(Neural) Architecture Search

Eilif Solberg

August 29, 2018

When a group of researchers from Toronto won the ImageNet competition in 2012 with a convolutional neural network now know as "AlexNet" [1], they set off a race for even better architectures. It turns out that the researchers of Toronto did *not* happen to stumble upon the *best* architecture, and the top-5 error rate on ImageNet has decreased from over 18% in 2012 to less than 4% today[1]. Hundreds of researchers have been involved in this quest for improved architectures and through these efforts some general principles for well-performing convolutional architectures have emerged. For image recognition we have seen that in general

- Smaller convolutions e.g. 3×3 or 5×5 works better than larger ones.
- Batch-normalization works well, and may actually replace Dropout in many scenarios.
- We should do batch-normalization *before* e.g. ReLU activation, not after it.
- Zero-padding to keep the output dimension the same is a good idea.
- Residual connections, or other ways of shortening the error path, improves performance.

However, new ideas come out all the time and we would like to improve architectures even further. With all the possible ideas, though, the number of possible configurations gets enormous. We also have to consider the possibility that new structures may even invalidate some of the general principles mentioned above. And what about other applications, where we have less experience and may not yet have guiding principles we can trust? What

[1] Single network, i.e. not ensemble, performance. The 18% score is obtained through averaging predictions on 10 crops.

we would like is an algorithm that could *search* through the huge space of architectures. Some attempts in this direction have indeed been made and we shall look into a couple of these. There are at least two important parts to algorithmic architecture search

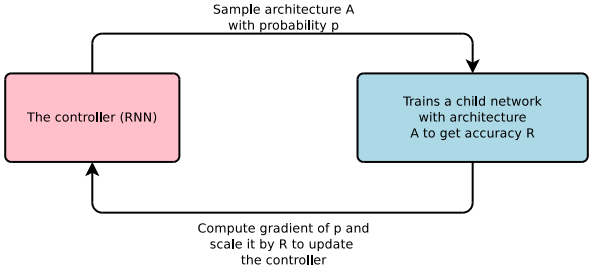
1. Defining an appropriate search space.
2. Deciding upon the optimization algorithm to find a good architecture *within* the search space.

We will look at two approaches that use similar optimization algorithms, but differs significantly in the way they define the search space. We will look at how they go about finding the best architecture on the CIFAR-10 image recognition dataset, though they have much broader applicability than this.

First we shall look at the approach in [2]. Quite a bit of the structure of the architecture is actually fixed. All networks in the search space is a series of layers of the form

$$\text{conv2D(FH, FW, N)} \longrightarrow \text{batch-norm} \longrightarrow \text{ReLU}$$

where FH is the filter height, FW is the filter width and N is the number of output filters. What is searched over is the parameters of the convolutional network, *filter width*, *filter height* and *number of filters* individually for each layer, as well as which of the previous *layers to take as input*. The filter height and filter widths were both restricted to be in $\{1, 3, 5, 7\}$, and the number of filters to be in $\{24, 36, 48, 64\}$. For some experiments they also allowed the stride height and width to be predicted, but they got slightly worse results then using a predefined policy. They number of layers is predefined for each search iteration, but is gradually increased as the search progresses. They used a *recurrent neural network*[2], which we call the *controller*, to predict both the parameters of the convolutional layers and the inputs to each layer. A predicted network is called a *child network* of the controller. The idea of *neural architecture search* is illustrated in Figure 1



[2]Thereof the name *neural* architecture search

Figure 1: An overview of Neural Architecture Search. Figure and caption from [2].

The controller is trained with a reinforcement learning algorithm called *policy gradient*. You will later in the course learn the necessary tools to understand this fully, for now we shall try to give the basic intuition only. The idea is that the network predicts the parameters *sequentially*, starting from the first convolutional layer. In this way it can use the network structure so far in deciding on the structure of the next convolutional layer. After the controller has finished predicting all layers in the network, the network is trained on the training set of CIFAR-10 for 50 epochs and then evaluated on the validation set. The controller gets as feedback the *accuracy* of the child network on the validation set. If the child network obtains *low* accuracy the controller is *discouraged* from selecting the chosen hyperparameters again. If the child network performs well on the other hand, the controller is updated so to predict the chosen hyperparameters more often in the future. The architecture search above predicted, trained and evaluated 12800 child networks altogether, spending a total of 22400 GPU-hours[3]. They achieved an accuracy *slightly worse* than the best hand-engineered networks. One of the discovered architectures from the architecture search can be seen in Figure 2.

