EVOLVING SPIKING CIRCUIT MOTIFS USING WEIGHT AGNOSTIC NEURAL NETWORKS

ABRAR ANWAR*, CRAIG M. VINEYARD†, WILLIAM M. SEVERA‡, SRIDEEP MUSUVATHY§, AND SUMA CARDWELL¶

Abstract. Neural networks have increasingly been applied as state-of-the-art solutions to tasks ranging from image and video analysis, to natural language processing, to strategic planning and control. These investigations have yielded many different neural network architectures as various optimizations are pursued with the objectives of improved performance as well as to improve computational costs. Furthering this exploration, neural architecture search (NAS) has emerged as an algorithmic method of developing neural network architectures. Weight Agnostic Neural Network (WANN) is an evolutionary-based NAS approach. Fundamentally, WANN pursues circuit motifs which enable decent performance on tasks largely due to the network structures that are relatively insensitive to weights and typically much smaller than an equivalent performance dense network. Here we extend the WANN framework to search for spiking circuits, and in doing so investigate whether spiking circuit motifs can also yield task performance that is weight agnostic. In doing so, we analyze properties such as the the complexity of the solution and performance. Our results successfully show the performance of spiking WANNs on several exemplar tasks.

1. Introduction. Neural networks are becoming exceedingly commonplace; however, limitations of traditional hardware which neural networks run on are becoming apparent, specifically in the low-power domain. For edge computing applications, such as drones, satellites, and micro-robots, running larger neural networks is not feasible due to the energy cost. Neuromorphic computing introduces a new paradigm for computing that is brain inspired with an added benefit of low energy usage.

In many cases, neural networks tend to be overparameterized. Recently, a shift towards pruning deep neural networks to make them sparser has become common. In addition, NAS has also been effective in finding architectures that reduce complexity and increase performance of neural networks [7, 22, 13, 5]. For spiking neural networks, Evolutionary Optimization for Neuromorphic Systems (EONS) [17] is such an approach to generate spiking neural networks. Recent work in searching for sparse topologies for various tasks in classical neural networks showed that neural network weight training can be skipped, as a universal parameter sharing approach is effective in evaluating the success of a potential network topology. We use this approach to find topologies in spiking neural networks for solving MNIST, swingup cartpole, bipedal walker, and Atari Atlantis problems. This work provides evidence that spiking networks benefit from weight agnostic graph structures in the same way scalar-weight networks do.

In Section 2, we provide a short background on neuromorphic computing, spiking networks, neural architecture methods, and Weight Agnostic Neural Networks. In Section 3, we define the spiking WANN, followed by the results on various tasks in Section 4. Lastly, Section 5 discusses considerations for future applications and work on spiking WANNs.

2. Background.

2.1. Neuromorphic Computing.

Neuromorphic computing relies on event-based spiking communication between neurons. Conversely, a typical artificial neural network (ANN) relies on dense communication of continuous values. In order for ANNs to work with this new paradigm, they must be converted into spiking neural networks (SNNs). The

*University of Texas at Austin, abraranwar@utexas.edu
†Sandia National Laboratories, cmviney@sandia.gov
‡Sandia National Laboratories, wmsever@sandia.gov
§Sandia National Laboratories, smusuva@sandia.gov
¶Sandia National Laboratories, sgcardw@sandia.gov

main motivation motivation for this difference is the promise of energy-efficient compute evidenced by biological systems. Hence SNNs try to replicate this by communicating in a fashion loosely inspired by biological neurons. We can define a spiking neuron computation by a threshold activation function. Although a binary threshold ANN is not strictly an SNN as it does not include the temporal domain, since it is compatible with neuromorphic hardware, it is referred to as such. Severa et al. [18] noted that converting ANNs to SNNs for neuromorphic computing is a non-trivial process, thus they iteratively sharpen various activation functions to be binary. We show that evolved weight agnostic neural networks with binary activation functions perform well and should be suitable to be transferred onto neuromorphic hardware.

