In [3]:
```python
import numpy as np
import pandas as pd
from keras.applications.mobilenet import MobileNet, preprocess_input
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dropout, Dense,BatchNormalization, Flatten, MaxPool2D
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, Callback
from keras.layers import Conv2D, Reshape
from keras.utils import Sequence
from keras.backend import epsilon
import tensorflow as tf
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from tensorflow.keras.layers import GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
import cv2


from tqdm.notebook import tqdm_notebook as tqdm

import os
```

In [ ]:

In [4]:
```python
print(os.listdir(r'C:\Users\adars\Exams.pdf\OneDrive - Amity University\Documents\real_and_fake_face'))
```

```
['training_fake', 'training_real']
```

In [5]:
```python
real = r"C:\Users\adars\Exams.pdf\OneDrive - Amity University\Documents\real_and_fake_face\training_real"
fake = r"C:\Users\adars\Exams.pdf\OneDrive - Amity University\Documents\real_and_fake_face\training_fake"

real_path = os.listdir(real)
fake_path = os.listdir(fake)
```

In [6]:
```python
def load_img(path):
    image = cv2.imread(path)
    image = cv2.resize(image,(224, 224))
    return image[...,::-1]
```

In [7]:
```python
    # fig = plt.figure(figsize=(10, 10))
     #    for i in range(1,16):
     #        plt.subplot(4, 4, i+1)
      #       plt.imshow(load_img(real + real_path[i]), cmap='gray')
       #      plt.suptitle("Real faces",fontsize=20)
        #     plt.axis('off')

    # plt.show()
```

In [8]:
```python
dataset_path = r"C:\Users\adars\Exams.pdf\OneDrive - Amity University\Documents\real_and_fake_face"
```

In [9]:
```python
data_with_aug = ImageDataGenerator(horizontal_flip=True,
                                   vertical_flip=False,
                                   rescale=1./255,
                                  validation_split=0.2)
```

In [10]:
```python
train = data_with_aug.flow_from_directory(dataset_path,
                                          class_mode="binary",
                                          target_size=(96, 96),
                                          batch_size=32,
                                          subset="training")
```

```
Found 1633 images belonging to 2 classes.
```

In [11]:
```python
val = data_with_aug.flow_from_directory(dataset_path,
                                          class_mode="binary",
                                          target_size=(96, 96),
                                          batch_size=32,
                                          subset="validation"
                                          )
```

```
Found 408 images belonging to 2 classes.
```

In [12]:
```python
mnet = MobileNetV2(include_top = False, weights = "imagenet" ,input_shape=(96,96,3))
```

In [13]:
```python
tf.keras.backend.clear_session()

model = Sequential([mnet,
                    GlobalAveragePooling2D(),
                    Dense(512, activation = "relu"),
                    BatchNormalization(),
                    Dropout(0.3),
                    Dense(128, activation = "relu"),
                    Dropout(0.1),
                    # Dense(32, activation = "relu"),
                    # Dropout(0.3),
                    Dense(2, activation = "softmax")])

model.layers[0].trainable = False

model.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics="accuracy")

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
mobilenetv2_1.00_96 (Functio (None, 3, 3, 1280)        2257984   
_________________________________________________________________
global_average_pooling2d (Gl (None, 1280)              0         
_________________________________________________________________
dense (Dense)                (None, 512)               655872    
_________________________________________________________________
batch_normalization (BatchNo (None, 512)               2048      
_________________________________________________________________
dropout (Dropout)            (None, 512)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664     
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258       
=================================================================
Total params: 2,981,826
Trainable params: 722,818
Non-trainable params: 2,259,008
_________________________________________________________________
```

In [14]:
```python
def scheduler(epoch):
    if epoch <= 2:
        return 0.001
    elif epoch > 2 and epoch <= 15:
        return 0.0001
    else:
        return 0.00001

lr_callbacks = tf.keras.callbacks.LearningRateScheduler(scheduler)
```

In [15]:
```python
hist = model.fit_generator(train,
                    epochs=20,
                    callbacks=[lr_callbacks],
                    validation_data=val)
```

