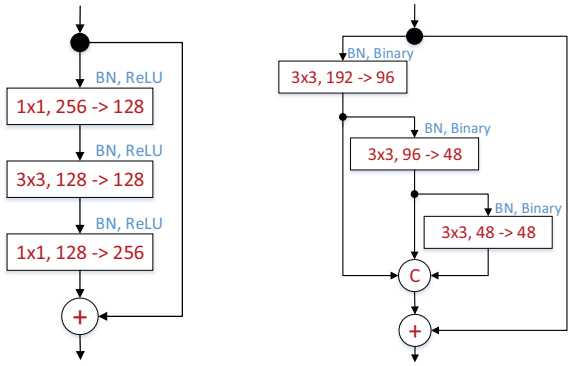
Binarized Convolutional Landmark Localizers for Human Pose Estimation and Face Alignment with Limited Resources

Adrian Bulat and Georgios Tzimiropoulos
Computer Vision Laboratory, The University of Nottingham
Nottingham, United Kingdom
{adrian.bulat, yorgos.tzimiropoulos}@nottingham.ac.uk

Abstract

Our goal is to design architectures that retain the groundbreaking performance of 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. Code can be downloaded from `https://www.adrianbulat.com/binary-cnn-landmarks`



(a) original (b) proposed

Figure 1: (a) The original bottleneck layer of [9]. (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.

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 [2, 20, 32] and face alignment [3]. 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 very recent results of binarized CNN architectures on image classification [24, 7]. 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 [24, 7], 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 completely novel hierar-

2380-7504/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCV.2017.400

chical, parallel and multi-scale residual block, as opposed to investigating ways to improve the binarization process as proposed in [24, 7]. In summary, **our contributions** are:

1. 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.
2. 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 residual block of [9] (about 6% in absolute terms when the same number of parameters are used (see Subsection 5.1, Tables 3 and 4)).
3. 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 5.2).
4. We perform a large number of ablation studies that shed light on the properties and the performance of the proposed block (see Sections 5 and 7).
5. 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 7).

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.

Network quantization. Prior work [10] suggests that high precision parameters are not essential for obtaining top results for image classification. In light of this, [5, 17] propose 16- and 8-bit quantization, showing negligible performance drop on a few small datasets [16]. [36] 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 [5]. Recently [26] showed this not to be the case and that by quantizing to $\{-1, 1\}$ good results can be actually obtained. [6] 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 [7] goes one step further and binarizes both parameters and activations. In this case multiplications can be replaced with elementary binary operations [7]. By estimating the binary weights with the help of a scaling factor, [24] is the first work to report good results on a large dataset (ImageNet). Notably, our method makes use of the recent findings from [24] and [7] 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.

Block design. The proposed method uses a residual-based architecture and hence the starting point of our work is the *bottleneck* block described in [8, 9]. More recently, [33] 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*. [28] 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 [27], 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.

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 [20] which makes use of the bottleneck block of [8]. Because we are interested in efficiency, all of our experiments are conducted using a single network i.e. we do not use stacking as in [20]. Our baseline was the binary HG obtained by directly quantizing it using [24]. 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.

Human Pose Estimation. Recent work using CNNs has

shown remarkable results [31, 30, 21, 11, 2, 20, 32], yet all these methods are computationally demanding, requiring at least one high-end GPU. In contrast, our network uses binary weights and activations and as such is intended to run on systems with limited resources (e.g. smartphones).

Face alignment. Current state-of-the-art for large pose 2D and 3D face alignment is also based on CNNs [14, 1, 3]; however, 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 resources.

3. Background

The ResNet consists of two type 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 [9], in which batch normalization [12] and the activation function precede the convolutional layer. This block is shown in Fig. 1a. Note that we used the version of bottleneck defined in [20] the middle layer of which has 128 channels (vs 64 used in [9]).

The residual block is the main building block of the HG 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 [18] allowing however for a more symmetric top-down and bottom-up processing. See also [20].

4. Method

Herein, we describe how we derive the proposed binary hierarchical, parallel and multi-scale block of Fig. 4e. In Section 5.1, 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. 4b, 4c and 4d) each of which is evaluated and compared against the binarized residual block of Subsection 4.1.
- Finally, by combining ideas from these architectures, we propose the binary hierarchical, parallel and multi-scale block of Fig. 4e. Note that the proposed block is not a trivial combination of the aforementioned architectures but a completely new structure.

