

## 8장. 다섯 가지 정규화 확장 : 꽃 이미지 분류 신경망

### 1. 개요

다섯 가지 정규화 확장 실험에서는 이전 실험에서 사용했었던 꽃 이미지와 오피스31 데이터셋에 대해 다양한 CNN구조와 정규화 기법을 실행하고 효과적인 모델 구성을 찾아보는 작업을 진행하였다. 정규화란 신경망이 학습 데이터에 지나치게 맞추어지는 과적합 방지를 위해 일부러 적당한 제약을 가해 학습을 방해하는 기법을 말한다. 다섯 가지 대표적인 정규화에는 L2 손실, L1손실, 드롭아웃, 잡음 주입, 배치 정규화가 있으며, 이들은 정규화 기법 도입을 위한 계층의 추가 방식에 차이가 있다. L2 손실과 L1손실은 기존의 처리 과정을 변형하여 계층을 추가하지만, 드롭아웃, 잡음주입, 배치정규화는 독립된 계층 형태로 지원한다는 특징을 가지고 있다. 이번 실험에서는 --

### 2. 학습 결과

#### 2.1. L1손실과 L2손실을 사용한 경우

##### (1) 정규화를 적용하지 않은 경우

구분	내용
조건	- 은닉 계층 : [30,10] - 에포크 횟수 : 10 - 정규화 기법 : X
데이터 정확도	31.1%
학습 시간	129s
0근처 파라미터 개수	192 / 829790 (0.0%)
파라미터 평균	-0.005
파라미터 분산	0.032

## 결과 화면

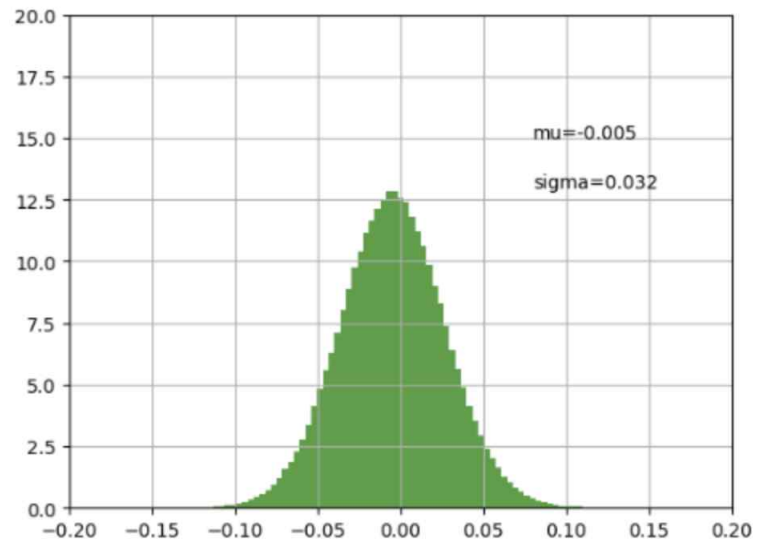
```
fml = CnnRegModel('flowers_model_1', fd, [30,10])
fml.exec_all(epoch_count=10, report=2, show_params=True)
```

Model flowers\_model\_1 train started:  
 Epoch 2: cost=1.460, accuracy=0.320/0.360 (26/26 secs)  
 Epoch 4: cost=1.467, accuracy=0.313/0.330 (26/52 secs)  
 Epoch 6: cost=1.431, accuracy=0.348/0.310 (26/78 secs)  
 Epoch 8: cost=1.457, accuracy=0.326/0.300 (25/103 secs)  
 Epoch 10: cost=1.403, accuracy=0.358/0.240 (26/129 secs)  
 Model flowers\_model\_1 train ended in 129 secs:  
 Model flowers\_model\_1 test report: accuracy = 0.311, (0 secs)

