Disentangled Neural Architecture Search

Xinyue Zheng  
AI Lab of Lenovo Research  
Beijing, China  
zhengxy7@lenovo.com

Peng Wang  
AI Lab of Lenovo Research  
Beijing, China  
wangpeng31@lenovo.com

Qigang Wang  
AI Lab of Lenovo Research  
Beijing, China  
wangqg1@lenovo.com

Zhongchao Shi  
AI Lab of Lenovo Research  
Beijing, China  
shicz2@lenovo.com

Jianping Fan  
AI Lab of Lenovo Research  
Beijing, China  
jfan1@Lenovo.com

Abstract—Neural architecture search (NAS) has emerged as a hot topic recently which makes artificial intelligence techniques easier to apply and reduces the demand for experts knowledge by generating deep neural network architectures automatically. However, most existing methods for neural architecture search (NAS) heavily rely on underlying black-box controllers to generate potential candidates of network architectures, and suffer from serious problems of lacking interpretability and search efficiency. In this paper, we propose Disentangled Neural Architecture Search (DNAS), which addresses the two issues by adopting disentangled NAS controller and efficient dense-sampling strategy. Specifically, DNAS learns disentangled factors of network architecture by explicitly encouraging the latent factors to be independent. This approach could not only achieve semantic interpretability but also allow us to conveniently identify the promising regions of representations corresponding to high-performance architectures. We further propose a dense-sampling strategy that conducts targeted architecture search within the promising regions to accelerate the searching process. Our DNAS owns several attractive features: 1) it can successfully learn semantic representations of architectures, including operation selection, skip connections, and layer order; 2) it can speed up the process for neural architecture search more than 13× by using dense-sampling and disentangled factors; 3) it can achieve higher accuracy under less computational cost—DNAS achieves state-of-the-art performance of 94.16% on NASBench-101, and 22.7% top-1 test error on ImageNet with 1.6 GPU-days.

I. INTRODUCTION

As the substitute of manual design, neural architecture search (NAS) aims to automatically generate neural architectures. NAS has been widely explored in various areas, such as computer vision [1], [2], language modeling [3], [4] and model compression [5]. However, most existing NAS methods suffer from two main problems. Firstly, different characteristics of neural architectures (such as accuracy, model complexity, and convergence rate) are entangled with each other, making it hard to generate the desired network architectures accurately. Secondly, the adopted black-box optimization scheme lacks of interpretability and search efficiency. Since such black-box optimization procedure is opaque to humans, it is hard for us to understand and interpret which components of neural architectures are strongly related to their performances. Therefore, it will lead to expensive and aimless search in NAS.

To address the above issues, we first explore some related open questions: (a) Whether the latent representations of black boxes can be translated to interpretable representations? (b) Are there any disentangled factors [6] that control the human-understandable architectural concepts? (c) Is there a systematic link between the disentangled factors and the performance of architectures? (d) Do well-performing neural networks own common structural characteristics? If the answer to all the above questions is ‘yes’, then analyzing the distribution of disentangled factors may help us to quickly locate the promising search space to identify good architectures, which may dramatically improve the searching efficiency for the NAS algorithms.

Regarding the representation interpretation, some empirical studies [6], [7] show the convexity of the Gaussian density makes linear operations between representations meaningful [8]. In computer vision, it can disentangle the representation into independent concepts (e.g. position, scale or color). Transferred to NAS scenario, the disentangled representation may be observed as skip connection, layer operation or layer order. In this work, we propose DNAS, a disentangled neural architecture search method, to learn the disentangled factors of neural network architecture to boost its performance. Unlike conventional practice of using black-box optimization in a large search space, our DNAS method can reduce the large search space into smaller but promising regions according to accuracy-related interpretable factors. Moreover, a dense-sampling strategy is developed to conduct a targeted search in the promising regions to identify good neural architectures with better performance efficiently.

