Image Similarity Search

Annoy (Approximate Nearest Neighbors oh yeah) is a C++ library with python bindings to search for points in space that are close to given query point

```
python
CopyEdit
import os
import torch
from torchvision import models,transforms
from PIL import Image
import torch.nn as nn
from annoy import AnnoyIndex
```

```
images_folder="D:/GANFORGE/training_set/training_set/dogs"
images=os.listdir(images_folder)
```

- `images_folder`: path to the folder containing dog images.
- `images`: list of filenames in that folder.

```
weights=models.ResNet18_Weights.IMAGENET1K_V1
model=models.resnet18(weights=weights)
model.fc = nn.Identity()
```

- Loading the ResNet18 model with pretrained ImageNet weights.
- We are replacing the final fully connected layer with identity (`nn.Identity()`), so it outputs 512-dimensional feature vectors instead of classification logits.

```
model.eval()
```

- Setting the model to evaluation mode

```
transform=transforms.Compose([
    transforms.Resize((224,224)),
    transforms.ToTensor()
])
```

- Defining image transformation to resize and convert to tensor.

```
annoy_index=AnnoyIndex(512,'angular')
```

- Creating an Annoy index for 512-dim vectors with angular distance

```
for i in range(len(images)):
    image=Image.open(os.path.join(images_folder,images[i]))
    input_tensor=transform(image).unsqueeze(0)
```

- For each image, opening and preprocessing it.
- Also adding batch dimension with `.unsqueeze(0)`.

```
    if input_tensor.size()[1]==3:
        output_tensor=model(input_tensor)
        annoy_index.add_item(i,output_tensor[0])
```

- Passing the image through model, and extracting 512-dim feature vector.
- Adding to Annoy index with item id = `i`.

```
    if i%100 ==0:
        print(f'Processed {i} images.')
```

- Printing the progress every 100 images.

```
annoy_index.build(10)
```

```
annoy_index.save('dog_index.ann')
```

- Building the index using 10 trees.
- Saving it to disk for future querying.

This part loads the saved Annoy index and queries it for similar images.

```

annoy_index=AnnoyIndex(512,'angular')
annoy_index.load('dog_index.ann')
```

- Loading the Annoy index from disk.

```
image_grid=Image.new('RGB',(1000,1000))
```

- Creating a blank image canvas (5×5 grid of 200×200 images).

```

for i in range(len(images)):
    image=Image.open(os.path.join(images_folder,images[i]))
    input_tensor=transform(image).unsqueeze(0)
```

- For each image, preprocessing again for feature extraction.

```

    if input_tensor.size()[1]==3:
        output_tensor=model(input_tensor)
        nns=annoy_index.get_nns_by_vector(output_tensor[0],24)
```

- Extracting feature and getting 24 nearest neighbors from Annoy.

```
        image=image.resize((200,200))
        image_draw= ImageDraw.Draw(image)
        image_draw.rectangle([(0,0),(199,199)],outline='red',width=8)
        image_grid.paste(image,((0,0)))

```

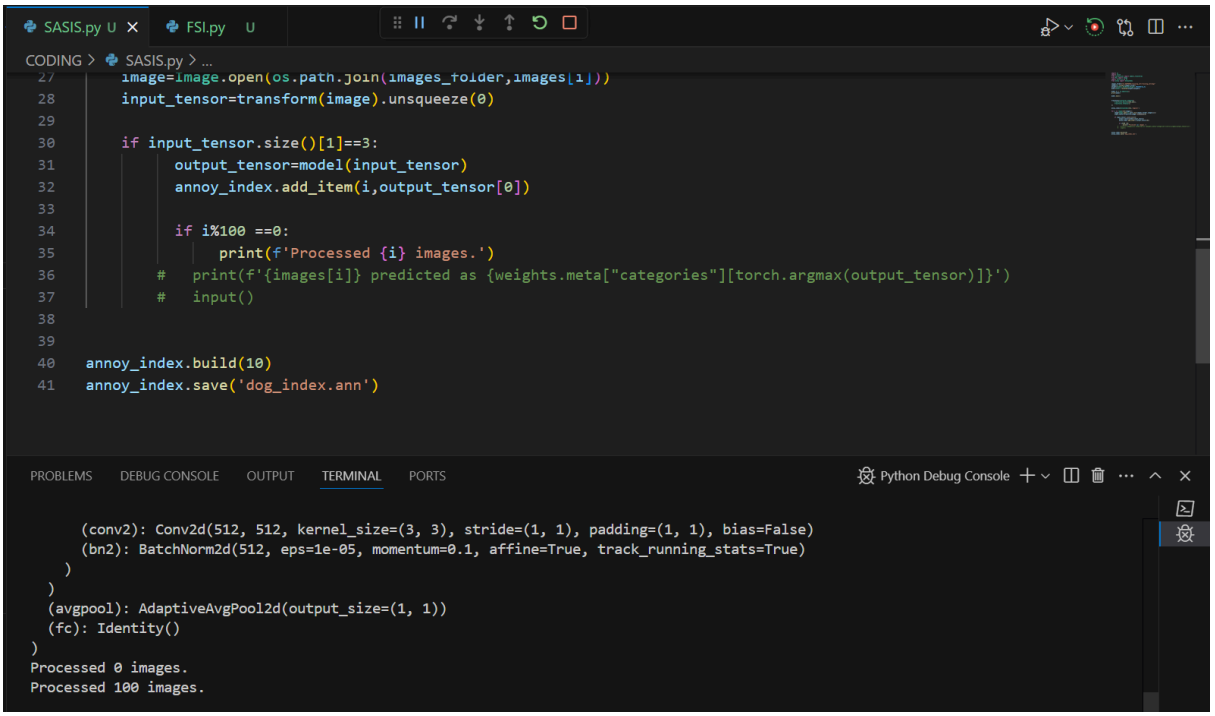
- Resized the query image, drew a red box around it, and pasted it in the top-left corner of the grid.

```
        for j in range(24):
            search_image=Image.open(os.path.join(images_folder,images[nns
[j]]))
            search_image=search_image.resize((200,200))
            image_grid.paste(search_image,(200 * ((j+1)%5),200*((j+1)//5)))
```

- Pasted each of the 24 nearest images in the grid.

```
image_grid.save(f'ImageDump/image_{i}.png')
```

- Saved the full grid for each query image to `ImageDump/`.



Adding the sample Images of the output


"Treat"
"Treat"


04/05/2005
EA & MORI
2007/02/01