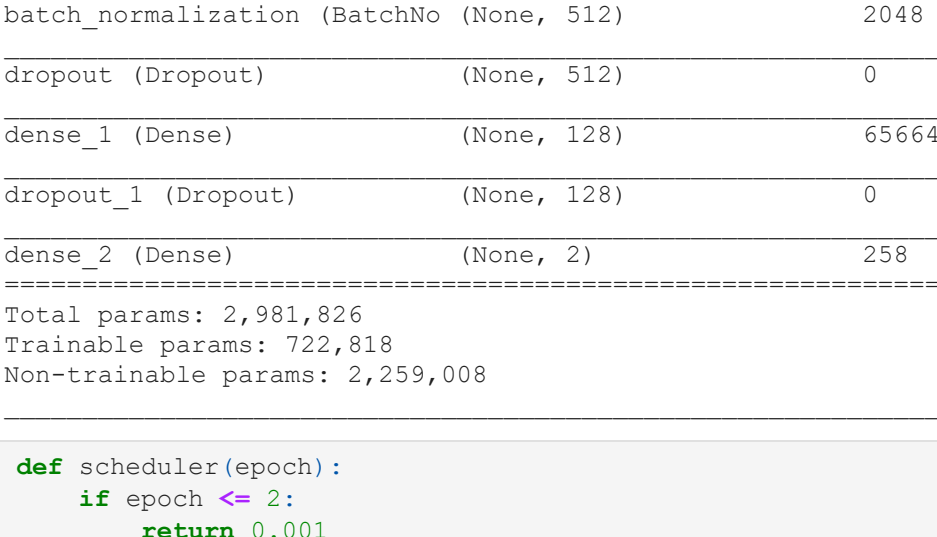
```
WARNING:tensorflow:From <ipython-input-15-abeff273e3ad>:1: Model.fit_generator (from tensorflow.python.keras.en
gine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/20
52/52 [==============================] - 39s 750ms/step - loss: 0.9010 - accuracy: 0.5928 - val_loss: 0.9107 -
val_accuracy: 0.5319
Epoch 2/20
52/52 [==============================] - 33s 632ms/step - loss: 0.5651 - accuracy: 0.7134 - val_loss: 1.2236 -
val_accuracy: 0.5025
Epoch 3/20
52/52 [==============================] - 37s 703ms/step - loss: 0.4984 - accuracy: 0.7630 - val_loss: 1.1185 -
val_accuracy: 0.5343
Epoch 4/20
52/52 [==============================] - 35s 666ms/step - loss: 0.3758 - accuracy: 0.8249 - val_loss: 1.0188 -
val_accuracy: 0.5319
Epoch 5/20
52/52 [==============================] - 33s 644ms/step - loss: 0.3342 - accuracy: 0.8616 - val_loss: 0.9989 -
val_accuracy: 0.5172
Epoch 6/20
52/52 [==============================] - 34s 646ms/step - loss: 0.3188 - accuracy: 0.8690 - val_loss: 1.0081 -
val_accuracy: 0.5172
Epoch 7/20
52/52 [==============================] - 33s 638ms/step - loss: 0.2968 - accuracy: 0.8775 - val_loss: 0.9940 -
val_accuracy: 0.5147
Epoch 8/20
52/52 [==============================] - 35s 670ms/step - loss: 0.2860 - accuracy: 0.8830 - val_loss: 1.0343 -
val_accuracy: 0.5270
Epoch 9/20
52/52 [==============================] - 33s 632ms/step - loss: 0.2915 - accuracy: 0.8800 - val_loss: 1.0603 -
val_accuracy: 0.5098
Epoch 10/20
52/52 [==============================] - 33s 628ms/step - loss: 0.2617 - accuracy: 0.8977 - val_loss: 1.0837 -
val_accuracy: 0.5049
Epoch 11/20
52/52 [==============================] - 33s 625ms/step - loss: 0.2617 - accuracy: 0.9032 - val_loss: 1.0360 -
val_accuracy: 0.5049
Epoch 12/20
52/52 [==============================] - 33s 643ms/step - loss: 0.2233 - accuracy: 0.9259 - val_loss: 1.0954 -
val_accuracy: 0.5049
Epoch 13/20
52/52 [==============================] - 36s 701ms/step - loss: 0.2131 - accuracy: 0.9259 - val_loss: 1.1098 -
val_accuracy: 0.5147
Epoch 14/20
52/52 [==============================] - 38s 722ms/step - loss: 0.2197 - accuracy: 0.9210 - val_loss: 1.1690 -
val_accuracy: 0.5172
Epoch 15/20
52/52 [==============================] - 37s 714ms/step - loss: 0.1959 - accuracy: 0.9296 - val_loss: 1.1783 -
val_accuracy: 0.5074
Epoch 16/20
52/52 [==============================] - 37s 717ms/step - loss: 0.2023 - accuracy: 0.9345 - val_loss: 1.2012 -
val_accuracy: 0.5196
Epoch 17/20
52/52 [==============================] - 39s 757ms/step - loss: 0.1864 - accuracy: 0.9314 - val_loss: 1.1892 -
val_accuracy: 0.5343
Epoch 18/20
52/52 [==============================] - 39s 753ms/step - loss: 0.1701 - accuracy: 0.9388 - val_loss: 1.2231 -
val_accuracy: 0.5074
Epoch 19/20
52/52 [==============================] - 37s 716ms/step - loss: 0.1823 - accuracy: 0.9400 - val_loss: 1.2161 -
val_accuracy: 0.5294
Epoch 20/20
52/52 [==============================] - 38s 736ms/step - loss: 0.1841 - accuracy: 0.9443 - val_loss: 1.1973 -
val_accuracy: 0.5074
```

