symmetry

MDPI

Article

Lightweight Architecture for Real-Time Hand Pose Estimation with Deep Supervision

Yufei Wu [1], Xiaofei Ruan [2], Yu Zhang [2], Huang Zhou [2], Shengyu Du [3] and Gang Wu [1,*]

1 School of Software, Shanghai Jiao Tong University, Shanghai 200240, China; wuyufei@sjtu.edu.cn
2 Internet of Things Group, Intel Corporation, Shanghai 200131, China; xiaofei.ruan@intel.com (X.R.); yu.d.zhang@intel.com (Y.Z.); jerry.zhou@intel.com (H.Z.)
3 School of Software, Tongji University, Shanghai 200072, China; shengyudu@gmail.com
* Correspondence: dr.wugang@sjtu.edu.cn

Received: 7 March 2019; Accepted: 18 April 2019; Published: 23 April 2019

Abstract: The high demand for computational resources severely hinders the deployment of deep learning applications in resource-limited devices. In this work, we investigate the under-studied but practically important network efficiency problem and present a new, lightweight architecture for hand pose estimation. Our architecture is essentially a deeply-supervised pruned network in which less important layers and branches are removed to achieve a higher real-time inference target on resource-constrained devices without much accuracy compromise. We further make deployment optimization to facilitate the parallel execution capability of central processing units (CPUs). We conduct experiments on NYU and ICVL datasets and develop a demo[1] using the RealSense camera. Experimental results show our lightweight network achieves an average running time of 32 ms (31.3 FPS, the original is 22.7 FPS) before deployment optimization. Meanwhile, the model is only about half parameters size of the original one with 11.9 mm mean joint error. After the further optimization with OpenVINO, the optimized model can run at 56 FPS on CPUs in contrast to 44 FPS running on a graphics processing unit (GPU) (Tensorflow) and it can achieve the real-time goal.

Keywords: hourglass; 3D regression; deep supervision; prune; deployment optimization; convolutional neural network

1. Introduction

Hand pose estimation has made rapid progress in recent years. The increased accuracy can be attributed to two aspects: depth camera and convolution neural networks (CNN). Firstly, commodity depth sensors, such as the Microsoft Kinect and the Intel RealSense, have an attractive price and stable performance. Moreover, depth cameras have been embedded in mobile devices. For example, Apple Inc. has adopted TrueDepthCameraSystem for face identification (FaceID) and extension of the selfie. With the help of depth information, algorithms can resolve many of the ambiguities compared to RGB images. Secondly, deep learning has significantly changed the way of solving the vision problem. Many backbones and networks play essential roles in different visual fields.

Deep neural networks (DNNs) have been successfully applied to 3D hand pose estimation [1–4]. 2.5D depth data can be treated both as 2D images and volumetric representations. Wan [5] designed careful parameterization to combine holistic 3D regression with 3D detection and outperformed all previous state-of-arts on three public datasets. However, they used the hourglass network [6] as the backbone and two repeated stacks to improve accuracy. With this structure, we found that the inference

1 Demo video: https://youtu.be/yFd6WrxmK3s

time still cannot meet the real-time performance requirement even in the case of one NVIDIA TITAN X and the requirement for expensive graphics processing units (GPUs) is a barrier to deployment on mobile devices. It is probably because previous work mainly focused on model accuracy alone while overlooking the importance of model efficiency and the deployed platform.

Nevertheless, deep learning should embrace more resources-limited devices for a broader application scenario. Deep learning networks with large depth and width are resource-intensive in the training and inference process. The industrial world has plenty of attempts for model speedup, such as binary neural network [7], knowledge distillation [8] and network compression [9]. With these network compressions and speed up technology, a small sacrifice of accuracy can result in significant time-saving. For hand pose estimation, as an interactive task, real-time performance is crucial as long as the error is within the acceptable range.

In this study, we carefully make two-level network pruning to improve hand pose estimation efficiency without accuracy degradation. First, we observe that the following stacked modules (such as [5,6]) contribute much less to the overall accuracy than the previous modules. It allows us to keep a high accuracy when pruning the latter blocks (also called: depth pruning). Secondly, we treat the hourglass block as a variant of U-Net with nested and dense skip connections. Therefore, we can further prune the hourglass block at inference time if it is trained using deep supervision modules. In the experiments, we find deep supervision significantly helps to overcome the underfit problem of a smaller network compared to knowledge distillation. Moreover, deep supervision can mitigate the overfitting of model and outperform its original model. The proposed pruned network enables much faster inference time with remarkably smaller (49%) model size while only 1.67mm mean joint error increase.

Our contributions can be summarized as follows:

- We investigate the less-studied hourglass network efficiency problem. Different from the most previous researches, we also focus on the model's inference cost while keeping the accuracy performance.
- We treat the hourglass network as a variant of U-Net and propose a new training strategy using deep supervision. It has a better generalization performance and allows us for finer-grained pruning.
- We implement the two-level pruning strategy in the public NYU and ICVL datasets and achieve satisfactory accuracy rates compared to the previous state-of-the-art approaches. We also extensively examine the redundancy of the hand pose estimation design and find the significant benefits brought from deep supervision. We apply our model using OpenVINO toolkit to accelerate the inference process on the CPUs and it runs faster in contrast to GPUs.