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 [2, 20].

4.1. Binarized HG

We start from the original bottleneck blocks of the HG network and, following [24], we binarize them keeping only the first and last layers of the network real. 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 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. 2 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.

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

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

Figure 2: 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.

4.2. On the Width of Residual Blocks

The original bottleneck block of Fig. 4a is composed of 3 convolutional layers with a filter size of 1×1, 3×3 and 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. 4b. 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 [25, 28] with models being solely made up by a combination of convolutional layers with 3×3 and/or 1×1 filters [8, 9, 25]. 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.

To address the limited representation power of 3×3 filters for the binary case, and similarly to [27], we largely depart from the block of Fig. 4b by proposing the multi-scale structure of Fig. 4c. 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. 4a. 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. 4a 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.

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. 4a.
Conclusion: When designing binarized networks, multi-scale filters should be preferred.

4.4. On 1×1 Convolutions

In the previously proposed block of Fig. 4c, 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. 4c, leading to the structure of Fig. 4d. 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. 3.

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.

4.5. On Hierarchical, Parallel & Multi-Scale

Binary networks are even more sensitive to the problem of fading gradients [7, 24], 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 resolu-

[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 [34] 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.

Figure 3: Examples of features before and after an 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.

tion, the gradients to have 2 different paths to follow, the shortest of them being always 1. The proposed block is depicted in Fig. 4e. 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. Furthermore, our design avoids the usage of an element-wise summation layer as for example in [33, 27], 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.

Block type	# params	PCKh
Bottleneck (original) (Fig. 4a)	3.5M	67.2%
Wider (Fig. 4b)	11.3M	70.7%
Multi-Scale (MS) (Fig. 4c)	4.0M	69.3%
MS without 1x1 filters (Fig. 4d)	9.3M	75.5%
Hierarchical, Parallel & MS (Ours, Final) (Fig. 4e)	**6.2M**	**76%**

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

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

5. 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. 4e) against that of the original bottleneck module (Fig. 4a) 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.
- 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.

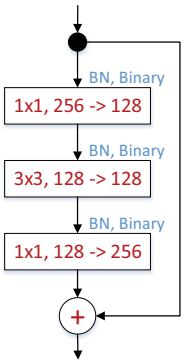
We conclude that, for the real case, increasing the number of parameters (by increasing width) results in performance increase; however this is not the case for binary networks where a tailored design as the one proposed here is needed.

Layer type	# parameters	PCKh
Bottleneck (Original) (Fig. 4a)	3.5M	67.2%
Wider (Fig. 4b)	11.3M	70.7%
Bottleneck (wider) + no 1×1	5.8M	69.5%
(Ours, Final) (Fig. 4e)	**6.2M**	**76%**

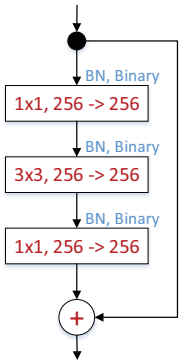
Table 3: PCKh-based performance on MPII validation set for binary blocks: the # parameters of the original bottleneck are increased to match the # parameters of the proposed block. This firstly gives rise to the Wider block and its variant without the 1×1 Convolutions.

5.1. Binary

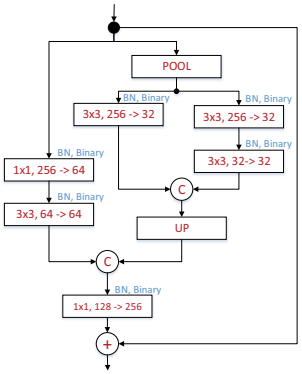
To match the number of parameters between the proposed and bottleneck block, we follow two paths. Firstly, we increase the number of parameters of the bottleneck: (a) a first way to do this is to make the block wider as described in Section 4.2. Note that in order to keep the number or input-output channels equal to 256, the resulting block of Fig. 4b 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 3). (b) Because we found that the 1×1 convolutional layers have detrimental effect to the performance of the Multi-Scale block of Fig. 4c, 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 3 clearly show that this modification is helpful but far from being close to the performance achieved by the proposed block.



