기초 이론부터 실무 실습까지

머신 러닝 익히기

programmers

Part 09. 딥러닝

정 정 민

programmers

Chapter 22. 이미지 분류 실습

1. 문제 설정
2. 모델 설명
3. 데이터 로드 및 전처리
4. 결과 확인

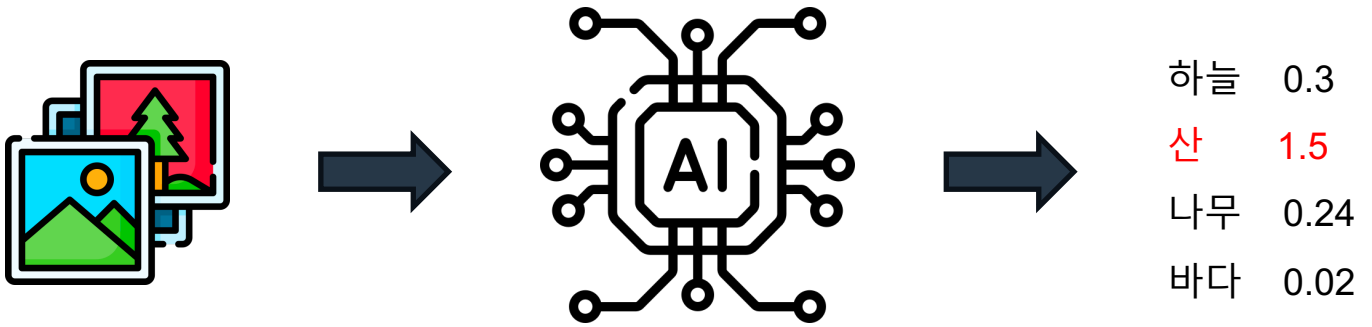
문제 설정

문제 설정

- 학습된 이미지 분류 모델을 활용해 이미지 분류 실습을 진행
  - 이미지 분류란,
    이미지의 대부분을 차지하는 객체의 종류를 예측하는 문제를 의미함

- 입력으로 하나의 이미지를 제공하고
- 해당 이미지의 종류(클래스)를 예측

- 일반적으로 선택 가능한 모든 클래스에 대한 정답 점수값을 예측
- 따라서 최종 결과는 그 점수 중 가장 큰 값을 선택하면서 얻을 수 있음
- 또한, 이 점수들을 활용해 확률 값을 취할 수 있음



모델 설명

ResNet

- 이번 실습에서 사용할 이미지 분류 모델

- 해당 모델은 연구적으로도 많이 사용되며
- 현업에서도 기본 모델로 많아 사용하는 유명한 모델

- 2015년에 제안된 딥러닝 모델로
- 당시 사람의 이미지 인식 능력을 넘어선 최초의 이미지 인식 모델

- 모델의 깊이인 Layer에 따라 18, 34, 50, 101, 152 이라는 5가지 종류가 존재
  - 깊이가 얕을수록 성능이 낮지만 빠르게 결과를 얻을 수 있음
  - 깊이가 깊을수록 성능이 높아지지만 결과를 도출하기 위한 시간이 오래

- 딥러닝 모델을 학습하는 시간은 이전 머신러닝 모델 대비 상당히 오래 걸림
- 따라서 Pytorch 내부에서 이미 학습 된 모델을 가져다가 활용! (링크)

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.

The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions[1], where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

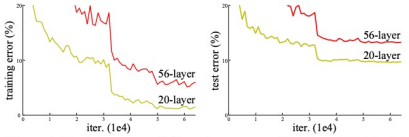


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

greatly benefited from very deep models.

Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious problem of vanishing/exploding gradients [1, 9], which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 9, 37, 13] and intermediate normalization layers [16], which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with backpropagation [22].

When deeper networks are able to start converging, a *degradation* problem has been exposed: with the network

https://arxiv.org/pdf/1512.03385.pdf

데이터 로드 및 전처리

강의 실습 코드 : 링크

데이터 가져오기

- 인터넷으로부터 이미지 url을 통해 이미지를 가져옴
- 단, webp 포멧의 데이터는 request 요청 거부 에러가 뜰 가능성이 있음

- 웹 상 이미지 링크 복사 후 url 변수에 할당!

```
def load_image_from_web(url):
    response = requests.get(url, stream=True).raw
    img = Image.open(response).convert('RGB')
    return img

def imshow(img):
    plt.imshow(img)
    plt.axis('off')

imshow(load_image_from_web(url))
```

이미지 전처리

- 머신러닝 모델은 학습 과정과 똑같은 전처리를 진행한 뒤 추론을 진행해야 함
- ResNet 모델도 마찬가지

- ResNet 모델이 학습할 당시의 전처리 과정이 저장된 class가 존재
  - ResNet 모델의 학습 결과물이 저장된 객체

- 만약 resnet18 모델을 사용한다면
- ResNet18_Weights 라는 이름의 클래스에 담겨있음 (링크)

- 따라서 전처리 코드를 가져와 사용할 이미지에 적용해야 함

```
import torchvision.models as models

model_name='resnet18'
weight_name = 'ResNet' + re.findall(r'\d+', model_name)[0] +'_Weights'
weights = getattr(models, weight_name).DEFAULT
transforms = weights.transforms()
```

결과 확인

강의 실습 코드 : 링크

모델 호출과 결과 추론

- 학습된 딥러닝 모델은 정해진 모델의 이름으로 호출이 가능
- 이미 호출한 weight 객체를 이용해 학습된 결과물을 학습이 완료된 상태로 변환

- 입력 이미지를 활용해 모델 추론을 진행

```
import torchvision.models as models

model_name='resnet18'
model = getattr(models, model_name)(weights=weights)
model.eval()

output = model(input_image)
```

출력 결과 후처리

- ResNet 모델은 입력된 이미지가 1,000개의 이미지 종류 중 어떤 부류에 속하는지를 판단하는 모델
- 따라서 모델의 출력은 1,000의 크기를 갖는 vector

- 이 vector의 값 중, 제일 큰 값을 갖는 위치의 class 가 실제 정답이 됨
  - 어떤 위치가 어떤 class 인지는 meta data의 형태로 weights 객체에 들어있음

- 또한, 출력으로 나온 결과 값을 이용해
- 딥러닝 모델이 얼마나 큰 정확도로 입력 이미지의 종류에 확신을 갖고 있는지 계산할 수 있음

```
output = model(input_image)
print(output.shape) # [1, 1000]

conf, predicted = F.softmax(output, dim=1).max(1)
total_class = meta_data["categories"]
cls = total_class[predicted.item()]
```

E.O.D