Hierarchical binary CNNs for landmark localization with limited resources

Adrian Bulat and Georgios Tzimiropoulos

Abstract—Our goal is to design architectures that retain the groundbreaking performance of Convolutional Neural Networks (CNNs) for landmark localization and at the same time are lightweight, compact and suitable for applications with limited computational resources. To this end, we make the following contributions: (a) we are the first to study the effect of neural network binarization on localization tasks, namely human pose estimation and face alignment. We exhaustively evaluate various design choices, identify performance bottlenecks, and more importantly propose multiple orthogonal ways to boost performance. (b) Based on our analysis, we propose a novel hierarchical, parallel and multi-scale residual architecture that yields large performance improvement over the standard bottleneck block while having the same number of parameters, thus bridging the gap between the original network and its binarized counterpart. (c) We perform a large number of ablation studies that shed light on the properties and the performance of the proposed block. (d) We present results for experiments on the most challenging datasets for human pose estimation and face alignment, reporting in many cases state-of-the-art performance. (e) We further provide additional results for the problem of facial part segmentation. Code can be downloaded from https://www.adrianbulat.com/binary-cnn-landmarks

Index Terms—Binary Convolutional Neural Networks, Residual learning, Landmark localization, Human pose estimation, Face alignment.

1 Introduction

This work is on localizing a predefined set of fiducial points on objects of interest which can typically undergo non-rigid deformations like the human body or face. Very recently, work based on Convolutional Neural Networks (CNNs) has revolutionized landmark localization, demonstrating results of remarkable accuracy even on the most challenging datasets for human pose estimation [1], [2], [3] and face alignment [4]. However, deploying (and training) such methods is computationally expensive, requiring one or more high-end GPUs, while the learned models typically require hundreds of MBs, thus rendering them completely unsuitable for real-time or mobile applications. This work is on highly accurate and robust yet efficient and lightweight landmark localization using binarized CNNs.

Our work is inspired by recent results of binarized CNN architectures on image classification [5], [6]. Contrary to these works, we are the first to study the effect of neural network binarization on fine-grained tasks like landmark localization. Similarly to [5], [6], we find that binarization results in performance drop, however to address this we opted to investigate and propose several architectural innovations which led to the introduction of a novel hierarchical, parallel and multi-scale residual block, as opposed to investigating ways to improve the binarization process as proposed in [5], [6]. In summary, our main methodological contributions are:

- We are the first to study the effect of binarization on state-of-the-art CNN architectures for the problem of localization, namely human pose estimation and face alignment. To this end, we exhaustively evaluate various design choices, and identify performance bottlenecks. More importantly, we describe multiple orthogonal ways to boost performance; see Subsections 4.2, 4.3 and 4.4.
- Based on our analysis, we propose a new hierarchical, parallel and multi-scale residual architecture (see Subsection 4.5) specifically designed to work well for the binary case. Our block results in large performance improvement over the baseline binary

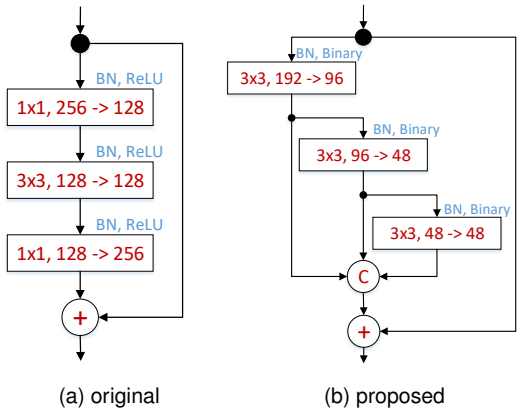


Fig. 1. (a) The original bottleneck layer of [7]. (b) The proposed hierarchical parallel & multi-scale structure: our block increases the receptive field size, improves gradient flow, is specifically designed to have (almost) the same number of parameters as the original bottleneck, does not contain 1×1 convolutions, and in general is derived from the perspective of improving the performance and efficiency for binary networks. **Note:** a layer is depicted as a rectangular block containing: its filter size, the number of input and output channels; "C" - denotes concatenation and "+" an element-wise sum.

- *A. Bulat and G. Tzimiropoulos are with the School of Computer Science, University of Nottingham.*
E-mail: {adrian.bulat, yorgos.tzimiropoulos}@nottingham.ac.uk

Manuscript received April 19, 2005; revised August 26, 2015.

residual block of [7] (about 6% in absolute terms when the same number of parameters are used (see Subsection 4.6.1, Table 2)). Fig. 1 provides a comparison between the baseline residual block of [7] and the one proposed in this work.
- We investigate the effectiveness of more advanced extensions of the proposed block (see Section 7) and improved network architectures including network stacking (see Section 8).

Further experimental contributions include:

- While our newly proposed block was developed with the goal of improving the performance of binary networks, we also show that the performance boost offered by the proposed architecture also generalizes to some extent for the case of real-valued networks (see Subsection 4.6.2).
- We perform a large number of ablation studies that shed light on the properties and the performance of the proposed block (see Sections 4.6 and 6).
- We present results for experiments on the most challenging datasets for human pose estimation and face alignment, reporting in many cases state-of-the-art performance (see Section 6).
- We further provide additional results for the problem of facial part segmentation (see Section 9).

Compared to our previous work in [8], this paper investigates the effectiveness of more advanced binary architectures (both at block and network level), provides a more in-depth analysis of the proposed methods and results (including more qualitative ones) and additionally includes the aforementioned experiment on facial part segmentation.

2 Closely Related Work

This Section reviews related work on network quantization, network design, and gives an overview of the state-of-the-art on human pose estimation and face alignment.

2.1 Network quantization

Prior work [9] suggests that high precision parameters are not essential for obtaining top results for image classification. In light of this, [10], [11] propose 16- and 8-bit quantization, showing negligible performance drop on a few small datasets [12]. [13] proposes a technique which allocates different numbers of bits (1-2-6) for the network parameters, activations and gradients.

Binarization (i.e. the extreme case of quantization) was long considered to be impractical due to the destructive property of such a representation [10]. Recently [14] showed this not to be the case and that by quantizing to $\{-1, 1\}$ good results can be actually obtained. [15] introduces a new technique for training CNNs that uses binary weights for both forward and backward passes, however, the real parameters are still required during training. The work of [6] goes one step further and binarizes both parameters and activations. In this case multiplications can be replaced with elementary binary operations [6]. By estimating the binary weights with the help of a scaling factor, [5] is the first work to report good results on a large dataset (ImageNet). Notably, our method makes use of the recent findings from [5] and [6] using the same way of quantizing the weights and replacing multiplications with bit-wise *xor* operations.

Our method differs from all aforementioned works in two key respects: (a) instead of focusing on image classification, we are the first to study neural network binarization in the context of a fine-grained computer vision task namely landmark localization (human pose estimation and facial alignment) by predicting a dense output (heatmaps) in a fully convolutional manner, and (b) instead of enhancing the results by improving the quantization method, we follow a completely different path, by enhancing the performance via proposing a novel architectural design for a hierarchical, parallel and multi-scale residual block.

2.2 Block design

The proposed method uses a residual-based architecture and hence the starting point of our work is the *bottleneck* block described in [7], [16]. More recently, [17] explores the idea of increasing the cardinality of the residual block by splitting it into a series of c parallel (and much smaller so that the number of parameters remains roughly the same) sub-blocks with the same topology which behave as an ensemble. Beyond bottleneck layers, Szegedy et. al. [18] propose the inception block which introduces parallel paths with different receptive field sizes and various ways of lowering the number of parameters by factorizing convolutional layers with large filters into smaller ones. In a follow-up paper [19], the authors introduce a number of inception-residual architectures. The latter work is the most related one to the proposed method.

