1.8956615e-01, 8.0749518e-01], [3.8221413e-01, 6.1053562e-01, 7.2502135e-03], [3.6263054e-03, 4.8065420e-02, 9.4830829e-01], [6.7344820e-01, 2.9335389e-04, 3.2625845e-01], [1.3854073e-01, 4.9493503e-04, 8.6096436e-01], [1.8495403e-02, 7.5790805e-01, 2.2359657e-01], [2.9311559e-01, 1.2287203e-03, 7.0565563e-01], [9.9999177e-01, 6.0371722e-06, 2.1204394e-06], [3.0496912e-03, 4.5857016e-02, 9.5109332e-01], [2.0478843e-03, 8.7778315e-02, 9.1017383e-01], [1.5935677e-01, 6.0889709e-01, 2.3174617e-01], [2.4250823e-03, 3.2274777e-01, 6.7482710e-01], [9.9825639e-01, 6.9058486e-05, 1.6745300e-03], [6.7467064e-01, 4.6162342e-04, 3.2486770e-01], [4.4254243e-01, 1.2800260e-04, 5.5732954e-01], [4.2883912e-01, 1.4337914e-01, 4.2778170e-01], [2.2990194e-03, 4.7549218e-01, 5.2220881e-01], [1.4494804e-01, 3.3754350e-03, 8.5167646e-01], [6.0540473e-01, 4.6156641e-04, 3.9413369e-01], [8.4831873e-03, 2.9771195e-03, 9.8853964e-01], [9.9953663e-01, 2.0721576e-05, 4.4270387e-04], [7.8166276e-01, 7.4751005e-02, 1.4358619e-01], [3.9737341e-03, 1.2506818e-03, 9.9477553e-01], [1.0304729e-01, 6.8670532e-05, 8.9688396e-01], [2.4468587e-03, 4.7023818e-01, 5.2731496e-01], [6.2474656e-01, 1.9672934e-04, 3.7505674e-01], [7.2963673e-01, 2.1797017e-04, 2.7014533e-01], [3.4616049e-02, 1.1217857e-01, 8.5320538e-01],

[9.9999213e-01, 5.1779721e-06, 2.6852465e-06], [7.1206033e-01, 2.8794897e-03, 2.8506020e-01], [1.7661953e-02, 5.4212469e-01, 4.4021326e-01], [9.9995399e-01, 1.4710591e-05, 3.1391424e-05], [5.4799408e-01, 2.5055784e-01, 2.0144817e-01], [9.8033798e-01, 1.1842236e-03, 1.8477876e-02], [8.3318172e-04, 8.4269887e-01, 1.5646794e-01], [4.5199049e-01, 5.2081835e-03, 5.4280126e-01], [9.9931014e-01, 3.3473730e-04, 3.5509953e-04], [2.6206675e-01, 4.8412481e-01, 2.5380847e-01], [5.1502228e-01, 5.2471957e-03, 4.7973043e-01], [9.9299532e-01, 2.7947510e-03, 4.2099394e-03], [9.9924880e-01, 2.5154385e-04, 4.9963343e-04], [2.4061392e-01, 7.0892781e-01, 5.0458238e-02], [8.3407557e-01, 6.4706593e-04, 1.6527739e-01], [2.6968198e-03, 3.2859525e-01, 6.6870791e-01], [3.6481342e-01, 2.1256846e-01, 4.2261812e-01], [9.9257267e-01, 7.2424468e-06, 7.4200686e-03], [2.1000134e-03, 3.5732456e-02, 9.6216756e-01], [2.2793360e-01, 1.2578903e-03, 7.7080852e-01], [5.3928500e-01, 2.6196602e-01, 1.9874898e-01],

[8.1827445e-04, 7.9394680e-01, 2.0523483e-01]], dtype=float32)

plt.plot(history.history['accuracy'], label='accuracy') plt.plot(history.history['val\_accuracy'], label = 'val\_accuracy') plt.xlabel('Epoch')

plt.ylabel('Accuracy') plt.ylim([0.5, 1]) plt.legend(loc='lower right') plt.show()

KeyError Traceback (most recent call last)

