Undergraduate Research Internship in Affective AI LAB.

혼자 공부하는 머신러닝 & 딥러닝

7주차 : ch08. 이미지를 위한 인공 신경망

컴퓨터공학과 12201682 공지윤

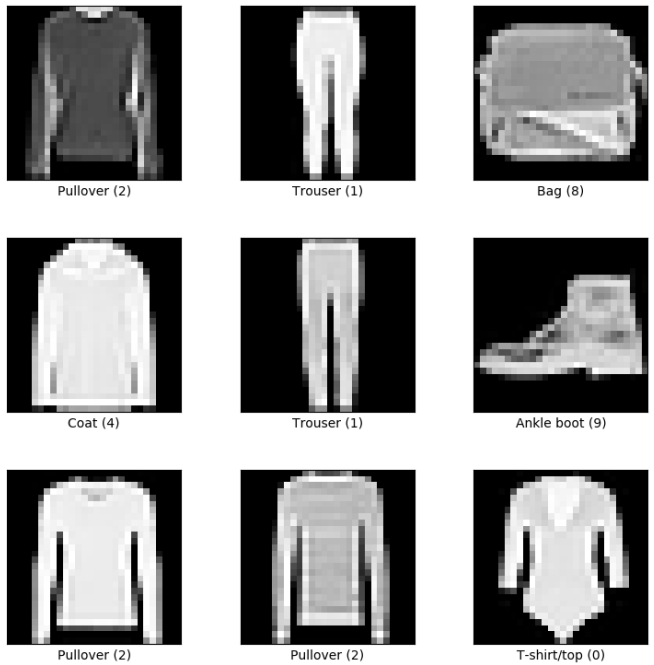
목차

1. 합성곱 신경망 (CNN)

[등장 배경]

< MNIST 데이터셋 >

1. 분류 대상이 이미지에서 고정된 위치에 존재한다.
2. 배경 값이 0으로 되어 있어서 인식하고자 하는 대상과 구분이 명확하다.
3. 분류 대상이 정가운데에 위치한다.
4. 크기가 28 X 28로 매우 작다.



- 이러한 특성으로 Dense Layer로만 학습해도 90% 정도의 정확도를 얻을 수 있음

1. 합성곱 신경망 (CNN)

[등장 배경]

< 일반적인 이미지 데이터 >

1. 분류 대상이 이미지에서 고정된 위치에 존재하지 않는다.
2. 배경에 다른 사물이 있을 수 있다.
3. 앞 모습이 아니라 옆 모습 또는 뒷모습을 나타낼 수 있다.
4. 크기가 작지 않은 경우가 대부분이다.

→ 이러한 특성으로 Flatten으로 뭉개서 학습시키는 방식은 불가능하다.

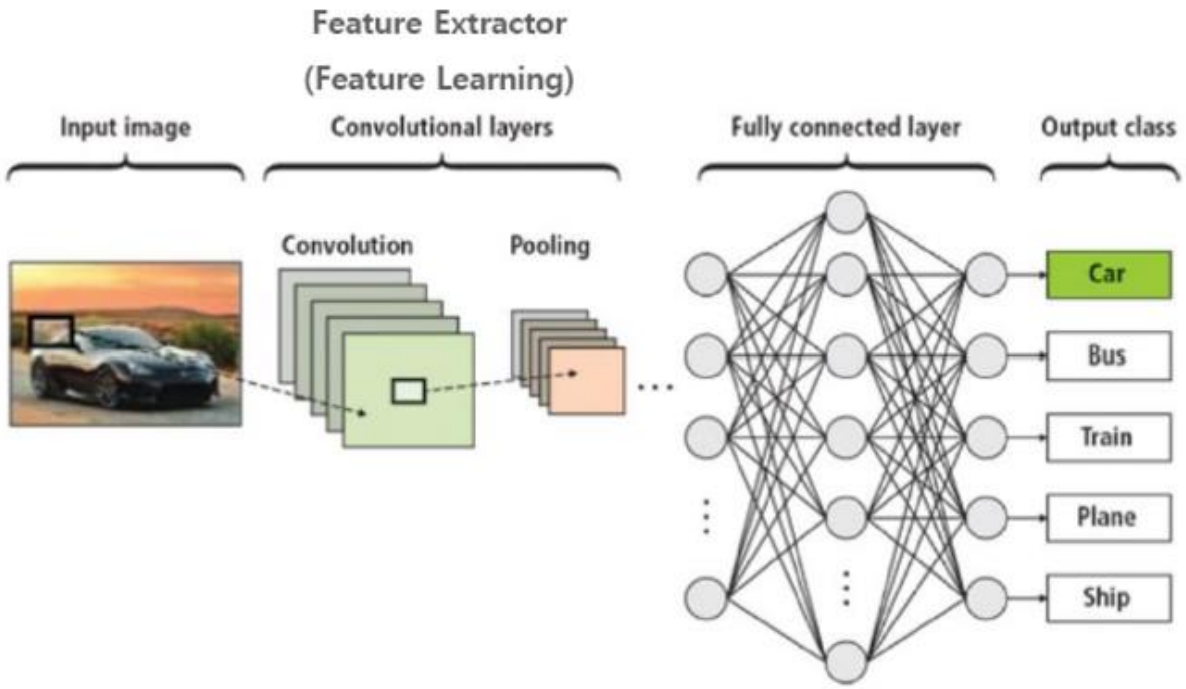
< 추가 > 이미지의 크기가 커질수록 너무 많은 Weight가 필요하다.