2. Related Work

2.1. Network Pruning

Determining the proper size of a neural network is recognized as crucial and has a long history. Early work can be traced back to 1997 [10] and it proposed the method of iteratively eliminating units and adjusting the remaining weight. After 2006, with the popularity of CNNs [11], researchers [12,13] reduced parameters of AlexNet [14] and VGG-16 [15] using connection pruning. However, most of the reduction is achieved on fully-connected layers and it has no apparent speedup of convolutional layers. Many new designs use fewer FCNs (Fully Convolutional Networks), for example, only 3.99% of the parameters of ResNet-152 [16] are from FCNs. Then, to reduce the cost of the convolution layers, several works [17,18] have studied removing the redundant filters from a well-trained network. Polyak et al. [18] removed the less frequently activated feature maps using input samples. Li [19] removed the whole filters and their connections based on L1-norm and this method did not depend on excessive hyperparameters and resulted in sparse connectivity patterns. In our method, we also focus on convolutional layers pruning. We make coarser-grained layers pruning based on the deep characteristics of the model. Our study is similar to that described in Reference [20] but we choose to prune instead of knowledge distillation [8]. Therefore, our method needs no additional training overhead compared to knowledge distillation and can achieve a satisfactory result.

2.2. Hand Pose Estimation

There is huge progress in hand pose estimation based on the deep learning method. The current best-performing methods [21–23] are all single stage (holistic pose regression), probably because they take full advantage of joint correlations. Wan's work [5] well exploits the 2D and 3D properties of depth images and achieves evident performance increases. However, these prior works focus only on accuracy by using stacked structures and expensive models while mostly ignoring the inference cost. Therefore, it will restrict their scalability in real-time applications. In our paper, we present a fast hand pose estimation model based on dense regression and the proposed model is more lightweight and usable in the real world.

2.3. Hourglass and U-Net

U-Net [24] was introduced by Ronneberger et al. and achieves good performance with biomedical image segmentation. U-Net concatenates the up-sampled features with features skipped from the encoder and adds convolutions layers between each up-sampling step. Inspired by DenseNet [25], much research has attempted to modify the skip connections: Li proposed H-denseunet [26] for liver segmentation and Drozdzal [27] introduced short skip connections within the encoder. Similar to UNet++ [28], an hourglass network has the encoder and decoder sub-networks connected through nested, dense and skip pathways. Opposite to fusing semantically dissimilar feature maps from different sub-networks, we exploit the semantic similarity to build an easier-learning task. Explicitly, we add supervision modules to every decoder and average their losses.

2.4. Deep Supervision

To train a deeper network more efficiently, many studies adopt the idea of adding auxiliary classifiers after some of the intermediate convolutional layers [29–32]. Deep supervision helps to relieve gradient disappearance and explosion and increase the transparency of hidden layers. Reference [31] raised the idea that a discriminative classifier trained using hidden layer feature maps can serve as a proxy for the quality of those hidden layers and further the upper layers. Inspired by these guidelines, we use deep supervision in an hourglass network and enabling it as (1) an accurate model since the overall loss is averaged from all branches and (2) a fast model since we can select the last branch and speed up the inference process.

3. Method

We leverage the redundancy of the hourglass network (hourglass is widely used, such as References [33–36]) and make a two-level pruning to the dense 3D regression for hand pose estimation. With empirical examination, we revealed that a half number of stages suffices to keep over 96% accuracy on the NYU dataset. Later, the resulting network, with deep supervision, can be further pruned to the L_3 network and suffice to achieve over 86% mean accuracy.

3.1. Compact Network Architecture

Hourglass is one of the most common building block units for hand pose estimation. The hourglass architecture uses repeated bottom-up and top-down processing to extract features from different scales and the pose predictions are generated after passing through each hourglass module. Subsequent hourglass modules allow the high-level features to be processed again and can correct the mistakes made earlier by the network.

Figure 1 shows the architecture of the compact network. It consists of stack pruning (Figure 1a) and level pruning (Figure 1b). We use the dense 3D regression as the original network because it is highly precise. The 2D, 3D joint heatmaps and unit vector fields (yellow blocks) are estimated by hourglass in a multi-task learning manner and sequentially the 3D hand pose is estimated by enforcing consensus between the 3D detection and 3D estimation. In our pruning strategy, we decrease the number of stacks in half. Although a tiny network is attractive, it is not easy to train a small network that can achieve a similar accuracy target to the larger one. Reference [18] argued that training the small pruned network from scratch can almost always achieve a comparable or higher level of accuracy. However, in our situation, we find that training a one-stack tiny network performs poorly and it is most likely because the tiny network does not have enough generalization ability (learning power) and is more difficult to train. It can explain why most researchers attempt to stack identical structures to achieve better performance.