Model flowers\_model\_1 Visualization



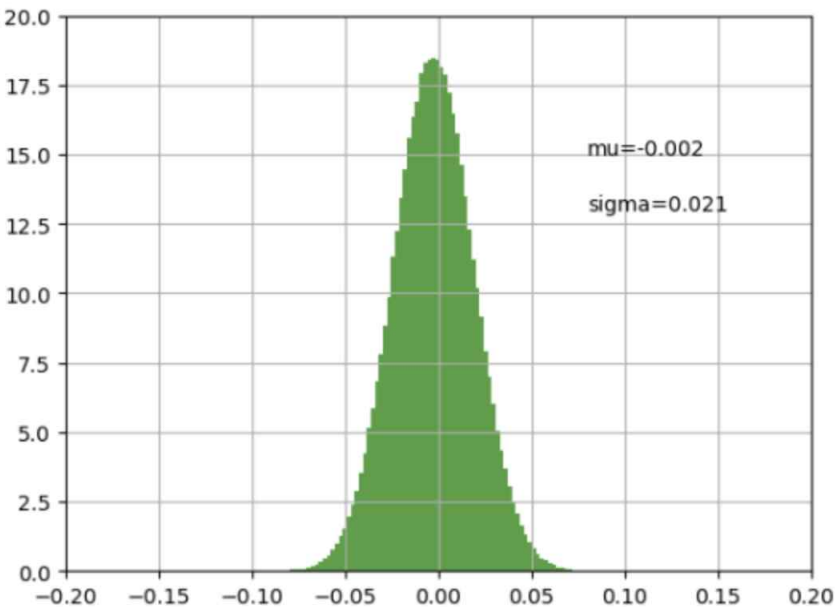
추정확률분포 [15,23,14,25,22] => 추정 sunflower : 정답 daisy => X  
 추정확률분포 [15,23,14,25,22] => 추정 sunflower : 정답 sunflower => O  
 추정확률분포 [15,23,14,25,22] => 추정 sunflower : 정답 tulip => X



Near 0 parameters = 0.0%(192/829790)

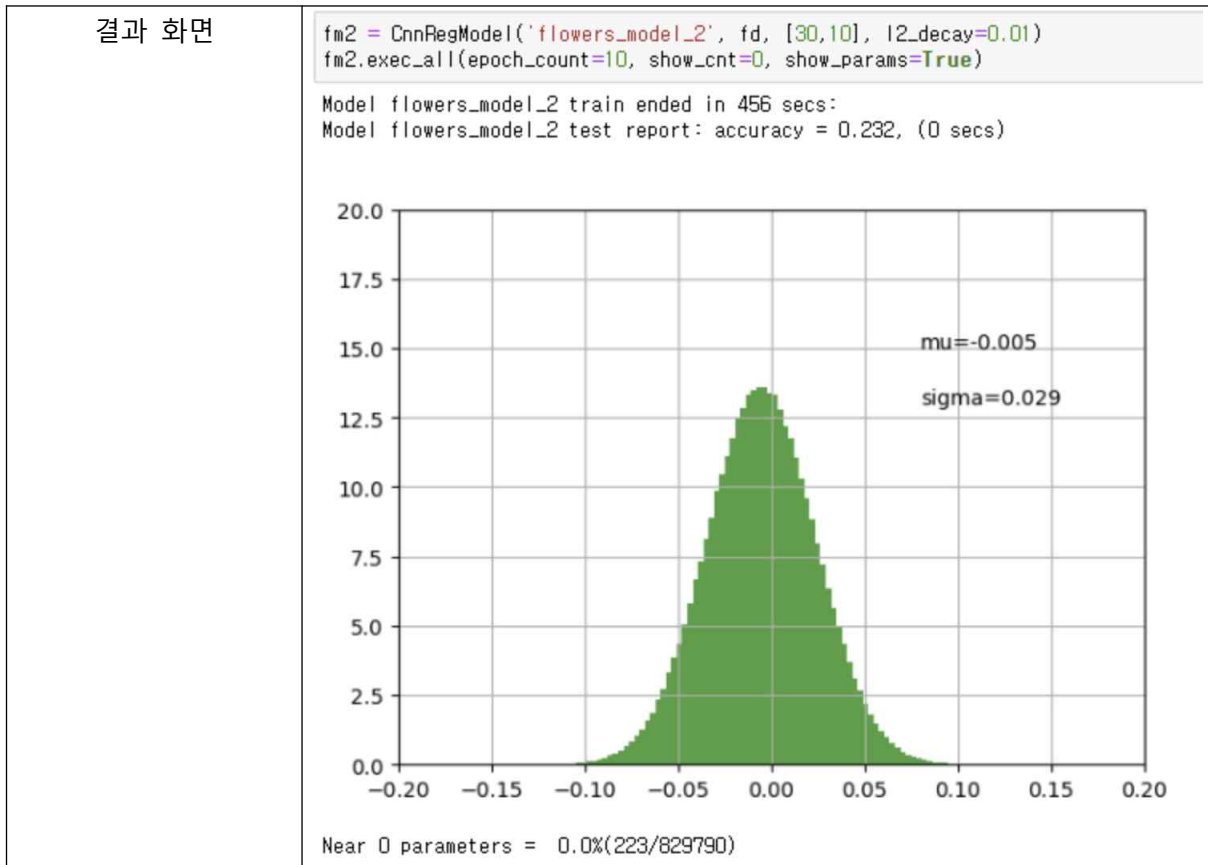
## (2) L2 손실을 적용한 경우1

구분	내용
조건	<ul style="list-style-type: none"> <li>- 은닉 계층 : [30,10]</li> <li>- 에포크 횟수 : 10</li> <li>- 정규화 기법 : L2 손실</li> <li>- 정규화 기법 강도 : 0.1</li> </ul>
데이터 정확도	23.2%
학습 시간	464s
0근처 파라미터 개수	297 / 829790 (0.0%)

