

DeepPose : Human Pose Estimation Via Deep Neural Networks

Abstract

해당 논문에서 처음으로 Deep Neural Network를 사용해서 사람의 관절을 예측하는 모델을 제안하였습니다. DNN을 계단식으로 구성하여 점진적으로 오차를 수정해가면 4가지 벤치마크에서 SOTA 성능을 보였다고 주장합니다.

Introduction



Figure 1. Besides extreme variability in articulations, many of the joints are barely visible. We can guess the location of the right arm in the left image only because we see the rest of the pose and anticipate the motion or activity of the person. Similarly, the left body half of the person on the right is not visible at all. These are examples of the need for *holistic reasoning*. We believe that DNNs can naturally provide such type of reasoning.

사람의 관절을 찾는 문제는 컴퓨터 비전 분야에서 관심을 갖고 있습니다. 하지만 위의 그림과 같이 많이 변할 수 있는 관절, 거의 변하지 않는 관절, 가려진 부분, 그리고 개별 관절들간의 상호작용이 주된 문제로 제기되고 있습니다.

Human pose estimation 분야의 주된 목표는 사람이 취할 수 있는 모든 자세 공간을 찾는 것 이었습니다. 하지만 초기 연구들의 경우 특정 관절이나 자세에 대한 연구가 활발했고, 모든 관절에 대한 상호작용을 하지 못했다는 한계가 존재하였습니다. 이에 해당 논문에서는 최근에 발달한 DNN 구조를 활용하여 모든 관절들이 상호작용하는 holistic human pose estimation을 제안합니다.

단순히 7계층을 갖는 신경망을 통해서 각각의 관절을 Regression 형식으로 예측하게 됩니다. 이렇게 정식화된 모델을 사용함으로써, 이미지 전체의 문맥을 파악할 수 있으며, 기존의 모델들 보다 매우 간단하게 모델을 설계할 수 있습니다. 추가로 Cascade 방식을 체택하여 점진적으로 해상들을 높혀 정확도를 올리는 방식을 사용하였다고 제안하고 있습니다.

이러한 방식을 통해서 가지 벤치마크에서 높은 성능을 달성했음을 제안하고 있습니다.

Deep Learning Model for Pose Estimation

본 연구에는 아래와 수식을 통해서 각 (x, y) 좌표를 수식화 하고 있습니다.

$$y = (\dots, y_i^T, \dots)^T$$

그리고 해당 y_i 는 (x, y) 로 2개의 좌표값을 가지고 있습니다. 하지만 단순히 이미지 내에서 절대 좌표를 활용하게 되면 다른 이미지와의 관계를 파악하기 불리합니다. 그래서 본 논문에서는 이미지 내 사람을 감싸는 박스를 함께 제공하게 됩니다. 그리고 박스 내 존재하는 각 관절의 좌표를 정규화하게 됩니다. 이를 통해서 모든 좌표값은 0 ~ 1 사이의 값을 갖도록 됩니다. 이를 통해서 모든 이미지들은 상대적 좌표를 활용하기에 모델이 관절 위치 패턴을 보다 쉽게 파악할 수 있도록 유도합니다. 이를 수식화 하면 아래와 같이 나타낼 수 있습니다.

$$N(y; b) = (\dots, N(y_i; b)^T, \dots)^T$$

$$N(y_i; b) = \begin{pmatrix} \frac{1}{b_w} & 0 \\ 0 & \frac{1}{b_h} \end{pmatrix} (y_i - b_c)$$

위의 수식을 보면 간단하게 박스의 중심 좌표인 b_c 좌표를 빼고, 박스의 높이, 너비로 나눠주어 정규화를 하게 됩니다. (행렬의 곱으로 좌표계의 기저 벡터 값의 크기를 줄이는 느낌으로 이해도 가능하다)

Pose Estimation as DNN-based Regression

본 논문에서는 이를 간단한 regression 문제로 치환하여 해결하고자 합니다. 그래서 간단하게 DNN 함수를 도입하고 이를 통해서 최종 결과가 2k 만큼 나오도록 합니다. 모든 값들은 0 ~ 1 사이로 매핑되어 나오고, 추가로 역정규화 과정을 통해서 최종 좌표값을 얻을 수 있게 됩니다. 이를 수식으로 표현하면 아래와 같이 표현할 수 있습니다.

$$y^* = N^{-1}(\psi(N(x); \theta))$$

본 논문에서는 당시 높은 성능을 보인 AlexNet을 활용하였다고 합니다. 이미지의 경우 3채널을 활용하였다고 합니다.