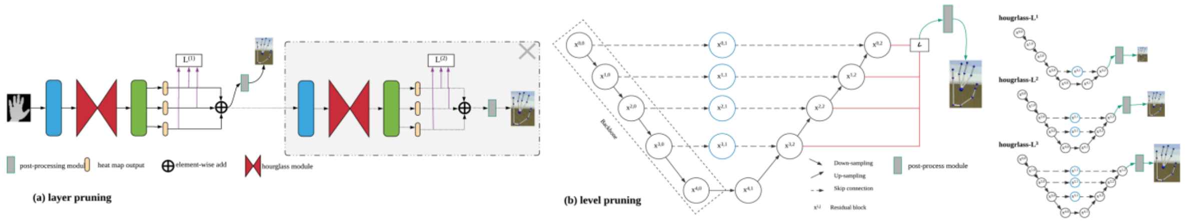

Figure 1. Compact network architecture. (**a**) Layer pruning: the second stack is pruned. Post-processing module includes coordinate reconstruction from three heat maps (yellow blocks). Blue and green blocks stand for a series of convolutional and pool operations which are omitted by limited space (refer to Reference [5] for details). (**b**) Level pruning: varying modes (L^1~L^4) can be selected for different complexity. Supervision modules are added to every decoder (red lines).

3.2. Hourglass with Deep Supervision

We propose using deep supervision in the hourglass module. Figure 1b shows the varying model complexity determined by the different choice of selected (pruned) branch. According to the hourglass module, different levels generate different resolution feature maps representing multiple semantic levels. We add supervision behind $\{x^{i,2}, i \in \{0,1,2,3\}\}$. We update the loss function with all the branch's losses and the loss function is described as:

$$\mathcal{L}_{ds} = \sum_{b=1}^{N}(\mathcal{L}_R^{(b)} + \mathcal{L}_S^{(b)} + \mathcal{L}_V^{(b)}) = \sum_{b=1}^{N}\sum_{j=1}^{J} ||R_j^{(b)} - R_j^{*}||^2 + ||S_j^{(b)} - S_j^{*}||^2 + ||V_j^{(b)} - V_j^{*}||^2 \quad (1)$$

where R_j^*, S_j^*, V_j^* represent the ground-truth 2D heat map, 3D heat maps and vector offsets of joint j and $R_j^{(b)}$, $S_j^{(b)}$, $V_j^{(b)}$ are the corresponding estimates from *b*th branch (level).

Owing to the skip connection (blue circles), the loss of accuracy caused by down-sampling can be compensated for and all the pruned networks have asymmetrical encoder-decoder architectures. The main drop of accuracy comes from the missing (pruned) decoder. However, in our experiment, we find the loss of accuracy is still always acceptable and the result can be explained in two aspects. Firstly, we average the loss with all semantics levels outputs and treat them equally importantly. The special position where we insert the deep supervision makes every decoder's feature maps as a proxy for the quality. By making appropriate use of the feature quality feedback, we can directly influence the update process of weights to get a highly discriminative feature map. This is the source of why deep supervision helps to increase performance. Secondly, there are no criteria for how deep an hourglass network should be. In general, higher resolution images should have a deeper network structure (128*128 for 4 ranks, 256*256 for 5 ranks). However, in a simpler regression task (for example the hand poses are not complicated and less self-occlusion), a deep network is sometimes redundant in the inference process. Therefore, we propose the compact network to trade off accuracy and efficiency.

3.3. Inference Optimization with OpenVINO

Different from most previous work, we mainly focus on the performance of CPUs because it is more practical if deep learning applications can be deployed on embedded devices or normal PCs and have fast execution during inference. Instead of depending on the expensive GPUs, we transform to leverage the resources of CPUs. The main advantage of GPUs over CPUs is the parallelism. However, the AVX (advanced vector extensions) instructions have extended the parallel execution capability. For example, with AVX-512, applications can pack 32 double precision and 64 single precision floating point operations per second per clock cycle within the 512-bit vectors, as well as eight 64-bit and sixteen 32-bit integers, with up to two 512-bit fused-multiply add (FMA) units. Thus, it is possible to speed up the DNN without any GPUs.

OpenVINO is a deep learning deployment toolkit which allows the deployment pre-trained deep learning models through a high-level Python or C++ inference engine API and supports many Intel platforms such as CPUs, VPU and FPGA and so forth. As shown in Figure 2, the toolkit comprises two components: model optimizer and inference engine. Model optimizer contains many state-of-the-art network optimization techniques such as path folding, batch normalization fusion, node merging, dead/no-op elimination and so forth. The inference engine contains the high-level inference API which can deliver the best hardware performance for each hardware type. The model optimizer converts the trained model to the intermediate representation (IR) and then runs on the inference engine.