2.2. Lottery Ticket Hypothesis and Network Pruning. Network pruning focuses on removing connections to create sparse networks that have a smaller number of connections and weights. Pruning typically requires prior training, and then reducing the number of weights [2]. The lottery ticket hypothesis solves the difficult problem of training sparse networks. It states that a randomly initialized neural network has a sparse subnetwork that performs just as well, if not better than its dense counterpart [8]. Building on this finding, it was discovered that these pruned networks perform better than chance with randomly initialized weights [21], further supporting the idea that the network topology influences performance.

2.3. Neural Architecture Search. In contrast to pruning methods, the goal of neural architecture search (NAS) is to learn a network topology that can achieve good performance on certain tasks, while sometimes ensuring a lower number of parameters. Zoph and Le's [22] pioneering work in NAS showed the intense computational resources needed to generate accurate neural networks due to the large search spaces. Most NAS approaches are split into three separate components: the search space, the search algorithm, and the evaluation strategy [7]. The search space consists of a set of operations such as convolutions or pooling layers and how these operators can be appended to form network topologies. The search algorithm is how NAS methods selects candidates from a population of network architectures and how they optimize these candidates. The evaluation strategy is where the performance of the models are evaluated, either by actually running the network or using some other metrics to estimate performance.

Typically, during the evaluation stage, if the algorithm needs to evaluate a test set, the network to first undergo training. This makes the evaluation strategy the most expensive part of the operation. Parameter sharing is one approach used to gain a speed up [15], where child models share parameters with their parent. Brock et al. [3] uses a HyperNet [10] to generate weights based on the encoding of the network architecture parameters. Weight agnostic neural networks (WANNs) [9] take an approach inspired by evolutionary methods to evolve neural network topologies, focusing on individual nodes rather than a set of operations.

2.4. Weight Agnostic Neural Networks. Weight agnostic neural networks are inspired by the fact precocial species can accomplish several tasks at birth, such as duck hatchlings being able to swim and eat [20]. WANNs follow an iterative topology search algorithm inspired by the NEAT evolutionary search method [19]. In most architecture search approaches, each generated topology requires individual weight training, which tends to be the most expensive portion of the algorithm. WANNs show that the network topology is important by enforcing weight-sharing across the whole network. Rather evaluating a network by its performance on a test set after training, WANNs evaluate on a set of shared weight values.

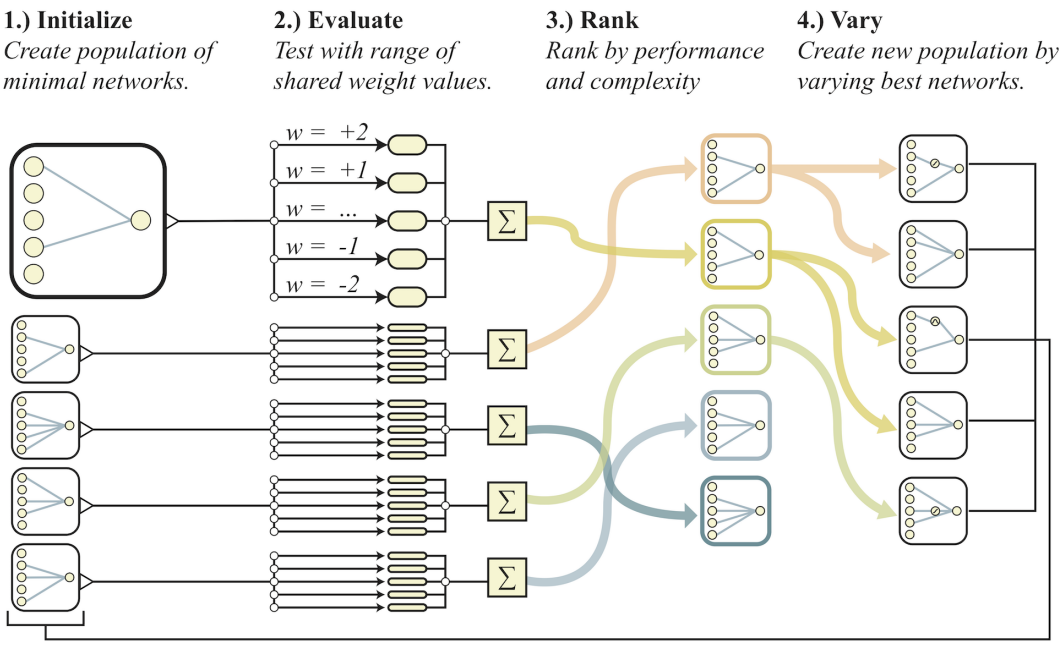

Fig. 2.1: Visualization of topology search used to evolve WANNs. Sourced from WANN paper [9]