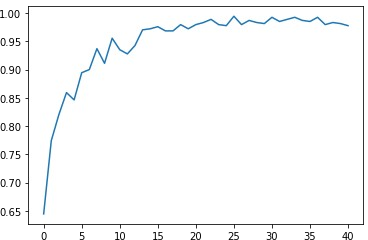
<ipython-input-44-f4ff4840f5bd> in <module>()

1 plt.plot(history.history['accuracy'], label='accuracy')

----> 2 plt.plot(history.history['val\_accuracy'], label = 'val\_accuracy')

1. plt.xlabel('Epoch')
2. plt.ylabel('Accuracy') 5 plt.ylim([0.5, 1])

KeyError: 'val\_accuracy'



# prepare data for predection.

*#get image path*

test\_images = os.listdir(image\_path+'/'+'test')

*#creat data frame*

df\_test =pd.DataFrame({ 'image': test\_images,

})

df\_test.head()

*#prepare generator*

val\_data\_gen = ImageDataGenerator( rescale = 1./255, )

testing = val\_data\_gen.flow\_from\_dataframe( df\_test,

image\_path+'/'+'test/', x\_col='image',

y\_col= None, target\_size=(128,128), class\_mode=None, shuffle=True, batch\_size=batch\_size

)

Found 300 validated image filenames.

# Predction Using CNN.

predection = cnn\_model.predict(testing) predection

array([[9.9828088e-01, 2.0129165e-04, 1.5178961e-03], [2.4121670e-01, 7.1161896e-01, 4.7164325e-02], [9.9995387e-01, 1.5913571e-05, 3.0276049e-05], [4.2148307e-01, 2.3551409e-04, 5.7828134e-01], [1.9690374e-01, 4.6537718e-01, 3.3771905e-01], [3.6556481e-03, 3.8270804e-03, 9.9251723e-01], [5.0667960e-02, 9.9274970e-05, 9.4923276e-01], [9.3969893e-01, 4.0891377e-05, 6.0260154e-02], [8.0193632e-04, 9.9667102e-01, 2.5270446e-03], [6.8162203e-01, 2.9663473e-01, 2.1743212e-02], [9.9515659e-01, 6.4511078e-06, 4.8370031e-03], [2.9665694e-01, 4.0969037e-04, 7.0293331e-01], [2.3660478e-03, 2.2710459e-01, 7.7052933e-01], [2.3359541e-02, 2.7761620e-04, 9.7636288e-01], [6.8452051e-03, 4.9811995e-01, 4.9503487e-01], [1.3979766e-01, 1.2548368e-03, 8.5894746e-01], [3.4633341e-01, 5.9526461e-01, 5.8401983e-02], [9.8789740e-01, 5.9893867e-03, 6.1133089e-03], [9.9912924e-01, 2.5582558e-04, 6.1493804e-04], [9.9372572e-01, 2.3019368e-04, 6.0440870e-03], [9.9977642e-01, 1.8566201e-04, 3.7878857e-05], [8.7758362e-01, 2.7597805e-03, 1.1965655e-01], [8.0193632e-04, 9.9667102e-01, 2.5270446e-03], [2.1991836e-02, 6.5768582e-01, 3.2032239e-01], [9.2812634e-01, 5.1498733e-02, 2.0374915e-02], [1.6488686e-03, 4.5081678e-01, 5.4753441e-01], [2.4121670e-01, 7.1161896e-01, 4.7164325e-02], [9.9995434e-01, 3.8402581e-05, 7.2279381e-06], [9.9914062e-01, 5.2614265e-05, 8.0666086e-04], [9.8789740e-01, 5.9893867e-03, 6.1133089e-03], [1.7117316e-02, 7.5299782e-01, 2.2988485e-01], [9.2812634e-01, 5.1498733e-02, 2.0374915e-02], [2.8870048e-02, 3.4802771e-01, 6.2310231e-01], [9.3969893e-01, 4.0891377e-05, 6.0260154e-02], [4.9046475e-01, 5.8280525e-04, 5.0895244e-01], [9.9977642e-01, 1.8566201e-04, 3.7878857e-05], [9.3268780e-03, 1.7129911e-04, 9.9050176e-01], [7.4946320e-01, 2.8282314e-04, 2.5025392e-01], [8.7758362e-01, 2.7597805e-03, 1.1965655e-01], [9.9828088e-01, 2.0129165e-04, 1.5178961e-03], [4.7587799e-03, 9.6232802e-02, 8.9900839e-01], [9.9976712e-01, 4.3191627e-05, 1.8972480e-04], [9.9992836e-01, 4.9762970e-05, 2.1956963e-05], [9.2946231e-02, 5.8283787e-03, 9.0122545e-01], [2.3660478e-03, 2.2710459e-01, 7.7052933e-01], [9.8789740e-01, 5.9893867e-03, 6.1133089e-03],