DNAS can be incorporated into various NAS methods in which the neural architectures are optimized in a continuous space [9]–[11]. We take semi-supervised neural architecture search (SemiNAS) as an example, since it leverages numerous unlabeled architectures (without evaluation) combined with gradient optimization to show better effectiveness and efficiency. As shown in Fig.1(a), the controller consists of encoder, decoder and predictor. Specifically, adopting the idea.
of $\beta$-VAE [12], 1) the encoder takes architecture sequence as input and learns to disentangle the information into human-understandable semantic concepts (Fig.1(b)), such as single layer operand traversal, layer order, and transform of skip connections; 2) the predictor approximates the performance and FLOPS of an architecture according to the corresponding disentangled embedding, so it can optimize accuracy and FLOPS simultaneously using gradient-based optimization; 3) the decoder converts the optimized continuous representation into a novel discrete architecture; 4) unlike previous methods who obtain neural architectures through random sampling, we establish relation between disentangled semantic factors and its corresponding accuracy, and dense-sampling is performed in the promising regions where better architectures can be generated easily (Cf. Fig.1(c)). In summary, our work makes the following contributions:

- We study the problem of interpretable Neural Architecture Search, and propose a rather efficient framework called DNAS.
- DNAS learns semantic representations of neural architectures, including operation selection, skip connections, and layer order, which have direct relevance with architecture performances.
- Based on the semantically disentangled representations, we also develop a dense-sampling strategy to accelerate the searching procedure of DNAS by conducting targeted search in promising regions.
- Extensive experiments show that DNAS not only successfully learns semantic relations between network architectures and their performances, but also achieves $13 \times$ speedup than baseline methods.

II. RELATED WORK

A. Interpretabilitly of Neural Network

The drawback of training the NAS black-box models is lack of interpretability. To make latent space of neural network interpretable, some researches [13]–[15] design specific loss function and network for mapping semantic concepts on the layers or paths of network to obtain an interpretable network. Other methods [7], [16] adopt disentanglement of hidden representation to encourage learning interpretability of neural network. $\beta$-VAE [12] enforces the latent code to obey the standard Gaussian distribution, then attributes the semantic concepts to latent code by increasing the information capacity of the latent code. In computer vision, a face image can be disentangled into some semantic representations, e.g., posture and color. Transferring to NAS field, we hope to obtain disentangled factors that are meaningful for neural network design. A very recent work [17] tries to achieve interpretable NAS by combining Bayesian optimization and graph kernel. However, it conducts architecture search in explicit architecture space which is less expressive and flexible than searching in disentangled hidden space as done in DNAS.

B. Neural Architecture Search

Recent NAS methods, mainly based on evolutionary algorithms [18], [19], reinforcement learning [1], [4], [20] and differentiable optimization [3], [9] usually design a large and expressive search space for maximal performance, which will result in very expensive training and search processes. To improve NAS efficiency, many studies propose targeted solutions. ENAS [4] adopts weight sharing among different searched models which alleviates the burden of training every model from scratch. Single-Path NAS [21] reduces the number of trainable parameters by employing a single-path over-parameterized convolution network, which can express all the network structure with shared convolutional kernel parameters. NAO [9] maps discrete space into continuous space to perform efficient optimization. Based on NAO framework, SemiNAS [11] trains the controller in a semi-supervised manner, where a large number of unlabeled architectures (without evaluation) can be used to train the controller. Taking search efficiency as
an important optimization goal, we adopt the semi-supervised setting like SemiNAS. However, instead of randomly generating unlabeled architectures, our method is oriented to generate potentially excellent architectures to reduce search time.

III. APPROACH

A. Framework

Continuous representation is able to represent the architectural topology in a more compact and efficient way, so we adopt the controller structure similar to SemiNAS [11], which conducts architecture search in the continuous space. As shown in Fig. 1(a), the controller consists of three modules: encoder, decoder and predictor. The encoder \( f_e \) maps an architecture sequence \( x \) into hidden representation \( z_x = f_e(x) \) through a single-layer LSTM. To make \( z_x \) interpretable, we train the encoder and decoder via unsupervised disentangled representation learning [6] which adopts an adjustable hyperparameter \( \beta \) to encourage disentangling of latent representation \( z_x \). The decoder \( f_d \) is a multi-layer LSTM, responsible for decoding latent variable \( z_x \) and reconstructing the architecture string \( x' \), which can be formalized as \( x' = f_d(z_x) \).

Taking model complexity into consideration [22], [23], our predictor is built to predict both accuracy and FLOPS. Feeding \( z_x \) into the MLP predictor, we can obtain predictive accuracy \( f_{acc}(z_x) \) and FLOPS \( f_{flops}(z_x) \) of the input architecture. In this way, our DNAs can simultaneously optimize the accuracy and FLOPS by gradient optimization to train the controller. Our DNAs can be trained in a semi-supervised setting like SemiNAS. However, instead of randomly generating unlabeled architectures, our method is oriented to generate potentially excellent architectures to reduce search time.

