[12] 데이터 학습 방법

Transfer Learning

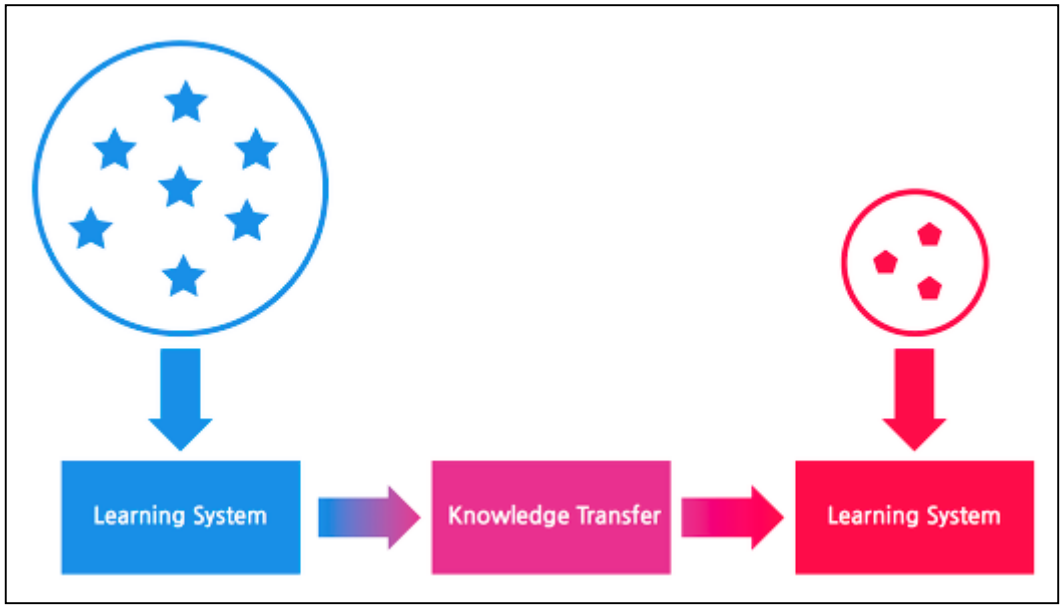
2018.07.31

작성자	검토자	승인자
이가영	변구훈	한미란

1. 전이학습 (Transfer Learning)

1.1 Transfer Learning 정의

- Deep learning을 feature extractor로만 사용하고 그렇게 추출한 feature를 가지고 다른 모델을 학습하는 방법이다.
- 기존에 특정 환경에서 만들어진 모델을 사용하여 새로운 모델을 만들 때 사용하여 학습을 빠르게 하며, 예측을 더 높이는 방법이다.



[그림] Transfer Learning 방법 구조도

1.2 Transfer Learning 원리

- 기존에 이미 잘 훈련된 모델이 있고, 특히 해당 모델과 유사한 문제를 해결할 때 transfer learning을 사용한다. Transfer learning을 사용하는 이유는 다음과 같다.
  - ■ 실질적으로 Convolution network을 처음부터 학습시키는 일은 많지 않다. 대부분의 문제는 이미 학습된 모델을 사용해서 문제를 해결할 수 있다.
  - ■ Layers의 개수, activation, hyper parameters 등등 고려해야 할 사항들이 많으며, 실질적으로 처음부터 학습시키려면 많은 시도가 필요하다.
  - ■ 복잡한 모델일수록 학습을 시키는데 오랜 시간이 걸린다.
- 일반적으로 이미 사전에 학습이 완료된 모델(Pre training model)을 가지고 우리가 원하는 학습에 미세 조정, 즉 작은 변화를 이용하여 학습을 시킨다.

- ImageNet과 같은 대형 데이터 셋을 사용해서 pretrain된 ConvNet을 이용하는데 크게 두 가지의 시나리오가 있다.
  - ■ ConvNet as fixed feature extractor.
    - ◆ 마지막 Fully connected된 계층을 제외한 모든 신경망의 가중치를 고정한다. Fully connected 계층은 새로운 가중치의 계층으로 대체를 하고, 이 계층만 학습한다.
    - ◆ Pretrained CNN에서 마지막 classification layer (보통 softmax layer)만 제거하면 feature extractor로 사용할 수 있다. 이렇게 추출한 feature들은 CNN codes라고 부른다.
  - ■ Fine-tuning the ConvNet
    - ◆ 신경망을 ImageNet 1000 데이터 셋 등으로 미리 학습한 신경망으로 초기화한다. 학습의 나머지 과정들은 기존의 학습 방법과 동일하다.
    - ◆ 마지막 classifier layer만을 retrain하는 것이 아니라 pretrain된 전체 네트워크를 재조정(fine-tuning)한다.
    - ◆ 상대적으로 사용할 수 있는 데이터가 많을 때 적합하다.
    - ◆ 경우에 따라 앞쪽 레이어는 고정시키고 뒤쪽 레이어만 fine-tuning하기도 한다.

