A Survey on One-shot Neural Architecture Search

Yao Xiao¹, Yahui Qiu¹ and Xiaofan Li¹*

¹ School of Digital Arts & Design, Dalian Neusoft University of Information, Liaoning, China

*Corresponding author: lixiaofan626@outlook.com

Abstract. Complex deep neural network architecture such as AlexNet has great success in image classification, natural language processing and other applications. The choice of network architecture has proven to be critical. And researchers begin to design more and more complex networks in order to obtain better performance. However, designing complex networks manually often consumes a lot of computing resources and time, so the automated Neural Architecture Search (NAS) has attracted more and more attention in recent years. Currently, NAS have shown great potential in designing new architectures with high performance and high efficiency. Thus, first of all, this paper briefly introduces the development context and different directions of NAS. But compared to other NAS surveys, this paper focuses on One-shot-NAS methods. Among different methods of NAS, One-shot-NAS can achieve better performance in less time. Through comprehensive comparison and analysis, we apply a novel categorization on One-shot-NAS methods that are based on DAG, Hypernetwork, and network transformation respectively.

Keywords: Deep learning, NAS, One-Shot

1 Introduction

The main work in traditional machine learning is manual feature engineering, such as SIFT[1] and HOG[2] have been widely used successfully. Deep learning frees researchers from feature designing which requires a high level of domain knowledge and ample experiments. Researchers can use features extracted by neural network directly and focus on adjusting the architecture of neural network according to the effect of feature expression. Considerable experiments also prove that features extracted by the neural network have better expressiveness, e.g. AlexNet[3]. But the laborious job of researchers hasn't been lightened, just shifts from manual feature engineering to network architecture engineering. Large-scale and effective networks, such as AlexNet, VGG[4], ResNet[5] etc., still require a large amount of computing resources and repeated trials. Therefore, researchers wish to find an automated way to quickly design a network architecture that works for a certain problem, instead of designing the network architecture manually by experts. Neural Architecture Search (NAS)[6,7] may be a reasonable solution in automating network architecture designing.

Any search method consists of two basic elements at least, namely search space and search strategy. As for the classification of NAS by search space, researchers' opinions are relatively consistent, and can generally be divided into global search[1] and cell-based search[9,10]. But the
situation will be more complex when classifying NAS by search strategy. Someone\cite{8} classifies search strategy into Reinforcement Learning(RL)-Based\cite{1}, Evolutionary Algorithms(EA)-Based\cite{28} and Bayesian Optimization(BO)-Based\cite{29} etc.. Others\cite{11} divide search strategy into search strategy and performance estimation strategy. These classification methods themselves will have some intersections. For example, Wistuba et al.\cite{8} divide NAS into RL-Based, EA-Based, Surrogate Model-Based Optimization(SMBO) and One-shot-NAS according to the optimization methods. But in fact, many One-shot-NAS methods are based on RL. However, Thomas et al.\cite{11} regard One-shot-NAS as an evaluation method, but also One-shot-NAS should be more like a search strategy. For reasons mentioned above, in this paper we just divide NAS into One-shot-NAS and Stand-alone-NAS simply. On the one hand we want to focus on One-shot-NAS and on the other hand there are empirical data showing that One-shot-NAS performs relatively well in all NAS methods(see Table 1).

Based on the above discussion, a closer look at One-shot-NAS looks like a better option. Our survey is divided into several sections. In section 2, we will introduce notations and definitions used in this paper. We will highlight the latest developments in One-shot-NAS in section 3. At the last section, summary of One-shot-NAS future directions will be discussed.

| Method          | Error(%) | Params(Millions) | GPU Days |
|-----------------|----------|------------------|----------|
| RL-NAS          | 3.735    | 129.78           | 3375.75  |
| EA-NAS          | 4.54     | 8.58             | 1101     |
| One-shot-NAS    | **2.64** | 5.8              | 2.46     |
| Random Search   | 3.35     | **3.8**          | **2.33** |
| Human           | 3.15     | 25.42            | -        |

2 Notations and Definitions

Definition 1 (NAS): Basically, NAS is no different from other search methods, except the target of NAS is the architecture or hyper-parameters of the neural network. We can define NAS in formula (1) \cite{8}:

$$\Lambda : D \times A \rightarrow M$$  \hspace{1cm} (1)

where $\Lambda$ denotes a specific deep learning algorithm, $D$ is the dataset, $A$ is the architecture search space and $M$ is the model space. That is, we map from architecture search space to model space through a deep learning algorithm.

