합성곱 신경망

부산대학교 정보컴퓨터공학부

15 학번 이연걸

1. 개요

딥러닝에서의 합성곱 신경망을 알아본다. 합성곱 신경망은 이미지 처리에 알맞게 다층 퍼셉트론을 변형시킨 신경망으로 흔히 CNN 으로 줄여 부른다.

합성곱 신경망에는 합성곱 계층과 풀링 계층이 새로운 유형의 은닉 계층으로 이용되는데, 합성곱 계층은 다층 퍼셉트론의 완전 연결 계층에 비해 획기적으로 줄어든 수의 파라미터만을 가지며 풀링 계층은 아예 파라미터를 갖지 않는다. 합성곱 신경망은 합성곱 계층이 갖는 소수의 파라미터를 집중적으로 학습시키는 방법으로 이미지 처리 분야에서의 학습 품질을 크게 향상시켰다.

좀 더 자세히 알아보면, 합성곱 신경망에서는 합성곱 계층에 있는 커널이라는 작은 가중치 텐서를 이미지의 모든 영역에 반복 적용해 패턴을 찾아 처리한다. 또한 풀링 계층은 처리할 이미지 해상도를 줄이는 기능을 제공해 다양한 크기의 패턴을 단계적으로 처리할 수 있게 한다.

이번 실험의 데이터셋은 5 장에서 이용된 꽃 이미지를 활용한다. 우선, 합성곱 신경망을 사용하였을 때와 그렇지 않은 경우를 비교한다. 또한, 합성곱 신경망에서 합성곱 계층과 풀링 계층 등을 수정하면서 정확도 차이를 비교하고 활성화 함수 등에도 변화를 준다. 그리고 하이퍼 파라미터에 대한 차이도 분석해본다.

2. 구현 과정

2.1 다층퍼셉트론 vs 합성곱 신경망

다층퍼셉트론을 사용하였을 때에 비해 합성곱신경망을 사용한 경우 정확도 향상이 어느정도 이루어지는지 살펴본다.

```
fm1 = CnnBasicModel('flowers_model_1', fd,
                    [['full', {'width':30}],
                     ['full', {'width':10}]])
fm1.use_adam = False
fm1.exec_all(epoch_count = 10, report = 2)
```

```
Model flowers_model_1 train started:
    Epoch 2: cost=1.597, accuracy=0.250/0.220 (14/14 secs)
    Epoch 4: cost=1.514, accuracy=0.300/0.270 (15/29 secs)
    Epoch 6: cost=1.361, accuracy=0.389/0.400 (13/42 secs)
    Epoch 8: cost=1.297, accuracy=0.413/0.440 (14/56 secs)
    Epoch 10: cost=1.253, accuracy=0.440/0.400 (13/69 secs)
Model flowers_model_1 train ended in 69 secs:
Model flowers_model_1 test report: accuracy = 0.399, (0 secs)

Model flowers_model_1 Visualization
```

```
추정확률분포 [44,35,13, 0, 7] => 추정 daisy : 정답 daisy => O
추정확률분포 [27,28,17, 9,17] => 추정 dandelion : 정답 dandelion => O
추정확률분포 [29,29,21, 5,17] => 추정 dandelion : 정답 daisy => X
```

그림 1 – 다층퍼셉트론을 사용한 경우

```
fm2 = CnnBasicModel('flowers_model_2', fd,
                    [['conv', {'ksize':5, 'chn':6}],
                     ['max', {'stride':4}],
                     ['conv', {'ksize':3, 'chn':12}],
                     ['avg', {'stride':2}]],
                    True)
fm2.exec_all(epoch_count = 10, report = 2)
```

```
Model flowers_model_2 train started:
    Epoch 2: cost=1.108, accuracy=0.563/0.520 (375/375 secs)
    Epoch 4: cost=0.946, accuracy=0.638/0.650 (362/737 secs)
    Epoch 6: cost=0.767, accuracy=0.709/0.560 (357/1094 secs)
    Epoch 8: cost=0.617, accuracy=0.768/0.530 (353/1447 secs)
    Epoch 10: cost=0.530, accuracy=0.803/0.590 (375/1822 secs)
Model flowers_model_2 train ended in 1822 secs:
Model flowers_model_2 test report: accuracy = 0.549, (41 secs)

Model flowers_model_2 Visualization
```