Our method is different from the aforementioned architectures in the following ways (see Fig. 1b): we create a hierarchical, parallel and multi-scale structure that (a) increases the receptive field size inside the block and (b) improves gradient flow, (c) is specifically designed to have (almost) the same number of parameters as the original bottleneck, (d) our block does not contain 1×1 convolutions, and (e) our block is derived from the perspective of improving the performance and efficiency of binary networks.

2.3 Network design

Our target was not to propose a new network architecture for landmark localization; hence we used the state-of-the-art *Hour-Glass* (HG) network of [2] which makes use of the bottleneck block of [16]. Because we are interested in efficiency, most of our experiments are conducted using a single network. Our baseline was the single binary HG obtained by directly quantizing it using [5]. As Table 1 shows, there is a significant performance gap between the binary and the real valued HGs. We bridge this gap by replacing the bottleneck block used in the original HG with the proposed block.

2.4 Human Pose Estimation

Traditionally, human pose estimation methods relied on tree structured graphical models [20], [21], [22], [23], [24], [25] to represent the spatial relationships between body parts and were usually built using hand crafted features. More

recently, methods based on CNNs have shown remarkable results outperforming traditional methods by large margin [1], [2], [3], [26], [27], [28], [29], [30]. Because learning a direct mapping from the image to the location of the body parts is a highly non-linear problem that is difficult to learn, most methods represent each landmark as a confidence map encoded as a 2D Gaussian centered at the landmark's location and adopt the fully convolutional framework of [31]. Furthermore, instead of making single-shot predictions, almost all methods follow a cascaded approach making a number of intermediate predictions, refined in a sequential manner [1], [2], [3]. Notably, to further reduce the number of parameters of the cascaded approaches the method introduced in [30] uses a recurrent neural network.

While achieving remarkable performance, all the aforementioned deep learning methods are computationally demanding, requiring at least one high-end GPU. In contrast, our network uses binary weights and activations and as such it is intended to run on systems with limited resources (e.g. embedded devices, smartphones).

2.5 Face Alignment

Current state-of-the-art for large pose 2D and 3D face alignment is also based on CNNs [4], [32], [33], [34], [35], [36]. However, despite their accuracy, these methods are computationally demanding. Our network produces state-of-the-art results for this task, yet it is designed to run on devices with limited computational resources.

3 Background

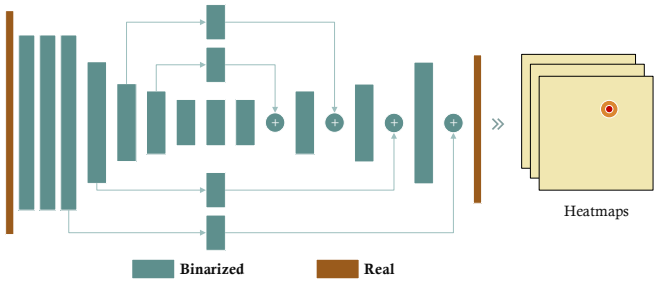


Fig. 2. The architecture of a single *Hour-Glass* (HG) network [2]. Following [5], the first and last layers (brown colour) are left real while all the remaining layers are binarized.

The ResNet consists of two types of blocks: *basic* and *bottleneck*. We are interested only in the latter one which was designed to reduce the number of parameters and keep the network memory footprint under control. We use the "pre-activation" version of [7], in which batch normalization [37] and the activation function precede the convolutional layer. Note that we used the version of bottleneck defined in [2] the middle layer of which has 128 channels (vs 64 used in [7]).

The residual block is the main building block of the Hourglass (HG) network, shown in Fig. 2, which is a state-of-the-art architecture for landmark localization that predicts a set of heatmaps (one for each landmark) in a fully convolutional fashion. The HG network is an extension of [31] allowing however for a more symmetric top-down and bottom-up processing. See also [2].

4 Method

Herein, we describe how we derive the proposed binary hierarchical, parallel and multi-scale block of Fig. 7e. In Section 4.6.1, by reducing the number of its parameters to match the ones of the original bottleneck, we further derive the block of Fig. 1b. This Section is organized as follows:

- We start by analyzing the performance of the binarized HG in Subsection 4.1 which provides the motivation as well as the baseline for our method.
- Then, we propose a series of architectural innovations in Subsections 4.2, 4.3, 4.4 and 4.5 (shown in Figs. 7b, 7c and 7d) each of which is evaluated and compared against the binarized residual block of Subsection 4.1.
- We continue, by combining ideas from these architectures, we propose the binary hierarchical, parallel and multi-scale block of Fig. 7e. Note that the proposed block is not a trivial combination of the aforementioned architectures but a completely new structure.
- Finally, we attempt to make a fair comparison between the performance of the proposed block against that of the original bottleneck module for both real and binary cases.

We note that all results for this Section were generated for the task of human pose estimation using the standard training-validation partition of MPII [1], [2].

4.1 Binarized HG

The binarization is accomplished using:

$$\mathrm{I} * \mathrm{W} \approx (sign(\mathrm{I}) \circledast sign(\mathrm{W})) * \alpha, \tag{1}$$

where I is the input tensor, W represents the layer weights, $\alpha \in \mathbb{R}^+$ is a scaling factor computed as the average of the absolute weight values and $\circledast$ denotes the binary convolution operation which can be efficiently implemented with XNOR.

We start from the original bottleneck blocks of the HG network and, following [5], we binarize them keeping only the first and last layers of the network real. See also Fig. 2. This is crucial, especially for the very last layer where higher precision is required for producing a dense output (heatmaps). Note that these layers account for less than 0.01% of the total number of parameters.

The performance of the original (real-valued) and the binarized HG networks can be seen in Fig. 3 and Table 1. We observe that binarization results in significant performance drop. As we may notice, for almost all parts, there is a large difference in performance which clearly indicates that the binary network has significant less representational power. Some failure cases are shown in Fig. 4 illustrating that the binary network was not able to learn some difficult poses. We address this with a better architecture as detailed in the next four Subsections.

4.2 On the Width of Residual Blocks

The original bottleneck block of Fig. 7a is composed of 3 convolutional layers with a filter size of 1×1, 3×3 and

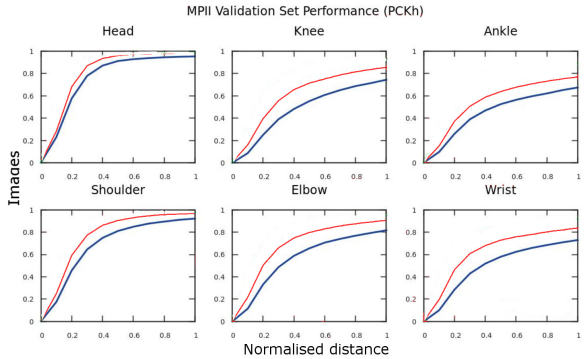


Fig. 3. Cumulative error curves on MPII validation set for real-valued (red) and binary (blue) bottleneck blocks within the HG network.

TABLE 1
PCKh error on MPII dataset for real-valued and binary bottleneck blocks within the HG network.

Crit.	Bottleneck (real)	Bottleneck (binary)
Head	94.9	90.5
Shld	85.8	79.6
Elbow	76.9	63.0
Wrist	71.3	57.2
Hip	78.1	71.1
Knee	70.1	58.2
Ankle	63.2	53.4
PCKh	76.5	67.2
# par.	3.5M	3.5M

Fig. 4. Examples of failure cases for the binarized HG (first row) and predictions of its real-valued counterpart (second row). The binary HG misses certain range of poses while having similar accuracy for the correct parts.

1×1, with the first layer having the role of limiting the width (i.e. the number of channels) of the second layer, thus greatly reducing the number of parameters inside the module. However, it is unclear whether the idea of having a bottleneck structure will be also successful for the binary case, too. Due to the limited representational power of the binary layers, greatly reducing the number of channels might reduce the amount of information that can be passed from one layer to another, leading to lower performance.

