# Week8_논문스터디1

## 1. Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift

https://arxiv.org/pdf/1502.03167.pdf

### Abstract

Deep Neural Networks가 학습될 때 특정 레이어에서 input 분포가 계속 변해서 생기는 문제

- Learning rate 감소
- Careful parameter initialization
- Saturation nonlinearities.

⇒ 이러한 문제를 ***internal covariate shift*** 라고 부른다.

이 논문에서는 이러한 현상을 layer의 input을 normalizing해서 해결한다.

Normalization을 model architecture의 일부로 만들고, 각 training mini-batch에서 normalization을 수행한다.

Batch normalization의 장점

- 높은 learning rate
- Initialization을 좀 더 대충해도 됨
- Regulrizer처럼 동작하여, dropout의 필요성을 없앤다.

### 1. Introduction

SGD는 loss를 최소화하는 parameters를 찾아가는 최적화 알고리즘이다.

$$\Theta = \arg\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(\mathrm{x}_i, \Theta)$$

여기서 N은 전체 training set 개수이다.

SGD는 심플하면서도 효과적이지만, 세심한 hyper parameter 튜닝을 필요로한다. 학습 시 parameter의 작은 변화들은 모델을 더 깊게 만든다.

각 layer는 새로운 input distribution에 적응해야 해서 input distribution의 변화는 ‘covariate shift’를 야기한다.

이러한 문제는 ‘domain adaptation’로 해결하려는 시도는 있었지만, covariate shift는 전체 네트워크를 넘어 layer의 관점으로 확장한다.

Gradient Vanishing과 saturation problem 해결방안

: ReLu, small learning Rate ⇒ batch normalization

**Batch Normalization**

- Layer input 데이터의 평균과 분산을 고정시켜 데이터가 일정ㅇ한 분포를 가지게 한다
- Parameter의 초기값과 크기에 대한 gradient 의존도를 줄여 gradient flow에 좋은 영향을 준다.

⇒ 더 빠른 Learning rate 사용 가능, Dropout 필요성 감소

## 2. Towards Reducing Internal Covariate Shift

Internal Covariate Shift 를 줄이는 방법

⇒ Layer의 input 분포가 고정(Normalization)

각 layer input을 평균0, 분산1로 whitening 시키면, input의 분포가 고정되므로 internal covariate shift를 줄일 수 있다.

모든 step혹은 일정 간격 interval 마다 optimization parameter를 수정하는 방법⇒ 최적화 단계 마다 수정이 일어나면 Gradient descent step은 정규화가 업데이트 되어야 하는 방식으로 parameter를 업데이트 하려고 할 수 있는데, 이는 gradient step의 효과를 감소시킨다.

## 3. Normalization via Mini-Batch Statistics

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

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

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

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

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

**Algorithm 1:** Batch Normalizing Transform, applied to activation $x$ over a mini-batch.

Batchsize 에 맞게 평균, 분산을 구해 정규화시킨다.

연산 수행 시 안정성을 위해 추가하는 상수,

감마 : scaling

베타 : shift

입력 데이터가 N(0,1)로 정규화될 시 대부분의 입력에 대하여 매우 선형적으로 동작한다. 따라서 non-linearity 를 유지하기 위해서 감마, 베타를 이용해 scaling, shift를 한다.

## 3.1 Training and Inference with Batch-Normalized Networks

Test를 진행할때는 Train때 mini-batch마다 구했던 평균과 분산의 평균을 구해서 사용한다.

**Input:** Network $N$ with trainable parameters $\Theta$;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$

**Output:** Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$

1: $N_{\text{BN}}^{\text{tr}} \leftarrow N$ // Training BN network
2: **for** $k = 1 \ldots K$ **do**
3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)
4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
5: **end for**
6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen // parameters
8: **for** $k = 1 \ldots K$ **do**
9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size $m$, and average over them:

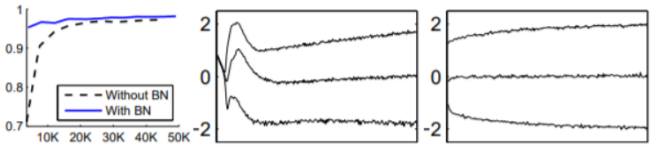
$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$
12: **end for**

**Algorithm 2:** Training a Batch-Normalized Network

## 4. Experiments

1. MNIST




(a) (b) Without BN (c) With BN

(a) MNIST data 에서 BN 사용 유무에 대한 convergence speed를 비교.

(b), (c) BN사용 유무에 대한 internal covariate shift 변화

⇒ BN을 사용했을 때 더 빨리 수렴하고 Accuracy가 높다는 것을 확인할 수 있다.

