similarity

March 25, 2025

0.1 Downloading the national gallery of art open data from kaggle

[ ]:
```
import kagglehub

path = kagglehub.dataset_download("peacehegemony/
 ↪the-national-gallery-of-art-open-data-program")

print("Path to dataset files:", path)
```

```
Path to dataset files: /home/ameya/.cache/kagglehub/datasets/peacehegemony/the-
national-gallery-of-art-open-data-program/versions/1
```

[2]:
```
path = path + "/opendata-main/data/"
```

0.2 Joining table based on appropriate columns to get data

[ ]:
```
import pandas as pd

objects_df = pd.read_csv(path + "objects.csv")
mediarel_df = pd.read_csv(path + "media_relationships.csv")
media_items_df = pd.read_csv(path + "media_items.csv")

obj_mediarel_df = objects_df.merge(
    mediarel_df,
    left_on="objectid",
    right_on="relatedid",
    how="inner"
)

merged_df = obj_mediarel_df.merge(
    media_items_df,
    on="mediaid",
    how="inner"
)

print("Merged DataFrame shape:", merged_df.shape)
print(merged_df.head())
```

```
/tmp/ipykernel_108246/2913492502.py:3: DtypeWarning: Columns (29) have mixed
types. Specify dtype option on import or set low_memory=False.
  objects_df = pd.read_csv(path + "objects.csv")
```

```
Merged DataFrame shape: (2886, 47)
   objectid  accessioned accessionnum  locationid            title_x
0        61            1    1937.1.54         NaN       The Lacemaker  \
1        62            1    1937.1.55         NaN    The Smiling Girl
2        62            1    1937.1.55         NaN    The Smiling Girl
3      2718            1  1942.9.1839         NaN  Oeuvres poissardes
4        62            1    1937.1.55         NaN    The Smiling Girl

      displaydate  beginyear  endyear visualbrowsertimespan
0         c. 1925     1925.0   1925.0          1901 to 1925  \
1         c. 1925     1925.0   1925.0          1901 to 1925
2         c. 1925     1925.0   1925.0          1901 to 1925
3  published 1796     1796.0   1796.0          1776 to 1800
4         c. 1925     1925.0   1925.0          1901 to 1925

                                  medium  … language
0                          oil on canvas  …       en  \
1                          oil on canvas  …       en
2                          oil on canvas  …       en
3  1 vol: ill: 4 color stipple engravings  …       en
4                          oil on canvas  …       en

                                        thumbnailurl
0  https://www.nga.gov/content/dam/ngaweb/audio-v…  \
1  https://www.nga.gov/content/dam/ngaweb/audio-v…
2  https://www.nga.gov/content/dam/ngaweb/audio-v…
3  https://www.nga.gov/content/dam/ngaweb/audio-v…
4  https://www.nga.gov/content/dam/ngaweb/audio-v…

                                             playurl
0  https://w.soundcloud.com/player/?url=https%3A%…  \
1  https://w.soundcloud.com/player/?url=https%3A%…
2  https://w.soundcloud.com/player/?url=https%3A%…
3  https://w.soundcloud.com/player/?url=https%3A%…
4  https://players.brightcove.net/1191289016001/d…

                                         downloadurl
0  https://api.soundcloud.com/tracks/78345258/dow…  \
1  https://api.soundcloud.com/tracks/78345258/dow…
2  https://api.soundcloud.com/tracks/475697055/do…
3  https://api.soundcloud.com/tracks/475697055/do…
4                                                NaN

                          keywords
```

```
0     vermeer, han van meegeren, forgery  \
1     vermeer, han van meegeren, forgery
2  jecmen, rosenwald, prints, drawings,
3  jecmen, rosenwald, prints, drawings,
4  jecmen, rosenwald, prints, drawings,

                                                tags
0  ngaweb:audio-video/audio,ngaweb:audio-video/au…  \
1  ngaweb:audio-video/audio,ngaweb:audio-video/au…
2  ngaweb:constituents/6/2/Constituent_62,ngaweb:…
3  ngaweb:constituents/6/2/Constituent_62,ngaweb:…
4  ngaweb:audio-video/podcast-video,ngaweb:audio-…

                                            imageurl        presentationdate
0  https://www.nga.gov/content/dam/ngaweb/audio-v…  2009-01-11 00:00:00-05  \
1  https://www.nga.gov/content/dam/ngaweb/audio-v…  2009-01-11 00:00:00-05
2  https://www.nga.gov/content/dam/ngaweb/audio-v…  2018-03-16 00:00:00-04
3  https://www.nga.gov/content/dam/ngaweb/audio-v…  2018-03-16 00:00:00-04
4  https://www.nga.gov/content/dam/ngaweb/audio-v…  2018-03-16 00:00:00-04

             releasedate            lastmodified
0  2009-01-20 00:00:00-05  2014-10-10 12:22:21-04
1  2009-01-20 00:00:00-05  2014-10-10 12:22:21-04
2  2018-07-24 00:00:00-04  2019-04-09 15:14:14-04
3  2018-07-24 00:00:00-04  2019-04-09 15:14:14-04
4  2018-07-24 00:00:00-04  2019-04-09 15:14:30-04

[5 rows x 47 columns]
```

