Memetic Evolution of Deep Neural Networks

Pablo Ribalta Lorenzo
Future Processing/Silesian University of Technology
Gliwice, Poland
pribalta@ieee.org

Jakub Nalepa
Future Processing/Silesian University of Technology
Gliwice, Poland
jakub.nalepa@polsl.pl

ABSTRACT

Deep neural networks (DNNs) have proven to be effective at solving challenging problems, but their success relies on finding a good architecture to fit the task. Designing a DNN requires expert knowledge and a lot of trial and error, especially as the difficulty of the problem grows. This paper proposes a fully automatic method with the goal of optimizing DNN topologies through memetic evolution. By recasting the mutation step as a series of progressively refined educated local-search moves, this method achieves results comparable to best human designs. Our extensive experimental study showed that the proposed memetic algorithm supports building a real-world solution for segmenting medical images, it exhibits very promising results over a challenging CIFAR-10 benchmark, and works very fast. Given the ever growing availability of data, our memetic algorithm is a very promising avenue for hands-free DNN architecture design to tackle emerging classification tasks.

CCS CONCEPTS

• **Computing methodologies** → *Artificial intelligence*; *Machine learning*; *Neural networks*; *Bio-inspired approaches*;

KEYWORDS

Memetic algorithm, deep neural network, image segmentation

ACM Reference Format:
Pablo Ribalta Lorenzo and Jakub Nalepa. 2018. Memetic Evolution of Deep Neural Networks. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205631

1 INTRODUCTION

Large amounts of computing power and data are widely accessible nowadays. As a result, machine learning systems have scaled up greatly and models have become increasingly sophisticated. This has led to the emergence of DNNs, which have significantly improved the state-of-the-art in computer vision, speech recognition, language processing, and other domains. However, such notable results depend directly on the topology and parametrization of these systems. As DNNs have developed and grown in their complexity, the challenge of successfully configuring them—both in their topology and hyper-parameter values—has surfaced. Conventionally, human experts resort to trial and error, but this quickly becomes infeasible once models are sufficiently intricate. Still, much of the work in DNNs focuses on hand-crafting deep architectures [2].

Latest advances in areas like semantic segmentation have come as a result of breakthroughs in topology design. Aspects as feature re-use [2] or multi-scale context aggregation [34] underline the importance of parallel flows of information and residual connections in DNNs. It is in contrast with the traditional approach for building DNNs, which favors deeper models that are less compact, slower to train, and require more computation effort in practice.

There exist evolutionary algorithms (EAs) for DNN design that perform on par with humans [19]. However, a majority of them are *wrapper* techniques, and they are still difficult to apply in practice due to a large cost of training each candidate solution during the evolution (especially in the case of very deep topologies). Also, their exploration of the search space is often unstructured. Although memetic algorithms (MAs)—the hybrids of EAs and local-search procedures—have been effectively used for solving a plethora of optimization tasks [20], they have not been exploited for designing DNNs. This paper develops an approach for the fully-automated and data-driven design of DNN topologies using an MA.

1.1 Contribution

In this paper, we introduce an MA for DNN topology design which incrementally evolves DNNs starting from only one initial layer. We present a novel mutation operator based on Gaussian process (GP) regression, which encompasses a series of progressively refined optimization steps that allow us to build a surrogate model of the search space. It, in turns, helps guide the search effectively. For feature extraction, we use dilated convolutions [34], which perform implicit feature compression (hence, they remove the need for pooling). We applied our MA to evolve DNNs for segmenting brain tumors from magnetic resonance images (MRI), and showed that it outperforms the state of the art when trained using fairly small and heterogeneous real-life MRI set (also, we investigated its abilities in image classification over the well-known CIFAR-10 benchmark).

An extensive experimental study, involving scenarios with various evolutionary setups, was undertaken. We showed that:

- Our approach designs DNNs that are very competitive in difficult image segmentation and classification tasks, within a very small evolution-time budget.
- Our approach constructs DNN topologies in an *incremental* fashion, starting from only one layer.
- Our approach renders (in a very short time) consistent and high-quality deep models that are more compact (compared with models designed by human experts), come out fully-trained from the

GECCO '18, July 15–19, 2018, Kyoto, Japan
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205631

MA (thus ready to be deployed in the wild), and operate fast and require less computation resources for inference.
- Our approach is data-driven, delivering topologies that adapt to the supplied input dataset.
- Our approach effectively discards redundant and non-promising individuals focusing on unexplored regions of the search space with the highest potential reward (thanks to our guided search).

Experiments are paired with statistical tests that study the main factors influencing the quality of the final DNNs, and were executed to verify the significance of the results. We provide a comprehensive set of illustrations which help get insights about the behavior and capabilities of our MA. The results emphasize its suitability to the topology design task, and its ability to yield DNN architectures that can consistently outperform human experts and other state-of-the-art methods in brain tumor segmentation from MRI.

1.2 Paper Structure

Section 2 discusses the state of the art on evolutionary DNN optimization. In Section 3, we introduce our MA for this task. Section 4 displays the experimental results alongside the discussion on the algorithm performance and behavior. Section 5 concludes the paper.

2 RELATED LITERATURE

Designing DNN topologies is still a big obstacle in applying such classification engines in practice. These tasks are often casted into optimization problems, where the main objective is to construct a DNN which minimizes the classification error over the validation set $\boldsymbol{V}$ when trained using a training set $\boldsymbol{T}$, where $\boldsymbol{T} \cap \boldsymbol{V} = \emptyset$ (ideally without sacrificing the inference speed for the test set Ψ). To keep the training time affordable, the DNN topologies should be only as deep as necessary (shallower networks can be trained much faster, and they are less prone to overfitting) [17].

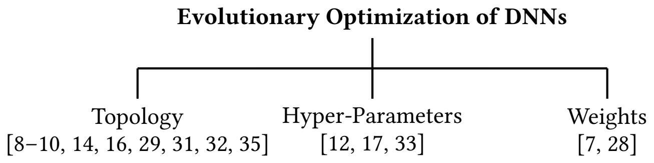

Figure 1: Evolutionary optimization of DNNs—a taxonomy.