2. ImageNet Classification

| Model | Steps to 72.2% | Max accuracy |
|---|---|---|
| Inception | $31.0 \cdot 10^6$ | 72.2% |
| *BN-Baseline* | $13.3 \cdot 10^6$ | 72.7% |
| *BN-x5* | $2.1 \cdot 10^6$ | 73.0% |
| *BN-x30* | $2.7 \cdot 10^6$ | 74.8% |
| *BN-x5-Sigmoid* | | 69.8% |

BN을 사용하면 더 큰 lr에 대해서도 더 빠른 수렴 및 성능을 보장한다고 할 수 있다.

3. 앙상블 모델

| Model | Resolution | Crops | Models | Top-1 error | Top-5 error |
|---|---|---|---|---|---|
| GoogLeNet ensemble | 224 | 144 | 7 | - | 6.67% |
| Deep Image low-res | 256 | - | 1 | - | 7.96% |
| Deep Image high-res | 512 | - | 1 | 24.88 | 7.42% |
| Deep Image ensemble | variable | - | - | - | 5.98% |
| BN-Inception single crop | 224 | 1 | 1 | 25.2% | 7.82% |
| BN-Inception multicrop | 224 | 144 | 1 | 21.99% | 5.82% |
| BN-Inception ensemble | 224 | 144 | 6 | 20.1% | **4.9%*** |

Figure 4: *Batch-Normalized Inception comparison with previous state of the art on the provided validation set comprising 50000 images. *BN-Inception ensemble has reached 4.82% top-5 error on the 100000 images of the test set of the ImageNet as reported by the test server.*

BN을 사용한 결과가 좋다.

# YOLO 기반의 광학 음악 인식 기술 및 가상현실 콘텐츠 제작 방법

## 1. 서론

OMR(Optical Music Recognition)

: 사진 형태로 있는 악보를 디지털화 시키는 것

⇒ 악보 정로를 컴퓨터로 불러들이는다는 것에서 활용성 높음

컴퓨터 비전, 음악과 작곡 연습, 디지털 음악학, 디지털 라이브러리, 음악 정보 검색과 관련 높음 ⇒ 딥러닝의 새로운 접근방식 제공

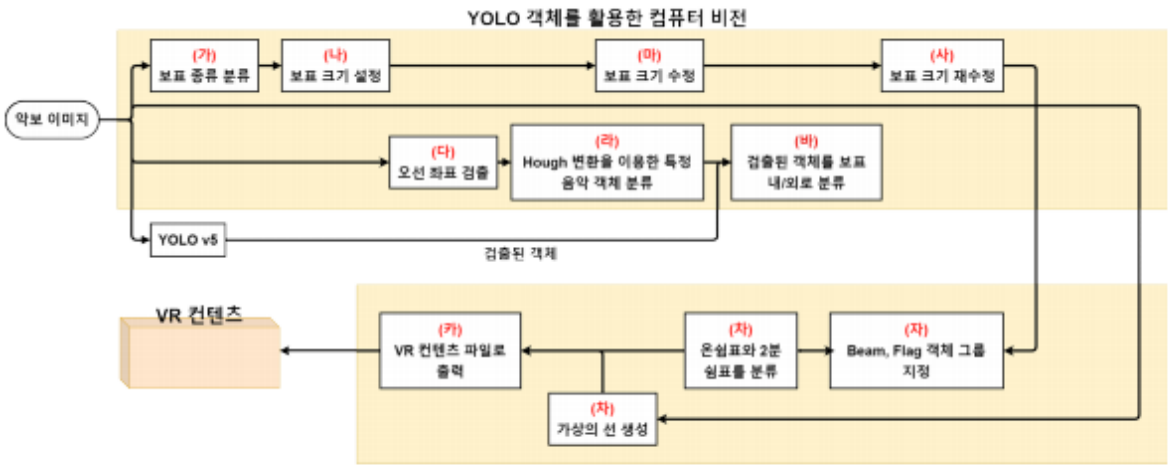



그림 1. YOLO V5로 활용한 음악 객체 추출 방법의 흐름도

Object Detection 기술의 발전 ⇒ OMR 발전

⇒ 이 논문에서는 VR의 콘텐츠 개발의 가능성 및 유효성에 연관하여 연구를 진행한다.

# 2. 본론

## 2-1. YOLO 기반의 악보 인식

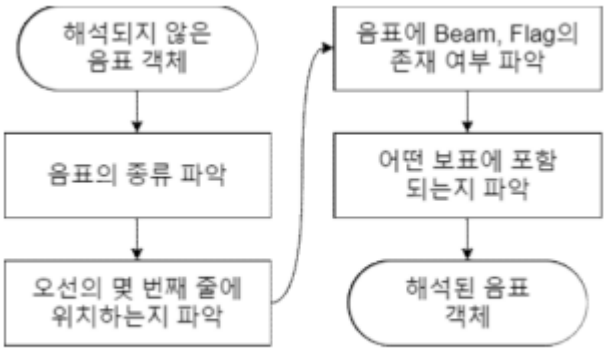



그림 2. 음표 해석 과정

악보에서 탐지해야 하는 요소들의 크기가 전체 이미지 크기에 비해 매우 작아 큰 이미지에서 학습해야 하고 필요한 연산량이 증가한다.