파라미터 평균	-0.002
파라미터 분산	0.021
결과 화면	<pre>fm2 = CnnRegModel('flowers_model_2', fd, [30,10], l2_decay=0.1) fm2.exec_all(epoch_count=10, show_cnt=0, show_params=True)</pre> <p>Model flowers_model_2 train ended in 454 secs: Model flowers_model_2 test report: accuracy = 0.232, (0 secs)</p>  <p>Near 0 parameters = 0.0%(297/829790)</p>

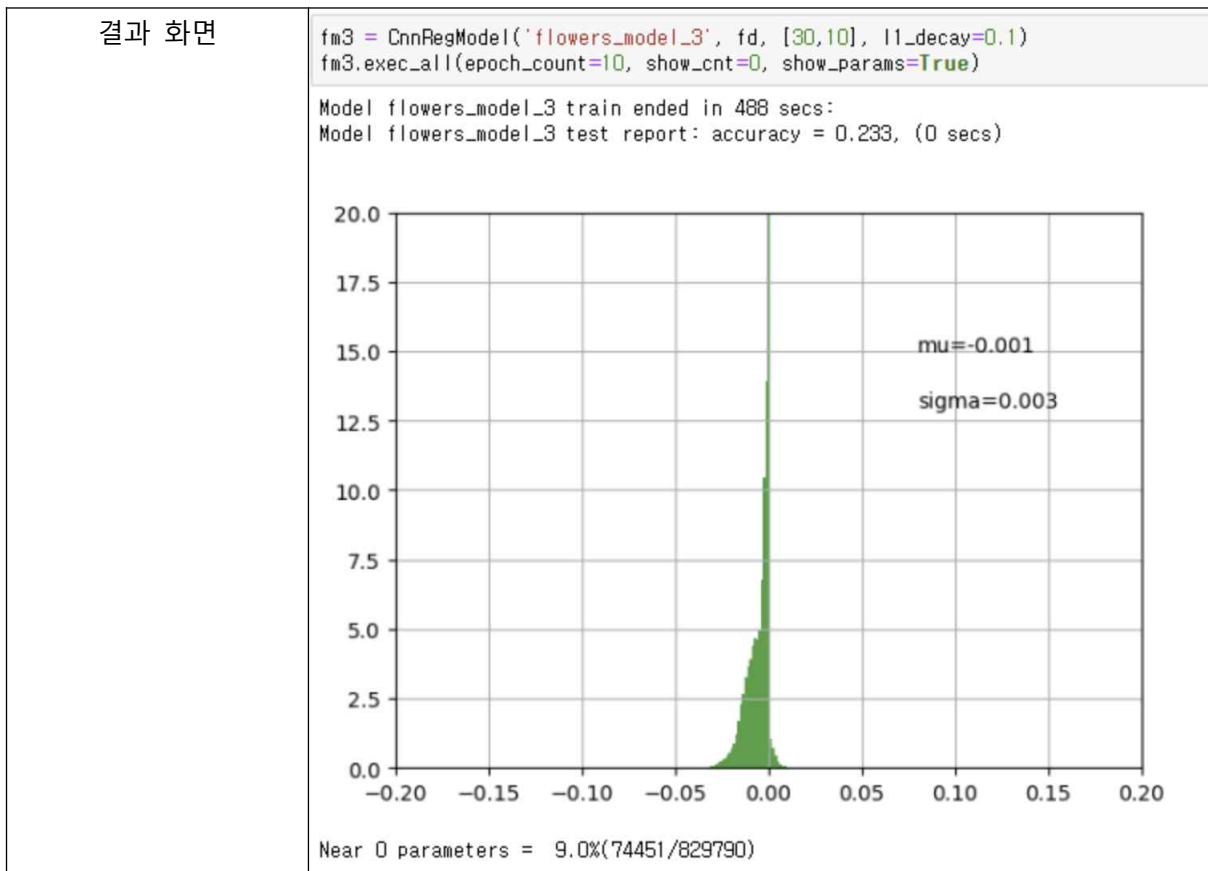
(2) L2 손실을 적용한 경우2

구분	내용
조건	<ul style="list-style-type: none"> <li>- 은닉 계층 : [30,10]</li> <li>- 에포크 횟수 : 10</li> <li>- 정규화 기법 : L2 손실</li> <li>- 정규화 기법 강도 : 0.01</li> </ul>
데이터 정확도	23.2%
학습 시간	456s
0근처 파라미터 개수	223 / 829790 (0.0%)
파라미터 평균	-0.005
파라미터 분산	0.029



