딥러닝 입문
(모델의 성능 끌어올리기)

Histroy

Date	Ver	내용
2020.12.28	v01	7.3 모델의 성능 끌어올리기
2022.07.14	v02	배치 정규화 내용 추가

목차

01 배치 정규화

- 2015년 아이오페와 세게디가 제안한 층의 한 종류
- 케라스에서는 BatchNormalization 클래스로 사용 가능
- 입력 값들을 정규화를 통해 균일하게 만드는 광범위한 방법
- 각 배치(Batch)별로 평균(0)과 분산(1)을 이용하여 정규화시키는 것을 의미.
- 배치 단위로 정규화하기에 배치 정규화라고도 한다.

01 배치 정규화

- Gradient Vanishing/Gradient Exploding이 일어나지 않도록 하는 아이디어 중의 하나.

  (1) 불안정한 변화가 일어나는 원인 중의 하나가 Network의 각 층이나 Activation마다 입력의 분포가 달라지는 현상이 있다.

  ▶ 입력의 분포를 평균 0, 표준편차 1인 입력으로 정규화시키는 방법. 이는 Whitening의 방법으로 해결 가능.

01 배치 정규화 - Whitening

- Whitening은 기본적으로 들어오는 입력의 특징(feature)을

(1) 상관성 없게(uncorrelated)게 만들어주고,

(2) 각각의 분산을 1로 만들어주는 작업.

▶ 이 방법은 계산 량이 많고, 일부 파라미터의 영향이 무시된다.

▶ [해결] Whitening를 해결하기 위한 트릭이 바로 **배치 정규화**

01 배치 정규화(Batch Normalization)

- 각 레이어마다 정규화하는 레이어를 두어, 변형된 분포가 나오지 않도록 조절한다.

- 미니 배치의 평균과 분산을 이용하여 정규화를 수행, scale 및 shift를 감마(γ)값, 베타(β) 값을 통해 실행. 감마, 베타 값을 통해 ReLU가 적용되더라도 기존의 음수 부분이 모두 0이 되지 않도록 방지. 감마와 베타 값은 학습을 통해 구할 수 있음.

- 배치 정규화는 단순하게 평균과 분산을 구하는 것 뿐만 아니라 감마(scale), 베타(shift)를 통한 변환을 통해 비선형 성질을 유지하면서 학습이 될 수 있게 해준다.

01 배치 정규화(Batch Normalization)

▶ $Batch\ Normalization(BN)$

$$BN(X) = \gamma\left(\frac{X - \mu_{batch}}{\sigma_{batch}}\right) + \beta$$

▶batch normalization을 적용하면 weight의 값이 평균이 0, 분산이 1인 상태로 분포. 이 상태에서 ReLU가 activatio으로 적용되면 전체 분포에서 음수에 해당하는 (1/2)비율이 0이 되어 버린다. 성능 개선을 위한 배치 정규화가 의미가 없어진다. 따라서..

▶ γ(감마) 와 β(베타)가 정규화에 곱해지고 더해져서 ReLU가 적용되더라도 기존의 음수 부분의 모두 0으로 되지 않도록 방지해준다. γ(감마) 와 β(베타)는 backpropagation을 통해 학습을 하게 된다.

01 배치 정규화(Batch Normalization)

▶ 장점

- ▶ 학습 속도가 빠르게 가능하다.
- ▶ 가중치 초기화에 대한 민감도를 감소시킨다.
- ▶ 모델의 일반화(regularization)효과가 있다.

