Towards Efficient U-Nets: A Coupled and Quantized Approach

Zhiqiang Tang, Xi Peng, Kang Li and Dimitris N. Metaxas, *Fellow, IEEE*

Abstract—In this paper, we propose to couple stacked U-Nets for efficient visual landmark localization. The key idea is to globally reuse features of the same semantic meanings across the stacked U-Nets. The feature reuse makes each U-Net light-weighted. Specially, we propose an $order$-K coupling design to trim off long-distance shortcuts, together with an iterative refinement and memory sharing mechanism. To further improve the efficiency, we quantize the parameters, intermediate features, and gradients of the coupled U-Nets to low bit-width numbers. We validate our approach in two tasks: human pose estimation and facial landmark localization. The results show that our approach achieves state-of-the-art localization accuracy but using ∼70% fewer parameters, ∼30% less inference time, ∼98% less model size, and saving ∼75% training memory compared with benchmark localizers.

Index Terms—Stacked U-Nets, Dense Connectivity, Network Quantization, Efficient AI, Human Pose Estimation, Face Alignment.

1 Introduction

The U-Net architecture [1] is a basic category of Convolution Neural Network (CNN). It has been widely used in the location-sensitive tasks: semantic segmentation [2], biomedical image segmentation [1], human pose estimation [3], facial landmark localization [4], etc. A U-Net contains several top-down and bottom-up blocks. There are shortcut connections between the corresponding top-down and bottom-up blocks. The essence of U-Net is integrating both the local visual cues and global context information to make the inference.

Recently, the stacked U-Nets, *e.g.* hourglasses (HGs) [5] become a standard baseline in landmark localization tasks. The multiple top-down and bottom-up processing could refine the inference stage-by-stage. Many techniques, such as adversarial training [6], attention modeling [7], etc, are used to further improve its inference accuracy. However, very few works try to improve the efficiency of stacked U-Nets.

The stacked U-Nets usually contain dozens of millions of float parameters. The massive high-precision computations require the high-end GPU devices with abundant memory. It is very challenging for the applications in resource-limited mobile devices. In this paper, we aim to improve the efficiency of staked U-Nets in three aspects: parameter, memory, and bit-width.

Parameter efficiency. The shortcut connections could promote feature reuse, thereby reducing many redundant parameters. For a single U-Net, it is straightforward to change each block into a dense block. Insides a dense block, several convolutional layers are densely connected.

However, adding the shortcut connections properly in the stacked U-Nets is nontrivial. Our solution is to couple the stacked U-Nets, generating the coupled U-Nets (CU-Net). The key idea is to directly connect blocks of the same semantic meanings, *i.e.* having the same resolution in either top-down or bottom-up context, from any U-Net to all subsequent U-Nets. Please refer to Fig. 1 for an illustration. This encourages the feature reuse across the stacks resulting in light-weighted U-net.

Yet there is an issue in designing the CU-Net. The number of shortcut connections would have a quadratic growth if we couple every U-Net pair, *e.g.* n stacked U-Nets would generate $O(n^2)$ connections. To balance parameter efficiency and inference accuracy, we propose the $order$-K coupling that couples a U-Net to its K instance successors. Besides, we employ intermediate supervisions to provide additional gradients, compensating the trimmed off shortcut connections. The $order$-K coupling cuts down ∼70% parameter number and ∼30% forward time without sacrificing inference accuracy compared with stacked U-Nets [5]. Furthermore, we propose an iterative design that can further reduce the parameter number to ∼50%. More specifically, the CU-Net output of the first pass is used as the input of the second pass, which is equivalent to a double-depth CU-Net.

Memory efficiency. The shortcut connections may have a severe memory issue. For instance, a naive implementation intends to make feature copies repeatedly for all shortcut connections. We adapt the memory efficient implementation [8] to share memories for features in connected blocks. This technique can reduce the training memory by ∼40%.

Bit-width efficiency. In addition to the parameter and memory efficiency, we also investigate model quantization to improve the bit-width efficiency. Different from the common setup, we quantize the parameters as well as the data flow (intermediate features and gradients). On the one hand, we ternarize or binarize the float parameters, which shrinks 16× or 32× model size in testing. On the other hand, we quantize the data flow with different bit-width setups, which saves ∼4× training memory without compromising the performance. To the best of our knowledge, this is the first study to simultaneously quantize the parameters and the

- *Zhiqiang Tang is with the Department of Computer Science, Rutgers University, NJ, USA. E-mail: zhiqiang.tang@rutgers.edu*
- *Xi Peng (corresponding author) is with the Department of Computer Science, Binghamton University, NY, USA. E-mail: xpeng@binghamton.edu*
- *Kang Li is with the Department of Orthopaedics, New Jersey Medical School, Rutgers University, NJ, USA. E-mail: kl419@rutgers.edu*
- *Dimitris Metaxas is with the Department of Computer Science, Rutgers University, NJ, USA. E-mail: dnm@cs.rutgers.edu*

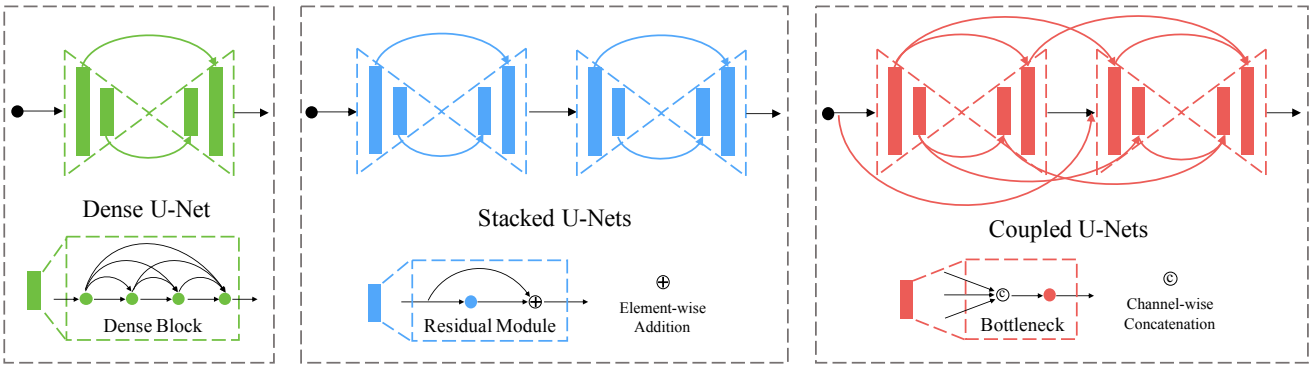

Fig. 1. Illustration of dense U-Net, stacked U-Nets and coupled U-Nets. The CU-Net is a hybrid of dense U-Net and stacked U-Nets, integrating the merits of both dense connectivity and multi-stage top-down and bottom-up refinement. The coupled U-Nets can save ∼70% parameters and ∼30% inference time of stacked U-Nets. Each block in the coupled U-Nets is a bottleneck module which is different from the dense block.

data flow in U-Nets.

In summary, we present a comprehensive study of efficient U-Nets in three aspects: parameter, memory, and bit-width. Coupled U-Nets (CU-Nets), $order$-K coupling and iterative refinement are proposed to balance the parameter efficiency and inference accuracy. Besides, a memory sharing technique is employed to significantly cut down the training memory. Moreover, we investigate the bit-width efficiency by quantizing the parameters as well as the data flow. Two popular tasks, human pose estimation and facial landmark localization, are studied to validate our approach in various aspects. The experimental results prove that our model cuts down ∼70% parameter number and ∼30% inference time. Together with the quantization, we can shrink the model size by ∼98% and reduces ∼75% training memory with comparable performance as state-of-the-art U-Nets designs.

This is an extension of the ECCV work [9]. We have improved it mainly in four aspects: giving a more comprehensive study of the U-Net efficiency, presenting a more complete solution for both single U-Net and stacked U-Nets, giving the detailed network architecture and the implementation of $order$-K coupling, conducting more thorough ablation study to evaluate every component. For details, please refer to the difference summary.

2 Related Work

In this section, we review the recent developments on designing convolutional network architectures, quantizing the neural networks and two landmark localization tasks: human pose estimation and facial landmark localization.

Network Architecture. The research on network architectures have been active since AlexNet [10] appeared. First, by using smaller filters, the VGG [11] network become several times deeper than the AlexNet and obtain much better performance. Then the Highway Networks [12] could extend its depths to more than 100 layers. The identity mappings make it possible to train very deep ResNet [13]. The popular stacked U-Nets [5] are designed based on the residual modules. More recently, the DenseNet [14] outperforms the ResNet in the image classification task. Some works [15], [16] try to use the dense connectivity in the U-Net. Following the DenseNet [14], their dense connectivity is still local within each block. However, the proposed coupling connectivity is global at the U-Net level. Besides, we aim to improve the U-Net efficiency whereas they focus on accuracy. Our method is also related to DLA [17] in the sense of feature aggregation. However, the proposed coupling connectivity is designed for multiple stacked U-Nets whereas DLA [17] is for single U-Net.

Network Quantization. Training deep neural networks usually consumes a large amount of computational power, which makes it hard to deploy on mobile devices. Recently, network quantization approaches [18], [19], [20], [21], [22] offer an efficient solution to reduce the network size by cutting down high precision operations and operands. TWN [19] utilize two symmetric thresholds to ternarize the parameters to +1, 0, or -1. XNOR-Net [22] quantize the parameters and intermediate features. It also uses a scaling factor to approximate the real-value parameters and features. DoReFa-Net [20] quantizes gradients to low bit-width numbers. WAGE [21] proposes an integer-based implementation for training and inference simultaneously. These quantization methods are mainly designed for the image classification networks. In the recent binarized convolutional landmark localizer (BCLL) [23] architecture, XNOR-Net [22] is utilized for network binarization. However, BCLL only quantizes parameters for inference. Due to its high precision demand for training, it cannot save training memory and improve training efficiency. Therefore, we explore to quantize the proposed CU-Net in training and inference simultaneously. That is, we quantize the parameters as well as the intermediate features and gradients.