그리고 모델을 학습하는 경우 모든 이미지마다 박스 크기가 다르고 절대 좌표가 다르기 때문에 상대 좌표를 활용하였다고 합니다. 그래서 우선 손실을 구하기 전에 GT를 상대 좌표로 변환해주어야 합니다. 아래와 같이 실제 정답 또한 box에 맞춰 상대 좌표로 변환해주게 됩니다.

$$D_N = \{(N(x), N(y)) \mid (x, y) \in D\}$$

그리고 이렇게 상대 좌표로 변환되었다면, 각 좌표가 실제 좌표와 가까워지도록 하기 위해서 $L_2 - norm$ 을 최소화 하는 손실 함수를 설정하였다고 합니다.

$$\theta^* = \arg \min_{\theta} \sum_{(x,y) \in D_N} \sum_{i=1}^k \|y_i - \psi_i(x; \theta)\|_2^2$$

학습의 경우에는 backpropagation을 활용하여 가중치를 업데이트 하였다고합니다.

Cascade of Pose Regressors

하지만 본 논문에서는 다음과 같이 간단하게 전체 이미지를 입력으로 학습을 하게 되면 각 관절의 좌표가 디테일하지 않다고 주장합니다. 그래서 각 관절에 해당하는 좌표를 기준으로 crop image를 생성하여 좌표의 오차를 학습하는 구조를 구현합니다. 이러한 방식을 Cascade 방식이라고 합니다. 총 S 단계를 거쳐서 보정이 들어가고 각각의 모델의 파라미터는 θ_s 로 표현하였습니다. 그리고 각 관절이 포함된 crop image의 박스는 아래와 같이 정의하였습니다. $diam$ 과 σ 는 이미지마다 최적화된 값이라고 생각하면 될것같습니다.

$$yi : bi(y;) = (yi, \sigma diam(y), \sigma diam(y))$$

그래서 결국 stage1의 경우 우리가 위에서 봤던 수식을 그대로 따르고 2단계 이후로는 각각의 관절을 crop한 데이터를 다루기 때문에 아래와 같은 수식으로 표현이 가능하게 됩니다.

$$\text{Stage } s : \quad y_i^s \leftarrow y_i^{(s-1)} + N^{-1}\left(\psi_i(N(x; b); \theta_s); b\right)$$

$$b_i^s \leftarrow \left(y_i^s, \sigma \text{diam}(y^s), \sigma \text{diam}(y^s)\right)$$

즉, 각각의 좌표값은 이전 좌표값을 기반으로 오차 만큼의 값을 더해서 각각의 관절 좌표를 보정하게 됩니다.

손실 함수를 구하기전에 stage1과 stageN ($N \geq 2$) 의 중요한 차이가 있습니다. 이는 모든 이미지의 박스의 크기가 다르다는 것입니다. 그래서 각각의 박스로 정규화가 필요하게 됩니다. 그리고 본 논문에서는 $y_i^s - y_i^{(s-1)}$ 값들을 사용하여 모든 관절의 오차를 통해서 평균과 분산을 통해서 특정 정규분포를 설정하게 됩니다. 그리고 이러한 분포에서 값들을 샘플링하게 되어 실제 정답에 더해

주게 됩니다. 이를 통해서 실제 없는 데이터지만 box 자체를 조금씩 이동시켜 데이터 증강 효과를 얻었다고 합니다. 이를 수식으로 표현하면 아래와 같이 나타낼 수 있습니다.

