Learning Feature Pyramids for Human Pose Estimation

Wei Yang[1] Shuang Li[1] Wanli Ouyang[1,2] Hongsheng Li[1] Xiaogang Wang[1]
[1] Department of Electronic Engineering, The Chinese University of Hong Kong
[2] School of Electrical and Information Engineering, The University of Sydney
{wyang, sli, wlouyang, hsli, xgwang}@ee.cuhk.edu.hk

Abstract

Articulated human pose estimation is a fundamental yet challenging task in computer vision. The difficulty is particularly pronounced in scale variations of human body parts when camera view changes or severe foreshortening happens. Although pyramid methods are widely used to handle scale changes at inference time, learning feature pyramids in deep convolutional neural networks (DCNNs) is still not well explored. In this work, we design a Pyramid Residual Module (PRMs) to enhance the invariance in scales of DCNNs. Given input features, the PRMs learn convolutional filters on various scales of input features, which are obtained with different subsampling ratios in a multi-branch network. Moreover, we observe that it is inappropriate to adopt existing methods to initialize the weights of multi-branch networks, which achieve superior performance than plain networks in many tasks recently. Therefore, we provide theoretic derivation to extend the current weight initialization scheme to multi-branch network structures. We investigate our method on two standard benchmarks for human pose estimation. Our approach obtains state-of-the-art results on both benchmarks. Code is available at https://github.com/bearpaw/PyraNet.

1. Introduction

Localizing body parts for human body is a fundamental yet challenging task in computer vision, and it serves as an important basis for high-level vision tasks, *e.g.*, activity recognition [60, 54], clothing parsing [57, 58, 36], human re-identification [65], and human-computer interaction. Achieving accurate localization, however, is difficult due to the highly articulated human body limbs, occlusion, change of viewpoint, and foreshortening.

Significant progress on human pose estimation has been achieved by deep convolutional neural networks (DCNNs) [53, 52, 11, 51, 42, 55, 39]. In these methods, the DCNNs learn body part detectors from images warped to the similar scale based on human body size. At inference time, testing images should also be warped to the same scale as that for training images.

(a) (b) (c)

Figure 1. Our predictions on the LSP dataset [31]. When images are warped to approximately the same scale, scales of different body parts may still be inconsistent due to camera view change and foreshortening. In (a), the scale of *hand* and *head* are larger than that of *foot*. In (b), the scale of *foot* is larger than that of *head*.

Although the right scale of the full human body is provided, scales for body parts may still be inconsistent due to inter-personal body shape variations and foreshortening caused by viewpoint change and body articulation. It results in difficulty for body part detectors to localize body parts. For example, severe foreshortening is present in Figure 1. When the images are warped to the same size according to human body scale, the *hand* in Figure 1 (a) has a larger scale than that in Figure 1 (b). Therefore, the *hand* detector that can detect the *hand* in Figure 1 (a) might not be able to detect the *hand* in Figure 1 (b) reliably. In DCNNs, this problem from scale change happens not only for high-level semantics in deeper layers, but also exists for low-level features in shallower layers.

To enhance the robustness of DCNNs against scale variations of visual patterns, we design a *Pyramid Residual Module* to explicitly learn convolutional filters for building feature pyramids. Given input features, the Pyramid Residual Module obtains features of different scales via subsampling with different ratios. Then convolution is used to learn filters for features in different scales. The filtered features are upsampled to the same resolution and are summed together for the following processing. This Pyramid Residual Module can be used as building blocks in DCNNs for learning

feature pyramids at different levels of the network.

There is a trend of designing networks with branches, *e.g.*, Inception models [47, 30, 48, 46] and ResNets [25, 26] for classification, ASPP-nets [9] for semantic segmentation, convolutional pose machines [55] and stacked hourglass networks [39] for human pose estimation, in which the input of a layer is from multiple other layers or the output of a layer is used by many other layers. Our pyramid residual module also has branches. We observe that the existing weight initialization scheme, *e.g.*, MSR [24] and Xavier [21] methods, are not proper for layers with branches. Therefore, we extend the current weight initialization scheme and provide theoretic derivation to show that the initialization of network parameters should take the number of branches into consideration. We also show another issue in the residual unit [26], where the variance of output of the residual unit accumulates as the depth increases. The problem is caused by the identity mapping.

Since Hourglass network, also called conv-deconv structure, is an effective structure for pose estimation [39], object detection [34], and pixel level tasks [10], we use it as the basic structure in experiments. We observe a problem of using residual unit for Hourglass: when outputs of two residual units are summed up, the output variance is approximately doubled, which causes difficulty in optimization. We propose a simple but efficient way with negligible additional parameters to solve this problem.

The main contributions are three folds:

- We propose a *Pyramid Residual Module*, which enhances the invariance in scales of deep models by learning feature pyramids in DCNNs with only a small increase of complexity.
- We identify the problem for initializing DCNNs including layers with multiple input or output branches. A weight initialization scheme is then provided, which can be used for many network structures including inception models [47, 30, 48, 46] and ResNets [25, 26].
- We observe that the problem of activation variance accumulation introduced by identity mapping may be harmful in some scenarios, *e.g.*, adding outputs of multiple residual units implemented by identity mapping [26] together in the Hourglass structure. A simple yet effective solution is introduced for solving this issue.

We evaluate the proposed method on two popular human pose estimation benchmarks, and report state-of-the-art results. We also demonstrate the generalization ability of our approach on standard image classification task. Ablation study demonstrates the effectiveness of the pyramid residual module, the new initialization scheme, and the approach in handling drastic activation variance increase caused by adding residual units.

2. Related Work

Human pose estimation. Graph structures, *e.g.*, Pictorial structures [19, 17, 61] and loopy structures [44, 49, 18], have been broadly used to model the spatial relationships among body parts. All these methods were built on handcrafted features such as HOG feature [15], and their performances relied heavily on image pyramid. Recently, deep models have achieved state-of-the-art results in human pose estimation [3, 29, 5, 55, 39, 12, 59, 13, 7, 40]. Among them, DeepPose [53] is one of the first attempts on using DCNNs for human pose estimation. It regressed the coordinates of body parts directly, which suffered from the problem that image-to-locations is a difficult mapping to learn. Therefore, later methods modeled part locations as Gaussian peaks in score maps, and predicted the score maps with fully convolutional networks. In order to achieve higher accuracy, multi-scale testing on image pyramids was often utilized, which produced a multi-scale feature representation. Our method is a complementary to image pyramids.