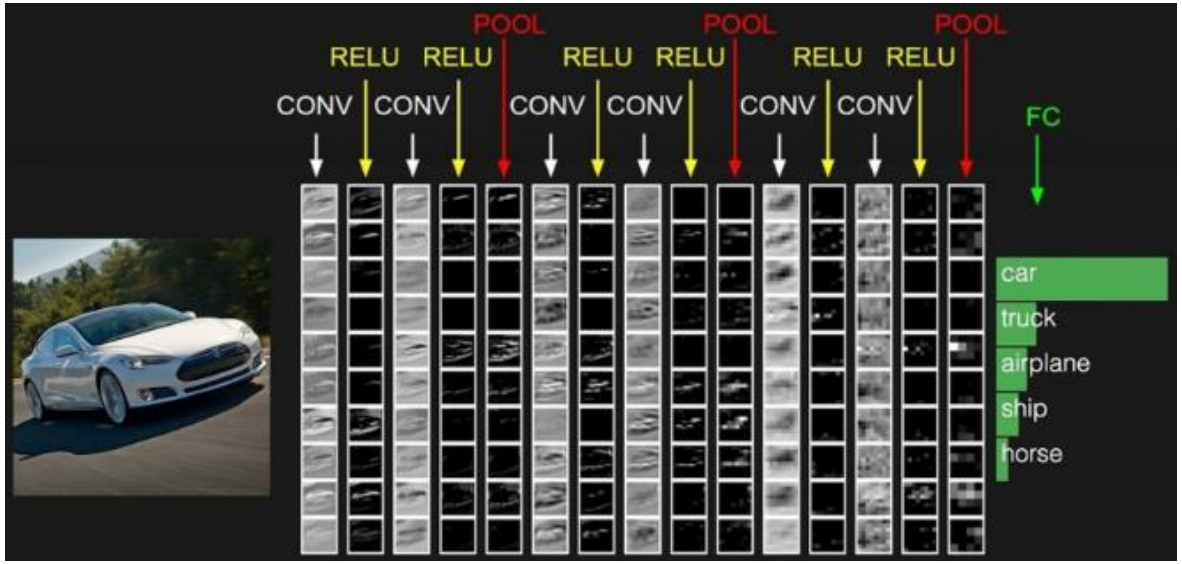
예) 500 X 500 크기의 이미지를 Flatten -> Dense로 학습한다고 하자.

