머신러닝

HW03

Implementation of ResNet using Pytorch

2019202050 이강현

Description

CIFAR-10(Canadian Institute For Advanced Research)는 기계학습 및 컴퓨터 비전에 널리 사용되는 데이터셋이다.

클래스 당 6000개의 이미지가 있고 클래스는 10개가 존재한다. 따라서 총 60000장의 데이터가 32x32 컬러 이미지로 구성되어 있다.

클래스는 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭이고 각각의 클래스당 사진의 개수가 같아 모델의 균형잡힌 학습에 도움을 줄 수 있는 데이터셋이다.

Network Architecture

ResNet은 Residual Network의 약자로 residual이라는 특성을 반영한 방법이다.

ResNet34에서는 하나의 block에 2개의 convolution layer가 들어있다.

이와 같은 block들이 총 16개 존재하며 총 34개의 layer들이 존재한다. 이렇게 많은 layer들을

이어내면 parameter의 수는 매우 커지고 따라서 큰 연산량을 요구한다. 따라서 차원을 늘렸다 줄였다하는 ResNet50에서 제안된 bottleneck block또한 존재한다.

또 ResNet은 identity Mapping이라는 방법을 제안한다.

Identity mapping이란 입력이 어떤 함수를 통과하더라도 다시 스스로가 나오는 개념이다.

이 방식의 효과를 정리하면 역전파를 진행할 때 곱에 의한 편미분이 아닌 합에 의한 편미분을 이용여 연산 복잡도를 낮출 수 있고 layer가 길어질 때 자주 발생하는 gradient vanishing 문제도 해결할 수 있다는 장점이 있다.

아래는 ResNet34를 직접 구현한 모습이다.

먼저 ResNetBlock이다.

```
ResNetBlock(
  (basicblock): Sequential(
    (0): Conv2d(in_channels, out_channels, kernel_size=(3, 3), stride=stride, padding=(1, 1), bias=False)
    (1): BatchNorm2d(out_channels, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(out_channels, out_channels, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (4): BatchNorm2d(out_channels, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (relu): ReLU()
  (downsample): Sequential(
    (0): Conv2d(in_channels, out_channels, kernel_size=(1, 1), stride=stride, bias=False)
    (1): BatchNorm2d(out_channels, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  ) if stride != 1 else Sequential()
)
```

제안된 3x3 커널 사이즈를 사용하였고 사이에 batch normalization과 relu함수가 있는 것을 확인한다. 추가로 block의 채널 수가 바뀔 때 stride를 2로 하여 사진의 크기가 달라지므로 이때 identity mapping개념을 적용하기 위해 원래 데이터의 크기 또한 element-wise sum을 위해 맞추어 주어야 한다. 이를 downsample이라고 한다.

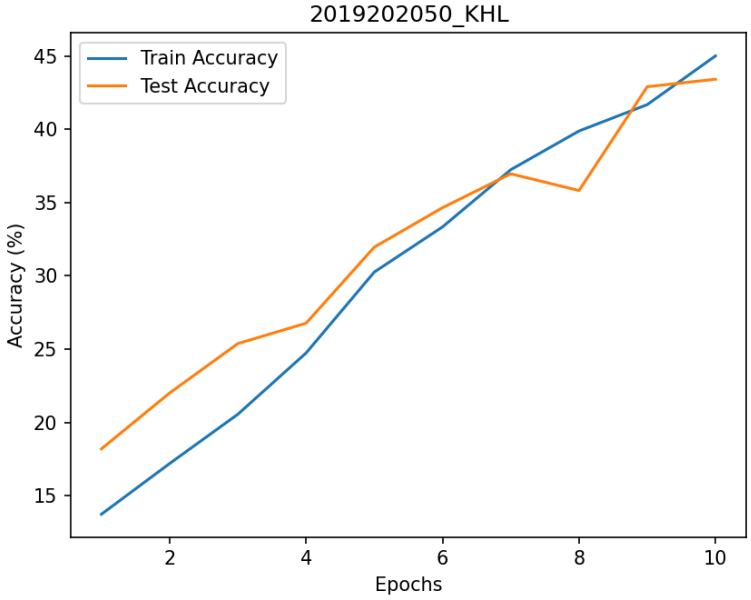
activation function이 적용되기 전 batch normalization을 적용하는 것 또한 Rasnet에서 사용된 방법이다. 이는 데이터 정규화 측면에서의 개선, 역전파 연산시 relu에서 truncated될 여지가 없다라고 설명한다.

