8장. 다섯 가지 정규화 확장 : 꽃 이미지 분류 신경망

부산대학교 항공우주공학과

정대현 201527137

1. 개요

학습을 수행할 때 의도적으로 제약을 가해 학습을 방해하는 기법을 사용합니다. 이런 제약을 가하는 이유는 학습모델이 지나치게 학습데이터를 따르는 과적합 현상을 방지하기 위해서입니다. 정규화는 학습을 방해하는 기법을 통칭하는 말이며 이번 장에서는 부적합과 과적합의 의미와 이를 해결하기 위한 다섯 가지 정규화 기법의 원리는 무엇이고 구현 방법, 동작 방식을 학습하게 됩니다.

2. 과적합 Overfitting 문제

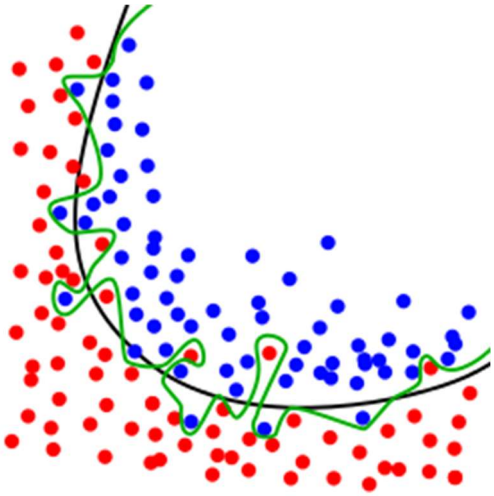


Fig 1. 초록색 선은 과적합 모델, 검은색 선은 일반 모델

과적합은 전체적인 문제나 성질의 특성을 파악하지 못한 채 학습데이터 자체의 지엽적인 특성을 외워버리는 현상을 뜻합니다. 즉 다시 말해 학습데이터를 과하게 학습(overfitting)합니다. 일반적으로 학습 데이터는 실제 데이터의 부분 집합으로 이루어집니다. 과적합이 발생하면 학습데이터에 대해서는 오차가 감소하지만 실제 데이터에 대해서는 오차가 증가하게 됩니다. 실제 모든 데이터를 수집해서 학습하는 것을 물리적, 시간적으로 불가능한 일입니다. 학습 데이터만 가지고 실제 데이터의 오차가 증가하는 지점을 예측하는 것은 매우 어렵거나 불가능합니다.

3. 학습모델 정규화

학습을 진행하다 보면 가중치 파라미터 절대값 중 일부 혹은 전부가 과도하게 커지는 방향으로 치닫을 때가 있다. 이때 L2 손실과 L1 손실은 절댓값이 큰 파라미터에 대해 불이익을 주어 값의 폭주를 막고 기왕이면 작은 절댓값의 파라미터들로 문제를 풀도록 압박하는 정규화 기법이다.

3.1L2 손실

L2손실 기법은 L2 페널티 값(노름)을 손실 함수에 더해주는 정규화 기법이다. L2 노름은 아래와 같이 정의한다.

$$||w_2|| = \sqrt{\sum_{i=1}^{n} |w_i|^2}$$

식 1. L2 노름

L1 규제와 동일하게 손실함수에 L2 규제를 적용했다. 알파는 L1 규제와 마찬가지로 규제의 양을 조절하기 위한 매개변수이고, 1/2 은 미분 결과를 보기 좋게 하기 위해 추가한다. 마지막으로 손실함수를 미분한 결과를 가중치 업데이트 식에 적용한다.

$$w = w - \eta \frac{\partial L}{\partial w} = w + \eta((y-a)x - \alpha \times w)$$

식 3. L2 규제를 적용한 가중치 업데이트

L2 손실이 적용된다고 모든 파라미터의 절댓값이 줄어드는 것은 아니다. L2 손실이 파라미터값에 영향을 주면 기존의 신경망 학습 과정에 변화가 일어난다. L2손실로 인하여 파라미터 값이 문제 풀이에 필요한 값보다 줄어들면 기존의 학습 과정은 과도하게 줄어 든 파라미터값을 다시 키우는 방향으로 손실 기울기를 만든다. 즉 L2손실로 인한 부담을 상쇄하는 방향으로 학습 내용에 변화가 생긴다. 기존의 가중치의 크기가 직접 영향을 미치기 때문에 L2 규제가 L1 규제 보다 가중치 규제에 좀 더 효과적이라고 한다. L2 규제가 L1 규제보다는 널리 쓰이는 규제 방식이다.