Human Pose Estimation. Starting from the DeepPose [3], CNNs based approaches [24], [25], [26], [27], [28], [29], [30], [31] become the mainstream in human pose estimation and prediction. Recently, the architecture of stacked U-Nets [5] has obviously beaten all the previous ones in terms of usability and accuracy. Therefore, all recent state-of-the-art methods [6], [7], [32], [33] build on its architecture. Chu *et. al.* add the Conditional Random Field to refine its prediction. Yang *et. al.* replace the residual modules in stacked U-Nets with more sophisticated ones. Chen *et. al.* [33] use an additional network to provide adversarial supervisions. Peng *et. al.* [6] use adversarial data augmentation to train more robust stacked U-Nets. All these approaches focus on boosting the inference accuracy. In contrast, we study to improve the efficiency of stacked U-Nets in various aspects.

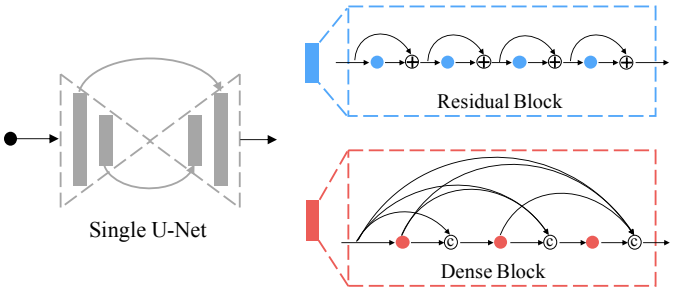

Fig. 2. Illustration of single residual U-Net and dense U-Net. Each top-down or bottom-up block in the residual U-Net is a residual block of several residual modules. In contrast, each block in the dense U-Net is a dense block with several densely connected layers. The dense connections promote the feature reuse inside each block. Therefore, we could largely reduce the parameters of single U-Net by replacing each residual block with a dense block.

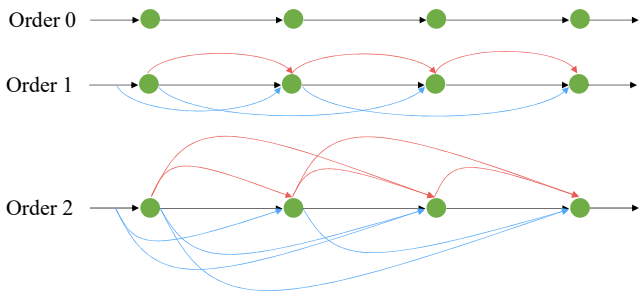

Fig. 3. Illustration of $order$-K coupling. For simplicity, each dot represents one U-Net. The red lines are shortcut connections for the same semantic blocks in different U-Nets. The initial input and the U-Net outputs pass through the blue lines. $Order$-0 coupling (**Top**) strings U-Nets together only by their inputs and outputs, i.e., stacked U-Nets. $Order$-1 coupling (**Middle**) has shortcut connections only for adjacent U-Nets. Similarly, $order$-2 coupling (**Bottom**) has shortcut connections for 3 nearby U-Nets.

Facial Landmark Localization. Similarly, CNNs have largely reshaped the field of facial landmark localization. Traditional methods could be easily outperformed by the CNNs based [34], [35], [36], [37]. Especially, the network cascade has shown its effectiveness in detecting the facial keypoints. Sun *et. al.* [38] propose to use three levels of neural networks to predict landmark locations. Zhang *et. al.* [39] study the problem via cascades of stacked auto-encoders which gradually refine the landmark positions. In the recent Menpo Facial Landmark Localization Challenge [40], stacked U-Nets [5] achieve state-of-the-art performance. The proposed $order$-K coupled U-Nets could produce comparable localization errors but with much fewer parameters, smaller model size, and less training memory.

3 Method

We first describe the dense connectivity to improve the efficiency of a single U-Net. Then we introduce the coupling connectivity to boost the efficiency of stacked U-Nets. The $order$-K coupling is presented to balance the accuracy and parameter efficiency. We also give an iterative refinement to cut the CU-Net in one half. At last, we quantize its the parameters, intermediate features and gradients.

3.1 Dense Connectivity for Single U-Net

A U-Net [1] is usually symmetric with the same number of top-down and bottom-up blocks. And a pair of top-down and bottom-up blocks with the same resolutions are connected. In this paper, we refer to each top-down or bottom-up block as a semantic block. We intend to add shortcut connections to compress a single U-Net. One straightforward way is to densely connect the convolutional layers of a semantic block, forming a dense block. An illustration is shown in Figure 2.

The dense connections could increase the information flow in the U-Net. However, the single dense U-Net follows the DenseNet design [14]. That is, the dense connections are only within the local blocks. If we have several stacked U-Nets, how could we add the shortcut connections? It is more meaningful to improve the efficiency of stacked U-Nets since they are commonly used in practice.

3.2 Coupling Connectivity for Stacked U-Nets

Suppose multiple U-Nets are stacked together, for the ℓ^{th} top-down and bottom-up blocks in the n^{th} U-Net, we use $f_\ell^n(\cdot)$ and $g_\ell^n(\cdot)$ to denote their non-linear transformations. Their outputs are represented by $\mathbf{x}_\ell^n$ and $\mathbf{y}_\ell^n$. $f_\ell^n(\cdot)$ and $g_\ell^n(\cdot)$ comprise operations of Convolution (Conv), Batch Normalization (BN) [41], rectified linear units (ReLU) [42], and pooling. Please note that the top-down and bottom-up blocks have independent index ℓ. To make the resolutions of x_ℓ^n and y_ℓ^n match, their indexes ℓ both increase from the high to low resolutions.

Stacked U-Nets. We recap the popular stacked U-Nets [5] based on the residual modules. Basically, each block is a residual module. The feature transitions at the ℓ^{th} top-down and bottom-up blocks of the n^{th} U-Net can be written as:

$$\mathbf{x}_\ell^n = f_\ell^n(\mathbf{x}_{\ell-1}^n), \mathbf{y}_{\ell-1}^n = g_{\ell-1}^n(\mathbf{y}_\ell^n + \mathbf{x}_\ell^n). \quad (1)$$

The shortcut connections only exist locally within each U-Net. It restricts the feature reuse across U-Nets. Thus, it contains many redundant parameters.

Coupled U-Nets. To facilitate the feature reuse across stacked U-Nets, we propose a global connectivity pattern. The same semantic blocks, i.e., blocks at the same locations of different U-Nets, have direct connections. Hence, we refer to this coupled U-Nets architecture as *CU-Net*. Figure 1 gives an illustration. It essentially merges features from multiple sources and then generates new features. Mathematically, the feature transitions at the ℓ^{th} top-down and bottom-up blocks of the n^{th} U-Net can be formulated as:

$$\mathbf{x}_\ell^n = f_\ell^n([\mathbf{x}_{\ell-1}^n, \mathbf{X}_\ell^{n-1}]), \mathbf{y}_{\ell-1}^n = g_{\ell-1}^n([\mathbf{y}_\ell^n, \mathbf{x}_\ell^n, \mathbf{Y}_\ell^{n-1}]), \quad (2)$$

where $\mathbf{X}_\ell^{n-1} = \mathbf{x}_\ell^0, \mathbf{x}_\ell^1, \cdots, \mathbf{x}_\ell^{n-1}$ are the outputs of the ℓ^{th} top-down blocks in all preceding U-Nets. Similarly, $\mathbf{Y}_\ell^{n-1} = \mathbf{y}_\ell^0, \mathbf{y}_\ell^1, \cdots, \mathbf{y}_\ell^{n-1}$ represent the outputs from the ℓ^{th} bottom-up blocks. $[\cdots]$ denotes the feature concatenation, which could make information flow more efficiently than the summation operation in Equation 1.

According to Equation 2, a block receives features not only from connected blocks in the current U-Net but also the output features of the same semantic blocks from all its preceding U-Nets. Each U-Net becomes light-weighted, benefiting from the feature reuse across stacked U-Nets. Thus,