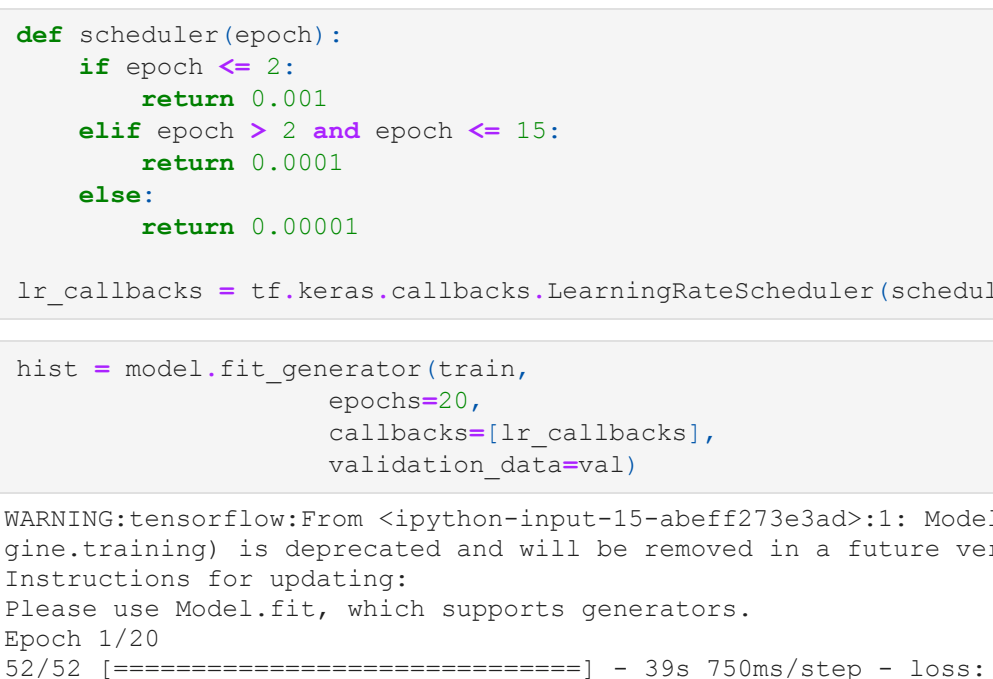
In [16]:
```python
#model.save("model.h5")
```

In [17]:
```python
epochs = 20
train_loss = hist.history['loss']
val_loss = hist.history['val_loss']
train_acc = hist.history['accuracy']
val_acc = hist.history['val_accuracy']
xc = range(epochs)

plt.figure(1,figsize=(7,5))
plt.plot(xc,train_loss)
plt.plot(xc,val_loss)
plt.xlabel('num of Epochs')
plt.ylabel('loss')
plt.title('train_loss vs val_loss')
plt.grid(True)
plt.legend(['train','val'])
#print plt.style.available # use bmh, classic,ggplot for big pictures
plt.style.use(['classic'])

plt.figure(2,figsize=(7,5))
plt.plot(xc,train_acc)
plt.plot(xc,val_acc)
plt.xlabel('num of Epochs')
plt.ylabel('accuracy')
plt.title('train_acc vs val_acc')
plt.grid(True)
plt.legend(['train','val'],loc=4)
#print plt.style.available # use bmh, classic,ggplot for big pictures
plt.style.use(['classic'])
```





In [18]:
```python
train = data_with_aug.flow_from_directory(dataset_path,
                                          class_mode="binary",
                                          target_size=(224, 224),
                                          batch_size=98,
                                          subset="training")

val = data_with_aug.flow_from_directory(dataset_path,
                                          class_mode="binary",
                                          target_size=(224, 224),
                                          batch_size=98,
                                          subset="validation"
                                          )
```

```
Found 1633 images belonging to 2 classes.
Found 408 images belonging to 2 classes.
```

In [19]:
```python
vgg16_model = tf.keras.applications.vgg16.VGG16(include_top=False, weights="imagenet", input_shape=(224,224,3))
```

In [20]:
```python
vgg16_model.output[-1]
```

Out[20]:
```
<tf.Tensor 'strided_slice:0' shape=(7, 7, 512) dtype=float32>
```

In [21]:
```python
# model = Sequential()
# for layer in vgg16_model.layers[:-1]:
#     model.add(layer)

# for layer in model.layers:
#     layer.trainable = False

# model.add(Dense(2, activation='softmax'))


model = Sequential([vgg16_model,
                    Flatten(),
#                      GlobalAveragePooling2D(),
#                      Dense(512, activation = "relu"),
#                      BatchNormalization(),
#                      Dropout(0.3),
#                      Dense(128, activation = "relu"),
#                      Dropout(0.1),
#                      # Dense(32, activation = "relu"),
#                      # Dropout(0.3),
                    Dense(2, activation = "softmax")])

model.layers[0].trainable = False

model.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics="accuracy")

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688  
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0         
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 50178     
=================================================================
Total params: 14,764,866
Trainable params: 50,178
Non-trainable params: 14,714,688
_________________________________________________________________
```