3.2 L1 손실

L1 손실 기법은 L1 페널티값을 손실함수에 더해주는 정규화 기법이다.

$$\|w\|_1 = \sum_{i=1}^{n} |w_i|$$

식 4. L1 노름

$$w = w - \eta \frac{\partial L}{\partial w} = w + \eta((y-a)x - \alpha \times sign(w))$$

L1

식 5. L1 규제를 적용한 가중치 업데이트

L2 손실과 마찬 가지로 역전파의 전체적인 처리과정은 손댈 필요가 없으며 단지 각각 파라미터 가중치 값이 수정될 때 L1손실을 고려하지 않고 계산된 기존의 손실 기울기에 $\alpha sign(w_i)$값만 저해서 처리하면 된다. L1 손실 역시 L2손실과 비슷하게 과적합 억제에 어느 정도의 효과는 있어 보이지만 그보다도 가중치 분포에 미치는 영향이 크다.

3.3 드롭아웃(Dropout)

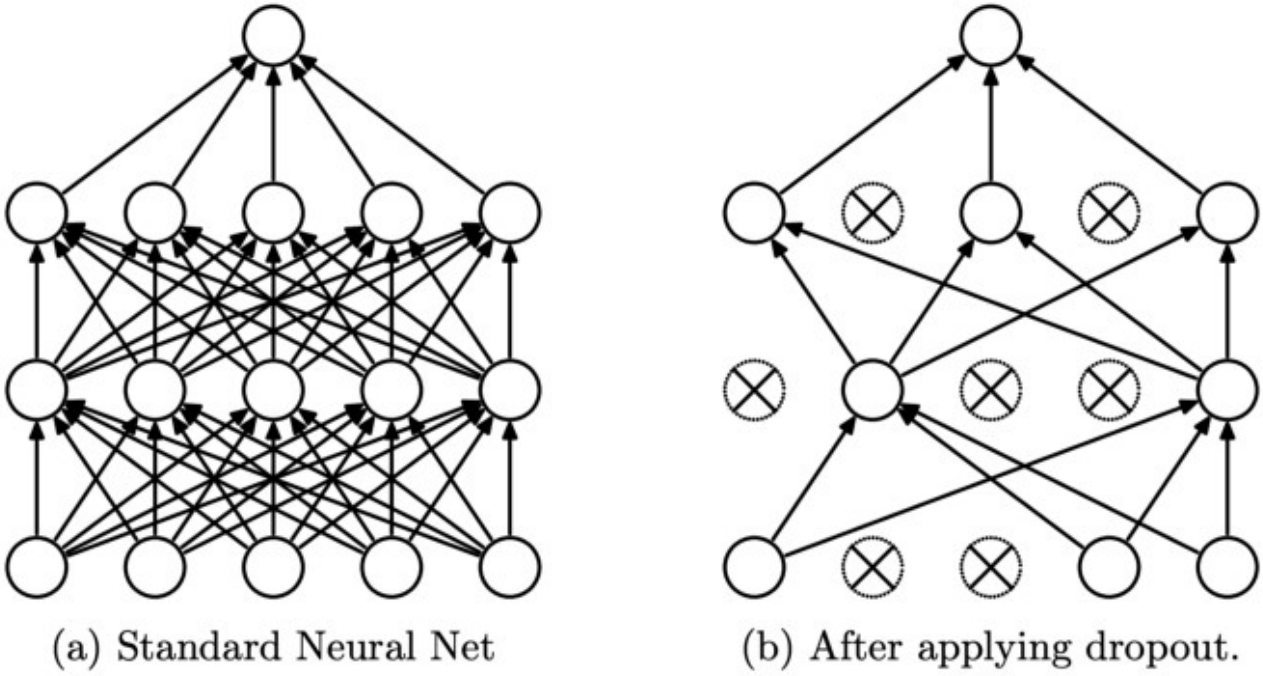


Fig 2. 드롭아웃 적용 전후 차이

드롭아웃은 입력 또는 어떤 계층의 출력을 다음 계층에서 모두 이용하지 않고 일부만 이용하면서 신경망을 학습하는 규제화 기법입니다. 규제화 기법 가운데 매우 간단하면서도 효과가 커서 널리 이용됩니다. 드롭아웃을 통해 배제되는 노드들은 일정한 규칙이 아닌 난수 함수를 이용해 랜덤으로 결정하며 노드 배치 분포는 미니배치 데이터를 처리할 때마다 달라질 뿐 아니라 같은 미니 배치 안에서도 달라집니다. 