CV701 Assignment-4

Mothilal Asokan **Murtadha Aljubran** **Santosh Sanjeev**
Mohammed bin Zayed University of Aritifical Intelligence
Masdar City, Abu Dhabi, UAE
{mothilal.asokan, murtadha.aljubran, santosh.sanjeev}@mbzuai.ac.ae

1. Introduction

Human pose estimation (HPE) is one the areas that can result in many useful and practical applications. In HPE, we estimate the pose of a person by identifying the locations of 13 multiple points at different areas of the body. Those locations include head, right/left shoulders, right/left elbows, right/left wrists, right/left hips, right/left knees, and lastly right/left ankles. Estimating right/left parts proves to be even more difficult due to being smaller and being inconspicuous parts. In addition, occlusion adds a further layer of complexity in HPE task especially in the case of estimating the pose of multiple persons in the image. It is imperative to have knowledge about relative positioning of joints for position for one person in order to have a more accurate estimation. In other words, we need to a model that can learn local features, such as recognizing what a knee looks like, as well as learn global features, such as interactions between the local feature to form the overall pose.

2. Objective

The objective of our work is to train and evaluate a pose estimation model (stacked hourglass model) on the MPII Human Pose dataset. Our task is to reduce the computational complexity of the baseline hourglass model without largely sacrificing the accuracy. More details about the model will be discussed in the next section. It is important to understand the computational complexity in this context extend to the environment we are bench-marked against which could be independent of the model size as we will observe when discussing the results.

3. Experiments

3.1. Baseline Architecture

The baseline architecture explored in our work is inspired by the stacked hourglass design in [4]. As shown in figures 1 and 2 [4], the baseline architecture is composed of multiple hourglasses stacked vertically.

In our exploration, we used two hourglass modules where each component of the hourglass is a single bottleneck residual block as shown in figure 3a. The image is downsampled at the beginning by a stem layer with 7x7 kernel. The total number of parameters of this baseline model is around 6.73M.

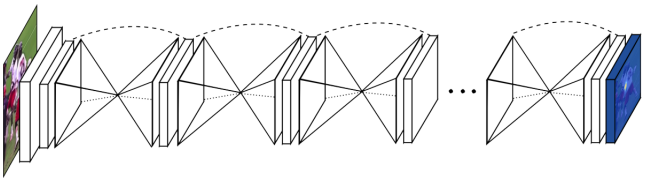


Figure 1: Stacked hourglass modules representing the whole architecture with skip connections. This architecture allows for repeated bottom-up and top-down inference.

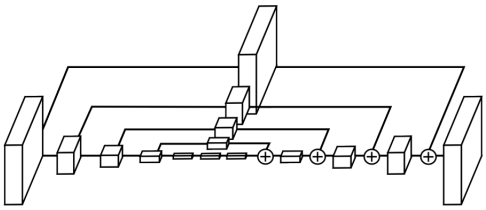
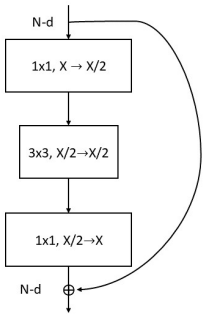
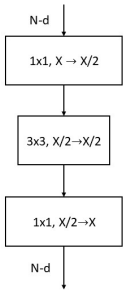


Figure 2: One hourglass module which resembles a U-Net architecture where each component is a residual block module with skip connections between downsampling and upsampling layers.



(a) Baseline Bottleneck residual block



(b) Bottleneck residual block without skip connection

Figure 3: Bottleneck module used in hourglass (X: number of feature maps, NxN: dimensions of image)

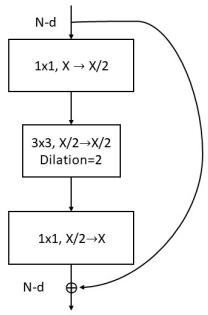
3.2. Where to optimize?

The HPE baseline architecture had many components to explore and optimize. Since the training time of the baseline model is around 2.5 hours, it would be quite difficult to test out many ideas. Hence, we designed two experiments at the beginning to identify if we should focus on enhancing learning over either local features or global features. We basically added a weighted sum between the activation maps from each hourglass to the next one, whereas in one exper-

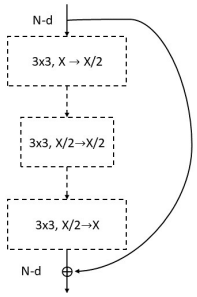
iment we used weights [0.3, 0.7], and weights [0.7, 0.3] in the other experiments to the output of first and second hourglass modules respectively.

