In [1]:
```
import cv2
```

In [6]:
```
from deepface import DeepFace
```

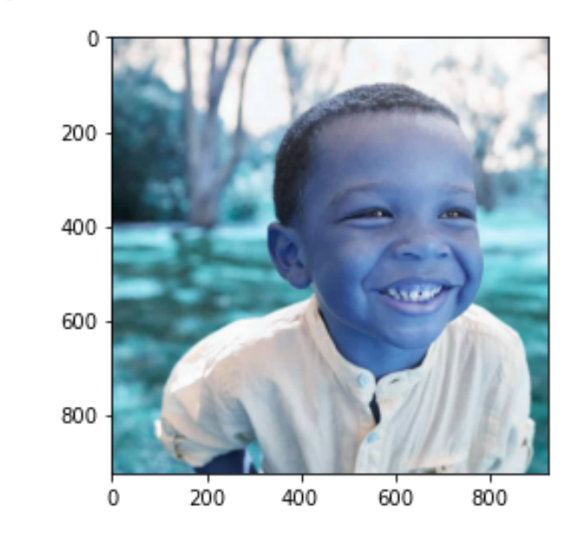
```
Directory  C:\Users\aryan /.deepface created
Directory  C:\Users\aryan /.deepface/weights created
```

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

In [8]:
```
import matplotlib.pyplot as plt
```

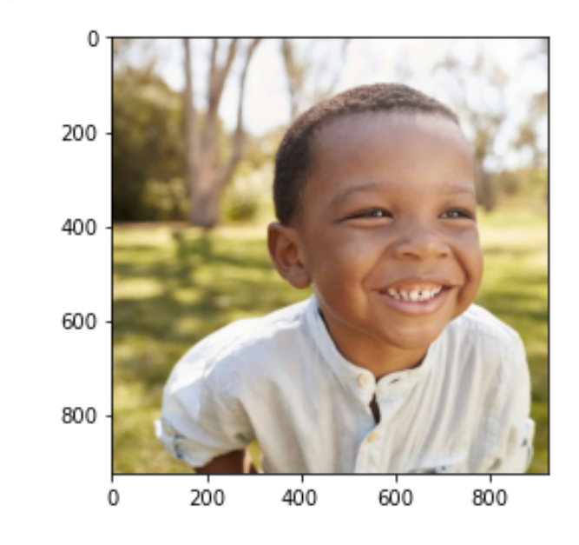
In [9]:
```
plt.imshow(img) ##BGR
```

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/serengil/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/serengil/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/serengil/deepface_models/releases/download/v1.0/gender_model_weights.h5
To: C:\Users\aryan\.deepface\weights\gender_model_weights.h5
100%|████████████████████████████████████████████████████████████████████████████████| 537M/537M [01:22<00:00, 6.52MB/s]
race_model_single_batch.h5 will be downloaded...
Downloading...
From: https://github.com/serengil/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': 2.9808040480702167e-06,
  'disgust': 5.285759666971214e-12,
  'fear': 0.0020419577902628097,
  'happy': 99.97650384762906,
  'sad': 0.0015866712637799428,
  'surprise': 0.0011964061544928644,
  'neutral': 0.01866750646822193},
 'dominant_emotion': 'happy',
 'region': {'x': 442, 'y': 213, 'w': 418, 'h': 418},
 'age': 30,
 'gender': 'Woman',
 'race': {'asian': 0.003887299581478027,
  'indian': 0.05631245932728537,
  'black': 99.9132216080163,
  'white': 1.530241422122523e-05,
  'middle eastern': 8.454656766008364e-06,
  'latino hispanic': 0.0265484062040262},
 '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

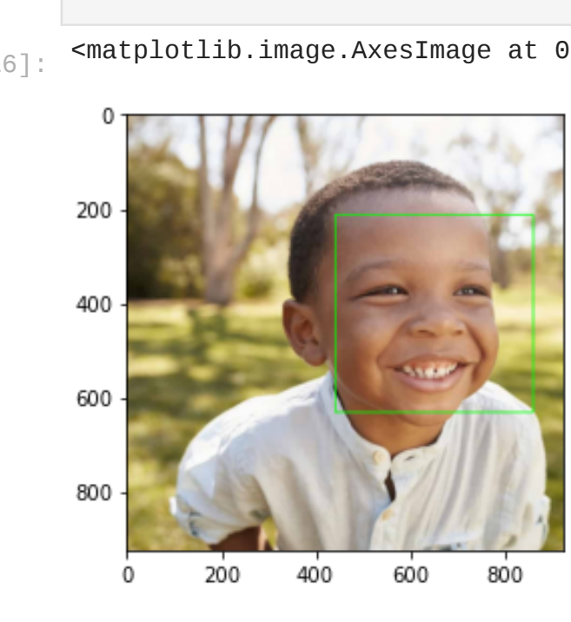
In [14]:
```
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
```