0.3 Creating a dictionary with objectid as the key and related imageurls as the values

[4]:
```python
images_dict = {}

for idx, row in merged_df.iterrows():
    obj_id = row["objectid"]
    img_path = row["imageurl"]

    if obj_id not in images_dict:
        images_dict[obj_id] = []

    images_dict[obj_id].append(img_path)

# test_ids = list(images_dict.keys())
# for test_id in test_ids:
#     print(f"Object ID: {test_id}")
#     print("Image paths:", images_dict[test_id])
```

```
print(len(images_dict))
```

807

[5]:
```
import os
import random
from PIL import Image
import torch
from torch.utils.data import Dataset, DataLoader
import torchvision.transforms as transforms
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import requests
from io import BytesIO
```

0.4 Loading image from the given url

[ ]:
```
issues = 0
total = 0
def load_image_from_url(url):
    global issues, total
    try:
        total += 1
        response = requests.get(url, timeout=10)
        response.raise_for_status()
        img = Image.open(BytesIO(response.content)).convert("RGB")
        return img
    except Exception as e:
        issues += 1
        return Image.new("RGB", (224, 224), (0, 0, 0))
```

0.5 Creating a dataset for siamese networks (consisting of negative and positive samples)

[ ]:
```
class SiameseDataset(Dataset):
    def __init__(self, images_dict, transform=None, target_size=(224, 224)):
        self.images_dict = images_dict
        self.object_ids = list(images_dict.keys())
        self.transform = transform
        self.target_size = target_size
        self.pairs = []
        self.labels = []
        self._create_pairs()

    def _create_pairs(self):
```

```
        for obj_id in self.object_ids:
            urls = self.images_dict[obj_id]
            if len(urls) < 2:
                continue
            for i in range(len(urls)):
                for j in range(i+1, len(urls)):
                    self.pairs.append((urls[i], urls[j]))
                    self.labels.append(1)

        num_positive = len(self.labels)
        neg_pairs = 0
        while neg_pairs < num_positive:
            id1, id2 = random.sample(self.object_ids, 2)
            if len(self.images_dict[id1]) == 0 or len(self.images_dict[id2]) ==␣
↪0:
                continue
            url1 = random.choice(self.images_dict[id1])
            url2 = random.choice(self.images_dict[id2])
            self.pairs.append((url1, url2))
            self.labels.append(0)
            neg_pairs += 1

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

    def __getitem__(self, idx):
        url1, url2 = self.pairs[idx]
        img1 = load_image_from_url(url1)
        img2 = load_image_from_url(url2)
        if self.transform:
            img1 = self.transform(img1)
            img2 = self.transform(img2)
        label = torch.tensor(self.labels[idx], dtype=torch.float32)
        return img1, img2, label

# Define transformations for the images
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
```