02 하이퍼 파라미터 최적화

▶ 하이퍼 파라미터 종류

- 학습률(Learning Rate) – cost가 최소화 되는 방향으로 얼마나 빠르게 이동할 것인가?
- 비용함수(Cost Function) – 입력에 따른 예측 값과 실제 값의 차이를 계산하는 함수
- 훈련 반복 횟수(Epochs)
- 은닉층의 뉴런 개수(Hidden Units)
- 규제 강도(Regularization Strength) – L1 또는 L2정규화 방법
- 가중치 초기값(Weight Initialization)
- 미니 배치 크기(Mini-batch Size) – 1회 학습을 수행(가중치 업데이트)을 위한 학습 데이터 크기
- 학습 조기 종료(Early Stopping) – 학습의 조기 종료를 결정하는 변수

02 하이퍼 파라미터 최적화

▶ 하이퍼 파라미터 적용시 고려하기

- 학습률(Learning Rate) – 너무 작으면 학습 속도 저하, 크면 학습 불가

- 미니 배치 크기(Mini-batch Size)

  가용 메모리 크기와 epoch 수행 성능을 고려한다. 최소 32

  배치 크기는 GPU의 물리적인 구조로 인해 2의 제곱으로 설정을 권장

- 은닉층의 뉴런 개수(Hidden Units)

  첫 Hidden Layer의 뉴런 수가 Input Layer보다 큰 것이 효과적.

02 하이퍼 파라미터 최적화

▶ 하이퍼 파라미터 최적화 방법

- A. Manual Search – 사람의 경험과 직관에 의지해서 찾기
- B. Grid Search – 범위를 정해두고 그 안에서 일정한 간격으로 값을 대입해 보기

  탐색 시간이 매우 오래 걸리고, 효율이 떨어진다.
- C. Random Search(랜덤 서치) – 범위를 정하고 무작위로 최적의 값을 탐색

  그리드 서치에 비해 더 효율적이고 결과도 더 우수하다. 모든 파라미터 중요도는 각각 영향력이 다르기 때문
- D. Bayesian Optimization – 베이즈 최적화

  베이즈 정리(Bayes' theorem)를 기반으로 미지의 목적함수(Objective Function)를 최대화 또는 최소화하는 최적해를 찾는 기법. 단, 모든 문제에 일반적으로 적용할 수 있는 알고리즘은 아니다는 의견이 있음.

02 하이퍼 파라미터 최적화

- 얼마나 많은 유닛이나 필터를 두어야 할까?
- relu 활성화 함수를 사용해야 할까?
- 어떤 층 뒤에 BatchNormalization을 사용해야 할까?
- Dropout은 얼마나 해야 할까?
- 얼마나 층을 쌓아야 할까?

03 하이퍼 파라미터 최적화 과정

(1) 일련의 하이퍼 파라미터를 (자동으로) 선택합니다.

(2) 선택된 하이퍼 파라미터로 모델을 만든다.

(3) 훈련 데이터로 학습하고 검증 데이터에서 최종 성능을 측정합니다.

(4) 다음으로 시도할 하이퍼 파라미터를 (자동으로) 선택합니다.

(5) 이 과정을 반복합니다.

(6) 마지막으로 테스트 데이터에서 성능을 측정합니다.

* Hyperopt와 Hyperas 라이브러리를 이용하여 하이퍼 파라미터 최적화를 수행할 수 있다.

04 모델 앙상블

- 앙상블은 여러 개 다른 모델의 예측을 합쳐서 더 좋은 예측을 만든다.

- 아주 뛰어난 단일 모델보다 여러 개를 합친 앙상블이 성능이 좋습니다.

- 장님과 코끼리에 관한 우화를 생각해 보자.

  => 서로 코끼리의 다른 부분을 만지고 각자의 관점으로 이해한 정답을 이야기합니다.

  이들의 관점을 모으면 훨씬 더 일반화된 코끼리를 만들 수 있습니다.

- 여러 모델의 예측을 합치는 가장 쉬운 방법은 예측의 평균을 내는 것.

04 모델 앙상블

- 앙상블의 핵심은 **다양한 모델을 만드는 것**이 중요합니다.

- 최대한 다르면서 좋은 모델을 앙상블해야 합니다.

- 실전에서 잘 동작하는 한 가지 방법은 트리 기반 모델(랜덤 포레스트나 그래디언트 부스팅 트리)나 심층 신경망을 앙상블하는 것.

- 딥러닝과 얕은 모델을 섞은 넓고 깊은 모델 사용. 심층 신경망과 선형 모델을 함께 훈련.