[3]Using Nvidia Tesla K40s

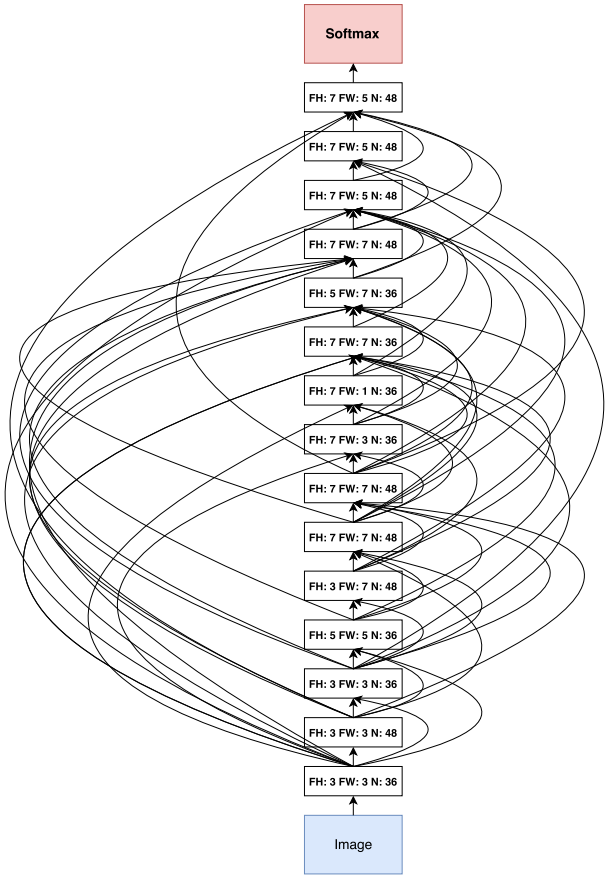


Figure 2: Convolutional architecture discovered by our method, when the search space does not have strides or pooling layers. FH is filter height, FW is filter width and N is number of filters. Note that the skip connections are not residual connections. If one layer has many input layers then all input layers are concatenated in the depth dimension. Figure and caption from [2].

We shall now move over to the second approach for neural architecture search. In [3] they define the search space quite differently. They note that many of the best performing convolutional neural network architectures consist of many repetitions of the same building block, the ResNet architecture [4] being a good example. Instead of optimizing over the whole architecture they instead optimize over an architectural building *cell*[4], which they then

[4]The paper uses the term *cell* to refer to the repeated building blocks, and the term *block* to mean a substructure of a cell!

put together in a hand-engineered way. Actually they are learning *two* different cells, named *normal cell* and *reduction cell* respectively. The *reduction cell* is forced to apply a stride of two to their initial input, thereby reducing the spatial resolution in the network with a factor of two in each direction, otherwise there is no difference to the possible structures they may take. From these two cells one may build a complete newtork to classify images from CIFAR-10 as shown in Figure 3.

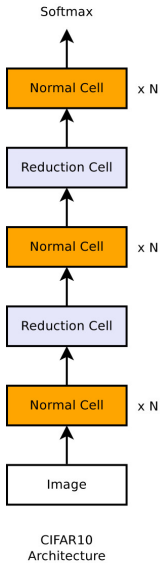


Figure 3: Architecure for CIFAR-10. The *normal* cells are repeated N times. Figure from [3].

In only searching over building cells, they put strong restrictions on the architectures that may be obtained. It might be that the reason structures of repeated cells dominate convolutional network architectures is due to the lack of human imagination rather than the superiority of the design pattern. Anyway, only optimizing over such cells reduces the search space consistently and may allow us to a more thorough search within this limited area. An even more important advantage is that a the learned building cells may later be used in different ways to build many different network architectures. This is useful both to adapt for different input sizes and to create networks with different speed-accuracy tradeoffs. An architecture for higher resolution images from ImageNet may be constructed as illustrated in Figure 4. In this way we can search for an architecture on a smaller dataset of low-resolution images, and hopefully the learned convolutional cells will generalize to higher

resolution images.