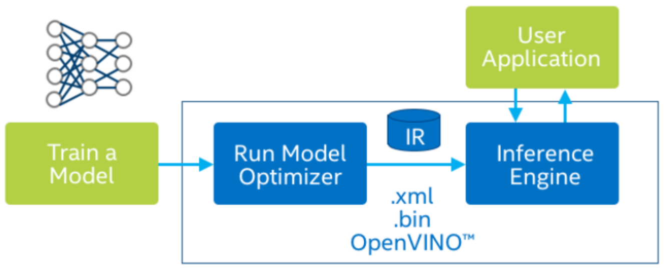

Figure 2. Workflow of OpenVINO toolkit: the trained model firstly optimized by the model optimizer to *IR* and then executed by the inference engine.

3.4. Implementation Details

In stack pruning, we cut off the second network stack without any training process. In level pruning, we add three supervision modules and retrain the one stack network. The network is implemented with Tensorflow and optimized using the Adam. We choose the checkpoint file carefully with the help of the validation set. We use a single NVIDIA Titan X for training and the Intel Core CPU (i9-7900X, 3.30GHz) during the testing to simulate the actual application scenarios. During training, we randomly rotate the image and change the aspect ratio for simple data augmentation.

During inference, the resulting network achieves a mean running time of 32 ms (31.25 FPS, the original is 22.7 FPS) and is 49% parameters size of the original one with only 16.3% loss increase (according to Table 1). Moreover, we implement the pipeline with the help of OpenVINO optimizer. It boosts the execution performance on Intel platforms and we will explain the details in Section 4.3.

Table 1. Comparison with the original network on NYU [1].

Architecture	**Average 3D Error**	**Prams**	**Inference Time (Per Frame)**
Wan et al. [5] (dense 3D)	10.2339 mm	5.83M	44 ms
1-stack-L_4 w/DS	10.9989 mm	2.92M	35 ms
1-stack-L_4 w/DS	10.6243 mm	2.92M	35 ms
1-stack-L_3 w/DS	11.9070 mm	2.86M	32 ms

Table 1. *Cont.*

Architecture	Average 3D Error	Prams	Inference Time (Per Frame)
1-stack-L_2 w/DS	15.5249 mm	2.80M	26 ms
1-stack-L_1 w/DS	43.2613 mm	2.75M	25 ms

DS: deep supervision; L_i: i^{th} level pruning; **M**: million.

4. Experiments

We conduct experiments on two publicly available datasets, that is, NYU and ICVL. We choose the NYU dataset to conduct the level-pruning and ablation experiments because it has broader coverage of hand poses as opposed to ICVL. We also implement the network with Intel OpenVINO toolkit. The toolkit extends workloads across Intel hardware and greatly maximizes performance.

To evaluate the efficiency of the model, we measure the parameters size and the running time per frame. To evaluate the accuracy quantitatively, we use two metrics: mean joint error (in mm) averaged over all joints and all frames and the percentage of frames in which all joints are below a certain threshold (in mm). To evaluate the whole pipeline performance, we break it down to three parts: pre-processing (crop, padding, etc.), regression network, the mean-shift algorithm (network output to estimated pose) and measure their efficiency separately. The results illustrate the Intel OpenVINO toolkit can optimize the pipeline's inference process in Intel CPUs and achieve the similar or even better performance than NVIDIA Titan X.

4.1. Baseline

In this section, we analyze the redundancy of models and the reasons why our pruning strategy works. Besides, we investigate the dominant layers and the benefits brought from deep supervision.

4.1.1. What Dominates the Accuracy of the Stacked Hourglass?

We examined the design of the popular stacked-hourglass network regarding feature number and stack number. To this end, we mainly tested two dimensions: depth (layers) and width (the channel/feature number). We revealed that the channel number has less impact on the accuracy than layer number.

From Table 2, we find that the half feature number only leads to quite limited performance degradation (when the stack number is the same). However, the half stack leads a relatively large performance collapse. As we discussed before, the result proved that training a small model from scratch is challenging and cannot always achieve the similar accuracy target as a complicated model because of the lack of generalization capability for large amounts of data. This is one of the reasons why we use network pruning: we want to exploit both the generalization capability from large models and the lightweight from small models.

Table 2. Impact of stack number and feature number.

# Stack	# Feature Number	Mean Joint Error (from Scratch)	Mean Joint Error (Knowledge Distillation)
1	128	14.0880 mm	16.5343 mm
1	64	14.0503 mm	16.7141 mm
2	128	10.2339 mm	/

We investigated which part of the hourglass should be pruned. In general, we should remove weights with small magnitudes (less important). According to Figure 3, we compare the L1-norm between different stages of the original network and decide to prune the second stack which has the smaller L1-norm. L1-norm measures the importance of layers by calculating the sum of its absolute weights $\sum|F_{i,j}|$ (i, j is the index of any kernel matrix). We use L1-norm because it is a good criterion for data-free selection [19].