Various EAs have been used for optimizing DNNs. They can be divided into three main groups (Figure 1)—those evolving DNN (i) topologies, (ii) hyper-parameters, and (iii) weights. There exist approaches that optimize two or more aspects of DNNs at once.

2.1 Evolution of DNN Topologies

Combining bio-inspired algorithms with neural nets has been present in the literature since the early 1990s, and has led to development of approaches for evolving feed-forward and recurrent neural networks, including the famous NeuroEvolution of Augmenting Topologies (NEAT) [27], which was an inspiration for HyperNEAT [26]. In [11], a cooperative synapse neuroevolution (CoSyNE) was introduced, whereas Salama and Abdelbar utilized ant colonies for building nets [23] (swarm intelligence was exploited for deep recurrent nets as well [9]). Very small recurrent neural network controllers (which are appended to the output of a fixed-topology convolutional neural network, CNN) were evolved in [16]. CNNs were consecutively optimized in a biologically-inspired manner in [35] and [8]. Fernando et al. introduced a Lamarckian algorithm which couples evolution of topologies and learning of weights [10]. In genetic CNNs [32], convolutional structures are encoded in fixed-length binary words, and they undergo a genetic evolution. In [29], Suganuma et al. exploited Cartesian genetic programming to elaborate CNNs from a set of pre-defined blocks. Although the experiments showed that the evolved CNNs are competitive (in terms of classification accuracy) with the state of the art, the proposed approach suffers from an extreme computation cost (the reported evolutions took weeks to finish—such massive execution time is infeasible for large sets). A neuroevolution-based technique to evolve memory-augmented architectures was presented in [14]. In this paper, we propose a technique which falls into this category and allows for an incremental evolution of DNNs with the use of a memetic algorithm (coupling local-search with global exploration).

2.2 Evolution of DNN Hyper-Parameters

Hyper-parameter selection of a fixed-topology DNN is an optimization problem with the objective of minimizing a loss function $\mathcal{L}(\boldsymbol{T}; \mathcal{M})$ for a deep model $\mathcal{M}$ over $\boldsymbol{T}$. The model $\mathcal{M}$ is built by a learning algorithm $\mathscr{A}$ (over $\boldsymbol{T}$), and can be parameterized by the hyper-parameters $\boldsymbol{\lambda}$. Hyper-parameter selection is aimed at finding $\boldsymbol{\lambda}^*$ that yields a deep model $\mathcal{M}^*$, that minimizes $\mathcal{L}(\boldsymbol{V}; \mathcal{M}^*)$ over the validation set $\boldsymbol{V}$. Automated approaches are divided into *model-free* and *model-based* [17]. The former algorithms do not build a surrogate model of the solution space (e.g., random and grid searches [4]), whereas the latter benefit from surrogates. Model-based techniques encompass Bayesian [24] and non-probabilistic optimization [13], and EAs [12, 17, 33].

2.3 Evolution of DNN Weights

DNNs are commonly trained using gradient-based approaches, with back-propagation being the mainstream. Although there exist evolution strategies which were used to substitute back-propagation, Such et al. [28] pointed out that these are in fact gradient-based techniques due to stochastic gradient descent being performed by operations similar to a finite-approximation of a gradient. They also applied non-gradient algorithms for evolving DNN weights. A similar GA was used to evolve deep autoencoders in [7].

2.4 Evolution of More DNN Aspects

There exist approaches that combine the evolution of multiple DNN aspects in a single EA. In [18], the authors introduced a GA to optimize the topology and parameters of DNNs. A similar line of research was presented in [30]—DNN topologies and initial weights were optimized for image classification. Miikkulainen et al. proposed CoDeepNEAT which extends neuroevolution to the DNN topology, components, and hyper-parameters [19].

3 MEMETIC ALGORITHM TO EVOLVE DNNS

Our MA is utilized to design DNN topologies in an incremental fashion, starting from only one layer of depth, and keeping this process ongoing within an evolution-time budget. During the evolution, candidate-solution architectures (*individuals*) are trained using examples from $\boldsymbol{T}$, and the validation accuracy (over $\boldsymbol{V}$) becomes their fitness f. In successive generations, topologies are optimized with the goal of maximizing the validation accuracy. Finally, the generalization performance of the best individual is verified using Ψ, which remains unseen during training. Figure 2 illustrates our method. A noteworthy feature of our MA is that it is *independent* from the dataset that the topology is being trained on, and it can be easily tailored to any new data (it is *data-driven*). Importantly, the final DNN obtained at the end of the evolution is fully-trained and ready to be used. An archive $\mathcal{A}$ is exploited to prevent redundant evaluations of already-known topologies, as well as to store the fittest candidate solution (we apply *elitism*). In the following sections, we discuss the pivotal components of our MA.

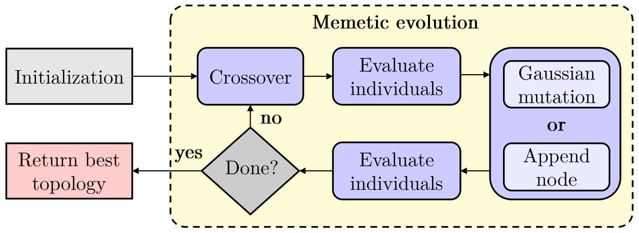

Figure 2: Overview of the proposed MA for evolving DNNs.

3.1 Representation of DNN Architectures

A fundamental feature of our MA is the use of the topology's adjacency matrix as an encoding scheme, to represent the architecture as a directed acyclic graph. Constraints are set in place to ensure its constructibility, as well as to prevent the appearance of recurrent edges while allowing for skip connections. In a formal sense, this premise requires that the entries in the adjacency matrix are $A_{i,j} = 0$, if $i \leq j$, and $A_{i,j} = 1$, if $i = j - 1$. This scheme focuses exclusively on the generation of an upper-right triangle-shaped part of a matrix, while it enforces an empty lower-left triangle-shaped part of it, together with its main diagonal. Also, it significantly limits the parts of the chromosome that can be subjected to mutation, allowing for the development of highly structured topologies through reasonable evolution cycles. This invariance plays a second paramount role, as it allows us to bound the fitting of the Gaussian process (Section 3.4), and ensures that new individuals produced by crossover are fully constructible (Section 3.3). An example of an encoded topology, and its network graph are rendered in Figure 3.