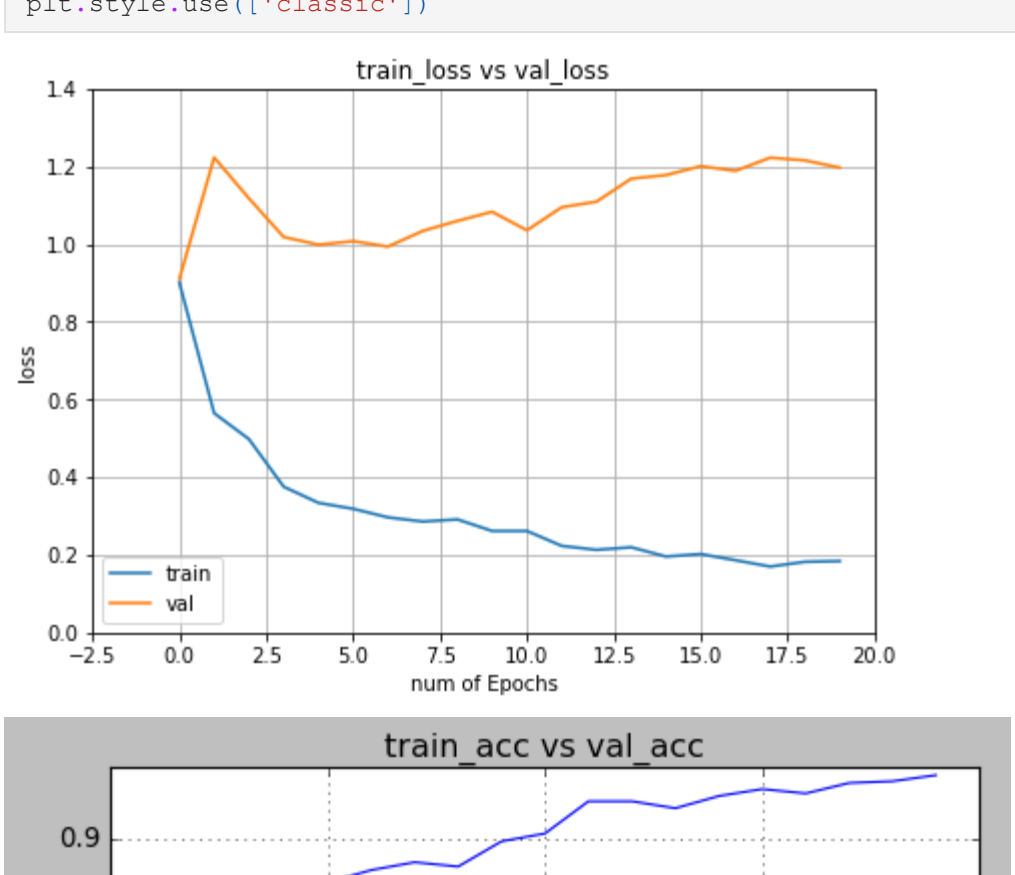
In [22]:
```python
hist =  model.fit_generator(train,
                    epochs=20,
                    callbacks=[lr_callbacks],
                    validation_data=val)
```

```
Epoch 1/20
17/17 [==============================] - 483s 28s/step - loss: 1.3123 - accuracy: 0.5530 - val_loss: 1.1234 - v
al_accuracy: 0.5319
Epoch 2/20
17/17 [==============================] - 475s 28s/step - loss: 0.6105 - accuracy: 0.6871 - val_loss: 0.7982 - v
al_accuracy: 0.5686
Epoch 3/20
17/17 [==============================] - 410s 24s/step - loss: 0.5131 - accuracy: 0.7514 - val_loss: 0.7756 - v
al_accuracy: 0.5564
Epoch 4/20
17/17 [==============================] - 331s 19s/step - loss: 0.4689 - accuracy: 0.7930 - val_loss: 0.7797 - v
al_accuracy: 0.5343
Epoch 5/20
17/17 [==============================] - 309s 18s/step - loss: 0.4491 - accuracy: 0.8059 - val_loss: 0.8356 - v
al_accuracy: 0.5637
Epoch 6/20
17/17 [==============================] - 317s 19s/step - loss: 0.4412 - accuracy: 0.8175 - val_loss: 0.8193 - v
al_accuracy: 0.5539
Epoch 7/20
17/17 [==============================] - 311s 18s/step - loss: 0.4419 - accuracy: 0.8065 - val_loss: 0.8377 - v
al_accuracy: 0.5490
Epoch 8/20
17/17 [==============================] - 317s 19s/step - loss: 0.4357 - accuracy: 0.8175 - val_loss: 0.7920 - v
al_accuracy: 0.5662
Epoch 9/20
17/17 [==============================] - 307s 18s/step - loss: 0.4254 - accuracy: 0.8236 - val_loss: 0.8052 - v
al_accuracy: 0.5637
Epoch 10/20
17/17 [==============================] - 313s 18s/step - loss: 0.4171 - accuracy: 0.8365 - val_loss: 0.7967 - v
al_accuracy: 0.5613
Epoch 11/20
17/17 [==============================] - 308s 18s/step - loss: 0.4171 - accuracy: 0.8298 - val_loss: 0.8216 - v
al_accuracy: 0.5539
Epoch 12/20
17/17 [==============================] - 302s 18s/step - loss: 0.4139 - accuracy: 0.8298 - val_loss: 0.8132 - v
al_accuracy: 0.5662
Epoch 13/20
17/17 [==============================] - 304s 18s/step - loss: 0.4063 - accuracy: 0.8285 - val_loss: 0.8331 - v
al_accuracy: 0.5392
Epoch 14/20
17/17 [==============================] - 309s 18s/step - loss: 0.4022 - accuracy: 0.8310 - val_loss: 0.8243 - v
al_accuracy: 0.5392
Epoch 15/20
17/17 [==============================] - 300s 18s/step - loss: 0.3992 - accuracy: 0.8469 - val_loss: 0.8076 - v
al_accuracy: 0.5539
Epoch 16/20
17/17 [==============================] - 303s 18s/step - loss: 0.3949 - accuracy: 0.8463 - val_loss: 0.8017 - v
al_accuracy: 0.5588
Epoch 17/20
17/17 [==============================] - 308s 18s/step - loss: 0.3884 - accuracy: 0.8512 - val_loss: 0.8162 - v
al_accuracy: 0.5466
Epoch 18/20
17/17 [==============================] - 319s 19s/step - loss: 0.3853 - accuracy: 0.8475 - val_loss: 0.8271 - v
al_accuracy: 0.5613
Epoch 19/20
17/17 [==============================] - 388s 23s/step - loss: 0.3913 - accuracy: 0.8396 - val_loss: 0.8361 - v
al_accuracy: 0.5466
Epoch 20/20
17/17 [==============================] - 388s 23s/step - loss: 0.3861 - accuracy: 0.8487 - val_loss: 0.8246 - v
al_accuracy: 0.5466
```

