SHIVAM MALVIYA

+91 9301596653| shivammalviya205@gmail.com
www.linkedin.com/in/shivammalviya205

EDUCATION

Madhav Institute of Technology and Science ,Gwalior
Gwalior, Madhya Pradesh
Bachelor of Technology in Information Technology |***CGPA:7.78***/10 Aug 2019- May 2023

Sunshine Higher Secondary School, Indore Indore,Madhya Pradesh
Intermediate(Class XII) | **Percentage**:81.2/100 July 2017- May 2018

Vimala Convent Higher Secondary School, Sanawad Sanawad,Madhya Pradesh
AISSE (Class X) | **CGPA** : 9/10 July 2015- May 2016

TECHNICAL SKILLS

Languages: C, **C++**, Javascript , HTML, CSS, SQL

Frameworks: React, Node.js, Express.js

Databases : Mongodb

CS Fundamentals : OOPS, DBMS

PROJECT

InterviewExp : web application for sharing interview experience | Github
- Build a **MERN** Stack application where users can Read and Write about interview experiences of tech giants.
- Included features like users can Upload, Edit and Delete their blogs and other users can leave a comment on it.
- **Technologies Used - ReactJS**, React Hooks, React Router DOM, Axios, NodeJS**,** MongoDB, ExpressJs.
- Implemented various functionalities such as **Login**,Register and also **Filter** blogs by company wise.
- **Live Project Demo link**

SheyCars : web application for booking a car on rent | Github
- Developed a **Full Stack** application for booking a car on rent where authorized users can book a car for required time.
- Included features like Filter based on availability and making payment through **Stripe** Payment.
- Implemented **Admin** panel features like Add new car ,Delete a car and Edit car information.
- Technologies Used - ReactJS, React Hooks, Axios, NodeJS, MongoDB, ExpressJS.
- **Live Project Demo link**

Skip-Ads : Google Chrome Extension | Github
- Skip Ad is a Chrome Extension that can automatically close the video ads on YouTube.
- Also hide the video overlay ads and ads in the column next to the video.
- Technologies Used - **Javascript,** Chrome Developer Tools.

CODING PROFILES

- Solved 400+ DSA Problems on various coding platforms.
- Leetcode : shivamm123
- geeksforgeeks : shivamm123

AUDITED COURSES

- Mastering Data Structures and Algorithms using C and C++ by abdul bari from Udemy
- The Complete 2022 Web Development Bootcamp by Angela Yu .

ANN_Classification

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
 !pip install plotly
import plotly
import plotly.express as px
from mpl_toolkits.mplot3d import Axes3D
```

```
Collecting plotly
 Downloading plotly-5.9.0-py2.py3-none-any.whl (15.2 MB) Collecting
tenacity>=6.2.0
 Downloading tenacity-8.0.1-py3-none-any.whl (24 kB) Installing
collected packages: tenacity, plotly
Successfully installed plotly-5.9.0 tenacity-8.0.1
```

Gathering Dataset

```
mnist = tf.keras.datasets.mnist

 (x_train, y_train), (x_test, y_test) = mnist.load_data() x_train, x_test =
x_train / 255.0, x_test / 255.0

print("shape of x_train: ", x_train.shape)
print("shape of x_test: ", x_test.shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf
keras-datasets/mnist.npz
11490434/11490434 [==============================] - 3s 0us/step shape of
x_train: (60000, 28, 28)
shape of x_test: (10000, 28, 28)
```

Instantiate or Build the Model