$$D_A^s = \left\{ (N(x; b), N(y_i; b)) \mid (x, y_i) \sim D, \delta \sim \mathcal{N}_i^{(s-1)}, b = (y_i + \delta, \sigma \operatorname{diam}(y)) \right\}$$

$$\theta_s = \arg \min_{\theta} \sum_{(x, y_i) \in D_A^s} \| y_i - \psi_i(x; \theta) \|_2^2$$

Experiments

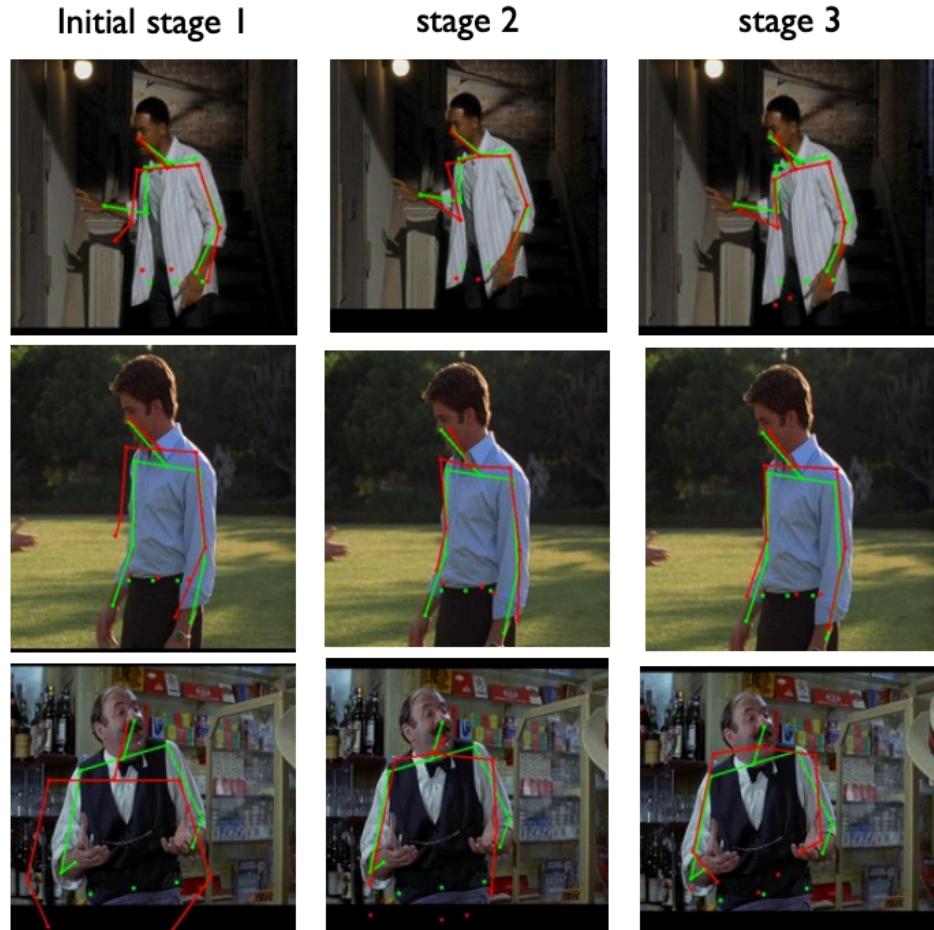


Figure 6. Predicted poses in red and ground truth poses in green for the first three stages of a cascade for three examples.

Method	Arm		Leg		Ave.
	Upper	Lower	Upper	Lower	
DeepPose	0.8	0.75	0.71	0.5	0.69
Pishchulin [18]	0.80	0.70	0.59	0.37	0.62
Johnson et al. [13]	0.75	0.67	0.67	0.46	0.64
Yang et al. [27]	0.69	0.64	0.55	0.35	0.56

Table 2. Percentage of Correct Parts (PCP) at 0.5 on Image Parse dataset for DeepPose as well as two state-of-art approaches on Image Parse dataset. Results obtained from [18].