05 깊이별 분리 합성곱

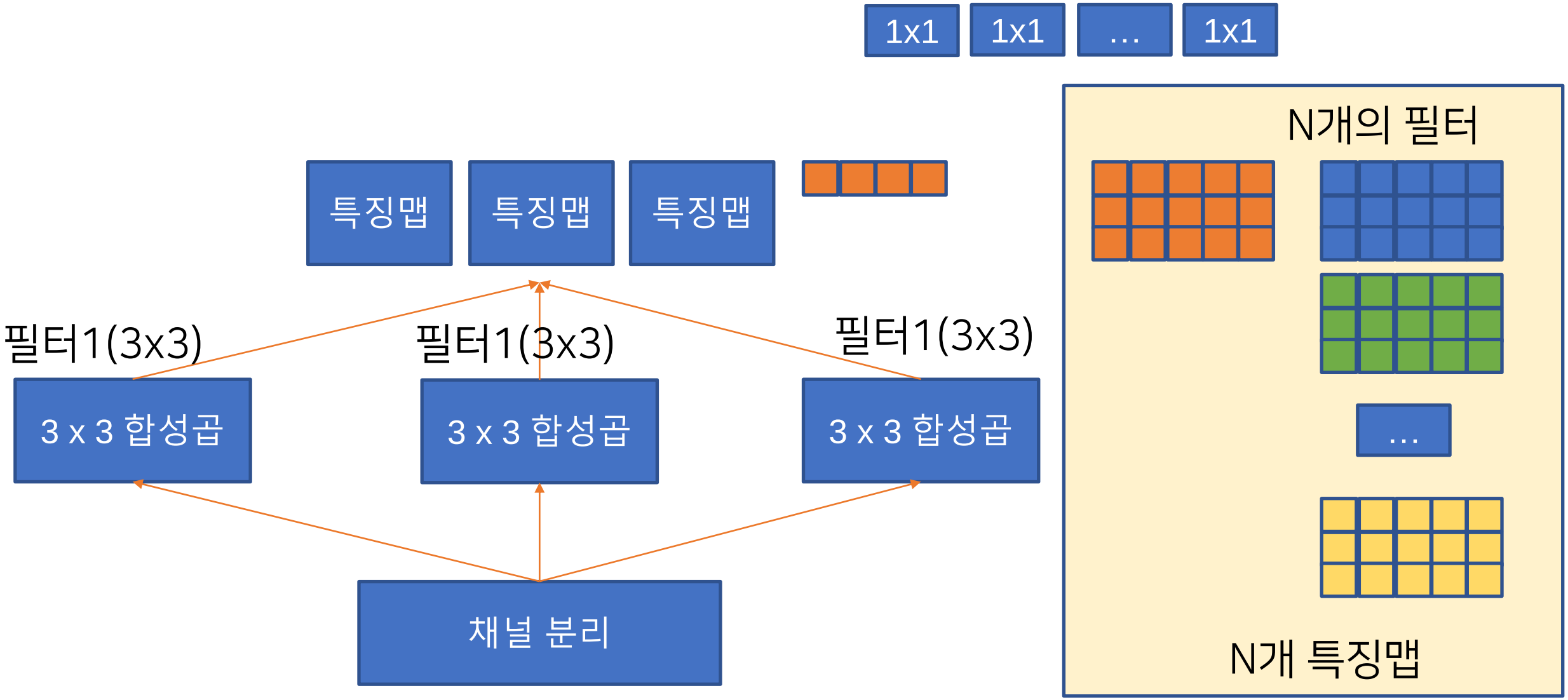
01. 1차적으로 채널별로 합성곱을 수행한다.

02. 각각의 결과물을 만든다.