3.2 Memetic Evolution

Let $\mathcal{I}_{i,k}$ be the i-th individual of rank k in a population $\mathcal{P}$, and $\mathcal{A}$ be an archive (a key-value store). The goal of the MA is to find a solution $\mathcal{I}^*$ for which $f(\mathcal{I}^*) \geq f(\mathcal{I})$ for all $\mathcal{I} \in \mathcal{A}$, where $f(\mathcal{I}_i)$ is defined as the accuracy of the trained i-th topology over $\boldsymbol{V}$. In MA, $\mathcal{P}$ undergoes a memetic evolution (Algorithm 1, lines 3–14) within a time budget denoted as $\tau_{\max}$. Initially, the archive $\mathcal{A}$ is empty, and $\mathcal{P}$ is initialized with two individuals with an adjacency matrix of rank $k = 1$, and it is subjected to several transformations during the

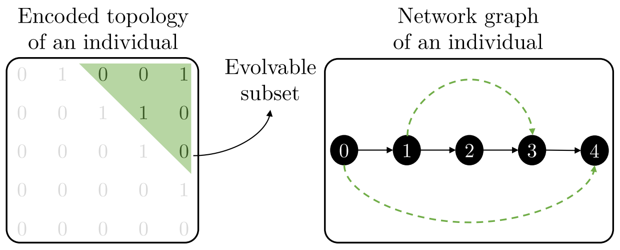

Figure 3: Example adjacency matrix with a mutable (green) part of an individual (left), and its resulting topology (right).

optimization. First, crossover is performed (Section 3.3)—it duplicates the number of individuals, and an intermediate population $\mathcal{P}'$ of four candidate solutions is created (line 4). For each individual in $\mathcal{P}'$, if not present in $\mathcal{A}$, its fitness is calculated and stored in $\mathcal{A}$. Afterwards, each child ($\mathcal{I}_1^{\text{child}}$ and $\mathcal{I}_2^{\text{child}}$) is mutated (Section 3.4), or its topology is augmented (Section 3.5), with equal probability (lines 8–13). The fitness of $\mathcal{I}_1^{\text{child}}$ and $\mathcal{I}_2^{\text{child}}$ are calculated (if they are not stored in $\mathcal{A}$ yet). Finally, the best two solutions from $\mathcal{P}'$ and $\mathcal{A}$ survive (line 14), and constitute $\mathcal{P}$ (the population size is kept constant and equals 2 during the evolution).

Algorithm 1 Memetic evolution of DNN topologies.

```
1: procedure MemeticEvolution($\tau_{\max}$)
2:     $\mathcal{P} \leftarrow \{I_{0,k=1}, I_{1,k=1}\}$; $\mathcal{A} \leftarrow \emptyset$          ▷ Initial population
3:     while $\tau \leq \tau_{\max}$ do
4:         $\mathcal{P}' \leftarrow crossover(\mathcal{P})$
5:         for each $\mathcal{I}_i \in \mathcal{P}'$ do
6:             if $\mathcal{I}_i \notin \mathcal{A}$ then
7:                 $\mathcal{A} \leftarrow \{\mathcal{I}_i, f(\mathcal{I}_i)\}$
8:             if $\epsilon \sim \mathcal{U}(0,1) \geq 0.5$ then
9:                 $mutate(\mathcal{I}_i)$                        ▷ Gaussian mutation
10:            else
11:                $appendNode(\mathcal{I}_i)$                    ▷ $\mathcal{I}_{i,k} \longrightarrow \mathcal{I}_{i,k+1}$
12:            if $\mathcal{I}_i \notin \mathcal{A}$ then
13:                $\mathcal{A} \leftarrow \{\mathcal{I}_i, f(\mathcal{I}_i)\}$
14:        $\mathcal{P} \leftarrow \{\text{best}(\mathcal{P}'), \text{best}(\mathcal{A})\}$
15:    return $best(\mathcal{A})$                                ▷ Best individual
```

3.3 Crossover

We introduce a single-point crossover for mating parents $\{\mathcal{I}_0, \mathcal{I}_1\}$. The crossover point is selected randomly (following a uniform distribution $\mathcal{U}(0, u^{\max})$, where $u^{\max} = \min(rank(\mathcal{I}_i))$ for all $\mathcal{I}_i \in \mathcal{P}$). If both $\mathcal{I}_0$ and $\mathcal{I}_1$ have the same rank, this defaults into a seamless exchange of rows between two individuals. In those instances where $\mathcal{I}_0$ and $\mathcal{I}_1$ have different ranks, columns are clipped or extended to their new dimensions as needed (an example is presented in Figure 4). The goal is to satisfy—whenever possible—the constructibility constraints (Section 3.1) for a resulting DNN topology.

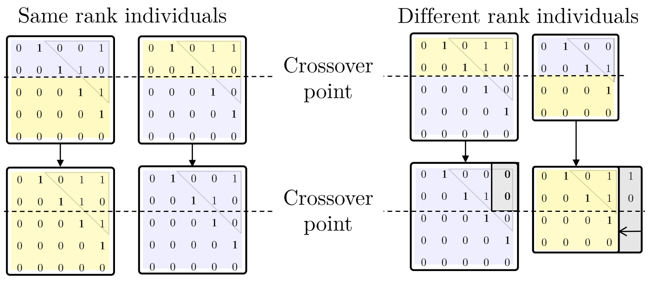

Figure 4: Crossing over the same-rank (left) and different-rank (right) chromosomes.

3.4 Gaussian Mutation

In this paper, we define a novel Gaussian mutation (GM) operator based on a Gaussian process [21], with the goal of ensuring that the new individuals produced by applying GM are positioned in the regions of the search space where the fitness values are likely to be high. In GM, visited positions (i.e., already-evaluated high-quality chromosomes) act as priors refining future predictions.