드롭아웃은 과적합 방지를 위해 도입되는 정규화 기법이지 계산량을 줄이기 위한 기법이 아닙니다. 학습이 끝난 과정에 더해 난수 함수를 이용한 드롭아웃 대상 선정과 선정된 대상에 대한 순전파와 역전파 처리에서의 마스킹 처리를 수행해야 하기 때문에 계산량은 오히려 늘어납니다. 정규화 기법이 학습 속도를 늦추는 점까지 고려하면 계산량이 많이 늘어나게 됩니다.

3.4 잡음주입

잡은 주입 기법은 두 계층 사이에 잡은 주입 계층을 삽입하여 이 계층이 아래 계층의 출력에 적당한 형태의 잡음을 추가하여 위 계층에 전달하는 정규화 기법입니다. 잡음이 주입되면 학습은 혼란을 겪게 되지만 결과적으로 오차와 잡음에 강한 학습이 이루어집니다. 입력의 변이에 대한 강건한 학습을 가능하게 하여 평가 단계에서의 품질을 높여줍니다 또한 매번 다른 잡음이 적용되어 다양하게 변형된 입력이 제공되면서 같은 처리가 이루어 지지 않게 하여 모델의 과적합을 방지하는 효과도 얻을 수 있습니다.

3.5 배치 정규화(batch normalization)

배치 정규화는 미니배치 내의 데이터들에 대해 벡터 성분별로 정규화를 수행하는 방식이다. 정규화는 대상 값들에 동일한 선형 변환을 가해 평균 0, 표준편차 1의 분포로 만들어주는 처리를 말한다.

