Assignment #7

AATG013/ANT5013 Deep Learning and Computer Vision

Assigned: Tuesday, 5/2, 2023

Due: Tuesday, 5/9, 2023 (by midnight)

- Colab code 2 set을 사이버캠퍼스에 제출
  - assignment7-1.ipynb
  - assignment7-2.ipynb

code는 colab에서 수행되어 결과를 보여야 합니다.

Problem [1] PyTorch code: Transfer Learning

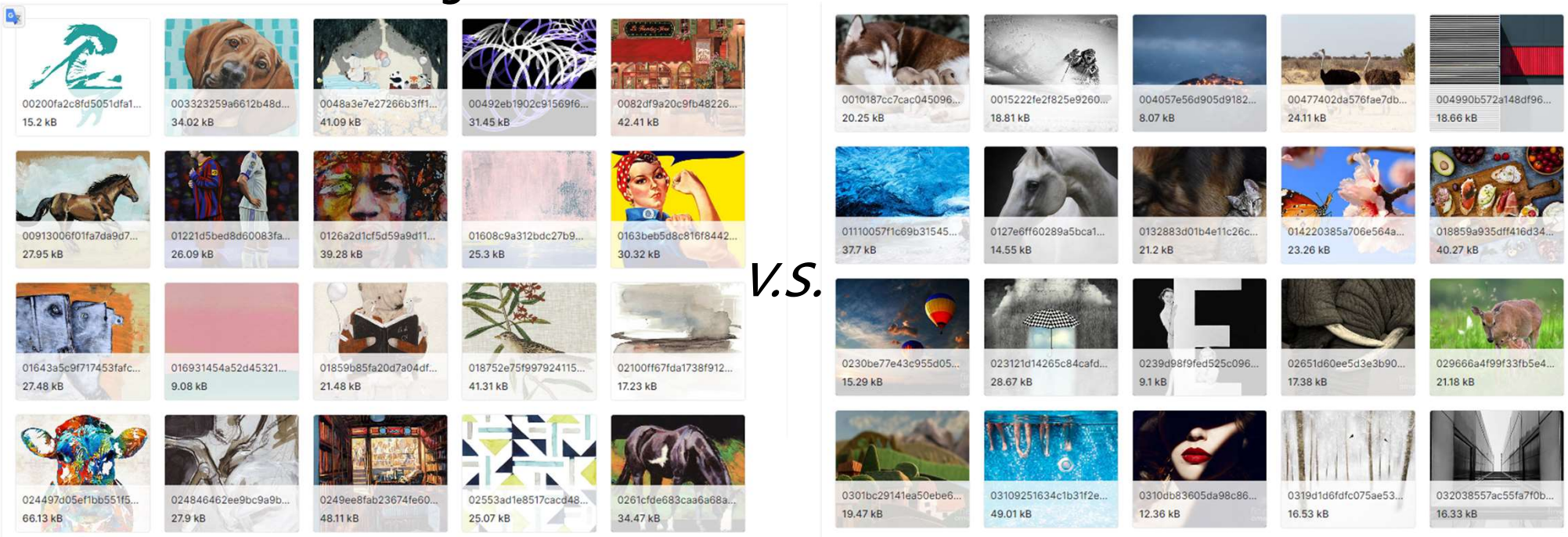
Problem [1] [70 points]

수업에서 pre-trained VGG-16모델을 활용하여, cat vs dog classification 문제를 해결하는 transfer learning방법에 대해서 공부하였다.

본 과제는 그 연장성상에 있으며 교재의 "Implementing_VGG16_for_image_classification.ipynb"코드를 과제에 맞게 변형하여 사용한다.

본 과제에서는 Kaggle의 "Painting vs Photograph Classification dataset"(https://www.kaggle.com/datasets/iiplutocrat45ii/painting-vs-photograph-classification-dataset)를 활용하여, painting 이미지와 photo 이미지를 classification/구분하는 문제를 다루려 한다.

Painting *V.S.* **Photo**



dataset	train images	validation images
	6,000	3,000

Problem [1] PyTorch code: Transfer Learning

Problem [1]

- Pretrained VGG-16 model을 transfer learning을 통해서 Binary classification을 수행한다. 본 rhk제에서는 아래 그림과 같이 2가지 경우에 대한 transfer learning 을 구현한다.