```
model = tf.keras.models.Sequential([
 tf.keras.layers.Flatten(input_shape=(28, 28)),
tf.keras.layers.Dense(128, activation= 'relu'),
tf.keras.layers.Dropout(0.2),
 tf.keras.layers.Dense(10, activation='softmax') ])
```

Compile & Train the model

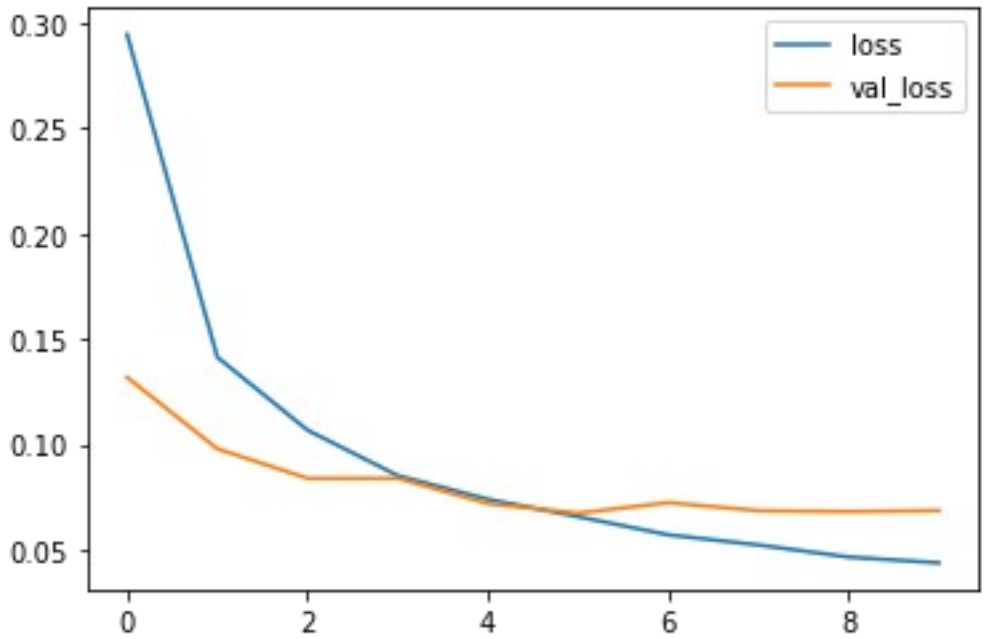
```
model.compile(optimizer='adam',
 loss='sparse_categorical_crossentropy',  metrics=['accuracy'])

 r = model.fit(x_train, y_train, validation_data=(x_test, y_test),  epochs=10)
Epoch 1/10
1875/1875 [==============================] - 4s 2ms/step - loss:  0.2945 -
accuracy: 0.9154 - val_loss: 0.1318 - val_accuracy: 0.9619 Epoch 2/10
1875/1875 [==============================] - 3s 2ms/step - loss:  0.1413 -
```

```
accuracy: 0.9586 - val_loss: 0.0979 - val_accuracy: 0.9710 Epoch 3/10
1875/1875 [==============================] - 3s 2ms/step - loss:  0.1067 -
accuracy: 0.9681 - val_loss: 0.0839 - val_accuracy: 0.9737 Epoch 4/10
1875/1875 [==============================] - 3s 2ms/step - loss:  0.0851 -
accuracy: 0.9741 - val_loss: 0.0838 - val_accuracy: 0.9735 Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss:  0.0739 -
accuracy: 0.9765 - val_loss: 0.0719 - val_accuracy: 0.9782 Epoch 6/10
1875/1875 [==============================] - 3s 2ms/step - loss:  0.0657 -
accuracy: 0.9789 - val_loss: 0.0673 - val_accuracy: 0.9806 Epoch 7/10
1875/1875 [==============================] - 3s 2ms/step - loss:  0.0570 -
accuracy: 0.9811 - val_loss: 0.0723 - val_accuracy: 0.9787 Epoch 8/10
1875/1875 [==============================] - 5s 3ms/step - loss:  0.0522 -
accuracy: 0.9831 - val_loss: 0.0685 - val_accuracy: 0.9794 Epoch 9/10
1875/1875 [==============================] - 4s 2ms/step - loss:  0.0465 -
accuracy: 0.9849 - val_loss: 0.0681 - val_accuracy: 0.9806 Epoch 10/10
1875/1875 [==============================] - 4s 2ms/step - loss:  0.0437 -
accuracy: 0.9851 - val_loss: 0.0686 - val_accuracy: 0.9789
```

#Plotting Loss per Iteration