On the other hand, to learn a model with strong scale invariance, a multi-branch network trained on three scales of image pyramid was proposed in [51]. However, when image pyramids are used for training, computation and memory linearly increases with the number of scales. In comparison, our pyramid residual module provides an efficient way of learning multi-scale features, with relatively small cost in computation and memory.

DCNNs combining multiple layers. In contrast to traditional plain networks (*e.g.*, AlexNet [33] and VGGnets [45]), multi-branch networks exhibit better performance on various vision tasks. In classification, the inception models [47, 30, 48, 46] are one of the most successful multi-branch networks. The input of each module is first mapped to low dimension by 1×1 convolutions, then transformed by a set of filters with different sizes to capture various context information and combined by concatenation. ResNet [25, 26] can be regarded as a two-branch networks with one identity mapping branch. ResNeXt [56] is an extension of ResNet, in which all branches share the same topology. The implicitly learned transforms are aggregated by summation. In our work, we use multi-branch network to explore another possibility: to learn multi-scale features.

Recent methods in pose estimation, object detection and segmentation used features from multiple layers for making predictions [37, 6, 23, 4, 39, 9]. Our approach is complementary to these works. For example, we adopt Hourglass as our basic structure, and replace its original residual units, which learn features from a single scale, with the proposed Pyramid Residual Module.

Weight initialization. Good initialization is essential for training deep models. Hinton and Salakhutdinov [27] adopted the layer-by-layer pretraining strategy to train a

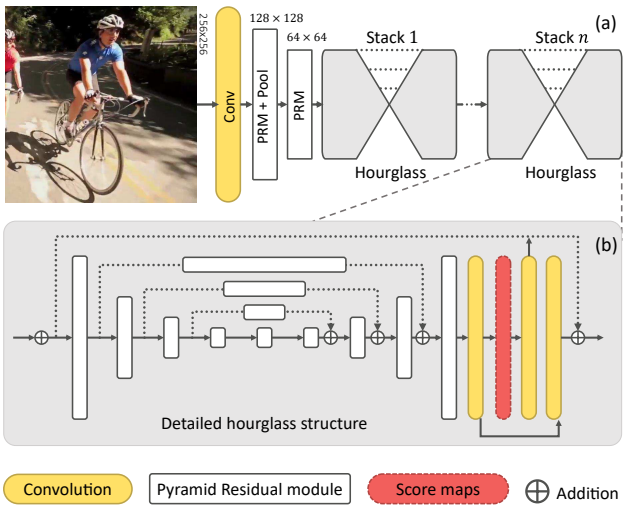

Figure 2. Overview of our framework. (a) demonstrates the network architecture, which has n stacks of hourglass network. Details of each stack of hourglass is illustrated in (b). Score maps of body joint locations are produced at the end of each hourglass, and a squared-error loss is also attached in each stack of hourglass.

deep autoencoder. Krizhevsky *et al*. [33] initialized the weight of each layer by drawing samples from a Gaussian distribution with zero mean and 0.01 standard deviation. However, it has difficulty in training very deep networks due to the instability of gradients [45]. Xavier initialization [21] has provided a theoretically sound estimation of the variance of weight. It assumes that the weights are initialized close to zero, hence the nonlinear activations like Sigmoid and Tanh can be regarded as linear functions. This assumption does not hold for rectifier [38] activations. Thus He *et al*. [24] proposed an initialization scheme for rectifier networks based on [21]. All the above initialization methods, however, are derived for plain networks with only one branch. We identify the problem of the initialization methods when applied for multi-branch networks. An initialization scheme for networks with multiple branches is provided to handle this problem.

3. Framework

An overview of the proposed framework is illustrated in Figure. 2. We adopt the highly modularized stacked Hourglass Network [39] as the basic network structure to investigate feature pyramid learning for human pose estimation . The building block of our network is the proposed *Pyramid Residual Module* (PRM). We first briefly review the structure of hourglass network. Then a detailed discussion of our pyramid residual module is presented.

3.1. Revisiting Stacked Hourglass Network

Hourglass network aims at capturing information at every scale in feed-forward fashion. It first performs bottom-up processing by subsampling the feature maps, and conducts top-down processing by upsampling the feature maps with the comination of higher resolution features from bottom layers, as demonstrated in Figure. 2(b). This bottom-up, top-down processing is repeated for several times to build a "stacked hourglass" network, with intermediate supervision at the end of each stack.

In [39], residual unit [26] is used as the building block of the hourglass network. However, it can only capture visual patterns or semantics at one scale. In this work, we use the proposed pyramid residual module as the building block for capturing multi-scale visual patterns or semantics.

3.2. Pyramid Residual Modules (PRMs)

The objective is to learn feature pyramids across different levels of DCNNs. It allows the network to capture feature pyramids from primitive visual patterns to high-level semantics. Motivated by recent progress on residual learning [25, 26], we propose a novel Pyramid Residual Module (PRM), which is able to learn multi-scale feature pyramids.

The PRM explicitly learns filters for input features with different resolutions. Let $\mathbf{x}^{(l)}$ and $\mathbf{W}^{(l)}$ be the input and the filter of the l-th layer, respectively. The PRM can be formulated as,

$$\mathbf{x}^{(l+1)} = \mathbf{x}^{(l)} + \mathcal{P}(\mathbf{x}^{(l)}; \mathbf{W}^{(l)}), \tag{1}$$

where $\mathcal{P}(\mathbf{x}^{(l)}; \mathbf{W}^{(l)})$ is feature pyramids decomposed as:

$$\mathcal{P}(\mathbf{x}^{(l)}; \mathbf{W}^{(l)}) = g\left(\sum_{c=1}^{C} f_c(\mathbf{x}^{(l)}; \mathbf{w}_{f_c}^{(l)}); \mathbf{w}_g^{(l)}\right) + f_0(\mathbf{x}^{(l)}; \mathbf{w}_{f_0}^{(l)}). \tag{2}$$

The C in (2) denotes the number of pyramid levels, $f_c(\cdot)$ is the transformation for the c-th pyramid level, and $\mathbf{W}^{(l)} = \{\mathbf{w}_{f_c}^{(l)}, \mathbf{w}_g^{(l)}\}_{c=0}^{C}$ is the set of parameters. Outputs of transformations $f_c(\cdot)$ are summed up together, and further convolved by filters $g(\cdot)$. An illustration of the pyramid residual module is illustrated in Figure. 3. To reduce the computational and space complexity, each $f_c(\cdot)$ is designed as a bottleneck structure. For example, in Figure. 3, the feature dimension is reduced by a 1×1 convolution, then new features are computed on a set of subsampled input features by 3×3 convolutions. Finally, all the new features are upsampled to the same dimension and are summed together.

