CNN Tutorial

Image Classification

목차

1. Data

CIFAR-10

10 개의 클래스

60,000 개의 32x32 컬러 이미지(3채널)

클래스 당 6,000 개의 이미지

50,000 개의 훈련 이미지와 10,000 개의 테스트 이미지

2. 개발환경

Version

python

- 3.6.12

pytorch

- 1.7.1+cu110

torchvision

- 0.8.2+cu110

3. 참고 문서 설명

Pytorch를 이용하여 이미지 분류

과정

1. torchvision을 사용하여 데이터셋을 불러오고, 정규화 (nomarlizing)

2. 합성곱 신경망(Convolution Neural Network)을 정의

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

출처: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

3. 참고 문서 설명

Pytorch를 이용하여 이미지 분류

과정

3. 손실 함수를 정의

4. 학습용 데이터를 사용하여 신경망을 학습(train)

5. 시험용 데이터를 사용하여 신경망을 검사(test)

손실함수: CrossEntropyLoss()

옵티마이저: SGD

lr = 0.001, momentum=0.9

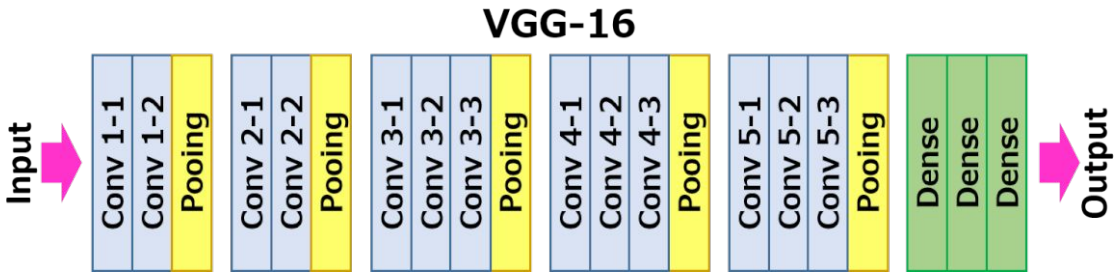
epochs 수: 10

정확도(accuracy): 64.19%

출처: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

4. 성능 향상을 위해 변경한 것

1. 신경망을 VGG16으로 변경



빠른 구현과 높은 성능을 위해서 VGG16 을 사용했다.

일반적인 VGG의 경우 input size가 224x224인 반면 CIFAR-10 data의 경우 32x32 이므로 Conv2d size 변경했다.

Accuracy가 64.19%에서 **74.83%**로 증가!

4. 성능 향상을 위해 변경한 것

2. Checkpoint 설정

Training 중간에 최적의 모델을 저장하기 위해서 체크포인트 사용했다.

torch.utils.data.dataset.random_split()을 이용해서 train set 과 validation set 분리(0.8 : 0.2)

```
dataset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                       download=True, transform=transform)
num = len(dataset)
trainset, valset = random_split(dataset, [int(num*(1-split_ratio)), int(num*split_ratio)])

trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                          shuffle=True, num_workers=2)

valloader = torch.utils.data.DataLoader(valset, batch_size=batch_size,
                                          shuffle=True, num_workers=2)
```

4. 성능 향상을 위해 변경한 것

2. Checkpoint 설정

validation set의 accuracy를 기준으로 모델 저장

```
if val_acc > best_val_acc:
    torch.save({
        'epoch': epoch,
        'model_state_dict': vgg16.state_dict(),
        'optimizer_state_dict': optimizer.state_dict(),
        'loss': loss.item()
        }, 'ckpt.pt')
    best_val_acc = val_acc
    print(f"Save new cp | val acc: {val_acc}, loss: {loss.item():.3f}")
```

4. 성능 향상을 위해 변경한 것

3. Learning rate 변경

batch_size = 16, epochs = 20 고정

나머지는 이전과 동일

lr = 0.001

Total Accuracy: 73.19%

epoch	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
loss	2.303	2.301	2.227	2.023	2.143	1.743	1.61	1.328	1.259	0.952	0.763	1.15	0.993	0.99	0.595	0.609	0.545	0.574	0.24	0.448