[3.6556481e-03, 3.8270804e-03, 9.9251723e-01], [9.9977571e-01, 1.6301816e-05, 2.0794869e-04], [2.4121670e-01, 7.1161896e-01, 4.7164325e-02], [9.6974462e-01, 2.0089166e-03, 2.8246393e-02], [9.9998128e-01, 1.0557359e-05, 8.1413300e-06], [8.7758362e-01, 2.7597805e-03, 1.1965655e-01], [9.9998128e-01, 1.0557359e-05, 8.1413300e-06], [1.3979766e-01, 1.2548368e-03, 8.5894746e-01], [4.2148307e-01, 2.3551409e-04, 5.7828134e-01], [9.9976712e-01, 4.3191627e-05, 1.8972480e-04], [9.9972755e-01, 2.7142967e-05, 2.4521991e-04], [3.2025689e-01, 1.0829883e-03, 6.7866015e-01], [5.2731745e-02, 1.0332831e-03, 9.4623500e-01], [1.8305203e-01, 3.6701714e-04, 8.1658095e-01], [9.6974462e-01, 2.0089166e-03, 2.8246393e-02], [9.9515659e-01, 6.4511078e-06, 4.8370031e-03], [3.2025689e-01, 1.0829883e-03, 6.7866015e-01], [1.3979766e-01, 1.2548368e-03, 8.5894746e-01], [9.9995387e-01, 1.5913571e-05, 3.0276049e-05], [7.3360247e-03, 5.3420341e-01, 4.5846057e-01], [4.4167370e-01, 2.4710665e-04, 5.5807912e-01], [5.2731745e-02, 1.0332831e-03, 9.4623500e-01], [9.3969893e-01, 4.0891377e-05, 6.0260154e-02], [9.9372572e-01, 2.3019368e-04, 6.0440870e-03], [2.1991836e-02, 6.5768582e-01, 3.2032239e-01], [4.8109978e-01, 3.0605334e-01, 2.1284688e-01], [2.3660478e-03, 2.2710459e-01, 7.7052933e-01], [2.3660478e-03, 2.2710459e-01, 7.7052933e-01], [1.7754199e-01, 8.2154214e-01, 9.1590441e-04], [9.2946231e-02, 5.8283787e-03, 9.0122545e-01], [2.9665694e-01, 4.0969037e-04, 7.0293331e-01], [4.7587799e-03, 9.6232802e-02, 8.9900839e-01], [9.9948776e-01, 2.6086860e-04, 2.5135145e-04], [4.4167370e-01, 2.4710665e-04, 5.5807912e-01], [9.9997878e-01, 1.3199990e-05, 7.9596593e-06], [8.7758362e-01, 2.7597805e-03, 1.1965655e-01], [8.0193632e-04, 9.9667102e-01, 2.5270446e-03], [9.9997878e-01, 1.3199990e-05, 7.9596593e-06], [9.9988723e-01, 2.6010881e-05, 8.6722292e-05], [1.7754199e-01, 8.2154214e-01, 9.1590441e-04], [4.2148307e-01, 2.3551409e-04, 5.7828134e-01], [6.8452051e-03, 4.9811995e-01, 4.9503487e-01], [2.1991836e-02, 6.5768582e-01, 3.2032239e-01], [7.3360247e-03, 5.3420341e-01, 4.5846057e-01], [1.8124431e-01, 1.3581369e-04, 8.1861985e-01], [1.3979766e-01, 1.2548368e-03, 8.5894746e-01], [2.3359541e-02, 2.7761620e-04, 9.7636288e-01], [5.0667960e-02, 9.9274970e-05, 9.4923276e-01], [4.8109978e-01, 3.0605334e-01, 2.1284688e-01], [9.9972755e-01, 2.7142967e-05, 2.4521991e-04],