3.3. Dataset and Validation Measure

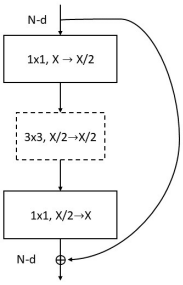
In our investigation of HPE task, we use MPII Human Pose dataset which is a benchmark for evaluating the system addressing HPE challenges [1]. The dataset consists of 25 thousand images covering over 40 thousand people with annotated body parts. In order to evaluate the effectiveness of our proposed methods, we use the percentage of correct key-points on the head (PCKh). A point is considered to be correctly classified if it gets detected at a distance not more than 50% percent of head length.



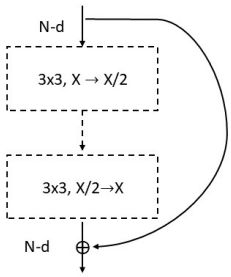
(a) Dilated Convolution



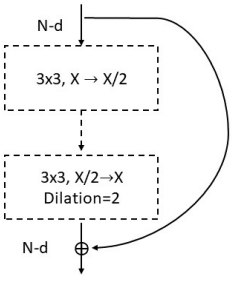
(b) 3x3 Depthwise Separable Convolutions



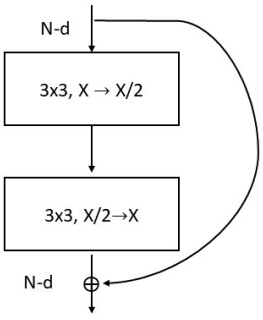
(c) Depthwise Separable Convolutions in only 3x3 layer



(d) Two 3x3 Depthwise Separable Convolutions



(e) Two 3x3 Dilated Depthwise Separable Convolutions



(f) Normal 2D Convolutions

Figure 4: Different Hourglass Block Designs.

3.4. Different Hourglass Block Designs

In order to reduce computational complexity, we explored multiple changes in the architecture's design. Our aim was to find a more efficient block design utilizing much less number of parameters without sacrificing alot on accuracy. In all our experiments, we used the same optimization parameters: 20 epochs, batch size of 24, and fixed learning rate of 0.001 which is reduced by a factor of 10 at epochs 15 and 17.

3.4.1 Dilated Convolution in Bottleneck

In this experiment, we added a dilation of 2 to the middle block of the original bottleneck residual block as shown in 4a. Dilation effectively increases the receptive field of the 3x3 kernel to 5x5 while using the same number of parameters. There is no reduction in the total number of parameters in this case.

3.4.2 Patchify technique used in ConvNext

In [3] they replaced the stem layer to match the Transformer model. Instead of a 7x7 kernel with stride 2 they used a 4x4 kernel with stride 4. We approximated the patchify with 3x3 convolutional layer and stride 2 instead of using 3x3 with stride 3 and again upsampling the output of the convolutional layer to match the dimensions.

3.4.3 Removal of skip connections in Bottleneck

We conducted another experiment to evaluate the importance of skip connections in the Bottleneck layer. We removed the skip connection in the bottleneck to understand the influence of skip connection on runtime and accuracy as shown in figure 3b. We conducted some of the other experiments in combination with this experiment.

3.4.4 Depthwise Separable Convolution in Bottleneck

We also converted only the 3x3 layer in the bottleneck of the baseline to depthwise separable convolution as shown in figure 4c.
Similarly we tried to use two separable depth-wise convolution layers as replacement to the original bottleneck residual block as shown in figure 4d.
Finally, we converted all the layers to separable depthwise convolutions having 3x3 convolutions. The 1x1 layers in the baseline are replaced by 3x3 depthwise separable convolutional layers as shown in figure 4b.

All these experiments reduce the number of parameters significantly. However, each separable depth-wise convolution is composed of two layers which turn out to be of importance as we will see in the results.

3.4.5 Dilated Depthwise Separable Convolution in Bottleneck

We also tried to combine both dilation with two separable depth-wise convolution layers as shown in figure 4e. In this case, we wouldn't have any increase in number of parameters while we allow for a bigger receptive field in the network. This would help in learning the global features.

3.4.6 Convolution Layers without Bottleneck

In this special experiment, we removed the three convolution layers in the original bottleneck residual block with two regular convolution layers as shown in figure 4f. This would increase the number of parameters significantly; nevertheless, it would reduce the number of layers or kernel launches required while training the network.

4. Results and Discussion

Before conducting any experiment, we probed the baseline model by changing the weighted sum between the activation maps outputs in the two hourglass modules. As shown in Table 1, we notice that the network perform worse when up-weight the activation maps of the first hourglass model output activation maps. This signifies that it is more difficult to learn global features in the architecture. All the discussion in the next sub sections refer to the results in Table 1.

4.1. Different Hourglass Block Designs

We summarize the multiple approaches we tried into the results of 4 major architecture changes to the hourglass module block design in addition to the baseline. Since the training time is affected by the environment and machine we use for a specific experiment, we would express the training time in relative measures.

