

```
In [1]: import cv2

In [6]: from deepface import DeepFace

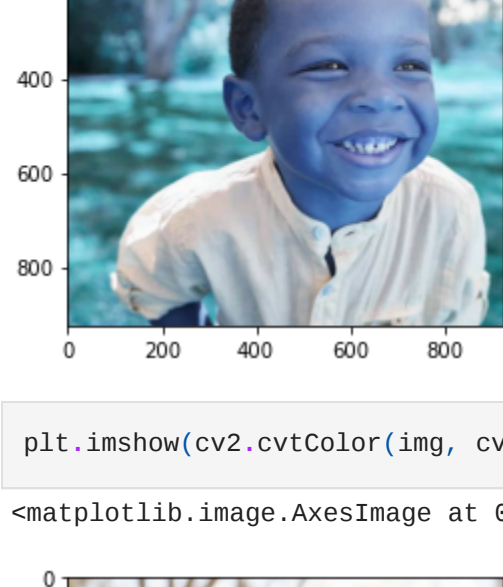
Directory C:\Users\aryan /.deepface created
Directory C:\Users\aryan /.deepface\weights created

In [7]: img = cv2.imread('happy-boy-smile.jpg')

In [8]: import matplotlib.pyplot as plt

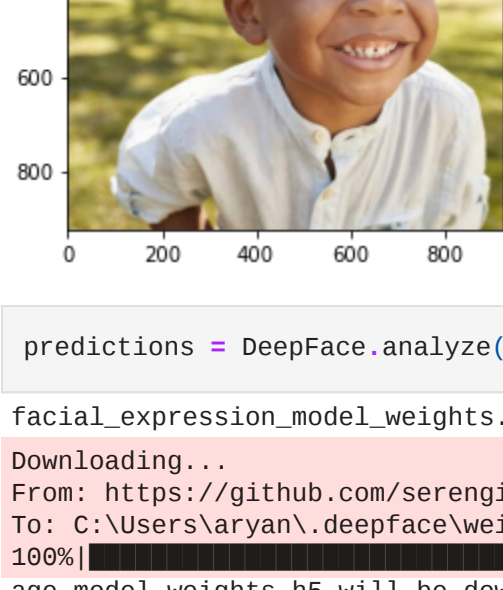
In [9]: plt.imshow(img) ##BGR

Out[9]: <matplotlib.image.AxesImage at 0x24155391520>



In [5]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)) ##BGR to RGB

Out[5]: <matplotlib.image.AxesImage at 0x24156c41490>



In [10]: predictions = DeepFace.analyze(img)

facial_expression_model_weights.h5 will be downloaded...
Downloading...
From: https://github.com/serenilj/deepface_models/releases/download/v1.0/facial_expression_model_weights.h5
To: C:\Users\aryan\.deepface\weights\facial_expression_model_weights.h5
100%|#####| 5.98M/5.98M [00:02<00:00, 2.47MB/s]
age_model_weights.h5 will be downloaded...
Downloading...
From: https://github.com/serenilj/deepface_models/releases/download/v1.0/age_model_weights.h5
To: C:\Users\aryan\.deepface\weights\age_model_weights.h5
100%|#####| 539M/539M [00:35<00:00, 15.0MB/s]
gender_model_weights.h5 will be downloaded...
Downloading...
From: https://github.com/serenilj/deepface_models/releases/download/v1.0/gender_model_weights.h5
To: C:\Users\aryan\.deepface\weights\gender_model_weights.h5
100%|#####| 537M/537M [01:17<00:00, 6.52MB/s]
race_model_single_batch.h5 will be downloaded...
Downloading...
From: https://github.com/serenilj/deepface_models/releases/download/v1.0/race_model_single_batch.h5
To: C:\Users\aryan\.deepface\weights\race_model_single_batch.h5
100%|#####| 537M/537M [01:17<00:00, 6.92MB/s]
Action: race: 100%|#####| 4/4 [00:03<00:00, 1.18it/s]
```

```
In [11]: predictions

Out[11]: {'emotion': {'angry': 2.9888040480702167e-06,
'disgust': 5.2857946681224e-12,
'fear': 0.0020419577902628097,
'happy': 99.97659384762906,
'sad': 0.001586112037799478,
'surprise': 0.0011964061544820644,
'neutral': 0.01860758646822193},
'dominant_emotion': 'happy',
'region': {'x': 442, 'y': 213, 'w': 418, 'h': 418},
'age': 30,
'gender': 'woman',
'race': {'asian': 0.003887299581478027,
'indian': 0.00631245932728537,
'black': 99.9132216000163,
'white': 1.530241422122523e-06,
'middle eastern': 8.454656766080304e-06,
'latino hispanic': 0.0205464002640262},
'dominant_race': 'black'}
```

```
In [12]: type(predictions)

Out[12]: dict

In [13]: predictions['dominant_emotion']

Out[13]: 'happy'
```

