```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Dense,Flatten,Dropout
import matplotlib.pyplot as plt
from keras.layers import BatchNormalization
from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# /content/drive/MyDrive/Face Recognition using CNN/Original Images
train_dir="/content/drive/MyDrive/face recognition/Original Images/Original Images"
generator = ImageDataGenerator()
train_ds = generator.flow_from_directory(train_dir, target_size=(224, 224), batch_size=32)
```

```
Found 5 images belonging to 31 classes.
```

```
classes = list(train_ds.class_indices.keys())
```

```
model = Sequential()
```

```
model.add(Conv2D(32, kernel_size = (3, 3), activation='relu', input_shape=(224,224,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
```

```
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
```

```
model.add(Conv2D(96, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
```

```
model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
```

```
model.add(Dropout(0.2))
model.add(Flatten())
```

```
model.add(Dense(128, activation='relu'))
model.add(Dense(len(classes),activation='softmax'))
```

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

```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
batch_normalization (BatchNormalization)	(None, 111, 111, 32)	128
conv2d_2 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
batch_normalization_1 (BatchNormalization)	(None, 54, 54, 64)	256
conv2d_3 (Conv2D)	(None, 52, 52, 96)	55,392
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 96)	0
batch_normalization_2 (BatchNormalization)	(None, 26, 26, 96)	384
conv2d_4 (Conv2D)	(None, 24, 24, 32)	27,680
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 32)	0
batch_normalization_3 (BatchNormalization)	(None, 12, 12, 32)	128
dropout (Dropout)	(None, 12, 12, 32)	0
flatten (Flatten)	(None, 4608)	0
dropout_1 (Dropout)	(None, 4608)	0
flatten_1 (Flatten)	(None, 4608)	0
dense (Dense)	(None, 128)	589,952
dense_1 (Dense)	(None, 31)	3,999
dense_2 (Dense)	(None, 128)	4,096
dense_3 (Dense)	(None, 31)	3,999

```
Total params: 705,406 (2.69 MB)
Trainable params: 704,958 (2.69 MB)
Non-trainable params: 448 (1.75 KB)
```

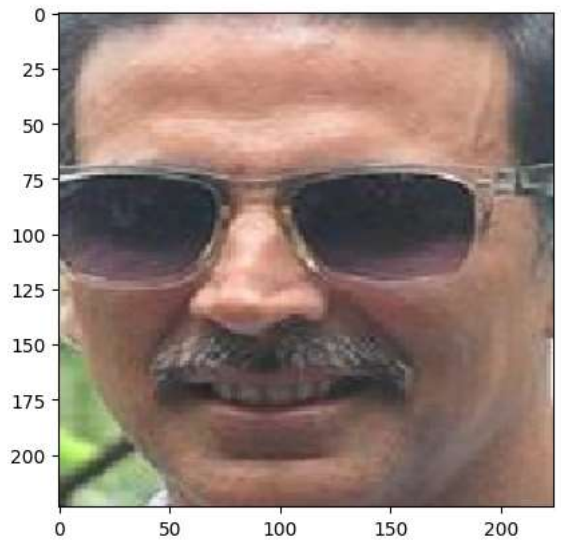
```
history = model.fit(train_ds,epochs= 10, batch_size=32)
```

```
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class
  self._warn_if_super_not_called()
1/1 ━━━━━━━━━━━━━━━━━━━━ 6s 6s/step - accuracy: 0.0000e+00 - loss: 3.4269
Epoch 2/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 821ms/step - accuracy: 0.0000e+00 - loss: 3.3558
Epoch 3/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 806ms/step - accuracy: 0.2000 - loss: 3.2992
Epoch 4/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.4000 - loss: 3.2667
Epoch 5/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.4000 - loss: 3.2391
Epoch 6/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - accuracy: 0.6000 - loss: 3.1868
Epoch 7/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - accuracy: 0.6000 - loss: 3.1593
Epoch 8/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - accuracy: 0.6000 - loss: 3.1320
Epoch 9/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 796ms/step - accuracy: 0.8000 - loss: 3.1044
Epoch 10/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.8000 - loss: 3.0766
```

```
from tensorflow.keras.utils import load_img, img_to_array


def predict_image(image_path):
    img = load_img(image_path, target_size=(224,224,3))
    plt.imshow(img)
    plt.show()
    x = img_to_array(img)
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])
    pred = model.predict(images, batch_size=32)
    print("Actual: "+(image_path.split("/")[-1]).split("_")[0])
    print("Predicted: "+classes[np.argmax(pred)])
```

```
predict_image("/content/drive/MyDrive/face recognition/Faces/Faces/Akshay Kumar_0.jpg")
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 355ms/step
Actual: Akshay Kumar
Predicted: Zac Efron
```