Lifting from the Deep: Convolutional 3D Pose Estimation from a Single Image Supplementary Material

Denis Tome
University College London
D.Tome@cs.ucl.ac.uk

Chris Russell
The Turing Institute
crussell@turing.ac.uk

Lourdes Agapito
University College London
l.agapito@cs.ucl.ac.uk

http://visual.cs.ucl.ac.uk/pubs/liftingFromTheDeep

1. Computing derivatives for back-propagation through our lifted model

As discussed by the Convolution Pose Machine paper [1] recurrent like architectures such as ours have problems with *vanishing gradients* and for effective training they require an additional loss function to be defined for each layer, that independently drives each individual layer to return correct predictions regardless of how this information is used in subsequent layers.

Before we give the derivation of the gradients it should be emphasized that it is entirely possible to train the network without using them – in fact similar results can be obtained by only using the 3D lifting for the forward pass, and not back-propagating the lifting derivatives through the rest of the network. As the additional layers make use of custom Python-based derivatives rather than an efficient implementation, for computational reasons it might preferable to avoid this step. Nonetheless for completeness we include the derivatives.

There are two reasons the gradients are unneeded: Our lifting 3D model we use makes its best predictions when the 2D predictions of the same layer are closest to ground truth, and this is a constraint naturally enforced by the objective of equation (8) of the main paper. Further, as with Convolutional Pose Machines [1] our architecture suffers from problems with vanishing gradients. To overcome this Wei *et al*. [1] defined an objective at each layer, which acted to locally strengthen the gradients. However, a side effect of this multi-stage objective is that most of the effects of back-propagation happen locally and gradients back-propagated from other layers have little effect on the learning. This makes subtle interactions between layers less influential, and forces the learning process to concentrate on simply making accurate 2D predictions in each layer.

We first give the results for computing the gradients of sparse predicted locations $\hat{Y}$ from Y (see section 5 of the main paper), before discussing the gradients induced on the confidence maps by these sparse locations.

1.1. Landmark Gradients

In the interests of readability we neglect the use of indices to indicate stages, the reader should assume that all variables are taken from the same stage. Similarly, when dealing with a mixture of Gaussians, as we are only interested in computing a sub-gradient, the reader should assume that the best model has already been selected in the forward pass and we are computing gradients using only this model.

Recall (section 5 of main body of paper) that the mapping from the initial landmarks Y to the projected 3D proposals $\hat{Y}$ is given by

$$\hat{Y} = \Pi\mathbf{R}(\mu + a \cdot \mathbf{e}) \tag{1}$$

where

$$\mathbf{R}, a = \underset{\{\mathbf{R}^* \in \mathcal{R}, a^* \in \mathbb{R}^J\}}{\arg\min} ||Y - \Pi\mathbf{R}^*(\mu + a^* \cdot \mathbf{e})||_2^2 + (\sigma \cdot a^*)^2 \tag{2}$$

where $\mathcal{R}$ is a discrete set of rotations we exhaustively minimize over, and J is the number of bases in $\mathbf{e}$. Owing to the use of discrete rotations, this mapping from Y to $\hat{Y}$ is a piecewise smooth approximation of the smooth function defined over a continious $\mathcal{R}$, and sub-gradients can be induced by fixing R to its current state. Hence:

$$\frac{d\hat{Y}}{da} = \Pi\mathbf{R}\mathbf{e} \tag{3}$$

For the remainder of the section, and to compact notation we will write E for the matrix of size $2L \times J$ (L the number of landmark points and J being the number of bases in $\mathbf{e}$) formed by unwrapping tensor $\Pi\mathbf{R}\mathbf{e}$. Similarly, we will unwrap the $2 \times L$ matrices Y and $\hat{Y}$ and write them as y and $\hat{y}$. We also write p for the vector representing the unwrapped set of 2D landmark positions $\Pi\mathbf{R}\mu$.

We will use $[y, 0]$ for the vector formed by vector y followed by J zeros, and $\bar{E}$ for the matrix of size $(2L + J) \times J$ formed by concatenating E with the matrix that has values σ along the diagonals and zero everywhere else. We can rewrite equation (3) in its new notation as:

$$\frac{d\hat{y}}{da} = E \tag{4}$$

and given R, we can rewrite equation (2) as

$$a = \underset{a* \in \mathbb{R}^J}{\arg\min} \, |[y, 0] - [p, 0] - a^* \bar{E}||_2^2 \tag{5}$$

or

$$a = [(y - p), 0]\bar{E}^\dagger \tag{6}$$

with $\bar{E}^\dagger$ continuing to represent the pseudo-inverse of $\bar{E}$. Hence

$$\frac{da}{d[y, 0]} = \bar{E}^\dagger \tag{7}$$

and

$$\frac{d\hat{y}}{dy} = \frac{d\hat{y}}{da}\frac{da}{dy} = E\,\bar{E}^t \tag{8}$$

where $\bar{E}^t$ is the truncation of $\bar{E}^\dagger$.

1.2. Mapping belief gradients to coordinate transform

The coordinates of each predicted landmark $\hat{Y_p}$ induce a Gaussian in the belief map $\hat{b}_p$. So a change in the x component of $\hat{Y_p}$ induces an update which is equivalent to a difference of Gaussians.

$$\frac{d\hat{b}_p}{d\hat{Y}_p^x} \approx \frac{G(\hat{Y_p}^{(x)} + \delta_x) - G(\hat{Y_p}^{(x)} - \delta_x)}{2\delta_x} \tag{9}$$

and the same for the y component as well. For computational purposes we take δ_x as one pixel. As such, an induced gradient on the projected belief map near the predicted location $\hat{Y}$ $\hat{b}_p$ induces an updating of $\hat{Y}$ that is propagated through to Y using the sub-gradients described in equation (8).

Updating B Writing B for the the set of all b_p, and assuming Y_p is not in the right location, i.e. given updates $\Delta\hat{B}$ on $\hat{B}$ such that

$$\Delta\hat{B}.\frac{d\hat{B}}{d\hat{Y}}\frac{d\hat{Y}}{dY_p} \neq 0,$$

any update of b in which we decrease the belief at $b^p_{Y_p}$ and increase anywhere else is a valid sub-gradient. We choose as a sensible update a negative step at b_p of magnitude $m = k||\Delta\hat{B}.\frac{d\hat{B}}{d\hat{Y}}\frac{d\hat{Y}}{dY_p}||$ and a positive update for each element Y of of B_p of the magnitude $m \cdot N(Y, \sigma^2)$ in the quadrant of a Gaussian of the same width used to generate $\hat{b}$ (i.e. $\sigma = 1$ see section 5.6 of main paper) and with the same direction as $\Delta\hat{B}.\frac{d\hat{B}}{d\hat{Y}}\frac{d\hat{Y}}{dY_p}$ in each x and y coordinate.

References

[1] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. *arXiv preprint arXiv:1602.00134*, 2016. 1