Hidden Layer 1에서 Weight를 1,000개로 설정하면? 250,000 X 1,000 = 250,000,000

1. 합성곱 신경망 (CNN)

[CNN 구조]





1. CNN은 Feature Extractor와 Classifier로 구성된다.

2. CNN은 Classification에 맞는 최적의 Feature를 추출한다.

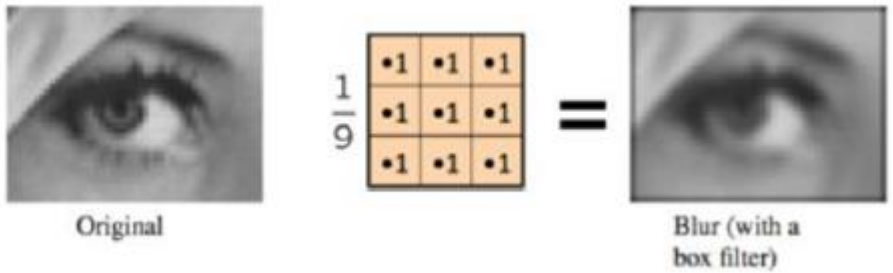
3. 최적의 Feature를 추출하기 위한 최적의 Weight 값을 계산한다. 이때, Weight는 '필터의 Weight' 값이다.

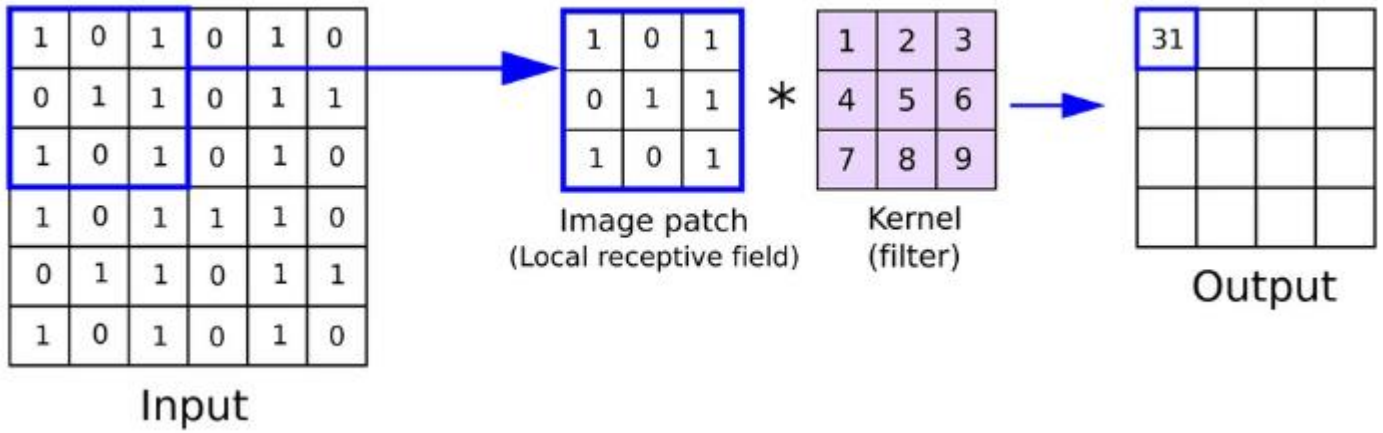
1. 합성곱 신경망 (CNN)

[CNN 구조]

< 필터 (Filters) >

- 원본 이미지에 변형을 주는 것
- 다양한 수식을 적용해서 픽셀 배열을 변경한다.



