siamese-network-using-face-dataset

May 13, 2023

1 Siamese Network using face dataset

[ ]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

Load data:

[ ]:
```
!unzip /content/gdrive/Shareddrives/data_face_HUS/data_face_HUS.zip -d "/
↪content/"
```

Import all neeeded libraries and preprocess

[ ]:
```
import os
import glob
import cv2
import numpy as np

folder_path = 'data_face_HUS/*/*'

# Find all files in subfolders
files = glob.glob(folder_path, recursive=True)

# Categorize files based on subfolder name
categories = {}

for file in files:
    subfolder = os.path.basename(os.path.dirname(file))
    if subfolder in categories:
        categories[subfolder].append(file)
    else:
        categories[subfolder] = [file]

X_train_label = []
X_train_list = []
X_test_label = []
X_test_list = []
```

```
for category, files in categories.items():
    # print(category)
    # for file in files:
    #     print('\t', file)
    train_list = glob.glob('data_face_HUS/'+category+'/*')
    for name in train_list:
        img = cv2.imread(name)
        if img is not None:
            img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            img = cv2.resize(img,(96,96))
            X_train_list.append(img)
            X_train_label.append(category)
```

Divide into train and test

[ ]:
```
X_train = np.array(X_train_list)
y_train = np.array(X_train_label)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train,␣
 ↪test_size=0.2, random_state = 42)
```

Build model CNN

[ ]:
```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Input
from tensorflow.keras.models import Model
from tensorflow.keras import backend as K

inp = Input(shape=(96,96,1))
x = Conv2D(filters = 8, kernel_size = 3, activation = 'relu')(inp)
x = MaxPooling2D(pool_size = (2,2))(x)

x = Flatten()(x)
x = Dense(units = 32, activation = 'relu')(x)
x = Dense(units = 2)(x)

cnn = Model(inputs = inp, outputs = x)
img1 = Input(shape = (96,96,1))
img2 = Input(shape = (96,96,1))

f1 = cnn(img1)
f2 = cnn(img2)

d = K.sqrt(K.sum(K.square(f1 - f2),axis = 1, keepdims = True))

model = Model(inputs = [img1,img2], outputs = d)
model.summary()
cnn.summary()
```

```
def loss(y_true, y_pred):
  proba = K.exp(-K.square(y_pred))
  return -K.mean(y_true * K.log(proba) + (1-y_true) * K.log(1-proba))

def loss1(y_true, y_pred):
  return K.mean(y_true * K.square(y_pred) + (1-y_true) * K.square(K.maximum(1.0␣
 ↪- y_pred, 0)))

model.compile(optimizer = 'adam', loss = loss1)
```

```
Model: "model_3"
__________________________________________________________________________________
__________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================
==================
 input_5 (InputLayer)           [(None, 96, 96, 1)]  0           []

 input_6 (InputLayer)           [(None, 96, 96, 1)]  0           []

 model_2 (Functional)           (None, 2)            565682
['input_5[0][0]',
'input_6[0][0]']

 tf.math.subtract_1 (TFOpLambda  (None, 2)           0
['model_2[0][0]',
 )
'model_2[1][0]']

 tf.math.square_1 (TFOpLambda)  (None, 2)            0
['tf.math.subtract_1[0][0]']

 tf.math.reduce_sum_1 (TFOpLamb  (None, 1)           0
['tf.math.square_1[0][0]']
 da)

 tf.math.maximum_1 (TFOpLambda)  (None, 1)           0
['tf.math.reduce_sum_1[0][0]']

 tf.math.sqrt_1 (TFOpLambda)    (None, 1)            0
['tf.math.maximum_1[0][0]']

==================================================================================
==================
Total params: 565,682
Trainable params: 565,682
Non-trainable params: 0
```

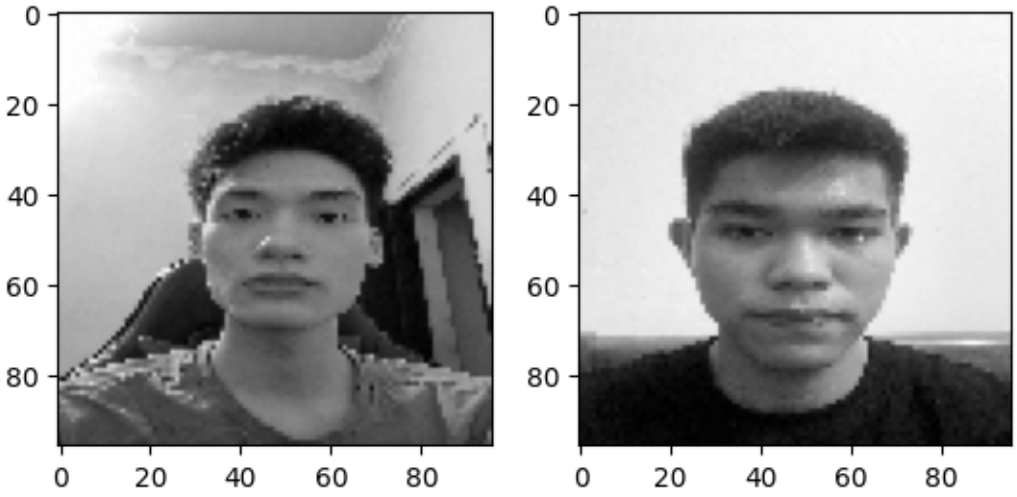
```
_________________________________________________________________
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 96, 96, 1)]       0

 conv2d_1 (Conv2D)           (None, 94, 94, 8)         80

 max_pooling2d_1 (MaxPooling  (None, 47, 47, 8)        0
 2D)

 flatten_1 (Flatten)         (None, 17672)             0

 dense_2 (Dense)             (None, 32)                565536

 dense_3 (Dense)             (None, 2)                 66

=================================================================
Total params: 565,682
Trainable params: 565,682
Non-trainable params: 0
_________________________________________________________________
```