To investigate this, we modify the bottleneck block by increasing the number of channels in the *thin* 3×3 layer from 128 to 256. By doing so, we match the number of channels from the first and last layer, effectively removing the "bottleneck", and increasing the amount of information that can be passed from one block to another. The resulting **wider** block is shown in Fig. 7b. Here, "wider"[1] refers to the increased number of channels over the initial *thin* layer.

As Table 2 illustrates, while this improves performance against the baseline, it also raises the memory requirements. **Conclusion**: Widening the *thin* layer offers tangible performance improvement, however at a high computational cost.

4.3 On Multi-Scale Filtering

Small filters have been shown both effective and efficient [18], [39] with models being solely made up by a combination of convolutional layers with 3×3 and/or 1×1 filters [7], [16], [39]. For the case of real-valued networks, a large number of kernels can be learned. However, for the binary case, the number of possible unique convolutional kernels is limited to 2^k states only, where k is the size of the filter. Examples of such 3×3 learned filters are shown in Fig. 5.

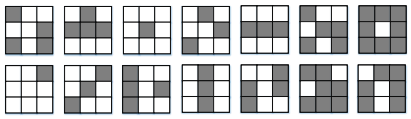


Fig. 5. Examples of learned 3×3 binary filters.

To address the limited representation power of 3×3 filters for the binary case, and similarly to [19], we largely depart from the block of Fig. 7b by proposing the multi-scale structure of Fig. 7c. Note that we implement our multi-scale approach using both larger filter sizes and max-pooling, which greatly increase the effective receptive field within the block. Also, because our goal is to analyze the impact of a multi-scale approach alone, we intentionally keep the number of parameters to a similar level to that of the original bottleneck block of Fig. 7a. To this end, we avoid a leap in the number of parameters, by (a) decomposing the 5×5 filters into two layers of 3×3 filters, and (b) by preserving the presence of *thin* layer(s) in the middle of the block.

Given the above, we split the input into two branches. The first (left) branch works at the same scale as the original bottleneck of Fig. 7a but has a 1×1 layer that projects the 256 channels into 64 (instead of 128) before going to the 3×3 one. The second (right) branch performs a multi-scale analysis by firstly passing the input through a max-pooling layer and then creating two branches, one using a 3×3 filter and a second one using a 5×5 decomposed into two 3×3. By concatenating the outputs of these two sub-branches, we obtain the remaining 64 channels (out of the 128 of the original bottleneck block). Finally, the two main branches are concatenated adding up to 128 channels, which are again back-projected to 256 with the help of a convolutional layer with 1×1 filters.

1. The term wider here strictly refers to a "moderate" increase in the number of channels in the *thin* layer (up to 256), effectively removing the "bottleneck". Except for the naming there is no other resemblance with [38] which performs a study of wide vs deep, using a different building block alongside a much higher number of channels (up to 2048) and without any form of quantization. A similar study falls outside the scope of our work.

The accuracy of the proposed structure can be found in Table 2. We can observe a healthy performance improvement at little additional cost and similar computational requirements to the original bottleneck of Fig. 7a.
Conclusion: When designing binarized networks, multi-scale filters should be preferred.

4.4 On 1×1 Convolutions

In the previously proposed block of Fig. 7c, we opted to avoid an increase in the number of parameters, by retaining the two convolutional layers with 1×1 filters. In this Subsection, by relaxing this restriction, we analyze the influence of 1×1 filters on the overall network performance.

In particular, we remove all convolutional layers with 1×1 filters from the multi-scale block of Fig. 7c, leading to the structure of Fig. 7d. Our motivation to remove 1×1 convolutions for the binary case is the following: because 1×1 filters are limited to two states only (either 1 or -1) they have a very limited learning power. Due to their nature, they behave as simple filters deciding when a certain value should be passed or not. In practice, this allows the input to pass through the layer with little modifications, sometimes actually blocking "good features" and hurting the overall performance by a noticeable amount. This is particularly problematic for the task of landmark localization, where a high level of detail is required for successful localization. Examples of this problem are shown in Fig. 6.

Results reported in Table 2 show that by removing 1×1 convolutions, performance over the baseline is increased by more than 8%. Even more interestingly, the newly introduced block outperforms the one of Subsection 4.2, while having less parameters, which shows that the presence of 1×1 filters limits the performance of binarized CNNs.
Conclusion: The use of 1×1 convolutional filters on binarized CNNs has a detrimental effect on performance and should be avoided.

Fig. 6. Examples of features before and after a 1×1 convolutional layer. Often the features are copied over with little modifications, usually consisting in the details' removal. The contrast was altered for better visualization.

4.5 On Hierarchical, Parallel & Multi-Scale

Binary networks are even more sensitive to the problem of fading gradients [5], [6], and for our network we found that the gradients are up to 10 times smaller than those corresponding to its real-valued counterpart. To alleviate this, we design a new module which has the form of a hierarchical, parallel multi-scale structure allowing, for each resolution, the gradients to have 2 different paths to follow, the shortest of them being always 1. The proposed block is depicted in Fig. 7e. Note that, in addition to better gradient flow, our design encompasses all the findings from the previous Subsections: (a) no convolutional layers with 1×1 filters should be used, (b) the block should preserve its width as much as possible (avoiding large drops in the number of channels), and (c) multi-scale filters should be used.

Contrary to the blocks described in Subsections 4.2 - 4.4, where the gradients may need to pass through two more layers before reaching the output of the block, in the newly proposed module, each convolutional layer has a direct path that links it to the output, so that at any given time and for all the layers within the module the shortest possible path is equal to 1. The presence of a hierarchical structure inside the module efficiently accommodates larger filters (up to 7×7), decomposed into convolutional layers with 3×3 filters. This allows for the information to be analysed at different scales because of the different filter sizes used (hence the term "multi-scale"). We opted not to use pooling because it results in loss of information. Furthermore, our design avoids the use of an element-wise summation layer as for example in [17], [19], further improving the gradient flow and keeping the complexity under control.

As we can see in Table 2, the proposed block matches and even outperforms the block proposed in Section 4.3 having far less parameters.

TABLE 2
PCKh-based comparison of different blocks on MPII validation set. # params refers to the number of parameters of the whole network.

Block type	# params	PCKh
Bottleneck (original) (Fig. 7a)	3.5M	67.2%
Wider (Fig. 7b)	11.3M	70.7%
Multi-Scale (MS) (Fig. 7c)	4.0M	69.3%
MS without 1x1 filters (Fig. 7d)	9.3M	75.5%
Bottleneck (wider) + no 1×1	5.8M	69.5%
Hierarchical, Parallel & MS (Ours, Final) (Fig. 1b)	**4.0M**	**72.7%**
Hierarchical, Parallel & MS (Ours, Final) (Fig. 7e)	**6.2M**	**76%**

Conclusion: Good gradient flow and hierarchical multi-scale filtering are crucial for high performance without excessive increase in the parameters of the binarized network.

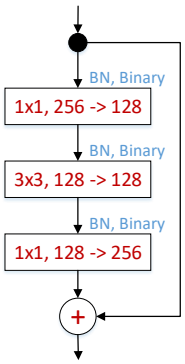
4.6 Proposed vs Bottleneck

In this Section, we attempt to make a fair comparison between the performance of the proposed block (**Ours, Final**, as in Fig. 7e) against that of the original bottleneck module (Fig. 7a) by taking two important factors into account:

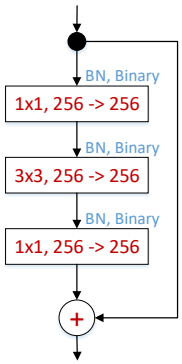
- Both blocks should have the same number of parameters.
- The two blocks should be compared for the case of binary but also real-valued networks.