Generation of input feature pyramids. Max-pooling or average-pooling are widely used in DCNNs to reduce the resolution of feature maps, and to encode the translation invariance. But pooling reduces the resolution too fast and coarse by a factor of an integer of at least two, which is unable to generate pyramids gently. In order to obtain input feature maps of different resolutions, we adopt the fractional max-pooling [22] to approximate the smoothing and subsampling process used in generating traditional image

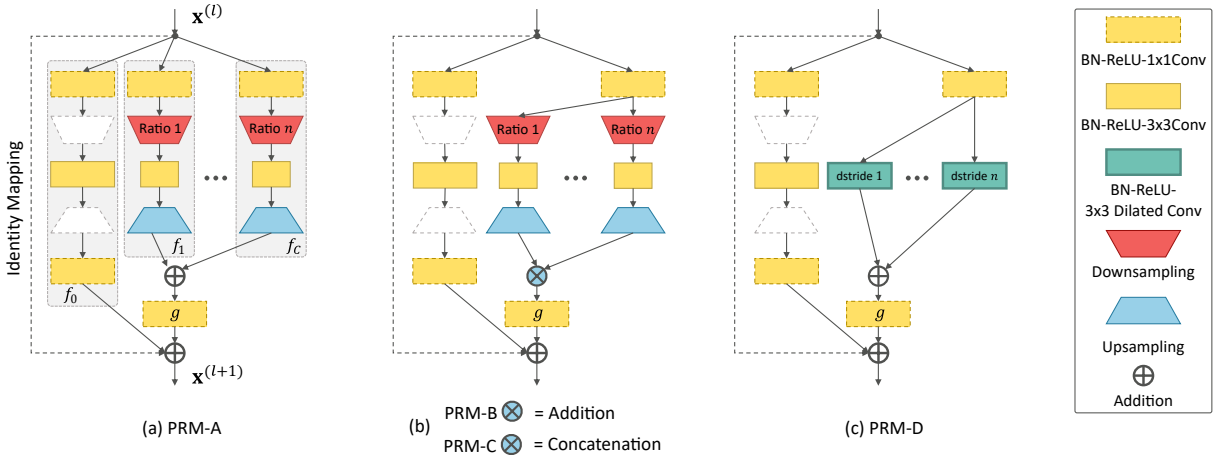

Figure 3. Structures of PRMs. Dashed links indicate identity mapping. (a) PRM-A produces separate input feature maps for different levels of pyramids, while (b) PRM-B uses shared input for all levels of pyramids. PRM-C use concatenation instead of addition to combine features generated from pyramids, which is similar to inception models. (c) PRM-D use dilated convolutions, which are also used in ASPP-net [9], instead of pooling to build the pyramid. The dashed trapezoids mean that the subsampling and upsampling are skipped.

pyramids. The subsampling ratio of the cth level pyramid is computed as:

$$s_c = 2^{-M\frac{c}{C}}, \quad c = 0, \cdots, C, M \geq 1, \tag{3}$$

where $s_c \in [2^{-M}, 1]$ denotes the relative resolution compared with the input features. For example, when $c = 0$, the output has the same resolution as its input. When $M = 1, c = C$, the map has half resolution of its input. In experiments, we set $M = 1$ and $C = 4$, with which the lowest scale in pyramid is half the resolution of its input.

3.3. Discussions

PRM for general CNNs. Our PRM is a general module and can be used as the basic building block for various CNN architectures, *e.g.*, stacked hourglass networks [39] for pose estimation, and Wide Residual Nets [64] and ResNeXt [56] for image classification, as demonstrated in experiments.

Variants in pyramid structure. Besides using fractional max-pooling, convolution and upsampling to learn feature pyramids, as illustrated in Figure. 3(a-b), one can also use dilated convolution [9, 63] to compute pyramids, as shown in Figure. 3(c)(PRM-D). The summation of features in pyramid can also replaced by concatenation, as shown in Figure. 3(b)(PRM-C). We discuss the performance of these variants in experiments, and show that the design in Figure. 3(b)(PRM-B) has comparable performance with others, while maintains relatively fewer parameters and smaller computational complexity.

Weight sharing. To generate the feature pyramids, traditional methods usually apply a same handcrafted filter, *e.g.*, HOG, on different levels of image pyramids [1, 16]. This process corresponds to sharing the weights $\mathbf{W}_{f_c}^{(l)}$ across different levels of pyramid $f_c(\cdot)$, which is able to greatly reduce the number of parameters.

Complexity. The residual unit used in [39] has 256-d input and output, which are reduced to 128-d within the residual unit. We adopt this structure for the branch with original scale (*i.e.*, f_0 in Eq.(2)). Since features with smaller resolution contain relatively fewer information, we use fewer feature channels for branches with smaller scales. For example, given a PRM with five branches and 28 feature channels for branches with smaller scale (*i.e.*, f_1 to f_4 in Eq.(2)), the increased complexity is about only 10% compared with residual unit in terms of both parameters and GFLOPs.

4. Training and Inference

We use score maps to represent the body joint locations. Denote the ground-truth locations by $\mathbf{z} = \{\mathbf{z}_k\}_{k=1}^K$, where $\mathbf{z}_k = (x_k, y_k)$ denotes the location of the kth body joint in the image. Then the ground-truth score map $\mathbf{S}_k$ is generated from a Gaussian with mean $\mathbf{z}_k$ and variance $\mathbf{\Sigma}$ as follows,

$$\mathbf{S}_k(\mathbf{p}) \sim \mathcal{N}(\mathbf{z}_k, \mathbf{\Sigma}), \tag{4}$$

where $\mathbf{p} \in R^2$ denotes the location, and $\mathbf{\Sigma}$ is empirically set as an identity matrix $\mathbf{I}$. Each stack of hourglass network predicts K score maps, *i.e.* $\hat{\mathbf{S}} = \{\hat{\mathbf{S}}_k\}_{k=1}^K$, for K body joints. A loss is attached at the end of each stack defined by the squared error

$$\mathcal{L} = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\|\mathbf{S}_k - \hat{\mathbf{S}}_k\|^2, \tag{5}$$

where N is the number of samples.