[1.8305203e-01, 3.6701714e-04, 8.1658095e-01], [8.0193632e-04, 9.9667102e-01, 2.5270446e-03], [9.6092962e-02, 3.9883092e-04, 9.0350825e-01], [9.9996245e-01, 2.0008356e-06, 3.5515037e-05], [9.9977571e-01, 1.6301816e-05, 2.0794869e-04], [9.9996245e-01, 2.0008356e-06, 3.5515037e-05], [9.9914062e-01, 5.2614265e-05, 8.0666086e-04], [9.6092962e-02, 3.9883092e-04, 9.0350825e-01], [9.9995387e-01, 1.5913571e-05, 3.0276049e-05], [9.3268780e-03, 1.7129911e-04, 9.9050176e-01], [1.7754199e-01, 8.2154214e-01, 9.1590441e-04], [9.9988723e-01, 2.6010881e-05, 8.6722292e-05], [3.1707939e-03, 9.9211520e-01, 4.7139917e-03], [9.9948776e-01, 2.6086860e-04, 2.5135145e-04], [9.9995387e-01, 1.5913571e-05, 3.0276049e-05], [9.9976712e-01, 4.3191627e-05, 1.8972480e-04], [2.3359541e-02, 2.7761620e-04, 9.7636288e-01], [7.4946320e-01, 2.8282314e-04, 2.5025392e-01], [9.9996245e-01, 2.0008356e-06, 3.5515037e-05], [7.1024038e-03, 7.2984193e-03, 9.8559916e-01], [9.9992836e-01, 4.9762970e-05, 2.1956963e-05], [3.4633341e-01, 5.9526461e-01, 5.8401983e-02], [9.9996245e-01, 2.0008356e-06, 3.5515037e-05], [9.9998164e-01, 7.6255260e-06, 1.0673826e-05], [4.9046475e-01, 5.8280525e-04, 5.0895244e-01], [1.8124431e-01, 1.3581369e-04, 8.1861985e-01], [9.9948776e-01, 2.6086860e-04, 2.5135145e-04], [7.3360247e-03, 5.3420341e-01, 4.5846057e-01], [3.4633341e-01, 5.9526461e-01, 5.8401983e-02], [1.8305203e-01, 3.6701714e-04, 8.1658095e-01], [2.8870048e-02, 3.4802771e-01, 6.2310231e-01], [7.1024038e-03, 7.2984193e-03, 9.8559916e-01], [1.8124431e-01, 1.3581369e-04, 8.1861985e-01], [9.9912924e-01, 2.5582558e-04, 6.1493804e-04], [9.9972755e-01, 2.7142967e-05, 2.4521991e-04], [9.9998128e-01, 1.0557359e-05, 8.1413300e-06], [3.6556481e-03, 3.8270804e-03, 9.9251723e-01], [1.8124431e-01, 1.3581369e-04, 8.1861985e-01], [2.6272081e-02, 8.3888775e-01, 1.3484021e-01], [2.1991836e-02, 6.5768582e-01, 3.2032239e-01], [9.9976712e-01, 4.3191627e-05, 1.8972480e-04], [2.3660478e-03, 2.2710459e-01, 7.7052933e-01], [1.8855730e-01, 2.1288090e-03, 8.0931389e-01], [2.3359541e-02, 2.7761620e-04, 9.7636288e-01], [4.8109978e-01, 3.0605334e-01, 2.1284688e-01], [7.4946320e-01, 2.8282314e-04, 2.5025392e-01], [9.9828088e-01, 2.0129165e-04, 1.5178961e-03], [9.9972755e-01, 2.7142967e-05, 2.4521991e-04], [9.6974462e-01, 2.0089166e-03, 2.8246393e-02], [1.9690374e-01, 4.6537718e-01, 3.3771905e-01],