With this in mind, in the following Sections, we show that:

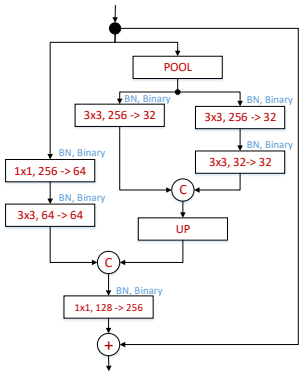
- The proposed block largely outperforms a bottleneck with the same number of parameters for the binary case.



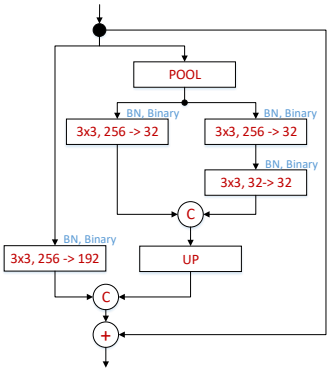
(a) The **Original Bottleneck** block with pre-activation, as defined in [7]. Its binarized version is described in Section 4.1.



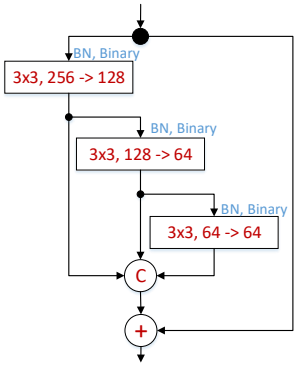
(b) The **Wider** version of (a) produced by increasing the number of filters in the second layer. See Subsection 4.2.



(c) Largely departing from (b), this block consists of **Multi-Scale (MS)** filters for analyzing the input at multiple scales. See Subsection 4.3.



(d) A variant of the MS block introduced in (c) after removing all convolutional layers with 1×1 filters (**MS Without** 1×1 **filters**). See Subsection 4.3.



(e) The proposed **Hierarchical, Parallel & MS** (denoted in the paper as (**Ours, final**) block incorporates all ideas from (b), (c) and (d) with an improved gradient flow. See Subsection 4.5

Fig. 7. Different types of blocks described and evaluated. Our best performing block is shown in figure (e). A layer is depicted as a rectangular block containing: its filter size, number of input channels and the number of output channels). "C" - denotes concatenation operation, "+" an element-wise sum and "UP" a bilinearly upsample layer.

- The proposed block also outperforms a bottleneck with the same number of parameters for the real case but in this case the performance difference is smaller.

We conclude that, for the real case, increasing the number of parameters (by increasing width) results in performance increase; however this is not the case for binary networks where a tailored design as the one proposed here is needed.

4.6.1 *Binary*

To match the number of parameters between the proposed and bottleneck block, we follow two paths. Firstly, we increase the number of parameters of the bottleneck: (a) a first way to do this is to make the block wider as described in Section 4.2. Note that in order to keep the number or input-output channels equal to 256, the resulting block of Fig. 7b has a far higher number of parameters than the proposed block. Despite this, the performance gain is only moderate (see Section 4.2 and Table 2). (b) Because we found that the 1×1 convolutional layers have detrimental effect to the performance of the Multi-Scale block of Fig. 7c, we opted to remove them from the bottleneck block, too. To this end, we modified the Wider module by (a) removing the 1×1 convolutions and (b) halving the number of parameters in order to match the number of parameters of the proposed block. The results in Table 2 clearly show that this modification is helpful but far from being close to the performance achieved by the proposed block.

Secondly, we decrease the number of parameters in the proposed block to match the number of parameters of the original bottleneck. This block is shown in Fig. 1b. To this end, we reduced the number of input-output channels of the proposed block from 256 to 192 so that the number of channels in the first layer are modified from $[256 \rightarrow 128, 3 \times 3]$ to $[192 \rightarrow 96, 3 \times 3]$, in the second layer from $[128 \rightarrow 64, 3 \times 3]$ to $[96 \rightarrow 48, 3 \times 3]$ and in the third layer from $[64 \rightarrow 64, 3 \times 3]$ to $[48 \rightarrow 48, 3 \times 3]$. Notice, that even in this case, the proposed binarized module outperforms the original bottleneck block by more than 5% (in absolute terms) while both have very similar number of parameters (see Table 2).

4.6.2 *Real*

While the proposed block was derived from a binary perspective, Table 3 shows that a significant performance gain is also observed for the case of real-valued networks. In order

to quantify this performance improvement and to allow for a fair comparison, we increase the number of channels inside the original bottleneck block so that both networks have the same depth and a similar number of parameters. For our binary block, in order to bring it back to the real valued domain, we simply replace the "sign" function with ReLU activations while keeping all the weights real. Even in this case, our block outperforms the original block although the gain is smaller than that observed for the binary case. We conclude that for real-valued networks performance increase can be more easily obtained by simply increasing the number of parameters, but for the binary case a better design is needed as proposed in this work.

TABLE 3
PCKh-based performance on MPII validation set for real-valued blocks: Our block is compared with a wider version of the original bottleneck so that both blocks have similar # parameters.

Layer type	# parameters	PCKh
Bottleneck (wider)	7.0M	83.1%
(Ours, Final)	**6.2M**	**85.5%**

5 Ablation studies

In this Section, we present a series of other architectural variations and their effect on the performance of our binary network. All reported results are obtained using the proposed block of Fig. 7e coined **Ours, Final**. We focus on the effect of augmentation and different losses which are novel experiments not reported in [5], and then comment on the effect of pooling, ReLUs and performance speed-up.

Is Augmentation required? Recent works have suggested that binarization is an extreme case of regularization [6], [15], [40]. In light of this, one might wonder whether data augmentation is still required. Table 4 shows that in order to accommodate the presence of new poses and/or scale variations, data augmentation is very helpful providing a large increase (4%) in performance. See Section 6.1 for more details on how augmentation was performed.

TABLE 4
The effect of using: augmentation, different losses (Sigmoid vs L2), different pooling methods and of adding a ReLU after the conv layer, when training our binary network in terms of PCKh-based performance on MPII validation set. We note that "(Ours, Final)" was trained using a Sigmoid Loss, Maxpooling and applying augmentation. The additional text after it denotes the change made.

Layer type	# parameters	PCKh
(Ours, Final) - No Aug.	6.2M	72.1%
(Ours, Final) - L2 loss	6.2M	73.8%
(Ours, Final) - AvgPool	6.2M	71.9%
(Ours, Final)	**6.2M**	**76%**
(Ours, Final) + ReLU	**6.2M**	**78.1%**

The effect of loss. We trained our binary network to predict a set of heatmaps, one for each landmark [27]. To this end, we experimented with two types of losses: the first one places a Gaussian around the correct location of each landmark and trains using a pixel-wise L2 loss [27]. However, the gradients generated by this loss are usually small even for the case of a real-valued network. Because binarized networks tend to amplify this problem, as an alternative, we also experimented with the Sigmoid cross-entropy pixel-wise loss typically used for detection tasks [41]. We found that the use of the Sigmoid cross-entropy pixel-wise loss increased the gradients by 10-15x (when compared to the L2 loss), offering a 2% improvement (see Table 4), after being trained for the same number of epochs.

Pooling type. In the context of binary networks, and because the output is restricted to 1 and -1, max-pooling might result in outputs full of 1s only. To limit this effect, we placed the activation function before the convolutional layers as proposed in [5], [7]. Additionally, we opted to replace max-pooling with average pooling. However, this leads to slightly worse results (see Table 4). In practice, we found that the use of blocks with pre-activation suffices and that the ratio of 1 and -1 is close to 50% even after max-pooling.