The objective function of NAS is defined in equation (2):

$$\Lambda(\alpha,d) = \arg \min_{a_\alpha \in M_\alpha} L(m_{\alpha,\theta},d_{\text{train}}) + R(\theta)$$  \hspace{1cm} (2)

where $L$ is the loss function, $d \in D$, $\alpha \in A$, $m_{\alpha,\theta} \in M_\alpha$, and $R$ is a parameter regularization method. The goal of NAS is to find a architecture $\alpha^*$ that maximizes the objective function $O$, as shown in equation (3):

$$\alpha^* = \arg \max_{\alpha \in A} O(\Lambda(\alpha,d_{\text{train}},d_{\text{valid}}))$$  \hspace{1cm} (3)

Many researchers divide NAS into three parts: search space, search strategy and performance estimation strategy, but as for One-shot-NAS, we believe search strategy and performance estimation strategy are very relevant, therefore we do not discuss them separately.

Definition 2 (Directed Acyclic Graph): The architecture of neural network is usually considered a
directed acyclic graph as shown in equation (4):

$$G = (V, E)$$  \hspace{1cm} (4)$$

where \( V \) denotes the set of nodes, \( E \) denotes the edges between nodes in graph. For any node \( v \in V \) and its operation \( o_i \) (such as, filter, pooling etc.), we have equation (5):

$$x_v = \sum_{(u,v) \in E} o_i(x_u, w_i)$$  \hspace{1cm} (5)$$

That is, the sum of operations of nodes and their previous nodes represents a network architecture.

Definition 3 (Search Space): Search space is a subspace of a network architecture. The operation in the subspace can be limited to a certain range, and other restrictions may be imposed on the architecture.

Definition 4 (One-shot-NAS): In NAS, we train only one architecture during the search process. The most common One-shot-NAS have four steps: (1)Design a search space that contains various operations of the One-shot model. (2)Train a one-shot model to obtain shared parameters. (3)Randomly sample some candidate architectures from the one-shot model. (4)Evaluate these candidate and select one of the best training from scratch. A typical One-shot-NAS process is shown in Figure 1.

![Fig 1. Illustration of One-shot-NAS](image)

3 One-shot Architecture Search

Hyperparameter optimization has always been an important direction in the field of deep learning\[^{[13,14]}\]. But, more and more research shows that structural improvement may be more important for network performance with the development of network such as VGG and ResNet in recent years. The works by Zoph and Le\[^{[9]}\] and Baker et al.\[^{[7]}\] mark the beginning of NAS that has outperformed manually designed architecture on image classification and language model. Since then NAS gets more and more attention. Most of the early works in NAS are Stand-alone-NAS, that is, train an architecture selected in search space from scratch. This process usually takes lots of GPU days. For example, Zoph and Le\[^{[9]}\] use 800 GPUs and 22400 GPU days. The core idea of the One-shot-NAS is to reduce repeated training processes of Stand-alone-NAS by weight sharing. Based on traditional NAS, there are three main research direction: the first is based on DAG, the second is based on Hypernetwork and the third is based on network transformation. The common goal of the three directions above is to reduce the weight calculation time of a large network. Methods with DAG apply pruning strategy, methods with Hypernetwork use a small Hypernetwork to generate weights of a large network, and methods with network transformation train a small network first which will be transformed into a large network later.