[9.9997878e-01, 1.3199990e-05, 7.9596593e-06], [9.8789740e-01, 5.9893867e-03, 6.1133089e-03], [9.9995434e-01, 3.8402581e-05, 7.2279381e-06], [9.6092962e-02, 3.9883092e-04, 9.0350825e-01], [9.9914062e-01, 5.2614265e-05, 8.0666086e-04], [4.2148307e-01, 2.3551409e-04, 5.7828134e-01], [1.9690374e-01, 4.6537718e-01, 3.3771905e-01], [4.4167370e-01, 2.4710665e-04, 5.5807912e-01], [9.9998164e-01, 7.6255260e-06, 1.0673826e-05], [6.8162203e-01, 2.9663473e-01, 2.1743212e-02], [9.9992836e-01, 4.9762970e-05, 2.1956963e-05], [4.8109978e-01, 3.0605334e-01, 2.1284688e-01], [9.2946231e-02, 5.8283787e-03, 9.0122545e-01], [9.9977642e-01, 1.8566201e-04, 3.7878857e-05], [3.4633341e-01, 5.9526461e-01, 5.8401983e-02], [4.1224059e-01, 3.8392053e-03, 5.8392018e-01], [9.9988723e-01, 2.6010881e-05, 8.6722292e-05], [3.1102851e-01, 6.7663401e-01, 1.2337525e-02], [9.9992836e-01, 4.9762970e-05, 2.1956963e-05], [1.7117316e-02, 7.5299782e-01, 2.2988485e-01], [9.9988723e-01, 2.6010881e-05, 8.6722292e-05], [8.7758362e-01, 2.7597805e-03, 1.1965655e-01], [9.3969893e-01, 4.0891377e-05, 6.0260154e-02], [9.9997878e-01, 1.3199990e-05, 7.9596593e-06], [1.7117316e-02, 7.5299782e-01, 2.2988485e-01], [6.8452051e-03, 4.9811995e-01, 4.9503487e-01], [4.4167370e-01, 2.4710665e-04, 5.5807912e-01], [7.4946320e-01, 2.8282314e-04, 2.5025392e-01], [4.7587799e-03, 9.6232802e-02, 8.9900839e-01], [9.9977571e-01, 1.6301816e-05, 2.0794869e-04], [4.9046475e-01, 5.8280525e-04, 5.0895244e-01], [4.7587799e-03, 9.6232802e-02, 8.9900839e-01], [9.9948776e-01, 2.6086860e-04, 2.5135145e-04], [7.1024038e-03, 7.2984193e-03, 9.8559916e-01], [2.6272081e-02, 8.3888775e-01, 1.3484021e-01], [4.1224059e-01, 3.8392053e-03, 5.8392018e-01], [9.9998128e-01, 1.0557359e-05, 8.1413300e-06], [9.9995434e-01, 3.8402581e-05, 7.2279381e-06], [9.2812634e-01, 5.1498733e-02, 2.0374915e-02], [9.3969893e-01, 4.0891377e-05, 6.0260154e-02], [1.7754199e-01, 8.2154214e-01, 9.1590441e-04], [7.4946320e-01, 2.8282314e-04, 2.5025392e-01], [1.7754199e-01, 8.2154214e-01, 9.1590441e-04], [4.1224059e-01, 3.8392053e-03, 5.8392018e-01], [9.9828088e-01, 2.0129165e-04, 1.5178961e-03], [9.8789740e-01, 5.9893867e-03, 6.1133089e-03], [2.8870048e-02, 3.4802771e-01, 6.2310231e-01], [2.6272081e-02, 8.3888775e-01, 1.3484021e-01], [1.8855730e-01, 2.1288090e-03, 8.0931389e-01], [3.2025689e-01, 1.0829883e-03, 6.7866015e-01],