With or without ReLU. Because during the binarization process all ReLU layers are replaced with the Sign function, one might wonder if ReLUs are still useful for the binary case. Our findings are in line with the ones reported in [5]. By adding a ReLU activation after each convolutional layer, we observe a 2% performance improvement (see Table 4), which can be attributed to the added non-linearity, particularly useful for training very deep architectures.

Performance. In theory, by replacing all floating-point multiplications with bitwise XOR and making use of the SWAR (Single instruction, multiple data within a register) [5], [6], the number of operations can be reduced up to 32x when compared against the multiplication-based convolution. However, in our tests, we observed speedups of up to 3.5x, when compared against cuBLAS, for matrix multiplications, a result being in accordance with those reported in [6]. We note that we did not conduct experiments on CPUs. However, given the fact that we used the same method for binarization as in [5], similar improvements in terms of speed, of the order of 58x, are to be expected: as the real-valued network takes 0.67 seconds to do a forward pass on a i7-3820 using a single core, a speedup close to x58 will allow the system to run in real-time.

In terms of memory compression, by removing the biases, which have minimum impact (or no impact at all) on performance, and by grouping and storing every 32 weights in one variable, we can achieve a compression rate of 39x when compared against the single precision counterpart of Torch.

6 Comparison with state-of-the-art

In this Section, we compare our method against the current state-of-the-art for human pose estimation and 3D face alignment. Our final system comprises a single HG network but replaces the real-valued bottleneck block used in [2] with the proposed binary, parallel, multi-scale block trained with the improvements detailed in Section 5.

6.1 Training

All human pose estimation and 3D face alignment models were trained from scratch following the algorithm described

in [5] and using rmsprop [42]. The initialization was done as in [16]. For human pose estimation, we randomly augmented the data with rotation (between -40^o and 40^o degrees), flipping and scale jittering (between 0.7 and 1.3). We trained the network for 100 epochs, dropping the learning rate four times, from 2.5e-4 to 5e-5. A similar procedure was applied to the models for 3D face alignment, with the difference that the training was done for 55 epochs only. The input was normalized between 0 and 1 and all described networks were trained using the binary cross-entropy loss, defined as:

$$\mathbf{l} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{W}\sum_{j=1}^{H}\mathbf{p}_{ij}^{n}\log\widehat{\mathbf{p}}_{ij}^{n} + (1-\mathbf{p}_{ij}^{n})\log(1-\widehat{\mathbf{p}}_{ij}^{n}), \quad (2)$$

where $\mathbf{p}_{ij}^{n}$ denotes the ground truth confidence map of the $n-$th part at the output pixel location (i,j) and $\widehat{\mathbf{p}}_{ij}^{n}$ is the corresponding predicted output at the same location.

The models were implemented with Torch7 [43].

6.2 Human Pose Estimation.

As in all previous experiments, we used the standard training-validation partition of MPII [1], [2]. We report the performance of (a) the proposed binary block, (b) the proposed block when implemented and trained with real values, (c) the real-valued stacked HG network consisting of 8 stacked single real-valued HG networks trained with intermediate supervision (state-of-the-art on MPII [2]) and, finally, (d) the same real-valued network as in (c) where the bottleneck block is replaced by our proposed block.

The results are shown in Table 5. We observe that when a single HG network with the proposed block is trained with real weights, its performance reaches that of [2]. This result clearly illustrates the enhanced learning capacity of the proposed block. Moreover, there is still a gap between the binary and real-valued version of the proposed block indicating that margin for further improvement is possible. We also observe that a full-sized model (with 8 HG networks) based on the proposed block performs slightly better than the original network from [2], indicating that, for the real-valued case, the new block is more effective than the original one when a smaller computational budget is used.

TABLE 5
PCKh-based comparison on MPII validation set. For "Ours, bin." we report the results of its best variation, which includes the ReLU layer introduced in Section 5.

Crit.	[2]	Ours, bin.	Ours[1x], real	Ours[8x], real
Head	97.3	94.7	96.8	97.4
Shld	96.0	89.6	93.8	96.0
Elbow	90.2	78.8	86.4	90.7
Wrist	85.2	71.5	80.3	86.2
Hip	89.1	79.1	87.0	89.6
Knee	85.1	70.5	80.4	86.1
Ankle	82.0	64.0	75.7	83.2
PCKh	89.3	78.1	85.5	89.8
# par.	25M	6M	6M	25M

6.3 Face alignment.

We used three very challenging datasets for large pose face alignment, namely AFLW [44], AFLW-PIFA [45], and AFLW2000-3D [46]. The evaluation metric is the Normalized Mean Error (NME) [45].

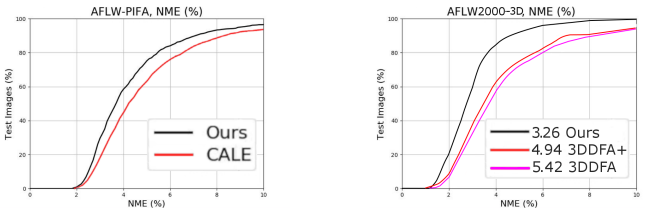
AFLW is a large-scale face alignment dataset consisting of 25,993 faces annotated with up to 21 landmarks. The images are captured in arbitrary conditions exhibiting a large variety of poses and expressions. As Table 6 shows, our binarized network outperforms the state-of-the-art methods of [47] and [34], both of which use large real-valued CNNs.

TABLE 6
NME-based (%) comparison on AFLW test set. The evaluation is done on the test set used in [34].

Method	[0,30]	[30,60]	[60,90]	mean
HyperFace [47]	3.93	4.14	4.71	4.26
AIO [34]	2.84	2.94	3.09	2.96
Ours	**2.77**	**2.86**	**2.90**	**2.85**

AFLW-PIFA [45] is a gray-scale subset of AFLW [44], consisting of 5,200 images (3,901 for training and 1,299 for testing) selected so that there is a balanced number of images for yaw angle in $[0°, 30°]$, $[30°, 60°]$ and $[60°, 90°]$. All images are annotated with 34 points from a 3D perspective. Fig. 8a and Tables 7 and 8 show our results on AFLW-PIFA. When evaluated on both visible and occluded points, our method improves upon the current best result of [33] (which uses real weights) by more than 10%.

AFLW2000-3D is a subset of AFLW re-annotated by [46] from a 3D perspective with 68 points. We used this dataset only for evaluation. The training was done using the first 40,000 images from 300W-LP [46]. As Fig. 8b shows, on AFLW2000-3D, the improvement over the state-of-the-art method of [46] (real-valued) is even larger. As further results in Fig. 9 show, while our method improves over the entire range of poses, the gain is noticeably higher for large poses ($[60° - 90°]$), where we outperform [46] by more than 40%.

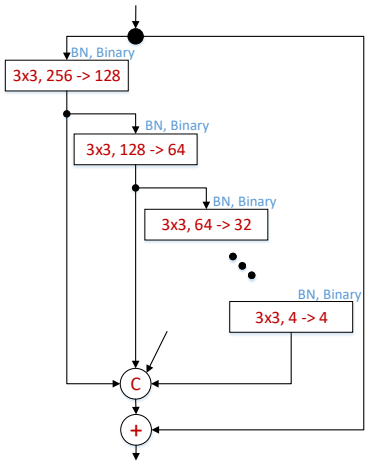


(a) (b)

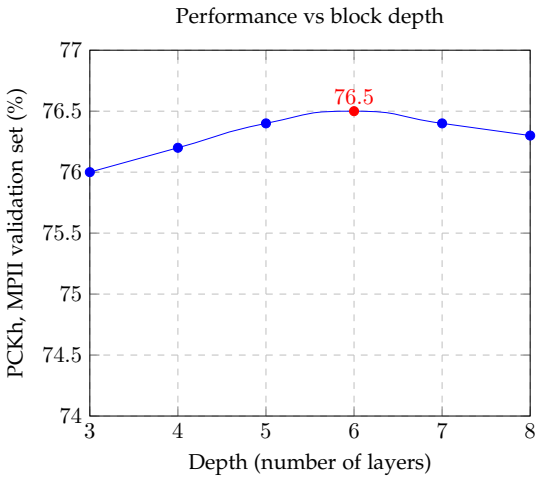
Fig. 8. Cumulative error curves (a) on AFLW-PIFA, evaluated on all 34 points (CALE is the method of [33]), (b) on AFLW2000-3D on all points computed on a random subset of 696 images equally represented in $[0°, 30°]$, $[30°, 60°]$, $[60°, 90°]$ (see also [46]).