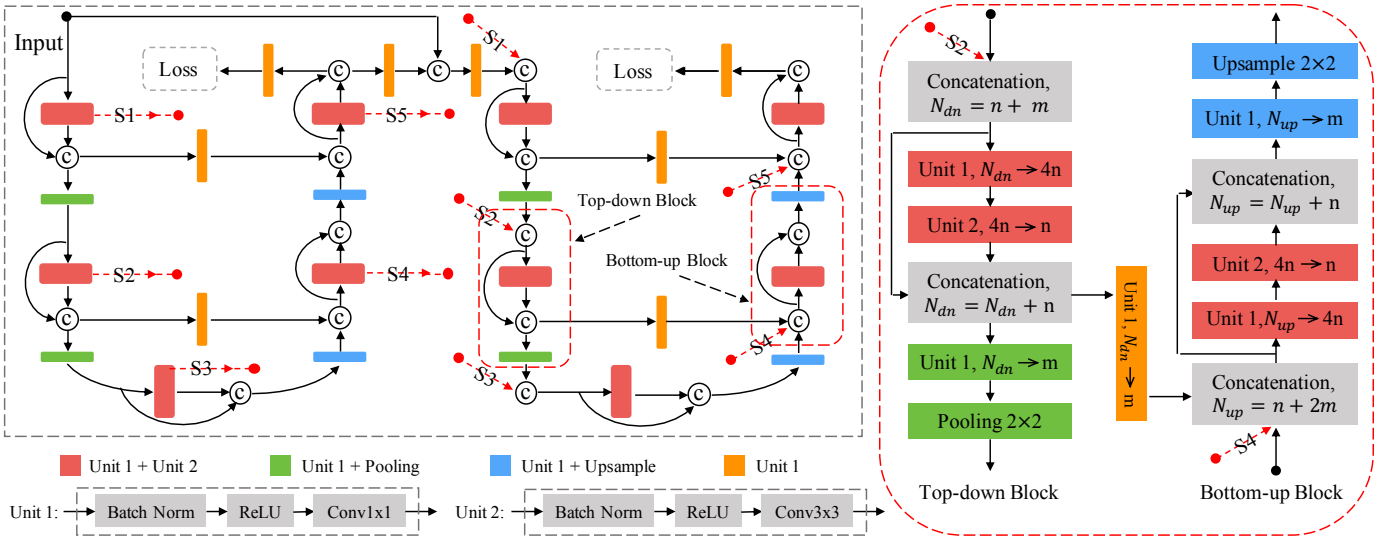

Fig. 4. Implementation of CU-Net. The left figure shows 2 U-Nets coupled together through the red dot lines. Each U-Net has its own supervision. For simplicity, we only show a pair of top-down and bottom-up semantic blocks. The right figure gives the detailed implementation of a pair of semantic blocks in the second U-Net. Basically, m features from the former block of current U-Net and n features from the same semantic block of the preceding U-Net are concatenated and transformed to $4n$ features by a conv1x1. A conv3x3 then generates n new features and another conv1x1 compresses the $m + n$ input and n generated features to m features. The bottom-up block needs to concatenate the additional m skipped features.

the parameter efficiency is largely improved. Please note that the coupling connectivity is global whereas the dense connectivity in DenseNet [14] is local within blocks.

3.3 Order-K Coupling

In the above formulation of CU-Net, we connect blocks with the same semantic meanings across all U-Nets. The shortcut connections would have quadratic growth depth-wise. To make CU-Net more parameter efficient, we propose to cut off some trivial connections. For compensation, we add a supervision at the end of each U-Net. Both the intermediate supervisions and the shortcut connections could alleviate the gradient vanish problem, helping train better CU-Net models. Mathematically, the features $\mathbf{X}_\ell^{n-1}$ and $\mathbf{Y}_\ell^{n-1}$ in Equation 2 turns into

$$\mathbf{X}_\ell^{n-1} = \mathbf{x}_\ell^{n-k}, \cdots, \mathbf{x}_\ell^{n-1}, \tag{3}$$

$$\mathbf{Y}_\ell^{n-1} = \mathbf{y}_\ell^{n-k}, \cdots, \mathbf{y}_\ell^{n-1}, \tag{4}$$

where $0 \leq k \leq n$ represents how many preceding nearby U-Nets connect with the current one. $k = n$ or $k = 0$ would result in the stacked U-Nets or fully densely connected U-Nets. A medium order could reduce the growth of CU-Net parameters from quadratic to linear. Therefore, it largely improves the parameter efficiency of CU-Net and could make CU-Net grow several times deeper.

The proposed $order$-K coupling has a similar philosophy as the Variable Order Markov (VOM) models [43]. Each U-Net can be viewed as a state in the Markov model. The current U-Net depends on a fixed number of preceding nearby U-Nets, instead of preceding either only one or all U-Nets. In this way, the long-range connections are cut off. Figure 3 illustrates the couplings of three different orders. The shortcut connections exist for the U-Net semantic blocks and U-Net inputs. We differentiate them by the red and blue colors in Figure 3. We define the coupling order based on either the red or blue lines. In Figure 3, both the red and blue shortcut connections follow the VOM patterns of $order$-0, $order$-1 and $order$-2.

We could extend the $order$-K coupling to more general $order$-K connectivity if each U-Net is simplified as a unit. Dense connectivity [14] is a special case of $order$-K connectivity on the limit of K. For small K, $order$-K connectivity is much more parameter efficient. But fewer connections may affect the inference accuracy of very deep CU-Net. To make CU-Net have both high parameter efficiency and inference accuracy, we propose to use $order$-K connectivity in conjunction with intermediate supervisions. In contrast, DenseNet [14] has only one supervision at the end. Thus, it cannot effectively take advantage of $order$-K connectivity.

3.4 Iterative Refinement

In order to further improve the parameter efficiency of CU-Net, we consider an iterative refinement. It uses only half of a CU-Net but may achieve comparable accuracy. In the iterative refinement, a CU-Net has two forward passes. In the first pass, we concatenate the inputs of the first and last U-Nets and merge them in a small dense block. More specifically, if the input features of a U-Net have n channels, concatenating the inputs of the first and last U-Nets results in $2n$ channel features. Then we forward them through 4 densely connected Conv3×3 layers. Each layer produces m channel new features. Then we concatenate all $2n + 4m$ features and use one Conv1×1 layer to compress them to n channel features, the modified input. Then the modified input is fed forward in the CU-Net again. An illustration of the iterative refinement is shown in Figure 5.

The iterative refinement may cause the overfitting. We provide two techniques to avoid this. First, independent batch normalization parameters are learned in the two iterations. Second, we call the backpropagation separately for each forward pass. Two different mini-batch images are used to update the two iterations. We forward the first batch only

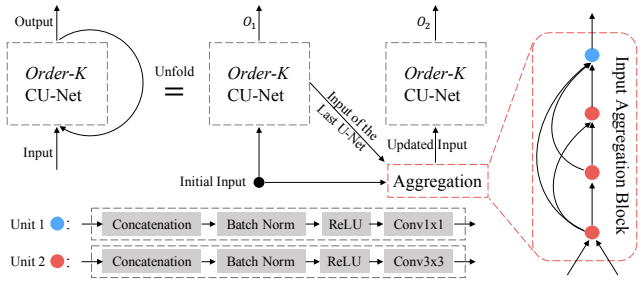

Fig. 5. Illustration of iterative refinement. The same $order$-K CU-Net is used twice in the iterative refinement. In the first iteration, the input of the last U-Net is generated on the basis of the initial input. Then they are concatenated and further aggregated in a dense block. The updated input is forwarded through the $order$-K CU-Net to get the final output. Given a long CU-Net cascade, the iterative refinement has the potential to reduce its depth by half and still maintain comparable accuracy.

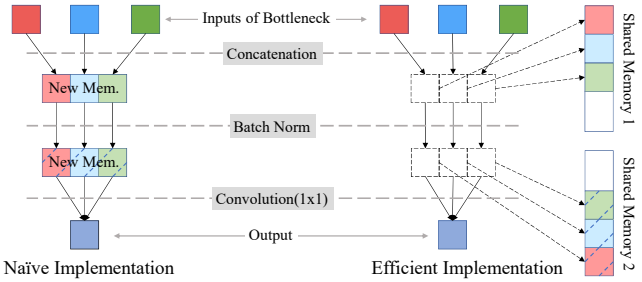

Fig. 6. Illustration of memory efficient implementation. It is for the Concat-BN-ReLU-Conv(1×1) in each bottleneck structure. ReLU is not shown since it is an in-place operation with no additional memory request. The efficient implementation pre-allocates two fixed memory space to store the concatenated and normalized features of connected blocks. In contrast, the naive implementation always allocates new memories for them, causing high memory consumption.

for the first pass and update the network using the gradient descent. The second batch is forwarded in two passes and the gradient descent is only applied on the second pass.

In this iterative pipeline, the CU-Net has two groups of supervisions in the first and second iterations. Both the detection supervision (binary cross entropy loss) and regression (mean squared error) supervision [26] are already used in the landmark detection tasks. However, there is no investigation of how they compare with each other. To this end, we try different combinations of detection and regression supervisions for two iterations. Our comparison could give some guidance for future research.

3.5 Quantization of Parameter, Feature and Gradient

Apart from shrinking the parameter number, we also investigate to quantize each parameter, intermediate features and gradients. Quantizing the parameters could improve the efficiency in both training and inference. For a parameter w_i in a convolutional filter W, we could binarize a parameter through the sign function:

$$q(w_i) = sign(clip(w_i, -1, 1)), \tag{5}$$

where $clip$ is a saturation function that clips parameter w_i to [-1, 1]. Or we ternarize w_i by a threshold-based function:

$$q(w_i) = \begin{cases} +1, & w_i > t \\ +1, & |w_i| \leq t \\ -1, & w_i < -t \end{cases}, \tag{6}$$

where $t \approx \frac{0.7}{n} \sum_{i=1}^{n} |w_i|$ is a positive threshold. As XNOR-Net [22], we also try to use a scaling factor α to approximate the real-value weight.

Quantizing the intermediate features and gradients, i.e., the dataflow, could boost the training efficiency by reducing the training memory. We follow the WAGE [21]. The dataflow is quantized to k-bit values by the following linear mapping function:

$$q(x, k) = clip(\sigma(k) \cdot round(x\sigma(k)) - 1 + \sigma(k), 1 - \sigma(k)) \tag{7}$$

where k is the pre-defined bit-width and $\sigma(k) = \frac{1}{2^{k-1}}$ is the unit distance function. In the following experiments, we explore different combinations of bit-widths to balance the accuracy and training memory consumption.

4 Implementation

In this section, we give the detailed architecture of CU-Net and the implementation of $order$-K coupling through the queue data structure. Besides, a memory efficient implementation for the shortcut connections is also described.

4.1 CU-Net Architecture

In the original stacked U-Nets, there is mainly one feature flow going through each U-Net. The coupled U-Nets still have a main feature flow along the U-Nets cascade. Let m denote the feature number in the main flow and n represent the generated feature number at each semantic block of U-Net, i.e. red block in Figure 4. The generated features are forwarded through the shortcut connections.