(4) L1 손실을 적용한 경우1

구분	내용
조건	<ul style="list-style-type: none"> <li>- 은닉 계층 : [30,10]</li> <li>- 에포크 횟수 : 10</li> <li>- 정규화 기법 : L1 손실</li> <li>- 정규화 기법 강도 : 0.1</li> </ul>
데이터 정확도	23.3%
학습 시간	488s
0근처 파라미터 개수	74451 / 829790 (9.0%)
파라미터 평균	-0.001
파라미터 분산	0.003



#### 4) L1 손실을 적용한 경우2

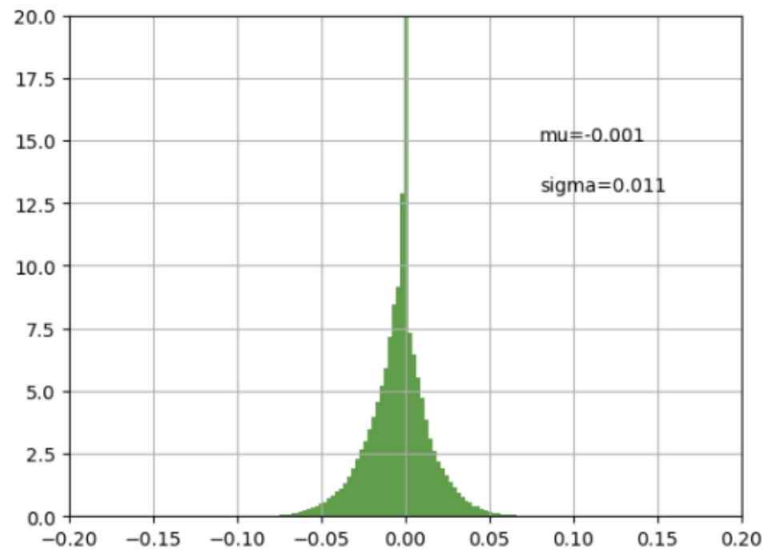
구분	내용
조건	<ul style="list-style-type: none"> <li>- 은닉 계층 : [30,10]</li> <li>- 에포크 횟수 : 10</li> <li>- 정규화 기법 : L1 손실</li> <li>- 정규화 기법 강도 : 0.01</li> </ul>
데이터 정확도	30.4%
학습 시간	548s
0근처 파라미터 개수	556584 / 829790 (67.1%)
파라미터 평균	-0.001
파라미터 분산	0.011

## 결과 화면

```
fm3 = CnnRegModel('flowers_model_3', fd, [30,10], l1_decay=0.01)
fm3.exec_all(epoch_count=10, show_cnt=0, show_params=True)
```

Model flowers\_model\_3 train ended in 548 secs:

Model flowers\_model\_3 test report: accuracy = 0.304, (0 secs)






Near 0 parameters = 67.1%(556584/829790)

## 2.2 드롭아웃, 잡음 주입, 배치 정규화를 이용한 경우

### (1) 정규화를 이용하지 않은 경우

구분	내용
조건	<ul style="list-style-type: none"> <li>- 은닉 계층 구성 : conv(3,6)-max(2)-conv(3,12)-max(2)-conv(3,24)-avg(3)</li> <li>- 에포크 횟수 : 10</li> <li>- 정규화 기법 : X</li> </ul>
데이터 정확도	63.2%
소요 시간	611s