7 Advanced Block Architectures

In this section, we explore the effectiveness of two architectural changes applied to our best performing block (**Ours, final**), namely varying its depth and its cardinality. Again, we used the standard training-validation partition of MPII.



(a) **(Ours, final)** binary block with varying depth. See also Subsection 7.1.



(b) Depth vs PCKh-based performance on the MPII validation set.

Fig. 9. The effect of varying the depth of the proposed binary block on performance.

TABLE 7
NME-based (%) comparison on AFLW-PIFA evaluated on visible landmarks only. The results for PIFA, RCPR and PAWF are taken from [32].

PIFA [45]	RCPR [48]	PAWF [32]	CALE [33]	Ours
8.04	6.26	4.72	**2.96**	3.02

TABLE 8
NME-based (%) based comparison on AFLW-PIFA evaluated on all 34 points, both visible and occluded.

CALE [33]	Ours
4.97	**4.47**

TABLE 9
NME-based (%) based comparison on AFLW2000-3D evaluated on all 68 points, both visible and occluded. The results for RCPR, ESR and SDM are taken from [46].

Method	[0,30]	[30,60]	[60,90]	Mean
RCPR(300W) [48]	4.16	9.88	22.58	12.21
RCPR(300W-LP) [48]	4.26	5.96	13.18	7.80
ESR(300W) [49]	4.38	10.47	20.31	11.72
ESR(300W-LP) [49]	4.60	6.70	12.67	7.99
SDM(300W) [50]	3.56	7.08	17.48	9.37
SDM(300W-LP) [50]	3.67	4.94	9.76	6.12
3DDFA [46]	3.78	4.54	7.93	5.42
3DDFA+SDM [46]	3.43	4.24	7.17	4.94
Ours	**2.47**	**3.01**	**4.31**	**3.26**

7.1 On the depth of the proposed block

To further explore the importance of the multi-scale component in the overall structure of the proposed block, we gradually increase its depth and as a result, the number of its layers, as shown in Fig. 9b. The advantage of doing this is twofold: (a) it increases the receptive field within the block, and (b) it analyses the input simultaneously at multiple scales. We ensure that by doing so the number of parameters remains (approximately) constant. To this end, we halve the number of channels of the last layer at each stage. In the most extreme case, the last layer will have a single channel. Because, the representational power of such a small layer is insignificant, in practice we stop at a minimum of 4, which corresponds to a depth equal to 8. The results, reported in Fig. 9b, show that the performance gradually improves up to 76.5% for a depth equal to 6, and then, further on, it saturates and eventually gradually degrades as the depth increases.

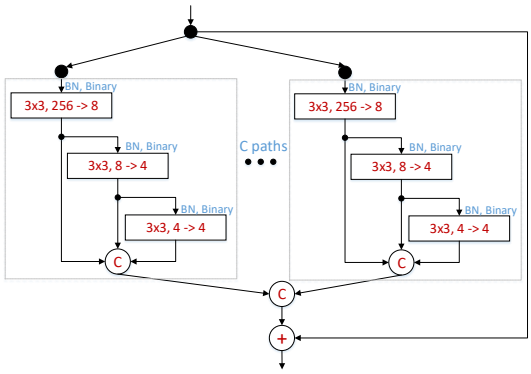
Conclusion: The depth of the multi-scale component is an important factor on the overall module performance. Increasing it, up to a certain point, is beneficial and can further improve the performance at no additional cost.

7.2 On the cardinality of the proposed block

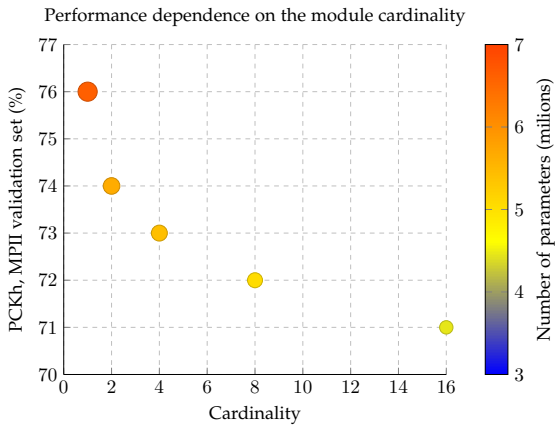
Inspired by the recent innovations of [17] for real-valued networks, in this section we explore the behavior of an increased cardinality (defined as in [17] as the size of the set of transformations) when applied to our binary hierarchical, parallel & multi-scale block.

Starting again from our block of Fig. 7e, we replicate its structure C times making the following adjustments in the process: (1) While the number of input channels of the first layer remains the same, the output and the input of the subsequent layers are reduced by a factor of C, and (2) the output of the replicated blocks is recombined via concatenation. The final module structure is depicted in Fig. 10b.

The full results with respect to the network size and the block cardinality (ranging from 1 to 16) are shown in Fig. 10b. Our findings are that increasing the block cardinality, while shown to provide good improvement on image classification using real-value networks, for the case of binary networks, given their significantly smaller size, depth and representational power, the same observation does not hold. In particular, when incorporated into the structure of our block with a similar number of parameters,



(a) ResNetXt-like extension of **(Ours, final)** binary block. C represents the cardinality of the block. See also Subsection 7.2.



(b) Cardinality vs PCKh-based performance on the MPII validation set. Notice how the efficiency (the ratio between the number of parameters and PCKh) decreases as we increase the block cardinality.

Fig. 10. The effect of varying the cardinality of the proposed binary block on performance.

the module under-performs by 1% compared to the original block (having a cardinality equal to one).
Conclusion: For the binary case, further increasing the block cardinality hurts performance.

8 Improved Network Architectures

In all previous sections, we investigated the performance of the various blocks by incorporating them into a single hourglass network, i.e. by keeping the network architecture fixed. In this section, we explore a series of architectural changes applied to the overall network structure. First, inspired by [51], we simplify the HG model, improving its performance without sacrificing accuracy for the binary case. Then, we study the effect of stacking multiple networks together and analyze their behavior.

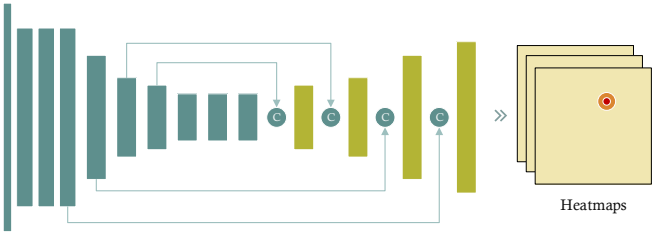


Fig. 11. Improved, U-Net inspired, HG architecture. The dark-green modules were left unchanged, while for the light-green ones we doubled the number of their input channels from 256 to 512.

8.1 Improved HG architecture

Motivated by the findings of Subsection 4.5 that shed light on the importance of the gradient flow and suggested that skip connections with shorter paths should be used where possible, we adopt a similar approach to the overall HG architecture.