[1.8124431e-01, 1.3581369e-04, 8.1861985e-01], [1.3979766e-01, 1.2548368e-03, 8.5894746e-01], [3.1102851e-01, 6.7663401e-01, 1.2337525e-02], [9.9995434e-01, 3.8402581e-05, 7.2279381e-06], [6.8452051e-03, 4.9811995e-01, 4.9503487e-01], [4.7587799e-03, 9.6232802e-02, 8.9900839e-01], [8.0193632e-04, 9.9667102e-01, 2.5270446e-03], [9.9912924e-01, 2.5582558e-04, 6.1493804e-04], [9.9998128e-01, 1.0557359e-05, 8.1413300e-06], [9.9995387e-01, 1.5913571e-05, 3.0276049e-05], [3.1707939e-03, 9.9211520e-01, 4.7139917e-03], [2.6272081e-02, 8.3888775e-01, 1.3484021e-01], [3.1707939e-03, 9.9211520e-01, 4.7139917e-03], [7.3360247e-03, 5.3420341e-01, 4.5846057e-01], [9.9515659e-01, 6.4511078e-06, 4.8370031e-03], [9.3268780e-03, 1.7129911e-04, 9.9050176e-01], [9.9977571e-01, 1.6301816e-05, 2.0794869e-04], [9.3268780e-03, 1.7129911e-04, 9.9050176e-01], [3.1102851e-01, 6.7663401e-01, 1.2337525e-02], [5.2731745e-02, 1.0332831e-03, 9.4623500e-01], [9.9998164e-01, 7.6255260e-06, 1.0673826e-05], [2.8870048e-02, 3.4802771e-01, 6.2310231e-01], [3.6556481e-03, 3.8270804e-03, 9.9251723e-01], [9.9914062e-01, 5.2614265e-05, 8.0666086e-04], [3.6556481e-03, 3.8270804e-03, 9.9251723e-01], [9.2946231e-02, 5.8283787e-03, 9.0122545e-01], [3.2025689e-01, 1.0829883e-03, 6.7866015e-01], [2.6272081e-02, 8.3888775e-01, 1.3484021e-01], [1.6488686e-03, 4.5081678e-01, 5.4753441e-01], [7.1024038e-03, 7.2984193e-03, 9.8559916e-01], [7.1024038e-03, 7.2984193e-03, 9.8559916e-01], [9.9977642e-01, 1.8566201e-04, 3.7878857e-05], [1.8855730e-01, 2.1288090e-03, 8.0931389e-01], [9.9998164e-01, 7.6255260e-06, 1.0673826e-05], [7.3360247e-03, 5.3420341e-01, 4.5846057e-01], [2.9665694e-01, 4.0969037e-04, 7.0293331e-01], [3.2025689e-01, 1.0829883e-03, 6.7866015e-01], [6.8162203e-01, 2.9663473e-01, 2.1743212e-02], [2.9665694e-01, 4.0969037e-04, 7.0293331e-01], [9.9988723e-01, 2.6010881e-05, 8.6722292e-05], [4.4167370e-01, 2.4710665e-04, 5.5807912e-01], [2.4121670e-01, 7.1161896e-01, 4.7164325e-02], [4.1224059e-01, 3.8392053e-03, 5.8392018e-01], [9.9976712e-01, 4.3191627e-05, 1.8972480e-04], [6.8452051e-03, 4.9811995e-01, 4.9503487e-01], [1.8855730e-01, 2.1288090e-03, 8.0931389e-01], [9.9372572e-01, 2.3019368e-04, 6.0440870e-03], [9.9998164e-01, 7.6255260e-06, 1.0673826e-05], [5.0667960e-02, 9.9274970e-05, 9.4923276e-01], [9.2812634e-01, 5.1498733e-02, 2.0374915e-02],