B. Disentangled Representation Learning

To achieve controllable powerful architecture generation, semantically interpretable and disentangled factors of variation are indispensable. We borrow the idea of \( \beta \)-VAE 2017beta-vae to learn a disentangled latent space:

\[
L(\theta, \phi, x) = L_{rec} + \beta L_{KL} = E_{q_\theta(z|x)}[\log p_\phi(x|z)] - \beta D_{KL}(q_\phi(z|x) || p(z)),
\]

where the \( \theta \) and \( \phi \) is to parameterize the distribution of the encoder and the decoder. The first term aims to learn the marginal likelihood of the data in generative procedure, and the second term is Kullback-Leibler (KL) divergence between the Gaussian prior \( p(z) = \mathcal{N}(0, 1) \) and the learned approximate posterior \( q_\phi(z|x) \). This constraint encourages disentangling property in the inferred \( q_\phi(z|x) \) due to the isotropic nature of the Gaussian prior \( p(z) \). The adjustable hyperparameter \( \beta \) balances reconstruction cost and latent channel capacity. An appropriately tuned \( \beta \) will impose restrictions on the capacity of latent information channel, and push the model to learn more statistically independent latent factors of the architecture. Previous work [12] demonstrates that \( \beta \)-VAE with \( \beta > 1 \) can learn the disentangled representation of input data, if the data contains some underlying factors of variation that are independent.

In our perspective, each well-disentangled factor can independently control single representation of an architecture [24], such as the operation of a certain layer (Fig.1(b)), skip connection (Fig.2) and layer order (Table V).

C. Efficient Dense-Sampling Strategy

As described above, the encoder maps the discrete architecture sequence \( x \) into continuous and interpretable representation \( z_x = f_e(x) \), then employs the predictor to predict its corresponding accuracy \( \hat{y}_{acc}^x = f_{acc}(z_x) \) and FLOPS \( \hat{y}_{flops}^x = f_{flops}(z_x) \). The training of the predictor is to minimize the prediction loss of both accuracy and FLOPS:

\[
L_{acc} = \sum_{x \in X} (\hat{y}_{acc}^x - f_{acc}(z_x))^2, \\
L_{flops} = \sum_{x \in X} (\hat{y}_{flops}^x - f_{flops}(z_x))^2,
\]

where \( X \) indicates all the candidate architecture sequences that have evaluated performances \( y_{acc}^x \) and FLOPS information \( y_{flops}^x \). The combination of accurate performance prediction and disentangled semantic factors helps us understand which concept affects the composition of high-accuracy network structures intuitively. Further, the obtained relation between accuracy and disentangled factors allows us to conduct a targeted dense-sampling in the hidden space where better architectures are easier to be generated (Fig.1(c)). Meanwhile, the FLOPS predictor uses gradient descent to optimize FLOPS, ensuring that the generated architectures meet the constraints of computing resources.

We now introduce how to perform dense-sampling based on disentangled factors:

1) Randomly generate \( M \) architectures \( x_1, x_2, ..., x_M \) from the search space. Evaluating the performances of these architectures on validation set and calculating their FLOPS, and we can obtain the dataset \( D' = \{ (x_i, y_{acc}^x, y_{flops}^x), i = 1, 2, ..., M \} \) to train the controller.

2) Use the trained encoder \( f_e \) to map \( x_1, x_2, ..., x_M \) into latent disentangled space \( z_{1,2,..,M} \). Suppose each latent code \( z_i \) contains \( S \) dimensions, for the \( s^{th} \) dimension, we choose the \( \pm \sigma \) region near the \( z \) values of the architectures with top-k accuracy as the promising region for dense-sampling:

\[
R_s = \bigcup_{i \in I_k} [z_i - \sigma, z_i + \sigma],
\]

where \( I_k \) is the set of top-k architectures.

3) Sample from promising region \( \mathbb{R}^{R_1, R_2, ..., R_S} \) with a probability of \( \epsilon_1 \). To avoid missing good architecture, we sample
from the entire search space with a probability of $\epsilon_2$, where $\epsilon_2 = 1 - \epsilon_1$.