3.1 NAS based on DAG

Compared with NAS, ENAS\[^{[9]}\] improves the process of the architecture search from random search to DAG-based search. The magnitude of ENAS search space is \(4^N \times N!\), where \(N = |V|\). Each node can
choose one of 4 activation functions and decide whether to take a skip connection form previous nodes. ENAS search strategy is essentially the same as NAS. First fix the parameters of the RNN controller and train the DAG sampled by the controller. The objective function is as shown in equation (6):

\[
V_o E_o[L(m; w)] \approx \frac{1}{M'} \sum_{i=1}^{M'} V_o L(m_i; w)
\]

(6)

where \(M'\) is the number of DAG sampled by controller. In the train process, ENAS calculates the loss function \(L\) of the model \(m\) only one epoch with the train data set, instead of running multiple epochs like a normal neural network. On the one hand, because the early performance of the network is consistent with the final performance with great probability\(^{(12)}\), it can significantly faster training process. On the other hand, training controller and training model \(m\) will alternate multiple times, the target in this phase isn't to obtain a stable model \(m\), but to train a better controller through multiple train epochs guided by reward. Thus it doesn't make much sense to train the temporary \(m\) in the middle process for more epochs. Finally, use the converged controller to sample multiple \(m\) at one time, and then select the best one based on reward. Only in the process of training \(m\) and controller in the above steps involves back propagation. But thanks to ENAS uses the same DAG, this is no different from traditional neural network training.

DARTS\(^{(15)}\) is similar to ENAS using cell-based search. However, when selecting the sub-network architecture, the path selection is not abstracted into discrete values. In ENAS the size of filter can be decide to be 3*3 or 5*5, but in DARTS the decision is abstracted into a probability calculated from different paths by softmax. The final decision is the path with the highest probability as show in equation (7):

\[
\overline{\sigma}^{(i,j)}(x) = \frac{1}{\sum_{i\in O} \sum_{j\in O} \exp(\sigma_{ij}^{(i,j)})} \sigma_{ij}^{(i,j)} o(x)
\]

(7)

where \(O\) is the set of actions, superscript \((i, j)\) denotes the node and \(\sigma\) is an embedding vector of dimension \(|O|\).

Although the final result of DARTS is not fundamentally different from ENAS. But in ENAS, you need to use reward to update the controller, which can be directly obtained by using back propagation in DARTS where the path selection probability is also trained as a hyperparameter.

Different from training the path selection probability as a hyperparameter in DARTS, SNAS\(^{(16)}\) uses a more complete approach to probabilistic modeling with graph. At the same time, in order to ensure that the strategy of sampling path is differentiable, SNAS uses gumbel-softmax transformation to implement the sampling reparameterization. Gumbel-softmax transformation can transform a discrete distribution into a continuously differentiable softmax form. Moreover, it can control the degree of continuity by modifying the temperature in the gumble-softmax transform.

ProxylessNAS\(^{(17)}\) also notices that the use of softmax for all paths still requires higher computing resources, especially when the network is large. Putting all the weights into memory at once will not work in many cases without a proxy strategy. ProxylessNAS converts path selection into a BinaryGate strategy selection directly. Only the selected path will be calculated each time, which reduces the memory requirement for a single training.

3.2 NAS based on Hypernetwork

SMASH\(^{(18)}\) uses the network architecture as the search space just like other NAS. The weights of the sampled network architecture is not trained on the validation set or the training set, but generated form a small Hypernetwork. The training process is not to optimize the network architecture, but to optimize the Hypernetwork to produce better weights. Only after Hypernetwork training is over, SMASH will randomly generate some architectures, uses weights generated by Hypernetwork for training from scratch, and finally sorts and selects the one with the best performance on the validation.
The reason why many NAS methods are based on DAG is that DAG can maintain the connection between nodes and improve the diversity and effectiveness of search space. Therefore, GHN\textsuperscript{[19]} combines GNN and Hypernetwork to achieve the requirement of retaining node information and reducing computing resources at the same time. And the core equation of GHN is shown in equation (8):

\[
w = \{w_v \mid v \in V\} = \{H(h^{(3)}_v; \phi) \mid v \in V\} = \{H(h; \phi) \mid h \in G_0 \} = \{\{h^{(0)}_v \mid v \in V\}; \phi\} = GHN(A; \phi; \theta)
\]

(8)

where \( H \) represents the mapping of Hypernetwork, \( h^{(3)}_v \) denotes the embedding of node \( v \) obtained after the forward propagation of \( T \) steps GNN, \( \phi \) is the parameters of GNN, \( \theta \) is the parameters of Hypernetwork. That is to say, the above equation indicates that the finally generated network parameters are results obtained by using \( H \) mapping on the embeddings generated by GNN.