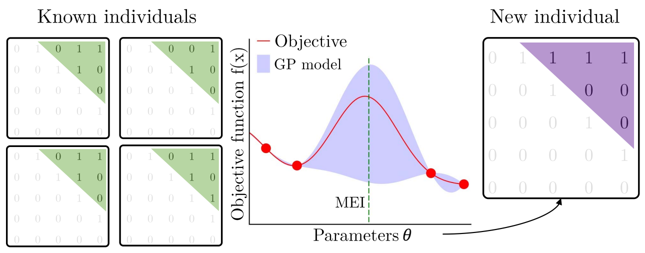

Figure 5: Example of how known individuals are fit to a Gaussian process in order to produce a new individual.

Let $\mathbb{I}_k$ denote all evaluated individuals of rank k, and $f(\mathcal{I}_i)$ be the fitness of an individual $\mathcal{I}_i$. Having n evaluated positions $\{\mathcal{I}_i, f(\mathcal{I}_i)\}$ in the search space, we consider them to be a training set T^{GP}, where $\mathcal{I}_i \in \mathbb{I}_k$ and $f(\mathcal{I}_i) \in \mathbb{R}$. A Gaussian-process regression model addresses the question of predicting the value of a response variable $f(\mathcal{I}_{\mathrm{new}})$, given the new input vector $\mathcal{I}_{\mathrm{new}}$, and the training set T^{GP}. Our predictions are obtained after fitting $\mathbb{I}_k$ to a Gaussian process regression with a prior mean assumed to be constant and zero, employing a stationary squared-exponential kernel. Once the Gaussian-process regression is fit to these known priors, candidate points are obtained by computing the maximum expected improvement (MEI) over the Gaussian-process in the evolvable chromosome boundaries (as introduced in Section 3.1). Figure 5 illustrates this procedure. A noteworthy feature of our guided mutation is the incremental build-up of information about the search landscape. It favors the creation of skip connections that re-use available information, as opposed to the construction of deeper networks that are ultimately slower to train.

3.5 Topology Augmentation

As a complement to GM, a second operator is introduced to ensure that the evolution retains the ability to produce deeper architectures by appending a node at the end of an individual's chromosome. This operator complements the mutation to guarantee the balance between *horizontal* (wider topologies) and *vertical* (deeper topologies) exploration of the search space. For this purpose, the rank of a chromosome's adjacency matrix is increased by one, while satisfying the constructibility requirements (Figure 6). It requires that $A'_{k,k-1} = 1$ for any new chromosome, where k is its rank.

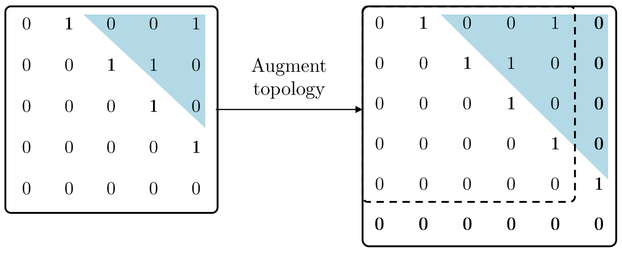

Figure 6: The process of augmenting a topology requires increasing the rank of the adjacency matrix while ensuring that the new topology is still constructible.

4 EXPERIMENTAL CASE STUDY: BRAIN TUMOR SEGMENTATION FROM MRI

In an application case study with real-world data, our MA is employed to design DNNs for detection and segmentation of brain tumors from fluid attenuation inversion recovery (FLAIR) MRI. Due to the scarcity of large sets of full-sized MRIs annotated by physicians, the need to design architectures that focus in feature re-use and context aggregation becomes primordial. In this task, our MA was trained with 256×256 pixel patches extracted from annotated scans (of size 512×512 pixels). The results should underpin the ability of our MA to maximize the re-use of available information while designing compact models in a constrained time budget.

4.1 Experimental Setup

Our MA was implemented in Python with NumPy, and DNNs were trained using Keras [5] with TensorFlow over CUDA 8.0 and CuDNN 5.1. For Gaussian processes, we used the George library [1]. Experiments ran on Intel i7-6850K (15M Cache, 3.80 GHz) with 32 GB RAM and NVIDIA Titan X Ultimate Pascal GPU 12GB GDDR5X.

We used a fixed evolution-time budget $\tau_{\max} = 60$ minutes. All layers are formed by dilated convolutions [34], with stride $s = 1 \times 1$, and 64 filters of size 5×5. Exponential linear units [6] are used as activation, and merge layers are employed on skip connections. The topologies are terminated by a non-dilated convolution with one 1×1 filter, to adapt the output to a grayscale mask. We exploit the ADAM optimizer [15], with an initial learning rate of 10^{-5}.

Topologies are trained until their fitness does not increase for 2 epochs, with each epoch being comprised of 500 training batches of 4 images of size 256×256 each, and 200 batches of the same size are used for validation, and 200 batches for testing. We employ a cross-validation (leave-one-out) scheme of 11/2/1 patients for training, validation, and test, respectively, with a total of 14 folds over 14 patients. DICE coefficient is used as the objective function to be optimized, representing a measure of similarity of the overlap between the ground truth (A) and the obtained annotations (B). It is given as DICE $= \frac{2 \cdot |A \cap B|}{|A|+|B|}$ (the value closer to 1.0, the better).

4.2 Dataset

The MRI brain-tumor data was collected for patients who underwent the imaging with MAGNETOM Prisma 3T (Siemens) equipped with a max. field gradient strength of 80 mT/m, and using a 20-channel quadrature head coil. The MRI sequences were acquired in the axial plane with a field of view of 230 × 190 mm, matrix size 256 × 256 and 1 mm slice thickness with no slice gap. We used the SPACE FLAIR (TE=386 ms, TR=5000 ms, inversion time: 1800 ms) sequences annotated by a medical physicist (7 years of experience).

This study consisted of 14 patients entered retrospectively into the study, each with a grade II WHO brain tumor (confirmed in a neurosurgical intervention), who have been imaged with the FLAIR MRI sequences. Patient ages at the time of scanning ranged from 24 to 60 (with the average of 38.43), and the cohort consisted of 7 females and 7 males. Our dataset exposes high variability of the tumor location (4 tumors in left, and 2 in right frontal, temporal and insular areas, 1 in left, and 2 in right frontal lobe, 2 in left, and 2 in right temporal areas, and 1 in left frontal and parietal area).

