Appendices

A. Efficiency of BlockQNN

We demonstrate the effectiveness of our proposed BlockQNN on network architecture generation on the CIFAR-100 dataset as compared to random search given an equivalent amount of training iterations, *i.e*. number of sampled networks. We define the effectiveness of a network architecture auto-generation algorithm as the increase in top auto-generated network performance from the initial random exploration to exploitation, since we aim to getting optimal auto-generated network instead of promoting the average performance.

Figure 1 shows the performance of BlockQNN and random search (RS) for a complete training process, *i.e*. sampling $11,392$ blocks in total. We can find that the best model generated by BlockQNN is markedly better than the best model found by RS by over 1% in the exploitation phase on CIFAR-100 dataset. We observe this in the mean performance of the top-5 models generated by BlockQNN compares to RS. Note that the compared random search method start from the same exploration phase as BlockQNN for fairness.

Figure 2 shows the performance of BlockQNN with limited parameters and adaptive block numbers (BlockQNN-L) and random search with limited parameters and adaptive block numbers (RS-L) for a complete training process. We can see the same phenomenon, BlockQNN-L outperform RS-L by over 1% in the exploitation phase. These results prove that our BlockQNN can learn to generate better network architectures rather than random search.

B. Evolutionary Process of Auto-Generated Blocks

We sample the block structures with median performance generated by our approach in different stage, *i.e*. at iteration $[1, 30, 60, 90, 110, 130, 150, 170]$, to show the evolutionary process. As illustrated in Figure 3 and Figure 4, *i.e*. BlockQNN and BlockQNN-L respectively, the block structures generated in the random exploration stage is much simpler than the structures generated in the exploitation stage.

In the exploitation stage, the multi-branch structures appear frequently. Note that the connection numbers is gradually increase and the block tend choose "Concat" as the last layer. And we can find that the short-cut connections and elemental add layers are common in the exploitation stage. Additionally, blocks generated by BlockQNN-L have less "Conv,5" layers, *i.e*. convolution layer with kernel size of 5, since the limitation of the parameters.

These prove that our approach can learn the universal design concepts for good network blocks. Compare to other automatic network architecture design methods, our generated networks are more elegant and model explicable.

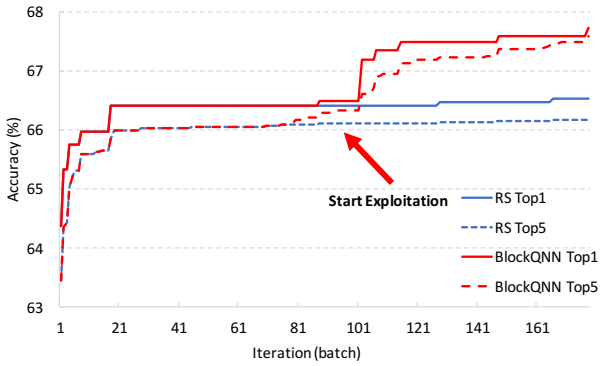

Figure 1. Measuring the efficiency of BlockQNN to random search (RS) for learning neural architectures. The x-axis measures the training iterations (batch size is 64), *i.e*. total number of architectures sampled, and the y-axis is the early stop performance after 12 epochs on CIFAR-100 training. Each pair of curves measures the mean accuracy across top ranking models generated by each algorithm. Best viewed in color.

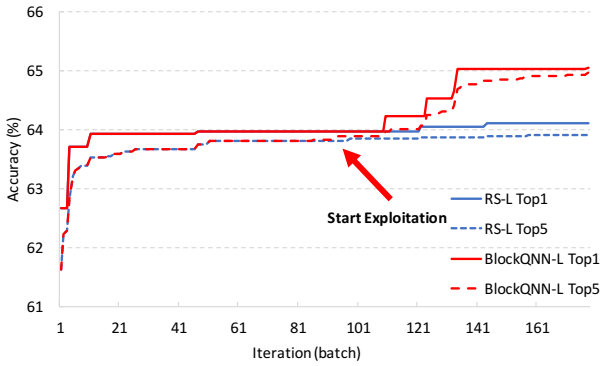

Figure 2. Measuring the efficiency of BlockQNN with limited parameters and adaptive block numbers (BlockQNN-L) to random search with limited parameters and adaptive block numbers (RS-L) for learning neural architectures. The x-axis measures the training iterations (batch size is 64), *i.e*. total number of architectures sampled, and the y-axis is the early stop performance after 12 epochs on CIFAR-100 training. Each pair of curves measures the mean accuracy across top ranking models generated by each algorithm. Best viewed in color.