In [23]:
```python
epochs = 20
train_loss = hist.history['loss']
val_loss = hist.history['val_loss']
train_acc = hist.history['accuracy']
val_acc = hist.history['val_accuracy']
xc = range(epochs)

plt.figure(1,figsize=(7,5))
plt.plot(xc,train_loss)
plt.plot(xc,val_loss)
plt.xlabel('num of Epochs')
plt.ylabel('loss')
plt.title('train_loss vs val_loss')
plt.grid(True)
plt.legend(['train','val'])
#print plt.style.available # use bmh, classic,ggplot for big pictures
plt.style.use(['classic'])

plt.figure(2,figsize=(7,5))
plt.plot(xc,train_acc)
plt.plot(xc,val_acc)
plt.xlabel('num of Epochs')
plt.ylabel('accuracy')
plt.title('train_acc vs val_acc')
plt.grid(True)
plt.legend(['train','val'],loc=4)
#print plt.style.available # use bmh, classic,ggplot for big pictures
plt.style.use(['classic'])
```



In [24]:
```python
model.save("model.h5")
```

In [ ]:
```python
#Creating an array of predicted test images

predictions = Model.predict_generator(val)
```

In [ ]:
```python
val_path = r"C:\Users\adars\Exams.pdf\OneDrive - Amity University\Documents\real_and_fake_face/"

plt.figure(figsize=(15,15))

start_index = 250

for i in range(16):
    plt.subplot(4,4, i+1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])

    preds = np.argmax(predictions[[start_index+i]])

    gt = val.filenames[start_index+i][9:13]


    if gt == "fake":
        gt = 0
    else:
        gt = 1

    if preds != gt:
        col ="r"
    else:
        col = "g"

    plt.xlabel('i={}, pred={}, gt={}'.format(start_index+i,preds,gt),color=col)
    plt.imshow(load_img(val_path+val.filenames[start_index+i]))
    plt.tight_layout()
```