In [15]:
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  #print(face.Cascade.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>
```

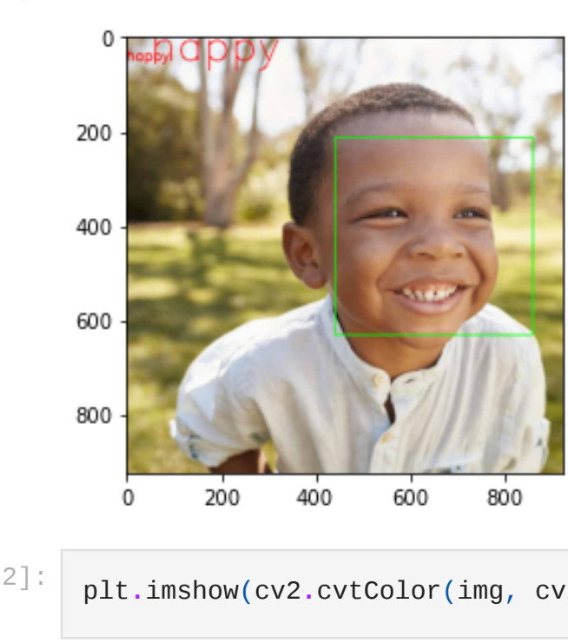


In [20]:
```
font = cv2.FONT_HERSHEY_SIMPLEX

  # 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) ;
```

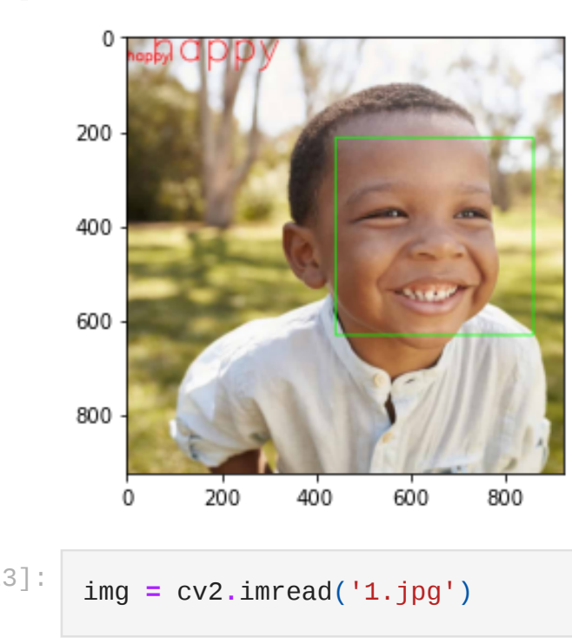
In [21]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

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



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

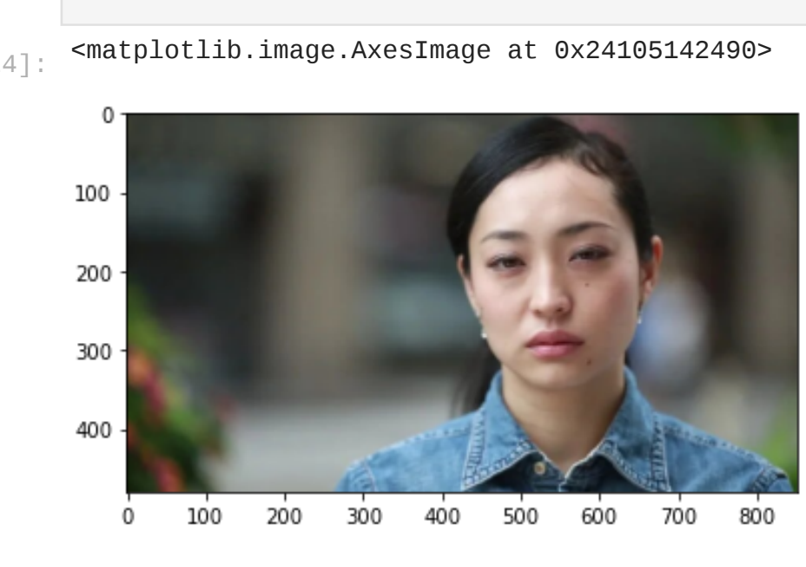
Out[22]:
```
<matplotlib.image.AxesImage at 0x24104f53610>
```



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]
```