To detail the approach in Figure 2.1, the algorithm goes as follows:

1. A population of various network topologies are generated.
2. For reinforcement learning/control tasks, the network runs through several rollouts, each using a different shared weight value. For classification, it simply evaluates the training set using the various shared weight values.
3. The networks are ranked in regards to their average performance and the number of connections as a loose estimate of model complexity.
4. The top networks reproduce by adding or mutating connections and activation functions.

The ranking process uses the crowding distance metric from Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [6], an evolutionary approach to multi-objective optimization. Two objectives are optimized over: the mean fitness across each of the iterations and an alternating objective of max fitness and the number of connections. The number of connections is minimized 80% of the time while the max fitness is maximized 20% of the time. This is to ensure that the network is able to grow in complexity if it leads to increased performance. The best performing network is chosen as the final network; however, there does exist a Pareto frontier of individual networks between network complexity and performance.

The mutation process involves adding new connections with random activation functions, changing existing activation functions, or adding a connection between two existing activation functions. The set of available activation functions are linear, step (binary/threshold), sin, cosine, Gaussian, tanh, sigmoid, inverse, absolute value, and ReLU. Though Gaier and Ha admit that they did not experiment much on the number of activation functions, they speculated that the variety of activation functions allowed for decent performance from the WANNs; however, as we will see, simply two activation functions are effective.

3. Spiking WANNs. The overall approach to generating Spiking WANNs is the same as the search in Figure 2.1. The evaluation step of the search is highly paralellizable and is asynchronously evaluated across hundreds of processes. A reduced set of activation functions are used, namely the threshold and linear activation functions. Threshold activation functions themselves can easily be transferred onto neuromorphic hardware; however when combined with a linear activation function, they mimic additive dendritic trees and can be approximated by leaky integrate-and-fire neurons with delays. We recognize that the inputs and outputs of our network may not be fully spiking; however, this can be overcome using approximating networks and/or expanding codings.

Fig. 4.1: The three tasks run are the swingup cartpole task [4] (left), the bipedal walker task [4] (center), and MNIST classification [12] (right)

4. Expiremental Results.

4.1. Tasks. We evaluated primarily four tasks. The first was a cartpole swingup task [4]. The cartpole task is a classic continuous control problem where a pole starting in an upright position must be balanced. The swingup version of this task starts in a resting position with the pole hanging down and needs to be swung upright and balanced, and unlike its simpler counterpart, cannot be solved using a linear controller. The input is angle of the pole, sines/cosines of the angle, and the x coordinate. The expected output is the force of ± 1.

The second task was the BipedalWalker-v2 task for OpenAI Gym [4]. The goal of the task is have a bipedal agent navigate across randomly generated terrain. A positive reward is awarded for distance, while a negative reward for motor torque is given to ensure efficient motions are made. The input is the state of the agent, consisting of the various speeds and positions of different joints and ten LiDAR measurements. Overall, the input consists of 24 dimensions.

The third task was MNIST digit classification [12]. Although for most computer vision tasks, MNIST is low in dimensionality, due to evolutionary approaches requiring making connections at random, convergence can take a long time. Standard MNIST is 28x28, which was reduced to 16x16 to reduce the dimensions.

The fourth task is the Atari Atlantis task, one of the many well-known games in the reinforcement learning community. These games became a prominent RL benchmark starting in 2013 when Mnih et al. [14] published their seminal work on DQN approaches surpassing human performance.

4.2. Comparison to WANNs. Experiments were run on each task using the same parameters from the WANN paper, as seen in Table 4.1, to ensure fair comparisons between them.

The results in Table 4.2 use the reward metric averaged over 100 rollouts for the relevant control tasks along with their standard deviations for the best evolved network topology.