In [27]:
```python
#!pip install CMake
```

In [28]:
```python
#!pip install face-recognition
```

In [29]:
```python
#96,96,3
# Real - 1
# Fake - 0
status=''
import cv2
import numpy as np
#import FF from cv2
cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)
from keras.models import load_model

model=load_model(r'C:\Users\adars\model.h5')

while True:
    success, img = cap.read()
    frame=cv2.resize(img,(96,96))
    pred=model.predict(np.expand_dims(frame,axis=0))
    if np.argmax(pred)==1:
        status='Real'
    else:
        status='Fake'

    # font
    font = cv2.FONT_HERSHEY_SIMPLEX

    # org
    org = (50, 50)

    # fontScale
    fontScale = 1

    # Blue color in BGR
    color = (255, 0, 0)

    # Line thickness of 2 px
    thickness = 2

    # Using cv2.putText() method
    img = cv2.putText(img, status, org, font, fontScale, color, thickness, cv2.LINE_AA)
    cv2.imshow("Video",img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
WARNING:tensorflow:Model was constructed with shape (None, 224, 224, 3) for input Tensor("vgg16_input_1:0", sha
pe=(None, 224, 224, 3), dtype=float32), but it was called on an input with incompatible shape (None, 96, 96,
3).
WARNING:tensorflow:Model was constructed with shape (None, 224, 224, 3) for input Tensor("input_1_1:0", shape=
(None, 224, 224, 3), dtype=float32), but it was called on an input with incompatible shape (None, 96, 96, 3).
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-29-97d3ebb49252> in <module>
     16     success, img = cap.read()
     17     frame=cv2.resize(img,(96,96))
---> 18     pred=model.predict(np.expand_dims(frame,axis=0))
     19     if np.argmax(pred)==1:
     20         status='Real'

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in _method_wrapper(self, *args, **kwar
gs)
    128       raise ValueError('{} is not supported in multi-worker mode.'.format(
    129           method.__name__))
--> 130     return method(self, *args, **kwargs)
    131
    132   return tf_decorator.make_decorator(

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in predict(self, x, batch_size, verbos
e, steps, callbacks, max_queue_size, workers, use_multiprocessing)
   1597           for step in data_handler.steps():
   1598             callbacks.on_predict_batch_begin(step)
-> 1599             tmp_batch_outputs = predict_function(iterator)
   1600             if data_handler.should_sync:
   1601               context.async_wait()

~\anaconda3\lib\site-packages\tensorflow\python\eager\def_function.py in __call__(self, *args, **kwds)
    778       else:
    779         compiler = "nonXla"
--> 780         result = self._call(*args, **kwds)
    781
    782       new_tracing_count = self._get_tracing_count()

~\anaconda3\lib\site-packages\tensorflow\python\eager\def_function.py in _call(self, *args, **kwds)
    821       # This is the first call of __call__, so we have to initialize.
    822       initializers = []
--> 823       self._initialize(args, kwds, add_initializers_to=initializers)
    824     finally:
    825       # At this point we know that the initialization is complete (or less

~\anaconda3\lib\site-packages\tensorflow\python\eager\def_function.py in _initialize(self, args, kwds, add_init
ializers_to)
    694     self._graph_deleter = FunctionDeleter(self._lifted_initializer_graph)
    695     self._concrete_stateful_fn = (
--> 696         self._stateful_fn._get_concrete_function_internal_garbage_collected(  # pylint: disable=protect
ed-access
    697             *args, **kwds))
    698

~\anaconda3\lib\site-packages\tensorflow\python\eager\function.py in _get_concrete_function_internal_garbage_co
llected(self, *args, **kwargs)
   2853       args, kwargs = None, None
   2854     with self._lock:
-> 2855       graph_function, _, _ = self._maybe_define_function(args, kwargs)
   2856     return graph_function
   2857

~\anaconda3\lib\site-packages\tensorflow\python\eager\function.py in _maybe_define_function(self, args, kwargs)
   3211
   3212       self._function_cache.missed.add(call_context_key)
-> 3213       graph_function = self._create_graph_function(args, kwargs)
   3214       self._function_cache.primary[cache_key] = graph_function
   3215       return graph_function, args, kwargs

~\anaconda3\lib\site-packages\tensorflow\python\eager\function.py in _create_graph_function(self, args, kwargs,
 override_flat_arg_shapes)
   3063     arg_names = base_arg_names + missing_arg_names
   3064     graph_function = ConcreteFunction(
-> 3065         func_graph_module.func_graph_from_py_func(
   3066             self._name,
   3067             self._python_function,

~\anaconda3\lib\site-packages\tensorflow\python\framework\func_graph.py in func_graph_from_py_func(name, python
_func, args, kwargs, signature, func_graph, autograph, autograph_options, add_control_dependencies, arg_names,
 op_return_value, collections, capture_by_value, override_flat_arg_shapes)
    984         _, original_func = tf_decorator.unwrap(python_func)
    985
--> 986       func_outputs = python_func(*func_args, **func_kwargs)
    987
    988       # invariant: `func_outputs` contains only Tensors, CompositeTensors,

~\anaconda3\lib\site-packages\tensorflow\python\eager\def_function.py in wrapped_fn(*args, **kwds)
    598         # __wrapped__ allows AutoGraph to swap in a converted function. We give
    599         # the function a weak reference to itself to avoid a reference cycle.
--> 600         return weak_wrapped_fn().__wrapped__(*args, **kwds)
    601     weak_wrapped_fn = weakref.ref(wrapped_fn)
    602

~\anaconda3\lib\site-packages\tensorflow\python\framework\func_graph.py in wrapper(*args, **kwargs)
    971           except Exception as e:  # pylint:disable=broad-except
    972             if hasattr(e, "ag_error_metadata"):
--> 973               raise e.ag_error_metadata.to_exception(e)
    974             else:
    975               raise

ValueError: in user code:

    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:1462 predict_function
  *
        return step_function(self, iterator)
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:1452 step_function  *
*
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\distribute\distribute_lib.py:1211 run
        return self._extended.call_for_each_replica(fn, args=args, kwargs=kwargs)
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\distribute\distribute_lib.py:2585 call_for_eac
h_replica
        return self._call_for_each_replica(fn, args, kwargs)
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\distribute\distribute_lib.py:2945 _call_for_ea
ch_replica
        return fn(*args, **kwargs)
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:1445 run_step  **
        outputs = model.predict_step(data)
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:1418 predict_step
        return self(x, training=False)
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\keras\engine\base_layer.py:985 __call__
        outputs = call_fn(inputs, *args, **kwargs)
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\keras\engine\sequential.py:372 call
        return super(Sequential, self).call(inputs, training=training, mask=mask)
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\keras\engine\functional.py:385 call
        return self._run_internal_graph(
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\keras\engine\functional.py:508 _run_internal_g
raph
        outputs = node.layer(*args, **kwargs)
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\keras\engine\base_layer.py:975 __call__
        input_spec.assert_input_compatibility(self.input_spec, inputs,
    C:\Users\adars\anaconda3\lib\site-packages\tensorflow\python\keras\engine\input_spec.py:212 assert_input_co
mpatibility
        raise ValueError(

    ValueError: Input 0 of layer dense_3 is incompatible with the layer: expected axis -1 of input shape to hav
e value 25088 but received input with shape [None, 4608]
```