1.3 Transfer Learning 의 장점

- 훈련 데이터를 새로 수집하거나 모델을 다시 만드는 과정에서 발생하는 비용을 줄일 수 있다.
  - ■ 특히 데이터가 자주 갱신될 경우 매우 많은 비용을 절감할 수 있다.
- 데이터가 부족한 분야에 transfer learning을 적용을 하면 더 좋은 성능을 추출할 수 있다.

1.4 Transfer Learning 사용 예시

- 아프리카 국가들의 빈곤 지도
  - ■ 빈곤과 관련된 데이터는 매우 부족하다. 인공위성 데이터 중 야간에 발생하는 조명의 정도를 부의 분포를 분류하는 지표로 활용하지만, 야간 조명만으로 부의 분포를 세부적으로 파악하기에는 유용하지 않기 때문에 한계가 있다.
  - ■ 이러한 한계를 극복하기 위해 풍부한 야간 조명 데이터를 활용한 전이 학습 모델로 빈곤과 연관할 수 있는 개체를 식별하는 알고리즘을 고안했다.
- 카사바(열대지방의 식량) 질병 진단
  - ■ 열대지방에서 중요한 식량 중 하나인 카사바는 바이러스에 취약한 작물이다. 따라서 카사바 잎을 카메라로 인식해 질병 발생 유무를 진단하는 어플리케이션을 만들었다.

Transfer Learning		기 능	연구자료	단 계	연 구
		작 성 자	이가영	페 이 지	5/ 9
문서번호	ML-01-012	작 성 일	2018-07-31	버 전	Ver 0.1

- 하지만 질병에 걸린 카사바 잎 사진을 머신러닝으로 학습시킬 만큼 확보하기 어렵다. 따라서 전이 학습을 활용하도록 한다.
- 총 2756개의 카사바 사진을 확보한 후, 전이 학습 모델을 사용하여 카사바 갈색 줄무늬 병과 카사바 모자이크 바이러스를 구분하도록 AI를 훈련했고, 98%의 정확도로 갈색 잎, 96%의 정확도로 붉은 잎 진드기를 확인했다.
- 전이학습을 이용하여 2756개의 부족한 데이터로 높은 정확도로 질병을 확인하는 모델을 구축할 수 있었다.

2.Transfer Learning 실습 예제

2.1 테스트 환경

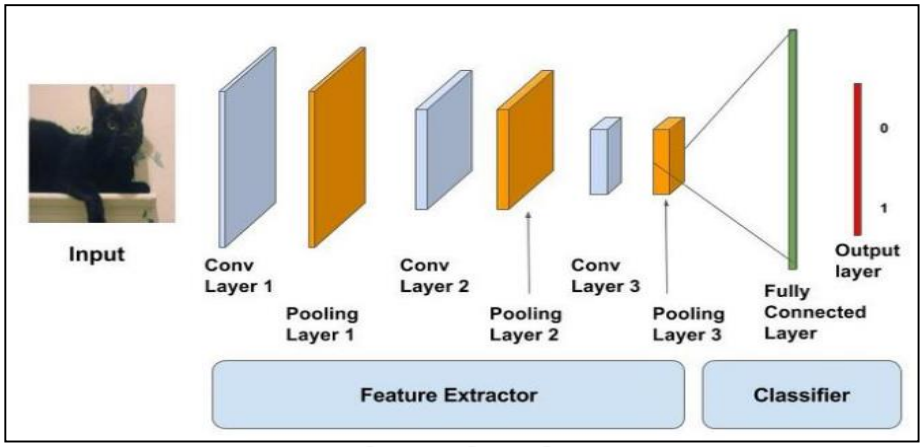
- Os : Windows10
- Jupyter notebook

2.2 라이브러리 설치

- $ pip install keras

2.3 예제 설명

- 본 예제에서는 토마토, 호박, 수박 이 세 가지를 transfer learning을 이용하여 분류하는 문제이다.
- 이 예제에서 convolution layer는 feature 추출기의 역할을 하고 fully connected layer는 분류자의 역할을 한다.
- 다음 실행할 모델의 구조도는 다음과 같다.



[그림] 학습 모델 layer 구조

- 현재 가지고 있는 데이터(토마토, 호박, 수박)의 set이 작으므로 이미 학습이 되어 있는 유사한 모델을 사용하는 transfer learning을 이용하여 학습을 진행한다.