```
plt.plot(r.history['loss'], label= 'loss')
plt.plot(r.history['val_loss'], label= 'val_loss')
plt.legend()
```

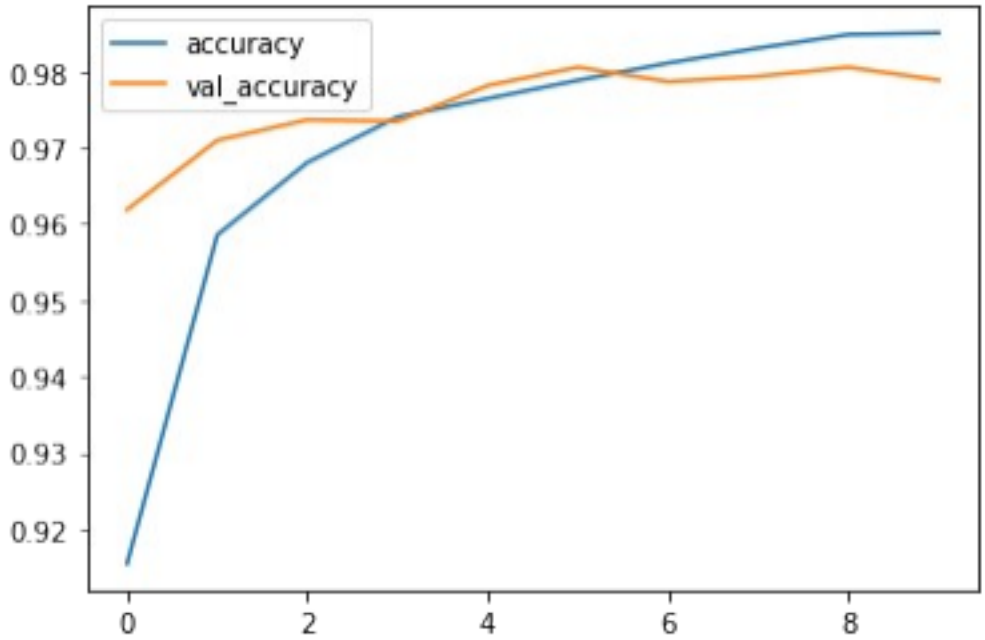
```
<matplotlib.legend.Legend at 0x17ac6ef8fa0>
```



#Plotting Accuracy per Iteration

```
plt.plot(r.history['accuracy'], label= 'accuracy') plt.plot(r.history['val_accuracy'],
label= 'val_accuracy') plt.legend()
```

```
<matplotlib.legend.Legend at 0x17ac7676d60>
```



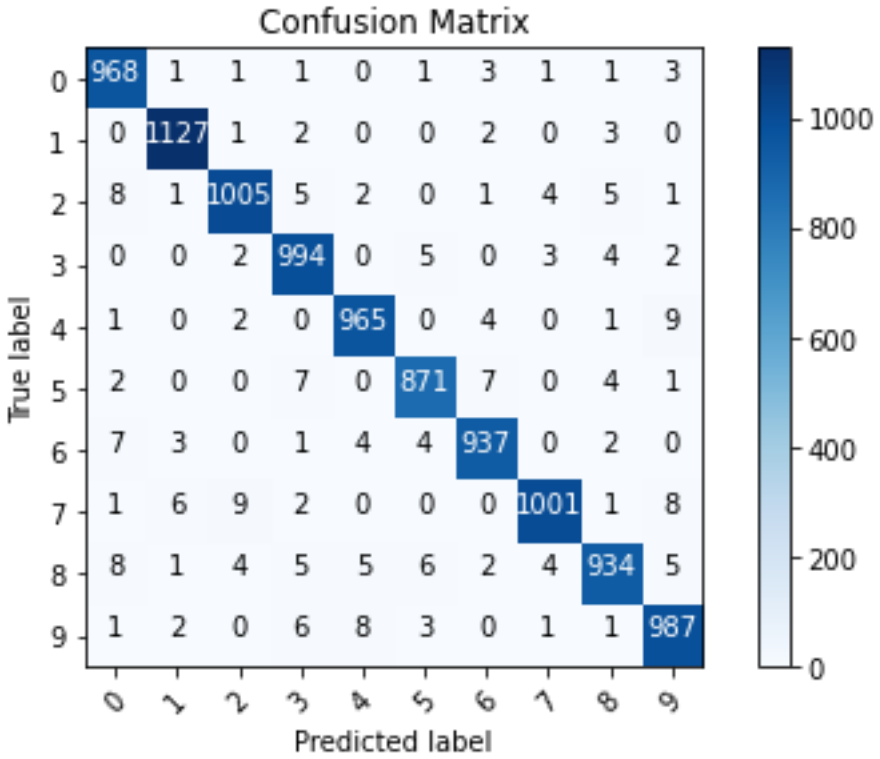