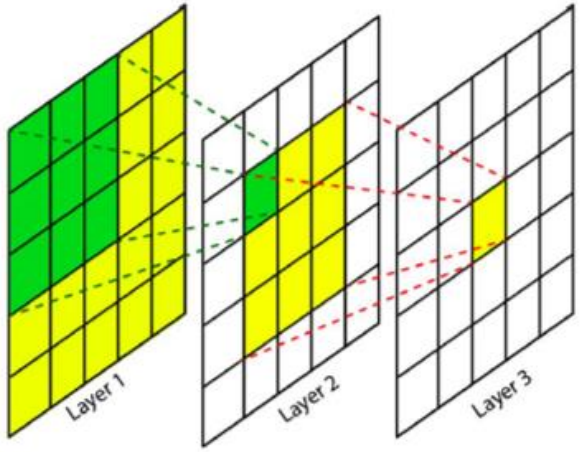


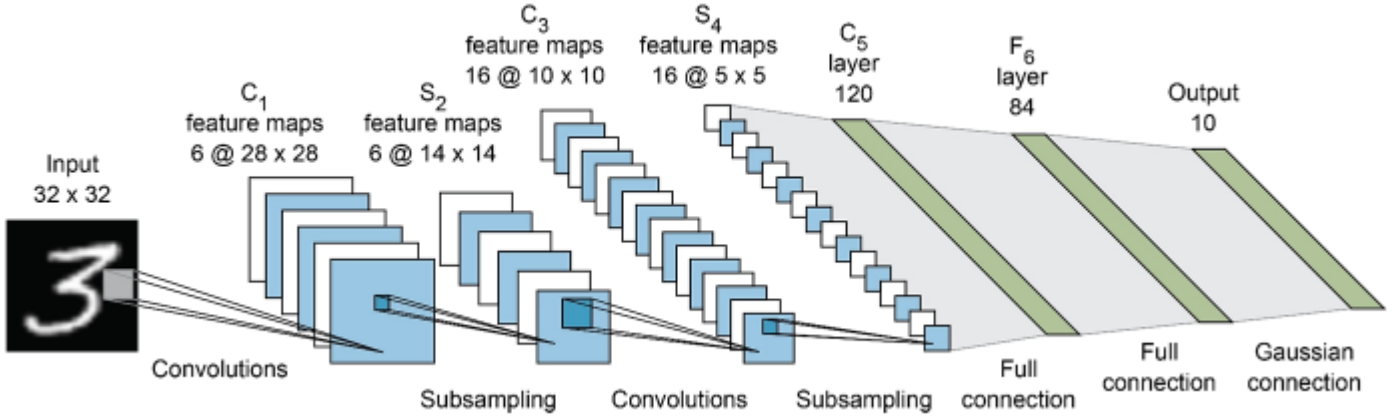
1. 합성곱 신경망 (CNN)

[CNN 구조]

< 커널 (Kernels) >

- 필터는 여러 개의 커널 (채널)로 구성됨
- 개별 커널은 필터 내에서 서로 다른 값을 가질 수 있음
- 정방행렬인 경우가 대부분!
- 커널의 크기가 클수록 입력 피처맵 (또는 원본 이미지)에서 더 큰 (더 많은) 피처 정보를 가져올 수 있음
- 큰 사이즈로 설정할 경우 연산을 더 많이 해야 하며, 더 많은 파라미터가 필요하다는 단점이 있다.



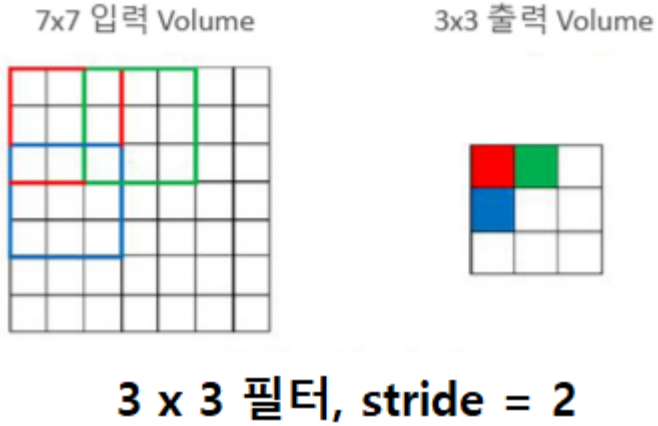


1. 합성곱 신경망 (CNN)

[CNN 구조]