(이 논문에서는 epoch 1000, batch size 2, image size <2048 x 2048 x 3> 으로 학습 진행)

학습 후 학습한 YOLO 모델 및 Computer Vision 알고리즘들을 활용하여 악보의 요소들을 탐지한다.

앞선 그림1의 흐름도 상세 설명

1. 악보 종류 분류
    a. 작은보표(음자리표가 한 가지로 구성된 악보)
    b. 큰보표(음자리표가 두 가지로 구성된 악보)
2. 보표 크기 설정

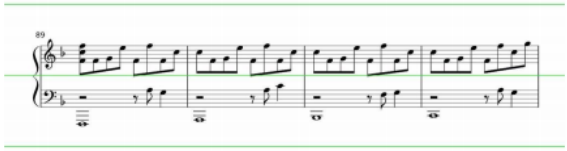


그림 5. 설정된 보표의 크기 범위

보표의  크기를 음자리표 또는 괄호의 중심점을 기준으로 설정

3. 오선 좌표 검출
    4.
    5. 보표의 크기 내에세 정렬시킨 반대 방향에서부터 수직으로 일정한 간격의 5개의 좌표 집합을 구한다.
4.

4. Hough transform 을 이용한 특정 음악 객체 검출 및 분류

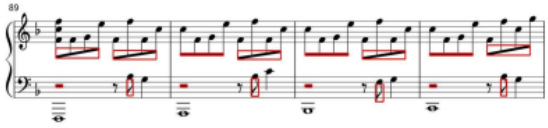


그림 7. 검출된 객체

YOLO v5로는 잘 검출되지 않아 Vision을 이용하여 검출 시도

5. 보표 크기 수정’

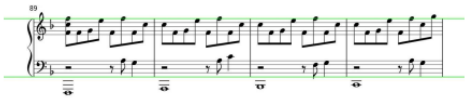


그림 8. 수정된 보표 크기

6. 보표 내, 외 음표 및 쉼표 개체 분류

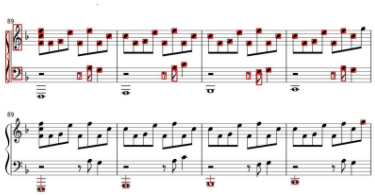


그림 9. 분류된 보표 내 / 외 음표 객체

탐색 범위 내에 포함된 음표 및 쉼표 객체들을 해당 탐색 범위를 가진 보표로 지정한다. 내부에 포함된 음표 및 쉼표 객체들은 저장 공간에서 제외한다. 저장 공간의 객체들은 보표들의 탐색 범위 외의 외부 객체들이다.

외부의 음표 및 쉼표 객체들이 어떤 보표에 포함되는지 확인한다.

확인 방법 : 외부의 음표 객체들의 중심을 기준으로 수직으로 검사를 해서 먼저 해당 보표 범위에 도달하면 해당 보표에 포함하는 원리

7. 보표 크기 재수정
8. Beam, Flage와 음표 객체 연결

- 검출한 사각형 중 1/8n 음표와 연결되어 있는지 확인하기 위해 4분음표로 검출한 음표와 사각형 간에 BFS을 이용해서 이어져 있는지를 확인
- 이어져 있으면 4분음표를 이어진 Beam
- Flage의 개수 n에 따라 1/8n 음표를 수정
- 수정한 사각형 객체는 사각형 객체 저장 공간에 제거

9. 온쉼표와 2분 쉼표 분류
10. 가상의 선 생성

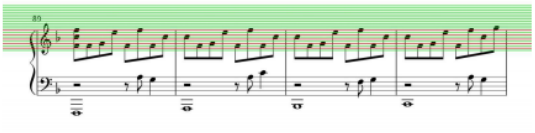


그림 13. 생성된 가상의 선 이미지

음표 객체들의 음높이를 파악하기 위해 기존의 5선의 간격을 측정하여 1번 줄과 5번 줄에서 부터 측정한 간격으로 좌표들을 탐색 범위 내에서 생성

11. 출력

가상의 선 또는 기존의 5선 좌표와 음표 객체의 중심 좌표의 차가 가장 적으면 해당 음표 객체는 해당 선에 위치하기 때문에 음높이를 파악할 수 있다.

보표를 높이 순으로 정렬한 뒤, 저장된 음표 및 쉼표 객체를 가로축에 놓은 순서대로 정렬한다.

음표 객체의 박자와 음높이, 쉼표 객체이 박자를 정렬된 순서대로 인코딩하여 출력한다.