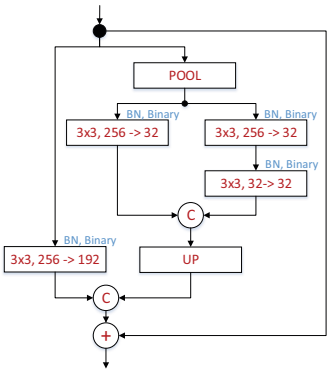
(a) The **Original Bottleneck** block with pre-activation, as defined in [9]. Its binarized version is described in Section 4.1.



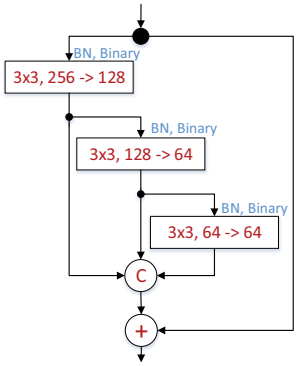
(b) The **Wider** version of (a) produced by increasing the number of filters in the second layer. See Subsection 4.2.



(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

Figure 4: 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 and "+" an element-wise sum.

Layer type	# parameters	PCKh
Bottleneck (original)	3.5M	67.2%
(Ours, Final) (Fig. 1b)	**4.0M**	**72.7%**

Table 4: PCKh-based performance on MPII validation set for binary blocks: the # parameters of the proposed block are decreased to match the # parameters of the bottleneck.

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 $[256 \rightarrow 128, 3\times3]$ to $[192\rightarrow96, 3 \times 3]$, in the second layer from $[128\rightarrow64, 3 \times 3]$ to $[96\rightarrow48, 3\times3]$ and in the third layer from $[64\rightarrow64, 3\times3]$ to $[48\rightarrow48, 3 \times 3]$. 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 4).

5.2. Real

While the proposed block was derived from a binary perspective, Table 5 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. 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.

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

Table 5: 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.

6. 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. 4e coined **Ours, Final**. We focus on the effect of augmentation and different losses which are novel experiments not reported in [24], and then comment on the effect of pooling, ReLUs and performance speed-up providing more details in the supplementary material.

Is Augmentation required? Recent works have suggested that binarization is an extreme case of regularization [6, 7, 19]. In light of this, one might wonder whether data augmentation is still required. Table 6 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.

Layer type	# parameters	PCKh
(Ours, Final) (No Aug.)	6.2M	72.1%
(Ours, Final) + Aug.	**6.M**	**76%**

Table 6: The effect of using augmentation when training our binary network in terms of PCKh-based performance on MPII validation set.

The effect of loss. We trained our binary network to predict a set of heatmaps, one for each landmark [30]. 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 [30]. 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 [35]. 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 7), after being trained for the same number of epochs.

Layer type	# parameters	PCKh
(Ours, Final) + L2	6.2M	73.8%
(Ours, Final) + Sigmoid	**6.2M**	**76%**

Table 7: The effect of using different losses (Sigmoid vs L2) when training our binary network in terms of PCKh-based performance on MPII validation set.

Pooling type. In line with [24], we found that max-pooling outperforms average pooling, resulting in 4% performance increase. See also supplementary material.

ReLUs. In line with [24], we found that by adding a ReLU activation after each convolutional layer performance is increased, observing a 2% performance improvement. See also supplementary material.

Performance. In line with [24], we observed speedups of up to 3.5x when compared against cuBLAS. We did not conduct experiments on CPUs. However, since we used the same method for binarization as in [24], speed improvements of the order of 58x, are to be expected allowing the system to run in real-time on a CPU using a single core. In terms of memory compression, we can achieve a compression rate of 39x when compared against its single precision counterpart from Torch. See also supplementary material.

7. 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 [20] with the proposed binary, parallel, multi-scale block trained with the improvements detailed in Section 6. Moreover, to show that the proposed block generalizes well producing consistent results across various datasets and tasks, our supplementary material provides the results of a facial part segmentation experiment.

Human Pose Estimation. As in all previous experiments, we used the standard training-validation partition of MPII [2, 20]. In Table 8, 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 [20]) 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 8. We observe that when a single HG network with the proposed block is trained with real weights, its performance reaches that of [20]. 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 indi-

cating 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 [20], indicating that, for the real-valued case, the new block is more effective than the original one when a smaller computational budget is used.

Crit.	[20]	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

Table 8: PCKh-based comparison on MPII validation set.