In Definition 2, Dong et al.\textsuperscript{[28]} notice that in the common One-shot-NAS, the model architecture will only be evaluated until the process of train is over. Since the search space of NAS is usually very large, in order to reduce the training time, a random sampling strategy is usually used. But the implicit assumption of this strategy is that the search space is evenly distributed. This assumption is usually not the fact. Therefore, based on using Hypernetwork to train shared parameters, Dong et al. add an evaluator before obtaining candidates to learn the distribution of sampling architectures with softmax. At last select the candidate with the highest probability.

### 3.3 NAS based on Network Transformation

The main issue with NAS is training the network from scratch every time, EAS\textsuperscript{[20]} tries to use the existing network to gradually build a target network, thereby reducing training time. EAS takes advantage of Net2Net\textsuperscript{[21]} for sharing weights. In fact, EAS is just looking for an architecture that functions similarly to the known networks. Since a lot of references are made to the existing network architecture and parameters, it will perform better, but it will not discover a completely new structure. The search space of EAS is a set of network transformation operations, such as adding, deleting, widening, etc., instead of searching the network architecture like other NAS methods. EAS and NAS, ENAS are all based on RL\textsuperscript{[22]}. In EAS, state is defined as the current network architecture, and action is defined as the corresponding network transformation operation like widening or deepening the actor network. First, given a custom small network structure \( G \), they utilize a meta-controller to generate actions for network transformation. After \( T \) steps action, the final searched network is obtained. The operation method of network wider is relatively simple and straightforward. First EAS needs to determine whether or not to widen the network. If deciding to widen the network, then EAS increases the width of a network architecture as follows:

\[
G_l(j) = \begin{cases} 
  j & 1 \leq j \leq f_{out}^l \\
  \text{random sample from } \{1, \ldots, f_{out}^l\} & f_{out}^l < j \leq \hat{f}_{out}^l 
\end{cases}
\]

(9)

that is \( f_{out}^l \rightarrow \hat{f}_{out}^l(> f_{out}^l) \) \((k_w^l, k_h^l, f^l, f_{in}^l)\), where \((k_w^l, k_h^l, f_{in}^l, f_{out}^l)\) represents the width and height of the filter and the input and output size of layer \( l \) respectively. The original output is maintained, then some original output values are randomly filled.

In EAS, authors just widen the width of a certain layer in the network or add a few more layers in the vertical direction. There is no learning about the way of linking in each layer (similar to LSTM). Path-level-EAS\textsuperscript{[23]} further adds the learning of the internal structure. The main idea is to replace a single convolutional layer with a multi-branched layer. In fact, this is the same method as EAS, except that the network structural unit is observed at different perspective scales.
Function-preserving transformation which can maintain the dimension of the input and output tensor and sharing weight is used in both of above papers. EAS function-preserving transformation is a modification of the layer itself, such as modifying the number of filters, and correspondingly modifying the size of the pool, and so on. Path-level-EAS is to break up this layer and use smaller units to calculate (similar to different paths in LSTM). In this way, the existing structure is unchanged in the overall structure with the layer as a unit. Thus, weights obtained from previous training can also be used directly. However, when weights are updated, since the layer has a Path-level path, new changes will occur during the update.

4 Future Research Directions and Conclusion
One-shot-NAS extracts network architecture candidates from an over-parameterized network, and this has been verified by some researchers as a reasonable choice[24,25]. But, in many experiments, it is difficult to say whether it is due to the superiority of the network structure or the result of using technologies such as BatchNormalization, Dropout, and Regularization. So, in future research, we should focus on the differences between different structures, rather than based on the recognition accuracy on a certain data set. On the other hand, research on NAS has been in its infancy since 2017. Most of the related work is based on CNN and applied to image classification, and some other areas of work are slowly emerging[26,27]. Researchers should further explore the role of NAS in other fields to see if NAS is a more general approach.