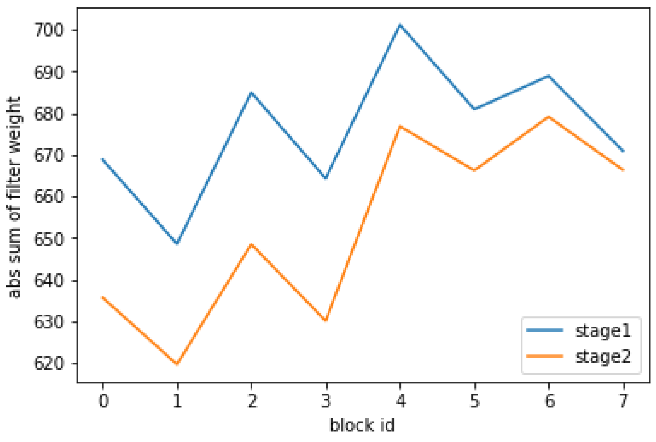

Figure 3. L1-norm of two stacks, block id stands for the id of skip and ResNet blocks.

4.1.2. Impact of Deep Supervision

To explore the impact of deep supervision, we implement it and retrain the one-stack network. The results from Table 3 show how the *prune and retrain* technique can complement the error increase caused by pruning. Training from scratch performed worse than retraining from pruned network. Instead, retraining with or without deep supervision has a similar accuracy performance (the difference is only 0.062mm).

Table 3. Impact of deep supervision.

Method	Pruned Network	Train from Scratch w/o DS	Retrain w/o DS	Retrain w/DS
Mean joint error	10.999 mm	12.204 mm	10.562 mm	10.624 mm

Retraining compensates for the accuracy drop. Retrain with or without deep supervision has similar mean joint error but retraining w/DS help to converge faster.

We deeply analyze the retraining process. As shown in Figure 4, although the network with deep supervision has a higher initial loss, the convergence rate is bigger than that without deep supervision. For example, as shown in Figure 4 (where the arrow points), the training process with deep supervision converges at step 4K and that without deep supervision only converges at step 10K. The reason why the Figure 4a has higher loss is that the two strategies have different loss function which can be found in Equation 1. The loss function of Figure 4a is the sum of different level branches, so it has a bigger loss than Figure 4b.

Thus, the strategy of deep supervision can not only provide the possibility of further pruning but also performs the shorter training process.

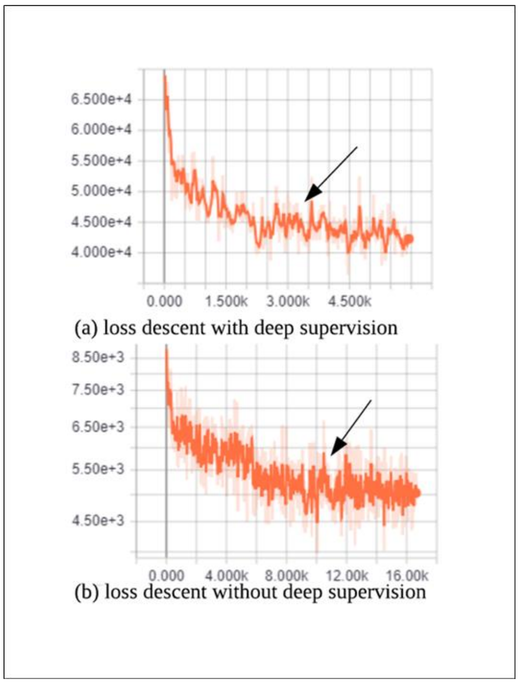

Figure 4. Loss descent comparison of w/ and w/o deep supervision (NYU dataset [1]).

4.2. *Comparision of State-of-the-Art*

We evaluated our proposed lightweight hand pose estimation method against the state-of-the-art methods on the NYU and ICVL dataset. The results show that our pruned network has traded off the accuracy and efficiency and achieve a satisfactory target on both datasets.

NYU Dataset [1]: The NYU hand dataset contains 72,757 training and 8252 testing samples (each view). The depth data are captured using MS Kinects from three points of view (left, right and middle). It covers a wide range of hand poses and also some noisy depth images which makes the dataset challenging. We only use 14 joints and one view (middle) for both training and testing despite that the dataset has 36 key hand locations.

We compare our method with the original regression network. According to Table 1, the 1-stack-L_4 w/DS method gets 6 FPS boost with only 4% ((10.64-10.23)/10.23) error increase. Also, 1-stack-L_3 w/DS achieve 9 FPS boost with 16.3% error increase, which is an acceptable trade-off. Similar to Table 3, deep supervision helps to converge faster but still has approximate error. As shown in Figure 5, our pruned network increases the percentage on the threshold of 10mm and has the close max joint error percentage compared to the original network.

However, we notice that a large level-pruning has the lousy result (1-stack-L_1 w/DS). It is because the L_1 branch only has a $4 * 4$ resolution, which is not enough for an accurate prediction. We also present the qualitative results in Figure 6. Failure cases include noisy images and severe self-occlusions shown in Figure 6b.