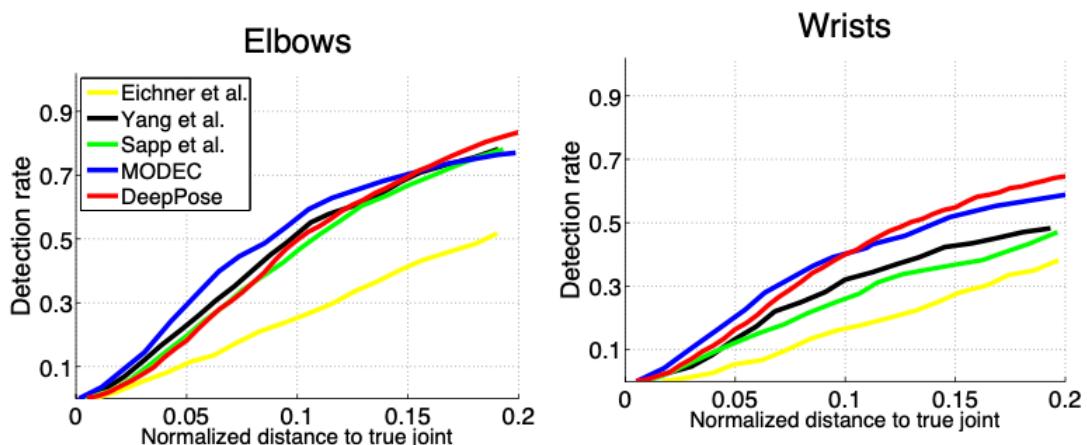


Figure 7. Percentage of detected joints (PDJ) on Buffy dataset for two joints: elbow and wrist. The models have been trained on FLIC. We compare DeepPose, after two cascade stages, with four other approaches.

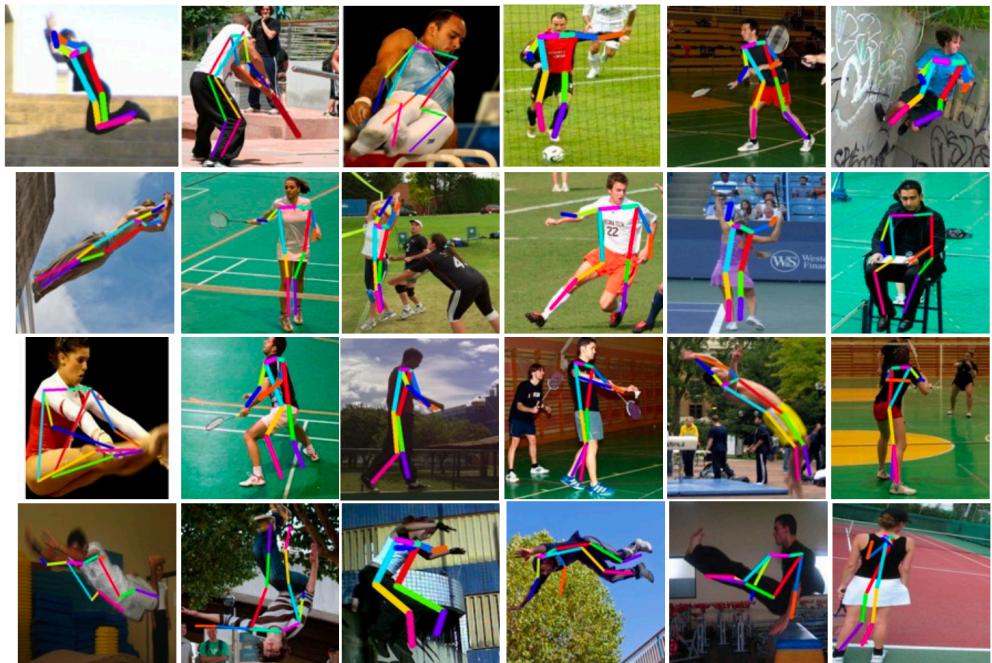
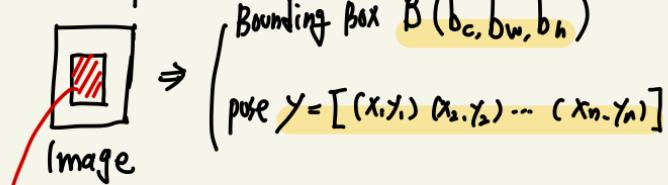


Figure 8. Visualization of pose results on images from LSP. Each pose is represented as a stick figure, inferred from predicted joints. Different limbs in the same image are colored differently, same limb across different images has the same color.