References
[1] David G. Lowe: Object recognition from local scale-invariant features. International Conference on Computer Vision, 1150-1157 (1999).
[2] N. Dalal ; B. Triggs:Histograms of oriented gradients for human detection. Computer Vision and Pattern Recognition (2005).
[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems (2012).
[4] Karen Simonyan and Andrew Zisserman: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).
[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun: Deep residual learning for image recognition. Computer Vision and Pattern Recognition (2016).
[6] Barret Zoph and Quoc V. Le: Neural architecture search with reinforcement learning. International Conference on Learning Representations (2017).
[7] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar: Designing neural network architectures using reinforcement learning. International Conference on Learning Representations (2017).
[8] Martin Wistuba, Ambrish Rawat, Tejaswini Pedapati: A Survey on Neural Architecture Search. arXiv:1905.01392v2 (2019).
[9] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean: Efficient neural architecture search via parameter sharing. In International Conference on Machine Learning (2018).
[10] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu: Practical block-wise neural network architecture generation. Computer Vision and Pattern Recognition, 2423-2432. (2018).
[11] Thomas Elsken, Jan Hendrik Metzen and Frank Hutter: Neural Architecture Search: A Survey. Machine Learning Research(20),1-21, (2019).
[12] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalka: Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. International Conference on Learning Representations (2017).
[13] James Bergstra, Remi Bardenet, Yoshua Bengio, and Balazs Kegl: Algorithms for hyper-parameter optimization. Advances in neural information processing systems (2011).
[14] Shreyas Saxena and Jakob Verbeek: Convolutional neural fabrics. Advances in neural information processing systems (2016).

[15] Hanxiao Liu, Karen Simonyan, and Yiming Yang: DARTS: differentiable architecture search. International Conference on Learning Representations (2019).

[16] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin: SNAS: stochastic neural architecture search. International Conference on Learning Representations (2019).

[17] Han Cai, Ligeng Zhu, and Song Han: ProxylessNAS: Direct neural architecture search on target task and hardware. International Conference on Learning Representations (2019).

[18] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston: SMASH: one-shot model architecture search through hypernetworks. International Conference on Learning Representations (2018).

[19] Chris Zhang, Mengye Ren, and Raquel Urtasun: Graph hypernetworks for neural architecture search. International Conference on Learning Representations (2019).

[20] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang: Efficient architecture search by network transformation. Thirty-Second AAAI Conference on Artificial Intelligence, the 30th innovative Applications of Artificial Intelligence, and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence 2787–2794, (2018).

[21] Tianqi Chen, Ian Goodfellow, Jonathon Shlens: Net2net: Accelerating learning via knowledge transfer. International Conference on Learning Representations (2015).

[22] Ronald J. Williams: Simple statistical gradient-following algorithms for connectionist reinforcement learning. In Machine Learning (1992).

[23] Han Cai, Jiacheng Yang, Weinan Zhang, Song Han, and Yong Yu: Path-level network transformation for efficient architecture search. International Conference on Machine Learning, 677–686 (2018).

[24] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le: Understanding and simplifying one-shot architecture search. International Conference on Machine Learning(80), 550–559 (2018).

[25] Chris Zhang, Mengye Ren, and Raquel Urtasun: Graph hypernetworks for neural architecture search. International Conference on Learning Representations (2019).

[26] Gong, Xinyu, Chang, Shiyu, Jiang, Yifan and Wang, Zhangyang: Autogan: neural architecture search for generative adversarial networks. International Conference on Computer Vision (2019).

[27] Kaixiong Zhou, Qingquan Song, Xiao Huang, Xia Hu: Auto-GNN: Neural Architecture Search of Graph Neural Networks, arXiv:1909.03184 (2019).

[28] Lingxi Xie and Alan L. Yuille. Genetic CNN. International Conference on Computer Vision (2017).

[29] Shahriari B , Swersky K , Wang Z et al: Taking the Human Out of the Loop: A Review of Bayesian Optimization. Proceedings of the Institute of Electrical and Electronics Engineers, 104(1), 148-175, (2016).