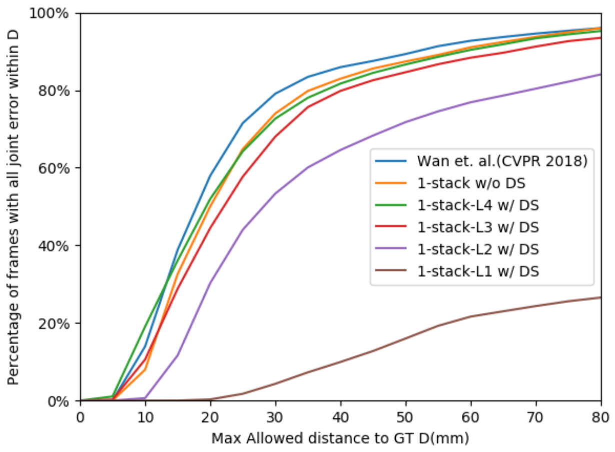

Figure 5. Comparison of the original network on NYU [1]. Percentage of frames in which all joints are below a threshold is plotted.

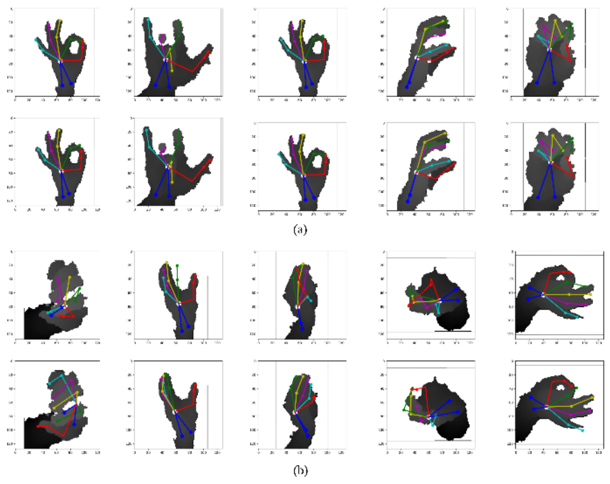

Figure 6. Qualitative results. Hand pose estimation results on NYU [1] dataset. (**a**) Successful samples (top row) (**b**) Failed samples (top row) and the corresponding ground-truth (bottom row).

ICVL Dataset [13]: The ICVL 3D hand posture dataset has 22K depth images for training and 1.5K for testing. The richness of pose is much smaller than the NYU dataset. ICVL has 16 joint locations from each frame and the data is captured using Intel Creative depth sensor. This dataset has less noisy input, owing to the sensor. We also select 14 joints to be consistent with the NYU dataset.

As is shown in Table 4, our method has a stable performance in the ICVL dataset. Table 4 shows that 0.17mm average 3D error sacrifice can bring 20% speed boost. Similar to NYU dataset, the reason for failure samples is severe self-occlusions (as shown in Figure 7). According to Figure 8, the percentage of max joint error is very close compared to the original network.

Table 4. Comparison with the original network on ICVL [13].

Architecture	**Average 3D Error**	**Inference Time**
Wan et al. [1] (dense 3D)	7.25 mm	47 ms
1-stack-L_4 w/o DS	7.42 mm	38 ms

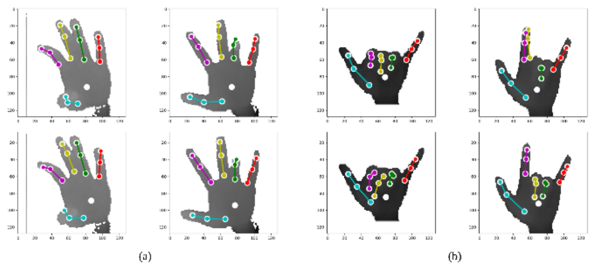

Figure 7. Qualitative results. Hand pose estimation results on ICVL dataset [13]. (**a**) Successful samples (top row) (**b**) Failed samples (top row) and the corresponding ground-truth (bottom row).

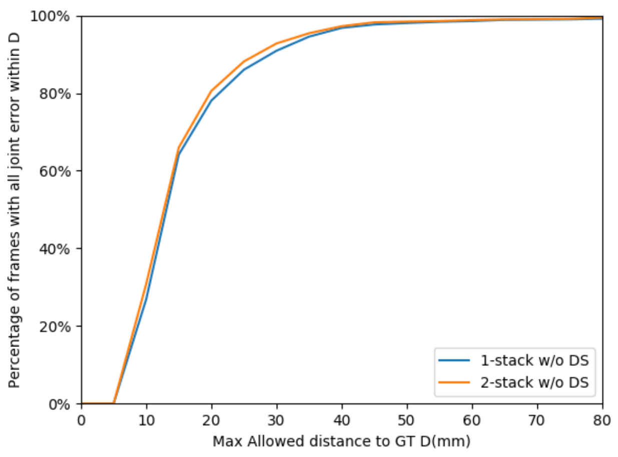

Figure 8. Comparison with state-of-the-art on ICVL [13]. Percentage of frames in which all joints are below a threshold is plotted.