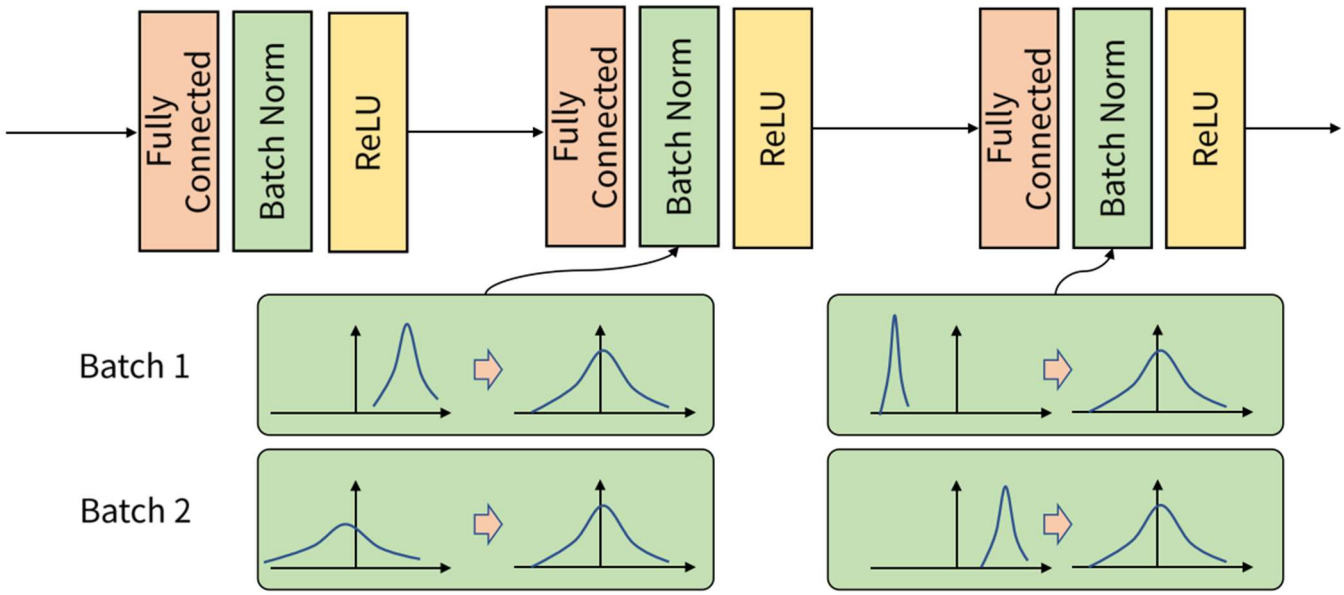


Fig 3. 배치 정규화 처리

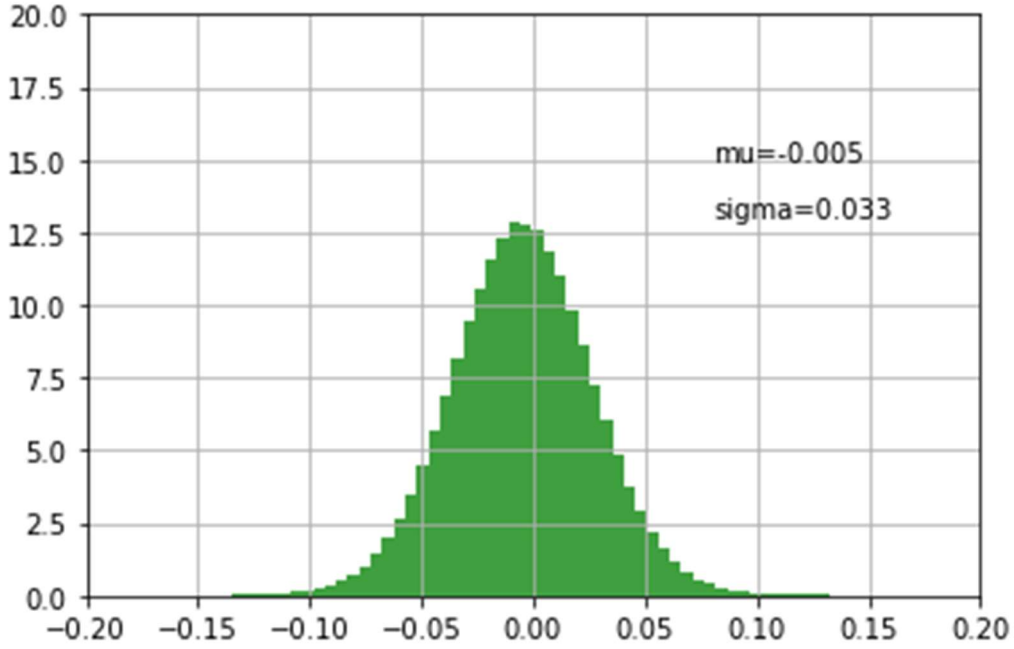
네트워크를 학습하다 보면 역전파 처리 중에 층을 거듭할수록 파라미터 성분에 따라 Cost 기울기가 급격히 소멸하거나 폭주하면서 학습이 제대로 수행하지 못하는 경우가 발생합니다. 이는 네트워크의 각 층의 입력을 구성하는 성분 별 분포가 심하게 달라 특성 입력 성분이 가중치 파라미터의 비용 기울기를 좌지우지할 때 쉽게 일어납니다. 이에 따라 여러 입력 성분 간에 적절한 균형을 잡아 주는 방법이 필요하게 됩니다.

4. 실험

이번 장의 실험은 단순히 정규화를 통해 정확도의 증가를 살펴보기보다 어떻게 작동하는지 알아보기 위함입니다. 기본 구조에 L1, L2 손실 효과를 적용 하였을 때 변화하는 파라미터 분포와 드롭아웃, 잡음주입, 배치 정규화의 작동을 살펴봅니다.