During inference, we obtain the predicted body joint locations $\hat{\mathbf{z}}_k$ from the predicted score maps generated from

the last stack of hourglass by taking the locations with the maximum score as follows:

$$\hat{\mathbf{z}}_k = \arg\max_{\mathbf{p}} \hat{\mathbf{S}}_k(\mathbf{p}), \quad k = 1, \cdots, K. \tag{6}$$

4.1. Initialization Multi-Branch Networks

Initialization is essential to train very deep networks [21, 45, 24], especially for tasks of dense prediction, where Batch Normalization [30] is less effective because of the small minibatch due to the large memory consumption of fully convolutional networks. Existing weight initialization methods [33, 21, 24] are designed upon the assumption of a plain networks without branches. The proposed PRM has multiple branches, and does not meet the assumption. Recent developed architectures with multiple branches, *e.g.*, Inception models [47, 30, 48, 46] and ResNets [25, 26], are not plain network either. Hence we discuss how to derive a proper initialization for networks adding multiple branches. Our derivation mainly follows [21, 24].

Forward propagation. Generally, multi-branch networks can be characterized by the number of input and output branches. Figure. 4 (a) shows an example where the lth layer has $C_i^{(l)}$ input branches and one output branch. Figure. 4 (b) shows an example where the lth layer has one input branch and $C_o^{(l)}$ output branches. During forward propagation, $C_i^{(l)}$ affects the variance for the output of the lth layer while $C_o^{(l)}$ does not. At the lth layer, assume there are $C_i^{(l)}$ input branches and $C_o^{(l)}$ output branches. There are $C_i^{(l)}$ input vectors $\{\mathbf{x}_c^{(l)} | c = 1, \ldots, C_i^{(l)}\}$. Take fully-connected layer for example, a response is computed as:

$$\mathbf{y}^{(l)} = \mathbf{W}^{(l)} \sum_{c=1}^{C_i^{(l)}} \mathbf{x}_c^{(l)} + \mathbf{b}^{(l)}, \tag{7}$$

$$\mathbf{x}^{(l+1)} = f\left(\mathbf{y}^{(l)}\right), \tag{8}$$

where $f(\cdot)$ is the non-linear activation function.

As in [21, 24], we assume that $\mathbf{W}^{(l)}$ and $\mathbf{x}^{(l)}$ are both independent and identically distributed (i.i.d.), and they are independent of each other. Therefore, we respectively denote $y^{(l)}, x^{(l)}$ and $w^{(l)}$ as the element in $\mathbf{y}^{(l)}, \mathbf{x}^{(l)}$ and $\mathbf{W}^{(l)}$. Then we have,

$$\mathrm{Var}\left[y^{(l)}\right] = C_i^{(l)} n_i^{(l)} \mathrm{Var}\left[w^{(l)} x^{(l)}\right], \tag{9}$$

where $n_i^{(l)}$ is the number of elements in $\mathbf{x}_c^{(l)}$ for $c = 1, \ldots, C_i^{(l)}$. Suppose $w^{(l)}$ has zero mean. The variance for the product of independent variables above is as follows:

$$\begin{aligned}\mathrm{Var}\left[y^{(l)}\right] &= C_i^{(l)} n_i^{(l)} \mathrm{Var}\left[w^{(l)}\right] \mathrm{E}\left[\left(x^{(l)}\right)^2\right] \\ &= \alpha C_i^{(l)} n_i^{(l)} \mathrm{Var}\left[w^{(l)}\right] \mathrm{Var}\left[y^{(l-1)}\right],\end{aligned}$$

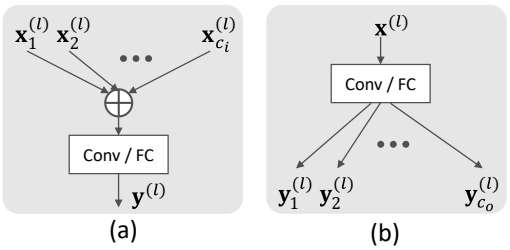

Figure 4. Examples of multi-branch networks when (a) the inputs might be an addition of multiple branches, or (b) the output might be forwarded to multiple branches.

where α depends on the activation function f in (8). $\alpha = 0.5$ for ReLU and $\alpha = 1$ for Tanh and Sigmoid. In order to make the variances of the output $y^{(l)}$ approximately the same for different layers l, the following condition should be satisfied:

$$\alpha C_i^{(l)} n_i^{(l)} \mathrm{Var}\left[w^{(l)}\right] = 1. \tag{10}$$

Hence in initialization, a proper variance for $W^{(l)}$ should be $1/(\alpha C_i^{(l)} n_i^{(l)})$.

Backward propagation. Denote $\frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(l)}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{y}^{(l)}}$ by $\Delta \mathbf{x}^{(l)}$ and $\Delta \mathbf{y}^{(l)}$ respectively. During backward propagation, the gradient is computed by chain rule,

$$\Delta \mathbf{x}^{(l)} = \sum_{c=1}^{C_o^{(l)}} \mathbf{W}^{(l)T} \Delta \mathbf{y}^{(l)}, \tag{11}$$

$$\Delta \mathbf{y}^{(l)} = f'(\mathbf{y}^{(l)}) \Delta \mathbf{x}^{(l+1)}. \tag{12}$$

Suppose $w^{(l)}$ and $\Delta y^{(l)}$ are i.i.d. and independent of each other, then $\Delta x^{(l)}$ has zero mean when $w^{(l)}$ is initialized with zero mean and symmetric with small magnitude. Let $n_o^{(l)}$ denote the number of output neurons. Then we have,

$$\mathrm{Var}\left[\Delta x^{(l)}\right] = C_o^{(l)} n_o^{(l)} \mathrm{Var}[w^{(l)}] \mathrm{Var}[\Delta y^{(l)}]. \tag{13}$$