we are trying to draw a rectangle across the face

```
In [14]: faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

In [15]: gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#print(faceCascade.empty())
faces = faceCascade.detectMultiScale(gray,1.1,4)

#Draw a rectangle around the face
for(x,y,w,h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

```
In [16]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[16]: <matplotlib.image.AxesImage at 0x24104d1deb0>

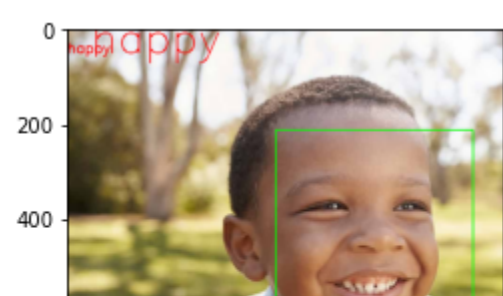

```

```
In [20]: font = cv2.FONT_HERSHEY_SIMPLEX

# Use putText() method for
# inserting text on video
cv2.putText(img,
            predictions['dominant_emotion'],
            (50,50),
            font, 3,
            (0, 0, 255),
            2,
            cv2.LINE_4) ;
```


```
In [21]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[21]: <matplotlib.image.AxesImage at 0x24104ef37f0>


```

```
In [22]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[22]: <matplotlib.image.AxesImage at 0x24104f53610>


```

```
In [23]: img = cv2.imread('1.jpg')

In [24]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[24]: <matplotlib.image.AxesImage at 0x24105142490>


```

```
In [25]: predictions = DeepFace.analyze(img)

Action: race: 100%|#####| 4/4 [00:02<00:00, 1.83it/s]
```

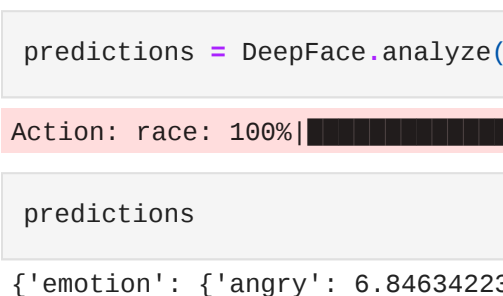
```
In [26]: predictions

Out[26]: {'emotion': {'angry': 6.846342235803604,
'disgust': 0.0031294679502025247,
'fear': 0.985193514602626,
'happy': 0.09778478415682012,
'sad': 29.653936824526978,
'surprise': 0.009203019412538277,
'neutral': 53.80426640232483},
'dominant_emotion': 'neutral',
'region': {'x': 395, 'y': 72, 'w': 287, 'h': 287},
'age': 30,
'gender': 'woman',
'race': {'asian': 99.90630747004658,
'indian': 0.03141463010471998,
'black': 1.5197158553982974e-06,
'white': 0.000352340647273376,
'middle eastern': 1.2957872162665608e-06,
'latino hispanic': 0.0538921152268376},
'dominant_race': 'asian'}
```

```
In [27]: img = cv2.imread('images.jpg')

In [28]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[28]: <matplotlib.image.AxesImage at 0x2410506b100>


```

```
In [29]: predictions = DeepFace.analyze(img)