결과 화면	<pre> cnn1 = [['conv',{'ksize':3, 'chn':6}],         ['max',{'stride':2}],         ['conv',{'ksize':3, 'chn':12}],         ['max',{'stride':2}],         ['conv',{'ksize':3, 'chn':24}],         ['avg',{'stride':3}]] fcnn1 = CnnRegModel('flowers_cnn_1', fd, cnn1) fcnn1.exec_all(epoch_count=10, report=2) </pre> <p>Model flowers_cnn_1 train started:</p> <p>Epoch 2: cost=1.157, accuracy=0.548/0.520 (172/172 secs)  Epoch 4: cost=0.939, accuracy=0.639/0.550 (176/348 secs)  Epoch 6: cost=0.814, accuracy=0.679/0.630 (85/433 secs)  Epoch 8: cost=0.716, accuracy=0.733/0.610 (89/522 secs)  Epoch 10: cost=0.623, accuracy=0.760/0.700 (89/611 secs)</p> <p>Model flowers_cnn_1 train ended in 611 secs:  Model flowers_cnn_1 test report: accuracy = 0.632, (2 secs)</p> <p>Model flowers_cnn_1 Visualization</p> <div style="display: flex; justify-content: space-around; align-items: center;">    </div> <p>추정확률분포 [ 1, 0.96, 0, 3] =&gt; 추정 rose : 정답 rose =&gt; 0  추정확률분포 [ 5.93, 1, 0, 1] =&gt; 추정 dandelion : 정답 dandelion =&gt; 0  추정확률분포 [ 0, 0.24, 1, 75] =&gt; 추정 tulip : 정답 tulip =&gt; 0</p>
-------	--

## (2) 드롭아웃을 적용한 경우

구분	내용
조건	<ul style="list-style-type: none"> <li>- 은닉 계층 구성 : conv(3,6)-max(2)-conv(3,12)-max(2)-conv(3,24)-avg(3)</li> <li>- 에포크 횟수 : 10</li> <li>- 정규화 기법 : 드롭아웃</li> <li>- 유지율 : 0.6</li> </ul>
데이터 정확도	45.2%
소요 시간	445s

결과 화면	<pre> cnn2 = [['conv',{'ksize':3, 'chn':6}],         ['max',{'stride':2}],         ['dropout', {'keep_prob':0.6}],         ['conv',{'ksize':3, 'chn':12}],         ['max',{'stride':2}],         ['dropout', {'keep_prob':0.6}],         ['conv',{'ksize':3, 'chn':24}],         ['avg',{'stride':3}],         ['dropout', {'keep_prob':0.6}]] fcnn2 = CnnRegModel('flowers_cnn_2', fd, cnn2) fcnn2.exec_all(epoch_count=10, report=2, show_cnt=0)  Model flowers_cnn_2 train started: Epoch 2: cost=1.271, accuracy=0.456/0.430 (88/88 secs) Epoch 4: cost=1.061, accuracy=0.586/0.520 (89/177 secs) Epoch 6: cost=0.925, accuracy=0.644/0.520 (89/266 secs) Epoch 8: cost=0.877, accuracy=0.665/0.630 (88/354 secs) Epoch 10: cost=0.865, accuracy=0.668/0.560 (91/445 secs) Model flowers_cnn_2 train ended in 445 secs: Model flowers_cnn_2 test report: accuracy = 0.452, (3 secs) </pre>
-------	--

### (3) 잡음 주입을 적용한 경우

구분	내용
조건	<ul style="list-style-type: none"> <li>- 은닉 계층 구성 : conv(3,6)-max(2)-conv(3,12)-max(2)-conv(3,24)-avg(3)</li> <li>- 에포크 횟수 : 10</li> <li>- 정규화 기법 : 잡음 주입</li> <li>- 잡음 조건 : 정규분포, 평균0, 표준편차0.01</li> </ul>
데이터 정확도	66.9%
소요 시간	483s
결과 화면	<pre> noise_std = 0.01 cnn3 = [['noise', {'type':'normal', 'mean':0, 'std':noise_std}],         ['conv',{'ksize':3, 'chn':6}],         ['max',{'stride':2}],         ['noise', {'type':'normal', 'mean':0, 'std':noise_std}],         ['conv',{'ksize':3, 'chn':12}],         ['max',{'stride':2}],         ['noise', {'type':'normal', 'mean':0, 'std':noise_std}],         ['conv',{'ksize':3, 'chn':24}],         ['avg',{'stride':3}]] fcnn3 = CnnRegModel('flowers_cnn_3', fd, cnn3) fcnn3.exec_all(epoch_count=10, report=2, show_cnt=0)  Model flowers_cnn_3 train started: Epoch 2: cost=1.114, accuracy=0.566/0.540 (91/91 secs) Epoch 4: cost=0.926, accuracy=0.653/0.590 (93/184 secs) Epoch 6: cost=0.812, accuracy=0.694/0.640 (100/284 secs) Epoch 8: cost=0.712, accuracy=0.729/0.640 (100/384 secs) Epoch 10: cost=0.606, accuracy=0.776/0.640 (99/483 secs) Model flowers_cnn_3 train ended in 483 secs: Model flowers_cnn_3 test report: accuracy = 0.669, (2 secs) </pre>