Before entering a red semantic block, the main feature flow and the shortcut feature flow are merged by channel-wise concatenation. For each top-down block of the $i^{th} (i \geq 0)$ U-Net, its inputs contain the m features in the main flow and another $n \times min(i, K)$ features from the shortcut connections of previous K U-Nets, where $min(i, K)$ indicates the lesser one of the order K and i. They are concatenated channel-wise to $m + n \times min(i, K)$ features. For each bottom-up block of the $i^{th} (i \geq 0)$ U-Net, its inputs include additional m shortcut features from the corresponding top-down blocks. Thus, its inputs have $2m + n \times min(i, K)$ features.

Then a 1×1 convolution compresses the input features to $4 \times n$ features. A following 3×3 convolution produces n new features. At last, the $m + n \times min(i, K)$ (top-down block) or $2m + n \times min(i, K)$ (bottom-up block) input features and the n generated features are concatenated. Another 1×1 convolution compresses them to m output features, flowing into the next block.

4.2 $Order$-K Coupling Implementation

The $order$-K coupling describes the connectivity for the whole U-Net. Inside a U-Net, each semantic block has $order$-K connectivity. That is, the $order$-K coupling consists of the $order$-K connectivity. Therefore, we only need to implement the $order$-K connectivity.

A shortcut connection means one feature reuse. In the $order$-K connectivity, a semantic block would use the features generated by the same semantic blocks in the preceding

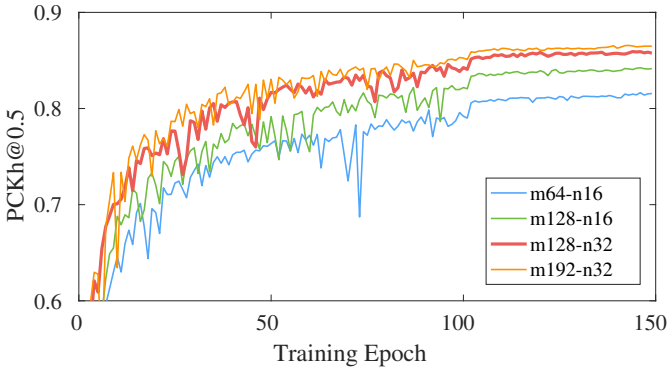

Fig. 7. Validation PCKh curves of 2 coupled U-Nets(CU-Net-2) under different hyper-parameters m and n. The converged curve reaches higher for larger m and n. But the gap between adjacent curves becomes smaller. $m = 128$ and $n = 32$ is a good trade-off of accuracy and efficiency. Besides, Larger m and n also make the curve smoother.

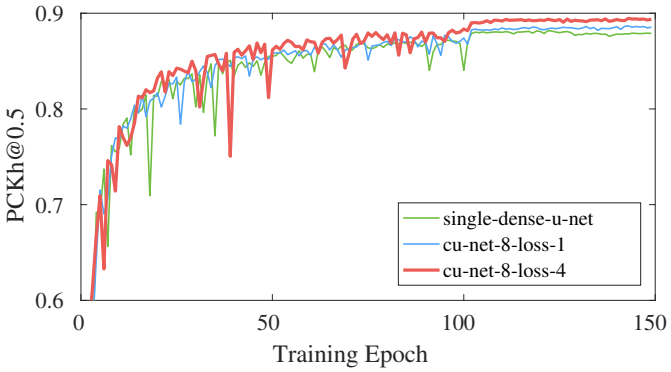

Fig. 8. Validation PCKh curves of single dense U-Net and 8 coupled U-Nets (CU-Net-8) with 1 and 4 supervisions. They have equivalent amounts of parameters. The CU-Net-8 get higher converged PCKh than the single dense U-Net. The additional intermediate supervisions bring more PCKh gains. But its curve fluctuates more before the convergence.

K U-Nets. Thus, for the current U-Net, we need to store the history features only from the preceding K U-Nets. As the U-Nets are used sequentially one-by-one, a length-K history sliding window also moves forward.

We use the dynamic queue structure to simulate this process. More specifically, we assign one queue for the same semantic blocks in all U-Nets. To save the memory, a queue only stores the references to the features instead of the feature copies. If the current U-Net index is less than K, we keep appending the feature reference to the queue rear. Otherwise, we first remove the old reference from the queue front and then append a new reference to the rear. The feature reference at the front is the least recent. Therefore, it is removed with the highest priority.

4.3 Memory Efficient Implementation

Benefitting from the $order$-K coupling, the CU-Net is quite parameter efficient. However, a naive implementation would prevent from training very deep CU-Net. Because the concatenation operation produces features detached from its inputs. In other words, the concatenated shortcut features require new space. Thus, the shortcut connections could increase the memory demand.

To reduce the training memory, we follow the efficient implementation [8]. More specifically, concatenation operations of the same semantic blocks in all U-Nets share a memory allocation and their subsequent batch norm operations share another memory allocation. Suppose a CU-Net includes N U-Nets each of which has L top-down blocks and L bottom-up blocks. We need to pre-allocate two memory space for each of $2L$ semantic blocks. For the ℓ^{th} top-down blocks, the concatenated features $[\mathbf{x}_{\ell-1}^1, \mathbf{X}_\ell^0], \cdots, [\mathbf{x}_{\ell-1}^{N-1}, \mathbf{X}_\ell^{N-2}]$ share the same memory space. Similarly, the concatenated features $[\mathbf{y}_{\ell-1}^0, \mathbf{x}_\ell^0], [\mathbf{y}_{\ell-1}^1, \mathbf{x}_\ell^1, \mathbf{Y}_\ell^0], \cdots, [\mathbf{y}_{\ell-1}^{N-1}, \mathbf{x}_\ell^{N-1}, \mathbf{Y}_\ell^{N-2}]$ in the ℓ^{th} bottom-up blocks share the same memory space.

In one shared memory allocation, later produced features would overlay the former features. Thus, the concatenations and their subsequent batch norm operations require to be recomputed in the backward phase. Figure 6 illustrates naive and efficient implementations.

5 Experiments

In this section, we evaluate each component in the proposed method. First, we select the hyper-parameters and evaluate the intermediate supervisions in the CU-Net. Then we compare the CU-Net with the single dense U-Net to show its accuracy superiority. After that, we evaluate the $order$-K coupling with the intermediate supervisions. Then we show the $order$-1 CU-Net is much more parameter efficient than the stacked U-Nets [5]. We also evaluate the iterative refinement to halve CU-Net parameters and test the quantization of parameters, intermediate features, and gradients. Finally, we compare the quantized $order$-1 CU-Net with state-of-the-art methods in both human pose estimation and facial landmark localization. Some qualitative results are also shown in Figure 11.

Network. The input resolution is normalized to 256×256. Before the CU-Net, a Conv7 × 7 filter with stride 2 and a max pooling would produce 128 features with resolution 64×64. Hence, the maximum resolution of CU-Net is 64×64. Each block in CU-Net has a bottleneck structure as shown on the right side of Figure 1. At the beginning of each bottleneck, features from different connections are concatenated and stored in shared memory. Then the concatenated features are compressed by the Conv1 × 1 to $4m$ features. At last, the Conv3 × 3 further produces m new features. The batch norm and ReLU are used before the convolutions.

Training. We implement the CU-Net using the PyTorch. The CU-Net is trained by the optimizer RMSprop. When training human pose estimators, the initial learning rate is 2.5×10^{-4} which is decayed to 5×10^{-5} after 100 epochs. The whole training takes 200 epochs. The facial landmark localizers are easier to train. Also starting from 2.5×10^{-4}, its learning rate is divided by 5, 2 and 2 at epoch 30, 60 and 90 respectively. The above settings remain the same for quantized CU-Net. To match the pace of dataflow, we set the same bit-width for gradients and features. We quantize dataflows and parameters all over the CU-Net except for the first and last convolutional layers, since localization is a fine-grained task requiring high precision heatmaps.

Human Pose Datasets. We use two benchmark human pose estimation datasets: MPII Human Pose [44] and Leeds Sports Pose (LSP) [45]. The **MPII** is collected from YouTube

TABLE 1
PCKhs of the CU-Net with varied intermediate supervisions on the MPII validation set. CU-Net-2 denotes a CU-Net with 2 U-Nets. The intermediate supervisions lower the PCKh of CU-Net-2. However, it improves the PCKh of deeper networks CU-Net-4 and CU-Net-8. Deeper CU-Net requires more intermediate supervisions to get the highest PCKh. But full intermediate supervisions are not optimal.

	CU-Net-2		CU-Net-4				CU-Net-8			
# Supervisions	1	2	1	2	3	4	1	2	4	8
PCKh@0.5 (%)	86.0	85.8	87.6	88.1	88.0	87.8	88.6	89.3	89.5	89.0

TABLE 2
Comparison of different hyper-parameters m and n measured by the parameter number and the PCKh on the MPII validation set. We use 2 coupled U-Nets(CU-Net-2). The PCKh increase becomes less from the left to the right while the parameter number growly consistently. A good trade-off between the PCKh and parameter number is m=128 and n=32.

m	64	128	128	128	192	192
n	16	16	24	32	24	32
# Parameters	0.5M	1.0M	1.4M	1.9M	2.4M	2.9M
PCKh@0.5 (%)	81.6	84.2	85.6	86.0	86.3	86.6

TABLE 3
CU-Net $v.s.$ single dense U-Net on MPII validation set measured by PCKh(%) and parameter #. The ratio is parameter # divided by the corresponding baselines(stacked 4 and 8 U-Nets). The CU-Net can get higher PCKh than the dense U-Net given a few more parameters.