In [ ]:
```python
import cv2
import face_recognition as fr
import numpy as np
import os
vid=cv2.VideoCapture(0)
from keras.models import load_model

model=load_model(r'C:\Users\adars\model.h5')

process_this_frame=True
status="Unknown"
face_locations = []
face_encodings = []
face_names = []
while(True):
    ret,frame=vid.read()



    # font
    font = cv2.FONT_HERSHEY_SIMPLEX

    # org
    org = (50, 50)

    # fontScale
    fontScale = 1

    # Blue color in BGR
    color = (255, 0, 0)

    # Line thickness of 2 px
    thickness = 2

    frame=cv2.flip(frame,1)
    if process_this_frame:
        small_frame=cv2.resize(frame,(0,0),fx=0.25,fy=0.25)
        rbg_small_frame=small_frame[:,:,::-1]
        face_locations=fr.face_locations(rbg_small_frame)

    process_this_frame=not process_this_frame
    print(face_locations)
    for(top,right,bottom,left) in (face_locations):
        top*=4
        right*=4
        bottom*=4
        left*=4

        cv2.rectangle(frame,(left-30,top-60),(right+30,bottom+30),(5,225,0),2)
        r = max(bottom, left) / 2
        centerx = top + bottom / 2
        centery = right + left / 2
        nx = int(centerx - r)
        ny = int(centery - r)
        nr = int(r * 2)

        faceimg = img[ny:ny+nr, nx:nx+nr]
        cv2.imshow("face",faceimg)
        resframe=cv2.resize(faceimg,(96,96))
        pred=model.predict(np.expand_dims(resframe,axis=0))
        if np.argmax(pred)==1:
            status='Real'
        else:
            status='Fake'

        #cv2.rectangle(frame, (left-30, bottom - 5), (right+30, bottom+30), (5,225,0), cv2.FILLED)
        #cv2.putText(frame,name,(left+6,bottom+20),cv2.FONT_HERSHEY_DUPLEX,1.0,(255,255,255),1,cv2.FILLED)

        cv2.putText(frame, status, org, font, fontScale, color, thickness, cv2.LINE_AA)
    #cv2.imshow("Frame",frame)
    #cv2.imshow("face",faceimg)
    if cv2.waitKey(1) & 0xFF==ord('q'):
        break

vid.release()
cv2.destroyAllWindows()
```

```
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
[]
```