### (4) 배치 정규화를 적용한 경우



구분	내용
조건	<ul style="list-style-type: none"> <li>- 은닉 계층 구성 : conv(3,6)-max(2)-conv(3,12)-max(2)-conv(3,24)-avg(3)</li> <li>- 에포크 횟수 : 10</li> <li>- 정규화 기법 : 배치 정규화</li> </ul>
데이터 정확도	56.7%
소요 시간	518s
결과 화면	<pre> cnn4 = [['batch_normal'],         ['conv',{'ksize':3, 'chn':6}],         ['max',{'stride':2}],         ['batch_normal'],         ['conv',{'ksize':3, 'chn':12}],         ['max',{'stride':2}],         ['batch_normal'],         ['conv',{'ksize':3, 'chn':24}],         ['avg',{'stride':3}]] fcnn4 = CnnRegModel('flowers_cnn_4', fd, cnn4) fcnn4.exec_all(epoch_count=10, report=2, show_cnt=0) </pre> <p>Model flowers_cnn_4 train started:  Epoch 2: cost=1.093, accuracy=0.570/0.390 (105/105 secs)  Epoch 4: cost=0.922, accuracy=0.656/0.350 (106/211 secs)  Epoch 6: cost=0.797, accuracy=0.695/0.500 (105/316 secs)  Epoch 8: cost=0.723, accuracy=0.728/0.430 (100/416 secs)  Epoch 10: cost=0.636, accuracy=0.765/0.640 (102/518 secs)  Model flowers_cnn_4 train ended in 518 secs:  Model flowers_cnn_4 test report: accuracy = 0.567, (2 secs)</p>

### 2.3 오피스31 데이터셋 적용

구분	내용
조건	<ul style="list-style-type: none"> <li>-CNN:cnn1-정규화 기법X, cnn2-드롭아웃, cnn3-잡음주입,cnn4-배치정규화</li> <li>- 에포크 횟수 : 10</li> </ul>
결과 화면	<pre> ocnn1 = CnnRegModel('office31_cnn_1', od, cnn1) ocnn2 = CnnRegModel('office31_cnn_2', od, cnn2) ocnn3 = CnnRegModel('office31_cnn_3', od, cnn3) ocnn4 = CnnRegModel('office31_cnn_4', od, cnn4)  ocnn1.exec_all(epoch_count=10, show_cnt=0) ocnn2.exec_all(epoch_count=10, show_cnt=0) ocnn3.exec_all(epoch_count=10, show_cnt=0) ocnn4.exec_all(epoch_count=10, show_cnt=0) </pre> <p>Model office31_cnn_1 train ended in 525 secs:  Model office31_cnn_1 test report: accuracy = 0.874+0.558, (3 secs)</p> <p>Model office31_cnn_2 train ended in 550 secs:  Model office31_cnn_2 test report: accuracy = 0.797+0.530, (3 secs)</p> <p>Model office31_cnn_3 train ended in 607 secs:  Model office31_cnn_3 test report: accuracy = 0.906+0.541, (4 secs)</p> <p>Model office31_cnn_4 train ended in 595 secs:  Model office31_cnn_4 test report: accuracy = 0.894+0.390, (4 secs)</p>

### 3. 분석

#### 3.1 결과 분석

L2 손실과 L1손실에 대한 실험 결과는 다음과 같다. 은닉계층 [30,10], 에포크 횟수 10의 조건에서 실험을 진행하였다.

정규화	강도	정확도(%)	학습 시간(s)	0근처 파라미터 개수(총 개수: 829790)
적용 X	-	31.1	129	192 (0.0%)
L2 손실	0.1	23.2	464	297 (0.0%)
	0.01	23.2	456	223 (0.0%)
L1 손실	0.1	23.3	488	74451 (9.0%)
	0.01	30.4	548	556584 (67.1%)

위 표의 내용과 같이, 정규화를 적용시켰을 때, 적용시키지 않았을 때보다 학습 시간이 약 3.6배 이상 오래 걸렸음을 알 수 있다. L1 손실을 적용하였을 때, 0근처 파라미터 개수가 눈에 띄게 많이 나타났는데, 이는 L1 손실이 가중치 값을 0에 가깝게 만들 수 있는 특성을 가지고 있기 때문일 수 있다. L1과 L2 손실의 강도를 조절하면서 정확도와 파라미터 분포의 변화를 관찰했는데, L1 손실의 경우, 정규화 강도를 낮추면서 모델의 정확도가 증가하고 파라미터 중 0에 가까운 값이 많아졌다는 것을 확인할 수 있었다.