Method	PCKh	# Para.	Baseline	Ratio
Dense U-Net (8)	88.1	6.1M	25.5M	23.9%
CU-Net-8	89.5	7.9M	25.5M	31.0%
Dense U-Net (4)	87.1	2.9M	12.9M	22.5%
CU-Net-4	88.1	3.9M	12.9M	30.2%

videos with a broad range of human activities. It has 25K images and 40K annotated persons, which are split into a training set of 29K and a test set of 11K. Following [46], 3K samples are chosen from the training set as the validation set. Each person has 16 labeled joints. The **LSP** dataset contains images from many sports scenes. Its extended version has 11K training samples and 1K testing samples. Each person in LSP has 14 labeled joints. Since there are usually multiple people in one image, we crop around each person and resize it to 256x256. We also use scaling (0.75-1.25), rotation (-/+30) and random flip to augment the data.

Facial Landmark Datasets. The experiments of the facial landmark localization are conducted on the composite of HELEN, AFW, LFPW, and IBUG which are re-annotated in the 300-W challenge [47]. Each face has 68 landmarks. Following [48] and [35], we use the training images of HELEN, LFPW and all images of AFW, totally 3148 images, as the training set. The testing is done on the common subset (testing images of HELEN and LFPW), challenge subset (all images from IBUG) and their union. We use the provided bounding boxes from the 300-W challenge to crop faces. The same augmentations of scaling and rotation as in human pose estimation are applied.

Metric. We use the standard metrics in both human pose estimation and face alignment. Specifically, Percentage of Correct Keypoints (PCK) is used to evaluate approaches for human pose estimation. A human joint is correctly detected if the L2 distance between the detected and groundtruth points is within a certain threshold. The MPII and LSP datasets use 50% of the head segment length and 20% of the L2 distance between the left shoulder and right hip as their thresholds. And the normalized mean error (NME) is employed to measure localizing facial landmarks. It measures the normalized L2 distance between the predicted and groundtruth landmarks. Following the convention of 300-W challenge, we use the inter-ocular distance to normalize mean error. For network quantization, we propose the balance index to balance accuracy and efficiency.

5.1 Hyper-Parameter Selection

There are two important hyper-parameters in designing the CU-Net. One is the feature number m in the main feature stream. In the experiments, m remains the same when the feature map resolution changes. The other hyper-parameter is the generated feature number n in a block of U-Net. We have tried 6 combinations of m and n in 2 coupled U-Nets. Table 2 gives the PCKhs on the MPII validation set. Besides, we choose 4 from the 6 settings and show how their validation PCKhs change during the training process in Figure 7.

In Table 2, the smallest m and n are 64 and 16. We set the increments 64 and 8 for m and n. We could observe how the accuracy (PCKh) and the parameter number change along with the two hyper-parameters. First, the accuracy increases when m and n grow. Furthermore, the increase is 2.6%, 1.4%, 0.4%, 0.3% and 0.3% from the left to the right. The increase slows down. Similar phenomena could be observed in Figure 7. The training is more stable when m and n become larger according to the curves in Figure 7.

Besides, the parameter number also grows as m and n become larger. Moreover, the growths are 0.5M, 0.4M, 0.5M, 0.5M and 0.5M. The parameter growth remains consistent. We would like to select a model with high accuracy and low model complexity. Through balancing the accuracy and parameter number, we choose m=128 and n=32. We fix this setting in the following experiments.

5.2 Evaluation of Intermediate Supervisions

Generally, the supervision of a CU-Net is the supervision of its last U-Net. Since a CU-Net contains several U-Nets, we consider adding supervisions for preceding U-Nets. More specifically, we only add the supervision at the end of a U-Net. Fortunately, the coupling connections do not prevent us from doing this. Note that if the supervision number is smaller than the U-Net number, we distribute the supervisions as uniformly as possible. For example, if 2 supervisions exist in 4 coupled U-Nets, they are at the end of the second and fourth U-Nets.

Table 1 gives the PCKh comparison of CU-Net with different numbers of supervisions. For CU-Net-2, adding

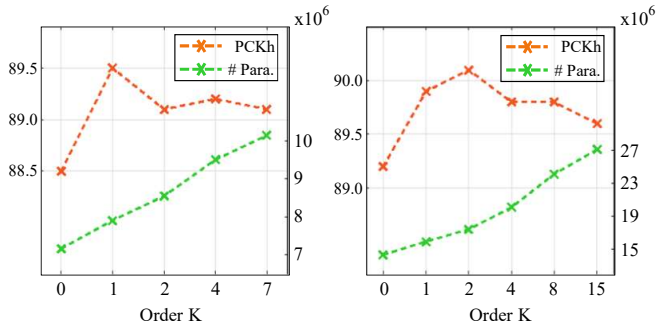

Fig. 9. Relation of PCKh(%), # parameters and $order$-K coupling on MPII validation set. The parameter number of CU-Net grows approximately linearly with the order of coupling. However, the PCKh first increases and then decreases. A small order 1 or 2 would be a good balance for prediction accuracy and parameter efficiency.

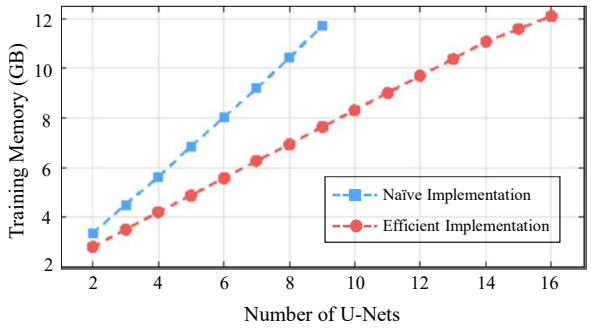

Fig. 10. Naive implementation $v.s.$ memory-efficient implementation. The $order$-1 coupling, batch size 16 and a 12GB GPU are used. The naive implementation can only support 9 U-Nets at most. In contrast, the memory-efficient implementation allows training 16 U-Nets, which nearly doubles the depth of CU-Net.

TABLE 4
$Order$-1 CU-Net-8 $v.s.$ $order$-7 CU-Net-8, measured by training and validation PCKhs(%) on MPII. $Order$-7 has higher training PCKh in all epochs. However, its validation PCKh is lower at last. Thus, $order$-7 with full intermediate supervisions overfits the training set a little bit.

PCKh on training set				
Epoch	1	50	100	150
$Order$-1 CU-Net-8	20.3	83.2	87.7	91.7
$Order$-7 CU-Net-8	**25.2**	**84.7**	**89.3**	**93.1**
PCKh on validation set				
Epoch	1	50	100	150
$Order$-1 CU-Net-8	29.4	82.8	**85.7**	**87.1**
$Order$-7 CU-Net-8	**36.6**	**84.0**	85.1	86.7

TABLE 5
$Order$-1 CU-Net $v.s.$ stacked residual U-Nets on MPII validation set measured by PCKh(%), parameter number, and inference time. With the same number of U-Nets, $Order$-1 CU-Net achieves comparable performance as stacked U-Nets. But it has only ∼30% parameters. The inference time is reduced by ∼30%, benefiting from fewer parameters.

Method	PCKh	# Para.	Ratio	Time(ms)	Ratio
Stacked U-Nets (16)	-	50.5M	100%	104.8	100%
CU-Net-16	89.9	15.9M	31.5%	70.8	67.6%
Stacked U-Nets (8)	89.3	25.5M	100%	48.9	100%
CU-Net-8	89.5	7.9M	31.0%	36.1	73.8%
Stacked U-Nets (4)	88.3	12.9M	100%	28.2	100%
CU-Net-4	88.2	3.9M	30.2%	18.9	67.0%

supervision for the first U-Net makes the validation PCKh drop by 0.2%. The coupling connections already strengthen the gradient propagation. The additional supervision makes the gradients too strong, leading to a little overfitting.

However, observations are different for more coupled U-Nets. According to Table 1, additional supervisions could improve the PCKh of 4 coupled U-Nets (CU-Net-4). The CU-Net-4 obtains the highest PCKh with 1 additional supervision. Similar results appear for the CU-Net-8. But 3 additional supervisions help it get the highest PCKh. CU-Net-4 and CU-Net-8 are much deeper than the CU-Net-2. Some intermediate supervisions could alleviate the gradient vanish problem. Thus, they help the CU-Net get better convergences. The CU-Net-8 is twice deeper than the CU-Net-4, thereby benefiting from more additional intermediate supervisions.

Both the intermediate supervisions and the coupling connections help train better CU-Net models. However, to obtain the highest accuracy, their amounts should have a balance. If one increases, the other needs to decrease. Since the intermediate supervisions require very few extra parameters, we would like to use full intermediate supervisions. That is, each U-Net in the CU-Net has one supervision. In this way, we could cut off some coupling connections to further reduce some parameters.

5.3 CU-Net $v.s.$ Single Dense U-Net

There are two ways of adding the shortcut connections in the U-Net. One is adding the dense connections in each block of a single U-Net, resulting in the single dense U-Net. The other is using the coupling connections in stacked U-Nets. Table 3 compares the PCKh and parameter number of CU-Net and single dense U-Net. For a fair comparison, the single dense U-Net and the CU-Net have the same number of conv3×3 layers. We add one layer in each dense block of the dense U-Net every time we increase one U-Net in the CU-Net.

According to Table 3, the CU-Net obviously outperforms the dense U-Net by 1.4% and 1.0% for the 4 and 8 U-Nets. We use the parameter numbers of stacked 4 and 8 U-Nets as the baselines since the main goal is to reduce the parameters of stacked U-Nets. Although the CU-Net has a few more parameters than the dense U-Net, the CU-Net is cost-effective. For example, the dense U-Net (8) and CU-Net-4 both increase 1.0% over the dense U-Net (4). However, they have 2.1× and 1.3× parameters of the dense U-Net (4).

We also show the validation PCKh curves of the 8 U-Nets setting in Figure 8. The converged PCKh curves of CU-Net and single dense U-Net have gaps. Besides, the CU-Net PCKh curve fluctuates more when adding the intermediate supervisions. Because additional supervisions can make the CU-Net parameters updated with larger steps in the training.