4.1 Reference 5장 은닉계층 3개 폭 [50, 25, 10] epoch 10회

```
Epoch 2: cost=1.499, accuracy=0.299/0.320 (34/34 secs)
    Epoch 4: cost=1.450, accuracy=0.328/0.290 (34/68 secs)
    Epoch 6: cost=1.404, accuracy=0.392/0.390 (35/103 secs)
    Epoch 8: cost=1.368, accuracy=0.387/0.340 (34/137 secs)
    Epoch 10: cost=1.314, accuracy=0.424/0.320 (37/174 secs)
Model flowers_model_1 train ended in 174 secs:
Model flowers_model_1 test report: accuracy = 0.397, (0 secs)
추정확률분포 [31,30,21, 1,16] => 추정 daisy : 정답 dandelion => X
추정확률분포 [16,18,26, 9,31] => 추정 tulip : 정답 daisy => X
추정확률분포 [38,34,17, 0,10] => 추정 daisy : 정답 dandelion => X
```

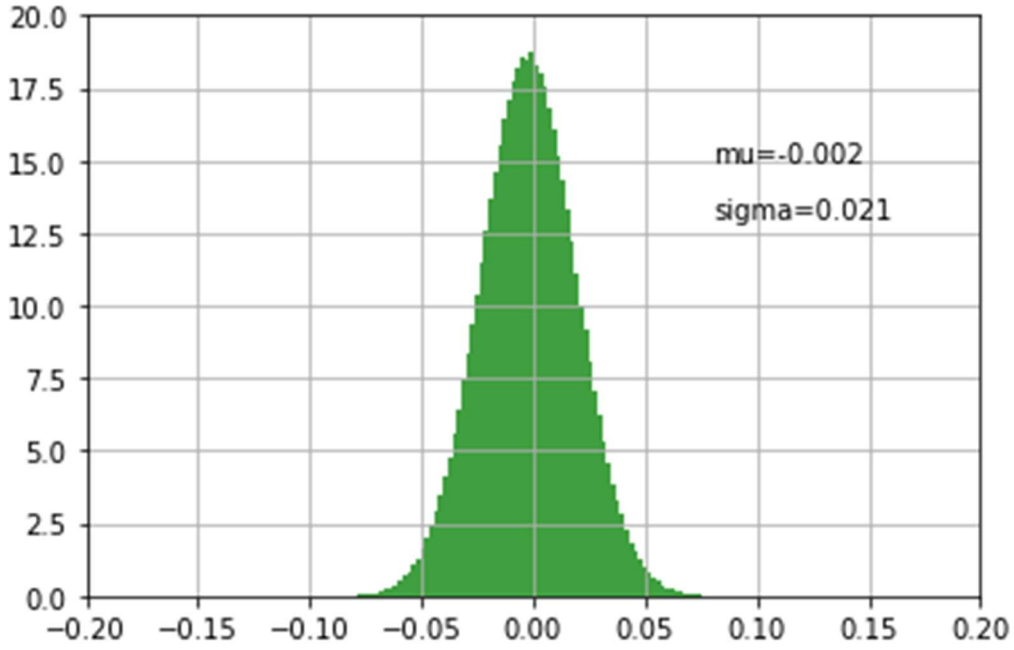


Near 0 parameters = 0.0%(337/1383950)

4.2 은닉계층 3개 폭 [50, 25, 10] epoch 10회 + L2 손실추가

Model flowers_model_2 train ended in 804 secs:

Model flowers_model_2 test report: accuracy = 0.232, (0 secs)