4.3 Memetic Evolution of DNNs

Initially, 4 different setups are created as part of an ablation study (Table 1), systematically increasing the number of features in the MA to characterize their overall effect. Setup 0 consists of a naive MA with the aim of expanding the topology *in depth* by adding nodes to the chromosome, which happens for each individual at the end of every generation. Mutation is performed in Setup 0 by flipping a bit in the evolvable regions of the chromosome (Section 3.1) with a fixed mutation rate of 0.05. Setup 1 retains the same features as Setup 0, however mutated individuals are initialized with the weights of their trained predecessors. Setup 2 prevents both mutation and topology augmentation from happening in the same generation, granting equal probability for them. Setup 3 substitutes the naive mutation operator by the Gaussian-process operator (Section 3.4), and directly maps to the Algorithm 1.

Table 1: Setups of our MA investigated in the ablation study.

Name	Weight re-use	Mutation probability	Augmentation probability	Mutation operator
Setup 0	No	Always	Always	Naive
Setup 1	Yes	Always	Always	Naive
Setup 2	Yes	.5	.5	Naive
Setup 3	Yes	.5	.5	GP

4.4 Parent-Offspring Weight Inheritance

In this experiment, we focus on characterizing the effect of propagating the trained weights from parent topologies to their offspring during mutation and crossover. Weights in DNNs are commonly initialized with values drawn from a probabilistic distribution. During training, these weights are progressively adjusted in order to minimize the error at the output of the network which turns the convolutional filters into powerful feature extractors. Especially in scenarios where mutation can add a residual connections to the offspring, weight re-use can be an important tool to shorten the training time of the new network. When performing crossover, this mechanism of weight inheritance can help discover new structures, as it allows for the recombination of feature extractors that may come from radically different contexts (low-level feature extractors from shallow topologies might blend in interesting ways with high-level feature extractors from deeper networks).

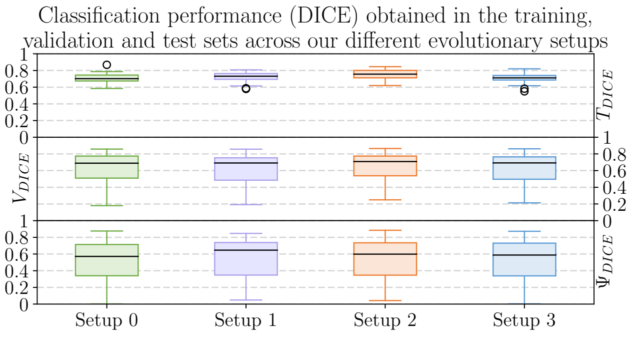

Figure 7: Classification performance (DICE) obtained in T (top), V (middle), and Ψ (bottom) by all setups shows that the *horizontal* exploration can rival exploration *in depth*.

It is observable in Figure 7 that there are no significant differences between Setup 0 and Setup 1 in terms of DICE measured on $\boldsymbol{T}$, $\boldsymbol{V}$ and Ψ, with average DICE for $\boldsymbol{T}$ close to 0.7 and small variance (this is confirmed by the Wilcoxon test, $p = .56$). On $\boldsymbol{V}$ and Ψ, results look fairly similar, with average V_{DICE} and Ψ_{DICE} approx. 0.7 and 0.6 respectively, although with an improved Ψ_{DICE} for Setup 1.

Table 2: Comparison of all metrics (mean±standard deviation) gathered across setups.

Name	Unique individuals evaluated	Total individuals evaluated	Evolution time (s)	Solution rank k
Setup 0	7.04 ± 1.16	12.17 ± 6.84	5,110 ± 818	8.28 ± 6.28
Setup 1	7.26 ± 1.21	12.84 ± 8.75	4,662 ± 569	8.83 ± 8.94
Setup 2	7.01 ± 1.28	14.91 ± 14.1	**4,008 ± 809**	7.74 ± 6.23
Setup 3	**8.34 ± 1.64**	**8.94 ± 2.52**	4,547 ± 850	**5.22 ± 1.02**

The contrast between these setups becomes visible when analyzing their behavior during the search. In Table 2, we observe that Setup 1 calculates (on average) the fitness of more individuals in the course of evolution. Total optimization time is reduced too in Setup 1 by an average of 10 minutes (17% of the total optimization budget). This is an indicator of the effectiveness of the weight inheritance, allowing to speed up the training without compromising the accuracy. Note that the total evolution time of any setup may exceed the assumed time budget $\tau_{\max}$ (1 hour)—it happens when the last generation is started just before the time limit, and hence it extends the total evolution time beyond $\tau_{\max}$. With respect to the average rank of the solution, faster training allows exploring deeper topologies in Setup 1 (it favors *vertical exploration*).

4.5 Balancing Vertical and Horizontal Search

In this experiment, we compare Setup 1 and 2, and investigate how balancing the vertical and horizontal search affects the production of deeper topologies while increasing the tendency to build residual connections. A possible downside of this approach is that naive horizontal mutation can result in *redundant* individuals (for which their fitness was already calculated in previous generations) whose evaluation (albeit instant, due to the presence of the archive) does not improve the optimization, as it cannot increase the diversity in the population. However, it represents our first step towards producing shallower, more compact and interconnected topologies.

In Table 2, we can observe the aforementioned tendency of Setup 2 to render redundant individuals, as the values for the average of total individuals evaluated rises by 14% with respect to Setup 1. Variance practically duplicates itself, which is a serious indicator of how compromised the population diversity becomes within this setup. The average rank of an individual is notably smaller, confirming the intuition that by balancing *horizontal* and *vertical* exploration, more compact topologies can be achieved. Although the results for T_{DICE} and Ψ_{DICE} remain similar (Figure 7), Setup 2 achieves higher DICE for V_{DICE} (Wilcoxon test, $p < .05$).