[9.9977642e-01, 1.8566201e-04, 3.7878857e-05], [9.6092962e-02, 3.9883092e-04, 9.0350825e-01], [5.0667960e-02, 9.9274970e-05, 9.4923276e-01], [3.1102851e-01, 6.7663401e-01, 1.2337525e-02], [3.1707939e-03, 9.9211520e-01, 4.7139917e-03], [9.9828088e-01, 2.0129165e-04, 1.5178961e-03], [6.8162203e-01, 2.9663473e-01, 2.1743212e-02], [1.9690374e-01, 4.6537718e-01, 3.3771905e-01], [5.2731745e-02, 1.0332831e-03, 9.4623500e-01], [9.9912924e-01, 2.5582558e-04, 6.1493804e-04], [1.8305203e-01, 3.6701714e-04, 8.1658095e-01], [1.6488686e-03, 4.5081678e-01, 5.4753441e-01], [4.1224059e-01, 3.8392053e-03, 5.8392018e-01], [9.6974462e-01, 2.0089166e-03, 2.8246393e-02], [1.6488686e-03, 4.5081678e-01, 5.4753441e-01], [2.9665694e-01, 4.0969037e-04, 7.0293331e-01], [1.7117316e-02, 7.5299782e-01, 2.2988485e-01], [9.9997878e-01, 1.3199990e-05, 7.9596593e-06], [4.2148307e-01, 2.3551409e-04, 5.7828134e-01], [9.9372572e-01, 2.3019368e-04, 6.0440870e-03], [4.8109978e-01, 3.0605334e-01, 2.1284688e-01], [1.7117316e-02, 7.5299782e-01, 2.2988485e-01], [5.0667960e-02, 9.9274970e-05, 9.4923276e-01], [9.6092962e-02, 3.9883092e-04, 9.0350825e-01], [5.2731745e-02, 1.0332831e-03, 9.4623500e-01], [3.1102851e-01, 6.7663401e-01, 1.2337525e-02], [9.3268780e-03, 1.7129911e-04, 9.9050176e-01], [9.9515659e-01, 6.4511078e-06, 4.8370031e-03], [9.9914062e-01, 5.2614265e-05, 8.0666086e-04], [9.9912924e-01, 2.5582558e-04, 6.1493804e-04], [1.6488686e-03, 4.5081678e-01, 5.4753441e-01], [2.1991836e-02, 6.5768582e-01, 3.2032239e-01], [6.8162203e-01, 2.9663473e-01, 2.1743212e-02], [3.4633341e-01, 5.9526461e-01, 5.8401983e-02], [9.9515659e-01, 6.4511078e-06, 4.8370031e-03], [9.2812634e-01, 5.1498733e-02, 2.0374915e-02], [9.2946231e-02, 5.8283787e-03, 9.0122545e-01], [9.9372572e-01, 2.3019368e-04, 6.0440870e-03], [9.9996245e-01, 2.0008356e-06, 3.5515037e-05], [1.8855730e-01, 2.1288090e-03, 8.0931389e-01], [3.1707939e-03, 9.9211520e-01, 4.7139917e-03], [9.9972755e-01, 2.7142967e-05, 2.4521991e-04], [9.9948776e-01, 2.6086860e-04, 2.5135145e-04], [1.9690374e-01, 4.6537718e-01, 3.3771905e-01], [9.6974462e-01, 2.0089166e-03, 2.8246393e-02], [2.4121670e-01, 7.1161896e-01, 4.7164325e-02], [9.9992836e-01, 4.9762970e-05, 2.1956963e-05], [2.8870048e-02, 3.4802771e-01, 6.2310231e-01], [2.3359541e-02, 2.7761620e-04, 9.7636288e-01], [4.9046475e-01, 5.8280525e-04, 5.0895244e-01],

[9.9977571e-01, 1.6301816e-05, 2.0794869e-04], [4.9046475e-01, 5.8280525e-04, 5.0895244e-01], [1.8305206e-01, 3.6701711e-04, 8.1658089e-01],

[9.9995434e-01, 3.8402581e-05, 7.2279381e-06]], dtype=float32)

np.argmax(predection,axis=1)[0:100]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| array([0, | 1, | 0, | 2, | 1, | 2, | 2, | 0, | 1, | 0, | 0, | 2, | 2, | 2, | 1, | 2, | 1, | 0, | 0, | 0, | 0, |
| 0, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1, | 1, | 0, | 2, | 1, | 0, | 0, | 0, | 1, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 0, | 0, | 2, | 0, | 0, |
| 2, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2, | 0, | 2, | 0, | 1, | 0, | 0, | 0, | 0, | 2, | 2, | 0, | 0, | 2, | 2, | 2, | 0, | 0, | 2, | 2, | 0, |
| 2, | 2, | 0, | 0, | 1, | 0, | 2, | 2, | 1, | 2, | 2, | 2, | 0, | 2, | 0, | 0, | 1, | 0, | 0, | 1, | 2, |
| 1, | 1, | 2, | 2, | 2, | 2, | 0, | 0, | 2, | 1, | 2, | 0]) |  |  |  |  |  |  |  |  |  |

1,

1,

plot\_predection(cnn\_model)