D. Implementation of DNAS

In this section, we will introduce how to use the disentangled representation and dense-sampling to discover more expressive architectures. In general, the DNAS takes architecture-accuracy-FLOPS pairs as the training data to jointly train the encoder $f_e$, predictor $f_{acc}$ and $f_{flops}$, and decoder $f_d$ by minimizing the following loss function:

$$ L_{total} = L_{rec} + \beta L_{kl} + \mu L_{acc} + \lambda L_{flops}, \quad (5) $$

where the $L_{acc}$ and $L_{flops}$ are the prediction losses introduced in Eqn (2) and Eqn (3). $L_{rec}$ and $\beta L_{kl}$ are the architecture reconstruction loss and KL loss described in Eqn (1). The hyper-parameters $\beta$, $\mu$, and $\lambda$ are used to trade off between these losses.

Our goal is to maximize the accuracy of the neural network architecture under any resource constraints, which can be formulated as:

$$ \max_{z_x} f_{acc}(z_x), \text{ s.t., } f_{flops}(z_x) \leq F. \quad (6) $$

To solve the above constrained optimization problem, we rewrite Eqn 6 to Lagrangian form:

$$ \min_{z_x} L_z(z_x) = -f_{acc}(z_x) + \gamma \cdot f_{flops}(z_x), \quad (7) $$

where the Lagrangian multiplier $\gamma$ is a hyperparameter. Then we optimize $L_z(z_x)$ by performing gradient optimization over $z_x$. The update rule is:

$$ z'_x = z_x + \eta \frac{\partial f_{acc}(z_x)}{\partial z_x} - \eta \gamma \frac{\partial f_{flops}(z_x)}{\partial z_x}, \quad (8) $$

where $\eta$ is the step size. In this way, the resultant new representation $z'_x$ will get higher prediction accuracy $f_{acc}(z'_x)$ and lower FLOPS $f_{flops}(z'_x)$ compared with $z_x$. Then we feed $z'_x$ into decoder to decode a better architecture in the sense of higher performance and efficiency. The detailed algorithm of DNAS is shown in Algorithm 1. First we train the controller to learn semantic disentangled factors. (line 5). Then we analyze the disentangled factors to locate the promising region where dense-sampling is performed with a certain probability (line 6-8). Finally, we use gradient optimization to generate better architectures that satisfy resource constraints (line 9-13).

IV. EXPERIMENTS

Considering a large number of candidate architectures that we want to explore, we first study the impact of modifying the disentangled factor $z_x$ on the semantic architecture on the NASBench-101 [25]. Then, we conduct further empirical research on both NASBench-101 and Imagenet [26] datasets to evaluate the effectiveness and efficiency of DNAS method.

A. Disentangling Architecture Representations

1) Dataset and Settings: NASBench-101 is a new tabular benchmark for neural architecture search which designs a compact and expressive search space for CIFAR-10. It includes 423,624 unique architectures, and each network architecture is mapped to its training and evaluation metrics. For disentangling experiments on NASBench-101, we use single-layer LSTM with a hidden size of 26 for both encoder and decoder, and the predictor is a 3-layer MLP. The trade-off in Eqn (5) is set as $\beta = 2.5$, $\mu = 4$, and $\lambda = 1.5$. For semi-supervised learning, we follow the [11] setting, where $N = 10000$ and $M = 100$. Finally, we run the controller for $L = 3$ iterations with an initial learning rate of 0.001.

2) Results: The controller of DNAS learns to reconstruct architectures from interpretable factors $z$, so we modify the value of $z$ to observe semantic architecture modification. The experimental results show that DNAS can successfully
disentangle the human-understandable representation of the architecture, including operation conversion (Fig.1(b)), the number of layers (Fig.2) and the layer order (Table V). The same disentangled factor controls the same semantic concept in different network architectures. For example, in Fig.2, we traverse the 7th latent factor z_7 from −0.4 to 0.4 for any given architecture, while keeping the remaining latent factor fixed. The disentangling is obvious: only the skip connections of the architecture are changed. We observe that these skip connections reduce the number of effective layers of the architecture, which leads to performance degradation, as shown in Fig.3(c).

We further investigate the relation between more disentangled factors z and architecture performances. Fig.3 demonstrates the strong relation between disentangled factor and its corresponding accuracy, from which we can infer that it will ultimately improve the performance of discovered architectures combined with dense-sampling strategy. The disentangled factors related to accuracy and FLOPS on ImageNet are shown in Fig.4.