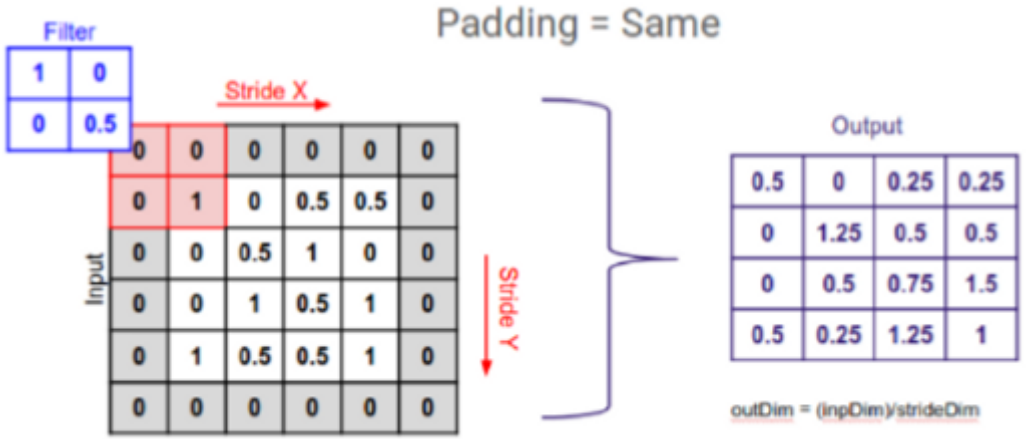
< Stride >

- 입력 데이터에 convolution filter를 적용할 때 Sliding Window가 이동하는 간격
- 연산 속도를 증가시키고 불필요한 특성을 제거할 수 있다.
- 공간적인 피처 특성을 손실할 가능성이 높아진다.



< Padding >

- Filter를 적용하여 피처맵의 크기가 점점 작아지는 것을 막기 위해 사용한다.
- Filter 적용 전 보존하려는 피처맵의 크기에 맞게 입력 피처맵의 좌우 끝, 상하 끝에 각각 열과 행을 추가하고 0 값을 채운다.

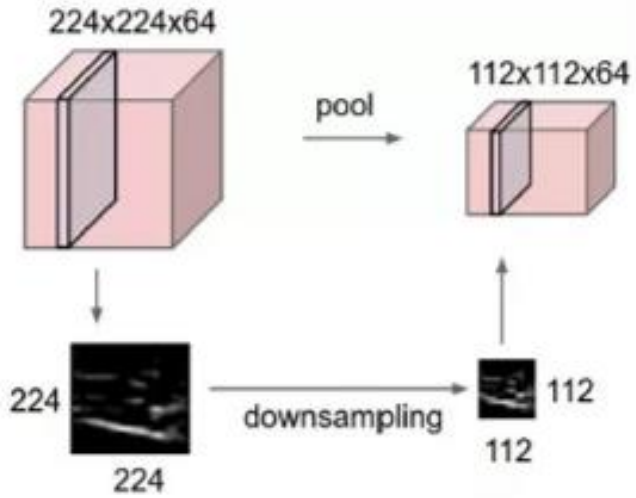


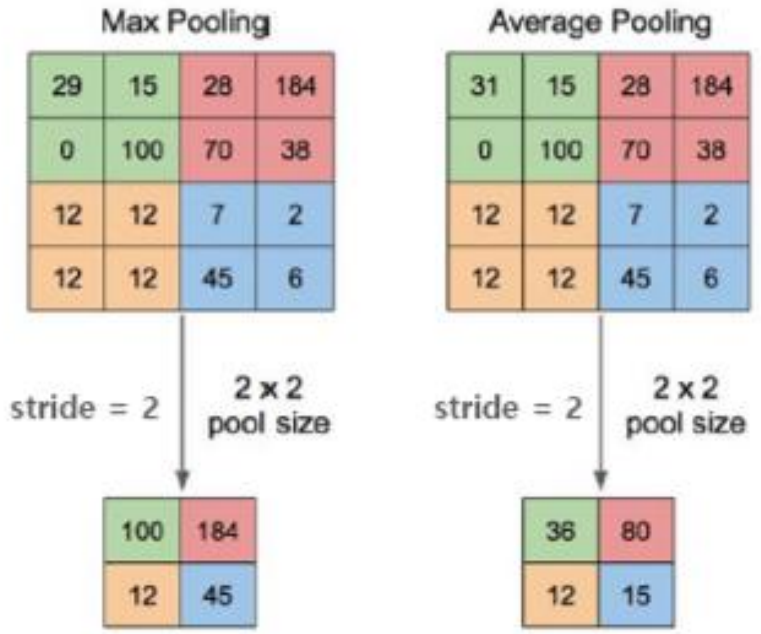
1. 합성곱 신경망 (CNN)

[CNN 구조]

< Pooling >

- Conv 적용된 피처맵이 일정 영역 별로 하나의 값을 추출하여 피처맵의 사이즈를 줄인다.
- Subsampling이라고도 불린다.
- 일반적으로 모든 값이 한번만 처리될 수 있도록 Pooling의 크기와 Stride의 크기를 동일하게 부여한다.
- 보통 Convolution 연산 -> Activation 적용 후에 Pooling을 적용한다.





