[SKT AI Course: Deep Learning Basics]

Practice #4: Transfer Learning

Fine-tuning, Pre-trained Model as Feature Extractor

TA: Jun-Sik Choi & Jee-Seok Yoon

Instructor: Heung-Il Suk

hisuk@korea.ac.kr

http://www.ku-milab.org

Department of Brain and Cognitive Engineering,
Korea University

October 26, 2017

Contents

Fine-tuning Pre-trained Model

Pre-trained Model

- 18-layer ResNet
  - Originally Trained with ImageNet with 1,000 Classes
  - Use residual connection for efficient training and better performance
  - Top-1 error on ImageNet validation: 27.88%
  - For detailed architecture ␣ Pretrained_ResNet_St
  - We will fine-tune this model to classify Bee/Ant

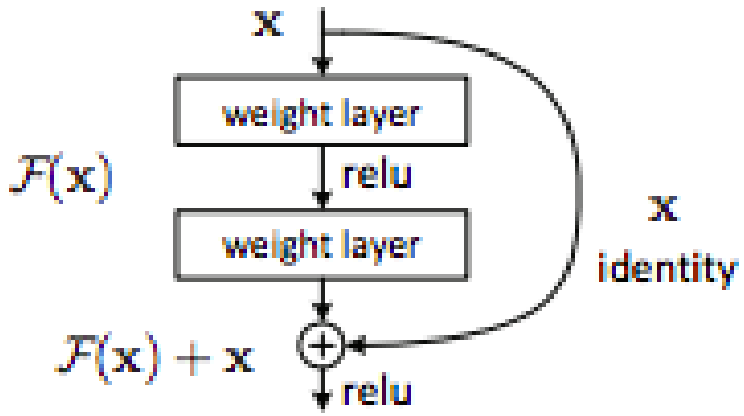


Figure 1. Residual connection

torchvision.transforms

- **Torchvision** provides simple ways to transform images and tensors.
  - transforms.**Compose** : Composes transforms together
  - transforms.**Scale** : Rescale the input PIL.Image to the given size.
  - transforms.**CenterCrop** : Crops the given PIL.Image at the center.
  - transforms.**RandomCrop** : Crop the given PIL.Image at a random location.
  - transforms.**RandomSizedCrop** : Crop the given PIL.Image to random size and aspect ratio.
  - transforms.**Pad** : Pad the given PIL.Image on all sides with the given “pad” value.
  - transforms.**Normalize** : Normalize an tensor image with mean and standard deviation.
  - transforms.**ToTensor** : Convert a PIL.Image or numpy.ndarray to tensor.
  - transforms.**ToPILImage** : Convert a tensor to PIL Image.
    - You can apply several transformations together to given dataset using **Compose**

Task #1 data_load.py

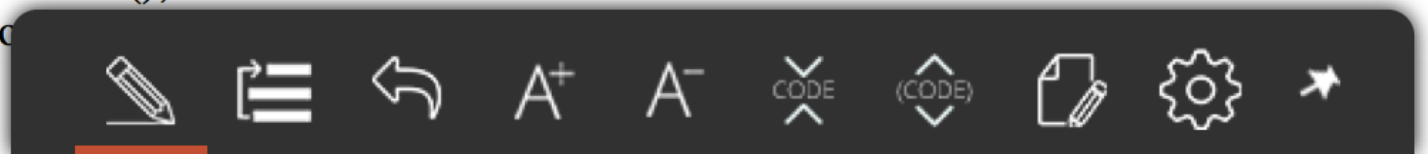
- Apply transforms (Scale, Centorcrop, ToTensor, Normalize) to 'Validation' Dataset

```
    # Define data transformer
    data_transforms = {
        # Transformation for training dataset
        'train': transforms.Compose([
            transforms.RandomSizedCrop(224),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406],
[0.229, 0.224, 0.225])
        ]),
        '''*** TASK 1 ***
        Apply transforms (Scale, Centorcrop, ToTensor, Normalize) to
'Validation' Dataset
        '''
        # Transformation for validation dataset
    }

```

- Scale Image to size 256 x 256 x 3
- Centercrop Image to size 224 x 224 x 3
- Transform Image to Tensor
- Normalize Tensor
  - Mean : [0.485, 0.456, 0.406]
  - Std : [0.229, 0.224, 0.225]

Answer

```
    # Define data transformer
    data_transforms = {
        # Transformation for training dataset
        'train': transforms.Compose([
            transforms.RandomSizedCrop(224),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406],
[0.229, 0.224, 0.225])
        ]),
      '''*** TASK 1 ***
       Apply transforms (Scale, Centorcrop, ToTensor, Normalize) to
    'Validation' Dataset
      '''
        # Transformation for validation dataset
        'val': transforms.Compose([
            transforms.Scale(256),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
            transforms.Nc
        ]),
    }
```



Load and modify model

- *load_state_dict(state_dict)*
  - Copies parameters and buffers from **state_dict** into this module and its descendants.
  - The keys of **state_dict** must exactly match the keys returned by this module's **state_dict()** function.

```
1  import torch
2
3  resnet = torch.load('./data/models/resnet18-5c1
4
5  type(resnet)
6
7  >> collections.OrderedDict
8
```

▭ loaded model is an **Ordered Dictionary**.