Denote $\mathrm{E}(f'(y^{(l)})) = \alpha$. $\alpha = 0.5$ for ReLU and $\alpha = 1$ for Tanh and Sigmoid. We further assume that $f'(y^{(l)})$ and $\Delta x^{(l)}$ are independent of each other, then from Eq. (12), we have $\mathrm{E}\left[\Delta \mathbf{y}^{(l)}\right] = \alpha\, \mathrm{E}\left[\Delta \mathbf{x}^{(l+1)}\right]$. Then we can derive that $\mathrm{Var}[\Delta y^{(l)}] = \mathrm{E}[(\Delta y^{(l)})^2] = \alpha\, \mathrm{Var}[x^{(l+1)}]$. Therefore, from Eq.(13) we have,

$$\mathrm{Var}\left[\Delta x^{(l)}\right] = \alpha C_o^{(l)} n_o^{(l)} \mathrm{Var}[w^{(l)}] \mathrm{Var}[\Delta x^{(l+1)}].. \tag{14}$$

To ensure $\mathrm{Var}[\Delta x^{(l)}] = \mathrm{Var}[\Delta x^{(l+1)}]$, we must have $\mathrm{Var}[w^{(l)}] = 1/(\alpha C_o^{(l)} n_o^{(l)})$.

In many cases, $C_i^{(l)} n_i^{(l)} \neq C_o^{(l)} n_o^{(l)}$. As in [21], a compromise between the forward and backward constraints is to have,

$$\mathrm{Var}[w^{(l)}] = \frac{1}{\alpha^2 (C_i^{(l)} n_i^{(l)} + C_o^{(l)} n_o^{(l)})}, \quad \forall l. \tag{15}$$

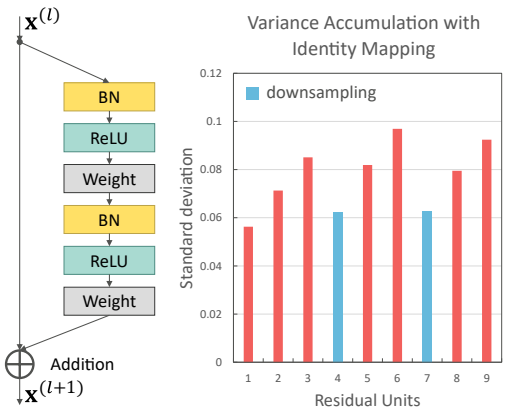

Figure 5. Response variances accumulate in ResNets. This accumulation can be reset (blue bar) when the identity mappings are replaced by convolution or batch normalization (*i.e.*, when the feature channels of feature resolutions changes between input and output features).

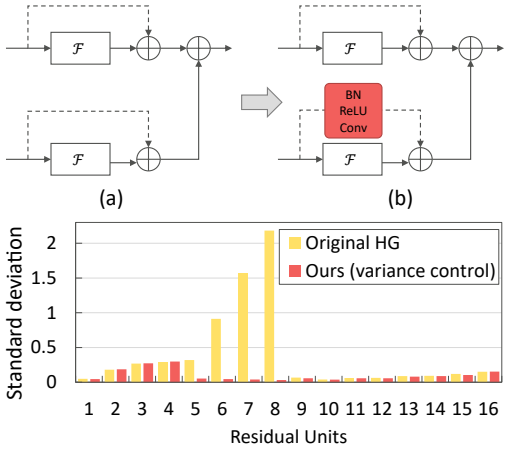

Figure 6. Top: (a) Addition of outputs of two identity mappings. (b) One identity mapping is replaced by a BN-ReLU-Conv block. Bottom: Statistics of response variances of the original hourglass network (yellow bar) and our structure (b) (red bar).

Special case. For plain networks with one input and one output branch, we have $C_i^{(l)} = C_o^{(l)} = 1$ in (15). In this case, the result in (15) degenerates to the conclusions obtained for Tanh and Sigmoid in [21] and the conclusion in [24] for ReLU.

General case. In general, a network with branches would have $C_i^{(l)} \neq 1$ or $C_o^{(l)} \neq 1$ for some ls. Therefore, the number of input branches and output branches should be taken into consideration when initializing parameters. Specifically, if several multi-branch layers are stacked together without other operations (*e.g.*, batch normalization,convolution, ReLU, *etc.*), the output variance would be increased approximately $\prod_l C_i^{(l)}$ times by using Xavier [21] or MSR [24] initialization.

4.2. Output Variance Accumulation

Residual learning [25, 26] allows us to train extremely deep neural networks due to identity mappings. But it is also the source of its drawbacks: identity mapping keeps increasing the variances of responses when the network goes deeper, which increases the difficulty of optimization.

The response of the residual unit is computed as follows:

$$\mathbf{x}^{(l+1)} = \mathbf{x}^{(l)} + \mathcal{F}\left(\mathbf{x}^{(l)}; \mathbf{W}^{(l)}\right), \tag{16}$$

where $\mathcal{F}$ denotes the residual function, *e.g.*, a bottleneck structure with three convolutions ($1 \times 1 \rightarrow 3 \times 3 \rightarrow 1 \times 1$). Assume $\mathbf{x}^{(l)}$ and $\mathcal{F}\left(\mathbf{x}^{(l)}; \mathbf{W}^{(l)}\right)$ are uncorrelated, then the variance of the response of residual unit is as

$$\begin{aligned} \mathrm{Var}\left[\mathbf{x}^{(l+1)}\right] &= \mathrm{Var}\left[\mathbf{x}^{(l)}\right] + \mathrm{Var}\left[\mathcal{F}\left(\mathbf{x}^{(l+1)}; \mathbf{W}^{(l)}\right)\right] \\ &> \mathrm{Var}\left[\mathbf{x}^{(l)}\right], \end{aligned} \tag{17}$$

where $\mathrm{Var}\left[\mathcal{F}\left(\mathbf{x}^{(l+1)}; \mathbf{W}^{(l)}\right)\right]$ is positive.