5.4 Evaluation of $Order$-K Coupling

The $order$-K coupling couples a U-Net only to its K successors. Each U-Net has its own independent supervision. In this experiment, we investigate how the PCKh and convolution parameter number change along with the order value. Figure 9 gives the results from MPII validation set. The left and

TABLE 6
NME(%) on 300-W using $order$-1 CU-Net-4 with iterative refinement, detection, and regression supervisions. Iterative refinement can lower errors and regression supervision outperforms detection supervision.

Method	Easy Subset	Hard Subset	Full Set	# Para.
Detection only	3.63	5.60	4.01	3.9M
Regression only	2.91	5.12	3.34	3.9M
Detection Detection	3.52	5.59	3.93	4.1M
Detection Regression	2.95	5.12	3.37	4.1M
Regression Regression	2.87	4.97	3.28	4.1M

TABLE 7
Iterative $order$-1 CU-Net-4 $v.s.$ non-iterative $order$-1 CU-Net-8 on 300-W measured by NME(%). Iterative CU-Net-4, with few additional parameters on CU-Net-4, achieves comparable performance as CU-Net-8. Thus, the iterative refinement has the potential to halve parameters of CU-Net but still maintain comparable performance.

Method	Easy Subset	Hard Subset	Full Set	# Parameters
CU-Net-4	2.91	5.12	3.34	3.9M
Iter. CU-Net-4	2.87	4.97	3.28	4.1M
CU-Net-8	2.82	5.07	3.26	7.9M

right figures show results of CU-Net with 8 and 16 U-Nets. It is clear that the convolution parameter number increases as the order becomes larger. However, the left and right PCKh curves have a similar shape of first increasing and then decreasing. The $order$-1 coupling is always better than the $order$-0 one.

However, high order may not be a good choice. Because we already use full intermediate supervisions. The balance of coupling and intermediate supervisions is broken for the high order ones. Too dense coupling make gradients accumulate too much, causing the overfitting. Further evidence of overfitting is shown in Table 4. The $order$-7 coupling has the higher training PCKh than $order$-1 in all training epochs. But its validation PCKh is a little lower in the last training epochs. Thus, we use $order$-1 coupling with full intermediate supervisions in the following experiments.

5.5 CU-Net $v.s.$ Stacked Residual U-Nets

The CU-Net is proposed to improve the parameter efficiency of stacked U-Nets with the residual modules. To validate this, we compare the $order$-1 CU-Net with the stacked residual U-Nets [5]. This experiment is done on the MPII validation set. Table 5 shows three pairs of comparisons with 4, 8 and 16 U-Nets. The PCKh, convolution parameter number and inference time are reported. The inference time refers to the time of forwarding one image under the testing mode. We compute the average inference time on the MPII validation set. For a fair comparison, we test all the models using the Torch toolbox and K40 GPU.

In Table 5, with the same number of U-Nets, the $order$-1 CU-Net could obtain comparable or even better accuracy. More importantly, the feature reuse across U-Nets make each U-Net in the CU-Net light-weighted. The parameter number and inference time of the stacked residual U-Nets decrease by $\sim$70% and $\sim$30%. Besides, the high parameter efficiency makes it possible to train $order$-1 CU-Net-16 in a 12G GPU with batch size 16.

5.6 Evaluation of Iterative Refinement

The iterative refinement is designed to reduce the parameters of CU-Net by one half. We validate the design in two steps. First, we verify adding an iteration on a CU-Net could improve the accuracy. The experiment is done on the 300-W dataset using $order$-1 CU-Net-4. Results are shown in Table 6. For both detection and regression supervisions, adding an iteration could lower the localization errors, demonstrating the effectiveness of iterative refinement. Meanwhile, the model parameters only increase 0.2M. Besides, the regression supervision outperforms the detection one no matter in the iterative or non-iterative setting, making it a better choice for the landmark localization.

The regression and detection supervisions have different groundtruths. The regression supervision draws a gaussian circle centered at a keypoint coordinate. The detection supervision needs to strictly divide a groundtruth heatmap into two areas: keypoint area and non-keypoint area. In practice, it is hard to specify an accurate area for a keypoint. The Gaussian groundtruth can alleviate this concern by decreasing the confidence scores gradually. We think the advantage of regression supervision comes from its more reasonable groundtruth heatmaps.

Second, we prove an iterative CU-Net could get comparable accuracy as a double-length CU-Net. More specifically, we compare iterative $order$-1 CU-Net-4 with $order$-1 CU-Net-8. Table 7 gives the comparison. We can find that the iterative CU-Net-4 could obtain comparable NME as CU-Net-8. However, the CU-Net-8 has double parameters of CU-Net-4, whereas the iterative CU-Net-4 has only 0.2M additional parameters.

5.7 Evaluation of Parameter and Dataflow Quantization

In this experiment, we study to quantize the parameters and dataflow, i.e., intermediate features and gradients. Through network quantization, high precision parameters and operations can be efficiently represented by a few discrete values. We try a series of bit-widths to find appropriate choices. We use the $order$-1 CU-Net-4 on the 300-W dataset and the $order$-1 CU-Net-2 on the MPII validation set. The results are shown in Tables 8, 9 and 10. BW and TW represent binarized weight and ternarized weight respectively. The suffixes α and QIG denote the float scaling factor α and quantized intermediate features and gradients.

Binary Parameters. According to Tables 8 and 9, the binarized parameters with the scaling factor α achieves PCKh 85.5% in human pose estimation and NME 3.58% in facial landmark localization. They are very close to the original 86.1% and 3.38%, without any quantization. Even without the scaling factor α, the decrease of PCKh and increase of NME are small. This indicates binarizing the CU-Net parameters does not affect much its localization accuracy. However, the model size is substantially decreased by 32$\times$.

Ternary Parameters. Ternary representation has one more bit than the binary one. Its stronger representation power could improve the accuracy of CU-Net. Based on Table 8, ternary parameters reduces the NME of binary parameters

TABLE 8
Performance of different combinations of bit-width values on the 300-W dataset measured by NME(%). All quantized networks are based on $order$-1 CU-Net-4. BW and TW are short for binarized and ternarized parameters, α represents float scaling factor, QFG is short for quantized intermediate features and gradients. Bit_F, Bit_P, Bit_G represents the bit-width of features, parameters, gradients respectively. Training memory and model size are represented by their compression ratios to those of the original CU-Net-4. The balance index is calculated by Equation 8. The CU-Net-4-BW-α gets the lowest error. Considering together accuracy, training memory and model size, the CU-Net-4-BW-α(818) has the smallest balance index.

Method	Bit_I	Bit_W	Bit_G	NME(%) Full set	NME(%) Easy set	NME(%) Hard set	Training Memory	Model Size	Balance Index
CU-Net-4	32	32	32	3.38	2.95	5.13	1.00	1.00	11.4
BW-QIG	6	1	6	5.93	5.10	9.34	0.17	0.03	0.18
BW-QIG	8	1	8	4.30	3.67	6.86	0.25	0.03	**0.14**
BW-α-QIG	8	1	8	4.47	3.75	7.40	0.25	0.03	0.15
BW	32	1	32	3.75	3.20	5.99	1.00	0.03	0.42
BW-α	32	1	32	**3.58**	3.12	5.45	1.00	0.03	0.38
TW	32	2	32	3.73	3.21	5.85	1.00	0.06	0.83
TW-QIG	6	2	6	4.27	3.70	6.59	0.17	0.06	0.19
TW-QIG	8	2	8	4.13	3.55	6.50	0.25	0.06	0.26

TABLE 9
Performance of different quantization configurations for $order$-1 CU-Net-2 on the MPII validation dataset measured by PCKh(%), training memory, model size, and balance index. The CU-Net-2-BW-α gets the highest accuracy. Considering together accuracy, training memory and model size, the CU-Net-2-BW-α(818) has the smallest balance index.

Method	Bit_F	Bit_P	Bit_G	PCKh (%)	Training Memory	Model Size	Balance Index
CU-Net-2	32	32	32	86.1	1.00	1.00	193
BW-QFG	6	1	6	62.7	0.17	0.03	7.10
BW-QFG	8	1	8	81.5	0.25	0.03	**2.57**
BW	32	1	32	84.7	1.00	0.03	6.84
BW-α	32	1	32	**85.5**	1.00	0.03	7.97
TW	32	2	32	84.9	1.00	0.06	14.0
TW-QFG	6	2	6	74.0	0.17	0.06	6.90
TW-QFG	8	2	8	81.7	0.25	0.06	5.02

TABLE 10
Detailed PCKh comparison of different quantization configurations for $order$-1 CU-Net-2 on MPII validation sets. The parameter binarization or ternarization have small influence on the accuracy of individual human joints. But the quantization of intermediate features and gradients lowers the accuracy of challenging human joints: elbow, wrist, ankle and knee.

Method	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Mean
CU-Net-2	96.1	94.5	86.7	81.1	86.9	81.0	76.3	86.1
CU-Net-2+BW-α	95.9	94.4	86.6	80.3	86.5	79.5	75.2	85.5
CU-Net-2+BW	96.3	94.2	85.2	79.1	85.8	78.2	74.3	84.7
CU-Net-2+BW-QIG(818)	95.6	92.4	82.0	74.7	82.6	74.8	68.6	81.5
CU-Net-2+BW-QIG(616)	87.7	77.4	61.8	49.8	64.2	50.8	47.2	62.7
CU-Net-2+TW	96.2	93.9	85.7	79.6	85.8	79.1	74.1	84.9
CU-Net-2+TW-QIG(828)	95.4	92.6	82.1	75.2	83.0	74.9	68.7	81.7
CU-Net-2+TW-QIG(626)	94.2	87.2	73.4	65.1	76.0	64.7	57.6	74.0

by 0.02%. With the quantization of features and gradients, we can observe more obvious NME decreases 1.66% and 0.17%. Consistent changes on the PCKh can be found in Table 9.