Action: race: 100%|#####| 4/4 [00:01<00:00, 2.01it/s]
```

```
In [30]: predictions

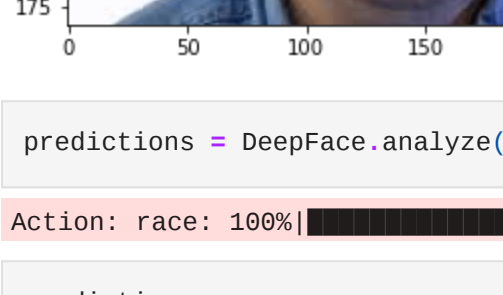
Out[30]: {'emotion': {'angry': 1.63613203587532,
'disgust': 2.096340457585678e-06,
'fear': 98.2707622680664,
'happy': 0.0443209934570575,
'sad': 3.168198920970414e-05,
'surprise': 0.03970868419855833,
'neutral': 2.99684214076501e-13},
'dominant_emotion': 'fear',
'region': {'x': 65, 'y': 11, 'w': 145, 'h': 145},
'age': 29,
'gender': 'man',
'race': {'asian': 5.110587924718857,
'indian': 11.027098447084427,
'black': 8.75202715306811,
'white': 24.37085211277008,
'middle eastern': 15.30234730243089,
'latino hispanic': 31.437084078780757},
'dominant_race': 'latino hispanic'}
```

```
In [31]: font = cv2.FONT_HERSHEY_SIMPLEX

# Use putText() method for
# inserting text on video
cv2.putText(img,
            predictions['dominant_emotion'],
            (50,50),
            font, 1,
            (0, 0, 255),
            2,
            cv2.LINE_4) ;
```

```
In [32]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[32]: <matplotlib.image.AxesImage at 0x241050cbb0>


```

```
In [33]: img = cv2.imread('images.jpg')

In [34]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[34]: <matplotlib.image.AxesImage at 0x24104fffe50>


```

```
In [35]: predictions = DeepFace.analyze(img)


Action: race: 100%|#####| 4/4 [00:01<00:00, 2.72it/s]
```

```
In [36]: font = cv2.FONT_HERSHEY_SIMPLEX

# Use putText() method for
# inserting text on video
cv2.putText(img,
            predictions['dominant_emotion'],
            (0,50),
            font, 1,
            (0, 0, 255),
            2,
            cv2.LINE_4) ;
```

```
In [37]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[37]: <matplotlib.image.AxesImage at 0x24109855340>


```

Real time video demo for Face Emotion Recognition

```
In [3]: import cv2

from deepface import DeepFace

faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(1)
# check if the webcam is opened correctly
if not cap.isOpened():
    cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise IOError("cannot open webcam")

while True:
    ret, frame = cap.read() ## read one image from a video
    predictions = DeepFace.analyze(frame, actions = ['emotion'])

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    #print(faceCascade.empty())
    faces = faceCascade.detectMultiScale(gray,1.1,4)

    # Draw a rectangle around the faces
    for(x,y,w,h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    font = cv2.FONT_HERSHEY_SIMPLEX

    #USE PUTTEXT() METHOD FOR
    #inserting text on video
    cv2.putText(frame,
                predictions['dominant_emotion'],
                (50,50),
                font, 1,
                (0, 0, 255),
                2,
                cv2.LINE_4)

    cv2.imshow("Demo video", frame)

    if cv2.waitKey(2) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()

-----
KeyboardInterrupt Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2104\3308763369.py in <module>
     15 ret, frame = cap.read() ## read one image from a video
--> 16
     17 result = DeepFace.analyze(frame, actions = ['emotion'])
     18
     19 gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

~\Anaconda3\lib\site-packages\deepface\DeepFace.py in analyze(img_path, actions, models, enforce_detection, detector_backend, prog_bar)
    391 img, region = functions.preprocess_face(img = img_path, target_size = (48, 48), grayscale = True, enforce_detection = enforce_detection, det
ector_backend = detector_backend, return_region = True)
--> 392
--> 393 emotion_predictions = models['emotion'].predict(img)[0,:]
    394
    395 sum_of_predictions = emotion_predictions.sum()

~\Anaconda3\lib\site-packages\keras\utils\traceback_utils.py in error_handler(*args, **kwargs)
    62 filtered_tb = None
    63 try:
--> 64     return fn(*args, **kwargs)
    65 except Exception as e: # pylint: disable=broad-exception
    66     filtered_tb = process_traceback_frames(e._traceback_...)