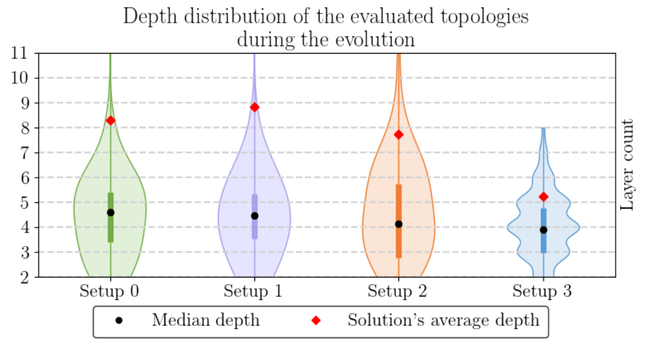

Figure 8: The depth distribution of the evolved topologies illustrates the ability of GM to effectively exploit neighborhoods of already-found architectures.

4.6 Exploiting Gaussian Mutation

In this last experiment, we focus on measuring the improvement delivered by GM (Section 3.4) when compared to the use of a naive mutation operator (thus confronting Setup 2 and 3). The introduction of GM has the effect of maximizing the chances of building residual connections in evolved topologies, priming the design of interconnected networks over deeper architectures. A very desirable property of the introduced operator is its ability to exploit unexplored regions of the search space, based on the attained knowledge about the space (we traverse the neighborhood of known DNNs). It virtually removes the presence of redundant individuals (i.e., individuals which were already evaluated in previous generations and are appended to the archive), and helps promote the diversity in the population. However rare, such redundant candidate solutions can still be generated as an output of crossover.

The search abilities of GM can be observed in Table 2, where Setup 3 consistently reports the largest number of unique individuals, which is remarkably close to the number of total individuals produced during the evolution. On average, there is less than 7% of chances of producing a redundant chromosome in Setup 3—ensuring the diversity and underpinning the ability of GM to perform highly-effective local search of the evolvable regions of the chromosomes. Another noteworthy feature of GM is its capacity to produce architectures that maximize feature re-use, providing—by a large margin—solutions of smaller rank than all other Setups.

Figure 8 provides insights about the depth distribution of the individuals that underwent evolution. Unlike Setups 0, 1 and 2, which display a tendency to explore deeper topologies, Setup 3 remains remarkably compact. It is also visible that Setup 3 performs *horizontal* exploration (see the layer count). Setup 3 retrieves the minimum median depth, as well as the average depth and variance (Table 2). Interestingly, Setups 0 and 1 operate at the same level of V_{DICE} as Setup 3 ($p = .98$ and $p = .48$, Wilcoxon test). It shows that compact DNNs *do not* sacrifice the quality of the final model. Although Setup 3 obtained worse DICE scores compared to Setup 2 ($p < .05$), the difference in Ψ_{DICE} was not statistically important (hence, both setups evolve DNNs which generalize fairly well).

4.7 Segmenting Full-Size MRI Scans

We tackled a real-life problem of segmenting full-size MRI scans (as opposed to segmenting patches, as reported in previous sections). We trained our DNNs using patches of size 256×256 displaying tumors (stage II WHO, see Section 4.2), and we confronted it with two state-of-the-art methods for medical image segmentation. For the first method (SSA–**S**uperpixel **S**egmentation **A**lforithm) [25], we replaced the SLIC superpixel grouping with its non-parametric version. We carefully investigated the number of superpixels to be found in each image, and set this parameter empirically (in the knee-point analysis) to 500. We extract all features mentioned in [25], and we additionally include two new ones (the average and variance of the Laplacian of an image). The whole training process for SSA lasts for approx. 17 hours. For the second method, we exploit a U-Net that was trained for 1 hour (the same as the evolution budget of our MA), with DICE as its loss coefficient. We apply the best topologies (on the validation set) retrieved by Setup 3 (denoted as $\overline{s_3}$) and U-Net to segment images with a sliding-window strategy, reporting as a result the patch with the highest average activation.

Observing Figure 9, it becomes apparent that the budget of 1 hour is insufficient for training a well-operating U-Net. Its high rate of false positives (FPs—pixels of healthy tissue annotated as tumorous) is the main factor driving down its overall performance over Ψ. Both SSA and $\overline{s_3}$ achieve decent average DICE, with $\overline{s_3}$ (despite $\overline{s_3}$ being trained for only a fraction of SSA's training time), delivering the highest average Ψ_{DICE} of nearly 0.6. However, there is still great variance between the results reported for each fold, which can respond to tumor distinctiveness—it becomes patent in folds 3 and 4. FPs remain very low for all folds for our MA and SSA.

4.8 Qualitative Analysis

In Figure 10, we present examples of images segmented using the best model retrieved by Setup 3. We can observe that the obtained

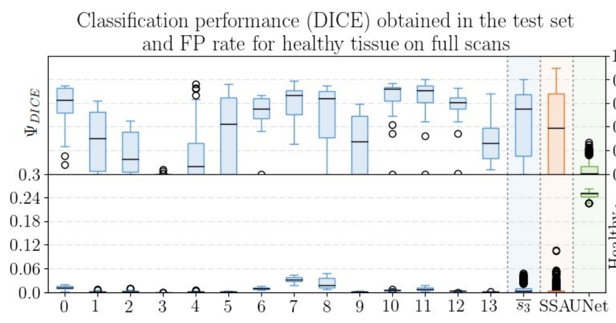

Figure 9: Segmentation results over full-size MRI scans demonstrate that our MA outperforms other state-of-the-art methods in a constrained training budget.