그림 2 – 합성곱 신경망을 사용한 경우

2.2 합성곱 신경망의 은닉 계층 수에 따른 변화

합성곱신경망에서 은닉 계층 수를 증가시켰을 때, 정확도 향상이 어느정도 이루어지는지 살펴본다.

```
fm3 = CnnBasicModel('flowers_model_3', fd,
                [['conv', {'ksize':3, 'chn':6}],
                 ['max', {'stride':2}],
                 ['conv', {'ksize':3, 'chn':12}],
                 ['max', {'stride':2}],
                 ['conv', {'ksize':3, 'chn':24}],
                 ['avg', {'stride':3}]])
fm3.exec_all(epoch_count = 10, report = 2)
```

```
Model flowers_model_3 train started:
    Epoch 2: cost=1.121, accuracy=0.552/0.600 (268/268 secs)
    Epoch 4: cost=0.961, accuracy=0.635/0.600 (260/528 secs)
    Epoch 6: cost=0.814, accuracy=0.689/0.630 (266/794 secs)
    Epoch 8: cost=0.694, accuracy=0.734/0.680 (280/1074 secs)
    Epoch 10: cost=0.600, accuracy=0.776/0.680 (283/1357 secs)
Model flowers_model_3 train ended in 1357 secs:
Model flowers_model_3 test report: accuracy = 0.606, (7 secs)

Model flowers_model_3 Visualization
```

```
추정확률분포 [98, 0, 1, 0, 0] => 추정 daisy : 정답 daisy => O
추정확률분포 [74, 3, 6, 5,12] => 추정 daisy : 정답 dandelion => X
추정확률분포 [74, 1, 0,16, 9] => 추정 daisy : 정답 sunflower => X
```

그림 3 – 그림 2 에서 합성곱 계층과 풀링 계층 하나씩 추가한 경우

2.3 활성화 함수에 따른 변화

기존 ReLU 함수가 아닌, 시그모이드와 쌍곡탄젠트 함수를 사용한 경우를 비교해본다.

```
fm4 = CnnBasicModel('flowers_model_4', fd,
                [['conv', {'ksize':3, 'chn':6, 'actfunc': 'sigmoid'}],
                 ['max', {'stride':2}],
                 ['conv', {'ksize':3, 'chn':12, 'actfunc': 'sigmoid'}],
                 ['max', {'stride':2}],
                 ['conv', {'ksize':3, 'chn':24, 'actfunc': 'sigmoid'}],
                 ['avg', {'stride':3}]])
fm4.exec_all(epoch_count = 10, report = 2)
```

```
Model flowers_model_4 train started:
    Epoch 2: cost=1.599, accuracy=0.244/0.200 (326/326 secs)
    Epoch 4: cost=1.483, accuracy=0.355/0.380 (320/646 secs)
    Epoch 6: cost=1.375, accuracy=0.391/0.430 (337/983 secs)
    Epoch 8: cost=1.277, accuracy=0.446/0.540 (327/1310 secs)
    Epoch 10: cost=1.194, accuracy=0.490/0.490 (347/1657 secs)
Model flowers_model_4 train ended in 1657 secs:
Model flowers_model_4 test report: accuracy = 0.479, (15 secs)

Model flowers_model_4 Visualization
```

```
추정확률분포 [10,44, 3,31,11] => 추정 dandelion : 정답 dandelion => O
추정확률분포 [24,54, 7, 5,10] => 추정 dandelion : 정답 daisy => X
추정확률분포 [ 2, 1,59, 2,35] => 추정 rose : 정답 tulip => X
```

그림 4 – 활성화 함수로 시그모이드 함수 사용한 경우

```
fm5 = CnnBasicModel('flowers_model_5', fd,
                [['conv', {'ksize':3, 'chn':6, 'actfunc': 'tanh'}],
                 ['max', {'stride':2}],
                 ['conv', {'ksize':3, 'chn':12, 'actfunc': 'tanh'}],
                 ['max', {'stride':2}],
                 ['conv', {'ksize':3, 'chn':24, 'actfunc': 'tanh'}],
                 ['avg', {'stride':3}]])
fm5.exec_all(epoch_count = 10, report = 2)
```

```
Model flowers_model_5 train started:
    Epoch 2: cost=1.235, accuracy=0.488/0.630 (346/346 secs)
    Epoch 4: cost=1.165, accuracy=0.512/0.590 (343/689 secs)
    Epoch 6: cost=1.105, accuracy=0.551/0.540 (323/1012 secs)
    Epoch 8: cost=1.066, accuracy=0.582/0.560 (261/1273 secs)
    Epoch 10: cost=1.012, accuracy=0.607/0.540 (262/1535 secs)
Model flowers_model_5 train ended in 1535 secs:
Model flowers_model_5 test report: accuracy = 0.498, (10 secs)

Model flowers_model_5 Visualization
```

```
추정확률분포 [ 0, 0,88, 0,11] => 추정 rose : 정답 rose => O
추정확률분포 [ 1, 0,12, 2,84] => 추정 tulip : 정답 tulip => O
추정확률분포 [17,10,23,21,30] => 추정 tulip : 정답 tulip => O
```