In [1]:

```
import cv2
import face_recognition as fr
import numpy as np
import os
vid=cv2.VideoCapture(0)
from keras.models import load_model

model=load_model(r'C:\Users\adars\model.h5')

process_this_frame=True
status="Unknown"
face_locations = []
face_encodings = []
face_names = []
while(True):
    ret,frame=vid.read()



    # font
    font = cv2.FONT_HERSHEY_SIMPLEX

    # org
    org = (50, 50)

    # fontScale
    fontScale = 1

    # Blue color in BGR
    color = (255, 0, 0)

    # Line thickness of 2 px
    thickness = 2

    frame=cv2.flip(frame,1)
    if process_this_frame:
        small_frame=cv2.resize(frame,(0,0),fx=0.25,fy=0.25)
        rbg_small_frame=small_frame[:,:,::-1]
        face_locations=fr.face_locations(rbg_small_frame)

    process_this_frame=not process_this_frame
    print(face_locations)
    for(top,right,bottom,left) in (face_locations):
        top*=4
        right*=4
        bottom*=4
        left*=4

        cv2.rectangle(frame,(left-30,top-60),(right+30,bottom+30),(5,225,0),2)
        r = max(bottom, left) / 2
        centerx = top + bottom / 2
        centery = right + left / 2
        nx = int(centerx - r)
        ny = int(centery - r)
        nr = int(r * 2)

        faceimg = frame[left:right, top:bottom]
        cv2.imshow("face",faceimg)
        resframe=cv2.resize(faceimg,(96,96))
        pred=model.predict(np.expand_dims(resframe,axis=0))
        if np.argmax(pred)==1:
            status='Real'
        else:
            status='Fake'

        #cv2.rectangle(frame, (left-30, bottom - 5), (right+30, bottom+30), (5,225,0), cv2.FILLED)
        #cv2.putText(frame,name,(left+6,bottom+20),cv2.FONT_HERSHEY_DUPLEX,1.0,(255,255,255),1,cv2.FILLED)

        cv2.putText(frame, status, org, font, fontScale, color, thickness, cv2.LINE_AA)
    cv2.imshow("Frame",frame)
    cv2.imshow("face",faceimg)
    if cv2.waitKey(1) & 0xFF==ord('q'):
        break

vid.release()
cv2.destroyAllWindows()
```

```
[]
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-57d80cb6344c> in <module>
     69         cv2.putText(frame, status, org, font, fontScale, color, thickness, cv2.LINE_AA)
     70     cv2.imshow("Frame",frame)
---> 71     cv2.imshow("face",faceimg)
     72     if cv2.waitKey(1) & 0xFF==ord('q'):
     73         break

NameError: name 'faceimg' is not defined
```

In [1]:

```
import cv2
import face_recognition as fr
import numpy as np
import os
vid=cv2.VideoCapture(0)
from keras.models import load_model

model=load_model(r'C:\Users\adars\model.h5')

process_this_frame=True
status="Unknown"
face_locations = []
face_encodings = []
face_names = []
while(True):
    ret,frame=vid.read()



    # font
    font = cv2.FONT_HERSHEY_SIMPLEX

    # org
    org = (50, 50)

    # fontScale
    fontScale = 1

    # Blue color in BGR
    color = (255, 0, 0)

    # Line thickness of 2 px
    thickness = 2

    frame=cv2.flip(frame,1)
    if process_this_frame:
        small_frame=cv2.resize(frame,(0,0),fx=0.25,fy=0.25)
        rbg_small_frame=small_frame[:,:,::-1]
        face_locations=fr.face_locations(rbg_small_frame)

    process_this_frame=not process_this_frame
    print(face_locations)
    for(top,right,bottom,left) in (face_locations):
        top*=4
        right*=4
        bottom*=4
        left*=4

        cv2.rectangle(frame,(left-30,top-60),(right+30,bottom+30),(5,225,0),2)
        r = max(bottom, left) / 2
        centerx = top + bottom / 2
        centery = right + left / 2
        nx = int(centerx - r)
        ny = int(centery - r)
        nr = int(r * 2)

        faceimg = frame[left:right, top:bottom]
       # cv2.imshow("face",faceimg)
        resframe=cv2.resize(faceimg,(96,96))
        pred=model.predict(np.expand_dims(resframe,axis=0))
        if np.argmax(pred)==1:
            status='Real'
        else:
            status='Fake'

        #cv2.rectangle(frame, (left-30, bottom - 5), (right+30, bottom+30), (5,225,0), cv2.FILLED)
        #cv2.putText(frame,name,(left+6,bottom+20),cv2.FONT_HERSHEY_DUPLEX,1.0,(255,255,255),1,cv2.FILLED)

        cv2.putText(frame, status, org, font, fontScale, color, thickness, cv2.LINE_AA)
    cv2.imshow("Frame",frame)
        #cv2.imshow("face",faceimg)
    if cv2.waitKey(1) & 0xFF==ord('q'):
        break

vid.release()
cv2.destroyAllWindows()
```

```
Thu, 06 Jun 2013
/home/pankaj/live/kernel/.git/commit-tmp 0000 Merge branch 'for-3.10-fixes' of git://git.kernel.org/pub/scm/linux/kernel/git/tj/cgroup
Merge: 0ac3a7d 8ab6f2e
Author: Linus Torvalds <torvalds@linux-foundation.org>
Date:   Thu Jun 6 09:45:27 2013 -0700
    Merge branch 'for-3.10-fixes' of git://git.kernel.org/pub/scm/linux/kernel/git/tj/cgroup
    Pull cgroup fixes from Tejun Heo:
     "Two fixes.  One is to fix a long standing but which makes a cgroup
      destruction path O(N) where N is the total number of cgroups in the
      system.  The other is to fix a recent regression which can lead to
      oops on cgroup creation failure."
    * 'for-3.10-fixes' of git://git.kernel.org/pub/scm/linux/kernel/git/tj/cgroup:
      cgroup: fix to allow mounting a hierarchy which has been bound to
      cgroup: remove css_set_lock from cgroup_task_count()
```