In particular, to improve the overall gradient flow, we removed the residual blocks in the upsampling branches that are tasked with the "injection" of high resolution information into the later stages of the network. To adjust to that change, the number of input channels of the first layer from the modules that are immediately after the point where the branch is merged via *concatenation* is increased by two times (to accommodate to the increase in the number of channels). The resulting architecture, depicted in Fig. 11, is a modified U-net architecture [51] which was binarized in the same way as the HG model.

The results, reported in Table 10, show that by removing the residual blocks from the upsampling branches, the performance, over the baseline HG is increased by 0.5%, further solidifying the importance of the gradient flow in the performance of binary networks. Furthermore, due to the decrease in the number of layers and parameters, an up to 20% speedup is observed. The network is trained using the same procedure described previously, for 100 epochs.

TABLE 10
Comparison between HG and Improved HG on the MPII validation set. Both networks are built with our proposed binarized block.

Network architecture	# parameters	PCKh
HG (Fig. 2)	6.2M	76%
Improved HG (Fig. 11)	**5.8M**	**76.6%**

8.2 Stacked Binarized HG networks

Network stacking was recently shown to achieve state-of-the-art results on human pose estimation [1], [2], [3] when real-valued models are used. In this subsection, we explore whether the same holds for the binary case.

Following [2], we stack and interconnect the networks as follows: The first network takes as input the RGB image and outputs a set of N heatmaps. The next network in the stack takes as input the sum of: (1) the input to the previous network, (2) the projection of the previously predicted heatmaps, and (3) the output of the last but one block from

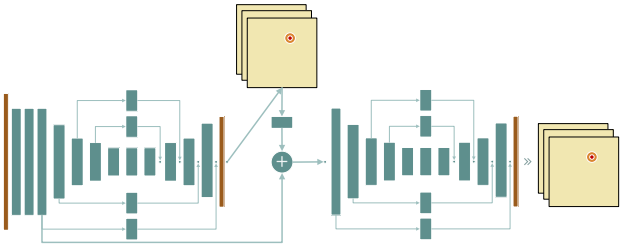


Fig. 12. A two-stack binarized HG. All blocks are binarized, except for the very first and last layers showed in red colour.

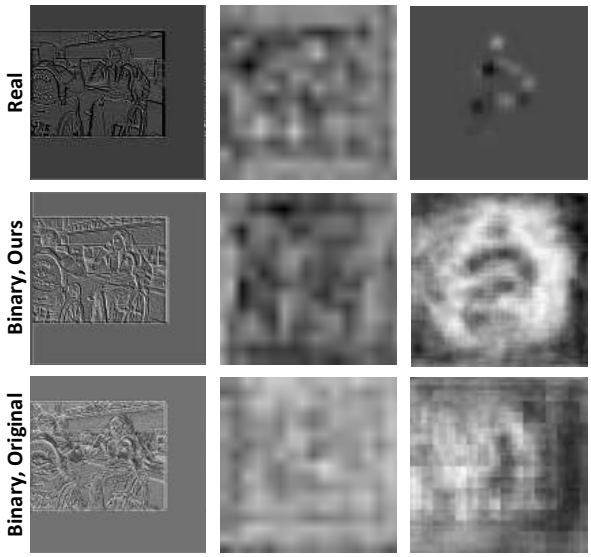


Fig. 13. Features extracted from the first layer (first column), middle layer (middle column) and right before the very last layer (right column) for real-valued and binary (ours and original) networks. As we move on to the last layers, activations become more noisy for the binary case, which we believe that it hurts the performance of the stacked networks.

the previous level. The resulting network for a stack of two is shown in Fig. 12.

As the results of Table 11 show, network stacking for the binary case behaves to some extent similarly to the real-valued case, however the gains from one stage to another are smaller, and performance seems to saturate faster. We believe that the main reason for this is that for the case of binary networks, activations are noisier especially for the last layers of the network. This is illustrated in Fig. 13 where we compare the feature maps obtained from a real and the two types of binary networks compared in this paper (original, based on bottleneck and proposed). Clearly the feature maps for the binary case are more noisy and blurry as we move on to the last layers of the network. As network stacking relies on features from the earlier networks of the cascade and as these are noisy, we conclude that this has a negative impact on the overall network's performance.

Training. To speedup the training process, we trained the stacked version in a sequential manner. First, we trained the first network until convergence, then we added the second one on top of it, freezing its weights and training the second one. The process is repeated until all networks are added. Finally, the entire stack is trained jointly for 50 epochs.

TABLE 11
Accuracy of stacked networks on MPII validation set. All networks are built with our proposed binarized block.

# stacks	# parameters	PCKh
1	6.2M	76%
2	11.0M	79.9%
3	17.8M	81.3%

9 Additional Experiments

In this section, we further show that the proposed block generalizes well producing consistent results across various datasets and tasks. To this end, we report results on the task of face parsing, also known as semantic facial part segmentation, which is the problem of assigning a categorical label to every pixel in a facial image. We constructed a dataset for facial part segmentation by joining together the 68 ground truth landmarks (originally provided for face alignment) to fully enclose each facial component. In total, we created seven classes: skin, lower lip, upper lip, inner mouth, eyes, nose and background. Fig. 14 shows an example of a ground truth mask. We trained the network on the 300W dataset (approximately 3,000 images) and tested it on the 300W competition test set, both Indoor&Outdoor subsets (600 images), using the same procedure described in Section 7.

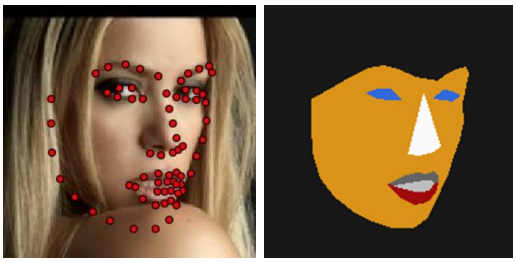


Fig. 14. Example of a ground truth mask (right) produced by joining the 68 ground truth landmarks (left). Each colour denotes one of the seven classes.

Architecture. We reused the same architecture for landmark localization, changing only the last layer in order to accommodate the different number of output channels (from 68 to 7). We report results for three different networks of interest: (a) a real-valued network using the original bottleneck block (called "Real, Bottleneck"), (b) a binary network using the original bottleneck block (called "Binary, Bottleneck"), and (c) a binary network using the proposed block (called "Binary, Ours"). To allow for a fair comparison, all networks have a similar number of parameters and depth. For training the networks, we used the Log-Softmax loss [31].

Results. Table 12 shows the obtained results. Similarly to our human pose estimation and face alignment experiments, we observe that the binarized network based on the proposed block significantly outperforms a similar-sized network constructed using the original bottleneck block, almost matching the performance of the real-valued network. Most of the performance improvement is due to the higher representation/learning capacity of our block, which

is particularly evident for difficult cases like unusual poses, occlusions or challenging lighting conditions. For visual comparison, see Fig. 16.

TABLE 12
Results on 300W (Indoor&Outdoor). The pixel acc., mean acc. and mean IU are computed as in [31].