In [25, 26], the identity mapping will be replaced by convolution layer when the resolution of feature maps is reduced, or when the dimension of feature channels are increased. This allows the networks to reset the variance of response to a small value, and avoid responses with very large variance, as shown in Figure. 5. The effect of increasing variance becomes more obvious in hourglass-like structures, where the responses of two residual units are summed together, as illustrated in Figure. 6(a). Assume branches are uncorrelated, then the variance will be increased as:

$$\begin{aligned} \mathrm{Var}\left[\mathbf{x}^{(l+1)}\right] &= \sum_{i=1}^{2}\left(\mathrm{Var}\left[\mathbf{x}_i^{(l)}\right] + \mathrm{Var}\left[\mathcal{F}_i\left(\mathbf{x}_i^{(l)}; \mathbf{W}_i^{(l)}\right)\right]\right) \\ &> \sum_{i=1}^{2}\mathrm{Var}\left[\mathbf{x}_i^{(l)}\right]. \end{aligned} \tag{18}$$

Hence the output variance is almost doubled. When the network goes deeper, the variance will increase drastically.

In this paper, we use a 1×1 convolution preceding with batch normalization and ReLU to replace the identity mapping when the output of two residual units are summed up, as illustrated in Figure. 6(b). This simple replacement stops the variance explosion, as demonstrated in Figure. 6(c). In experiments, we find that breaking the variance explosion also provide a better performance (Section 5.1.3).

5. Experiments

5.1. Experiments on Human Pose Estimation

We conduct experiments on two widely used human pose estimation benchmarks. (i) The MPII human pose dataset [2], which covers a wide range of human activities with 25k images containing over 40k people. (ii) The Leeds Sports Poses (LSP) [31] and its extended training dataset, which contains 12k images with challenging poses in sports.

Table 1. Comparisons of PCKh@0.5 score on the MPII test set. *Ours-A* is trained using the training set used in [51]. *Ours-B* is trained with the same settings but using all the MPII training set.

Method	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Mean
Pishchulin *et al.* [41]	74.3	49.0	40.8	34.1	36.5	34.4	35.2	44.1
Tompson *et al.* [52]	95.8	90.3	80.5	74.3	77.6	69.7	62.8	79.6
Carreira *et al.* [8]	95.7	91.7	81.7	72.4	82.8	73.2	66.4	81.3
Tompson *et al.* [51]	96.1	91.9	83.9	77.8	80.9	72.3	64.8	82.0
Hu&Ramanan [28]	95.0	91.6	83.0	76.6	81.9	74.5	69.5	82.4
Pishchulin *et al.* [42]	94.1	90.2	83.4	77.3	82.6	75.7	68.6	82.4
Lifshitz *et al.* [35]	97.8	93.3	85.7	80.4	85.3	76.6	70.2	85.0
Gkioxary *et al.* [20]	96.2	93.1	86.7	82.1	85.2	81.4	74.1	86.1
Rafi *et al.* [43]	97.2	93.9	86.4	81.3	86.8	80.6	73.4	86.3
Insafutdinov *et al.* [29]	96.8	95.2	89.3	84.4	88.4	83.4	78.0	88.5
Wei *et al.* [55]	97.8	95.0	88.7	84.0	88.4	82.8	79.4	88.5
Bulat&Tzimiropoulos [5]	97.9	95.1	89.9	85.3	89.4	85.7	81.7	89.7
Newell *et al.* [39]	98.2	96.3	91.2	87.1	90.1	87.4	83.6	90.9
Ours-A	98.4	96.5	91.9	88.2	91.1	88.6	85.3	91.8
Ours-B	**98.5**	**96.7**	**92.5**	**88.7**	**91.1**	**88.6**	**86.0**	**92.0**

Table 2. Comparisons of PCK@0.2 score on the LSP dataset.

Method	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Mean
Belagiannis&Zisserman [3]	95.2	89.0	81.5	77.0	83.7	87.0	82.8	85.2
Lifshitz *et al.* [35]	96.8	89.0	82.7	79.1	90.9	86.0	82.5	86.7
Pishchulin *et al.* [42]	97.0	91.0	83.8	78.1	91.0	86.7	82.0	87.1
Insafutdinov *et al.* [29]	97.4	92.7	87.5	84.4	91.5	89.9	87.2	90.1
Wei *et al.* [55]	97.8	92.5	87.0	83.9	91.5	90.8	89.9	90.5
Bulat&Tzimiropoulos [5]	97.2	92.1	88.1	85.2	92.2	91.4	88.7	90.7
Ours	**98.3**	**94.5**	**92.2**	**88.9**	**94.4**	**95.0**	**93.7**	**93.9**

5.1.1 Implementation Details

Our implementation follows [39]. The input image is 256×256 cropped from a resized image according to the annotated body position and scale. For the LSP test set, we simply use the image center as the body position, and estimate the body scale by the image size. Training data are augmented by scaling, rotation, flipping, and adding color noise. All the models are trained using Torch [14]. We use RMSProp [50] to optimize the network on 4 Titan X GPUs with a mini-batch size of 16 (4 per GPU) for 200 epochs. The learning rate is initialized as 7×10^{-4} and is dropped by 10 at the 150th and the 170th epoch. Testing is conducted on six-scale image pyramids with flipping.

5.1.2 Experimental Results

Evaluation measure. Following previous work, we use the Percentage Correct Keypoints (PCK) measure [62] on the LSP dataset, and use the modified PCK measure that uses the matching threshold as 50% of the head segment length (PCKh) [2] on the MPII dataset.

MPII Human Pose. We report the performance on MPII dataset in Table 1. Ours-A is trained using the training and validation set used in [51]. Ours-B is trained with the same settings but using all the MPII training set. Our approach achieves 92.0% PCKh score at threshold of 0.5, which is the new state-of-the-art result. Specifically, our method achieves 1.6% and 2.4% improvements on *wrist* and *ankle*, which are considered as the most challenging parts to be detected. Qualitative results are demonstrated in Figure. 7.

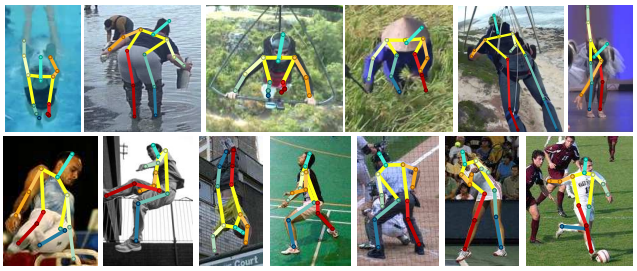

Figure 7. Results on the MPII (top) and the LSP dataset (bottom).