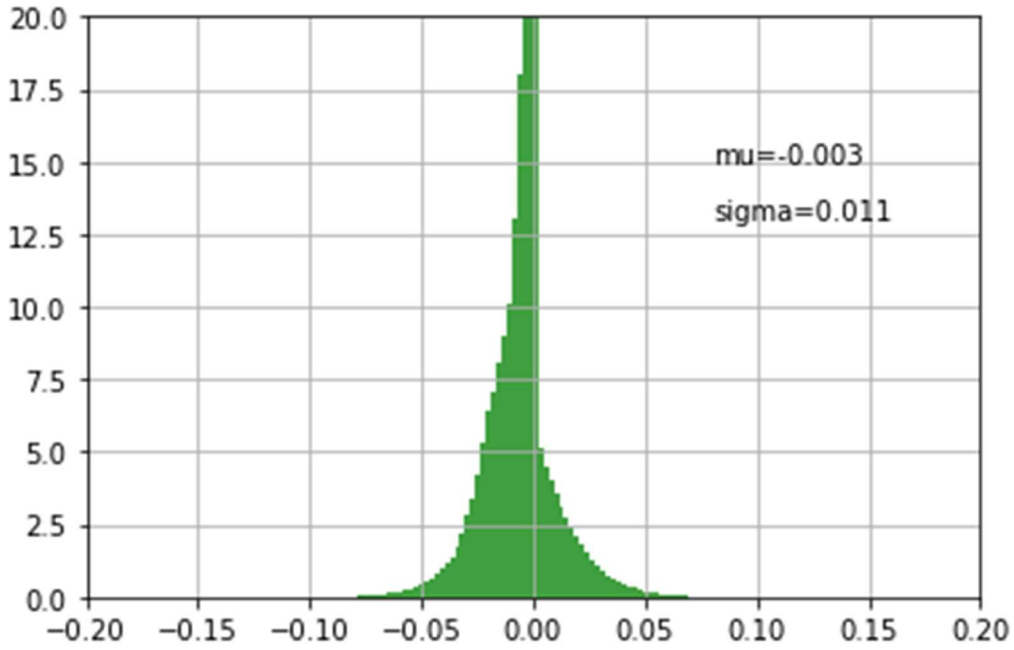


Near 0 parameters = 0.0%(518/1383950)

4.3 은닉계층 3개 폭 [50, 25, 10] epoch 10회 + L1 손실추가

Model flowers_model_3 train ended in 840 secs:

Model flowers_model_3 test report: accuracy = 0.232, (0 secs)



Near 0 parameters = 53.8%(744015/1383950)

4.4 size 3, ch 6, max stride 2 + size 3, ch 12, max stride 2

+ size 3, ch 24, avg stride 2 epoch 20회 [CNN적용]

Epoch 2: cost=1.131, accuracy=0.543/0.550 (100/100 secs)

Epoch 6: cost=0.828, accuracy=0.691/0.600 (129/340 secs)

Epoch 10: cost=0.597, accuracy=0.779/0.690 (122/582 secs)

Epoch 14: cost=0.384, accuracy=0.858/0.660 (130/839 secs)

Epoch 18: cost=0.236, accuracy=0.921/0.700 (132/1100 secs)

Model flowers_cnn_1 train ended in 1237 secs:

Model flowers_cnn_1 test report: accuracy = 0.638, (3 secs)

추정확률분포 [49,51, 0, 0, 0] => 추정 dandelion : 정답 daisy => X

추정확률분포 [78, 7,13, 0, 2] => 추정 daisy : 정답 daisy => O

추정확률분포 [0,24, 0,76, 0] => 추정 sunflower : 정답 tulip => X

4.5 size 3, ch 6, max stride 2 + size 3, ch 12, max stride 2

+ size 3, ch 24, avg stride 2 epoch 50회 드롭아웃 적용 [CNN적용]

```
Model flowers_cnn_2 train started:
    Epoch 10: cost=0.908, accuracy=0.655/0.460 (1179/1179 secs)
    Epoch 20: cost=0.809, accuracy=0.686/0.440 (618/1797 secs)
    Epoch 30: cost=0.781, accuracy=0.695/0.560 (744/2541 secs)
    Epoch 40: cost=0.747, accuracy=0.713/0.460 (729/3270 secs)
    Epoch 50: cost=0.732, accuracy=0.715/0.560 (697/3967 secs)
Model flowers_cnn_2 train ended in 3967 secs:
Model flowers_cnn_2 test report: accuracy = 0.561, (4 secs)
```

4.6 size 3, ch 6, max stride 2 + size 3, ch 12, max stride 2

+ size 3, ch 24, avg stride 2 epoch 20회 잡음주입 적용 [CNN적용]

```
Model flowers_cnn_3 train started:
    Epoch 10: cost=0.498, accuracy=0.817/0.710 (741/741 secs)
    Epoch 20: cost=0.162, accuracy=0.942/0.740 (716/1457 secs)
    Epoch 30: cost=0.067, accuracy=0.975/0.680 (671/2128 secs)
    Epoch 40: cost=0.125, accuracy=0.961/0.580 (734/2862 secs)
    Epoch 50: cost=0.113, accuracy=0.966/0.580 (715/3577 secs)
Model flowers_cnn_3 train ended in 3577 secs:
Model flowers_cnn_3 test report: accuracy = 0.581, (3 secs)
```