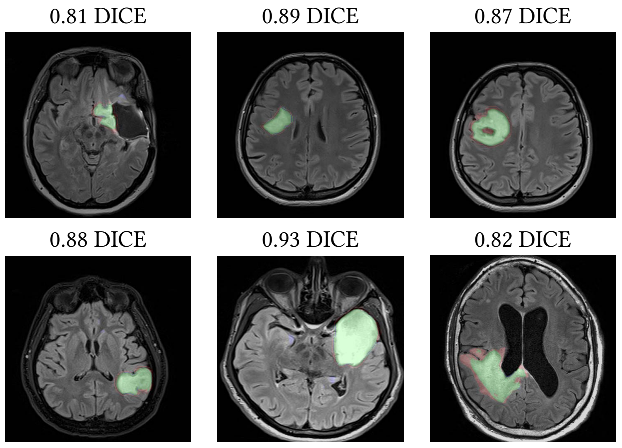

Figure 10: Examples of segmentations (alongside the corresponding DICE) provided by the best Setup 3 DNN over full-size MRI scans. True positives are rendered in green, false negatives in red, and false positives in blue.

segmentations match closely the annotations provided as ground truth by a human expert. In green, we render the areas of overlap between the DNN-elaborated annotations and the provided manual segmentation, FPs are shown in blue, and false negatives (tumorous pixels labeled as healthy by our DNN)—in red. Finding the tumor boundaries is not a trivial task, and its difficulty strongly depends on the tumor characteristics (deciding on the lesion boundary is in fact an open issue in the literature). Also, even two different physicians may disagree in the very same image which makes a preparation of the gold standard extremely challenging. From the qualitative examples shown for our evolved deep model, we can conclude that it effectively detects and segments suspicious parts of the brain tissue. Although the automated segmentation may potentially undergo further investigation by an experienced reader (to minimize the number of FNs, hence to decrease the chance of not spotting cancerous tissue in brain), our DNN currently exhibits high-quality segmentation abilities (although being evolved within a very constrained computation budget). Another interesting insight arises while analyzing the FNs around a tumor (see the 3rd image in the 2nd row of Figure 10)—it may have happened that a radiologist highlighted too large area (including healthy tissue). Such cases should be inter-agreed by two (or more) experienced physicians.

4.9 Discussion

Our ablation study showed that the Gaussian process based mutation plays a pivotal role in the generation of well-performing offspring solutions, while keeping a high degree of variety of chromosomes in the population. Compared to a naive, probability-based mutation operator, GM practically eliminates the necessity of evaluating redundant individuals. The cost of fitting a Gaussian regression appears to not have a significant impact on the evolution time, possibly due to the fact that this regression is bounded, and the number of priors is not extremely large. Balancing *horizontal* and *vertical* exploration reinforces the ability of GM to render densely interconnected topologies that offer a high degree of feature re-use, even under a constrained training budget. Weight inheritance is also an aspect that cannot be overlooked, as it delivers a significant speedup in training without compromising the final performance. On average, MA-evolved DNN segments a 512×512 MRI scan in 10 ms, which is orders of magnitude faster than any practitioner, and it outperforms the investigated state-of-the-art algorithms (SSA took approx. 200 ms, and U-Net consumed approx. 30 ms on average).

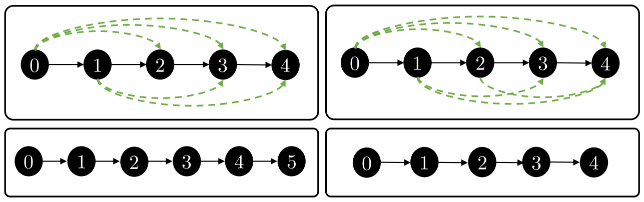

Figure 11: Examples of best MA-evolved topologies for different folds suggest that deeper architectures can achieve a higher accuracy than wider ones—even with skip connections—in different subsets of the same training set.

The ability of designing DNN topologies that exploit the available information as best as possible becomes a key aspect of the success in a real-world medical image analysis. Our MA is able to design architectures that obtain competitive results in brain tumor segmentation of full-size MRI scans. Automatically designed DNNs can be trained on small patches, and they can be effectively applied over a 512×512 scan in a sliding-window manner, mitigating the need for large and annotated sets of full-sized MRIs. Additionally, our data-driven approach renders topologies that adapt to the tumors seen in each fold (returning deeper or shallower topologies as needed—see examples in Figure 11). This is a very desirable feature that can be exploited to create powerful DNN ensembles (however, it may also indicate overfitting, hence should be analyzed carefully).

5 CONCLUSIONS AND OUTLOOK

In this paper, we proposed an MA for the automated design of DNN topologies. A population of individuals is incrementally evolved to yield the best architecture design that can be fit to a particular task. Improvements like weight inheritance and a novel Gaussian process based mutation operator are integral components of this approach. An extensive ablation study was performed on a real-world application of brain tumor segmentation from MRI, to characterize the behavior of the algorithm, and to understand how these key improvements affect the outcome of our MA. This study revealed that weight inheritance accelerates the training of DNNs without compromising their performance, and showed the very desirable properties of Gaussian mutation at generating high-quality offspring via search space exploitation. DNNs designed with our MA surpassed the performance of manually designed models, showing a remarkable ability to deliver solutions (in a very constrained time budget) adapted to the input data. Given the ever growing availability of data, MAs are a promising avenue for hands-free DNN design to tackle new classification tasks in a data-driven manner.

Table 3: Results obtained on a CIFAR-10 benchmark for different evolutionary topology design methods.

Model	Error	Execution time	# GPUs
SimpleNet-1_3 [17]	40.41	90 min	1
Memetic Evolution	27.73	120 min	1
MetaQNN [3]	9.09	8-10 days	10
CoDeepNEAT [19]	7.30	n/a	n/a
L-S Evolution [22]	5.90	~10 days	250 CPUs
CGP-CNN (Conv.) [29]	6.34	15.2 days	2
CGP-CNN (Res) [29]	6.05	13.7 days	2
NA Search [35]	3.65	n/a	800

We currently focus on adapting our MA to different problems beyond image segmentation. A straightforward implementation of our method for a classification task over the multi-class CIFAR-10 set shows fairly promising results elaborated in a very short time—orders of magnitude lower than reported in other works (Table 3). By replacing dilated convolutions with regular convolutions, and terminating the network with a combination of max-pooling and fully-connected layers, we observe that new topologies can successfully classify images. Additionally, we observed that increasing the MA computation budget allowed us to further decrease the error rate over CIFAR-10, but it requires further investigation.

ACKNOWLEDGMENTS

This work was supported by the Polish National Centre for Research and Development under Grant POIR.01.02.00-00-0030/15.

REFERENCES

[1] Sivaram Ambikasaran, Daniel Foreman-Mackey, Leslie Greengard, David W. Hogg, and Michael O'Neil. 2016. Fast Direct Methods for Gaussian Processes. *IEEE Trans. Pattern Anal. Mach. Intell.* 38, 2 (2016), 252–265.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. 2017. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 12 (Dec 2017), 2481–2495.