2. CNN 성능 향상

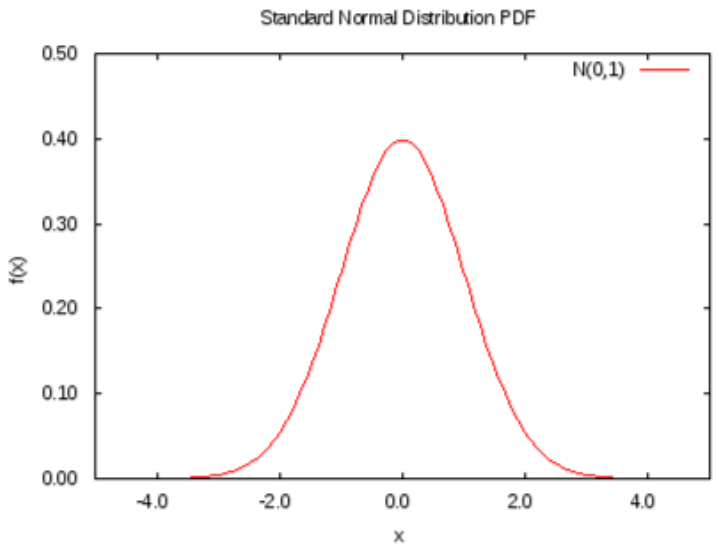
[가중치 초기화]

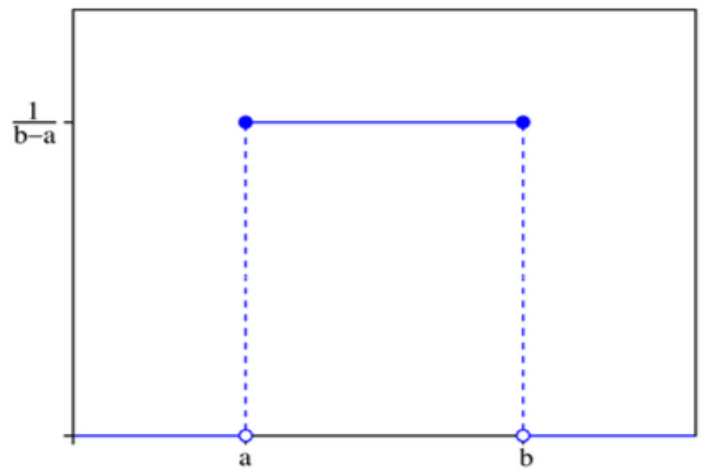
- 경사하강법으로 가중치를 업데이트할 때 가중치의 초기값이 무엇이었는지에 따라서 성능 차이가 매우 크다.
- 값이 너무 크다면 가중치의 수렴이 어렵고, 값이 너무 작다면 가중치 값이 거의 0이 되어버린다.

< 좋은 가중치 초기화 조건 >

1. 값이 동일하면 안 된다.
2. 값이 충분히 작아야 한다.
3. 적당한 분산 (또는 표준편차)을 가져야 한다.

1. 표준 정규 분포 (Normal Distribution)
2. 균등 분포 (Uniform Distribution)



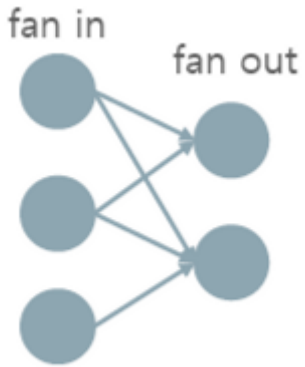


2. CNN 성능 향상

[가중치 초기화]

< Xavier Glorot Initialization >

- 입력 노드와 출력 노드의 개수를 감안하여 동적으로 Weight 초기화
- Tanh, Sigmoid 활성화 함수에서 쓰이는 가중치 초기화



$$std = \sqrt{\frac{2}{fan_{in} + fan_{out}}}$$

$$limit = \sqrt{\frac{6}{fan_{in} + fan_{out}}}$$

< He Initialization >

- 입력 노드의 개수만을 감안하여 동적으로 Weight 초기화
- ReLU 활성화 함수에서 쓰이는 가중치 초기화

$$std = \sqrt{\frac{2}{fan_{in}}}$$

$$limit = \sqrt{\frac{6}{fan_{in}}}$$

Keras의 기본 weight_initializer는 glorot_uniform!