Table 4.1: Parameters used for each task

Task	# of Generations	Population Size
Swingup Cartpole	1024	192
Bipedal Walker	2048	480
MNIST	4096	960

Table 4.2: Results for the various tasks.

	WANN		
	Tuned Shared Weight	Tuned Weights	# of Connections
Swingup Cartpole	723 ± 16	932 ± 6	62
Bipedal Walker	261 ± 58	322 ± 7	338
MNIST	91.9%	94.2%	1228
	Spiking WANN		
	Tuned Shared Weight	Tuned Weights	# of Connections
Swingup Cartpole	745 ± 11	912 ± 5	56
Bipedal Walker	290 ± 22	281 ± 31	210
MNIST	87.7%	88.2%	576

For MNIST classification, the accuracy is given on the test set. The tuned shared weight category is the best shared weight value for the evolved network topology. The tuned weights is when the network's weights are individually trained using a population-based REINFORCE algorithm. The tuned shared weights results are comparable to the original WANN, but there is a degradation in performance when converting into the spiking-like WANN. Other ANN to SNN conversion methods have also shown performance degradation during the switch to threshold activation functions [18].

Interestingly, the tuned shared weights for the spiking WANNs have generally higher performance than the WANN, but the finetuned weights perform worse. This can potentially be attributed to fewer number of weights to finetune, as we see spiking WANNs consistently generate smaller networks.

4.3. Classification. With good results on reinforcement learning tasks, Gaier and Ha explored the capability of WANNs in MNIST classification. They state that classification is unforgiving, as the algorithm is either right or wrong; there is no possibility of recovery as there is in an episode in RL tasks. Table 4.2 shows the performance to be worse across the board for the classification task. Again, this might be related to the significantly smaller sized network generated by the spiking WANN.

4.4. Multi-Objective Optimization. Although all results in Table 4.2 show the best individual, there exists a set of Pareto-optimal solutions since it's a multi-objective optimization problem. Figure 4.2 shows all individuals over the evolutionary process. We can see a Pareto frontier develop on the right hand side of the graph, as we are minimizing the number of connections and maximizing the mean fitness. As generations increase, we see the number of connections are increasing, which is easily noticable by the gradient towards red. The charts only plot the mean fitness and the number of connections; however, the algorithm does an alternating objective optimization, where 20% of the time, the number of connections objective is swapped out with a maximization of peak fitness. This is to encourage growth in the number of connections, as well as performance.

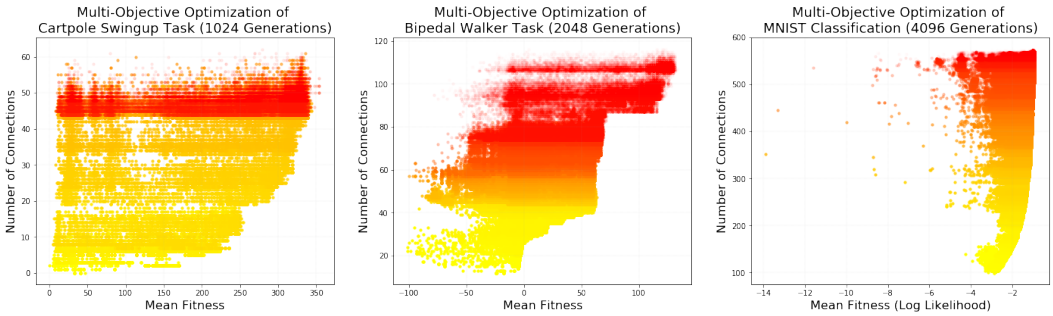

Fig. 4.2: Fitnesses of individuals across multiple generations. The color map is from yellow to red, where the more red a point is, the later generation it comes from

Table 4.3: Results for Atari Atlantis task. Average Human and Random Agents results sourced from [1]. DQN and HyperNEAT results sourced from [16].

Game	Spiking WANN	Average Human	Random Agent	DQN	HyperNEAT
Atlantis	51180.0	29028.1	12850.0	76108.0	61260.0

The color gradient for the cartpole swingup task looks odd due to the lack of a gradient. This is because the cartpole task reached its objective significantly earlier, as seen in Figure 4.3. We see a clear correlation between the number of connections and the fitness values. This is further justification on why the agent is encouraged to ignore the number of connections a certain percentage of the time.