```
# Model Evaluation
print(model.evaluate(x_test, y_test))
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.0686  - accuracy:
0.9789
[0.06858374923467636, 0.9789000153541565]
```

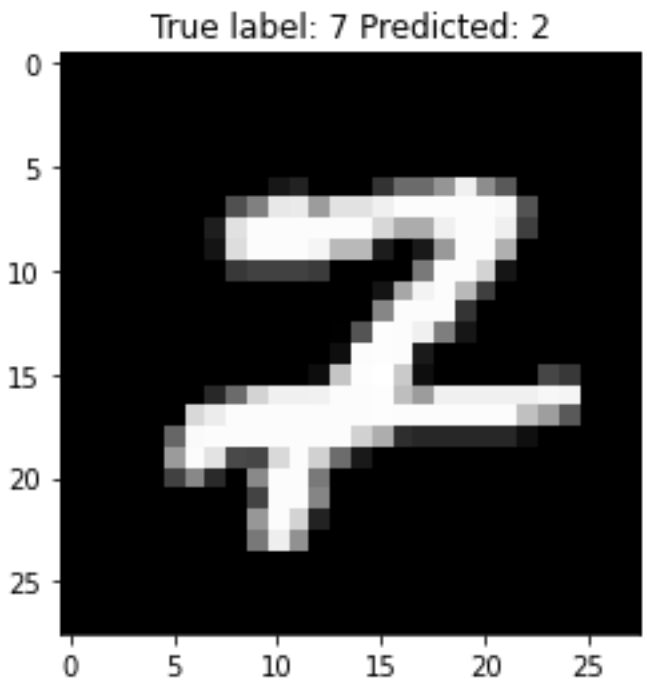
```
# Confusion Matrix

from sklearn.metrics import confusion_matrix
import itertools

def plot_confusion_matrix(cm, classes,
 normalize = False,
 title="Confusion Matrix",
 cmap = plt.cm.Blues):

 """
 This function prints and plots the confusion matrix.  Normalize can be
applied by setting 'normalize = True'  """

 if normalize:
 cm = cm.astype('float') / cm.sum(axis=1)[:,np.newaxis]  print("Normalized
Confusion Matrix")

 else:
 print("Confusion matrix without normalization")

 print(cm)

 plt.imshow(cm, interpolation='nearest', cmap=cmap)
 plt.title(title)
 plt.colorbar()
 tick_marks = np.arange(len(classes))
```

```
 plt.xticks(tick_marks, classes, rotation=45)
 plt.yticks(tick_marks, classes)

 fmt = '.2f' if normalize else 'd'
 thresh = cm.max() / 2.
 for i, j in itertools.product(range(cm.shape[0]),
 range(cm.shape[1])):
 plt.text(j, i, format(cm[i, j], fmt),
 horizontalalignment = 'center',
 color = "white" if cm[i, j] > thresh else "black")
 plt.tight_layout()
 plt.ylabel('True label')
 plt.xlabel('Predicted label')
 plt.show()
p_test = model.predict(x_test).argmax(axis=1)
cm = confusion_matrix(y_test, p_test)
plot_confusion_matrix(cm, list(range(10)))
```

```
313/313 [==============================] - 0s 1ms/step
Confusion matrix without normalization
[[ 968  1  1  1  0  1  3  1  1  3]  [ 0 1127  1  2  0  0  2  0  3  0]  [ 8  1 1005  5  2  0  1  4  5
1]  [ 0  0  2 994  0  5  0  3  4  2]  [ 1  0  2  0 965  0  4  0  1  9]  [ 2  0  0  7  0 871  7  0  4
1]  [ 7  3  0  1  4  4 937  0  2  0]  [ 1  6  9  2  0  0  0 1001  1  8]  [ 8  1  4  5  5  6  2  4
934  5]  [ 1  2  0  6  8  3  0  1  1 987]]
```