Task #2 finetuning.py

```
    # 마지막 레이어를 타겟 task에 맞게 수정
    model_ft.fc = nn.Linear(num_ftrs, 2)
    print("Modified Fully connected layer of resnet18 (Last layer):", mo

    if use_gpu:
        model_ft = model_ft.cuda()

    # Loss function 설정
    criterion = nn.CrossEntropyLoss()

    '''***TASK #2***
    define SGD Optimizer (learning rate=0.001, momentum=0.9)
    '''
    # 모델의 Optimizer 설정
    #optimizer_ft =
```

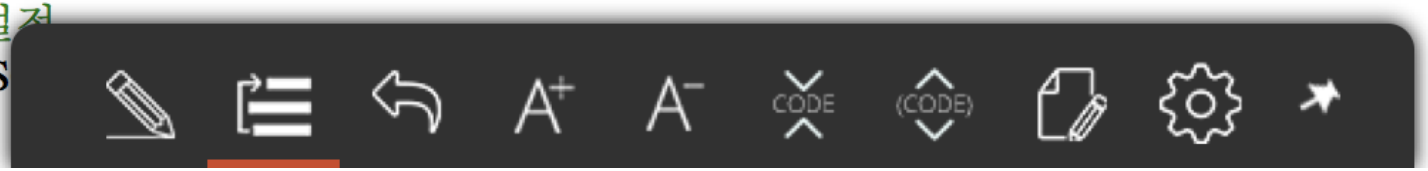
- Task 2.1
  Using *load_state_dict*,
  Copy parameters from
  './data/models/resnet18-5c106cde.pth'
  to model_ft

- ** Check How we change **the last layer (fully connected layer)** of the pre-trained model.

- Task 2.2
  Define SGD Optimizer
  - Learning rate = 0.001
  - momentum = 0.9

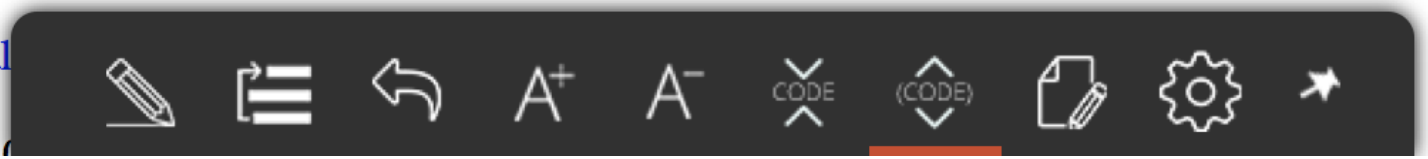
Answer

```
# Pretrain된 18 layer residual network를 로드
model_ft = torchvision.models.resnet18()
# Copy parameter from './data/models/resnet18-5c106cde.pth' to model_ft
model_ft.load_state_dict(torch.load('./data/models/resnet18-5c106cde.pth'))

# 마지막 레이어의 입력 feature 수
print("Original Fully connected layer of resnet18 (Last layer):", model_ft.fc)
num_ftrs = model_ft.fc.in_features
# 마지막 레이어를 타겟 task에 맞게 수정
model_ft.fc = nn.Linear(num_ftrs, 2)
print("Modified Fully connected layer of resnet18 (Last layer):", model_ft.fc)


if use_gpu:
    model_ft = model_ft.cuda()
# Loss function 설정
criterion = nn.CrossEntropyLoss()

# 모델의 Optimizer 설정
optimizer_ft = optim.S
```



Train_model

```
'''
Training
'''
def train_model(model, criterion, optimizer, lr_scheduler, use_gpu, dset_loaders, dset_sizes, num_epochs=25):
    since = time.time()

    best_model = model
    best_acc = 0.0

    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)

        # 각 epoch 별로 train phase와 validation phase를 진행
        for phase in ['train', 'val']:
            if phase == 'train':
                optimizer = lr_scheduler(optimizer, epoch)
                model.train(True)  # 모델을 train 모드로 설정
            else:
                model.train(Fal
```