드롭아웃, 잡음 주입, 배치 정규화를 이용한 실험 결과는 다음과 같다. 은닉 계층 구성 conv(3,6)-max(2)-conv(3,12)-max(2)-conv(3,24)-avg(3), 에포크 횟수 10의 조건에서 실험을 진행하였다.

정규화	조건	정확도(%)	학습 시간(s)
X	-	63.2	611
드롭아웃	유지율 : 0.06	45.2	445
잡음 주입	정규분포, 평균0, 표준편차0.01	66.9	483
배치 정규화	-	56.7	518

표에서 볼 수 있듯이, 정규화를 이용하지 않았을 때보다 이용했을 때, 학습시간이 단축되어 나타났다. 드롭아웃을 사용한 모델은 정확도가 줄었지만 학습 시간은 비교적 짧은 것으로 보아 드롭아웃은 모델의 복잡성을 줄이는 데 도움이 될 수 있음을 유추해볼 수 있다. 잡음 주입을 사용한 모델은 상대적으로 높은 정확도를 달성했고, 학습 시간은 중간 정도였으므로

일반화 능력을 향상시키는 데 도움이 되는 것으로 보인다. 배치 정규화를 사용한 모델은 성능은 다른 방법에 비해 조금 떨어지게 나타났지만 학습 속도를 향상시키고 수렴 속도를 높일 수 있는 방법이 될 것이다.

이를 오피스31 데이터셋에 적용시켜본 결과는 다음과 같다.

정규화	X	드롭아웃	잡음 주입	배치 정규화
정확도	0.874+0.558	0.797+0.530	0.906+0.541	0.894+0.390
학습 시간(s)	525	550	607	595

꽃 이미지 데이터셋을 이용했을 때와 비슷한 결과를 나타냄을 알 수 있다. 드롭아웃의 경우 정규화를 적용시키지 않았을 때보다 낮은 정확도를 보였고, 잡음 주입이 높은 정확도를 보였음을 알 수 있다. 꽃 이미지 데이터셋을 이용했을 때는 정규화를 사용하지 않은 경우가 학습 시간이 길었지만 오피스31 데이터셋을 가지고 테스트했을 때는 이 경우가 학습시간이 가장 짧게 나타났다.

### 3.2 추가 분석

#### (1) 과적합(overfitting)현상과 판단 기준

과적합 : 모델이 학습 데이터에 너무 맞춰져서 새로운 데이터에 대한 일반화 성능이 떨어지는 현상을 말한다. 전체적인 문제 특성 파악 없이 학습 데이터의 지역적인 특성을 암기하여 학습 단계에서는 좋은 성과를 보이지만, 평가 단계에서 정확도가 크게 저하하는 모습을 보인다. 문제 난도에 비해 모델 용량이 크면서 데이터가 부족할 때 주로 발생하게 된다.

과적합 판단 기준 : 학습 시 주어진 데이터에 대한 오차는 줄어들이지만, 검증 데이터에 대한 오차는 어느 순간부터 상승하는 현상이 나타나는 경우 과적합이 발생할 가능성이 높다. 따라서, 검증 데이터에 대한 정확도가 감소하는 지점을 찾아내고, 이 지점 이후에 모델이 과적합될 가능성이 높다고 판단할 수 있다.

\* 과적합 예방을 위한 방법 : 데이터가 적거나, 학습에 사용된 데이터와 검증에 사용된 데이터가 유사한 패턴을 가질 때 과적합이 발생할 가능성이 높으므로, 다양한 데이터를 사용하고 모델이 보지 못한 데이터에 대해서도 일반화 할 수 있도록 해야한다.

이 뿐만 아니라, 모델에 규제 기법을 적용하여 가중치를 제한하거나, 드롭아웃 등의 기법을 통해 과적합을 줄일 수 있을 것이다.

#### (2) 정규화의 필요성 및 효과

정규화는 모델의 복잡도를 감소시키고, 과적합을 줄이며, 일반화 성능을 향상시키는 효과가 있다. 이는 가중치 값을 제한하고 모델의 복잡성을 줄여 새로운 데이터에 대한 일반화 능력

을 향상시킬 수 있다. 더불어, 데이터 스케일의 차이를 줄여주고, 수렴 속도를 향상시키는 등 모델의 안정성을 높이는 효과도 가진다.