```
# Plotting few Places where the Model went wrong

misclassified_idx = np.where(p_test != y_test)[0]
i = np.random.choice(misclassified_idx)
plt.imshow(x_test[i], cmap= 'gray')
                                    plt.title("True label: %s Predicted: %s" % (y_test[i], p_test[i]));
```


True label: 7 Predicted: 2
0
5
10
15
20
25
0
5
10
15
20
25

Emotional classification

Importing Required Libraries

```
import pandas as pd
import os
import glob as gb
from tensorflow import keras

from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
```

Assigning Path for Dataset

```
TRAIN_DIR = r"C:\Users\Lenovo\Downloads\archive (4)\
eINTERFACE_2021_Image\train"
TEST_DIR = r"C:\Users\Lenovo\Downloads\archive (4)\
eINTERFACE_2021_Image\test"
BATCH_SIZE=64
```

Will see how many categories and images present

```
for folder in os.listdir(TRAIN_DIR):
 files = gb.glob(pathname= str(TRAIN_DIR+ '/'+ folder + '/*.jpg'))  print(f'For training data, found
{len(files)} in folder {folder}')
```

```
For training data, found 1896 in folder Anger
For training data, found 1891 in folder Disgust
For training data, found 1922 in folder Fear
For training data, found 1922 in folder Happiness
For training data, found 1922 in folder Sadness
For training data, found 1922 in folder Surprise
```

```
for folder in os.listdir(TEST_DIR):
 files = gb.glob(pathname= str(TEST_DIR+ '/'+ folder + '/*.jpg'))  print(f'For testing data, found
{len(files)} in folder {folder}')
```

```
For testing data, found 237 in folder Anger
For testing data, found 237 in folder Disgust
For testing data, found 241 in folder Fear
For testing data, found 241 in folder Happiness
For testing data, found 241 in folder Sadness
For testing data, found 241 in folder Surprise
```

Will see some random images withe their labels

```
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

def view_random_image(target_dir, target_class):
 # We will view images from here
 target_folder = target_dir + target_class
```

```
 # Get a random image path
 random_image = random.sample(od.listdir(target_folder), 1)
 # read in the image and plot it using matplolib
 img = mpimg.imread(target_folder+'/'+random_image[0])  plt.imshow(img)
 plt.title(target_class)
 plt.axis('off')
 print(f"Image shape {img.shape}")

 return img

class_names =
                ['Anger','Disgust','Fear','Happiness','Sadness','Surprise']

plt.figure(figsize=(20,10))
for i in range(18):
 plt.subplot(3, 6. i+1)
 class_name = random.choice(class_names)
 img =
view_random_image(target_dir="/content/drive/MyDrive/Colab_Notebook/
Emotion-Detection/train", target_class=class_name)

 File "<ipython-input-9-997932e3db8c>", line 3
 plt.subplot(3, 6. i+1)
 ^
SyntaxError: invalid syntax
```

Preparing data for training

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,  shear_range = 0.2,
zoom_range = 0.2,  horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory(TRAIN_DIR,  target_size = (128,  128),
 batch_size = BATCH_SIZE,
 class_mode = 'categorical')

test_set = test_datagen.flow_from_directory(TEST_DIR,  target_size = (128, 128),
 batch_size = BATCH_SIZE,  class_mode = 'categorical')

Found 11475 images belonging to 6 classes.
Found 1438 images belonging to 6 classes.
```

Basic model building (CNN Classifier)

```
# Initialising the CNN
classifier = Sequential()

# Step 1 - Convolution
classifier.add(Conv2D(16, (3, 3), input_shape = (128, 128, 3),  activation = 'relu'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))
```

```
# Adding a second convolutional layer
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))



# Step 3 - Flattening
classifier.add(Flatten())

# Step 4 - Full connection
classifier.add(Dense(units = 128, activation = 'relu'))

classifier.add(Dense(units = 6, activation = 'softmax'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss =
 'categorical_crossentropy', metrics = ['accuracy'])

## model summary
classifier.summary()
```

Model: "sequential"