B. Dense-Sampling for Fast NAS

1) NASBench-101: In this section, we follow the Eqn (4) to determine the promising regions of dense-sampling. We set \( \sigma = 0.05 \) and take the z-value regions of the top-3 architectures for dense-sampling. We initialize the dense-sampling probability as 5% and increase it to 25% gradually, because the promising region will become more accurate with training time increases. Since the search space of NASBench-101 is relatively simple, there is no obvious difference in the composition of different architectures in FLOPS. The resource restrictions are not considered here.

In Table I, "Queries" indicates how many architecture-accuracy pairs have been queried from NASBench-101, which is equivalent to evaluating the architectures. Therefore, reducing the number of queries can be considered as reducing computational cost. With the help of dense-sampling, DNAS discovers top 0.002% architecture with 94.16% best performance in NASBench-101 using the same number of queries as SemiNAS, and outperforms NAO and regularized evolution using less than 1/13 computational cost (150 queries vs. 2000 queries). The experimental results indicate the potential of using dense-sampling for speeding up the search process and improving the performance of neural architecture search.

2) ImageNet: ImageNet includes 1.28M training images and 5K test images, which are categorized into 1,000 classes. In experiments on this dataset, the controller is a single layer LSTM with 32 hidden size, the accuracy predictor is a 3-layer MLP and FLOPS predictor is a 2-layer MLP.

Taking computational resources under consideration, the searched architecture \( x \) needs to meet the restriction of \( FLOPS(x) \leq F \), otherwise, it will postpone to the sub-optimal architecture until a satisfactory architecture is found. According to expert priors, well-performing neural networks generally have deeper layers and higher FLOPS [32], [33], so we perform dense-sampling in both promising region and FLOPS-constrained edge region on ImageNet. The search process runs on 4 P100 GPUs within half a day. For the final evaluation, we train the searched network architecture for 300 epochs.

Table II shows the performance of our DNAS method across 3 levels of resource constraints (500M \( \leq FLOPS < 550M \), 550M \( \leq FLOPS < 600M \), 600M \( \leq FLOPS \)). Compared to other baseline methods, DNAS achieves higher accuracy under the same computational constraints. Specifically, it outperforms the previous state-of-the-art method SemiNAS algorithm under any experimental setting, indicating that the disentangling and dense-sampling methods work effectively in neural architecture search. Our DNAS completes search process in 1.6 GPU-days, which is approximately 26× faster than DARTS, 5× faster than ProxylessNAS, and 2.4× faster.

\[
\begin{array}{|c|c|c|c|}
\hline
\text{Method} & \text{Search cost (Queries)} & \text{Acc.} & \text{SD} \\
\hline
\text{Random Search} & 2000 & 93.64 & 0.25 \\
\text{RE [19]} & 2000 & 93.96 & 0.05 \\
\text{NAO [9]} & 2000 & 93.87 & 0.04 \\
\text{SemiNAS [11]} & 2000 & 94.02 & 0.05 \\
\text{SemiNAS} & 300 & 93.89 & 0.06 \\
\text{DNAS} & 300 & \textbf{94.16} & 0.05 \\
\text{DNAS} & 150 & 94.01 & 0.06 \\
\hline
\end{array}
\]
Fig. 4. Correlation between disentangled factors, architecture performance and FLOPS on ImageNet. For $z_4$, (a) and (b) show there is no obvious relation between FLOPS and accuracy, so the idea sample region is within red frame in (e) where can reach the highest performance with lower FLOPS (darker color presents higher accuracy). However, in (c), (d) and (f), accuracy and FLOPS have a positive relation, so the sample area may locate slightly to the upper left to compromise accuracy and FLOPS.