[8]:
```
dataset = SiameseDataset(images_dict, transform=transform)
# split into test and train
train_size = int(0.8 * len(dataset))
test_size = len(dataset) - train_size
```

```
train_dataset, test_dataset = torch.utils.data.random_split(dataset,
    [train_size, test_size])

train_loader = DataLoader(train_dataset, batch_size=32,
    shuffle=True,num_workers=4)
test_loader = DataLoader(test_dataset, batch_size=32,
    shuffle=False,num_workers=4)
```

[ ]:
```
class SiameseNetwork(nn.Module):
    def __init__(self):
        super(SiameseNetwork, self).__init__()
        self.cnn = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
            nn.Conv2d(64, 128, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2)
        )
        self.fc = nn.Sequential(
            nn.Linear(128 * 28 * 28, 512),
            nn.ReLU(inplace=True),
            nn.Linear(512, 128)
        )

    def forward_once(self, x):
        output = self.cnn(x)
        output = output.view(output.size(0), -1)
        output = self.fc(output)
        return output

    def forward(self, input1, input2):
        output1 = self.forward_once(input1)
        output2 = self.forward_once(input2)
        return output1, output2
```

0.6 Contrastive loss function for training

[ ]:
```
class ContrastiveLoss(nn.Module):
    def __init__(self, margin=1.0):
        super(ContrastiveLoss, self).__init__()
        self.margin = margin
```

```
    def forward(self, output1, output2, label):
        euclidean_distance = F.pairwise_distance(output1, output2)
        loss_contrastive = torch.mean(
            label * torch.pow(euclidean_distance, 2) +
            (1 - label) * torch.pow(torch.clamp(self.margin -␣
↪euclidean_distance, min=0.0), 2)
        )
        return loss_contrastive
```

0.7 Model declaration

[11]:
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = SiameseNetwork().to(device)
criterion = ContrastiveLoss(margin=1.0)
optimizer = optim.Adam(model.parameters(), lr=1e-3)
```

[12]:
```
from tqdm import tqdm
```

0.8 Training the model

[13]:
```
num_epochs = 11

for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    train_bar = tqdm(train_loader, desc=f"Epoch {epoch+1}/{num_epochs} -␣
↪Training", leave=False)
    for img1, img2, label in train_bar:
        img1, img2, label = img1.to(device), img2.to(device), label.to(device)

        optimizer.zero_grad()
        output1, output2 = model(img1, img2)
        loss = criterion(output1, output2, label)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        train_bar.set_postfix(loss=f"{loss.item():.4f}")

    avg_loss = running_loss / len(train_loader)
    print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {avg_loss:.4f}")

    model.eval()
    test_loss = 0.0
    test_bar = tqdm(test_loader, desc=f"Epoch {epoch+1}/{num_epochs} -␣
↪Testing", leave=False)
```

```
    with torch.no_grad():
        for img1, img2, label in test_bar:
            img1, img2, label = img1.to(device), img2.to(device), label.
↪to(device)
            output1, output2 = model(img1, img2)
            loss = criterion(output1, output2, label)
            test_loss += loss.item()
            test_bar.set_postfix(loss=f"{loss.item():.4f}")
    avg_test_loss = test_loss / len(test_loader)
    print(f"Test Loss: {avg_test_loss:.4f}")
    if(epoch % 5 == 0):
        torch.save(model.state_dict(), f"siamese_model_epoch{epoch}.pt")

print("Training complete.")
```

```
Epoch 1/11 - Training:   0%|          | 0/742 [00:00<?, ?it/s]