~\Anaconda3\lib\site-packages\keras\engine\training.py in predict(self, x, batch_size, verbose, steps, callbacks, max_queue_size, workers, use_multiprocessing)
   1449
   1450     data_adapter = data_adapter.get_data_handler(
   1451         stacklevel=2)
--> 1452     x, y, sample_weight, batch_size=batch_size,
   1453     batch_size=batch_size,
   1454     data_adapter=data_adapter)

~\Anaconda3\lib\site-packages\keras\engine\data_adapter.py in data_handler(*args, **kwargs)
   1397 if getattr(cls, "model"), _cluster_coordination, None):
   1398     return _ClusterCoordinatorDataHandler(*args, **kwargs)
--> 1399
   1400     return DataHandler(*args, **kwargs)
   1401

~\Anaconda3\lib\site-packages\keras\engine\data_adapter.py in _init(self, x, y, sample_weight, batch_size, steps_per_epoch, initial_epoch, epochs, shuffle, class_weight, max_queue_size, workers, use_multiprocessing, model, steps_per_execution, distribute)
   1167 self._insufficient_data = False
--> 1168
   1169 self._configure_dataset_and_inferred_steps(strategy, x, steps_per_epoch,
   1170 class_weight, distribute)
   1171

~\Anaconda3\lib\site-packages\keras\engine\data_adapter.py in configure_dataset_and_inferred_steps(**kwargs)
   1184 dataset = strategy.experimental_distribute_dataset(dataset)
   1185 self._dataset = dataset
--> 1186 self._validate_data_handler()
   1187
   1188 def enumerate_epochs(self):
   1189
   1190     def _validate_data_handler(self):
   1191         def _validate_data_handler(self):
   1192             try:
   1193                 result = gen_resource_variable_ops.read_variable_op(
   1194                     self._handle, self._dtype)
   1195                 _maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1196                 return result
   1197             except ValueError:
   1198                 raise ValueError("numpy() is only available when eager execution is enabled.")
   1199
   1200     def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1201         if not context.executing_eagerly():
   1202             return
   1203         self._handle = handle
   1204         self._dtype = dtype
   1205         self._result = result
   1206         self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1207         return result
   1208
   1209 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1210     if not context.executing_eagerly():
   1211         return
   1212     self._handle = handle
   1213     self._dtype = dtype
   1214     self._result = result
   1215     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1216     return result
   1217
   1218 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1219     if not context.executing_eagerly():
   1220         return
   1221     self._handle = handle
   1222     self._dtype = dtype
   1223     self._result = result
   1224     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1225     return result
   1226
   1227 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1228     if not context.executing_eagerly():
   1229         return
   1230     self._handle = handle
   1231     self._dtype = dtype
   1232     self._result = result
   1233     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1234     return result
   1235
   1236 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1237     if not context.executing_eagerly():
   1238         return
   1239     self._handle = handle
   1240     self._dtype = dtype
   1241     self._result = result
   1242     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1243     return result
   1244
   1245 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1246     if not context.executing_eagerly():
   1247         return
   1248     self._handle = handle
   1249     self._dtype = dtype
   1250     self._result = result
   1251     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1252     return result
   1253
   1254 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1255     if not context.executing_eagerly():
   1256         return
   1257     self._handle = handle
   1258     self._dtype = dtype
   1259     self._result = result
   1260     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1261     return result
   1262
   1263 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1264     if not context.executing_eagerly():
   1265         return
   1266     self._handle = handle
   1267     self._dtype = dtype
   1268     self._result = result
   1269     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1270     return result
   1271
   1272 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1273     if not context.executing_eagerly():
   1274         return
   1275     self._handle = handle
   1276     self._dtype = dtype
   1277     self._result = result
   1278     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1279     return result
   1280
   1281 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1282     if not context.executing_eagerly():
   1283         return
   1284     self._handle = handle
   1285     self._dtype = dtype
   1286     self._result = result
   1287     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1288     return result
   1289
   1290 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1291     if not context.executing_eagerly():
   1292         return
   1293     self._handle = handle
   1294     self._dtype = dtype
   1295     self._result = result
   1296     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1297     return result
   1298
   1299 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1300     if not context.executing_eagerly():
   1301         return
   1302     self._handle = handle
   1303     self._dtype = dtype
   1304     self._result = result
   1305     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1306     return result
   1307
   1308 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1309     if not context.executing_eagerly():
   1310         return
   1311     self._handle = handle
   1312     self._dtype = dtype
   1313     self._result = result
   1314     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1315     return result
   1316
   1317 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1318     if not context.executing_eagerly():
   1319         return
   1320     self._handle = handle
   1321     self._dtype = dtype
   1322     self._result = result
   1323     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1324     return result
   1325
   1326 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1327     if not context.executing_eagerly():
   1328         return