| Method               | Top-1 (%) | Top-5 (%) | Params(M) | FLOPS(M) | GPU-days |
|----------------------|-----------|-----------|-----------|----------|----------|
| SemiNAS* [11]        | 24.0      | 7.1       | 6.11      | 611      | 2.5      |
| DNAS                 | 23.4      | 6.7       | 6.20      | 628      | 1.6      |
| DARTS [3]            | 26.9      | 9.0       | 4.90      | 595      | 41.7     |
| PNAS [27]            | 25.8      | 8.1       | 5.10      | 588      | 225      |
| PC-DARTS [28]        | 24.2      | 7.3       | 5.30      | 597      | 3.8      |
| SemiNAS* [11]        | 24.6      | 7.4       | 4.80      | 580      | 2.5      |
| DNAS                 | 24.0      | 6.9       | 6.08      | 569      | 1.6      |
| SNAS [29]            | 27.3      | 9.2       | 4.30      | 522      | 1.5      |
| ProxylessNAS [30]    | 24.9      | 7.5       | 7.10      | 465      | 8.3      |
| Single Path One-shot [31] | 25.3 | -         | -         | 328      | -        |
| SemiNAS* [11]        | 24.9      | 7.4       | 4.55      | 504      | 2.5      |
| DNAS                 | 24.5      | 7.2       | 5.05      | 518      | 1.6      |

than PC-DARTS. When the FLOPS limit increases to 800M, we run SemiNAS for 3 times, the discovered models are still distributed around 600M with no significant improvement in accuracy. In contrast, our DNAS method can find qualified model with 22.7% high accuracy (Fig.5). In practical engineering, DNAS will be more competent for given tasks.

C. Study of DNAS

We first conduct an ablation study to investigate the effects of disentanglement and dense-sampling. DNAS queries 200 architectures from NASBench-101. Table III shows that 1): dense-sampling strategy significantly improves the performance under any settings; 2): the disentangled factors alone may not improve NAS performance evidently, since the searched architectures are still based on randomly sampled architectures, which does not take advantage of disentanglement; 3): but disentanglement can benefit dense-sampling by providing strong guidance on promising regions. Compared to the full model, the DNAS variant with disentanglement removed results in significant performance degradation. The t-test experiment verifies the performance improved by DNAS are statistically significant ($p$-value $<<0.001$).

We further conduct in-depth analysis towards the effectiveness of disentanglement. We range $\beta$ from 1 to 4, and run experiments for 50 times to plot Fig. 6. The picture shows that both the accuracy and the number of accuracy-related disentangled factors decrease when $\beta$ is too low or too high. Well-disentangled representations emerge when the right balance is found between information preservation and latent
channel capacity. Due to that, we can easily identify which factors are more relevant to the accuracy and further perform dense-sampling effectively, which is usually extremely difficult for entangled factors.

We study the latent variables and analysis the quality and quantity of disentangled factors under different $\beta$ value (Table IV). Although not all the latent variables are interpretable, the observed factors including skip-connection, number of layers, and the operation of a certain layer have been able to compose all the representations of the architecture. Analogous to the computer vision field, the observed disentangled factors are usually color, size, or position on 3D objects dataset [12].

### D. More Experiments on ImageNet

We have shown the semantic representations of operation selection and skip connection on NASBench-101 above. In this section, we will demonstrate the disentangled representation on ImageNet. There are 7 candidate operations in the search space, and we use the following numbers to represent architectures in Table V: 1. MBConv ($k=3, r=3$); 2. MBConv ($k=3, r=6$); 3. MBConv ($k=5, r=3$); 4. MBConv ($k=5, r=6$); 5. MBConv ($k=7, r=3$); 6. MBConv ($k=7, r=6$); 7. Zero-out layer, where the MBConv refers to the mobile inverted bottleneck convolution, $k$ is the kernel size and $r$ is the expansion ratio.

### V. Conclusion

A novel DNAS method is developed in this paper to improve both the interpretability and the efficiency of neural architecture search. Our DNAS method can learn the disentangled architecture representations, and dense-sampling is performed accurately within the promising regions to generate better network architectures with higher performance by utilizing...
the accuracy-related disentangled factors. The extensive experiments have demonstrated that the DNAS can consistently achieve impressive performance and searching efficiency. On the NASBench-101 dataset, our DNAS can achieve state-of-the-art performance of 94.16%. On ImageNet dataset, our DNAS can discover the competitive architecture that achieves 22.7% test error.