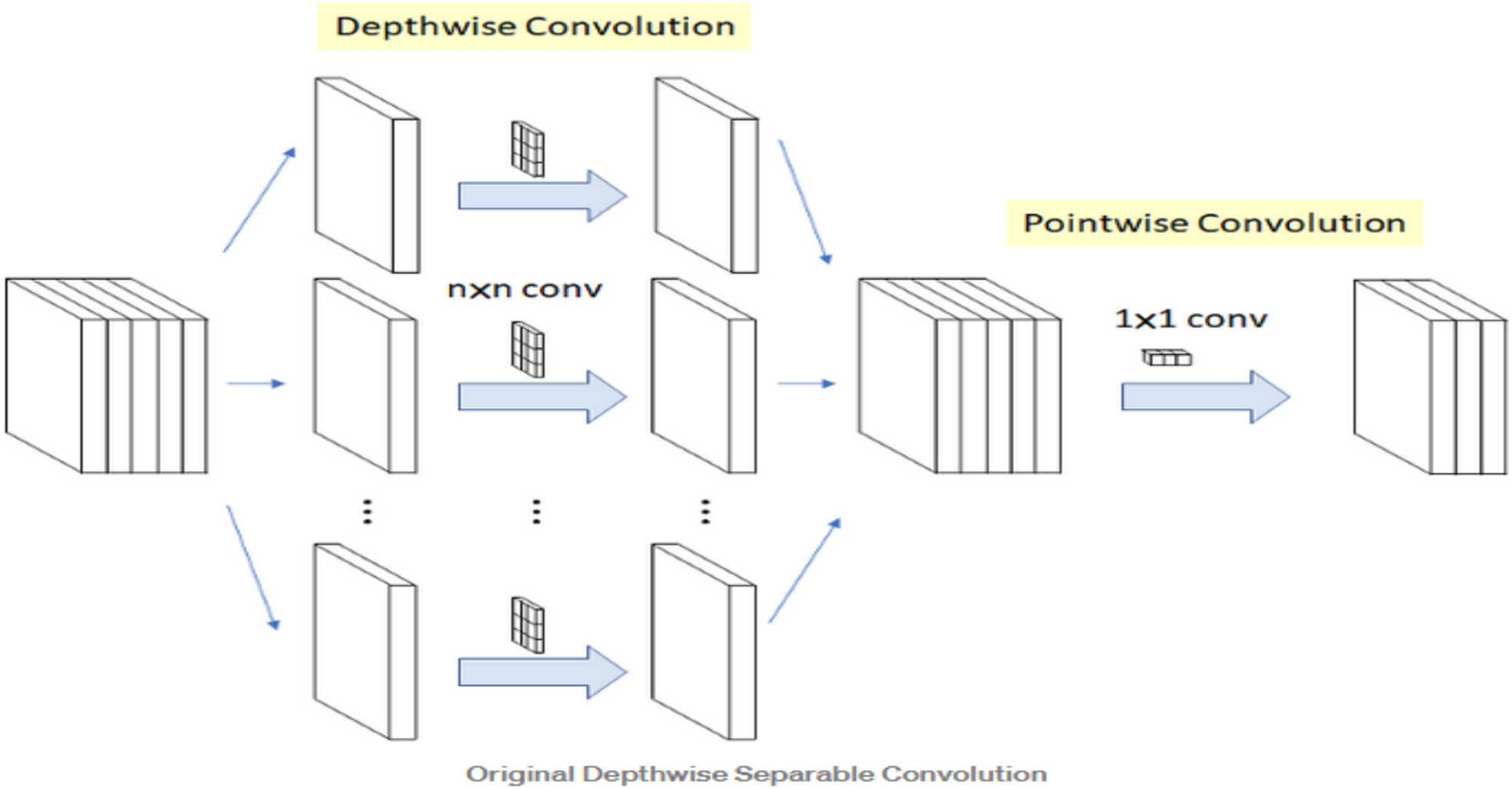
03. 1x1 필터를 이용하여 각각의 특징맵을 만든다. (N개의 필터)

* 케라스에서 SeparableConv2D를 이용하여 수행할 수 있다.

05 깊이별 분리 합성곱


1x1
1x1
...
1x1
N개의 필터
특징맵
특징맵
특징맵
필터1(3x3)
필터1(3x3)
필터1(3x3)
3 x 3 합성곱
3 x 3 합성곱
3 x 3 합성곱
...
채널 분리
N개 특징맵

05 깊이별 분리 합성곱



(참조) https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568