### (3) 5가지 정규화 기법의 특징

- 1. L2 손실 : 절댓값이 큰 파라미터에 불이익을 주는 정규화 기법, 파라미터 값의 폭주를 막고 기왕이면 작은 절댓값을 갖도록 압박한다. 본연의 학습 효과 때문에 무조건 모든 파라미터의 절댓값이 줄지는 않는다. 손실함수는 기존의 손실 함수에 L2 페널티(가중치의 제곱의 합)를 더한 값이다.

- 2. L1 손실 : 절댓값이 큰 파라미터에 불이익을 주는 정규화 기법, 전체적으로 L2 손실과 비슷하지만 더욱 강한 압박을 준다. 손실함수는 기존의 손실 함수에 L1 페널티(가중치의 절댓값의 합)를 더한 값이다.

- 3. 드롭아웃 : 계층 입력 중 일부만 이용해 신경망을 학습시키는 규제화 기법이다. 과적합 방지를 위한 기법이지 계산량을 줄이는 기법은 아니다.

학습 단계 순전파 처리를 위해서 미니배치 데이터 전체 형태의 마스크 생성한다. keep\_ratio 확률로 1, 나머지 확률로 0 되는 난수 함수로 마스크 생성하는데, 생성된 마스크를 입력에 곱해 중간 출력을 생성하고 기대값 보정 위해 중간 출력에  $1/\text{keep\_ratio}$ 를 곱해 출력을 생성한다.

학습 단계 역전파 처리를 위해서 순전파 처리에 사용된 마스크를 재활용해 처리한다. 출력 손실기울기에  $1/\text{keep\_ratio}$ 를 곱한 후 순전파 처리의 마스크를 곱해 입력 손실 기울기로 반환한다.

- 4. 잡음 주입 : 두 계층 사이에 삽입되어 은닉 벡터에 잡음을 추가해 전달하는 방식이다. 매번 다른 잡음을 주입함으로써 판에 박힌 처리를 방해하기에 모델의 과적합을 방지할 수 있다. 순전파 처리 단계에서는 학습 단계에 한해 랜덤한 잡음을 생성해 주입하고, 그 결과 강건한 학습이 이루어질 수 있다. 역전파 처리 단계에서는 별도의 처리 없이 출력 손실 기울기를 바로 입력 손실 기울기로 반환하면 된다.

- 5. 배치 정규화 : 미니배치 내 데이터들에 대해 벡터 성분 별로 정규화를 수행하는 기법이다. 정규화는 동일한 선형변환으로 대상값들을 평균 0, 표준편차 1 분포로 만드는 처리를 말하는데, 입력 성분 간 분포 차이로 인한 학습 불균형을 방지하기 위해 도입한다. 데이터셋이 아닌 은닉 벡터에도 적용이 가능하며, 에포크마다 무작위로 미니배치 구성이 바뀌기 때문에 강건한 학습이 가능하다. 모든 데이터를 학습에 충분히 이용할 수 있으므로 일부 성분만을 결정에 참여시키는 드롭아웃보다 유리하다.

### (4) 정규화 기법 적용 시 모델 성능 변화

- 1. L1 손실 : 덜 중요한 가중치를 0으로 만들어 모델의 복잡도를 줄이고 일반화 성능을 향상시킬 수 있다.
- 2. L2 손실 : 모델의 가중치를 줄여 과적합을 방지하고 일반화 성능을 개선할 수 있다.
- 3. 드롭아웃 : 단일한 패턴이나 뉴런에 지나치게 의존하지 않도록 하여 과적합을 방지하고 일반화 성능을 향상시킬 수 있다.
- 4. 잡음 주입 : 모델이 잡음에 대한 내성을 가지도록하여 일반화 성능을 향상시킨다.
- 5. 배치 정규화 : 학습 시 각 레이어의 입력을 정규화함으로써 그래디언트 소실과 폭주 문제를 완화시켜 학습을 안정화시킬 수 있다.

#### 4. 결론

실험 결과, 정규화를 적용하는 것이 모델의 학습 시간을 늘리지만, 일반적으로 더 나은 성능을 보였다. L1 손실은 파라미터 값을 0에 가깝게 만들어 특정 가중치를 줄이는 데 효과적이었으며, 정확도와 파라미터 분포에 영향을 미쳤다. 반면, 드롭아웃은 정확도는 다소 떨어졌지만 모델의 복잡성을 줄이는 데 도움이 되었고, 잡음 주입은 상대적으로 높은 정확도를 달성하며 일반화 능력을 향상시키는 모습을 보였다. 마지막으로, 배치 정규화는 학습 속도를 향상시키고 수렴 속도를 높일 수 있었으나 성능 면에서 약간의 저하가 나타났다.