C. Additional Experiment

We also use BlockQNN to generate optimal model on person key-points task. The training process is conducted on MPII dataset, and then, we transfer the best model found in MPII to COCO challenge. It costs 5 days to complete the searching process. The auto-generated network for key-points task outperform the state-of-the-art hourglass 2

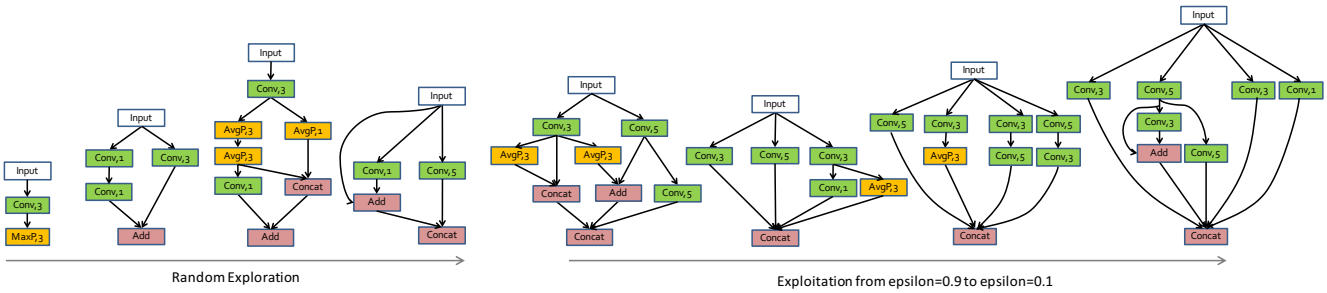

Figure 3. Evolutionary process of blocks generated by BlockQNN. We sample the block structures with median performance at iteration $[1, 30, 60, 90, 110, 130, 150, 170]$ to compare the difference between the blocks in the random exploration stage and the blocks in the exploitation stage.

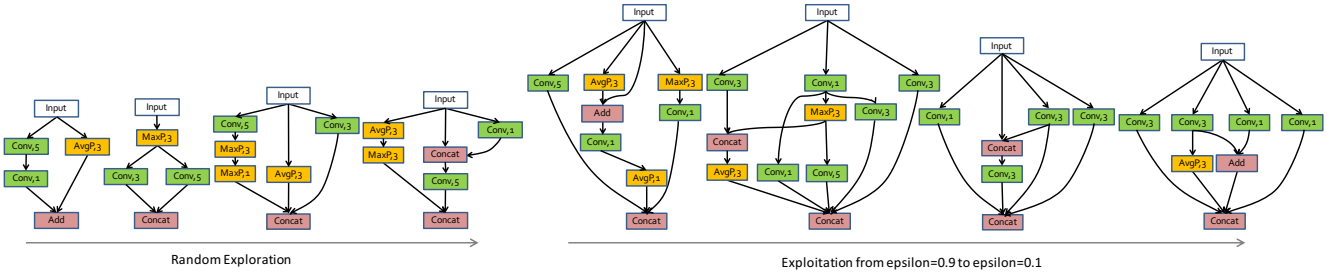

Figure 4. Evolutionary process of blocks generated by BlockQNN with limited parameters and adaptive block numbers (BlockQNN-L). We sample the block structures with median performance at iteration $[1, 30, 60, 90, 110, 130, 150, 170]$ to compare the difference between the blocks in the random exploration stage and the blocks in the exploitation stage.

stacks network, *i.e*. 70.5 AP compares to 70.1 AP on COCO validation dataset.