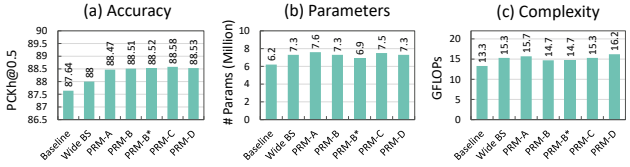

Figure 8. Statistics of (a) accuracy, (b) number of parameters, and (c) computational complexity in terms of GFLOPs on different designs of PRMs in Figure. 3.

Complexity. Our model increases the number of parameters by 13.5% from 23.7M to 26.9M given an eight-stack hourglass network. Our model needs 45.9 GFLOPs for a 256×256 RGB image, which is a 11.4% increase compared to hourglass network (41.2 GFLOPs). As reported in [39], deeper hourglass with more stacks hardly improves result.

LSP dataset. Table 2 presents the PCK scores at the threshold of 0.2. We follow previous methods [42, 55, 29] to train our model by adding MPII training set to the LSP and its extended training set. Our method improves the previous best result with a large margin by 3.2%. For difficult body parts, *e.g.*, *wrist* and *ankle*, we have 3.7% and 5.0% improvements, respectively. Our method gains a lot due to the high occurrence of foreshortening and extreme poses presented in this dataset, as demonstrated in Figure. 7.

5.1.3 Ablation Study

We conduct ablation study on the MPII validation set used in [51] with a 2-stack hourglass network as the basic model.

Architectures of PRM. We first evaluate different designs of PRM, as discussed in Section 3.2, with the same number of branches, and the same feature channels for each branch (*e.g.*, 5 branches with 28 feature channels for each pyramidal branch). We use PRM-A to PRM-D, which corresponds to Figure. 3, to denote the different architectures. Specifically, PRM-A produces separate input feature maps for different levels of pyramids, while PRM-B uses shared feature maps for all levels of pyramids. PRM-C uses concatenation instead of addition to combine features generated from pyramid, which is similar to inception models. PRM-D uses dilated convolutions, which are also used in ASPP-net [9], instead of pooling to build the pyramid. The validation ac-

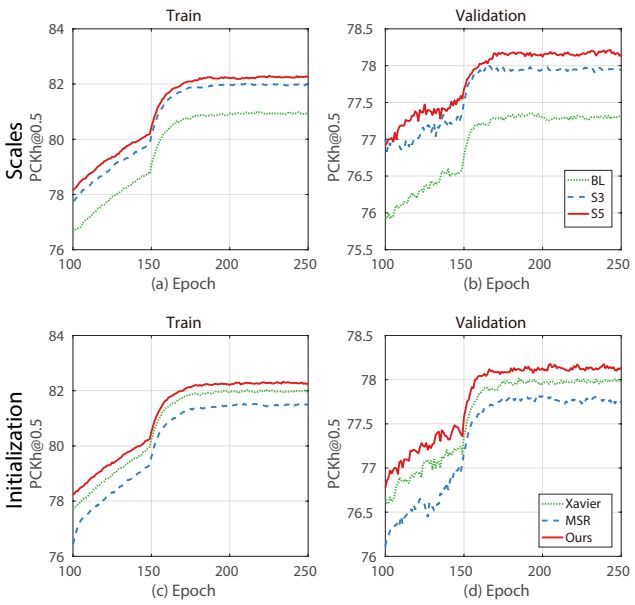

Figure 9. Training and validation curves of PCKh scores *vs*. epoch on the MPII validation set. (a-b) Investigate the number of scales in the pyramid. BL stands for baseline model (two-stack hourglass network), S2 to S8 indicate PRM-B* with four scales to eight scales. (c-d) Comparison of our initialization scheme with Xavier method [21] and MSR method [24].

curacy is reported in Figure. 8(a). All the PRMs have better accuracy compared with the baseline model. We observe that the difference in accuracy between PRM-A to PRM-D is subtle, while the parameters of PRM-A/C are higher than PRM-B/B*/D (Figure. 8(b)), and the computational complexity (GFLOPs) of PRM-A/C/D are higher than PRM-B/B*. Therefore, we use PRM-B* in the rest of the experiments. Noted that increasing the number of channels to make the baseline model has the similar model size as ours (*Wide BS*) would slightly improve the performance. But it is still worse than ours.

Scales of pyramids. To evaluate the trade-off between the scales of pyramids C, we vary the scales from 3 to 5, and fix the model size by tuning the feature channels in each scale. We observe that increasing scales generally improves the performance, as shown in Figure. 9(a-b).

Weight initialization. We compare the performance of our initialization scheme with Xavier [21] and MSR [24] methods. The training and validation curves of accuracy *vs*. epoch are reported in Figure 9(c-d). It can be seen that the proposed initialization scheme achieves better performance than both methods.

Controlling variance explosion. Controlling variance explosion, as discussed in Section 4.2, obtains higher validation score (88.0) compared with the baseline model (87.6). With our pyramid residual module, the performance could be further improved to 88.5 PCKh score.

Table 3. Top-1 test error (%), model size (million) and GFLOPs on CIFAR-10. *WRN-28-10* denote the Wide ResNet with depth 29 and widen factor 10. *ResNeXt-29,* $m \times nd$ denote ResNeXt with depth 29, cardinality m and base width n.

method	#params	GFLOPs	top-1
WRN-28-10 [64]	36.5	10.5	4.17
Ours-28-9	36.4	9.5	3.82
Ours-28-10	42.3	11.3	3.67
ResNeXt-29, 8 × 64d [56]	34.4	48.8	3.65
ResNeXt-29, 16 × 64d [56]	68.2	184.5	3.58
Ours-29, 8 × 64d	45.6	50.5	3.39
Ours-29, 16 × 64d	79.3	186.1	**3.30**

5.2. Experiments on CIFAR-10 Image Classification

The CIFAR-10 dataset [32] consists of 50k training images and 10k test images with size 32 × 32 drawn from 10 classes. We follow previous works for data preparation and augmentation. We incorporate the proposed pyramid branches into two state-of-the-art network architectures, *i.e*., Wide residual networks [64] and ResNeXt [56]. We add four pyramid branches with scales ranging from 0.5 to 1 into the building block of both Wide ResNet and ResNeXt. For Wide ResNet, the total width of all pyramid branches is equal to the width of the output of each residual module. For ResNeXt, we simply use the same width as its original branches for our pyramid branches. Table 3 shows the top-1 test error, model sizes and GFLOPs. Our method with similar or less model size (*Ours-28-9* vs. *WRN-28-10* and *Ours-29,* 8 × 64*d* vs. *ResNeXt-29,* 16 × 64*d*) achieve better results. A larger model with our pyramid module (*Ours-29,* 16 × 64*d*) achieves 3.30% test error, which is the state-of-the-art result on CIFAR-10.