Case 1.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 224, 224]	1,792
ReLU-2	[-1, 64, 224, 224]	0
Conv2d-3	[-1, 64, 224, 224]	36,928
ReLU-4	[-1, 64, 224, 224]	0
MaxPool2d-5	[-1, 64, 112, 112]	0
Conv2d-6	[-1, 128, 112, 112]	73,856
ReLU-7	[-1, 128, 112, 112]	0
Conv2d-8	[-1, 128, 112, 112]	147,584
ReLU-9	[-1, 128, 112, 112]	0
MaxPool2d-10	[-1, 128, 56, 56]	0
Conv2d-11	[-1, 256, 56, 56]	295,168
ReLU-12	[-1, 256, 56, 56]	0
Conv2d-13	[-1, 256, 56, 56]	590,080
ReLU-14	[-1, 256, 56, 56]	0
Conv2d-15	[-1, 256, 56, 56]	590,080
ReLU-16	[-1, 256, 56, 56]	0
MaxPool2d-17	[-1, 256, 28, 28]	0
Conv2d-18	[-1, 512, 28, 28]	1,180,160
ReLU-19	[-1, 512, 28, 28]	0
Conv2d-20	[-1, 512, 28, 28]	2,359,808
ReLU-21	[-1, 512, 28, 28]	0
Conv2d-22	[-1, 512, 28, 28]	2,359,808
ReLU-23	[-1, 512, 28, 28]	0
MaxPool2d-24	[-1, 512, 14, 14]	0
Conv2d-25	[-1, 512, 14, 14]	2,359,808
ReLU-26	[-1, 512, 14, 14]	0
Conv2d-27	[-1, 512, 14, 14]	2,359,808
ReLU-28	[-1, 512, 14, 14]	0
Conv2d-29	[-1, 512, 14, 14]	2,359,808
ReLU-30	[-1, 512, 14, 14]	0
MaxPool2d-31	[-1, 512, 7, 7]	0
AdaptiveAvgPool2d-32	[-1, 512, 1, 1]	0
Flatten-33	[-1, 512]	0
Linear-34	[-1, 128]	65,664
ReLU-35	[-1, 128]	0
Dropout-36	[-1, 128]	0
Linear-37	[-1, 1]	129
Sigmoid-38	[-1, 1]	0

Total params: 14,780,481
Trainable params: 65,793
Non-trainable params: 14,714,688

Classification layers만 재학습

Case 2.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 224, 224]	1,792
ReLU-2	[-1, 64, 224, 224]	0
Conv2d-3	[-1, 64, 224, 224]	36,928
ReLU-4	[-1, 64, 224, 224]	0
MaxPool2d-5	[-1, 64, 112, 112]	0
Conv2d-6	[-1, 128, 112, 112]	73,856
ReLU-7	[-1, 128, 112, 112]	0
Conv2d-8	[-1, 128, 112, 112]	147,584
ReLU-9	[-1, 128, 112, 112]	0
MaxPool2d-10	[-1, 128, 56, 56]	0
Conv2d-11	[-1, 256, 56, 56]	295,168
ReLU-12	[-1, 256, 56, 56]	0
Conv2d-13	[-1, 256, 56, 56]	590,080
ReLU-14	[-1, 256, 56, 56]	0
Conv2d-15	[-1, 256, 56, 56]	590,080
ReLU-16	[-1, 256, 56, 56]	0
MaxPool2d-17	[-1, 256, 28, 28]	0
Conv2d-18	[-1, 512, 28, 28]	1,180,160
ReLU-19	[-1, 512, 28, 28]	0
Conv2d-20	[-1, 512, 28, 28]	2,359,808
ReLU-21	[-1, 512, 28, 28]	0
Conv2d-22	[-1, 512, 28, 28]	2,359,808
ReLU-23	[-1, 512, 28, 28]	0
MaxPool2d-24	[-1, 512, 14, 14]	0
Conv2d-25	[-1, 512, 14, 14]	2,359,808
ReLU-26	[-1, 512, 14, 14]	0
Conv2d-27	[-1, 512, 14, 14]	2,359,808
ReLU-28	[-1, 512, 14, 14]	0
Conv2d-29	[-1, 512, 14, 14]	2,359,808
ReLU-30	[-1, 512, 14, 14]	0
MaxPool2d-31	[-1, 512, 7, 7]	0
AdaptiveAvgPool2d-32	[-1, 512, 1, 1]	0
Flatten-33	[-1, 512]	0
Linear-34	[-1, 128]	65,664
ReLU-35	[-1, 128]	0
Dropout-36	[-1, 128]	0
Linear-37	[-1, 1]	129
Sigmoid-38	[-1, 1]	0

Total params: 14,780,481
Trainable params: 65,793
Non-trainable params: 14,714,688

Classification layers과 마지막 conv-block까지 재학습

- blue 영역: frozen layers
- red 영역: trainable layers

Problem [1] PyTorch code: Transfer Learning

Problem [1-1] [30 points]

교재의 "cat vs dog dataset"과 "painting vs photo dataset"의 데이터 저장 구조가 약간 다르다. 따라서 이를 반영하여 코드를 변경해야 한다. Kaggle에서 painting vs photo dataset 를 다운받는 코드는 **Transfer_learning_case1.ipynb**에 이미 반영되어 있다.

- Case 1의 transfer learning을 수행하기 위하여 **Transfer_learning_case1.ipynb**의 아래 클래스의 코드를 완성하고,
  - train-dataset과 validation dataset의 accuracy 비교 그래프를 출력하시오.