4.3. OpenVINO Optimization

We break down the pipeline into three parts: pre-processing (crop, align and normalization), regression network (regression for heat map estimation) and mean-shift algorithm (coordinate 2D and 3D heat map to restructure hand poses). We find that the regression network is not the only impact of inference time. From Table 5, we find the mean-shift algorithm takes much time on GPU, which is even longer than the time of regression network (17.6 ms versus 3.6 ms). This is because the mean-shift algorithm cannot benefit much from parallel computing and requires fast data transmission. That can also explain why some small networks can perform better on CPU than on GPU.

Table 5. Comparison of pipeline time on different devices.

Framework	Device	Preprocessing	Regression Network	Mean-Shift Algorithm	Total
Tensorflow	GPU (TITAN X)	1.5 ms	3.6 ms	17.6 ms	22.7 ms (44FPS)
	CPU (i9)	1.3 ms	20.8 ms	4.8 ms	26.9 ms (37FPS)
OpenVINO	CPU (i9)	1.4 ms	11.4 ms	5.0 ms	17.8 ms (56FPS)

Regression network: 1-stack-L_3 w/DS.

As shown in Table 6, we compare the execution time on different framework and device for our proposed network (1-stack-L_3 w/DS). The results show that the OpenVINO framework can boost the inference process of regression network on CPUs (row 2 and 3). Moreover, the benefits brought by faster mean-shift algorithm execution speed can further decrease the total inference time of the pipeline (row 1 and 3). Therefore, our method runs faster on CPUs (56 FPS versus 44 FPS).

Compared to state-of-the-art methods shown in Table 6, our method achieves comparable accuracy and the real-time goal.

Table 6. Comparison with state-of-the-art on different devices.

Method	**Average 3D Error**	**Use CPU**	**FPS**
DeepPrior++ [21]	12.3 mm	×	30
Crossing Nets [28]	15.5 mm	√	91
Dense 3D [5]	10.2 mm	√	23
Ours	11.9 mm	√	56

4.4. Exploration Studies

We implement the existing network compression technique (knowledge distillation [8]) to compare it with our method. Knowledge distillation is the approach of training student networks from the supervision of a teacher network. We consider the 2-stack network as the teacher model and 1-stack network as the resulting student model. Subsequently, the student network is trained under the supervision of a teacher network. The results are appended to Table 2 and show an even worse result than training from scratch. A possible explanation is that knowledge distillation is inadequate for the multi-task regression problem instead of for the classification problem. Previous network compression works mainly focus on compressing universal networks such as VGG-16, ResNet and so forth. However, we notice the general-purpose compression methods may not apply to all kinds of networks, more dedicated strategies should be considered for better performance targets.

5. Conclusions and Discussion

We propose a lightweight architecture for real-time 3D hand pose estimation from depth images. Given the original network, we make two-level pruning under deep supervision and further boost the inference performance on the CPU platform. This is made possible to scale up hand pose estimation applications in reality. We have carried out extensive comparative evaluations on two 3D hand pose datasets and different hardware. The results show our network gains both cost-effectiveness and high accuracy. For practical applications, we prune the pre-trained network before inference and retrain to improve the reduced accuracy. Subsequently, we use the structure-optimized network in inference time.

Also, the inference time on CPUs can outperform that on GPUs with the help of new instructions and a newly proposed toolkit. Further optimization directions include pipeline throughput, heterogeneous computation and batch process.

Author Contributions: Conceptualization, Y.W. and S.D.; Data curation, X.R.; Methodology, Y.Z.; Supervision, G.W.; Validation, H.Z.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tompson, J.; Stein, M.; Lecun, Y.; Perlin, K. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Trans. Gr. (TOG)* **2014**, *33*, 169. [CrossRef]
2. Oberweger, M.; Wohlhart, P.; Lepetitt, V. Hands deep in deep learning for hand pose estimation. 2015. Available online: https://arxiv.org/pdf/1502.06807 (accessed on 4 April 2019).
3. Ye, Q.; Yuan, S.; Kim, T.K. Spatial attention deep net with partial PSO for hierarchical hybrid hand pose estimation. In *Lecture Notes in Computer Science*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016.
4. Ge, L.H.; Liang, H.; Yuan, J.; Thalmann, D. Robust 3D hand pose estimation in single depth images: From singleview cnn to multi-view CNNs. 2016. Available online: https://arxiv.org/pdf/1606.07253 (accessed on 4 April 2019).
5. Wan, C.; Probst, T.; Van Gool, L.; Yao, A. Dense 3D Regression for Hand Pose. Available online: https://arxiv.org/pdf/1711.08996 (accessed on 4 April 2019).