4.1.1 Dilated Convolution in Bottleneck

Unlike what we expected, using dilated convolution in bottleneck has resulted in slight drop in validation accuracy with mean validation PCH. This could be attributed that there is a loss of some information in the network due to dilation. In order to address this issue, we implemented the multidilated residual block [2]. However, we didn't complete the training because of the extreme slow training progress in addition to the significant increase in number of parameters.

4.1.2 Patchify technique used in ConvNext

There was not much variation in the metrics using the Patchify technique and the contribution to the reduction of parameters was also not significant.

4.1.3 Separable Depth-wise Convolution in Bottleneck

This configuration reduces the network size by around 3.5x factor. The network scores a PCKh@0.5 mean of 72.19% using 2 depthwise separable convolutions in the bottleneck. Varying with only 1 and all 3 depthwise separable convolutions we observed significant difference and a high PCKh@0.5 score of 78.67(using all 3 depthwise separable convolutions). But as each layer is converted to a depthwise and a pointwise, the number of layers increased which might contribute to the longer time regardless of the network being significantly smaller.

4.1.4 Removal of skip connections in Bottleneck

As expected there was a decrease in the PCKh@0.5 mean score due to the removal of skip connections, but there was not a significant improvement in other parameters.

4.1.5 Dilated Separable Depth-wise Convolution in Bottleneck

Adding dilation to one of the separable depth-wise convolution layers didn't have any change in training time while maintaining the same number of parameters as without using dilation. The performance has decreased slightly which could be attributed to the same reasons discussed in section 4.1.1.

4.2. Convolution Layers without Bottleneck

By using two normal 3x3 convolution layers, the number of parameters has significantly increased about 3x times the baseline architecture size. However, this network achieves the best performance in terms of PCKh@0.5 as shown in table 1 while requiring significantly less training time. This was counter intuitive at the first look until we realize the major factor affecting the training time isn't the architecture size but rather the number of layers in the network.

It is worth to note that the architecture has two convolution layers rather than one bottleneck residual layer. This allows for a better generalization.

4.3. Training Time Analysis

As discussed in the last section, we observed that the training time wasn't a function of the number of parameters. It was rather a function of number of layers or required kernel launches while training the network. Therefore, we notice that the two normal convolution layer network run faster than the bottleneck residual block which has three layers. The two separable depth-wise convolution layers have four layers which explains why such architecture requires more training time regardless of have the lowest number of parameters.

Experiment Configuration	Validation Accuracy PCKh@0.5 (%)								No. of Parameters	Avg. Training Time(epochs*)
	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	Mean		
Baseline	93.18	89.16	79.10	72.61	76.89	69.72	62.66	77.84	6.73M	2hr 30min(20)
Weights [0.3, 0.7] applied to HG2 outputs	92.50	88.71	77.77	70.64	75.71	67.12	60.04	76.33		
Weights [0.7, 0.3] applied to HG2 outputs	90.35	81.69	69.17	58.88	66.94	56.62	49.46	67.84		
Dilated Convolution	92.84	87.36	76.61	69.18	73.86	68.15	61.22	75.85		
Patchify technique from ConvNext	91.78	87.16	77.57	70.22	73.31	67.48	60.18	75.66	6.72M	2hr 34min(20)
Residual block without Skip connection in Bottleneck	92.84	88.03	78.32	71.42	75.97	68.31	61.36	76.84	6.69M	2hr 28min(20)
3x3 Depthwise Separable Convolutions	93.25	89.86	80.55	72.57	78.05	70.56	64.27	78.67	2.92M	2hr 20min(17)
Depthwise Separable Convolutions in only 3x3 layer	93.11	88.67	78.73	71.50	75.89	68.81	61.83	77.15	2.80M	2hr 30min(20)
Two 3x3 Depthwise Separable Convolutions	92.19	85.09	73.55	63.80	72.17	62.42	54.77	72.19	1.90M	2hr 40min(20)
Two 3x3 Dilated Depthwise Separable Convolutions	91.30	84.17	71.83	62.36	71.59	61.13	54.77	71.28		
Convolution Layers without Bottleneck	92.67	89.50	80.65	73.19	78.26	70.42	62.97	78.52	17.95M	2hr 10min(17)

Table 1: This table summarizes our experiments in investigation the performance and training time of different hourglass block designs. All experiments had the same settings of no. of epochs, LR scheduler, and batch size.
*20 epochs or minimum epochs to reach baseline mean score.

5. Conclusion

We learned that the number of parameters in the network isn't necessarily the most important factor to focus on while optimizing the training time. This is especially true for deep networks. It is important to understand what is the overhead involved in loading the data and running computations withing the GPU accelerators. The number of layers proved to be an important factor to consider. We were able to use a bigger network to achieve better performance with less training time.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2

[2] Seung-Taek Kim and Hyo Jong Lee. Lightweight stacked hourglass network for human pose estimation. *Applied Sciences*, 10(18), 2020. 3

[3] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022. 2

[4] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 483–499, Cham, 2016. Springer International Publishing. 1