[3] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2016. Designing Neural Network Architectures using Reinforcement Learning. *CoRR* abs/1611.02167 (2016), 1–18. arXiv:1611.02167

[4] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research* 13 (2012), 281–305.

[5] François Chollet et al. 2015. Keras. https://github.com/fchollet/keras. (2015).

[6] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *CoRR* abs/1511.07289 (2015), 1–14.

[7] Omid E. David and Iddo Greental. 2014. Genetic Algorithms for Evolving Deep Neural Networks. In *Proc. GECCO Companion*. ACM, USA, 1451–1452.

[8] Travis Desell. 2017. Large Scale Evolution of Convolutional Neural Networks Using Volunteer Computing. In *Proc. GECCO*. ACM, USA, 127–128.

[9] Travis Desell, Sophine Clachar, James Higgins, and Brandon Wild. 2015. Evolving Deep Recurrent Neural Networks Using Ant Colony Optimization. In *Proc. EvoCOP*, Gabriela Ochoa and Francisco Chicano (Eds.). Springer, Cham, 86–98.

[10] Chrisantha Fernando, Dylan Banarse, Malcolm Reynolds, Frederic Besse, David Pfau, Max Jaderberg, Marc Lanctot, and Daan Wierstra. 2016. Convolution by Evolution: Differentiable Pattern Producing Networks. In *Proc. GECCO*. ACM, USA, 109–116.

[11] Faustino Gomez, Juergen Schmidhuber, and Risto Miikkulainen. 2008. Accelerated Neural Evolution through Cooperatively Coevolved Synapses. *Journal of Machine Learning Research* (2008), 937–965.

[12] Delowar Hossain, Genci Capi, and Mitsuru Jindai. 2017. Evolution of Deep Belief Neural Network Parameters for Robot Object Recognition and Grasping. *Procedia Computer Science* 105 (2017), 153 – 158.

[13] Ilija Ilievski, Taimoor Akhtar, Jiashi Feng, and Christine Annette Shoemaker. 2017. Efficient Hyperparameter Optimization for Deep Learning Algorithms Using Deterministic RBF Surrogates. In *Proc. AAAI*. 822–829.

[14] Shauharda Khadka, Jen Jen Chung, and Kagan Tumer. 2017. Evolving Memory-augmented Neural Architecture for Deep Memory Problems. In *Proc. GECCO*. ACM, USA, 441–448.

[15] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014), 1–15.

[16] Jan Koutník, Juergen Schmidhuber, and Faustino Gomez. 2014. Evolving Deep Unsupervised Convolutional Networks for Vision-based Reinforcement Learning. In *Proc. GECCO*. ACM, USA, 541–548.

[17] Pablo Ribalta Lorenzo, Jakub Nalepa, Michal Kawulok, Luciano Sanchez Ramos, and José Ranilla Pastor. 2017. Particle Swarm Optimization for Hyper-parameter Selection in Deep Neural Networks. In *Proc. GECCO*. ACM, USA, 481–488.

[18] A. Martín, F. Fuentes-Hurtado, V. Naranjo, and D. Camacho. 2017. Evolving Deep Neural Networks architectures for Android malware classification. In *Proc. IEEE CEC*. 1659–1666.

[19] Risto Miikkulainen, Jason Zhi Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, and Babak Hodjat. 2017. Evolving Deep Neural Networks. *CoRR* abs/1703.00548 (2017), 1–8.

[20] Jakub Nalepa and Michal Kawulok. 2016. Adaptive memetic algorithm enhanced with data geometry analysis to select training data for SVMs. *Neurocomputing* 185 (2016), 113 – 132.

[21] Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning*. The MIT Press. 1–266 pages.

[22] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Quoc V. Le, and Alex Kurakin. 2017. Large-Scale Evolution of Image Classifiers. *CoRR* abs/1703.01041 (2017), 1–18.

[23] Khalid Salama and Ashraf M. Abdelbar. 2014. A Novel Ant Colony Algorithm for Building Neural Network Topologies. In *Proc. ANTS*. Springer, 1–12.

[24] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Proc. NIPS*. Curran, 2951–2959.

[25] Mohammadreza Soltaninejad and et al. 2017. Automated brain tumour detection and segmentation using superpixel-based extremely randomized trees in FLAIR MRI. *Int. J. of Computer Assisted Radiol. and Surgery* 12, 2 (2017), 183–203.

[26] Kenneth O. Stanley, David B. D'Ambrosio, and Jason Gauci. 2009. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life* 15, 2 (2009), 185–212.

[27] Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving Neural Networks Through Augmenting Topologies. *Evol. Computation* 10, 2 (2002), 99–127.

[28] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2017. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *CoRR* abs/1712.06567 (2017), 1–15.

[29] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. 2017. A Genetic Programming Approach to Designing Convolutional Neural Network Architectures. In *Proc. GECCO*. ACM, USA, 497–504.

[30] Yanan Sun, Bing Xue, and Mengjie Zhang. 2017. Evolving Deep Convolutional Neural Networks for Image Classification. *CoRR* abs/1710.10741 (2017), 1–14.

[31] Yanan Sun, Gary G. Yen, and Zhang Yi. 2017. Evolving Unsupervised Deep Neural Nets for Learning Meaningful Representations. *CoRR* abs/1712.05043 (2017), 1–23.

[32] Lingxi Xie and Alan L. Yuille. 2017. Genetic CNN. In *Proc. ICCV*. 1388–1397.

[33] Steven R. Young, Derek C. Rose, Travis Johnston, William T. Heller, Thomas P. Karnowski, Thomas E. Potok, Robert M. Patton, Gabriel N. Perdue, and Jonathan Miller. 2017. Evolving Deep Networks Using HPC. In *Proc. MLHPC@SC*. 7:1–7:7.

[34] Fisher Yu and Vladlen Koltun. 2015. Multi-Scale Context Aggregation by Dilated Convolutions. *CoRR* abs/1511.07122 (2015), 1–13.

[35] Barret Zoph and Quoc V. Le. 2016. Neural Architecture Search with Reinforcement Learning. *CoRR* abs/1611.01578 (2016), 1–16.