6. Newell, A.; Yang, K.; Deng, J. Stacked hourglass networks for human pose estimation. *ECCV*. In *Computer Vision.*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; Volume 9912, pp. 483–499.
7. Matthieu, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; Bengio, Y.S. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. 2016. Available online: https://arxiv.org/pdf/1602.02830 (accessed on 4 April 2019).
8. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. 2015. Available online: https://arxiv.org/pdf/1503.02531?context=cs (accessed on 4 April 2019).
9. Han, S.; Mao, H.Z.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. 2015. Available online: https://arxiv.org/pdf/1510.00149 (accessed on 4 April 2019).
10. Castellano, G.; Fanelli, A.M.; Pelillo, M. An iterative pruning algorithm for feed forward neural networks. *IEEE Trans. Neural Netw.* **1997**, *8*, 519. [CrossRef] [PubMed]
11. Lécun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
12. Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems. 2015. Available online: https://arxiv.org/pdf/1506.02626 (accessed on 4 April 2019).
13. Tang, D.; Chang, H.J.; Tejani, A.; Kim, T.K. Latent regression forest: Structured estimation of 3D articulated hand posture. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014.
14. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2012**, *60*, 84–90. [CrossRef]
15. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, Malaysia, 3–6 November 2015.
16. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
17. Anwar, S.; Hwang, K.; Sung, W.Y. Structured Pruning of Deep Convolutional Neural Networks. 2015. Available online: https://arxiv.org/pdf/1512.08571 (accessed on 4 April 2019).
18. Polyak, A.; Wolf, L. Channel-Level Acceleration of Deep Face Representations. *IEEE Access* **2015**, *3*, 2163–2175. [CrossRef]
19. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning Filters for Efficient ConvNets. 2016. Available online: https://arxiv.org/pdf/1608.08710 (accessed on 4 April 2019).
20. Zhang, F.; Zhu, X.T.; Ye, M. Fast Human Pose Estimation. 2019. Available online: https://arxiv.org/pdf/1811.05419 (accessed on 4 April 2019).
21. Oberweger, M.; Lepetit, V. Deepprior++: Improving fast and accurate 3D hand pose estimation. In Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy, 22–29 October 2017.
22. Chen, X.; Wang, G.; Guo, H.; Zhang, C. Pose Guided Structured Region Ensemble Network for Cascaded Hand Pose Estimation. Available online: https://arxiv.org/pdf/1708.03416 (accessed on 4 April 2019).
23. Guo, H.K.; Wang, G.J.; Chen, X.H.; Zhang, C.R.; Qiao, F.; Yang, H.Z. Region Ensemble Network: Improving Convolutional Network for Hand Pose Estimation. 2017. Available online: https://arxiv.org/pdf/1702.02447 (accessed on 4 April 2019).
24. Olaf, R.; Fischer, P.; Brox, T. U-net: Convolutional Networks for Biomedical Image Segmentation. 2015. Available online: https://arxiv.org/pdf/1505.04597 (accessed on 4 April 2019).
25. Huang, G.; Liu, Z.; Laurens, V.D.M.; Weinberger, K.Q. Densely Connected Convolutional Networks. 2016. Available online: https://arxiv.org/pdf/1608.06993 (accessed on 4 April 2019).
26. Li, X.; Chen, H.; Qi, X.; Dou, Q.; Fu, C.W.; Heng, P.A. H-DenseUNet: Hybrid densely connected UNet for liver and liver tumor segmentation from CT volumes. *IEEE Trans. Med. Imaging* **2018**, *37*, 2663–2674. [CrossRef] [PubMed]

27. Drozdzal, M.; Vorontsov, E.; Chartrand, G.; Kadoury, S.; Pal, C. The importance of skip connections in biomedical image segmentation. In *Deep Learning and Data Labeling for Medical Applications*; Carneiro, G., Ed.; Springer: Cham, Switzerland, 2016.
28. Zhou, Z.W.; Siddiquee, M.R.; Tajbakhsh, N.; Liang, J.M. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*; Springer: Cham, Switzerland, 2018; pp. 3–11.
29. Wang, L.; Lee, C.Y.; Tu, Z.; Lazebnik, S. Training Deeper Convolutional Networks with Deep Supervision. 2015. Available online: https://arxiv.org/pdf/1505.02496 (accessed on 4 April 2019).
30. Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H.; Montral, U.D.; Qubec, M. Greedy layer-wise training of deep networks. In Proceedings of the 19th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 4–7 December 2006; MIT Press: Cambridge, MA, USA, 2006; pp. 153–160.
31. Lee, C.Y.; Xie, S.; Gallagher, P.; Zhang, Z.Y.; Tu, Z. Deeply-Supervised Nets. 2015. Available online: https://arxiv.org/pdf/1409.5185 (accessed on 4 April 2019).
32. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
33. Carreira, J.; Agrawal, P.; Fragkiadaki, K.; Malik, J. Human pose estimation with iterative error feedback. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
34. Wei, S.E.; Ramakrishna, V.; Kanade, T.; Sheikh, Y. Convolutional pose machines. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
35. Chu, X.; Yang, W.; Ouyang, W.; Ma, C.; Yuille, A.L.; Wang, X. Multi-context attention for human pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
36. Yang, W.; Li, S.; Ouyang, W.; Li, H.; Wang, X. Learning feature pyramids for human pose estimation. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.