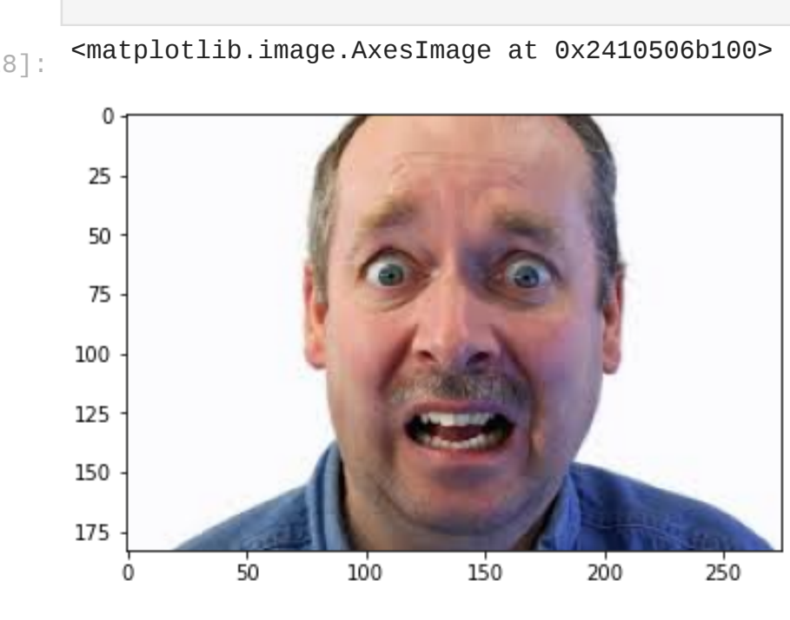
In [26]:
```
predictions
```

Out[26]:
```
{'emotion': {'angry': 6.846342235803604,
  'disgust': 0.0031294679502025247,
  'fear': 9.585193544626236,
  'happy': 0.09778478415682912,
  'sad': 29.653936624526978,
  'surprise': 0.009353019413538277,
  'neutral': 53.80426049232483},
 'dominant_emotion': 'neutral',
 'region': {'x': 395, 'y': 72, 'w': 287, 'h': 287},
 'age': 30,
 'gender': 'Woman',
 'race': {'asian': 99.90630747004658,
  'indian': 0.03141463010471998,
  'black': 1.5197155853982974e-05,
  'white': 0.008352946047273376,
  'middle eastern': 1.2957872162665608e-05,
  'latino hispanic': 0.0538921152258378},
 '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
```

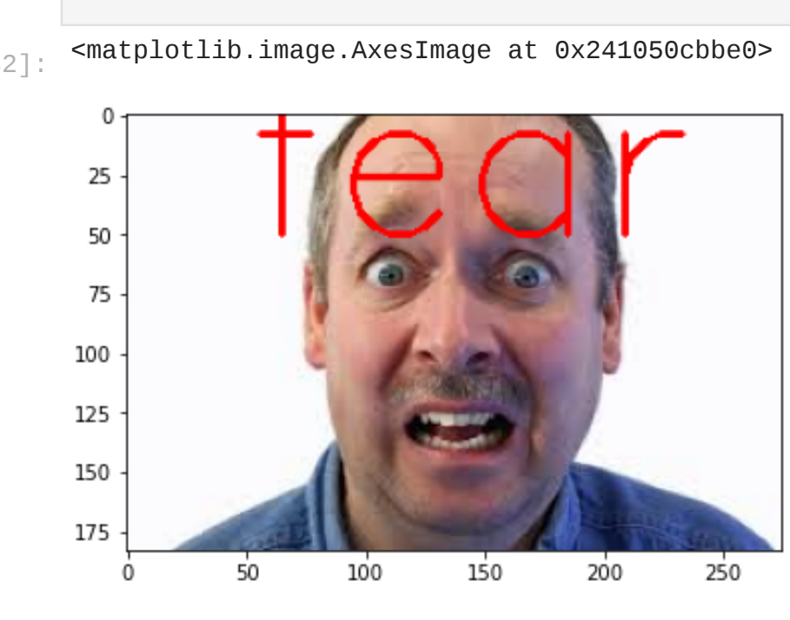
Out[30]:
```
{'emotion': {'angry': 1.636173203587532,
  'disgust': 2.0963403457585628e-06,
  'fear': 98.2797622680664,
  'happy': 0.04432209534570575,
  'sad': 3.168198929870414e-05,
  'surprise': 0.03970868419855833,
  'neutral': 2.99684214076501e-11},
 'dominant_emotion': 'fear',
 'region': {'x': 65, 'y': 11, 'w': 145, 'h': 145},
 'age': 29,
 'gender': 'Man',
 'race': {'asian': 5.110587924718857,
  'indian': 11.027098447084427,
  'black': 8.752027153968811,
  'white': 24.37085211277008,
  'middle eastern': 19.30234730243683,
  'latino hispanic': 31.437084078788757},
 '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, 3,
           (0, 0, 255),
           2,
           cv2.LINE_4) ;
```