Epoch [1/11], Loss: 0.3343

Test Loss: 0.0724

Epoch [2/11], Loss: 0.0443

Test Loss: 0.0493

Epoch [3/11], Loss: 0.0306

Test Loss: 0.0361

Epoch [4/11], Loss: 0.0248

Test Loss: 0.0300

Epoch [5/11], Loss: 0.0222

Test Loss: 0.0277

Epoch [6/11], Loss: 0.0210
```

```
Test Loss: 0.0253
```

```
Epoch [7/11], Loss: 0.0188
```

```
Test Loss: 0.0286
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[13], line 8
      6 # Wrap the training dataloader with tqdm directly
      7 train_bar = tqdm(train_loader, desc=f"Epoch {epoch+1}/{num_epochs} -␣
 ↪Training", leave=False)
----> 8 for img1, img2, label in train_bar:
      9     img1, img2, label = img1.to(device), img2.to(device), label.
 ↪to(device)
     11     optimizer.zero_grad()

File ~/.local/lib/python3.10/site-packages/tqdm/std.py:1178, in tqdm.
 ↪__iter__(self)
   1175 time = self._time
   1177 try:
-> 1178     for obj in iterable:
   1179         yield obj
   1180         # Update and possibly print the progressbar.
   1181         # Note: does not call self.update(1) for speed optimisation.

File ~/.local/lib/python3.10/site-packages/torch/utils/data/dataloader.py:708,␣
 ↪in _BaseDataLoaderIter.__next__(self)
    705 if self._sampler_iter is None:
    706     # TODO(https://github.com/pytorch/pytorch/issues/76750)
    707     self._reset()  # type: ignore[call-arg]
--> 708 data = self._next_data()
    709 self._num_yielded += 1
    710 if (
    711     self._dataset_kind == _DatasetKind.Iterable
    712     and self._IterableDataset_len_called is not None
    713     and self._num_yielded > self._IterableDataset_len_called
    714 ):

File ~/.local/lib/python3.10/site-packages/torch/utils/data/dataloader.py:1458,
 ↪in _MultiProcessingDataLoaderIter._next_data(self)
   1455     return self._process_data(data)
```

```
   1457 assert not self._shutdown and self._tasks_outstanding > 0
-> 1458 idx, data = self._get_data()
   1459 self._tasks_outstanding -= 1
   1460 if self._dataset_kind == _DatasetKind.Iterable:
   1461     # Check for _IterableDatasetStopIteration

File ~/.local/lib/python3.10/site-packages/torch/utils/data/dataloader.py:1420,
 ↪in _MultiProcessingDataLoaderIter._get_data(self)
   1416     # In this case, `self._data_queue` is a `queue.Queue`,. But we don'
   1417     # need to call `.task_done()` because we don't use `.join()`.
   1418 else:
   1419     while True:
-> 1420         success, data = self._try_get_data()
   1421         if success:
   1422             return data

File ~/.local/lib/python3.10/site-packages/torch/utils/data/dataloader.py:1251,
 ↪in _MultiProcessingDataLoaderIter._try_get_data(self, timeout)
   1238 def _try_get_data(self, timeout=_utils.MP_STATUS_CHECK_INTERVAL):
   1239     # Tries to fetch data from `self._data_queue` once for a given
 ↪timeout.
   1240     # This can also be used as inner loop of fetching without timeout,
 ↪with
   (…)
   1248     # Returns a 2-tuple:
   1249     #   (bool: whether successfully get data, any: data if successful
 ↪else None)
   1250     try:
-> 1251         data = self._data_queue.get(timeout=timeout)
   1252         return (True, data)
   1253     except Exception as e:
   1254         # At timeout and error, we manually check whether any worker ha
   1255         # failed. Note that this is the only mechanism for Windows to
 ↪detect
   1256         # worker failures.

File /usr/lib/python3.10/multiprocessing/queues.py:113, in Queue.get(self,
 ↪block, timeout)
    111 if block:
    112     timeout = deadline - time.monotonic()
--> 113     if not self._poll(timeout):
    114         raise Empty
    115 elif not self._poll():

File /usr/lib/python3.10/multiprocessing/connection.py:257, in _ConnectionBase.
 ↪poll(self, timeout)
    255 self._check_closed()
    256 self._check_readable()
```

```
--> 257 return self._poll(timeout)