   1329     self._handle = handle
   1330     self._dtype = dtype
   1331     self._result = result
   1332     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1333     return result
   1334
   1335 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1336     if not context.executing_eagerly():
   1337         return
   1338     self._handle = handle
   1339     self._dtype = dtype
   1340     self._result = result
   1341     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1342     return result
   1343
   1344 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1345     if not context.executing_eagerly():
   1346         return
   1347     self._handle = handle
   1348     self._dtype = dtype
   1349     self._result = result
   1350     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1351     return result
   1352
   1353 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1354     if not context.executing_eagerly():
   1355         return
   1356     self._handle = handle
   1357     self._dtype = dtype
   1358     self._result = result
   1359     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1360     return result
   1361
   1362 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1363     if not context.executing_eagerly():
   1364         return
   1365     self._handle = handle
   1366     self._dtype = dtype
   1367     self._result = result
   1368     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1369     return result
   1370
   1371 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1372     if not context.executing_eagerly():
   1373         return
   1374     self._handle = handle
   1375     self._dtype = dtype
   1376     self._result = result
   1377     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1378     return result
   1379
   1380 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1381     if not context.executing_eagerly():
   1382         return
   1383     self._handle = handle
   1384     self._dtype = dtype
   1385     self._result = result
   1386     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1387     return result
   1388
   1389 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1390     if not context.executing_eagerly():
   1391         return
   1392     self._handle = handle
   1393     self._dtype = dtype
   1394     self._result = result
   1395     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1396     return result
   1397
   1398 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1399     if not context.executing_eagerly():
   1400         return
   1401     self._handle = handle
   1402     self._dtype = dtype
   1403     self._result = result
   1404     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1405     return result
   1406
   1407 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1408     if not context.executing_eagerly():
   1409         return
   1410     self._handle = handle
   1411     self._dtype = dtype
   1412     self._result = result
   1413     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1414     return result
   1415
   1416 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1417     if not context.executing_eagerly():
   1418         return
   1419     self._handle = handle
   1420     self._dtype = dtype
   1421     self._result = result
   1422     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1423     return result
   1424
   1425 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1426     if not context.executing_eagerly():
   1427         return
   1428     self._handle = handle
   1429     self._dtype = dtype
   1430     self._result = result
   1431     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1432     return result
   1433
   1434 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1435     if not context.executing_eagerly():
   1436         return
   1437     self._handle = handle
   1438     self._dtype = dtype
   1439     self._result = result
   1440     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1441     return result
   1442
   1443 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1444     if not context.executing_eagerly():
   1445         return
   1446     self._handle = handle
   1447     self._dtype = dtype
   1448     self._result = result
   1449     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1450     return result
   1451
   1452 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1453     if not context.executing_eagerly():
   1454         return
   1455     self._handle = handle
   1456     self._dtype = dtype
   1457     self._result = result
   1458     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1459     return result
   1460
   1461 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1462     if not context.executing_eagerly():
   1463         return
   1464     self._handle = handle
   1465     self._dtype = dtype
   1466     self._result = result
   1467     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1468     return result
   1469
   1470 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1471     if not context.executing_eagerly():
   1472         return
   1473     self._handle = handle
   1474     self._dtype = dtype
   1475     self._result = result
   1476     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1477     return result
   1478
   1479 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1480     if not context.executing_eagerly():
   1481         return
   1482     self._handle = handle
   1483     self._dtype = dtype
   1484     self._result = result
   1485     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1486     return result
   1487
   1488 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1489     if not context.executing_eagerly():
   1490         return
   1491     self._handle = handle
   1492     self._dtype = dtype
   1493     self._result = result
   1494     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1495     return result
   1496
   1497 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1498     if not context.executing_eagerly():
   1499         return
   1500     self._handle = handle
   1501     self._dtype = dtype
   1502     self._result = result
   1503     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1504     return result
   1505
   1506 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1507     if not context.executing_eagerly():
   1508         return
   1509     self._handle = handle
   1510     self._dtype = dtype
   1511     self._result = result