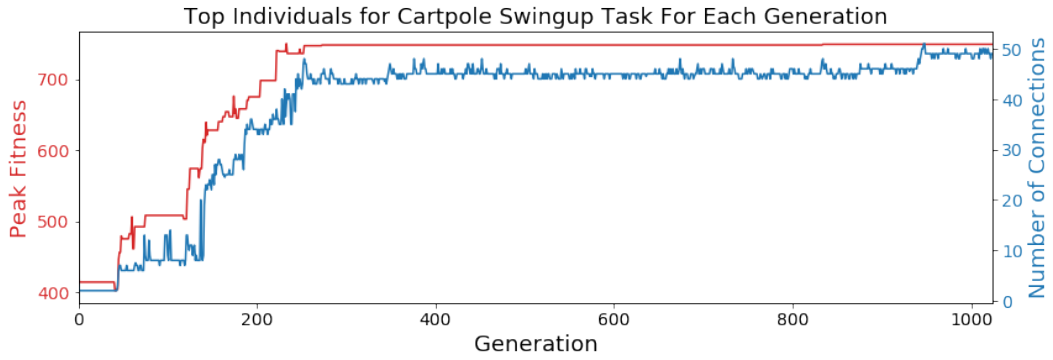

Fig. 4.3: Peak fitness and number of connections for the best individual in each generation. The red peak fitness lines' score is on the left while the number of connections is blue and on the right hand side.

4.5. Atari. Testing on the Atari game, Atlantis, we were able to see that, even without considerable extensions, WANNs are capable of achieving DQN-like performance at the fraction of the computational cost. Due to the high dimensionality of the frames, the input was fed through a ResNet trained on ImageNet, whose final layer was cut off and fed in as

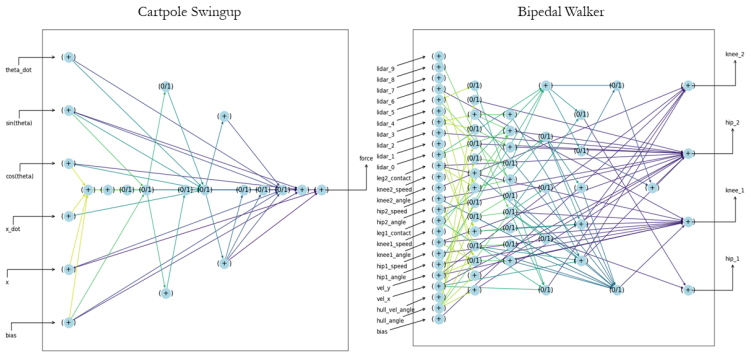

Fig. 4.4: Network topologies of the top individuals for the cartpole swingup task (left) and the bipedal walker task (right).

the input to the WANN. This method would allow the spiking WANN to converge faster from the smaller input space.

The results, as seen in Table 4.3, shows the reward given for the task across various agents. The longer the agent is able to play, the performance increases. The Spiking WANN results are comparable to a DQN, as well as HyperNEAT from Hauskenect et al. [11]. HyperNEAT is a neuroevolution method which evolves an artificial neural network topology using compositional pattern producing networks, allowing for it to efficiently handle large input sizes. The spiking WANN score shows a slight loss in performance compared to the other two methods, but clearly beats a random agent and the average human. The spiking WANN network for this task is using a shared fixed weight rather than finetuned weights due to time constraints. A slight performance boost should occur if the weights are individually trained, as seen in the previous tasks. In addition, the performance of the network is after only 64 generations, where increased generations are likely to increase the performance.

5. Conclusion. We hope to map these, or similar networks, to physical neuromorphic hardware. Again, some inputs and outputs may not be fully spiking, such as the softmax operation used for classification tasks. Methods around this will be useful to explore. In addition, the complexity metric used was the number of connections. This is meant to be a loose approximation of energy usage, but different target architectures perform differently with different network topologies. Exploring energy-based constraints by changing the complexity metric to be true to the target platform would allow for neural network-hardware co-design. In addition, exploring the use of WANNs on a broader set of classification tasks and reinforcement learning tasks would allow us to evaluate its capabilities. It may be worthwhile to investigate changing the parameters, as these spiking-like networks have far fewer activation functions available.