다음은 이 모든 과정을 제가 손으로 정리한 내용들입니다.

Data Input.



$$N(y_i; b) = \begin{pmatrix} 1/b_w & 0 \\ 0 & 1/b_h \end{pmatrix} (y_i - b_c)$$

상향변환. b_c 를 0점으로 이동

$$\Rightarrow N(y; b) = [\dots, N(y_i; b)^T, \dots]^T$$

($y_i = (x_i, y_i)$ 은 0~1 사이의 상대좌표로 매핑된다.
 \Rightarrow 다른 DNN의 입력으로 사용할 수 있다.)

$$N(y; b) \Rightarrow \boxed{\text{DNN}} \Rightarrow \begin{matrix} \text{2k} \\ \text{현재 상대좌표.} \end{matrix}$$

Model에서 y 는 $2k$ vector를
 $\Psi(x; \theta) \in \mathbb{R}^{2k}$ 로 표기
 ↳ regression Model
 ↳ 절대좌표로 변환.

$$y^* = N^{-1}(\Psi(N(x); \theta))$$

최종 Model에서 예측한 pose 좌표.

즉, 이를 통해서 전처리 이미지에서 pose를 예측 >

$$\begin{matrix} 220 \\ \text{...} \end{matrix} \rightarrow N(y; b) \rightarrow \boxed{\text{DNN}} \rightarrow \Psi(N(x; b); \theta) \xrightarrow{\text{Inverse normalize}} y^* = N^{-1}(\Psi(N(x); \theta))$$

Train 데이터는 Box \hat{y} 가 각의 정답과 표준偏差를 베이스로 한다.

$$D_N = \left\{ N(x), N(y) \mid (x, y) \in D \right\}$$

예측 GT

두 좌표의 차를 minimize하기 위해 L_2 -Norm을 minimize

$$\arg \min_{\theta} \sum_{(x, y) \in D} \sum_{i=1}^k \left\| \hat{y}_i - \psi_i(x_i; \theta) \right\|_2^2$$

이미지당 손실 (loss)
모든 (image) loss

ex) $\hat{y}_i = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.8 \\ 0.9 \end{bmatrix}$ $\psi(x_i; \theta) = \begin{bmatrix} 0.2 \\ 0.1 \\ 0.3 \\ 0.5 \\ 0.9 \\ 0.7 \end{bmatrix}$

$$\text{loss} \Rightarrow (0.2)^2 + (0.1)^2 + (0.1)^2 + (0.1)^2 + (0.1)^2 + 0^2 \\ = 0.04 + 0.01 + 0.01 + 0.01 + 0.01 + 0 \Rightarrow 0.06$$

\therefore 마지막 pose는 거의 맞았지만 10%가 틀려들어간다.

Stage 1.

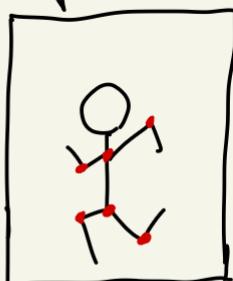
단순히 $220 \times 220 \times 3$ 을 사용하면 pose의 detail이 떨어짐.

그래서 각 관절을 crop한 image를 사용하여 detail↑

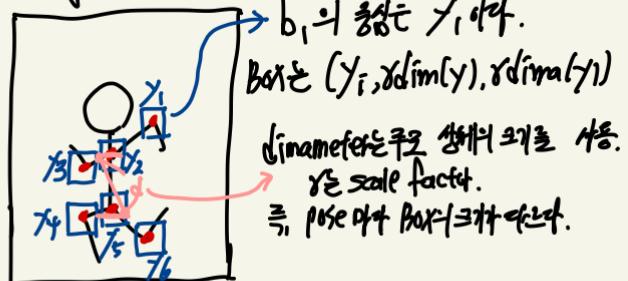
$$S = \{S_0, S_1, \dots, S_k\} \text{ 중 } k \text{ 가지 반복}$$

\rightarrow 각 step마다 θ_s 로 초기화.