Network type	pixel acc.	mean acc.	mean IU
Real, bottleneck	97.98%	77.23%	69.29%
Binary, bottleneck	97.41%	70.35%	62.49%
Binary, Ours	97.91%	76.02%	68.05%

10 Conclusion

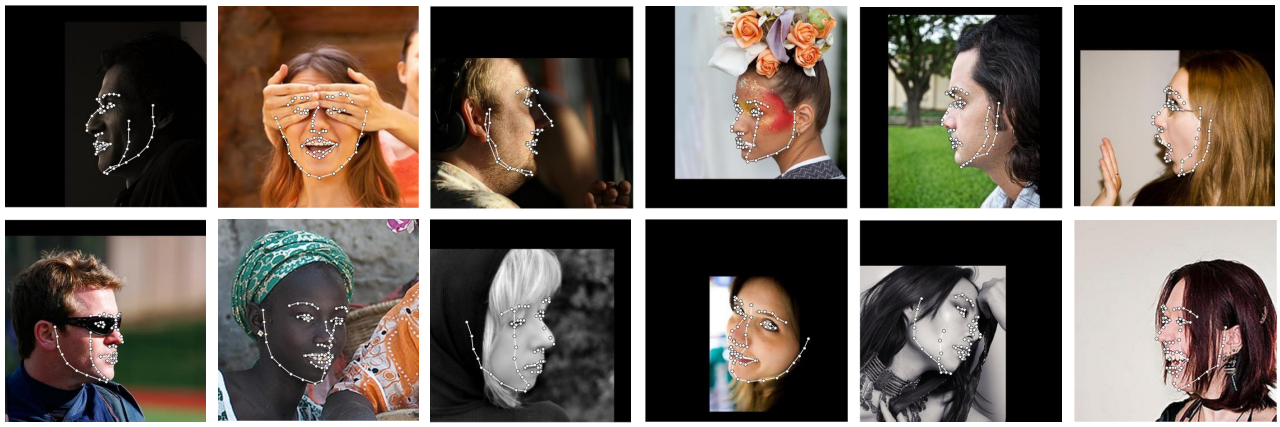
We proposed a novel block architecture, particularly tailored for binarized CNNs and localization visual tasks. During the process, we exhaustively evaluated various design choices, identified performance bottlenecks and proposed solutions. We showed that our hierarchical, parallel and multi-scale block enhances representational power, allowing for stronger relations to be learned without excessively increasing the number of network parameters. The proposed architecture is efficient and can run on limited resources. We verified the effectiveness of the proposed block on a wide range of fine-grained recognition tasks including human pose estimation, face alignment, and facial part segmentation.

Acknowledgments

Adrian Bulat was funded by a PhD scholarship from the University of Nottingham. Georgios Tzimiropoulos was supported in part by the EPSRC project EP/M02153X/1 Facial Deformable Models of Animals.

References

[1] A. Bulat and G. Tzimiropoulos, "Human pose estimation via convolutional part heatmap regression," in *ECCV*, 2016.
[2] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016.
[3] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.
[4] A. Bulat and G. Tzimiropoulos, "Two-stage convolutional part heatmap regression for the 1st 3d face alignment in the wild (3dfaw) challenge," in *ECCV*. Springer International Publishing, 2016, pp. 616–624.
[5] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, 2016.
[6] 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.
[7] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV*, 2016.
[8] A. Bulat and G. Tzimiropoulos, "Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources," in *ICCV*, 2017.
[9] J. L. Holi and J.-N. Hwang, "Finite precision error analysis of neural network hardware implementations," *IEEE Transactions on Computers*, vol. 42, no. 3, pp. 281–290, 1993.
[10] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," *arXiv*, 2014.
[11] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," *arXiv*, 2015.
[12] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.
[13] 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.
[14] D. Soudry, I. Hubara, and R. Meir, "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights," in *NIPS*, 2014.
[15] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *NIPS*, 2015.
[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[17] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *arXiv*, 2016.
[18] 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.
[19] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, 2017, pp. 4278–4284.
[20] M. Eichner, M. Marin-Jimenez, A. Zisserman, and V. Ferrari, "2d articulated human pose estimation and retrieval in (almost) unconstrained still images," *International journal of computer vision*, vol. 99, no. 2, pp. 190–214, 2012.
[21] P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman, "Upper body detection and tracking in extended signing sequences," *International journal of computer vision*, vol. 95, no. 2, pp. 180–197, 2011.
[22] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1385–1392.
[23] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele, "Strong appearance and expressive spatial models for human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 3487–3494.
[24] B. Sapp and B. Taskar, "Modec: Multimodal decomposable models for human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3674–3681.
[25] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic, "3d pictorial structures for multiple human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1669–1676.
[26] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *CVPR*, 2014.
[27] 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.
[28] T. Pfister, J. Charles, and A. Zisserman, "Flowing convnets for human pose estimation in videos," in *ICCV*, 2015.
[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*, 2016.
[30] V. Belagiannis and A. Zisserman, "Recurrent human pose estimation," in *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*. IEEE, 2017, pp. 468–475.
[31] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.
[32] A. Jourabloo and X. Liu, "Large-pose face alignment via cnn-based dense 3d model fitting," in *CVPR*, 2016.
[33] A. Bulat and G. Tzimiropoulos, "Convolutional aggregation of local evidence for large pose face alignment," in *BMVC*, 2016.
[34] R. Ranjan, S. Sankaranarayanan, C. D. Castillo, and R. Chellappa, "An all-in-one convolutional neural network for face analysis," in *IEEE Face & Gesture*, 2017.
[35] A. Bulat and G. Tzimiropoulos, "How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks)," in *ICCV*, 2017.
[36] Y. Wu, S. K. Shah, and I. A. Kakadiaris, "Godp: Globally optimized dual pathway deep network architecture for facial landmark localization in-the-wild," *Image and Vision Computing*, 2017.
[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv*, 2015.
[38] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv*, 2016.
[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv*, 2014.



(a) Fitting examples produced by our binary network on AFLW2000-3D dataset. Notice that our method copes well with extreme poses, facial expressions and lighting conditions.

(b) Examples of human poses obtained using our binary network. Observe that our method produces good results for a wide variety of poses and occlusions.

Fig. 15. Qualitative results produced by our method on (a) AFLW2000-3D and (b) MPII datasets.

[40] P. Merolla, R. Appuswamy, J. Arthur, S. K. Esser, and D. Modha, "Deep neural networks are robust to weight binarization and other non-linear distortions," *arXiv*, 2016.

[41] N. Zhang, E. Shelhamer, Y. Gao, and T. Darrell, "Fine-grained pose prediction, normalization, and recognition," *arXiv preprint arXiv:1511.07063*, 2015.

[42] 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*, vol. 4, no. 2, 2012.

[43] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *NIPS-W*, no. EPFL-CONF-192376, 2011.

[44] M. Köstinger, P. Wohlhart, P. M. Roth, and H. Bischof, "Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization," in *ICCV-W*, 2011.

[45] A. Jourabloo and X. Liu, "Pose-invariant 3d face alignment," in *ICCV*, 2015.

[46] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, "Face alignment across large poses: A 3d solution," in *CVPR*, 2016.

[47] R. Ranjan, V. M. Patel, and R. Chellappa, "Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," *arXiv preprint arXiv:1603.01249*, 2016.

[48] X. P. Burgos-Artizzu, P. Perona, and P. Dollár, "Robust face landmark estimation under occlusion," in *ICCV*, 2013.

[49] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 177–190, 2014.

[50] X. Xiong and F. De la Torre, "Supervised descent method and its applications to face alignment," in *CVPR*, 2013.

[51] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

Adrian Bulat is currently a PhD student with the Computer Vision Laboratory at the University of Nottingham, under the supervision of Dr. Georgios Tzimiropoulos. He received his B.Eng. in Computer Engineering (2015) from the Technical University "Gheorghe Asachi" (Romania).

Fig. 16. Qualitative results on 300W (Indoor&Outdoor). Observe that the proposed binarized network significantly outperforms the original binary one, almost matching the performance of the real-valued network.

Georgios (Yorgos) Tzimiropoulos received the M.Sc. and Ph.D. degrees in Signal Processing and Computer Vision from Imperial College London, U.K. He is Assistant Professor with the School of Computer Science at the University of Nottingham, U.K. Prior to this, he was a Senior Researcher in the iBUG group, Department of Computing, Imperial College London. He is currently Associate Editor of the Image and Vision Computing Journal. He has worked on the problems of object detection and tracking, alignment and pose estimation, and recognition with humans and faces being the focal point of his research. For his work, he has used a variety of tools from Mathematical Optimization and Machine Learning. His current focus is on Deep Learning.