그림 5 - 활성화 함수로 쌍곡탄젠트 함수 사용한 경우

2.4 하이퍼 파라미터에 따른 변화

하이퍼 파라미터를 바꾸어가면서 비교해본다.

```
fm6 = CnnBasicModel('flowers_model_6', fd,
                [['conv', {'ksize':3, 'chn':6}],
                 ['max', {'stride':2}],
                 ['conv', {'ksize':3, 'chn':12}],
                 ['max', {'stride':2}],
                 ['conv', {'ksize':3, 'chn':24}],
                 ['avg', {'stride':3}]])
fm6.exec_all(epoch_count = 10, report = 2, learning_rate=0.0001)
```

```
Model flowers_model_6 train started:
    Epoch 2: cost=1.167, accuracy=0.518/0.560 (206/206 secs)
    Epoch 4: cost=1.072, accuracy=0.575/0.490 (233/439 secs)
    Epoch 6: cost=1.006, accuracy=0.611/0.600 (234/673 secs)
    Epoch 8: cost=0.960, accuracy=0.619/0.620 (216/889 secs)
    Epoch 10: cost=0.922, accuracy=0.655/0.610 (221/1110 secs)
Model flowers_model_6 train ended in 1110 secs:
Model flowers_model_6 test report: accuracy = 0.559, (6 secs)

Model flowers_model_6 Visualization
```

```
추정확률분포 [ 0, 0,31, 0,68] => 추정 tulip : 정답 rose => X
추정확률분포 [ 1, 3, 0,89, 7] => 추정 sunflower : 정답 sunflower => O
추정확률분포 [29,49, 5, 3,14] => 추정 dandelion : 정답 dandelion => O
```

그림 6 – 학습률이 0.0001 인 경우

```
fm6 = CnnBasicModel('flowers_model_6', fd,
                [['conv', {'ksize':3, 'chn':6}],
                 ['max', {'stride':2}],
                 ['conv', {'ksize':3, 'chn':12}],
                 ['max', {'stride':2}],
                 ['conv', {'ksize':3, 'chn':24}],
                 ['avg', {'stride':3}]])
fm6.exec_all(epoch_count = 10, report = 2, batch_size=5)
```

```
Model flowers_model_6 train started:
    Epoch 2: cost=1.077, accuracy=0.577/0.660 (312/312 secs)
    Epoch 4: cost=0.878, accuracy=0.663/0.550 (284/596 secs)
    Epoch 6: cost=0.734, accuracy=0.722/0.680 (277/873 secs)
    Epoch 8: cost=0.625, accuracy=0.770/0.590 (277/1150 secs)
    Epoch 10: cost=0.501, accuracy=0.817/0.670 (293/1443 secs)
Model flowers_model_6 train ended in 1443 secs:
Model flowers_model_6 test report: accuracy = 0.645, (6 secs)

Model flowers_model_6 Visualization
```

```
추정확률분포 [ 3, 8, 3,86, 0] => 추정 sunflower : 정답 sunflower => O
추정확률분포 [86, 4, 4, 2, 4] => 추정 daisy : 정답 daisy => O
추정확률분포 [ 7,92, 0, 0, 0] => 추정 dandelion : 정답 dandelion => O
```

그림 7 – 미니배치 크기가 5 인 경우

3. 총평

2 에서 진행한 구현 결과를 정리하면 다음과 같다.

표 – 1 다층 퍼셉트론 vs 합성곱 신경망

	그림 1 (다층 퍼셉트론)	그림 2 (합성곱 신경망)
은닉 계층	완전 연결 계층	합성곱, 풀링계층
학습 횟수	10	10
정확도	39.9%	54.9%

합성곱 신경망을 사용한 경우 다층 퍼셉트론을 사용하였을 때보다 15%에 해당하는 정확도 향상을 보였다. 확실히 이미지 처리에 있어서 합성곱 신경망이 도움이 되는 모습이다. 다만, 파라미터 수가 많아짐에 따라 걸리는 시간이 오래걸렸다.

표 – 2 합성곱 신경망의 계층 변화

	그림 2	그림 3
은닉 계층 구성	conv – max – conv - avg	conv – max – conv – max – conv - avg
합성곱 커널크기	[5, 5], [3, 3]	[3, 3], [3, 3], [3, 3]
합성곱 채널수	6, 12	6, 12, 24
풀링 계층 보폭크기	[4, 4], [2, 2]	[2, 2], [2, 2], [3, 3]
정확도	54.9%	60.6%

합성곱 신경망에서 합성곱 계층과 풀링계층을 하나씩 추가시켜서 진행해보았다. 약 6%의 정확도 향상을 얻어낼 수 있었다. 학습 횟수가 10 임에도 불구하고 손실값이 계속 줄어들고 정확도 또한 지속적으로 상승하므로 학습 횟수를 늘리고 하이퍼 파라미터를 수정하면 더욱 높은 정확도를 얻을 것으로 예상된다.

표 – 3 활성화 함수에 따른 변화

	그림 4	그림 5
활성화 함수	시그모이드	쌍곡 탄젠트
정확도	47.9% (-12.7%)	49.8% (-10.8%)

시그모이드와 쌍곡 탄젠트 함수 모두 정확도에서 낮은 결과가 나왔다. 따라서 합성곱 신경망에서는 활성화 함수에서 ReLU 함수를 시그모이드와 쌍곡 탄젠트함수로 대체하기는 어려워보인다.

표 – 4 하이퍼 파라미터에 따른 변화

	그림 6 (학습률)	그림 7 (미니배치 크기)
변화 수치	0.001 -> 0.0001	10 -> 5
정확도	55.9% (-4.7%)	64.5% (+3.9%)

학습률을 낮추었을 때 오히려 낮은 정확도가 나왔다. 미니배치 크기를 줄였을 때는 더 높은 정확도를 얻을 수 있었고, 손실값도 주기적으로 낮아지는 좋은 경향을 보였다.