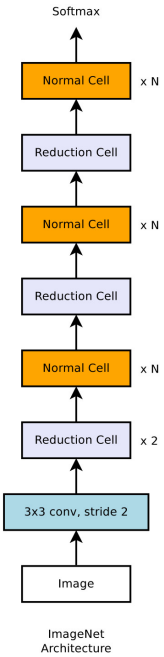


Figure 4: Architecure for ImageNet. Figure from [3].

We shall not go into the details of the search for now[5] but point out that the same search space was used for the two cells, and they were generated as a pair, i.e. not independent of each other. A total of 20000 architectures where sampled[6], spending a total of 2000 GPU-hours[7]. The best performing convolutional cell pair can be seen in Figure 5. What is not shown in the figure is that all convolutions are followed by batch normalization and then the ReLU activation function. This is not learned, but fixed for all convolutional cells.

[5] Can *you* think of a natural sequential generation process that could have resulted in the two cells in Figure 5? You start with only h_i and h_{i-1} in the graph. At each step you add either an edge or a node, what are the rules?

[6] They were trained for 20 epochs instead of 50 as above

[7] Using Nvidia Tesla P100s

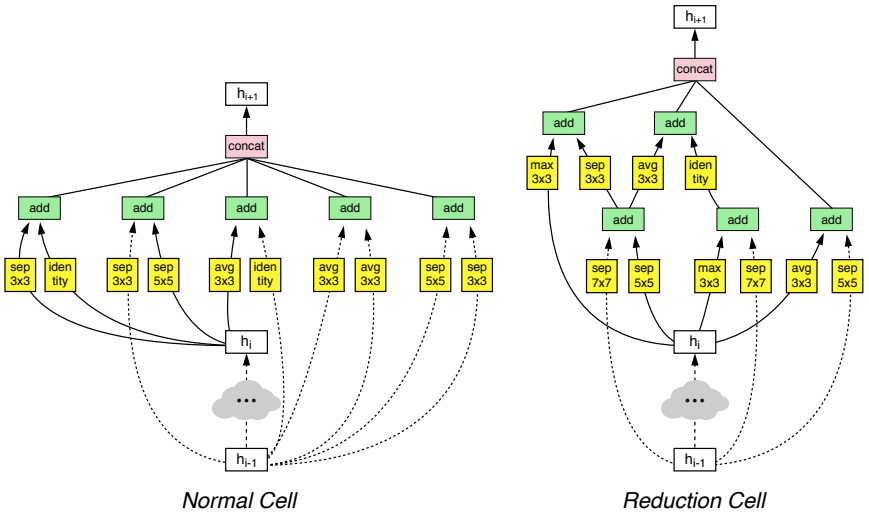


Figure 5: Architecture of the best convolutional cells (NASNet-A) with $B = 5$ blocks identified with CIFAR-10 . The input (white) is the hidden state from previous activations (or input image). The output (pink) is the result of a concatenation operation across all resulting branches. Each convolutional cell is the result of B blocks. A single block is corresponds to two primitive operations (yellow) and a combination operation (green). Figure and caption from [3].

How did the cells perform we build an architecture of the form in Figure 4 and train it on ImageNet instead? This time the architecture search actually led to *better* performance than those hand-engineered by humans so far. By varying the number of repetitions of the learned cells (the N in Figure 4) and number of output filters in the convolution layers, they were able to produce models of different levels of computational complexity which performance exceeded that of previous hand-crafted models at that computational level, see Figure 6. We note that e.g. 6@4032 implies $N = 6$ and that the *penultimate*[8] layer has a feature depth of 4032. (During training on CIFAR-10 they used $N = 2$, but they did not specify how many output filters they used! They use the heuristic of doubling the number of filters after each reduction cell, but the initial number of output filters is not specified. . .).