Task #3

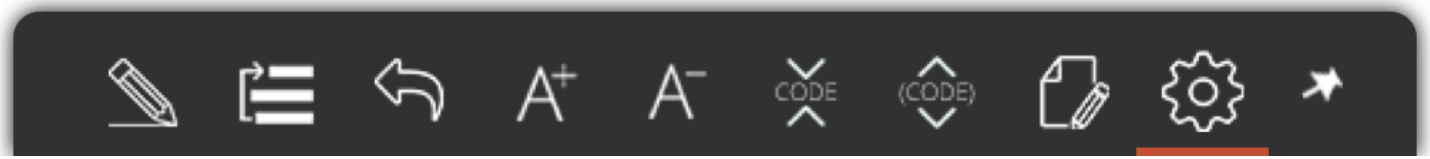
```
    # 마지막 레이어를 타겟 task에 맞게 수정
    model_ft.fc = nn.Linear(num_ftrs, 2)
    print("Modified Fully connected layer of resnet18 (Last layer):", mo

    if use_gpu:
        model_ft = model_ft.cuda()

    # Loss function 설정
    criterion = nn.CrossEntropyLoss()

    '''***TASK #2***
    define SGD Optimizer (learning rate=0.001, momentum=0.9)
    '''
    # 모델의 Optimizer 설정
    #optimizer_ft =
```

- Task 3.1
  Read *train_model()* and *exp_lr_scheduler()* in functions.py

- Task 3.2
  Write your own codes to train model_ft in finetuning.py

Answer

```
'''
3. Train & Evaluate
'''
# model training
model_ft = train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler,use_gpu, dset_loaders,dset_sizes,num_epochs=25)
```



Fine-tuning & evaluation

- Comment
  - #from feature_extractor import feature_extract
  - #feature_extract()

- Run main.py
  - Check the model predict well on new tasks (Bee / Ant Classification)

Pre-trained Model as a Feature Extractor

Use ResNet as Feature Extractor

- This time, we will use ResNet-18 as a feature extractor.
  - Not fine-tuning existing model, we freeze the ResNet-18 except the last lay er..
  - The pre-trained model only act as a feature extractor.
  - The last layer (fully-connected layer) is the only layer to be trained.

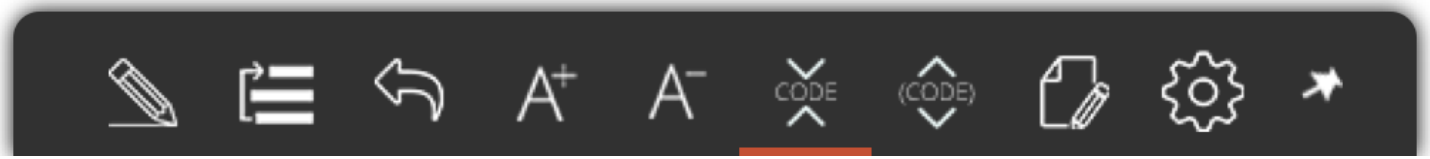
Task #4 feature_extractor.py

- If ***requires_grad = False*** for a certain parameter, the parameter is not affected during backpropagation.
- You can see the parameters of the model through ***model.parameters()***

```
'''
2. Model Load and Modify
'''
# Pretrain된 18 layer residual network를 로드
model_conv = torchvision.models.resnet18()
model_conv.load_state_dict(torch.load('./data/models/resnet18-5c106cde.pth'))
# Load한 모델의 파라미터를 train 과정에서 변경하지 않도록 설정 (requires_grad = False)

'''***TASK #4 ***
Set all params in model_conv, so the params are not affected during training
'''
for param in #here:
    #here
```

Answer

```
'''
2. Model Load and Modify
'''
# Pretrain된 18 layer residual network를 로드
model_conv = torchvision.models.resnet18()
model_conv.load_state_dict(torch.load('./data/models/resnet18-5c106cde.pth'))

# Load한 모델의 파라미터를 train 과정에서 변경하지 않도록 설정 (requires_grad = False)
for param in model_conv.parameters():
    param.requires_grad = False

# 마지막 레이어를 타겟 task에 맞게 수정
print("Original Fully connected layer of resnet18 (Last layer):", model_conv.fc)
num_ftrs = model_conv.fc.in_features
model_conv.fc = nn.Linear(num_ftrs, 2)
print("Modified Fully connected layer of resnet18 (Last layer):", model_conv.fc)
```



Training the last layer

- Comment
  - #finetuning()

- Run main.py

- Check the model predict well on new tasks (Bee / Ant Classificatio n)

Thank you

for your attention!!!

(Q & A)

hisuk (AT) korea.ac.kr

http://www.ku-milab.org