File /usr/lib/python3.10/multiprocessing/connection.py:424, in Connection.
↪_poll(self, timeout)
    423 def _poll(self, timeout):
--> 424     r = wait([self], timeout)
    425     return bool(r)

File /usr/lib/python3.10/multiprocessing/connection.py:931, in wait(object_list,␣
↪timeout)
    928     deadline = time.monotonic() + timeout
    930 while True:
--> 931     ready = selector.select(timeout)
    932     if ready:
    933         return [key.fileobj for (key, events) in ready]

File /usr/lib/python3.10/selectors.py:416, in _PollLikeSelector.select(self,␣
↪timeout)
    414 ready = []
    415 try:
--> 416     fd_event_list = self._selector.poll(timeout)
    417 except InterruptedError:
    418     return ready

KeyboardInterrupt:
```

0.9 Dataset creation (consisting of anchor, positive and negative images)

[14]:
```
class TripletDataset(Dataset):
    def __init__(self, images_dict, transform=None, target_size=(224, 224)):
        self.images_dict = images_dict
        self.object_ids = list(images_dict.keys())
        self.transform = transform
        self.target_size = target_size
        self.all_images = [(obj_id, url) for obj_id, urls in images_dict.
↪items() for url in urls]

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

    def __getitem__(self, idx):
        anchor_obj_id, anchor_url = self.all_images[idx]
        anchor_img = load_image_from_url(anchor_url)
        if self.transform:
            anchor_img = self.transform(anchor_img)
```

```python
        positive_urls = self.images_dict[anchor_obj_id]
        if len(positive_urls) < 2:
            positive_img = anchor_img.clone()
        else:
            candidate_urls = [url for url in positive_urls if url != anchor_url]
            if candidate_urls:
                pos_url = random.choice(candidate_urls)
            else:
                pos_url = anchor_url
            positive_img = load_image_from_url(pos_url)
            if self.transform:
                positive_img = self.transform(positive_img)

        negative_obj_id = random.choice([oid for oid in self.object_ids if oid !
 ↪= anchor_obj_id])
        neg_url = random.choice(self.images_dict[negative_obj_id])
        negative_img = load_image_from_url(neg_url)
        if self.transform:
            negative_img = self.transform(negative_img)

        return anchor_img, positive_img, negative_img

# Define image transformations
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
```

[15]:
```python
triplet_dataset = TripletDataset(images_dict, transform=transform)

# Split into train and test sets
train_size = int(0.8 * len(triplet_dataset))
test_size = len(triplet_dataset) - train_size
triplet_train_dataset, triplet_test_dataset = torch.utils.data.
 ↪random_split(triplet_dataset, [train_size, test_size])
triplet_train_loader = DataLoader(triplet_train_dataset, batch_size=16,␣
 ↪shuffle=True, num_workers=4)
triplet_test_loader = DataLoader(triplet_test_dataset, batch_size=16,␣
 ↪shuffle=False, num_workers=4)
```

[16]:
```python
class FeatureExtractor(nn.Module):
    def __init__(self):
        super(FeatureExtractor, self).__init__()
        self.cnn = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, padding=1),
```

```
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
            nn.Conv2d(64, 128, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2)
        )
        self.fc = nn.Sequential(
            nn.Linear(128 * 28 * 28, 512),
            nn.ReLU(inplace=True),
            nn.Linear(512, 128)
        )

    def forward(self, x):
        x = self.cnn(x)
        x = x.view(x.size(0), -1)
        x = self.fc(x)
        return x
```

[17]:
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = FeatureExtractor().to(device)
margin = 1.0
criterion = nn.TripletMarginLoss(margin=margin, p=2)
optimizer = optim.Adam(model.parameters(), lr=1e-3)
```