Features and Gradients Quantization. Quantizing the intermediate features and gradients of CU-Net could largely reduce the training memory, improving the training efficiency. We try 6-bits and 8-bits with either the binary or ternary parameters. The 8-bits quantization is obviously better than the 6-bits one, especially for the binary parameters. The 8-bits quantization with binary parameters decreases the PCKh by 4.6% and increases the NME by 0.92%. However, the training memory is significantly reduced by 75%. For mobile devices with limited computational resources, slightly performance drop is tolerable provided the large efficiency enhancement.

Balancing Accuracy and Quantization. The quantization of parameters and dataflow could significantly increase the testing and training efficiency but at the cost of some accuracy decrease. Thus, we need a trade-off between them. To this end, we propose the balance index (BI) in Equation 8 to balance the quantization and accuracy.

$$BI = E^2 \cdot TM \cdot MS \tag{8}$$

where TM and MS are short for training memory and model size. Instead of using their raw values, we use their ratios to those without any quantization. E denotes the error in the landmark localization. We set E as the NME for facial landmark localization while 1-PCKh for human pose estimation. The E^2 is calculated in the above formula to emphasize the prior importance of accuracy. Smaller BI indicates better balance.

In Tables 8 and 9, BW-α-QIG(818) gets the smallest BI. Its NME increases only 0.92% and its PCKh decreases only 4.6%. However, it reduces 4$\times$ training memory and 32$\times$ model size. It has the best balance for accuracy and efficiency.

Quantization Impact on Individual Human Joints. In addition to the average PCKhs In Table 9, we also give the PCKhs of individual human joints under various quantization settings in Table 10. The parameter binarization does not affect much the joint accuracy. However, the quantization of intermediate features and gradients causes obvious decreases of challenging joints like wrist, knee, and ankle. This means that, although the parameter quantization does not lose much useful information, the parameter update still requires high precision representations. A possible solution is to explore better input and gradient quantization strategies.

5.8 Evaluation of Memory Efficient Implementation

The memory-efficient implementation makes it possible to train very deep CU-Net. Figure 10 shows the training memory consumption of both naive and memory-efficient implementations of $order$-1 CU-Net. The linear growths of training memory along with number of U-Nets is because of the fixed order coupling. But the memory growth of efficient implementation is much slower than that of the naive one.

TABLE 11
Comparison of convolution parameter number (Million), model size (Megabyte) and inference time (millisecond) with state-of-the-art methods. The CU-Net-8 can largely reduce ∼69%-∼86% parameter number and ∼26%-∼86% inference time.

Method	Yang *et al.* [32]	Wei *et al.* [24]	Chu *et al.* [7]	Newell *et al.* [5]	$Order$-1 CU-Net-8	$Order$-1 CU-Net-16
# Parameters (M)	28.0	29.7	58.1	25.5	**7.9**	15.9
Inference Time (ms)	137.7	112.4	251.0	48.9	**36.1**	70.8

TABLE 12
NME(%) comparison with state-of-the-art facial landmark localization methods on 300-W dataset. The CU-Net-BW-α refers to the CU-Net with binarized parameters and scaling factor α. It obtains comparable error with state-of-the-art method [5]. But it has ∼**50**× smaller model size.

Method	CFAN [39]	Deep Reg [49]	CFSS [48]	TCDCN [34]	DDN [50]	MDM [51]	TSR [35]	HGs(4) [5]	$Order$-1 CU-Net-8	$Order$-1 CU-Net-8-BW-α
Easy subset	5.50	4.51	4.73	4.80	-	4.83	4.36	2.90	**2.82**	3.00
Hard subset	16.78	13.80	9.98	8.60	-	10.14	7.56	5.15	**5.07**	5.36
Full set	7.69	6.31	5.76	5.54	5.59	5.88	4.99	3.35	**3.26**	3.46

TABLE 13
PCKh(%) comparison on MPII test sets. The CU-Net-BW-α refers to the CU-Net with binary parameters and scaling factor α. The $order$-1 CU-Net-16-BW-α could achieve comparable accuracy. More importantly, it has ∼**2%** model size compared with other state-of-the-art approaches.

Method	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Mean
Pishchulin *et al.* [52]	74.3	49.0	40.8	34.1	36.5	34.4	35.2	44.1
Tompson *et al.* [53]	95.8	90.3	80.5	74.3	77.6	69.7	62.8	79.6
Carreira *et al.* [25]	95.7	91.7	81.7	72.4	82.8	73.2	66.4	81.3
Tompson *et al.* [46]	96.1	91.9	83.9	77.8	80.9	72.3	64.8	82.0
Hu *et al.* [54]	95.0	91.6	83.0	76.6	81.9	74.5	69.5	82.4
Pishchulin *et al.* [27]	94.1	90.2	83.4	77.3	82.6	75.7	68.6	82.4
Lifshitz *et al.* [29]	97.8	93.3	85.7	80.4	85.3	76.6	70.2	85.0
Gkioxary *et al.* [55]	96.2	93.1	86.7	82.1	85.2	81.4	74.1	86.1
Rafi *et al.* [56]	97.2	93.9	86.4	81.3	86.8	80.6	73.4	86.3
Belagiannis *et al.* [30]	97.7	95.0	88.2	83.0	87.9	82.6	78.4	88.1
Insafutdinov *et al.* [28]	96.8	95.2	89.3	84.4	88.4	83.4	78.0	88.5
Wei *et al.* [24]	97.8	95.0	88.7	84.0	88.4	82.8	79.4	88.5
Bulat *et al.* [26]	97.9	95.1	89.9	85.3	89.4	85.7	81.7	89.7
Newell *et al.* [5]	98.2	96.3	91.2	87.1	90.1	87.4	83.6	90.9
Chu *et al.* [7]	**98.5**	96.3	91.9	**88.1**	**90.6**	**88.0**	**85.0**	**91.5**
$Order$-1 CU-Net-8	97.4	96.2	91.8	87.3	90.0	87.0	83.3	90.8
$Order$-1 CU-Net-16	97.4	**96.4**	**92.1**	87.7	90.2	87.7	84.3	91.2
+BW+α	97.6	96.4	91.7	87.3	90.4	87.3	83.8	91.0

TABLE 14
PCK(%) comparison on LSP test set. The CU-Net-BW-α refers to the CU-Net with binary parameters and scaling factor α. The $order$-1 CU-Net-16-BW-α could obtain comparable state-of-the-art accuracy. But it has ∼**50**× smaller model size than other state-of-the-art methods.

Method	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Mean
Belagiannis *et al.* [30]	95.2	89.0	81.5	77.0	83.7	87.0	82.8	85.2
Lifshitz *et al.* [29]	96.8	89.0	82.7	79.1	90.9	86.0	82.5	86.7
Pishchulin *et al.* [27]	97.0	91.0	83.8	78.1	91.0	86.7	82.0	87.1
Insafutdinov *et al.* [28]	97.4	92.7	87.5	84.4	91.5	89.9	87.2	90.1
Wei *et al.* [24]	97.8	92.5	87.0	83.9	91.5	90.8	89.9	90.5
Bulat *et al.* [26]	97.2	92.1	88.1	85.2	92.2	91.4	88.7	90.7
Chu *et al.* [7]	98.1	93.7	89.3	86.9	93.4	94.0	92.5	92.6
Newell *et al.* [5]	98.2	94.0	91.2	87.2	93.5	**94.5**	92.6	93.0
Yang *et al.* [32]	**98.3**	94.5	92.2	88.9	**94.4**	95.0	93.7	93.9
$Order$-1 CU-Net-8	97.1	94.7	91.6	89.0	93.7	94.2	93.7	93.4
$Order$-1 CU-Net-16	97.5	**95.0**	**92.5**	**90.1**	93.7	**95.2**	94.2	**94.0**
+BW+α	97.8	94.3	91.8	89.3	93.1	94.9	**94.4**	93.6

With batch size 16, we could train the CU-Net-16 in one 12GB GPU. Under the same setting, the naive implementation could support only CU-Net-9.

5.9 Comparison with State-of-the-art Methods

We compare the CU-Net with state-of-the-art approaches for both human pose estimation and facial landmark localization. Both the efficiency and accuracy are compared.

Efficiency. Table 11 compares the CU-Net with state-of-the-art methods in terms of efficiency. The CU-Net-8 has only ∼14%-∼31% parameter number of them. Fewer parameters can usually accelerate the inference speed. According to Table 11, the CU-Net-8 uses ∼74% time of Newell *et. al* [5] (stacked 8 U-Nets). Yang *et. al* [32] and Chu *et. al* [7] use more sophisticated modules and graphical models based on Newell *et. al* [5], resulting in much higher time cost. Wei *et. al* [24] also has high time expense mainly due to its larger input resolution and convolution kernel size. The CU-Net-16 consumes more time than Newell *et. al* [5], albeit its fewer parameters. Because the CU-Net-16 has double depth of Newell *et. al* [5]. However, the CU-Net-16 uses only ∼28%-∼63% time of other more complex methods.

Besides, the network quantization can also improve the model efficiency. For example, the parameter binarization can reduce the model size by ∼32×. The parameter binarization and feature quantization can bring ∼2×-∼58× speedup according to the theoretical analysis of [22]. With the binarized parameters, the convolutions only have the addition and subtraction without the multiplication operations, resulting in ∼2× speedup. If the features are also binarized, it can achieve ∼58× speedup with the bit-counting operations [18]. Generally, measuring the inference time of the quantized networks requires special software and hardware supports. We only provide a theoretical analysis here due to our resource limitations.

Accuracy. Tables 12, 13 and 14 show accuracy comparisons on both facial landmark localization and human pose estimation. Despite its high efficiency, the CU-Net can still achieve obtain comparable state-of-the-art accuracy on the benchmark 300-W, MPII and LSP test sets.