2.3.1 데이터 다운로드 및 전처리

- ImageNet을 이용하여 토마토, 호박, 수박 세 가지의 사진을 분류하여 다운받는다.
- 다운을 받은 후 학습에 적절하지 않는 데이터를 분류하는 전처리 과정을 거친다.

2.3.2 pre-trained model 불러오기

```
from keras.applications import VGG16

vgg_conv = VGG16(weights='imagenet',
                 include_top=False,
                 input_shape=(224, 224, 3))
```

- keras의 VGG16 모델을 불러온다.
  - 이 모델에서 classifier 역할을 하는 마지막 두 개의 fully-connected layer는 로드하지 않는다.

2.3.3 Extract Features

```
datagen = ImageDataGenerator(rescale=1./255)
batch_size = 20

train_features = np.zeros(shape=(nTrain, 7, 7, 512))
train_labels = np.zeros(shape=(nTrain,3))

train_generator = datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=shuffle)
```

- 수집한 데이터를 80 : 20의 비율로 나눈 후 train 폴더와 validation 폴더에 각 라벨 별로 분류하여 넣는다.
- 이후 ImageDataGenerator 클래스를 이용하여 이미지를 로드한 후, flow_from_directory 함수를 이용하여 이미지 및 레이블의 배치를 생성한다.

```
i = 0
for inputs_batch, labels_batch in train_generator:
    features_batch = vgg_conv.predict(inputs_batch)
    train_features[i * batch_size : (i + 1) * batch_size] = features_batch
    train_labels[i * batch_size : (i + 1) * batch_size] = labels_batch
    i += 1
    if i * batch_size >= nImages:
        break

train_features = np.reshape(train_features, (nTrain, 7 * 7 * 512))
```

- model.predict() 함수를 이용해서 이미지를 전달하면 결과적으로 7 * 7 * 512 차원의 Tensor가 된다.

2.3.4 Create own model

```
from keras import models
from keras import layers
from keras import optimizers

model = models.Sequential()
model.add(layers.Dense(256, activation='relu', input_dim=7 * 7 * 512))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(3, activation='softmax'))
```

- 3개의 클래스를 가진 softmax 출력 layer로 feed forward network를 만든다.

2.3.5 Train the model

```
model.compile(optimizer=optimizers.RMSprop(lr=2e-4),
              loss='categorical_crossentropy',
              metrics=['acc'])

history = model.fit(train_features,
                    train_labels,
                    epochs=20,
                    batch_size=batch_size,
                    validation_data=(validation_features,validation_labels))
```

- Keras의 model.fit() 함수를 이용하여 학습을 진행한다.

2.4 실행 결과

- 학습이 진행되는 과정은 다음과 같다.
  - ■ 학습은 총 20번 진행된다.

```
Train on 600 samples, validate on 150 samples
Epoch 1/20
600/600 [==============================] - 0s - loss: 2.2658 - acc: 0.5583 - val_loss: 0.9076 - val_acc: 0.6267
Epoch 2/20
600/600 [==============================] - 0s - loss: 0.4247 - acc: 0.8533 - val_loss: 0.7782 - val_acc: 0.7133
Epoch 3/20
600/600 [==============================] - 0s - loss: 0.3868 - acc: 0.8567 - val_loss: 0.4165 - val_acc: 0.8333
Epoch 4/20
600/600 [==============================] - 0s - loss: 0.2655 - acc: 0.9100 - val_loss: 0.4703 - val_acc: 0.8667
Epoch 5/20
600/600 [==============================] - 0s - loss: 0.1497 - acc: 0.9367 - val_loss: 0.5814 - val_acc: 0.8467
Epoch 6/20
600/600 [==============================] - 0s - loss: 0.0710 - acc: 0.9717 - val_loss: 0.3836 - val_acc: 0.8867
Epoch 7/20
600/600 [==============================] - 0s - loss: 0.1263 - acc: 0.9567 - val_loss: 0.3535 - val_acc: 0.9067
Epoch 8/20
600/600 [==============================] - 0s - loss: 0.0308 - acc: 0.9933 - val_loss: 0.4506 - val_acc: 0.8733
Epoch 9/20
600/600 [==============================] - 0s - loss: 0.0972 - acc: 0.9700 - val_loss: 0.5249 - val_acc: 0.8467
Epoch 10/20
600/600 [==============================] - 0s - loss: 0.0636 - acc: 0.9783 - val_loss: 0.4594 - val_acc: 0.9067
Epoch 11/20
600/600 [==============================] - 0s - loss: 0.0119 - acc: 0.9983 - val_loss: 0.4150 - val_acc: 0.9067
Epoch 12/20
600/600 [==============================] - 0s - loss: 0.0252 - acc: 0.9967 - val_loss: 0.4691 - val_acc: 0.9067
Epoch 13/20
600/600 [==============================] - 0s - loss: 0.0737 - acc: 0.9767 - val_loss: 0.3852 - val_acc: 0.9133
Epoch 14/20
600/600 [==============================] - 0s - loss: 0.0063 - acc: 0.9983 - val_loss: 0.4188 - val_acc: 0.9067
Epoch 15/20
600/600 [==============================] - 0s - loss: 0.0046 - acc: 1.0000 - val_loss: 0.4766 - val_acc: 0.9133
Epoch 16/20
600/600 [==============================] - 0s - loss: 0.0079 - acc: 0.9983 - val_loss: 0.4681 - val_acc: 0.9000
Epoch 17/20
600/600 [==============================] - 0s - loss: 0.0020 - acc: 1.0000 - val_loss: 0.5510 - val_acc: 0.8867
Epoch 18/20
600/600 [==============================] - 0s - loss: 0.0037 - acc: 0.9983 - val_loss: 0.4843 - val_acc: 0.9000
Epoch 19/20
600/600 [==============================] - 0s - loss: 0.0069 - acc: 0.9983 - val_loss: 0.5764 - val_acc: 0.8733
Epoch 20/20
600/600 [==============================] - 0s - loss: 0.0175 - acc: 0.9933 - val_loss: 0.4407 - val_acc: 0.9067
```