0.10 Training model with triplet loss function

[18]:
```
num_epochs = 11
for epoch in range(num_epochs):
    running_loss = 0.0
    for batch_idx, (anchor, positive, negative) in␣
↪enumerate(tqdm(triplet_train_loader, desc=f"Epoch {epoch+1}/{num_epochs}")):
        anchor, positive, negative = anchor.to(device), positive.to(device),␣
↪negative.to(device)
        optimizer.zero_grad()

        # Compute embeddings
        anchor_out = model(anchor)
        positive_out = model(positive)
        negative_out = model(negative)

        loss = criterion(anchor_out, positive_out, negative_out)
        loss.backward()
        optimizer.step()
```

```
        running_loss += loss.item()
    avg_loss = running_loss / len(triplet_train_loader)
    print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {avg_loss:.4f}")
    if(epoch % 5 == 0):
        torch.save(model.state_dict(), f"triplet_model_epoch{epoch}.pt")

    # Evaluate the model on the test set
    model.eval()
    test_loss = 0.0
    with torch.no_grad():
        for anchor, positive, negative in tqdm(triplet_test_loader,␣
↪desc=f"Epoch {epoch+1}/{num_epochs} - Testing"):
            anchor, positive, negative = anchor.to(device), positive.
↪to(device), negative.to(device)
            anchor_out = model(anchor)
            positive_out = model(positive)
            negative_out = model(negative)
            loss = criterion(anchor_out, positive_out, negative_out)
            test_loss += loss.item()
    avg_test_loss = test_loss / len(triplet_test_loader)
    print(f"Test Loss: {avg_test_loss:.4f}")


print("Training complete.")
```

```
Epoch 1/11: 100%|      | 145/145 [08:12<00:00,  3.40s/it]

Epoch [1/11], Loss: 0.5880

Epoch 1/11 - Testing: 100%|      | 37/37 [01:43<00:00,  2.81s/it]

Test Loss: 0.5996

Epoch 2/11: 100%|      | 145/145 [05:28<00:00,  2.26s/it]

Epoch [2/11], Loss: 0.5261

Epoch 2/11 - Testing: 100%|      | 37/37 [01:19<00:00,  2.14s/it]

Test Loss: 0.7084

Epoch 3/11: 100%|      | 145/145 [04:37<00:00,  1.91s/it]

Epoch [3/11], Loss: 0.5170

Epoch 3/11 - Testing: 100%|      | 37/37 [01:08<00:00,  1.85s/it]

Test Loss: 0.4872

Epoch 4/11: 100%|      | 145/145 [04:24<00:00,  1.82s/it]

Epoch [4/11], Loss: 0.5462
```

```
Epoch 4/11 - Testing: 100%|     | 37/37 [01:07<00:00,  1.82s/it]
Test Loss: 0.6398
Epoch 5/11: 100%|     | 145/145 [04:29<00:00,  1.86s/it]
Epoch [5/11], Loss: 0.5368
Epoch 5/11 - Testing: 100%|     | 37/37 [01:10<00:00,  1.91s/it]
Test Loss: 0.4366
Epoch 6/11: 100%|     | 145/145 [04:31<00:00,  1.87s/it]
Epoch [6/11], Loss: 0.5100
Epoch 6/11 - Testing: 100%|     | 37/37 [01:07<00:00,  1.82s/it]
Test Loss: 0.5094
Epoch 7/11: 100%|     | 145/145 [04:02<00:00,  1.67s/it]
Epoch [7/11], Loss: 0.4166
Epoch 7/11 - Testing: 100%|     | 37/37 [01:03<00:00,  1.72s/it]
Test Loss: 0.3756
Epoch 8/11: 100%|     | 145/145 [04:15<00:00,  1.76s/it]
Epoch [8/11], Loss: 0.3526
Epoch 8/11 - Testing: 100%|     | 37/37 [01:01<00:00,  1.66s/it]
Test Loss: 0.3378
Epoch 9/11: 100%|     | 145/145 [04:03<00:00,  1.68s/it]
Epoch [9/11], Loss: 0.3199
Epoch 9/11 - Testing: 100%|     | 37/37 [01:04<00:00,  1.75s/it]
Test Loss: 0.3420
Epoch 10/11: 100%|     | 145/145 [04:07<00:00,  1.71s/it]
Epoch [10/11], Loss: 0.2642
Epoch 10/11 - Testing: 100%|     | 37/37 [01:02<00:00,  1.69s/it]
Test Loss: 0.2602
Epoch 11/11: 100%|     | 145/145 [04:19<00:00,  1.79s/it]
Epoch [11/11], Loss: 0.2014
Epoch 11/11 - Testing: 100%|     | 37/37 [01:12<00:00,  1.96s/it]
Test Loss: 0.2251
Training complete.
```