_____

Layer (type) Output Shape Param #

==========

conv2d (Conv2D) (None, 126, 126, 16) 448

max_pooling2d (MaxPooling2D (None, 63, 63, 16) 0)

conv2d_1 (Conv2D) (None, 61, 61, 32) 4640

max_pooling2d_1 (MaxPooling (None, 30, 30, 32) 0 2D)

flatten (Flatten) (None, 28800) 0 dense (Dense) (None, 128) 3686528 dense_1

(Dense) (None, 6) 774

==========

Total params: 3,692,390

Trainable params: 3,692,390

Non-trainable params: 0

_____

```
history = classifier.fit(training_set,
 epochs = 5,
 validation_data = test_set)


                    classifier.save('model1.h5') # creates a HDF5 file 'my_model.h5'
```

Epoch 1/5

180/180 [==========] - 154s 858ms/step - loss: 1.2606 - accuracy: 0.4966 - val_loss: 1.1545 - val_accuracy: 0.5584 Epoch 2/5

180/180 [==========] - 154s 852ms/step - loss: 1.1580 - accuracy: 0.5461 - val_loss: 1.1027 - val_accuracy: 0.5577 Epoch 3/5

180/180 [==============================] - 153s 850ms/step - loss: 1.0857 - accuracy: 0.5728 - val_loss: 1.0417 - val_accuracy: 0.5869 Epoch 4/5
180/180 [==============================] - 153s 850ms/step - loss: 0.9949 - accuracy: 0.6159 - val_loss: 0.9250 - val_accuracy: 0.6426 Epoch 5/5
180/180 [==============================] - 153s 853ms/step - loss: 0.9364 - accuracy: 0.6418 - val_loss: 1.0033 - val_accuracy: 0.6127

Evaluating the model

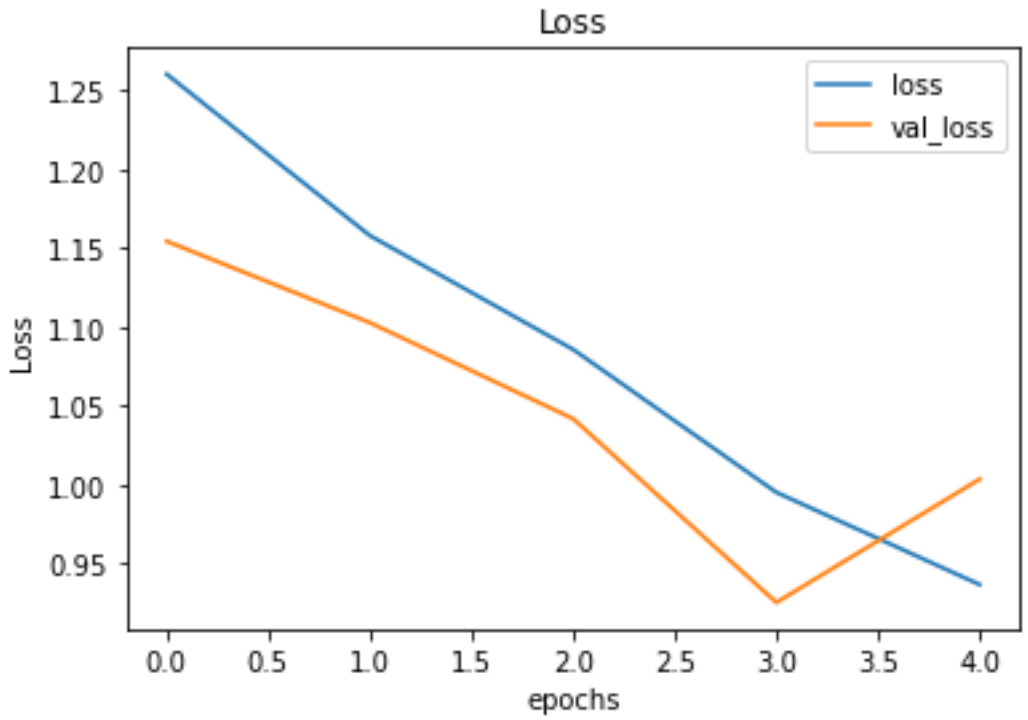
```
classifier.evaluate(test_set)
```

23/23 [==============================] - 8s 339ms/step - loss: 1.0033 - accuracy: 0.6127

[1.0033340454101562, 0.6126564741134644]