Make all pairs or other strategies; some innovation here

[ ]:
```python
import numpy as np
from matplotlib import pyplot as plt

def generator(X,y,k = 8):
  unique_labels = np.unique(y)

  while True:
    X1 = []
    X2 = []
    y_batch = []
    for label in unique_labels:
      label_idx = np.where(y == label)[0]
      other_labels = set(unique_labels) - {label}

      for i in range(k):
        i1 = np.random.choice(label_idx)
        i2 = np.random.choice(label_idx)
        # i1 must be different from i2
        # while i1 == i2:
        #   i2 = np.random.choice(label_idx)
        # create positive example
```

```
        X1.append(X[i1][:,:,None])
        X2.append(X[i2][:,:,None])
        y_batch.append(1.0)

        # create negative example
        i1 = np.random.choice(label_idx)
        my_label = np.random.choice(list(other_labels))
        i2 = np.random.choice(list(np.where(y == my_label)[0]))
        X1.append(X[i1][:,:,None])
        X2.append(X[i2][:,:,None])
        y_batch.append(0.0)
    yield [np.array(X1) / 255., np.array(X2) / 255.], np.array(y_batch)

# For testing
for pair, y in generator(X_test, y_test):
  print('Batch size: ', len(y))
  idx = np.random.choice(range(len(y)))
  print(pair[0][idx].shape)
  print('Pair label:', y[idx])
  plt.subplot(121)
  plt.imshow(pair[0][idx].reshape(96,96), cmap = 'gray')
  plt.subplot(122)
  plt.imshow(pair[1][idx].reshape(96,96), cmap = 'gray')

  break
```

```
Batch size:  848
(96, 96, 1)
Pair label: 0.0
```



Fit model

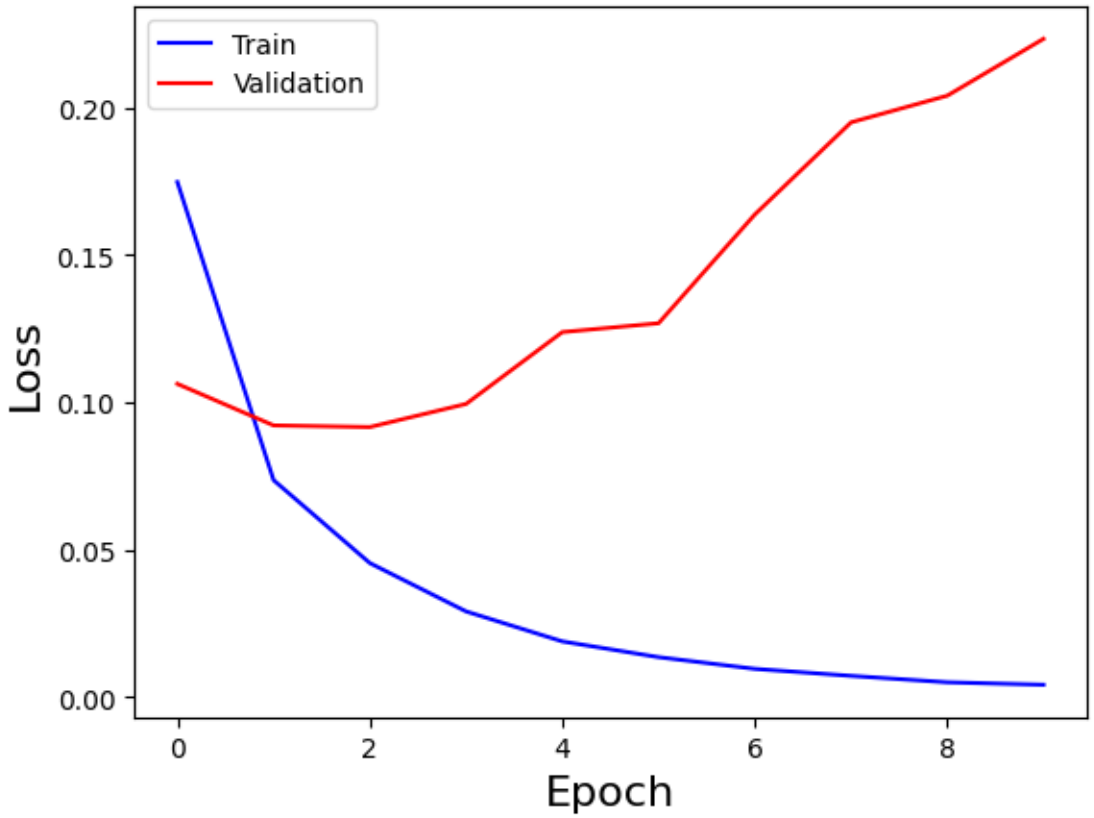
[ ]:
```
history = model.fit(generator(X_train, y_train, k = 32),
                    steps_per_epoch = 10,
                    epochs = 10,
                    validation_data = generator(X_test, y_test, k = 8),
                    validation_steps = 5)
```

```
Epoch 1/10
10/10 [==============================] - 11s 951ms/step - loss: 0.1748 -
val_loss: 0.1063
Epoch 2/10
10/10 [==============================] - 10s 1s/step - loss: 0.0736 - val_loss:
0.0922
Epoch 3/10
10/10 [==============================] - 8s 891ms/step - loss: 0.0455 -
val_loss: 0.0917
Epoch 4/10
10/10 [==============================] - 10s 1s/step - loss: 0.0291 - val_loss:
0.0995
Epoch 5/10
10/10 [==============================] - 9s 918ms/step - loss: 0.0189 -
val_loss: 0.1239
Epoch 6/10
10/10 [==============================] - 8s 907ms/step - loss: 0.0136 -
val_loss: 0.1269
Epoch 7/10
10/10 [==============================] - 9s 930ms/step - loss: 0.0096 -
val_loss: 0.1638
Epoch 8/10
10/10 [==============================] - 8s 878ms/step - loss: 0.0072 -
val_loss: 0.1951
Epoch 9/10
10/10 [==============================] - 9s 914ms/step - loss: 0.0050 -
val_loss: 0.2041
Epoch 10/10
10/10 [==============================] - 9s 926ms/step - loss: 0.0042 -
val_loss: 0.2234
```

Visualize learning process

[ ]:
```
plt.plot(history.history['loss'], label = 'Train', c = 'b')
plt.plot(history.history['val_loss'], label = 'Validation', c = 'r')
plt.legend()
plt.xlabel('Epoch', fontsize = 16)
plt.ylabel('Loss', fontsize = 16)
```