Whetstone refines the activation functions of a typical deep neural network to become threshold activation functions, which can then be used on neuromorphic hardware. Leveraging the representational capabilities of a Whetstone network with a spiking WANN may increase performance on datasets with large input sizes.

There also exists a potential for noise resilience. The evolutionary process for developing the network topology ideally generates networks robust to noise, potentially in the input

space or in the synaptic weights. Future exploration of this domain would make it an ideal candidate for generating networks on neuromorphic hardware where the weights are noisy. Spiking WANNs have been shown to perform well on a variety of tasks. Once a spiking WANN has been implemented onto neuromorphic hardware, we hope to observe significant power savings and reduced energy consumption compared to its traditional counterparts.

6. Acknowledgment. This work was supported by DOE NA-22 funding at Sandia National Laboratories.

REFERENCES

[1] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, and C. Blundell, *Agent57: Outperforming the atari human benchmark*, 2020.
[2] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, *What is the state of neural network pruning?*, in Proceedings of Machine Learning and Systems 2020, 2020, pp. 129–146.
[3] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, *SMASH: one-shot model architecture search through hypernetworks*, CoRR, abs/1708.05344 (2017).
[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *Openai gym*, CoRR, abs/1606.01540 (2016).
[5] H. Cai, L. Zhu, and S. Han, *Proxylessnas: Direct neural architecture search on target task and hardware*, CoRR, abs/1812.00332 (2018).
[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: Nsga-ii*, IEEE transactions on evolutionary computation, 6 (2002), pp. 182–197.
[7] T. Elsken, J. H. Metzen, and F. Hutter, *Neural architecture search: A survey*, 2018.
[8] J. Frankle and M. Carbin, *The lottery ticket hypothesis: Training pruned neural networks*, CoRR, abs/1803.03635 (2018).
[9] A. Gaier and D. Ha, *Weight agnostic neural networks*, (2019).
[10] D. Ha, A. M. Dai, and Q. V. Le, *Hypernetworks*, CoRR, abs/1609.09106 (2016).
[11] M. Hausknecht, J. Lehman, R. Miikkulainen, and P. Stone, *A neuroevolution approach to general atari game playing*, IEEE Transactions on Computational Intelligence and AI in Games, 6 (2014), pp. 355–366.
[12] Y. LeCun and C. Cortes, *MNIST handwritten digit database*, (2010).
[13] H. Liu, K. Simonyan, and Y. Yang, *DARTS: differentiable architecture search*, CoRR, abs/1806.09055 (2018).
[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, *Playing atari with deep reinforcement learning*, CoRR, abs/1312.5602 (2013).
[15] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, *Efficient neural architecture search via parameter sharing*, 2018.
[16] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, *Evolution strategies as a scalable alternative to reinforcement learning*, 2017.
[17] C. D. Schuman, J. P. Mitchell, R. M. Patton, T. E. Potok, and J. S. Plank, *Evolutionary optimization for neuromorphic systems*, in Proceedings of the Neuro-Inspired Computational Elements Workshop, NICE '20, New York, NY, USA, 2020, Association for Computing Machinery.
[18] W. Severa, C. M. Vineyard, R. Dellana, S. J. Verzi, and J. B. Aimone, *Training deep neural networks for binary communication with the whetstone method*, Nature Machine Intelligence, 1 (2019), pp. 86–94.
[19] K. O. Stanley and R. Miikkulainen, *Evolving neural networks through augmenting topologies*, Evolutionary Computation, 10 (2002), pp. 99–127.
[20] J. M. Starck and R. E. Ricklefs, *Patterns of development: the altricial-precocial spectrum*, Oxford Ornithology Series, 8 (1998), pp. 3–30.
[21] H. Zhou, J. Lan, R. Liu, and J. Yosinski, *Deconstructing lottery tickets: Zeros, signs, and the supermask*, in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., Curran Associates, Inc., 2019, pp. 3597–3607.
[22] B. Zoph and Q. V. Le, *Neural architecture search with reinforcement learning*, CoRR, abs/1611.01578 (2016).