Case 1의 숙제는 아래 부분의 코드에서 photos와 painting의 해당 데이터들의 경로를 설정하면되고, **self.targets**은 photo이미지는 **True**의 값을 가지고, painting 이미지는 **False**의 값을 가지도록 코드를 작성하시오. .

```
class PhotosPainting(Dataset):
    def __init__(self, folder, num_images = 1500):
        # 아래의 코드를 완성하세요.
        photos = # photo images 경로 설정
        painting = # painting images 경로 설정
        self.fpaths = photos[:num_images] + painting[:num_images]
        self.normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],\
                                               std=[0.229, 0.224, 0.225])
        from random import shuffle, seed; seed(10); shuffle(self.fpaths)
        # 아래의 코드를 완성하세요.
        self.targets = # photo and painting images의 label (true/false) 설정
    def __len__(self): return len(self.fpaths)
    def __getitem__(self, ix):
        f = self.fpaths[ix]
        target = self.targets[ix]
        im = (cv2.imread(f)[:,:,::-1])
        im = cv2.resize(im, (224,224))
        im = torch.tensor(im/255)
        im = im.permute(2,0,1)
        im = self.normalize(im)
        return im.float().to(device), torch.tensor([target]).float().to(device)
```

Problem [1] PyTorch code: Transfer Learning

Problem [1-2] [40 points]

Case 2는 아래 그림의 적색 블록으로 표시한 부분을 재학습하는 문제이다. Case 2는 Fully-connected layers로 정의된 classification layers 뿐만 아니라, 마지막 Convolution-block의 convolution layers도 재학습을 함으로써, image feature도 부분적으로 재학습한다. 제공된 코드(**Transfer_learning_case2.ipynb**)의 **get_model()**을 완성하고, Case 2의 accuracy graph를 plot하여 Case 1과 성능을 비교하시오.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 224, 224]	1,792
ReLU-2	[-1, 64, 224, 224]	0
Conv2d-3	[-1, 64, 224, 224]	36,928
ReLU-4	[-1, 64, 224, 224]	0
MaxPool2d-5	[-1, 64, 112, 112]	0
Conv2d-6	[-1, 128, 112, 112]	73,856
ReLU-7	[-1, 128, 112, 112]	0
Conv2d-8	[-1, 128, 112, 112]	147,584
ReLU-9	[-1, 128, 112, 112]	0
MaxPool2d-10	[-1, 128, 56, 56]	0
Conv2d-11	[-1, 256, 56, 56]	295,168
ReLU-12	[-1, 256, 56, 56]	0
Conv2d-13	[-1, 256, 56, 56]	590,080
ReLU-14	[-1, 256, 56, 56]	0
Conv2d-15	[-1, 256, 56, 56]	590,080
ReLU-16	[-1, 256, 56, 56]	0
MaxPool2d-17	[-1, 256, 28, 28]	0
Conv2d-18	[-1, 512, 28, 28]	1,180,160
ReLU-19	[-1, 512, 28, 28]	0
Conv2d-20	[-1, 512, 28, 28]	2,359,808
ReLU-21	[-1, 512, 28, 28]	0
Conv2d-22	[-1, 512, 28, 28]	2,359,808
ReLU-23	[-1, 512, 28, 28]	0
MaxPool2d-24	[-1, 512, 14, 14]	0
Conv2d-25	[-1, 512, 14, 14]	2,359,808
ReLU-26	[-1, 512, 14, 14]	0
Conv2d-27	[-1, 512, 14, 14]	2,359,808
ReLU-28	[-1, 512, 14, 14]	0
Conv2d-29	[-1, 512, 14, 14]	2,359,808
ReLU-30	[-1, 512, 14, 14]	0
MaxPool2d-31	[-1, 512, 7, 7]	0
AdaptiveAvgPool2d-32	[-1, 512, 1, 1]	0
Flatten-33	[-1, 512]	0
Linear-34	[-1, 128]	65,664
ReLU-35	[-1, 128]	0
Dropout-36	[-1, 128]	0
Linear-37	[-1, 1]	129
Sigmoid-38	[-1, 1]	0

Total params: 14,780,481
Trainable params: 65,793
Non-trainable params: 14,714,688

Classification layers과 마지막 conv-block까지 재학습

```
def get_model():
    model = models.vgg16(pretrained=True)
    # Hint: print(model)로 model 구조를 출력하고, 적색 부분 layer
    # number를 확인하시오. 그리고 아래의 코드를  완성하시오.
    print(model)
    #
    for param in model.parameters():
    #
    # 청색부분만 parameter를 고정
    #
    #

    model.avgpool = nn.AdaptiveAvgPool2d(output_size=(1,1))
    model.classifier = nn.Sequential(nn.Flatten(),
    nn.Linear(512, 128),
    nn.ReLU(),
    nn.Dropout(0.2),
    nn.Linear(128, 1),
    nn.Sigmoid())
    loss_fn = nn.BCELoss()

    optimizer = torch.optim.Adam(model.parameters(), lr= 1e-3)
    return model.to(device), loss_fn, optimizer
```