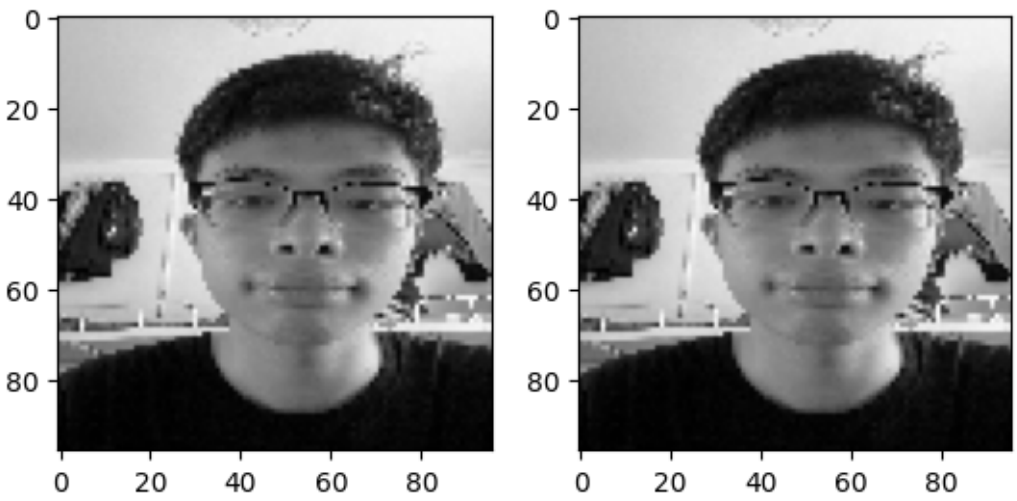
[ ]:
```
Text(0, 0.5, 'Loss')
```



[ ]:
```
for pair, y in generator(X_test, y_test):
  y_pred = model.predict(pair)
  print('Batch_size: ', len(y))
  idx = np.random.choice(range(len(y)))
  print('Pair label:', y[idx])
  print('Distance:', y_pred[idx])

  f1 = cnn(pair[0])
  f2 = cnn(pair[1])
  d = np.sqrt(np.sum((f1-f2)**2,axis = 1, keepdims = True))
  print('Distance by features:', d[idx])

  plt.subplot(121)
  plt.imshow(pair[0][idx].reshape(96,96),cmap = 'gray')
  plt.subplot(122)
  plt.imshow(pair[1][idx].reshape(96,96),cmap = 'gray')
  break
```

```
27/27 [==============================] - 0s 5ms/step
Batch_size:  848
Pair label: 1.0
```

```
Distance: [0.]
Distance by features: [0.]
```



Visualize new feature space

[ ]:
```
f = cnn.predict(X_test/255.)
p = plt.scatter(f[:,0],f[:,1], c = y_test, s=1)
plt.colorbar(p)
```

```
7/7 [==============================] - 0s 7ms/step
```

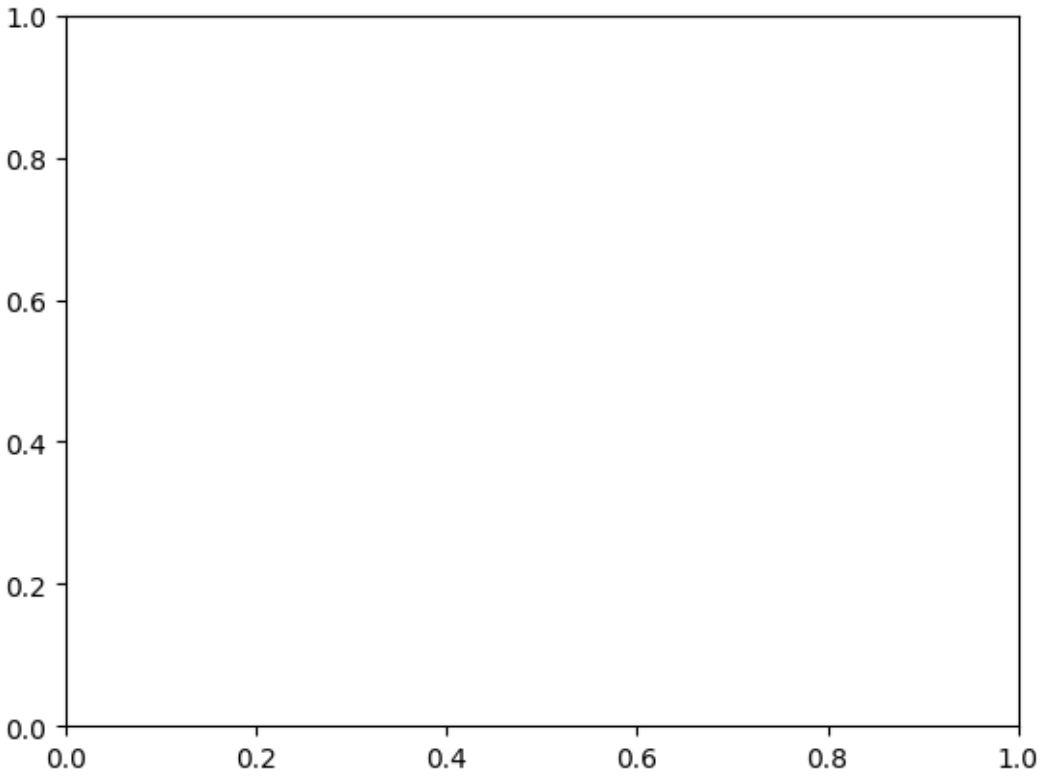
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_axes.py in␣
 ↪_parse_scatter_color_args(c, edgecolors, kwargs, xsize, get_next_color_func)
   4438             try:  # Is 'c' acceptable as PathCollection facecolors?
-> 4439                 colors = mcolors.to_rgba_array(c)
   4440             except (TypeError, ValueError) as err:

/usr/local/lib/python3.10/dist-packages/matplotlib/colors.py in to_rgba_array(c␣
 ↪alpha)
    486     else:
--> 487         rgba = np.array([to_rgba(cc) for cc in c])
    488

/usr/local/lib/python3.10/dist-packages/matplotlib/colors.py in <listcomp>(.0)
    486     else:
--> 487         rgba = np.array([to_rgba(cc) for cc in c])
    488
```

```
/usr/local/lib/python3.10/dist-packages/matplotlib/colors.py in to_rgba(c, alpha)
    298     if rgba is None:  # Suppress exception chaining of cache lookup␣
 ↪failure.
--> 299         rgba = _to_rgba_no_colorcycle(c, alpha)
    300         try:

/usr/local/lib/python3.10/dist-packages/matplotlib/colors.py in␣
 ↪_to_rgba_no_colorcycle(c, alpha)
    373             return c, c, c, alpha if alpha is not None else 1.
--> 374         raise ValueError(f"Invalid RGBA argument: {orig_c!r}")
    375     # turn 2-D array into 1-D array

ValueError: Invalid RGBA argument: 'NguyenDinhTrungKien_20002141'

The above exception was the direct cause of the following exception:

ValueError                                Traceback (most recent call last)
<ipython-input-14-9ccde014cddc> in <cell line: 2>()
      1 f = cnn.predict(X_test/255.)
----> 2 p = plt.scatter(f[:,0],f[:,1], c = y_test, s=1)
      3 plt.colorbar(p)

/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py in scatter(x, y, s␣
 ↪c, marker, cmap, norm, vmin, vmax, alpha, linewidths, edgecolors,␣
 ↪plotnonfinite, data, **kwargs)
   2860         vmin=None, vmax=None, alpha=None, linewidths=None, *,
   2861         edgecolors=None, plotnonfinite=False, data=None, **kwargs):
-> 2862     __ret = gca().scatter(
   2863         x, y, s=s, c=c, marker=marker, cmap=cmap, norm=norm,
   2864         vmin=vmin, vmax=vmax, alpha=alpha, linewidths=linewidths,

/usr/local/lib/python3.10/dist-packages/matplotlib/__init__.py in inner(ax,␣
 ↪data, *args, **kwargs)
   1440     def inner(ax, *args, data=None, **kwargs):
   1441         if data is None:
-> 1442             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1443
   1444         bound = new_sig.bind(ax, *args, **kwargs)

/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_axes.py in␣
 ↪scatter(self, x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths,␣
 ↪edgecolors, plotnonfinite, **kwargs)
   4600             orig_edgecolor = kwargs.get('edgecolor', None)
   4601         c, colors, edgecolors = \
-> 4602             self._parse_scatter_color_args(
   4603                 c, edgecolors, kwargs, x.size,
```

```
  4604                 get_next_color_func=self._get_patches_for_fill.
↪get_next_color)

/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_axes.py in␣
↪_parse_scatter_color_args(c, edgecolors, kwargs, xsize, get_next_color_func)
  4446                     # Both the mapping *and* the RGBA conversion failed␣
↪pretty
  4447                     # severe failure => one may appreciate a verbose␣
↪feedback.
-> 4448                     raise ValueError(
  4449                         f"'c' argument must be a color, a sequence of␣
↪colors, "
  4450                         f"or a sequence of numbers, not {c!r}") from er

ValueError: 'c' argument must be a color, a sequence of colors, or a sequence o␣
↪numbers, not array(['NguyenDinhTrungKien_20002141',␣
↪'NguyenTrieuVuong_20002182',
       'nguyenvanhung_20002130', 'NguyenCongVu_20002181',
       'HoangManhHung_20002128', 'PhamHoangAn_20002102',
       'NguyenTrieuVuong_20002182', 'nguyenvanhung_20002130',
       'PhamHoangAn_20002102', 'NguyenHoangAnh_20002103',
       'Nguyễn Thị Hà 20002120', 'DamVanVu_20002180',
       'DinhKhaVy_20002183', 'VuMinhNgoc_20002148',
       'LeHongThach_20002162', 'TranMinhHoang_20002127',
       'Nguyễn Thị Hà 20002120', 'NguyenDinhHung_20002129',
       'HoangMinhQuan_20002156', 'NguyenKieuTrang_20002168',
       'TruongTuanAnh_20002107', 'NguyenDInhHai_20002122',
       'NguyenMinhDuc_20002116', 'Nguyễn Thị Hà 20002120',
       'ChuPhamDinhTu_20002173', 'LuuTheTrung_20002170',
       'NguyenTienDat_20002113', 'DuongDucThinh_20002166',
       'NguyenHoangAnh_20002103', 'TranMinhHoang_20002127',
       'ChuPhamDinhTu_20002173', 'DoMinhTuan_20002175',
       'PhamHoangAn_20002102', 'NguyenDInhHai_20002122',
       'LeHongThach_20002162', 'PhungPhucHau_20002123',
       'TranKimPhuong_20002154', 'NguyenDinhQuang_20002155',
       'DinhKhaVy_20002183', 'TranQuocKhang_20002138',
       'NguyenBaKien_20002140', 'DaoAnhDuong_20002112',
       'Nguyen_Manh_Trung_20002171', 'Nguyễn Thị Hà 20002120',
       'DuongXuanDuc_20002114', 'Nguyễn Thị Thanh_20002164',
       'Nguyễn Thị Thanh_20002164', 'HoangManhHung_20002128',
       'NguyenThePhong_20002150', 'NguyenTienDat_20002113',
       'Nguyen_Manh_Trung_20002171', 'NguyenHoangAnh_20002103',
       'HoHaiPhong_20002149', 'DuongDucThinh_20002166',
       'NguyenTrieuVuong_20002182', 'NguyenDInhHai_20002122',
       'ViAnhQuan_20002157', 'PhamHoangAn_20002102', 'DinhKhaVy_20002183',
       'TraafnThanhPhong_20002152', 'DoMinhTuan_20002175',
       'PhamHoangAn_20002102', 'DuongXuanDuc_20002114',
       'TruongNgonNghia_20002147', 'NguyenDinhTrungKien_20002141',
```

```
'TranQuocKhang_20002138', 'LuuTheTrung_20002170',
'NguyenThanhLong_20002142', 'HoangManhHung_20002128',
'NguyenMinhDuc_20002116', 'NguyenCongVu_20002181',
'NguyenThanhLong_20002142', 'LuuTheTrung_20002170',
'DamVanVu_20002180', 'DuongXuanDuc_20002114',
'TraafnThanhPhong_20002152', 'TruongTuanAnh_20002107',
'NguyenAnhTuan_20002176', 'TranAnhMinh_20002144',
'HoangHuuHieu_20002125', 'TranQuocDuc_20002117',
'VuMinhNgoc_20002148', 'LeTanHiep_20002124',
'NguyenDinhQuang_20002155', 'TranQuocKhang_20002138',
'ToAnhTu_20002174', 'NguyenDinhQuang_20002155',
'NguyenMinhDuc_20002116', 'TranAnhMinh_20002144',
'NguyenBaKien_20002140', 'TranQuocKhang_20002138',
'LuyenThiQuyen_20002158', 'NguyenThePhong_20002150',
'VuMinhNgoc_20002148', 'nguyenvanhung_20002130',
'NguyenTruongDanh_20002110', 'TruongTuanAnh_20002107',
'DuongDucThinh_20002166', 'TruongTuanAnh_20002107',
'Nguyễn Thị Hà 20002120', 'TranAnhMinh_20002144',
'Nguyễn Thị Thanh_20002164', 'HoangHuuHieu_20002125',
'DuongXuanDuc_20002114', 'DamVanVu_20002180',
'HoHaiPhong_20002149', 'Nguyễn Thị Thanh_20002164',
'DuongXuanDuc_20002114', 'NguyenKieuTrang_20002168',
'NguyenTienDat_20002113', 'PhamTienAnh_20002106',
'ChuPhamDinhTu_20002173', 'DinhKhaVy_20002183',
'DamVanVu_20002180', 'VuMinhNgoc_20002148',
'TruongNgonNghia_20002147', 'Nguyễn Thị Thanh_20002164',
'TranQuocDuc_20002117', 'LuyenThiQuyen_20002158',
'DinhKhaVy_20002183', 'NguyenDinhHung_20002129',
'DuongDucThinh_20002166', 'nguyenvanhung_20002130',
'TranKimPhuong_20002154', 'NguyenThuHa_20002121',
'LuuTheTrung_20002170', 'HoangMinhQuan_20002156',
'ChuPhamDinhTu_20002173', 'NguyenDinhQuang_20002155',
'TruongTuanAnh_20002107', 'NguyenDinhQuang_20002155',
'NguyenTienDat_20002113', 'NguyenCongDung_20002111',
'NguyenMinhDuc_20002116', 'NguyenDInhHai_20002122',
'DuongXuanDuc_20002114', 'DaoAnhDuong_20002112',
'TranQuocKhang_20002138', 'NguyenTruongDanh_20002110',
'ViAnhQuan_20002157', 'LuuTheTrung_20002170',
'DuongXuanDuc_20002114', 'NguyenKieuTrang_20002168',
'NguyenCongVu_20002181', 'TranMinhHoang_20002127',
'TranBinhHuong_20002137', 'TranQuocDuc_20002117',
'DoPhuongNam_20002145', 'TranQuocDuc_20002117',
'LeHongThach_20002162', 'DuongXuanDuc_20002114',
'Nguyễn Thị Hà 20002120', 'ChuPhamDinhTu_20002173',
'DuongDucThinh_20002166', 'TranQuocDuc_20002117',
'DuongXuanDuc_20002114', 'NguyenDinhQuang_20002155',
'LuyenThiQuyen_20002158', 'NguyenThuHa_20002121',
'NguyenDinhTrungKien_20002141', 'NguyenDinhTrungKien_20002141',
```

```
'TranQuocDuc_20002117', 'Nguyen_Manh_Trung_20002171',
'nguyenvanhung_20002130', 'TranQuocDuc_20002117',
'NguyenDinhQuang_20002155', 'NguyenThanhLong_20002142',
'TranBinhHuong_20002137', 'DuongXuanDuc_20002114',
'PhungPhucHau_20002123', 'HoangHuuHieu_20002125',
'NguyenCongVu_20002181', 'Nguyễn Thị Thanh_20002164',
'NguyenDinhHung_20002129', 'HoangMinhQuan_20002156',
'NguyenDinhQuang_20002155', 'NguyenXuanTuanAnh_20002105',
'TranBinhHuong_20002137', 'nguyenvanhung_20002130',
'nguyenvanhung_20002130', 'HoangManhHung_20002128',
'DuongDucThinh_20002166', 'LuyenThiQuyen_20002158',
'PhamHoangAn_20002102', 'DaoAnhDuong_20002112',
'HoHaiPhong_20002149', 'NguyenDinhQuang_20002155',
'TranBinhHuong_20002137', 'NguyenTienDat_20002113',
'NguyenDInhHai_20002122', 'ViAnhQuan_20002157',
'NguyenDinhHung_20002129', 'NguyenTienDat_20002113',
'TraafnThanhPhong_20002152', 'NguyenThePhong_20002150',
'PhungPhucHau_20002123', 'ChuPhamDinhTu_20002173',
'TruongTuanAnh_20002107', 'ChuPhamDinhTu_20002173',
'TranQuocKhang_20002138', 'HoangHuuHieu_20002125',
'NguyenBaKien_20002140', 'TranAnhMinh_20002144',
'Nguyễn Thị Thanh_20002164', 'NguyenCongDung_20002111',
'TranBinhHuong_20002137', 'TranMinhHoang_20002127',
'ViAnhQuan_20002157', 'TranQuocDuc_20002117',
'LuyenThiQuyen_20002158', 'NguyenXuanTuanAnh_20002105',
'PhamTienAnh_20002106', 'Nguyễn Thị Hà 20002120',
'DuongDucThinh_20002166', 'HoHaiPhong_20002149',
'NguyenDinhQuang_20002155', 'DinhKhaVy_20002183',
'NguyenThePhong_20002150', 'DuongDucThinh_20002166',
'LuyenThiQuyen_20002158'], dtype='<U28')
```



Save model

[ ]:
```
cnn.save('cnn_loss1.h5')
```

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.
```

Load model and test

[ ]:
```
from tensorflow.keras.models import load_model
m = load_model('cnn_loss1.h5')

f1 = m.predict(X_test / 255.)
p = plt.scatter(f1[:,0],f1[:,1],c=y_test,s=1)
plt.colorbar(p)
```

```
WARNING:tensorflow:No training configuration found in the save file, so the
model was *not* compiled. Compile it manually.
```

```
7/7 [==============================] - 0s 2ms/step
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
```

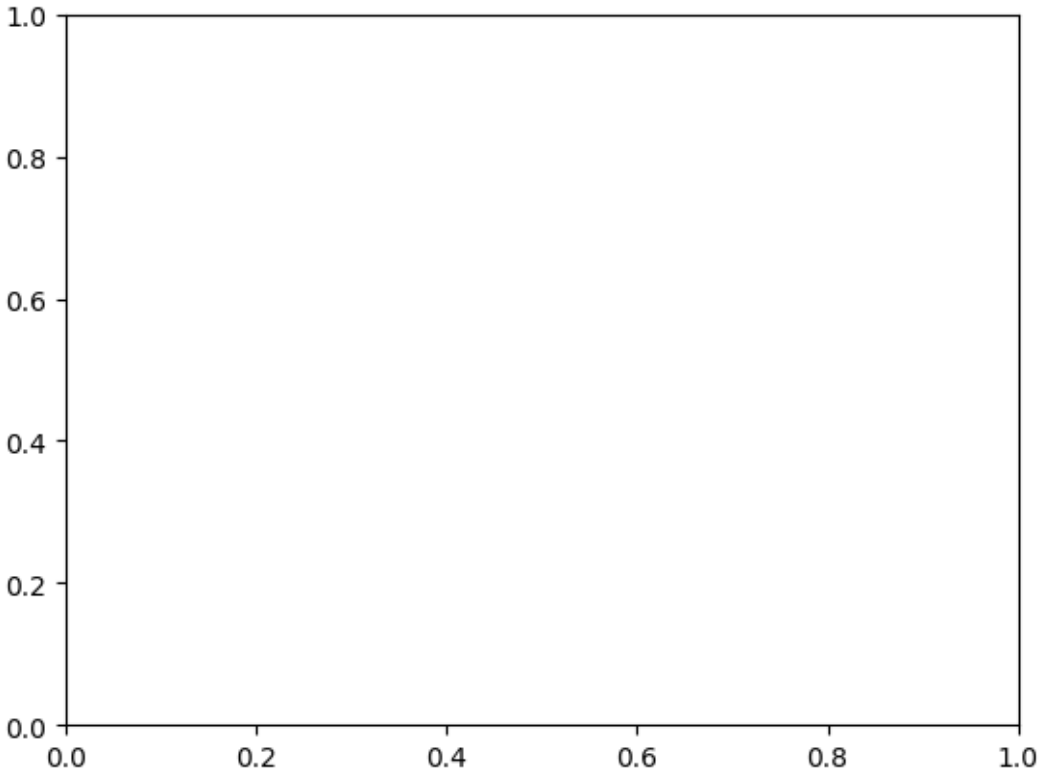
```
/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_axes.py in␣
 ↪_parse_scatter_color_args(c, edgecolors, kwargs, xsize, get_next_color_func)
   4438             try:  # Is 'c' acceptable as PathCollection facecolors?
-> 4439                 colors = mcolors.to_rgba_array(c)
   4440             except (TypeError, ValueError) as err:

/usr/local/lib/python3.10/dist-packages/matplotlib/colors.py in to_rgba_array(c␣
 ↪alpha)
    486     else:
--> 487         rgba = np.array([to_rgba(cc) for cc in c])
    488

/usr/local/lib/python3.10/dist-packages/matplotlib/colors.py in <listcomp>(.0)
    486     else:
--> 487         rgba = np.array([to_rgba(cc) for cc in c])
    488

/usr/local/lib/python3.10/dist-packages/matplotlib/colors.py in to_rgba(c, alpha)
    298     if rgba is None:  # Suppress exception chaining of cache lookup␣
 ↪failure.
--> 299         rgba = _to_rgba_no_colorcycle(c, alpha)
    300         try:

/usr/local/lib/python3.10/dist-packages/matplotlib/colors.py in␣
 ↪_to_rgba_no_colorcycle(c, alpha)
    373             return c, c, c, alpha if alpha is not None else 1.
--> 374         raise ValueError(f"Invalid RGBA argument: {orig_c!r}")
    375     # turn 2-D array into 1-D array

ValueError: Invalid RGBA argument: 'NguyenDinhTrungKien_20002141'

The above exception was the direct cause of the following exception:

ValueError                                Traceback (most recent call last)
<ipython-input-16-a810ea0dddf5> in <cell line: 5>()
      3
      4 f1 = m.predict(X_test / 255.)
----> 5 p = plt.scatter(f1[:,0],f1[:,1],c=y_test,s=1)
      6 plt.colorbar(p)

/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py in scatter(x, y, s␣
 ↪c, marker, cmap, norm, vmin, vmax, alpha, linewidths, edgecolors,␣
 ↪plotnonfinite, data, **kwargs)
   2860         vmin=None, vmax=None, alpha=None, linewidths=None, *,
   2861         edgecolors=None, plotnonfinite=False, data=None, **kwargs):
-> 2862     __ret = gca().scatter(

   2863         x, y, s=s, c=c, marker=marker, cmap=cmap, norm=norm,
```

```
   2864         vmin=vmin, vmax=vmax, alpha=alpha, linewidths=linewidths,

/usr/local/lib/python3.10/dist-packages/matplotlib/__init__.py in inner(ax,␣
 ↪data, *args, **kwargs)
   1440     def inner(ax, *args, data=None, **kwargs):
   1441         if data is None:
-> 1442             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1443
   1444         bound = new_sig.bind(ax, *args, **kwargs)

/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_axes.py in␣
 ↪scatter(self, x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths,␣
 ↪edgecolors, plotnonfinite, **kwargs)
   4600             orig_edgecolor = kwargs.get('edgecolor', None)
   4601         c, colors, edgecolors = \
-> 4602             self._parse_scatter_color_args(
   4603                 c, edgecolors, kwargs, x.size,
   4604                 get_next_color_func=self._get_patches_for_fill.
 ↪get_next_color)

/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_axes.py in␣
 ↪_parse_scatter_color_args(c, edgecolors, kwargs, xsize, get_next_color_func)
   4446                     # Both the mapping *and* the RGBA conversion failed␣
 ↪pretty
   4447                     # severe failure => one may appreciate a verbose␣
 ↪feedback.
-> 4448                     raise ValueError(
   4449                         f"'c' argument must be a color, a sequence of␣
 ↪colors, "
   4450                         f"or a sequence of numbers, not {c!r}") from er

ValueError: 'c' argument must be a color, a sequence of colors, or a sequence o␣
 ↪numbers, not array(['NguyenDinhTrungKien_20002141',␣
 ↪'NguyenTrieuVuong_20002182',
       'nguyenvanhung_20002130', 'NguyenCongVu_20002181',
       'HoangManhHung_20002128', 'PhamHoangAn_20002102',
       'NguyenTrieuVuong_20002182', 'nguyenvanhung_20002130',
       'PhamHoangAn_20002102', 'NguyenHoangAnh_20002103',
       'Nguyễn Thị Hà 20002120', 'DamVanVu_20002180',
       'DinhKhaVy_20002183', 'VuMinhNgoc_20002148',
       'LeHongThach_20002162', 'TranMinhHoang_20002127',
       'Nguyễn Thị Hà 20002120', 'NguyenDinhHung_20002129',
       'HoangMinhQuan_20002156', 'NguyenKieuTrang_20002168',
       'TruongTuanAnh_20002107', 'NguyenDInhHai_20002122',
       'NguyenMinhDuc_20002116', 'Nguyễn Thị Hà 20002120',
       'ChuPhamDinhTu_20002173', 'LuuTheTrung_20002170',
       'NguyenTienDat_20002113', 'DuongDucThinh_20002166',
```

```
'NguyenHoangAnh_20002103', 'TranMinhHoang_20002127',
'ChuPhamDinhTu_20002173', 'DoMinhTuan_20002175',
'PhamHoangAn_20002102', 'NguyenDInhHai_20002122',
'LeHongThach_20002162', 'PhungPhucHau_20002123',
'TranKimPhuong_20002154', 'NguyenDinhQuang_20002155',
'DinhKhaVy_20002183', 'TranQuocKhang_20002138',
'NguyenBaKien_20002140', 'DaoAnhDuong_20002112',
'Nguyen_Manh_Trung_20002171', 'Nguyễn Thị Hà 20002120',
'DuongXuanDuc_20002114', 'Nguyễn Thị Thanh_20002164',
'Nguyễn Thị Thanh_20002164', 'HoangManhHung_20002128',
'NguyenThePhong_20002150', 'NguyenTienDat_20002113',
'Nguyen_Manh_Trung_20002171', 'NguyenHoangAnh_20002103',
'HoHaiPhong_20002149', 'DuongDucThinh_20002166',
'NguyenTrieuVuong_20002182', 'NguyenDInhHai_20002122',
'ViAnhQuan_20002157', 'PhamHoangAn_20002102', 'DinhKhaVy_20002183',
'TraafnThanhPhong_20002152', 'DoMinhTuan_20002175',
'PhamHoangAn_20002102', 'DuongXuanDuc_20002114',
'TruongNgonNghia_20002147', 'NguyenDinhTrungKien_20002141',
'TranQuocKhang_20002138', 'LuuTheTrung_20002170',
'NguyenThanhLong_20002142', 'HoangManhHung_20002128',
'NguyenMinhDuc_20002116', 'NguyenCongVu_20002181',
'NguyenThanhLong_20002142', 'LuuTheTrung_20002170',
'DamVanVu_20002180', 'DuongXuanDuc_20002114',
'TraafnThanhPhong_20002152', 'TruongTuanAnh_20002107',
'NguyenAnhTuan_20002176', 'TranAnhMinh_20002144',
'HoangHuuHieu_20002125', 'TranQuocDuc_20002117',
'VuMinhNgoc_20002148', 'LeTanHiep_20002124',
'NguyenDinhQuang_20002155', 'TranQuocKhang_20002138',
'ToAnhTu_20002174', 'NguyenDinhQuang_20002155',
'NguyenMinhDuc_20002116', 'TranAnhMinh_20002144',
'NguyenBaKien_20002140', 'TranQuocKhang_20002138',
'LuyenThiQuyen_20002158', 'NguyenThePhong_20002150',
'VuMinhNgoc_20002148', 'nguyenvanhung_20002130',
'NguyenTruongDanh_20002110', 'TruongTuanAnh_20002107',
'DuongDucThinh_20002166', 'TruongTuanAnh_20002107',
'Nguyễn Thị Hà 20002120', 'TranAnhMinh_20002144',
'Nguyễn Thị Thanh_20002164', 'HoangHuuHieu_20002125',
'DuongXuanDuc_20002114', 'DamVanVu_20002180',
'HoHaiPhong_20002149', 'Nguyễn Thị Thanh_20002164',
'DuongXuanDuc_20002114', 'NguyenKieuTrang_20002168',
'NguyenTienDat_20002113', 'PhamTienAnh_20002106',
'ChuPhamDinhTu_20002173', 'DinhKhaVy_20002183',
'DamVanVu_20002180', 'VuMinhNgoc_20002148',
'TruongNgonNghia_20002147', 'Nguyễn Thị Thanh_20002164',
'TranQuocDuc_20002117', 'LuyenThiQuyen_20002158',
'DinhKhaVy_20002183', 'NguyenDinhHung_20002129',
'DuongDucThinh_20002166', 'nguyenvanhung_20002130',
'TranKimPhuong_20002154', 'NguyenThuHa_20002121',
```

```
'LuuTheTrung_20002170', 'HoangMinhQuan_20002156',
'ChuPhamDinhTu_20002173', 'NguyenDinhQuang_20002155',
'TruongTuanAnh_20002107', 'NguyenDinhQuang_20002155',
'NguyenTienDat_20002113', 'NguyenCongDung_20002111',
'NguyenMinhDuc_20002116', 'NguyenDInhHai_20002122',
'DuongXuanDuc_20002114', 'DaoAnhDuong_20002112',
'TranQuocKhang_20002138', 'NguyenTruongDanh_20002110',
'ViAnhQuan_20002157', 'LuuTheTrung_20002170',
'DuongXuanDuc_20002114', 'NguyenKieuTrang_20002168',
'NguyenCongVu_20002181', 'TranMinhHoang_20002127',
'TranBinhHuong_20002137', 'TranQuocDuc_20002117',
'DoPhuongNam_20002145', 'TranQuocDuc_20002117',
'LeHongThach_20002162', 'DuongXuanDuc_20002114',
'Nguyễn Thị Hà 20002120', 'ChuPhamDinhTu_20002173',
'DuongDucThinh_20002166', 'TranQuocDuc_20002117',
'DuongXuanDuc_20002114', 'NguyenDinhQuang_20002155',
'LuyenThiQuyen_20002158', 'NguyenThuHa_20002121',
'NguyenDinhTrungKien_20002141', 'NguyenDinhTrungKien_20002141',
'TranQuocDuc_20002117', 'Nguyen_Manh_Trung_20002171',
'nguyenvanhung_20002130', 'TranQuocDuc_20002117',
'NguyenDinhQuang_20002155', 'NguyenThanhLong_20002142',
'TranBinhHuong_20002137', 'DuongXuanDuc_20002114',
'PhungPhucHau_20002123', 'HoangHuuHieu_20002125',
'NguyenCongVu_20002181', 'Nguyễn Thị Thanh_20002164',
'NguyenDinhHung_20002129', 'HoangMinhQuan_20002156',
'NguyenDinhQuang_20002155', 'NguyenXuanTuanAnh_20002105',
'TranBinhHuong_20002137', 'nguyenvanhung_20002130',
'nguyenvanhung_20002130', 'HoangManhHung_20002128',
'DuongDucThinh_20002166', 'LuyenThiQuyen_20002158',
'PhamHoangAn_20002102', 'DaoAnhDuong_20002112',
'HoHaiPhong_20002149', 'NguyenDinhQuang_20002155',
'TranBinhHuong_20002137', 'NguyenTienDat_20002113',
'NguyenDInhHai_20002122', 'ViAnhQuan_20002157',
'NguyenDinhHung_20002129', 'NguyenTienDat_20002113',
'TraafnThanhPhong_20002152', 'NguyenThePhong_20002150',
'PhungPhucHau_20002123', 'ChuPhamDinhTu_20002173',
'TruongTuanAnh_20002107', 'ChuPhamDinhTu_20002173',
'TranQuocKhang_20002138', 'HoangHuuHieu_20002125',
'NguyenBaKien_20002140', 'TranAnhMinh_20002144',
'Nguyễn Thị Thanh_20002164', 'NguyenCongDung_20002111',
'TranBinhHuong_20002137', 'TranMinhHoang_20002127',
'ViAnhQuan_20002157', 'TranQuocDuc_20002117',
'LuyenThiQuyen_20002158', 'NguyenXuanTuanAnh_20002105',
'PhamTienAnh_20002106', 'Nguyễn Thị Hà 20002120',
'DuongDucThinh_20002166', 'HoHaiPhong_20002149',
'NguyenDinhQuang_20002155', 'DinhKhaVy_20002183',
'NguyenThePhong_20002150', 'DuongDucThinh_20002166',
```

```
       'LuyenThiQuyen_20002158'], dtype='<U28')
```



Visualize negative distance and positive distance

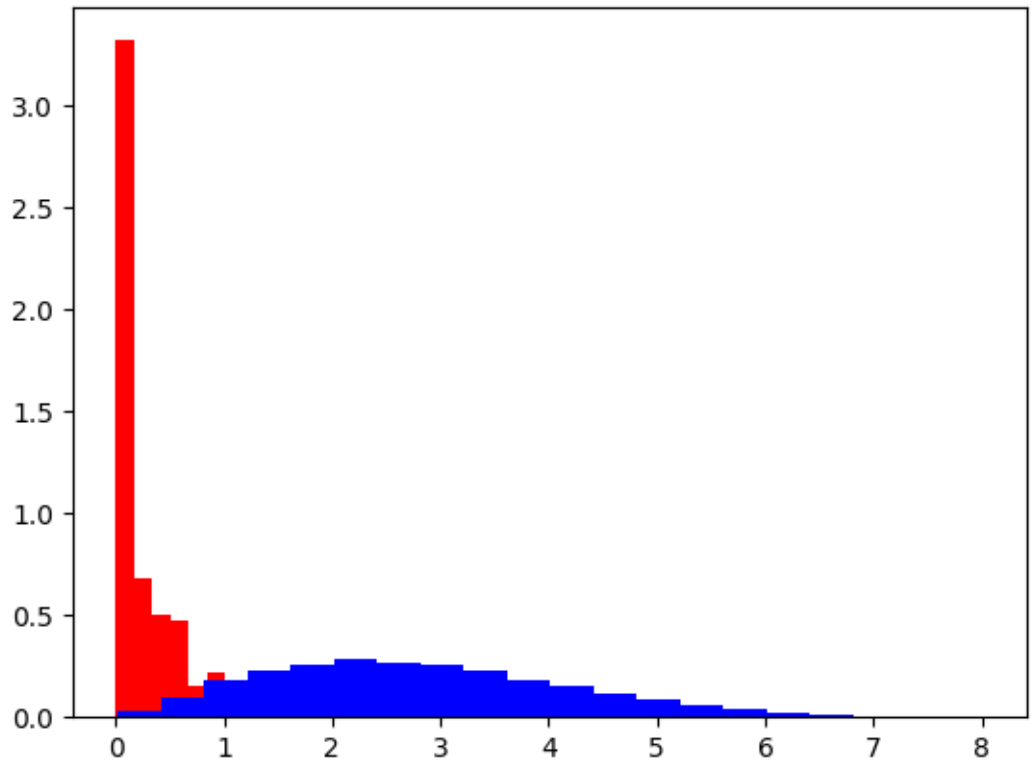
[ ]:
```
i = 0
y_true = []
y_pred = []
for pair,y in generator(X_test,y_test):
  f1 = cnn(pair[0])
  f2 = cnn(pair[1])
  d = np.sqrt(np.sum((f1 - f2)**2, axis = 1, keepdims = True))
  y_pred +=list(d.ravel())
  y_true +=list(y)
  i+=1
  if i>500:
    break
```

Plot histogram

[ ]:
```
y_pred = np.array(y_pred)
y_true = np.array(y_true)
```

```
positive_distances = y_pred[y_true == 1]
negative_distances = y_pred[y_true == 0]

plt.hist(positive_distances, color = 'r', density = True, bins = 20)
plt.hist(negative_distances, color = 'b', density = True, bins = 20)
```

[ ]:
```
(array([0.02634509, 0.10031626, 0.17699612, 0.22444551, 0.25864591,
        0.28979592, 0.26608888, 0.25474781, 0.22733087, 0.17763216,
        0.15015638, 0.11891207, 0.0917426 , 0.05465692, 0.03801606,
        0.02003263, 0.01449747, 0.00660688, 0.00369797, 0.00104815]),
 array([0.01694522, 0.41667154, 0.81639785, 1.21612418, 1.61585057,
        2.01557684, 2.41530323, 2.81502962, 3.21475577, 3.61448216,
        4.01420832, 4.41393471, 4.8136611 , 5.21338749, 5.61311388,
        6.01284027, 6.41256666, 6.81229258, 7.21201897, 7.61174536,
        8.01147175]),
 <BarContainer object of 20 artists>)
```



Check report using sklearn

[ ]:
```
thresh = 0.5
y_pred_ = y_pred < thresh
y_pred_.astype('uint8')
```

```
from sklearn.metrics import classification_report
print(classification_report(y_true, y_pred_))
```

```
              precision    recall  f1-score   support

         0.0       0.80      0.98      0.88    212424
         1.0       0.98      0.75      0.85    212424

    accuracy                           0.87    424848
   macro avg       0.89      0.87      0.87    424848
weighted avg       0.89      0.87      0.87    424848
```