대부분의 activation='relu'인데 initializer를 He로 바꿔야 하는 것 아닐까?

→ 실제로 거의 유사한 성능을 보이므로 큰 의미를 두지는 않는다.

2. CNN 성능 향상

[Batch Normalization]

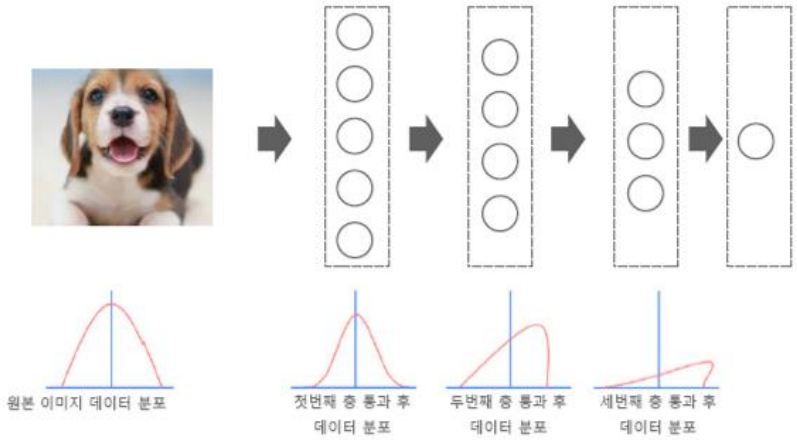
< Normalization vs. Standardization >

- 한국어로 모두 '정규화'
- Normalization은 대표적으로 min-max scaling이다. 0 ~ 1 사이의 값으로 scaling 한다.
- Standardization은 Z-score 변환이다. 평균이 0이고 표준편차가 1인 정규분포로 변환한다.

하지만, Batch Normalization의 정규화는 Standardization이라는 것!

< Internal Covariate Shift >

- 신경망 내부의 각 층을 통과할 때마다 입력 데이터의 분포가 조금씩 변경되는 현상
- 특정 가중치가 다른 가중치보다 독보적으로 크다면? 해당 가중치가 미치는 영향이 너무 커지면서, loss 함수가 해당 가중치에만 집중하게 된다.



2. CNN 성능 향상

[Batch Normalization]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

1. 미니배치 평균과 분산을 구한다.
2. 정규화를 진행한다.
3. Scaling, Shift 파라미터를 가지고 정규화된 weighted sum을 변형한다.

- 딥러닝 학습 과정에서 동적으로 갱신
- 그냥 Z 정규화만 하면 한정된 값만 가지게 된다.
  (예. Sigmoid는 0.5로, ReLU는 0으로)