Face alignment. We used three very challenging datasets for large pose face alignment, namely AFLW [15], AFLW-PIFA [13], and AFLW2000-3D [37]. The evaluation metric is the Normalized Mean Error (NME) [13].

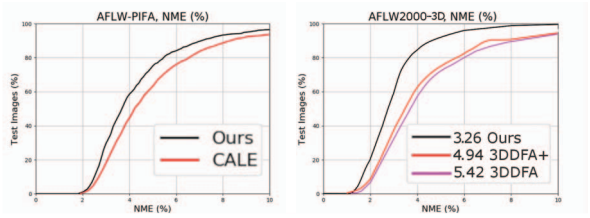
AFLW is a large-scale face alignment dataset consisting of 25993 faces annotated with up to 21 landmarks. The images are captured in arbitrary conditions exhibiting a large variety of poses and expressions. As Table 9 shows, our binarized network outperforms the current state-of-the-art methods, all of which use large real-valued CNNs.

Method	[0,30]	[30,60]	[60,90]	mean
HyperFace [22]	3.93	4.14	4.71	4.26
AIO [23]	2.84	2.94	3.09	2.96
Ours	**2.77**	**2.86**	**2.90**	**2.85**

Table 9: NME-based (%) comparison on AFLW test set. The evaluation is done on the test set used in [23].

AFLW-PIFA [13] is a grey-scale subset of AFLW [15], consisting of 5200 images (3901 for training and 1299 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. 5a shows our results on AFLW-PIFA. When evaluated on both visible and occluded points, our method improves upon the current best result of [1] (which uses real weights) by more than 10%. For additional numerical results on AFLW-PIFA, see also our supplementary material.

AFLW2000-3D is a subset of AFLW re-annotated by [37] from a 3D perspective with 68 points. We used this dataset only for evaluation. The training was done using the first 40000 images from 300W-LP [37]. As Fig. 5b shows, on AFLW2000-3D, the improvement over the state-of-the-art method of [37] (real-valued) is even larger. As further results in our supplementary material show, while our method improves over the entire range of poses, the gain is noticeably higher for large poses ($[60° - 90°]$) where we outperform [37] by more than 40%.



(a) (b)

Figure 5: Cumulative error curves (a) on AFLW-PIFA, evaluated on all 34 points (CALE is the method of [1]), (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 [37]).

Training. All models were trained from scratch following the algorithm described in [24] and using rmsprop [29]. The initialization was done as in [8]. 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. The models were implemented with Torch7 [4].

8. Conclusion

We proposed a novel block architecture, particularly tailored for binarized CNNs for the tasks of human pose estimation and face alignment. 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.

9. Acknowledgment

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. Convolutional aggregation of local evidence for large pose face alignment. In *BMVC*, 2016.

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

[3] A. Bulat and G. Tzimiropoulos. Two-stage convolutional part heatmap regression for the 1st 3d face alignment in the wild (3dfaw) challenge. In *ECCV*, pages 616–624. Springer International Publishing, 2016.

[4] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *NIPS-W*, number EPFL-CONF-192376, 2011.

[5] M. Courbariaux, Y. Bengio, and J.-P. David. Training deep neural networks with low precision multiplications. *arXiv*, 2014.

[6] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, 2015.

[7] 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.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[10] J. L. Holi and J.-N. Hwang. Finite precision error analysis of neural network hardware implementations. *IEEE Transactions on Computers*, 42(3):281–290, 1993.

[11] 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.

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015.

[13] A. Jourabloo and X. Liu. Pose-invariant 3d face alignment. In *ICCV*, 2015.

[14] A. Jourabloo and X. Liu. Large-pose face alignment via cnn-based dense 3d model fitting. In *CVPR*, 2016.

[15] 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.

[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[17] D. D. Lin, S. S. Talathi, and V. S. Annapureddy. Fixed point quantization of deep convolutional networks. *arXiv*, 2015.

[18] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[19] 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.

[20] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[21] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *ICCV*, 2015.

[22] 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.

[23] 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.

[24] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.

[26] D. Soudry, I. Hubara, and R. Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *NIPS*, 2014.

[27] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[28] 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.

[29] 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*, 4(2), 2012.

[30] 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.

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

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

[33] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *arXiv*, 2016.

[34] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv*, 2016.

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

[36] 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.

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