[8] Feature layer used for classiciation

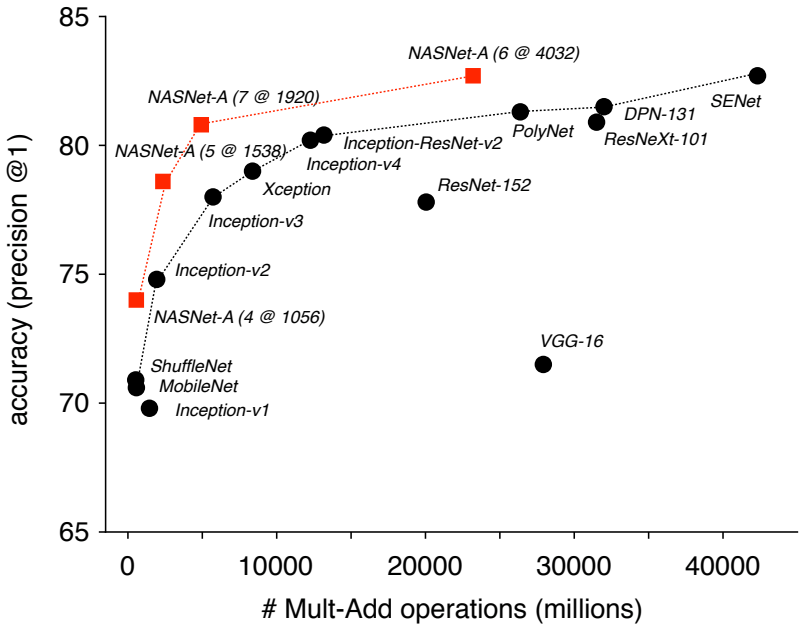


Figure 6: Performance on ILSVRC12 as a function of number of floating-point multiply-add operations needed to process an image. Comparison with published results (black circles). Figure from [3].

If we instead look at performance as a number of parameters, again we get a curve that *envelopes* the results of previous hand-crafted architectures, as can be seen in Figure 7.

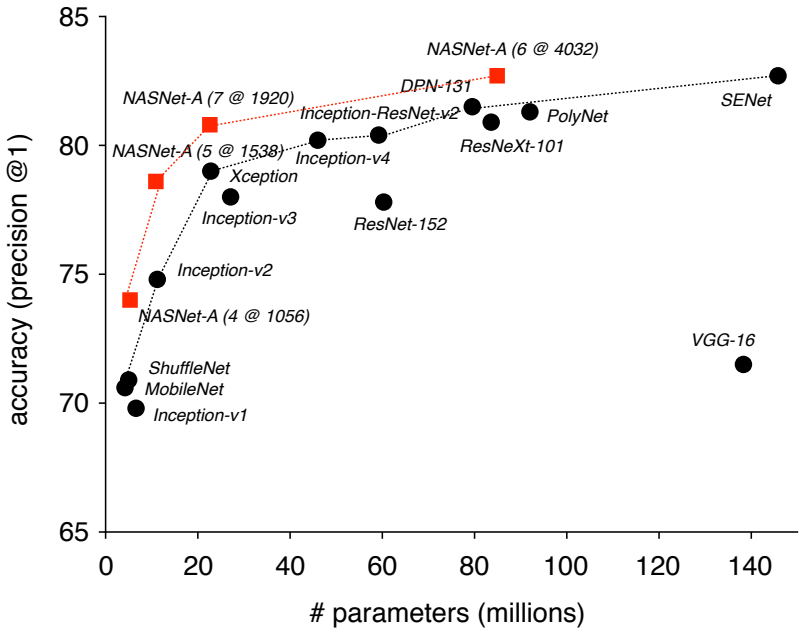


Figure 7: Performance on ILSVRC12 as a function of number of parameters. Comparison with published results (black circles). Figure from [3].

Although a clever learning strategy using recurrent neural networks and reinforcement learning[9] where used in the experiments, they also showed that actually a simple *random search* over the same search space produced good, though slightly worse, results. This shows that designing a good search space may be just as important as using a clever optimization algorithm.

What was the *cost* of the architecture search? With a price of say 1.5 USD per hour for renting Nvidia Tesla P100, and 2000 hours spend, this gives a cost of 3000 USD, or 25000 NOK, in total for GPU-rental. This is in many respects surprisingly cheap given (1) the value of good convolutional architectures and (2) the human effort usually put into architecture design. Still you would perhaps want it to be both faster and cheaper. Recent research on architecture search goes in this direction [5, 6]. As a word of caution, one should watch out for seemingly efficient architecture searches that really narrows down the search space. Perhaps you can think of an efficient architecture search algorithm that matches state-of-the art on ImageNet only using a single sample?

[9] A somewhat more advanced variant of policy gradient was used here.

1 Bibliography

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[3] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2(6), 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *arXiv preprint arXiv:1712.00559*, 2017.

[6] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.