< 효과 >

1. 뛰어난 성능 향상 효과
2. Regularization 효과

: 정규화 후에 스케일링, shift로 일종의 노이즈 추가 효과

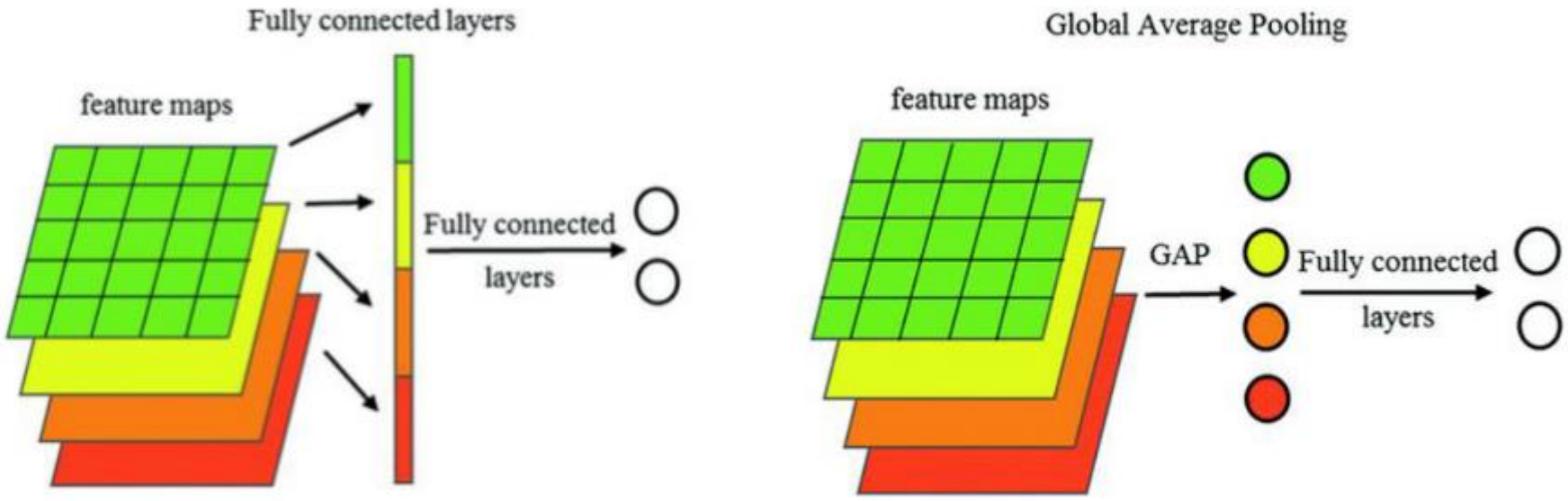
3. 가중치 초기화 설정을 크게 신경 쓸 필요가 없음

2. CNN 성능 향상

[Global Average Pooling]

< 등장 배경 >

- CNN 말단의 Flatten, Dense Layer에서 너무 많은 파라미터들이 생성되었다.
- Global Average Pooling을 이용하여 효과적으로 노드와 파라미터를 줄이고자 한다.



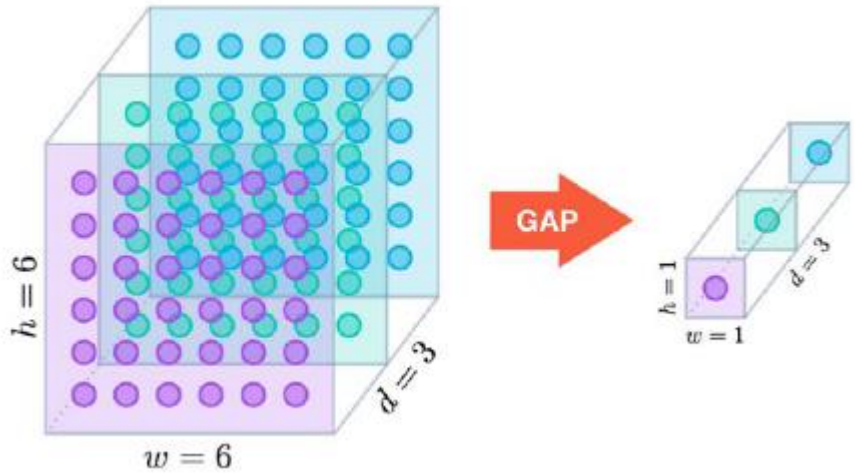
각 채널 별로 평균값을 출력한다.

2. CNN 성능 향상

[Global Average Pooling]

< 특이사항 >

- 피처맵의 채널 개수가 많을 경우에 사용한다. 채널 개수가 적다면 그냥 Flatten을 사용하는 편이 낫다.
- 채널 수가 많다는 것은 적어도 512개의 채널보다 많다는 것이다.



flatten_1 (Flatten)	(None, 8192)	0
dropout_2 (Dropout)	(None, 8192)	0
dense_2 (Dense)	(None, 300)	2457900
dropout_3 (Dropout)	(None, 300)	0
dense_3 (Dense)	(None, 10)	3010

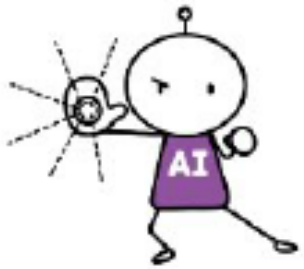
(None, 8192)

global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dropout_4 (Dropout)	(None, 512)	0
dense_4 (Dense)	(None, 300)	153900
dropout_5 (Dropout)	(None, 300)	0
dense_5 (Dense)	(None, 10)	3010

(None, 512)

Undergraduate Research Internship in Affective AI LAB.

감사합니다 :>



컴퓨터공학과 12201682 공지윤