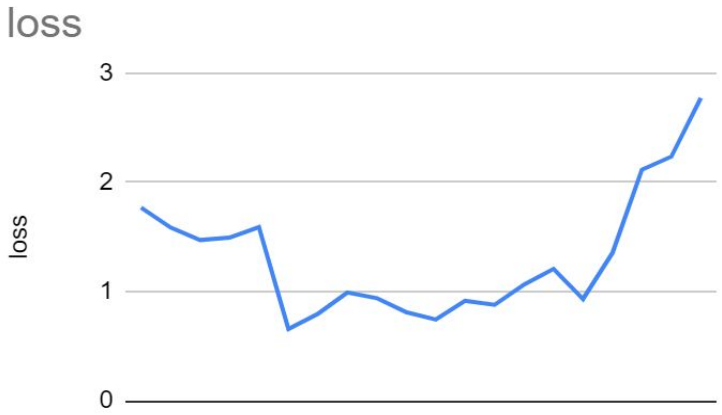
4. 성능 향상을 위해 변경한 것

3. Learning rate 변경

batch_size = 16, epochs = 20 고정

나머지는 이전과 동일

lr = 0.01



Total Accuracy: 70.99%

epoch	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
loss	1.767	1.585	1.471	1.493	1.589	0.656	0.795	0.991	0.939	0.81	0.743	0.914	0.879	1.064	1.207	0.931	1.352	2.114	2.234	2.77

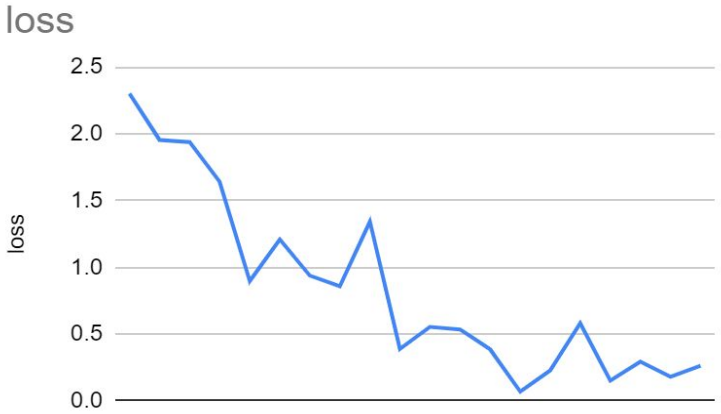
4. 성능 향상을 위해 변경한 것

3. Learning rate 변경

batch_size = 16, epochs = 20 고정

나머지는 이전과 동일

lr = 0.005



Total Accuracy: 76.82%

epoch	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
loss	2.301	1.953	1.938	1.642	0.897	1.209	0.938	0.858	1.342	0.389	0.554	0.535	0.387	0.07	0.228	0.581	0.152	0.294	0.181	0.264

4. 성능 향상을 위해 변경한 것

3. Learning rate 변경

Learning rate	0.01	0.001	0.005
Accuracy(%)	70.99	73.19	76.82

Learning rate가 0.005일때, 가장 높은 정확도 **76.82%**를 보여주었다.

4. 성능 향상을 위해 변경한 것

4. epoch 변경

더 많은 학습을 통해 성능을 향상 시키기 위해서 **epochs**를 증가 시켰다.

lr = 0.005						
epochs = 50	epoch	10	20	30	40	50
	loss	0.851	0.261	0.865	2.116	2.303

이전에 가장 높은 정확도를 보여준 **lr**(0.005)값을 이용해서
epochs를 20에서 50으로 증가시켰다.

어느정도 이상부터는 **loss**값이 증가되면서, 정확도가 감소했다.
(76.82% => 75.94%)

4. 성능 향상을 위해 변경한 것

4. epoch 변경

Learning rate를 0.005에서 0.001로 변경 후 epochs를 증가시켰다.

lr = 0.001						
epochs = 50	epoch	10	20	30	40	50
	loss	1.363	0.217	0.108	0.094	0.004

Total Accuracy: 76.9%

lr = 0.001						
epochs = 100	epoch	20	40	60	80	100
	loss	0.185	0.088	0.018	0.004	0.02

Total Accuracy: 78.4%

Learning rate가 0.001이고 epochs가 100일때, **78.4%**로 가장 높은 정확도를 보였다.

5. 결론

결과적으로 초기 형태에서

신경망을 VGG16으로 변경하고, epochs를 100으로 증가 시켰다.

그 결과

Accuracy가

64.19%에서 **78.4%** 로 증가했다.

더 크게 성능을 향상시키기 위해서는 모델을 변경해봐야 할 것 같다.

Ground Truth	cat	ship	ship	plane
Before	frog	ship	ship	plane
After (vgg16)	cat	ship	ship	plane

END