0.11 Evaluating the models

[20]:
```
model1 = FeatureExtractor().to(device)
model1.load_state_dict(torch.load("triplet_model_epoch10.pt"))

model2 = SiameseNetwork().to(device)
model2.load_state_dict(torch.load("siamese_model_epoch5.pt"))
```

[20]: <All keys matched successfully>

[21]:
```
## Using the validation set to test the models

def evaluate_model(model, dataloader):
    model.eval()
    embeddings1 = []
    embeddings2 = []
    labels = []
    with torch.no_grad():
        for img1, img2, label in tqdm(dataloader):
            img1, img2 = img1.to(device), img2.to(device)
            output1, output2 = model(img1, img2)
            embeddings1.append(output1)
            embeddings2.append(output2)
            labels.append(label)
    embeddings1 = torch.cat(embeddings1)
    embeddings2 = torch.cat(embeddings2)
    labels = torch.cat(labels)
    return embeddings1, embeddings2, labels

siamese_embeddings1, siamese_embeddings2, siamese_labels = evaluate_model(model2, test_loader)

## Calculate the average cosine similarity for positive and negative pairs respectively
def calculate_cosine_similarity(embeddings1, embeddings2):
    cosine_similarity = nn.CosineSimilarity(dim=1)
    return cosine_similarity(embeddings1, embeddings2)

positive_indices = siamese_labels == 1
negative_indices = siamese_labels == 0

positive_similarities = calculate_cosine_similarity(siamese_embeddings1[positive_indices], siamese_embeddings2[positive_indices])
negative_similarities = calculate_cosine_similarity(siamese_embeddings1[negative_indices], siamese_embeddings2[negative_indices])
```

```
print(f"Average Cosine Similarity for Positive Pairs: {positive_similarities.
↪mean().item():.4f}")
print(f"Average Cosine Similarity for Negative Pairs: {negative_similarities.
↪mean().item():.4f}")
```

```
100%|          | 186/186 [07:09<00:00,  2.31s/it]
```

```
Average Cosine Similarity for Positive Pairs: 0.9815
Average Cosine Similarity for Negative Pairs: 0.2679
```

[23]:

```
cosine_similarities_similar = []
cosine_similarities_dissimilar = []

with torch.no_grad():
    for anchor, positive, negative in tqdm(triplet_test_loader):
        anchor, positive, negative = anchor.to(device), positive.to(device),␣
↪negative.to(device)
        anchor_out = model1(anchor)
        positive_out = model1(positive)
        negative_out = model1(negative)

        cos_sim_similar = F.cosine_similarity(anchor_out, positive_out, dim=1)
        cos_sim_dissimilar = F.cosine_similarity(anchor_out, negative_out,␣
↪dim=1)

        cosine_similarities_similar.append(cos_sim_similar)
        cosine_similarities_dissimilar.append(cos_sim_dissimilar)

avg_cos_sim_similar = torch.cat(cosine_similarities_similar).mean().item()
avg_cos_sim_dissimilar = torch.cat(cosine_similarities_dissimilar).mean().item()

print(f"Average Cosine Similarity (Anchor-Positive, Similar):␣
↪{avg_cos_sim_similar:.4f}")
print(f"Average Cosine Similarity (Anchor-Negative, Dissimilar):␣
↪{avg_cos_sim_dissimilar:.4f}")
```

```
100%|          | 37/37 [00:58<00:00,  1.58s/it]
```

```
Average Cosine Similarity (Anchor-Positive, Similar): 0.7788
Average Cosine Similarity (Anchor-Negative, Dissimilar): 0.2432
```