4.7 size 3, ch 6, max stride 2 + size 3, ch 12, max stride 2

+ size 3, ch 24, avg stride 2 epoch 20회 배치 정규화 적용 [CNN적용]

```
Model flowers_cnn_4 train started:
    Epoch 10: cost=0.621, accuracy=0.765/0.530 (705/705 secs)
    Epoch 20: cost=0.363, accuracy=0.865/0.710 (760/1465 secs)
    Epoch 30: cost=0.272, accuracy=0.899/0.660 (1104/2569 secs)
    Epoch 40: cost=0.148, accuracy=0.954/0.650 (3577/6146 secs)
    Epoch 50: cost=0.114, accuracy=0.960/0.630 (659/6805 secs)
Model flowers_cnn_4 train ended in 6805 secs:
Model flowers_cnn_4 test report: accuracy = 0.661, (3 secs)
```

실험	0에 근접한 값의 수	평균	표준편차
기본	337/1383950	-0.005	0.033
L2소실	518/1383950	-0.002	0.021
L1소실	744015/1383950	-0.003	0.011

표1. L2손실, L1손실 적용 비교

실험	Epoch	시간	정확도
기본	**20**	1237s	0.638
드롭아웃적용	50	3967s	0.561
잡음주입	50	3577s	0.581
미니배치정규화	50	6805s	0.661

표2. 드롭아웃, 잡음주입, 배치 정규화 적용 비교

5. 결론

이번 실험에서 중요한 것은 학습률이 올라간 것 보다 어떻게 학습이 진행했는가? 이다. 지난 실험에서 가장 학습이 좋았던 은닉계층 [50, 25, 10]을 사용했다. 베이스라인이 되는 4.1은 함수의 파라미터 분포 평균은 -0.005 표준편차 0.033이었다. 그리고 83만개에 138만개정도 되는 파라미터 가운데 0에 가까운 파라미터는 337개였다. 여기서 4.2는 L2손실을 추가하였다. 평균은 -0.002로 감소하였으며 표준편차 역시 0.021로 줄었다. 0에 가까운 파라미터는 518개로 소폭 늘어났으나 그다지 많은 수는 아니다. 이번에는 L1을 적용하였다. 히스토그램모양에서 좀더 찌그러진 형태를 보이여 평균은 -0.003 표준편차는 0.011로 이전보다 표준편차는 조금 더 줄었다. 그리고 0부분의 확률밀도가 매우 커졌으며 전체 파라미터의 53.8%가 0에 가까운 값을 가지며 확실히 효과가 있는 것을 확인할 수 있다.

그 다음으로 지난 7장실험에서 가장 성능이 잘 나온 세팅을 베이스라인으로 두고 드롭아웃, 잡음주입, 미니배치정규화등을 적용한 다음 얼마나 달라지는지 실험해보았다. 풀링 계층의 처리가 끝날때마다 60%만 살리고 나머지 40%를 0으로 만들어버리는 드롭아웃 계층을 삽입했을 때 베이스라인보다 정확도는 감소하였다. 학습과정에 오히려 불리한 부담을 주었기에 학습 성과가 떨어졌지만 Epoch반복횟수를 2.5배 늘리는 정도로는 정확도를 개선할 수 없었다. 잡음주입의 경우 입력에 평균 0, 표준편차 0.01의 잡음을 추가하여 학습을 교란시켰다. 학습은 문제없이 진행되었으며 베이스라인보다 약간 떨어진 0.581로 구해졌다. 주입되는 평균과 표준편차 역시 중요한 하이퍼파라미터로 학습에 영향을 끼칠 수 있다. 배치 정규화는 합성곱 계층 앞에 배치 정규화 계층을 삽입하여 정규화된 입력이 합성곱 계층에 제공되도록 해보았다. 일종의 잡음 주입과 비슷한 효과를 내면서 학습을 교란시키는 배치정규화 계층의 도입으로 정확도는 0.661로 베이스라인보다 소폭 상승했다. 기본적으로 정규화방법은 학습을 방해하는 것이기 때문에 Epoch를 똑같이 한 상태에서는 나은 결과를 얻었기가 힘들며 실험에서는 2.5배 많이 학습했음에도 불구하고 정확도가 떨어졌다. 정규화방법을 도입했을 때 결과물을 더 좋게 만들려면 기존보다 더 많은 학습횟수와 모델 데이터가 요구되는 것을 확인 할 수 있다.