REFERENCES

[1] B. Zoph, V. Vasudevan, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8697–8710, 2018.
[2] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, “Hierarchical representations for efficient architecture search,” International Conference on Learning Representations, 2018.
[3] H. Liu, K. Simonyan, and Y. Yang, “Darts: Differentiable architecture search,” International Conference on Learning Representations, 2019.
[4] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, “Efficient neural architecture search via parameters sharing,” International Conference on Machine Learning, pp. 4095–4104, 2018.
[5] Y. He, J. Lin, L.-J. Li, and S. Han, “Ampc: Autml for model compression and acceleration on mobile devices,” Proceedings of the European Conference on Computer Vision (ECCV), pp. 784–800, 2018.
[6] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, “Understanding disentangling in beta-vae,” arXiv preprint arXiv:1804.03599, 2018.
[7] L. Ma, Q. Sun, S. Georgoulis, L. Van Gool, B. Schiele, and M. Fritz, “Disentangled person image generation,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 99–108, 2018.
[8] P. Esser, R. Rombach, and B. Ommer, “A disentangling invertible interpretation network for explaining latent representations,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9223–9232, 2020.
[9] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, “Neural architecture optimization,” Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 7827–7838, 2018.
[10] X. Dai, A. Wan, P. Zhang, B. Wu, Z. He, Z. Wei, K. Chen, Y. Tian, M. Yu, P. Vajda, and J. E. Gonzalez, “Fbnetv3: Joint architecture-recipe search using neural acquisition function,”arXiv:1907.05737, 2019.
[11] R. Luo, X. Tan, R. Wang, T. Qin, E. Chen, and T.-Y. Liu, “Semi-supervised neural architecture search,” Advances in Neural Information Processing Systems, vol. 33, 2020.
[12] L. M. Irina Higgins and A. Pal, “vae: Learning basic visual concepts with a constrained variational framework,” International Conference on Learning Representations, ICLR 2017, 2017.
[13] Z. Guo, X. Zhang, H. Mu, Z. Liu, Y. Wei, and J. Sun, “Single path one-shot neural architecture search with uniform sampling,” Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1294–1303, 2019.
[14] S. Xie, H. Zheng, C. Liu, and L. Lin, “Snas: stochastic neural architecture search,” arXiv preprint arXiv:1812.09926, 2018.
[15] H. Cai, L. Zhu, and S. Han, “Proxylessnas: Direct neural architecture search on target task and hardware,” International Conference on Learning Representations, 2018.
[16] B. Cheung, J. A. Livezeys, A. K. Bansal, and B. A. Olshausen, “Discovering hidden factors of variation in deep networks,” arXiv preprint arXiv:1412.6583, 2014.
[17] B. Ru, X. Fan, X. Dong, and M. Osborne, “Interpretable neural architecture search via bayesian optimisation with weissfeiler-lehman kernels,” ICLR 2021: The Ninth International Conference on Learning Representations, 2021.
[18] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Smets, J. Tan, Q. V. Le, and A. Kurakin, “Large-scale evolution of image classifiers,” International Conference on Machine Learning, pp. 2902–2911, 2017.
[19] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2018.
[20] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” arXiv preprint arXiv:1611.01578, 2016.
[21] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, “Mnasnet: Platform-aware neural architecture search for mobile,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2820–2828, 2019.
[22] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” International Conference on Machine Learning, pp. 6105–6114, 2019.
[23] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 8, pp. 1798–1828, 2013.
[24] C. Yang, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hut- ter, “Nas-bench-101: Towards reproducible neural architecture search,” International Conference on Machine Learning, pp. 7105–7114, 2019.
[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., “Imagenet large scale visual recognition challenge,” International journal of computer vision, vol. 115, no. 3, pp. 211–252, 2015.
[26] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, “Mnasnet: Platform-aware neural architecture search for mobile,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2820–2828, 2019.
[27] X. Chen, L. Xie, J. Wu, and Q. Tian, “Progressive differentiable architecture search: Bridging the depth gap between search and evaluation,” Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1294–1303, 2019.
[28] Z. Guo, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, “Pc-darts: Partial channel connections for memory-efficient architecture search,” arXiv preprint arXiv:1907.05737, 2019.
[29] S. Xie, H. Zheng, C. Liu, and L. Lin, “Snas: stochastic neural architecture search,” arXiv preprint arXiv:1812.09926, 2018.
[30] H. Cai, L. Zhu, and S. Han, “Proxylessnas: Direct neural architecture search on target task and hardware,” International Conference on Learning Representations, 2018.
[31] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, “SimpNet: Single path one-shot neural architecture search with uniform sampling,” European Conference on Computer Vision, pp. 544–560, 2020.
[32] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, “On the expressive power of deep neural networks,” European Conference on Computer Vision, pp. 544–560, 2020.