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

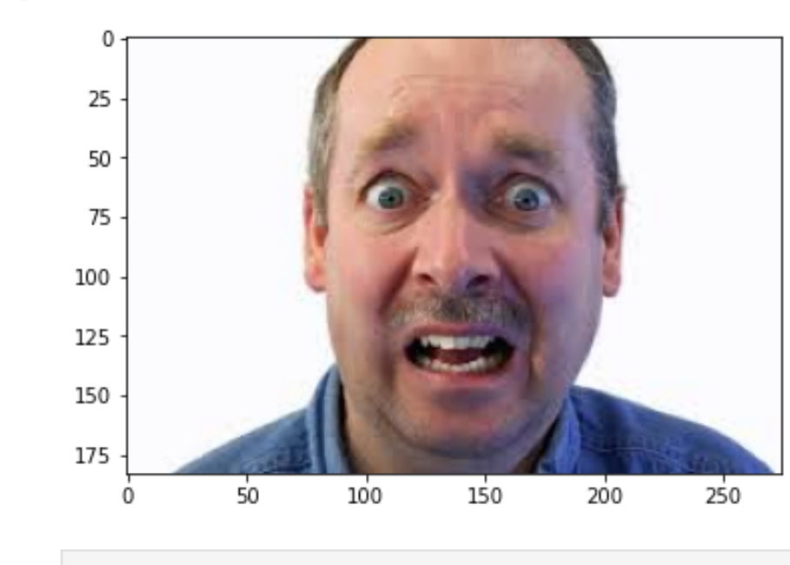
Out[32]:
```
<matplotlib.image.AxesImage at 0x241050cbbe0>
```



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 [1]:
```
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

    result = 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,
                result['dominant_emotion'],
                (50,50),
                font, 3,
                (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_2104/3308761369.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-except
     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)
   1949               stacklevel=2)
   1950 
-> 1951       data_handler = data_adapter.get_data_handler(
   1952           x=x,
   1953           batch_size=batch_size,

~\anaconda3\lib\site-packages\keras\engine\data_adapter.py in get_data_handler(*args, **kwargs)
   1397   if getattr(kwargs["model"], "_cluster_coordinator", None):
   1398     return _ClusterCoordinatorDataHandler(*args, **kwargs)
-> 1399   return DataHandler(*args, **kwargs)
   1400 
   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_que
ue_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(***failed resolving arguments***)
   1184       dataset = strategy.experimental_distribute_dataset(dataset)
   1185     self._dataset = dataset
-> 1186     self._validate_data_handler()
   1187 
   1188   def enumerate_epochs(self):

~\anaconda3\lib\site-packages\keras\engine\data_adapter.py in _validate_data_handler(self)
   1328   def _validate_data_handler(self):
   1329     # TODO(b/152094471): Support this with DistIter.get_next_as_optional.
-> 1330     if self._steps_per_execution.numpy().item(
   1331     ) > 1 and self._inferred_steps is None:
   1332       raise ValueError(

~\anaconda3\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py in numpy(self)
    672   def numpy(self):
    673     if context.executing_eagerly():
--> 674       return self.read_value().numpy()
    675     raise NotImplementedError(
    676         "numpy() is only available when eager execution is enabled.")

~\anaconda3\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py in read_value(self)
    747     """
    748     with ops.name_scope("Read"):
--> 749       value = self._read_variable_op()
    750     # Return an identity so it can get placed on whatever device the context
    751     # specifies instead of the device where the variable is.

~\anaconda3\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py in _read_variable_op(self)
    726           result = read_and_set_handle()
    727     else:
--> 728       result = read_and_set_handle()
    729 
    730     if not context.executing_eagerly():

~\anaconda3\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py in read_and_set_handle()
    716 
    717     def read_and_set_handle():
--> 718       result = gen_resource_variable_ops.read_variable_op(
    719           self.handle, self._dtype)
    720       _maybe_set_handle_data(self._dtype, self.handle, result)

~\anaconda3\lib\site-packages\tensorflow\python\ops\gen_resource_variable_ops.py in read_variable_op(resource, dtype, name)
    476   if tld.is_eager:
    477     try:
--> 478       _result = pywrap_tfe.TFE_Py_FastPathExecute(
    479         _ctx, "ReadVariableOp", name, resource, "dtype", dtype)
    480       return _result

KeyboardInterrupt: 
```

In [ ]:

In [ ]: