```
!pip install segmentation-models-pytorch
!pip install -U git+https://github.com/albumentations-
team/albumentations
!pip install --upgrade opencv-contrib-python
```

```
Requirement already satisfied: segmentation-models-pytorch in
/usr/local/lib/python3.12/dist-packages (0.5.0)
,Requirement already satisfied: huggingface-hub>=0.24 in
/usr/local/lib/python3.12/dist-packages (from segmentation-models-
pytorch) (0.34.4)
,Requirement already satisfied: numpy>=1.19.3 in
/usr/local/lib/python3.12/dist-packages (from segmentation-models-
pytorch) (2.0.2)
,Requirement already satisfied: pillow>=8 in
/usr/local/lib/python3.12/dist-packages (from segmentation-models-
pytorch) (11.3.0)
,Requirement already satisfied: safetensors>=0.3.1 in
/usr/local/lib/python3.12/dist-packages (from segmentation-models-
pytorch) (0.6.2)
,Requirement already satisfied: timm>=0.9 in
/usr/local/lib/python3.12/dist-packages (from segmentation-models-
pytorch) (1.0.19)
,Requirement already satisfied: torch>=1.8 in
/usr/local/lib/python3.12/dist-packages (from segmentation-models-
pytorch) (2.8.0+cu126)
,Requirement already satisfied: torchvision>=0.9 in
/usr/local/lib/python3.12/dist-packages (from segmentation-models-
pytorch) (0.23.0+cu126)
,Requirement already satisfied: tqdm>=4.42.1 in
/usr/local/lib/python3.12/dist-packages (from segmentation-models-
pytorch) (4.67.1)
,Requirement already satisfied: filelock in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.24-
>segmentation-models-pytorch) (3.19.1)
,Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.24-
>segmentation-models-pytorch) (2025.3.0)
,Requirement already satisfied: packaging>=20.9 in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.24-
>segmentation-models-pytorch) (25.0)
,Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.24-
>segmentation-models-pytorch) (6.0.2)
,Requirement already satisfied: requests in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.24-
>segmentation-models-pytorch) (2.32.4)
,Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.24-
>segmentation-models-pytorch) (4.15.0)
,Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in
```

```
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.24-
>segmentation-models-pytorch) (1.1.9)
,Requirement already satisfied: setuptools in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (75.2.0)
,Requirement already satisfied: sympy>=1.13.3 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (1.13.3)
,Requirement already satisfied: networkx in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (3.5)
,Requirement already satisfied: jinja2 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (3.1.6)
,Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (12.6.77)
,Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77
in /usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (12.6.77)
,Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (12.6.80)
,Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (9.10.2.21)
,Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (12.6.4.1)
,Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (11.3.0.4)
,Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (10.3.7.77)
,Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (11.7.1.2)
,Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (12.5.4.2)
,Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (0.7.1)
,Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (2.27.3)
,Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
```

```
>segmentation-models-pytorch) (12.6.77)
,Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (12.6.85)
,Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (1.11.1.6)
,Requirement already satisfied: triton==3.4.0 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.8-
>segmentation-models-pytorch) (3.4.0)
,Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3-
>torch>=1.8->segmentation-models-pytorch) (1.3.0)
,Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.12/dist-packages (from jinja2->torch>=1.8-
>segmentation-models-pytorch) (3.0.2)
,Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests->huggingface-
hub>=0.24->segmentation-models-pytorch) (3.4.3)
,Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests->huggingface-
hub>=0.24->segmentation-models-pytorch) (3.10)
,Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests->huggingface-
hub>=0.24->segmentation-models-pytorch) (2.5.0)
,Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests->huggingface-
hub>=0.24->segmentation-models-pytorch) (2025.8.3)
,Collecting git+https://github.com/albumentations-team/albumentations
,  Cloning https://github.com/albumentations-team/albumentations to
/tmp/pip-req-build-z8rz45gk
,  Running command git clone --filter=blob:none --quiet
https://github.com/albumentations-team/albumentations /tmp/pip-req-
build-z8rz45gk
,  Resolved https://github.com/albumentations-team/albumentations to
commit 66212d77a44927a29d6a0e81621d3c27afbd929c
,  Installing build dependencies ... ents to build wheel ... etadata
(pyproject.toml) ... ent already satisfied: numpy>=1.24.4 in
/usr/local/lib/python3.12/dist-packages (from albumentations==2.0.8)
(2.0.2)
,Requirement already satisfied: scipy>=1.10.0 in
/usr/local/lib/python3.12/dist-packages (from albumentations==2.0.8)
(1.16.1)
,Requirement already satisfied: PyYAML in
/usr/local/lib/python3.12/dist-packages (from albumentations==2.0.8)
(6.0.2)
,Requirement already satisfied: pydantic>=2.9.2 in
/usr/local/lib/python3.12/dist-packages (from albumentations==2.0.8)
(2.11.7)
```

```
,Requirement already satisfied: albucore==0.0.28 in
/usr/local/lib/python3.12/dist-packages (from albumentations==2.0.8)
(0.0.28)
,Requirement already satisfied: opencv-python-headless>=4.9.0.80 in
/usr/local/lib/python3.12/dist-packages (from albumentations==2.0.8)
(4.12.0.88)
,Requirement already satisfied: stringzilla>=3.10.4 in
/usr/local/lib/python3.12/dist-packages (from albucore==0.0.28-
>albumentations==2.0.8) (3.12.6)
,Requirement already satisfied: simsimd>=5.9.2 in
/usr/local/lib/python3.12/dist-packages (from albucore==0.0.28-
>albumentations==2.0.8) (6.5.1)
,Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.12/dist-packages (from pydantic>=2.9.2-
>albumentations==2.0.8) (0.7.0)
,Requirement already satisfied: pydantic-core==2.33.2 in
/usr/local/lib/python3.12/dist-packages (from pydantic>=2.9.2-
>albumentations==2.0.8) (2.33.2)
,Requirement already satisfied: typing-extensions>=4.12.2 in
/usr/local/lib/python3.12/dist-packages (from pydantic>=2.9.2-
>albumentations==2.0.8) (4.15.0)
,Requirement already satisfied: typing-inspection>=0.4.0 in
/usr/local/lib/python3.12/dist-packages (from pydantic>=2.9.2-
>albumentations==2.0.8) (0.4.1)
,Requirement already satisfied: opencv-contrib-python in
/usr/local/lib/python3.12/dist-packages (4.12.0.88)
,Requirement already satisfied: numpy<2.3.0,>=2 in
/usr/local/lib/python3.12/dist-packages (from opencv-contrib-python)
(2.0.2)
```

Download Full Dataset

Market-1501 dataset : https://www.kaggle.com/pengcw1/market-1501

```
!git clone https://github.com/parth1620/Person-Re-Id-Dataset

Cloning into 'Person-Re-Id-Dataset'...
,remote: Enumerating objects: 12942, done.ote: Counting objects: 100%
(12942/12942), done.ote: Compressing objects: 100% (12942/12942),
done.ote: Total 12942 (delta 0), reused 12942 (delta 0), pack-reused 0
(from 0)
```

Deep Learning with PyTorch : Siamese Network

Author: Eda AYDIN

Siamese Network

[Eng]

A Siamese Network is a type of neural network architecture that is used for tasks that involve finding similarities or differences between two input samples. The network consists of two identical subnetworks that share the same set of weights and are trained simultaneously.

The basic idea behind a Siamese Network is to learn a similarity metric between two input samples. In other words, the network is trained to output a high value when the two input samples are similar and a low value when they are dissimilar. This makes it useful for a variety of applications, such as image or text similarity matching, face recognition, and signature verification.

One of main advantages of a Siamese Network is that it can be trained with very few examples, making it useful for applications where data is limited. Additionally, the shared weights between the two subnetworks allow the model to generalize well to new inputs.

Siamese Networks have been shown to be effective in a wide range of applications, including image recognition, classification, and speech recognition. They have also been applied to natural language processing tasks such as sentence similarity and paraphrase detection.

[Tr]

Siamese Network, iki giriş örneği arasındaki benzerlik ve farklılıkları bulmak için kullanılan bir sinir ağı mimarisidir. Ağ, iki aynı alt ağdan oluşur ve her biri aynı ağırlık kümesini paylaşır.

Siamese Network'ün temel fikri, iki giriş örneği arasındaki benzerliği öğrenmektir. Bu nedenle, ağ, iki giriş örneği benzer olduğunda yüksek bir çıktı değeri verir ve farklı olduğunda ise düşük bir çıktı değeri verir. Bu, örneğin görüntü veya metin benzerliği eşleştirme, yüz tanıma veya imza doğrulama gibi birçok uygulama için kullanışlıdır.

Siamese Network'ü kullanmanın en büyük avantajı, sınırlı veriyle bile eğitilebilmesidir. Ayrıca, alt ağlar arasındaki ağırlık paylaşımı, modelin yeni girişlere iyi genelleme yapabilmesine olanak tanır.

Siamese Network, görüntü tanıma, sınıflandırma ve konuşma tanıma gibi birçok alanda etkili olduğu kanıtlanmıştır. Ayrıca, cümle benzerliği ve paraphrase tespiti gibi doğal dil işleme görevleri için de uygulanabilir.

```
import sys
sys.path.append('/content/Person-Re-Id-Dataset')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import torch

"""
Timm: PyTorch Image Models (timm) is a library for state-of-the-art-
image classification, containing a collection of image models,
optimizers, schedulers, augmentations and much more.
"""
import timm

import torch.nn.functional as F
from torch import nn
from torch.utils.data import Dataset, DataLoader

from skimage import io
from sklearn.model_selection import train_test_split

"""
tqdm is a library that is used for creating Python Progress Bars. It
gets its name from the Arabic name taqaddum, which means 'progress. '
"""
from tqdm import tqdm
```

Configurations

```
DATA_DIR = "/content/Person-Re-Id-Dataset/train/"
CSV_FILE = "/content/Person-Re-Id-Dataset/train.csv"

BATCH_SIZE = 32
LR = 0.001
EPOCHS = 15

DEVICE = 'cuda'

df = pd.read_csv(CSV_FILE)
df.head()
```

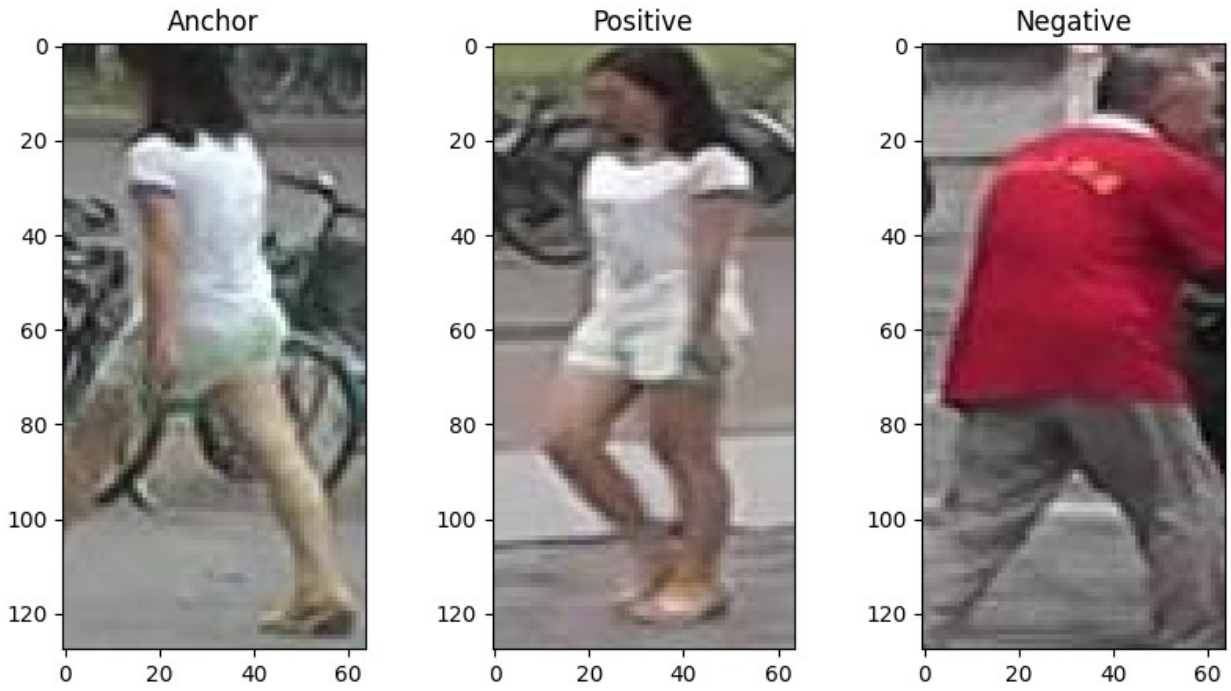

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 4000,\n  \"fields\":
[\n    {\n      \"column\": \"Anchor\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 3285,\n
```

```
\"samples\": [\n          \"0206_c5s1_052801_04.jpg\",\n
\"0482_c3s1_139183_03.jpg\",\n          \"1080_c3s2_144344_01.jpg\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Negative\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 3417,\n        \"samples\": [\n
\"0340_c6s1_080451_02.jpg\",\n          \"0741_c6s2_061343_01.jpg\",\n
\"0868_c3s2_108203_11.jpg\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Positive\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 3219,\n
\"samples\": [\n          \"1152_c2s3_012407_01.jpg\",\n
\"0105_c2s1_017601_01.jpg\",\n          \"1286_c6s3_051367_01.jpg\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

```
row = df.iloc[11]
```

```
A_img = io.imread(DATA_DIR + row.Anchor)
P_img = io.imread(DATA_DIR + row.Positive)
N_img = io.imread(DATA_DIR + row.Negative)

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize = (10,5))

ax1.set_title("Anchor")
ax1.imshow(A_img)

ax2.set_title("Positive")
ax2.imshow(P_img)

ax3.set_title("Negative")
ax3.imshow(N_img)
```

```
<matplotlib.image.AxesImage at 0x7a934b1048c0>
```




```
train_df, valid_df = train_test_split(df, test_size = 0.20,
random_state = 42)
```

Create APN Dataset

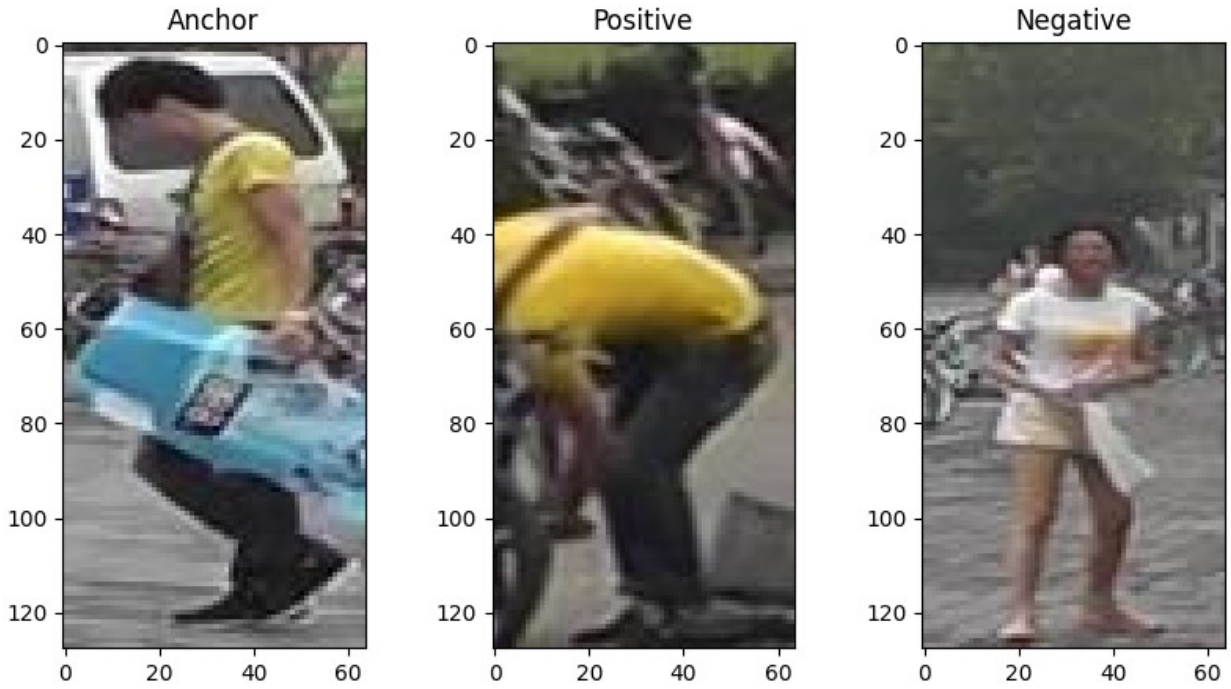
```
class APN_Dataset(Dataset):

    def __init__(self, df):
        self.df = df

    def __len__(self):
        return len(self.df)

    def __getitem__(self,idx):
        row = self.df.iloc[idx]

        A_img = io.imread(DATA_DIR + row.Anchor.lstrip('/'))
        P_img = io.imread(DATA_DIR + row.Positive.lstrip('/'))
        N_img = io.imread(DATA_DIR + row.Negative.lstrip('/'))

        A_img = torch.from_numpy(A_img).permute(2, 0 ,1).float() /
255.0
        P_img = torch.from_numpy(P_img).permute(2, 0 ,1).float() /
255.0
        N_img = torch.from_numpy(N_img).permute(2, 0 ,1).float() /
255.0
```

```
        return A_img, P_img, N_img

trainset = APN_Dataset(train_df)
validset = APN_Dataset(valid_df)

print(f"Size of trainset : {len(trainset)}")
print(f"Size of validset : {len(validset)}")
```

```
Size of trainset : 3200
,Size of validset : 800
```

```
idx = 40
A,P,N = trainset[idx]

f, (ax1, ax2, ax3) = plt.subplots(1,3,figsize= (10,5))

ax1.set_title('Anchor')
ax1.imshow(A.numpy().transpose((1,2,0)), cmap = 'gray')

ax2.set_title('Positive')
ax2.imshow(P.numpy().transpose((1,2,0)), cmap = 'gray')

ax3.set_title('Negative')
ax3.imshow(N.numpy().transpose((1,2,0)), cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7a9348f61ca0>
```



Load Dataset into Batches

```
trainloader = DataLoader(trainset, batch_size = BATCH_SIZE,shuffle =
True)
validloader = DataLoader(validset, batch_size = BATCH_SIZE)

print(f"No. of batches in trainloader : {len(trainloader)}")
print(f"No. of batches in validloader : {len(validloader)}")
```

```
No. of batches in trainloader : 100
,No. of batches in validloader : 25
```

```
for A, P, N in trainloader:
    break;

print(f"One image batch shape : {A.shape}")
```

```
One image batch shape : torch.Size([32, 3, 128, 64])
```

Create Model

```
class APN_Model(nn.Module):

    def __init__(self, emb_size = 512):
        super(APN_Model, self).__init__()

        self.efficientnet = timm.create_model('efficientnet_b0',
pretrained=True)
        self.efficientnet.classifier = nn.Linear(in_features =
self.efficientnet.classifier.in_features,
                                                 out_features =
emb_size)


    def forward(self, images):
        embeddings = self.efficientnet(images)
        return embeddings

model = APN_Model()
model.to(DEVICE)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
,The secret `HF_TOKEN` does not exist in your Colab secrets.
,To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
,You will be able to reuse this secret in all of your notebooks.
,Please note that authentication is recommended but still optional to
```

```
access public models or datasets.
,  warnings.warn(

{"model_id":"066e0b8e6680460298abcd0d094a4099","version_major":2,"vers
ion_minor":0}

APN_Model(
  (efficientnet): EfficientNet(
    (conv_stem): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
    (bn1): BatchNormAct2d(
      32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
      (drop): Identity()
      (act): SiLU(inplace=True)
    )
    (blocks): Sequential(
      (0): Sequential(
        (0): DepthwiseSeparableConv(
          (conv_dw): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=32, bias=False)
          (bn1): BatchNormAct2d(
            32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(32, 8, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(8, 32, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pw): Conv2d(32, 16, kernel_size=(1, 1), stride=(1, 1),
bias=False)
          (bn2): BatchNormAct2d(
            16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
      )
      (1): Sequential(
        (0): InvertedResidual(
          (conv_pw): Conv2d(16, 96, kernel_size=(1, 1), stride=(1, 1),
```

```
bias=False)
          (bn1): BatchNormAct2d(
            96, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(96, 96, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), groups=96, bias=False)
          (bn2): BatchNormAct2d(
            96, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(96, 4, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(4, 96, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(96, 24, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            24, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
        (1): InvertedResidual(
          (conv_pw): Conv2d(24, 144, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(144, 144, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), groups=144, bias=False)
          (bn2): BatchNormAct2d(
            144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
```

```
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(144, 6, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(6, 144, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(144, 24, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            24, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
      )
      (2): Sequential(
        (0): InvertedResidual(
          (conv_pw): Conv2d(24, 144, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(144, 144, kernel_size=(5, 5), stride=(2,
2), padding=(2, 2), groups=144, bias=False)
          (bn2): BatchNormAct2d(
            144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(144, 6, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(6, 144, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
```

```
          (conv_pwl): Conv2d(144, 40, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            40, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
        (1): InvertedResidual(
          (conv_pw): Conv2d(40, 240, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(240, 240, kernel_size=(5, 5), stride=(1,
1), padding=(2, 2), groups=240, bias=False)
          (bn2): BatchNormAct2d(
            240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(240, 10, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(10, 240, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(240, 40, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            40, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
      )
      (3): Sequential(
        (0): InvertedResidual(
```

```
          (conv_pw): Conv2d(40, 240, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(240, 240, kernel_size=(3, 3), stride=(2,
2), padding=(1, 1), groups=240, bias=False)
          (bn2): BatchNormAct2d(
            240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(240, 10, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(10, 240, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(240, 80, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            80, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
        (1): InvertedResidual(
          (conv_pw): Conv2d(80, 480, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(480, 480, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), groups=480, bias=False)
          (bn2): BatchNormAct2d(
            480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
```

```
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(480, 20, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(20, 480, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(480, 80, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            80, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
        (2): InvertedResidual(
          (conv_pw): Conv2d(80, 480, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(480, 480, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), groups=480, bias=False)
          (bn2): BatchNormAct2d(
            480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(480, 20, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(20, 480, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(480, 80, kernel_size=(1, 1), stride=(1,
```

```
1), bias=False)
          (bn3): BatchNormAct2d(
            80, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
      )
      (4): Sequential(
        (0): InvertedResidual(
          (conv_pw): Conv2d(80, 480, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(480, 480, kernel_size=(5, 5), stride=(1,
1), padding=(2, 2), groups=480, bias=False)
          (bn2): BatchNormAct2d(
            480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(480, 20, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(20, 480, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(480, 112, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            112, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
        (1): InvertedResidual(
          (conv_pw): Conv2d(112, 672, kernel_size=(1, 1), stride=(1,
```

```
1), bias=False)
          (bn1): BatchNormAct2d(
            672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(672, 672, kernel_size=(5, 5), stride=(1,
1), padding=(2, 2), groups=672, bias=False)
          (bn2): BatchNormAct2d(
            672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(672, 28, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(28, 672, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(672, 112, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            112, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
        (2): InvertedResidual(
          (conv_pw): Conv2d(112, 672, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(672, 672, kernel_size=(5, 5), stride=(1,
1), padding=(2, 2), groups=672, bias=False)
          (bn2): BatchNormAct2d(
            672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
```

```
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(672, 28, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(28, 672, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(672, 112, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            112, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
      )
      (5): Sequential(
        (0): InvertedResidual(
          (conv_pw): Conv2d(112, 672, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(672, 672, kernel_size=(5, 5), stride=(2,
2), padding=(2, 2), groups=672, bias=False)
          (bn2): BatchNormAct2d(
            672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(672, 28, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(28, 672, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
```

```
          (conv_pwl): Conv2d(672, 192, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
        (1): InvertedResidual(
          (conv_pw): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1,
1), padding=(2, 2), groups=1152, bias=False)
          (bn2): BatchNormAct2d(
            1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(1152, 48, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(48, 1152, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
        (2): InvertedResidual(
          (conv_pw): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1,
1), bias=False)
```

```
          (bn1): BatchNormAct2d(
            1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1,
1), padding=(2, 2), groups=1152, bias=False)
          (bn2): BatchNormAct2d(
            1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(1152, 48, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(48, 1152, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
        (3): InvertedResidual(
          (conv_pw): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1,
1), padding=(2, 2), groups=1152, bias=False)
          (bn2): BatchNormAct2d(
            1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
```

```
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(1152, 48, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(48, 1152, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn3): BatchNormAct2d(
            192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
      )
      (6): Sequential(
        (0): InvertedResidual(
          (conv_pw): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1,
1), bias=False)
          (bn1): BatchNormAct2d(
            1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (conv_dw): Conv2d(1152, 1152, kernel_size=(3, 3), stride=(1,
1), padding=(1, 1), groups=1152, bias=False)
          (bn2): BatchNormAct2d(
            1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): SiLU(inplace=True)
          )
          (aa): Identity()
          (se): SqueezeExcite(
            (conv_reduce): Conv2d(1152, 48, kernel_size=(1, 1),
stride=(1, 1))
            (act1): SiLU(inplace=True)
            (conv_expand): Conv2d(48, 1152, kernel_size=(1, 1),
stride=(1, 1))
            (gate): Sigmoid()
          )
          (conv_pwl): Conv2d(1152, 320, kernel_size=(1, 1), stride=(1,
1), bias=False)
```

```
          (bn3): BatchNormAct2d(
            320, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
            (drop): Identity()
            (act): Identity()
          )
          (drop_path): Identity()
        )
      )
    )
    (conv_head): Conv2d(320, 1280, kernel_size=(1, 1), stride=(1, 1),
bias=False)
    (bn2): BatchNormAct2d(
      1280, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
      (drop): Identity()
      (act): SiLU(inplace=True)
    )
    (global_pool): SelectAdaptivePool2d(pool_type=avg,
flatten=Flatten(start_dim=1, end_dim=-1))
    (classifier): Linear(in_features=1280, out_features=512,
bias=True)
  )
)
```

Create Train and Eval Function

```python
def train_fn(model, dataloader, optimizer, criterion):
    model.train() # ON Dropout
    total_loss = 0.0

    for A,P,N in tqdm(dataloader):
        A,P,N = A.to(DEVICE), P.to(DEVICE), N.to(DEVICE)

        A_embs = model(A)
        P_embs = model(P)
        N_embs = model(N)

        loss = criterion(A_embs, P_embs, N_embs)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    return total_loss / len(dataloader)
```

```
def eval_fn(model, dataloader, criterion):
    model.eval()  # OFF Dropout
    total_loss = 0.0

    with torch.no_grad():
        for A,P,N in tqdm(dataloader):
            A,P,N = A.to(DEVICE), P.to(DEVICE), N.to(DEVICE)

            A_embs = model(A)
            P_embs = model(P)
            N_embs = model(N)

            loss = criterion(A_embs, P_embs, N_embs)

            total_loss += loss.item()

        return total_loss / len(dataloader)

criterion = nn.TripletMarginLoss()
optimizer = torch.optim.Adam(model.parameters(), lr = LR)
```

Create Training Loop

```
best_valid_loss = np.inf

for i in range(EPOCHS):
    train_loss = train_fn(model, trainloader, optimizer, criterion)
    valid_loss = eval_fn(model, validloader, criterion)

    if valid_loss < best_valid_loss:
        torch.save(model.state_dict(), "best_model.pt")
        best_valid_loss = valid_loss
        print("SAVED_WEIGHT_SUCCESS")

    print(f"EPOCHS: {i+1} train_loss: {train_loss} valid_loss:
{valid_loss}")
```

```
100%|██████████| 100/100 [00:22<00:00,  4.53it/s]
,100%|██████████| 25/25 [00:02<00:00,  9.47it/s]

SAVED_WEIGHT_SUCCESS
,EPOCHS: 1 train_loss: 0.5968920367956162 valid_loss:
0.5999924647808075

100%|██████████| 100/100 [00:20<00:00,  4.96it/s]
,100%|██████████| 25/25 [00:02<00:00,  9.57it/s]
```

```
SAVED_WEIGHT_SUCCESS
,EPOCHS: 2 train_loss: 0.28535742178559304 valid_loss:
0.3713403022289276

100%|██████████| 100/100 [00:20<00:00,  4.98it/s]
,100%|██████████| 25/25 [00:02<00:00,  9.52it/s]

SAVED_WEIGHT_SUCCESS
,EPOCHS: 3 train_loss: 0.20957083165645599 valid_loss:
0.24891111195087434

100%|██████████| 100/100 [00:20<00:00,  4.89it/s]
,100%|██████████| 25/25 [00:02<00:00,  9.66it/s]

SAVED_WEIGHT_SUCCESS
,EPOCHS: 4 train_loss: 0.12201815888285637 valid_loss:
0.19391652703285217

100%|██████████| 100/100 [00:20<00:00,  4.98it/s]
,100%|██████████| 25/25 [00:02<00:00,  8.99it/s]

EPOCHS: 5 train_loss: 0.04998568296432495 valid_loss:
0.2600224596261978

100%|██████████| 100/100 [00:20<00:00,  4.96it/s]
,100%|██████████| 25/25 [00:02<00:00,  9.45it/s]

EPOCHS: 6 train_loss: 0.05846977993845939 valid_loss:
0.22203351736068724

100%|██████████| 100/100 [00:19<00:00,  5.06it/s]
,100%|██████████| 25/25 [00:03<00:00,  8.16it/s]

EPOCHS: 7 train_loss: 0.048933724462985995 valid_loss:
0.24937967240810394

100%|██████████| 100/100 [00:19<00:00,  5.09it/s]
,100%|██████████| 25/25 [00:03<00:00,  7.69it/s]

EPOCHS: 8 train_loss: 0.03886239975690842 valid_loss:
0.235456138253212

100%|██████████| 100/100 [00:19<00:00,  5.06it/s]
,100%|██████████| 25/25 [00:02<00:00,  8.57it/s]

EPOCHS: 9 train_loss: 0.08804476633667946 valid_loss:
0.22347100734710693

100%|██████████| 100/100 [00:19<00:00,  5.00it/s]
,100%|██████████| 25/25 [00:02<00:00,  9.31it/s]

EPOCHS: 10 train_loss: 0.0637633104622364 valid_loss:
0.24486860811710356
```

```
100%|██████████| 100/100 [00:20<00:00,  4.98it/s]
,100%|██████████| 25/25 [00:02<00:00,  9.31it/s]

SAVED_WEIGHT_SUCCESS
,EPOCHS: 11 train_loss: 0.060202498137950894 valid_loss:
0.17849076211452483

100%|██████████| 100/100 [00:20<00:00,  4.99it/s]
,100%|██████████| 25/25 [00:02<00:00,  9.39it/s]

EPOCHS: 12 train_loss: 0.03959650836884975 valid_loss:
0.18477850109338761

100%|██████████| 100/100 [00:20<00:00,  4.98it/s]
,100%|██████████| 25/25 [00:02<00:00,  9.44it/s]

SAVED_WEIGHT_SUCCESS
,EPOCHS: 13 train_loss: 0.043007399737834934 valid_loss:
0.15608882516622544

100%|██████████| 100/100 [00:20<00:00,  4.97it/s]
,100%|██████████| 25/25 [00:02<00:00,  9.48it/s]

EPOCHS: 14 train_loss: 0.04459821283817291 valid_loss:
0.1647336047887802

100%|██████████| 100/100 [00:20<00:00,  5.00it/s]
,100%|██████████| 25/25 [00:02<00:00,  9.46it/s]

EPOCHS: 15 train_loss: 0.03372037291526794 valid_loss:
0.16118236005306244
```

Get Anchor Embeddings

```python
def get_encoding_csv(model, anc_img_names):
    anc_img_names_arr = np.array(anc_img_names)
    encodings = []

    model.eval()
    with torch.no_grad():
        for i in tqdm(anc_img_names_arr):
            A = io.imread(DATA_DIR + i)
            A = torch.from_numpy(A).permute(2, 0, 1) / 255.0
            A = A. to(DEVICE)
            A_enc = model(A.unsqueeze(0)) # c,h,w --> (1,c,h,w)
            encodings.append(A_enc.squeeze().cpu().detach().numpy())

        encodings = np.array(encodings)
```

```
        encodings = pd.DataFrame(encodings)
        df_enc = pd.concat([anc_img_names, encodings], axis=1)

    return df_enc

model.load_state_dict(torch.load("best_model.pt"))
df_enc = get_encoding_csv(model, df["Anchor"])
```

```
100%|██████████| 4000/4000 [00:39<00:00, 102.42it/s]
```

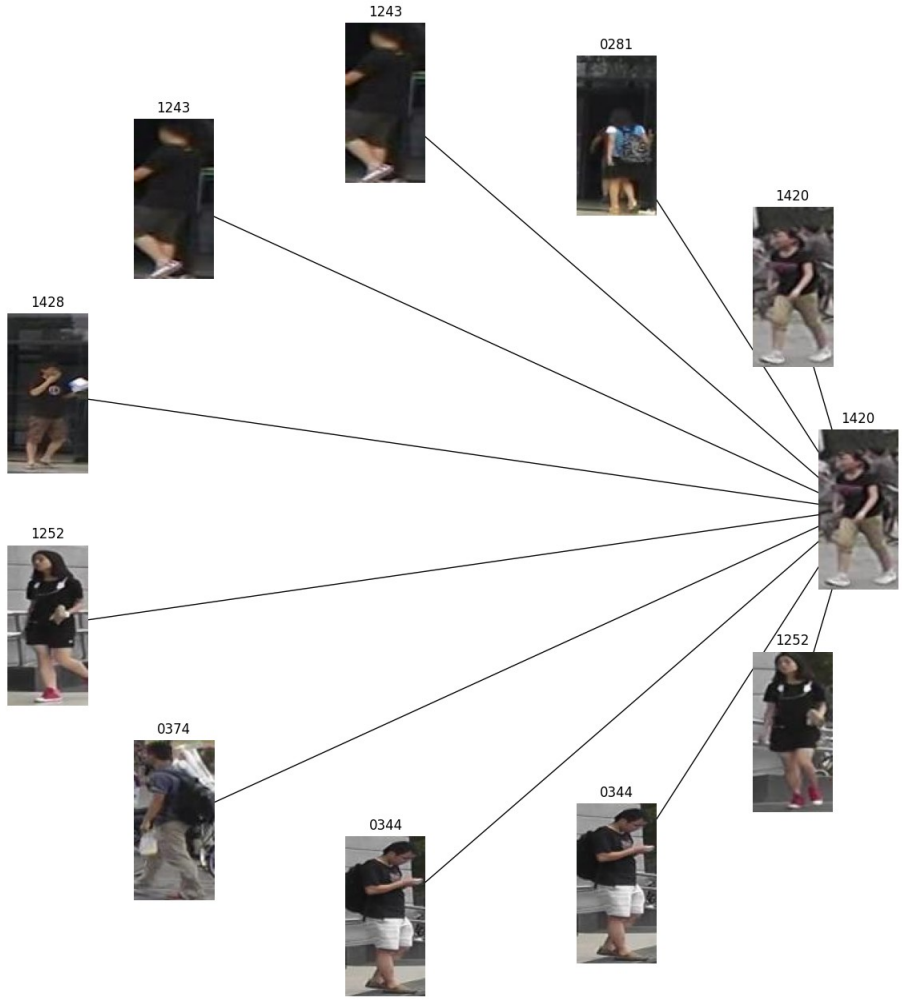
```
df_enc.to_csv("database.csv", index=False)
df_enc.head()
```

{"type":"dataframe","variable_name":"df_enc"}

Inference

```
def euclidean_dist(img_enc, anc_enc_arr):
    dist = np.sqrt(np.dot(img_enc-anc_enc_arr, (img_enc -
anc_enc_arr).T))
    return dist

idx = 0
img_name = df_enc["Anchor"].iloc[idx]
img_path = DATA_DIR + img_name

img = io.imread(img_path)
img = torch.from_numpy(img).permute(2, 0, 1) / 255.0

model.eval()
with torch.no_grad():
    img = img.to(DEVICE)
    img_enc = model(img.unsqueeze(0))
    img_enc = img_enc.detach().cpu().numpy()

anc_enc_arr = df_enc.iloc[:, 1:].to_numpy()
anc_img_names = df_enc["Anchor"]

distance = []

for i in range(anc_enc_arr.shape[0]):
    dist = euclidean_dist(img_enc, anc_enc_arr[i : i+1, :])
    distance = np.append(distance, dist)

closest_idx = np.argsort(distance)

from utils import plot_closest_imgs

plot_closest_imgs(anc_img_names, DATA_DIR, img, img_path, closest_idx,
distance, no_of_closest = 10);
```

```
/usr/local/lib/python3.12/dist-packages/networkx/drawing/
layout.py:982: RuntimeWarning: divide by zero encountered in divide
,  costargs = (np, 1 / (dist_mtx + np.eye(dist_mtx.shape[0]) * 1e-3),
meanwt, dim)
```



Resources

- How to train your siamese neural network