[그림] 모델 학습 과정

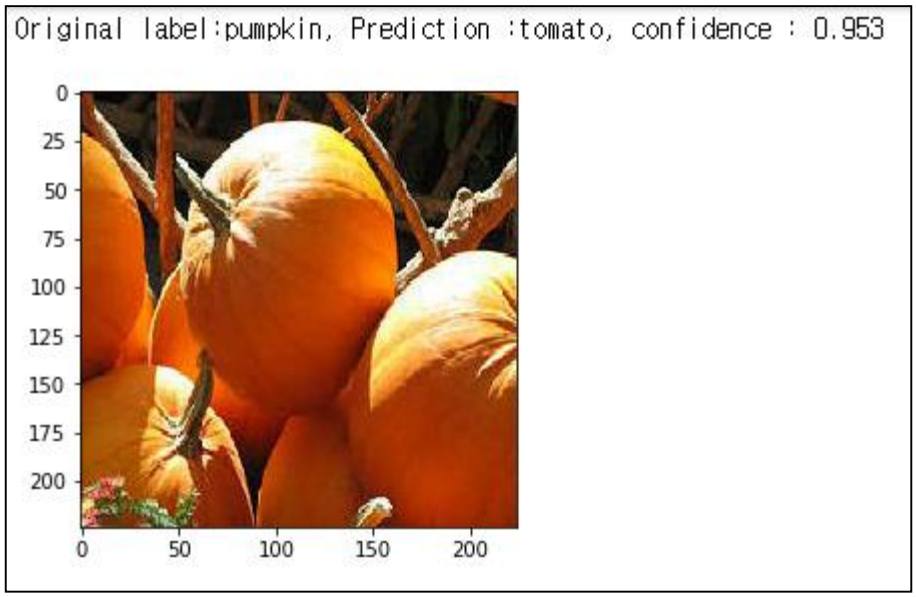
- 결과적으로 다음과 같은 출력이 나오는 것을 볼 수 있다.
  - ■ 150개의 이미지 중 14개의 이미지를 제외하고 정답을 찾아냈다.

```
errors = np.where(predictions != ground_truth)[0]
print("No of errors = {}/{}".format(len(errors),nVal))
```

```
No of errors = 14/150
```

Transfer Learning		기 능	연구자료	단 계	연 구
		작 성 자	이가영	페 이 지	9/ 9
문서번호	ML-01-012	작 성 일	2018-07-31	버 전	Ver 0.1

- 정답을 찾지 못한 이미지를 출력하는 예는 다음과 같다.



[그림] 정답이 아닌 이미지 출력 예

3.참고 문헌

1) 부족한 데이터로 하는 머신러닝! '전이 학습'
http://blog.lgcns.com/1563
2) 전이학습이란? Transfer Learning
https://sites.google.com/site/lifeiyagi/computer-science/jeon-ihagseub-ilantransferlearning
3) 전이학습(Transfer Learning) 튜토리얼
https://9bow.github.io/PyTorch-tutorials-kr-0.3.1/beginner/transfer_learning_tutorial.html
4) Opencv 배우기
https://www.learnopencv.com/keras-tutorial-transfer-learning-using-pre-trained-models/
5) https://github.com/spmallick/learnopencv
6)