Stage 1의 결과



Stage 2 입력



Input : 0번지 & 1번 Box

Input : k번지 이미지 & k번지 Box

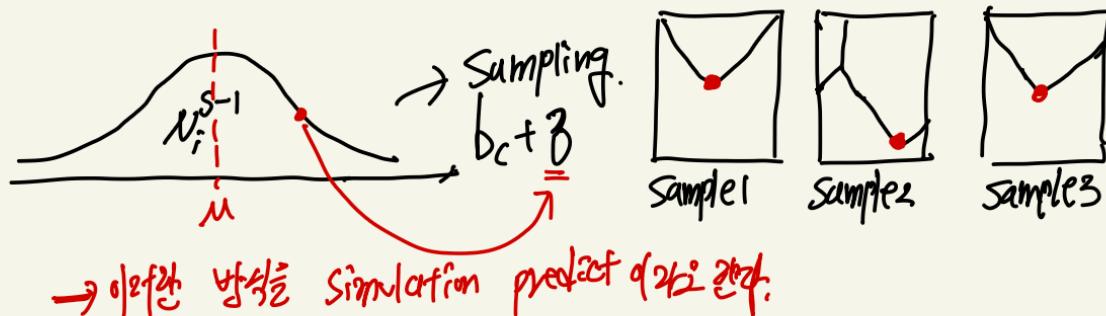
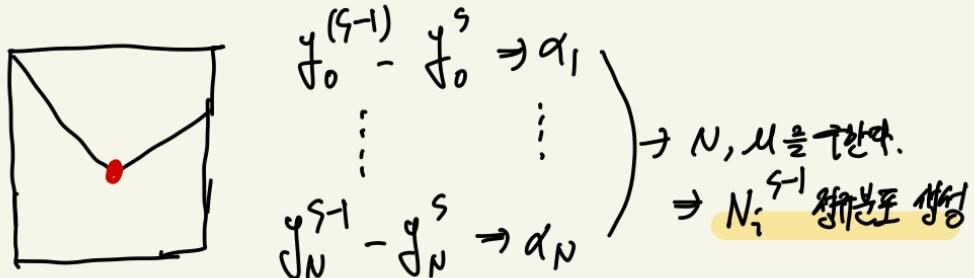
Stage N ($N \geq 2$)

$$y_i^s \leftarrow \underbrace{y_i^{(s-1)}}_{\text{이전값}} + \underbrace{\tilde{N}^{-1}(\psi_i(N(x; b); \theta_s); b)}_{\text{현재 Model에서 나온 값}} \quad] \text{ 좌표 업데이트}$$

for : $b = b_i^{s-1} \Rightarrow$ Box는 이전 결과기반.

$$b_i^s \leftarrow (y_i^s, x_diam(y^s), y_diam(y^s)) \Rightarrow \text{이제 } b \text{ update. }] \text{ Box update}$$

한정된 stage가 몇 개지만 stage s 는 단계마다 서로 다른
Bounding Box를 기반으로 학습한다.



$$D_A^S = \left\{ (N(x; b), N(y; b)) \mid (x, y_i) \sim D, f \sim N_i^{S-1}, b = (y_i + \delta, \sigma \text{diam}(y)) \right\}$$

augmentation dataset

이후 Loss function을 뜯어내거나 정의한다.

$$\theta_S = \underset{\theta}{\operatorname{argmin}} \sum_{(x, y) \in D_A^S} \|y_i - \psi_i(x; \theta)\|_2^2$$

간단 정리

본 논문은 사람의 관절을 예측하는 문제를 간단한 CNN 모델을 통해서 구현하였습니다. 추가로 좌표 값의 정확도를 올리기 위해서 각 관절을 기준으로 crop image들을 cascade 방식으로 학습하여 각 좌표의 정확도를 향상시켰습니다.