   1512     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1513     return result
   1514
   1515 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1516     if not context.executing_eagerly():
   1517         return
   1518     self._handle = handle
   1519     self._dtype = dtype
   1520     self._result = result
   1521     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1522     return result
   1523
   1524 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1525     if not context.executing_eagerly():
   1526         return
   1527     self._handle = handle
   1528     self._dtype = dtype
   1529     self._result = result
   1530     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1531     return result
   1532
   1533 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1534     if not context.executing_eagerly():
   1535         return
   1536     self._handle = handle
   1537     self._dtype = dtype
   1538     self._result = result
   1539     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1540     return result
   1541
   1542 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1543     if not context.executing_eagerly():
   1544         return
   1545     self._handle = handle
   1546     self._dtype = dtype
   1547     self._result = result
   1548     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1549     return result
   1550
   1551 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1552     if not context.executing_eagerly():
   1553         return
   1554     self._handle = handle
   1555     self._dtype = dtype
   1556     self._result = result
   1557     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1558     return result
   1559
   1560 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1561     if not context.executing_eagerly():
   1562         return
   1563     self._handle = handle
   1564     self._dtype = dtype
   1565     self._result = result
   1566     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1567     return result
   1568
   1569 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1570     if not context.executing_eagerly():
   1571         return
   1572     self._handle = handle
   1573     self._dtype = dtype
   1574     self._result = result
   1575     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1576     return result
   1577
   1578 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1579     if not context.executing_eagerly():
   1580         return
   1581     self._handle = handle
   1582     self._dtype = dtype
   1583     self._result = result
   1584     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1585     return result
   1586
   1587 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1588     if not context.executing_eagerly():
   1589         return
   1590     self._handle = handle
   1591     self._dtype = dtype
   1592     self._result = result
   1593     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1594     return result
   1595
   1596 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1597     if not context.executing_eagerly():
   1598         return
   1599     self._handle = handle
   1600     self._dtype = dtype
   1601     self._result = result
   1602     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1603     return result
   1604
   1605 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1606     if not context.executing_eagerly():
   1607         return
   1608     self._handle = handle
   1609     self._dtype = dtype
   1610     self._result = result
   1611     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1612     return result
   1613
   1614 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1615     if not context.executing_eagerly():
   1616         return
   1617     self._handle = handle
   1618     self._dtype = dtype
   1619     self._result = result
   1620     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1621     return result
   1622
   1623 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1624     if not context.executing_eagerly():
   1625         return
   1626     self._handle = handle
   1627     self._dtype = dtype
   1628     self._result = result
   1629     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1630     return result
   1631
   1632 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1633     if not context.executing_eagerly():
   1634         return
   1635     self._handle = handle
   1636     self._dtype = dtype
   1637     self._result = result
   1638     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1639     return result
   1640
   1641 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1642     if not context.executing_eagerly():
   1643         return
   1644     self._handle = handle
   1645     self._dtype = dtype
   1646     self._result = result
   1647     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1648     return result
   1649
   1650 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1651     if not context.executing_eagerly():
   1652         return
   1653     self._handle = handle
   1654     self._dtype = dtype
   1655     self._result = result
   1656     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1657     return result
   1658
   1659 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1660     if not context.executing_eagerly():
   1661         return
   1662     self._handle = handle
   1663     self._dtype = dtype
   1664     self._result = result
   1665     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1666     return result
   1667
   1668 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1669     if not context.executing_eagerly():
   1670         return
   1671     self._handle = handle
   1672     self._dtype = dtype
   1673     self._result = result
   1674     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1675     return result
   1676
   1677 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1678     if not context.executing_eagerly():
   1679         return
   1680     self._handle = handle
   1681     self._dtype = dtype
   1682     self._result = result
   1683     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1684     return result
   1685
   1686 def _maybe_set_handle_data_adapter(self, handle, dtype, result):
   1687     if not context.executing_eagerly():
   1688         return
   1689     self._handle = handle
   1690     self._dtype = dtype
   1691     self._result = result
   1692     self._maybe_set_handle_data_adapter(self._handle, self._dtype, result)
   1693     return result
   1694
   1695 def _maybe_set_handle
```