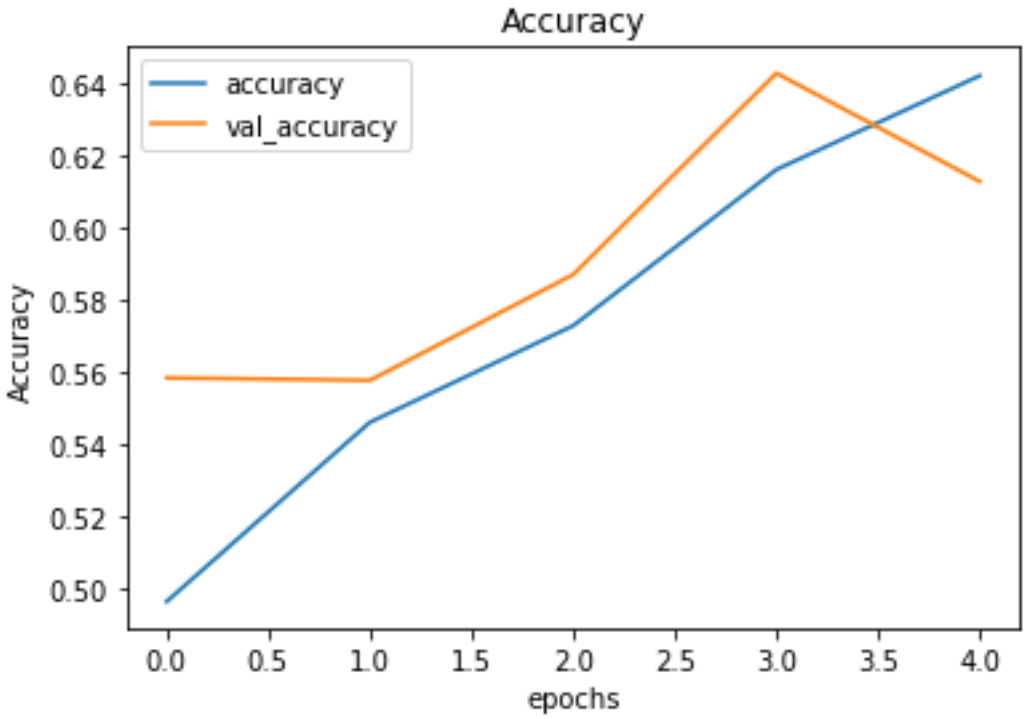
```
pd.DataFrame(history.history)[['loss','val_loss']].plot() plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('Loss')
```

Text(0, 0.5, 'Loss')




```
pd.DataFrame(history.history)[['accuracy','val_accuracy']].plot() plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('Accuracy')
```

Text(0, 0.5, 'Accuracy')




```
model_path = "model1.h5"
loaded_model = keras.models.load_model(model_path)

import matplotlib.pyplot as plt
import numpy as np
import cv2
from PIL import Image

image = cv2.imread("00000.png")

image_fromarray = Image.fromarray(image, 'RGB')
 resize_image = image_fromarray.resize((128, 128))
expand_input = np.expand_dims(resize_image,axis=0)
input_data = np.array(expand_input)
input_data = input_data/255

pred = loaded_model.predict(input_data)
 result = pred.argmax()
 result

 ---------------------------------------------------------------------------- -----
AttributeError Traceback (most recent call  last)
<ipython-input-27-15e09e4c29a6> in <module>
 9 image = cv2.imread("00000.png")
 10
 ---> 11 image_fromarray = Image.fromarray(image, 'RGB')
 12 resize_image = image_fromarray.resize((128, 128))  13 expand_input =
np.expand_dims(resize_image,axis=0)

~\anaconda3\lib\site-packages\PIL\Image.py in fromarray(obj, mode)  2760 ..
versionadded:: 1.1.6
 2761 """
```

```
-> 2762 arr = obj.__array_interface__
 2763 shape = arr["shape"]
 2764 ndim = len(shape)

AttributeError: 'NoneType' object has no attribute
'__array_interface__'
```

training_set.class_indices

```
{'Anger': 0,
 'Disgust': 1,
 'Fear': 2,
 'Happiness': 3,
 'Sadness': 4,
 'Surprise': 5}
```