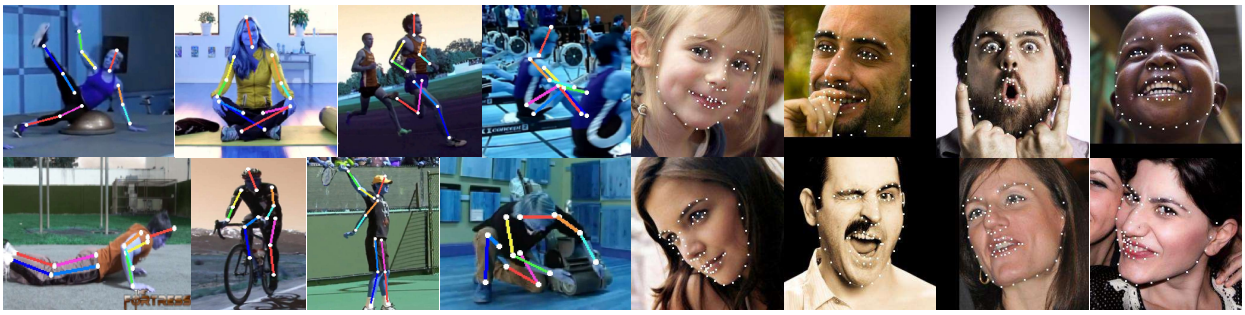

Fig. 11. Qualitative results of human pose estimation and facial landmark localization. The quantized CU-Net could handle a wide range of human poses, even with occlusions. It could also detect accurate facial landmarks with various head poses and expressions.

6 Conclusion

We design the efficient CU-Net. It connects blocks with the same semantic meanings in stacked U-Nets. The *order-K* coupling and iterative refinement are introduced to further improve the parameter efficiency. We also study the quantization of parameters, intermediate features and gradients. Experiments on both human pose estimation and facical landmark localization show that the CU-Net could achieve comparable state-of-the-art accuracy but with only ∼30% parameters, ∼70% inference time, ∼2% model size and ∼25% training memory.

7 Acknowledgment

This work was partially supported by the National Science Foundation under grants: Cyber-Human Systems (Grants #1703883 and #1763523), INDUSTRY/UNIV COOP RES CENTERS CARTA (Grant #1747778), Algorithms in the Field (Grant #1733843), and also partially funded by Multidisciplinary University Research Initiatives (MURI #W911NF1610342).

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.
[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.
[3] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *CVPR*, 2014.
[4] X. Xiong and F. De la Torre, "Supervised descent method and its applications to face alignment," in *CVPR*, 2013.
[5] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016.
[6] X. Peng, Z. Tang, F. Yang, R. S. Feris, and D. Metaxas, "Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation," in *CVPR*, 2018.
[7] X. Chu, W. Yang, W. Ouyang, C. Ma, A. Yuille, and X. Wang, "Multi-context attention for human pose estimation," in *CVPR*, 2016.
[8] G. Pleiss, D. Chen, G. Huang, T. Li, L. M., and K. Q. Weinberger, "Memory-efficient implementation of densenets," *arXiv*, 2017.
[9] Z. Tang, X. Peng, S. Geng, L. Wu, S. Zhang, and D. Metaxas, "Quantized densely connected u-nets for efficient landmark localization," in *ECCV*, 2018.
[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.
[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv*, 2014.
[12] R. K. Srivastava, K. Greff, and J. S., "Training very deep networks," in *NIPS*, 2015.
[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[14] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *CVPR*, 2017.
[15] X. Li, H. Chen, X. Qi, Q. Dou, C.-W. Fu, and P. A. Heng, "H-denseunet: Hybrid densely connected unet for liver and liver tumor segmentation from ct volumes," *arXiv*, 2017.
[16] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *CVPRW*, 2017.
[17] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *CVPR*, 2018.
[18] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv*, 2016.
[19] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv*, 2016.
[20] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv*, 2016.
[21] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," in *ICLR*, 2018.
[22] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, 2016.
[23] A. Bulat and G. Tzimiropoulos, "Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources," in *ICCV*, 2017.
[24] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.
[25] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," in *CVPR*, 2016.
[26] A. Bulat and G. Tzimiropoulos, "Human pose estimation via convolutional part heatmap regression," in *ECCV*, 2016.
[27] L. Pishchulin, E. Insafutdinov, S. Tang, B. A., M. A., P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in *CVPR*, 2016.
[28] E. Insafutdinov, L. Pishchulin, B. Andres, M. And., and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in *ECCV*, 2016.
[29] I. Lifshitz, E. Fetaya, and S. Ullman, "Human pose estimation using deep consensus voting," in *ECCV*, 2016.
[30] V. Belagiann. and A. Zisserman, "Recurrent human pose estimation," in *FG*, 2017.
[31] L. Zhao, X. Peng, Y. Tian, M. Kapadia, and D. Metaxas, "Learning to forecast and refine residual motion for image-to-video generation," in *ECCV*, 2018.
[32] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, "Learning feature pyramids for human pose estimation," in *ICCV*, 2017.
[33] Y. Chen, C. Shen, X.-S. Wei, L. Liu, and J. Yang, "Adversarial posenet: A structure-aware convolutional network for human pose estimation," in *ICCV*, 2017.
[34] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Facial landmark detection by deep multi-task learning," in *ECCV*, 2014.
[35] J. Lv, X. Shao, J. Xing, C. Cheng, and X. Zhou, "A deep regression architecture with two-stage re-initialization for high performance facial landmark detection," in *CVPR*, 2017.
[36] X. Peng, R. S. Feris, X. Wang, and D. N. Metaxas, "A recurrent encoder-decoder network for sequential face alignment," in *ECCV*, 2016.

[37] F. Liu, Q. Zhao, X. Liu, and D. Zeng, "Joint face alignment and 3d face reconstruction with application to face recognition," *ArXiv*, 2017.
[38] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *CVPR*, 2013.
[39] J. Zhang, S. Shan, M. Kan, and X. Chen, "Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment," in *ECCV*, 2014.
[40] S. Zafeiriou, G. Trigeorgis, G. Chrysos, J. Deng, and J. Shen, "The menpo facial landmark localisation challenge: A step towards the solution," in *CVPRW*, 2017.
[41] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.
[42] X. Glorot, A. B., and Y. B., "Deep sparse rectifier neural networks," in *AISTAT*, 2011.
[43] R. Begleiter, R. El-Yaniv, and G. Yona, "On prediction using variable order markov models," *Journal of Artificial Intelligence Research*, 2004.
[44] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *CVPR*, 2014.
[45] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation," in *BMVC*, 2010.
[46] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. B., "Efficient object localization using convolutional networks," in *CVPR*, 2015.
[47] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *ICCVW*, 2013.
[48] S. Zhu, C. Li, C. Change Loy, and X. Tang, "Face alignment by coarse-to-fine shape searching," in *CVPR*, 2015.
[49] B. Shi, X. Bai, W. Liu, and J. Wang, "Deep regression for face alignment," *arXiv*, 2014.
[50] X. Yu, F. Zhou, and M. Chandraker, "Deep deformation network for object landmark localization," in *ECCV*, 2016.
[51] G. Trigeorgis, P. Snape, M. A. Nicolaou, E. A., and S. Zafeiriou, "Mnemonic descent method: A recurrent process applied for end-to-end face alignment," in *CVPR*, 2016.
[52] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele, "Strong appearance and expressive spatial models for human pose estimation," in *ICCV*, 2013.
[53] J. 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.
[54] P. Hu and D. Ramanan, "Bottom-up and top-down reasoning with hierarchical rectified gaussians," in *CVPR*, 2016.
[55] G. Gkioxari, A. Toshev, and N. Jaitly, "Chained predictions using convolutional neural networks," in *ECCV*, 2016.
[56] U. Rafi, B. Leibe, J. Gall, and I. Kostrikov, "An efficient convolutional network for human pose estimation." in *BMVC*, 2016.

Zhiqiang Tang is a Ph.D. candidate in Computer Science at Rutgers University, NJ, USA. His advisor is Prof. Dimitris N. Metaxas. He received his B.S. degree in Computer Science from Hebei University of Technology, Tianjin, China in 2012. His research interests lie in machine learning, deep learning and their applications in computer vision.

Xi Peng is an Assistant Professor with the Department of Computer Science, Binghamton University (NY, USA). He is also a Visiting Professor with the Department of Computer Science, Rutgers University (NJ, USA). He received the Ph.D. degree in Computer Science from Rutgers University (NJ, USA, 2018), the M.S. degrees from Chinese Academy of Sciences (Beijing, China, 2011), and the B.E. degree from Beihang University (Beijing, China, 2008). His research focuses on structural and model-oriented deep learning, explainable machine learning, and intelligent data analysis.

Kang Li received the Ph.D. degree in Mechanical Engineering from University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2009. He is an Associate Professor with the Department of Orthopaedics, New Jersey Medical School (NJMS), Rutgers University, Newark, NJ, USA, and a graduate faculty member of the Department of Computer Science, Rutgers University. He is also a Visiting Professor with the School of Mechatronics Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include AI in healthcare, musculoskeletal biomechanics, medical imaging, healthcare engineering, design and biorobotics, and human factors/ergonomics.

Dimitris N. Metaxas received the B.E. degree from the National Technical University of Athens, Athens, Greece, in 1986, the M.S. degree from the University of Maryland, College Park, MD, USA, in 1988, and the Ph.D. degree from the University of Toronto, Toronto, ON, Canada, in 1992. He is a Distinguished Professor with the Computer Science Department, Rutgers University, New Brunswick, NJ, USA, where he is directing the Computational Biomedicine Imaging and Modeling Center. His current research interests include the development of formal methods upon which computer vision, computer graphics, and medical imaging can advance synergistically.