Supplemental Material: ElePose: Unsupervised 3D Human Pose Estimation by Predicting Camera Elevation and Learning Normalizing Flows on 2D Poses

Bastian Wandt, James J. Little, and Helge Rhodin
University of British Columbia
{wandt,little,rhodin}@cs.ubc.ca

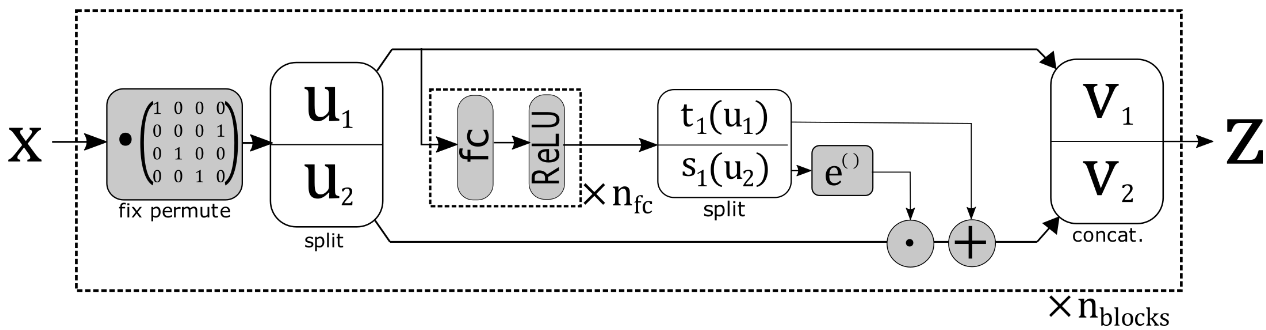

Figure 1. The normalizing flow consists of multiple consecutive coupling blocks. Each block performs a random permutation of the input vector and splits it into two parts. The upper part is used to predict an element-wise scale and a translation which deform the lower part. After deformation the upper part is concatenated with the transformed lower part.

1. Normalizing Flow Architecture

Informally, a normalizing flow is a tool to efficiently map distributions back and forth between two spaces. It applies to density estimation and also serves well as a generative model.

Let $\mathcal{Z} \in \mathbb{R}^N$ be a known distribution (in our case a normal distribution) and g be an invertible function $g(\mathbf{z}) = \mathbf{x}$, with $\mathbf{x} \in \mathbb{R}^N$ as a vector representing the joints of a human pose[1]. With the change of variables formula the probability density function of $\mathbf{x}$ is computed as

$$p_{\mathcal{X}}(\mathbf{x}) = p_{\mathcal{Z}}(f(\mathbf{x})) \left| \det\left(\frac{\partial f}{\partial \mathbf{x}}\right) \right|, \tag{1}$$

where f is the inverse of g and $\frac{\partial f}{\partial \mathbf{x}}$ is the Jacobian of f. That means given an invertible function f the density of a 2D pose $\mathbf{x}$ can be calculated by the product of the density of its projection $f(\mathbf{x})$ with the respective Jacobian determinant. In our case f is the trainable neural network proposed in [1]. It consists of multiple consecutive *affine coupling blocks*. As shown in Fig. 1 each coupling block splits the input vector into two parts, u_1 and u_2. In the forward pass, a scale s and a translation t is computed from u_1 and applied to u_2 such that

$$v_2 = \exp(s(u_1))u_2 + t(u_1) \quad \text{and} \quad v_1 = u_1. \tag{2}$$

The backward path is defined by

$$u_1 = v_1 \quad \text{and} \quad u_2 = (v_2 - t(v_1)) \exp(-s(v_1)). \tag{3}$$

The benefit of this formulation is the tractable computation of the determinant $\det(\frac{\partial f}{\partial \mathbf{x}})$. The Jacobian is given by

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{pmatrix} \boldsymbol{I}_N & \mathbf{0} \\ \frac{\partial v_2}{\partial u_1^T} & \operatorname{diag}(\exp(s(u_1))) \end{pmatrix}, \tag{4}$$

where $\operatorname{diag}(\cdot)$ is a diagonal matrix. Since we are only interested in the determinant of the Jacobian, it simplifies to

$$\det\left(\frac{\partial f}{\partial \mathbf{x}}\right) = \exp\left(\sum_j s(u_1)_j\right). \tag{5}$$

Note that the Jacobian of f does not require computing the Jacobian of s and t. That means s and t can be arbitrarily complex. Since u_1 remains unchanged in one coupling layer, the input vector to each coupling layer is randomly permuted.

[1] In our case, this is either the 2D pose vector $\mathbf{x}$ or its image in the PCA subspace.

References

[1] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 1