6. Conclusion

This paper has proposed a Pyramid Residual Module to enhance the invariance in scales of the DCNNs. We also provide a derivation of the initialization scheme for multi-branch networks, and demonstrate its theoretical soundness and efficiency through experimental analysis. Additionally, a simple yet effective method to prevent the variances of response from explosion when adding outputs of multiple identity mappings has been proposed. Our PRMs and the initialization scheme for multi-branch networks are general, and would help other tasks.

Acknowledgment: This work is supported by SenseTime Group Limited, the General Research Fund sponsored by the Research Grants Council of Hong Kong (Project Nos. CUHK14213616, CUHK14206114, CUHK14205615, CUHK419412, CUHK14203015, CUHK14207814, and CUHK14239816), the Hong Kong Innovation and Technology Support Programme (No.ITS/121/15FX), National Natural Science Foundation of China (No. 61371192), and ONR N00014-15-1-2356.

References

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 1984. 4

[2] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014. 6, 7

[3] V. Belagiannis and A. Zisserman. Recurrent human pose estimation. *FG*, 2017. 2, 7

[4] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016. 2

[5] A. Bulat and G. Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In *ECCV*, 2016. 2, 7

[6] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV*, 2016. 2

[7] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017. 2

[8] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *CVPR*, 2016. 7

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 2, 4, 7

[10] W. Chen, Z. Fu, D. Yang, and J. Deng. Single-image depth perception in the wild. In *NIPS*, 2016. 2

[11] X. Chen and A. L. Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *NIPS*, 2014. 1

[12] X. Chu, W. Ouyang, H. Li, and X. Wang. Structured feature learning for pose estimation. In *CVPR*, 2016. 2

[13] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang. Multi-context attention for human pose estimation. *CVPR*, 2017. 2

[14] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011. 7

[15] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 2

[16] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2010. 4

[17] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005. 2

[18] V. Ferrari, M. Marín-Jiménez, and A. Zisserman. 2d human pose estimation in tv shows. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, pages 128–147. Springer, 2009. 2

[19] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67–92, 1973. 2

[20] G. Gkioxari, A. Toshev, and N. Jaitly. Chained predictions using convolutional neural networks. In *ECCV*, 2016. 7

[21] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010. 2, 3, 5, 6, 8

[22] B. Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014. 3

[23] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 2

[24] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 2, 3, 5, 6, 8

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3, 5, 6

[26] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 2, 3, 5, 6

[27] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 2006. 2

[28] P. Hu and D. Ramanan. Bottom-up and top-down reasoning with hierarchical rectified gaussians. In *CVPR*, 2016. 7

[29] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *ECCV*. Springer, 2016. 2, 7

[30] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015. 2, 5

[31] S. Johnson and M. Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *BMVC*, 2010. 1, 6

[32] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. In *Tech report*, 2009. 8

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2, 3, 5

[34] H. Li, Y. Liu, W. Ouyang, and X. Wang. Zoom out-and-in network with recursive training for object proposal. *arXiv preprint arXiv:1702.05711*, 2017. 2

[35] I. Lifshitz, E. Fetaya, and S. Ullman. Human pose estimation using deep consensus voting. In *ECCV*, 2016. 7

[36] S. Liu, X. Liang, L. Liu, X. Shen, J. Yang, C. Xu, L. Lin, X. Cao, and S. Yan. Matching-cnn meets knn: Quasi-parametric human parsing. In *CVPR*, 2015. 1

[37] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2

[38] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 3

[39] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*. Springer, 2016. 1, 2, 3, 4, 7

[40] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy. Towards accurate multi-person pose estimation in the wild. In *CVPR*, 2017. 2

[41] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele. Strong appearance and expressive spatial models for human pose estimation. In *ICCV*, 2013. 7

[42] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *CVPR*, 2016. 1, 7

[43] U. Rafi, J. Gall, and B. Leibe. An efficient convolutional network for human pose estimation. In *ECCV*, 2016. 7

[44] X. Ren, A. C. Berg, and J. Malik. Recovering human body configurations using pairwise constraints between parts. In *ICCV*, 2005. 2

[45] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 3, 5

[46] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016. 2, 5

[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2, 5

[48] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 2, 5

[49] T.-P. Tian and S. Sclaroff. Fast globally optimal 2d human detection with loopy graph models. In *CVPR*, 2010. 2

[50] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 2012. 7

[51] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *CVPR*, 2015. 1, 2, 7

[52] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, 2014. 1, 7

[53] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 1, 2

[54] C. Wang, Y. Wang, and A. L. Yuille. An approach to pose-based action recognition. In *CVPR*, 2013. 1

[55] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016. 1, 2, 7

[56] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016. 2, 4, 8

[57] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg. Parsing clothing in fashion photographs. In *CVPR*, 2012. 1

[58] W. Yang, P. Luo, and L. Lin. Clothing co-parsing by joint image segmentation and labeling. In *CVPR*, 2014. 1

[59] W. Yang, W. Ouyang, H. Li, and X. Wang. End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. In *CVPR*, 2016. 2

[60] W. Yang, Y. Wang, and G. Mori. Recognizing human actions from still images with latent poses. In *CVPR*, 2010. 1

[61] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, 2011. 2

[62] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *TPAMI*, 35(12):2878–2890, 2013. 7

[63] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016. 4

[64] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016. 4, 8

[65] L. Zheng, Y. Huang, H. Lu, and Y. Yang. Pose invariant embedding for deep person re-identification. *arXiv preprint arXiv:1701.07732*, 2017. 1