Batch Normalization

1. 높은 learning rate 사용 가능
2. 초기화를 신경쓰지 않아도 됨.
3. regularizer 역할. 경우에 따라 Dropout을 사용하지 않아도 됨.

Training Time에는 Mini-Batch에 포함된 데이터 포인트들로 평균과 분산을 구하고, Inference Time에는 안정적인 Inference를 위해 각 Mini-Batch의 평균과 분산을 이용하여 모든 데이터 포인트에 일괄적으로 적용되는 평균과 분산을 구한다. 선형 변환을 수행하는 β ,γ는 Learnable 파라미터.

SGD는 단순하고 효과적이지만 모델 매개 변수의 초기값뿐만 아니라 모델의 하이퍼파라미터, 특히 최적화에 사용되는 learning rate를 조정해야함.

학습에서 불안정화가 일어나는 이유
=> Internal Covariate Shift : 매 스텝마다 hidden later에 입력으로 들어오는 데이터의 분포가 달라지는 현상. Layer가 깊을수록 심화.
Covariate Shift(공변량 변화) : 이전 레이어의 파라미터 변화로 인해 현재 레이어의 입력 분포가 바뀌는 현상

Covariate Shift의 단점

saturated regime에 빠지고 vanishing gradient(대부분 activation으로 sigmoid를 사용하기 때문 => ReLU 함수를 사용하거나, 적은 learning rate 적용으로 문제 해결) 및 exploding 발생

Towards Reducing Internal Covariate Shift

Whitening : 데이터의 평균을 0, 그리고 공분산을 단위행렬로 갖는 정규분포 형태로 변환하는 기법으로 Decorrelation + Standardization. 보통 PCA를 이용해 데이터를 decorrelated 시킴.
training 효율을 높이기 위해서는 Internal Covariate Shift를 줄여야 함.
기존의 연구는 신경망의 parameter를 Activation의 값에 따라 바꾸면서 Whitening하는 방식 사용 => Gradient Descent의 효과를 줄이는 결과

단순히 whitening만 시키면 parameter를 계산하기 위한 Backpropagation과 무관하게 진행되기 때문에 특정 parameter가 계속 커지는 상태로 whitening이 진행될 수 있음.
=> gradient descent 효과가 감소하는 이유

Normalization via Mini Batch Statistics

- whitening의 문제점을 해결하기 위한 두 가지의 간소화 방법

1. layer의 입력과 출력을 동시에 whitening해주는 대신 각 scalar feature을 평균 0, 분산 1을 갖도록 독립적으로 normalize해줌. 이 Normalize는 feature가 decorrelated 되지

않아도 수렴 속도를 높여줌. 하지만 sigmoid의 입력을 normalize하는 것은 nonlinearity의 linear regime로 제한하게 되기 때문에 비선형성을 잃게 됨.

이를 해결하기 위해 normalize된 값을 scale, shift 해주는 학습 가능한 parameter 감마와 베타를 추가.

2. normalize는 전체 dataset을 처리하지만 SGD는 batch 단위로 data를 처리. 따라서 SGD를 이용하게 되면 전체 dataset을 normalize하는 것은 비현실적. normalize도 batch 단위로 해줌. 각각의 mini batch data로 layer의 입력을 normalize. 이 방법으로 normalization에 사용된 통계값은 모두 backpropagation에 사용 가능. 또한 normalize를 mini batch 단위로 처리하게 되면 한번에 처리하는 연산량도 줄어들게 되는 이점 존재.

==========

YOLO 기반의 광학 음악 인식 기술 및 가상현실 콘텐츠 제작 방법

OMR(Optical Music Recognition) : 광학 음악 인식. 사진 형태로 있는 악보를 디지털화 시키는 것. 스캔한 악보 정보를 이용하여 악보 분석 및 편집하거나 악보 정보를 토대로 한 오디오 재생 등으로 기존 편집 프로그램의 성능을 향상.

1966, Prerau : 오선 보표 자동 인식
1970s : 악보의 요소들을 감지하기 위한 이미지 분할 개념 도입
현재 : end-to-end 방식으로 사용

악상 기호 : 작곡가가 연주자에게 연주에 대한 정보를 제공하기 위한 모든 기호
모양 : 악상 기호의 종류 인식
위치 : 음 높이 인식

Objection Detection : Classification + Localization 동시 수행

1. 1-Stage-Detector : 물체의 범주 확률과 위치 정보를 1회 검출로 결과를 출력하기 때문에 속도 빠름. YOLO, SSD
2. 2-Stage-Detector : 물체의 범주 확률과 위치 정보를 순차적으로 검출하기 때문에 속도는 느리나 정확도가 더 높음. Fast R-CNN, Faster R-CNN

YOLO 기반의 악보 인식

- 음표가 어떤 음높이, 박자를 가지는지 해석하는 과정

1. 악보 종류 분류 : 첫 번째로 검출한 음악 객체들을 가져오고 음자리표와 Brace를 따로 분리
2. 보표 크기 설정 : 음자리표 또는 괄호의 중심점을 기준으로 설정
3. 오선 좌표 검출 : YOLO v5로 검출된 객체를 제외한 객체를 검출하기 위해 오선 제거
4. Hough transform을 이용한 특정 음악 객체 검출 및 분류 : Vision을 이용하여 검출 시도. 먼저 검출한 오선을 악보에서 제거한 다음 Hough transform을 이용해 악보 내의 사각형을 전부 검출하여 저장
5. 보표 크기 수정 : 음표와 쉼표가 어떤 보표에 해당하는지 정하기 위해 해당 보표의 오선의 최대, 최소 높이로 크기 수정
6. 보표 내, 외 음표 및 쉼표 개체 분류 : 탐색 범위 내에 포함된 음표 및 쉼표 객체들을 해당 탐색 범위를 가진 보표에 저장

7. 보표 크기 재수정 : Beam 객체 그룹 연결 및 온쉼표 2분 쉼표 분류와 보표별 가상의 선 생성을 위해 보표 크기를 음표 및 쉼표 객체 중에서 최대, 최소 높이로 변경
8. Beam, Flag와 음표 객체 연결 : 검출한 사각형 중에 1/8n 음표와 연결되어있는지 확인
9. 온쉼표와 2분 쉼표 분류 : 너비 우선 탐색으로 5선의 4번째 줄과 3번째 줄에 있는 사각형을 온쉼표와 2분 쉼표로 변경
10. 가상의 선 생성

VR 게임 : 실행을 위한 채보 디렉터리 처리 필요
리듬에 맞춰 누르는 게임의 민감한 부분인 노트들의 밀림 현상을 막기 위해 최적화 필요 => 타임라인을 이용하여 방지
리소스를 많이 차지하는 광원을 가지고 있는 Particle => Object Pooling 방식 사용