다음은 ResNet이다.

```
ResNet(
  (conv1): Sequential(
    (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(5, 5))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (maxpool): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): ResNetBlock(64, 64, stride=1)
    (1): ResNetBlock(64, 64, stride=1)
    (2): ResNetBlock(64, 64, stride=1)
  )
  (layer2): Sequential(
    (0): ResNetBlock(64, 128, stride=2)
    (1): ResNetBlock(128, 128, stride=1)
    (2): ResNetBlock(128, 128, stride=1)
    (3): ResNetBlock(128, 128, stride=1)
  )
  (layer3): Sequential(
    (0): ResNetBlock(128, 256, stride=2)
    (1): ResNetBlock(256, 256, stride=1)
    (2): ResNetBlock(256, 256, stride=1)
    (3): ResNetBlock(256, 256, stride=1)
    (4): ResNetBlock(256, 256, stride=1)
    (5): ResNetBlock(256, 256, stride=1)
  )
  (layer4): Sequential(
    (0): ResNetBlock(256, 512, stride=2)
    (1): ResNetBlock(512, 512, stride=1)
    (2): ResNetBlock(512, 512, stride=1)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc1): Linear(in_features=512, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=10, bias=True)
)
```

ResNet34와 같이 동일한 block의 수를 유지했으나 마지막에 fully connected layer를 하나 더 추가하였다. 또한 사진의 크기가 일정하게 1/2로 감소하게 하고 싶어 초기 convolution layer에 padding을 5로 하였다.

Result




```
Epoch 1/10
Train Accuracy: 13.74 %
Test Accuracy: 18.2 %
Epoch 2/10
Train Accuracy: 17.2 %
Test Accuracy: 22.01 %
Epoch 3/10
Train Accuracy: 20.56 %
Test Accuracy: 25.38 %
Epoch 4/10
Train Accuracy: 24.74 %
Test Accuracy: 26.77 %
Epoch 5/10
Train Accuracy: 30.26 %
Test Accuracy: 31.96 %
Epoch 6/10
Train Accuracy: 33.34 %
Test Accuracy: 34.65 %
Epoch 7/10
Train Accuracy: 37.24 %
Test Accuracy: 36.96 %
Epoch 8/10
Train Accuracy: 39.88 %
Test Accuracy: 35.82 %
Epoch 9/10
Train Accuracy: 41.68 %
Test Accuracy: 42.9 %
Epoch 10/10
Train Accuracy: 45.0 %
Test Accuracy: 43.41 %
```

main.py에서 32x32라는 cifar-10 dataset의 이미지 특성상 resnet의 많은 parameter를 학습하기 쉽지 않을 것으로 판단하여 128x128로 resize하였고 정확도를 상승시킬 수 있었다. 추가적으로 256x256, 64 x 64로 resize하고 learning rate와 momentum, 추가적인 증강 기법들을 실험해보았으나 128x128 resize, learning rate 0.01, momentum 0.9가 랜덤하게 뽑는 데이터마다 편차는 존재하나 안정적으로 잘 나오는 것 같았다.

위 plot은 매 epoch마다 train accuracy와 test accuracy를 얻어내 plot한 것이다. epoch을 10으로 한다는 기준에 맞게 10번만 진행하였으나 train accuracy와 test accuracy가 epoch이 10일때도 여전히 우상향하는 경향을 보고 epoch을 20으로 하여 추가적인 실험을 진행했으나 test accuracy는 epoch 10이 넘어가니 50% 정도의 최고 정확도를 보이고 훈련이 진행될수록 감소하였다. 따라서 위 정확도가 현재 모델의 구성과 파라미터에 따른 최선의 정확도를 나타낸 것을 확인했다. train을 5000장 정도 진행하였고 label당 사진의 수가 500장이므로 데이터셋을 더 늘려 학습하거나 데이터셋에 최적화된 증강 기법을 시도해본다면 더 높은 정확도를 얻어낼 수 있을 것이라 생각한다.

reference

머신러닝 수업 pdf와 과제 제안서

https://computistics.tistory.com/3 - Resnet 설명

https://pytorch.org/vision/stable/generated/torchvision.transforms.Resize.html -resize pytorch document