IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 26, 2019, accepted December 24, 2019, date of publication January 6, 2020, date of current version January 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2964424

Neural Architecture Search for Skin Lesion Classification

ARKADIUSZ KWASIGROCH, MICHAŁ GROCHOWSKI, AND AGNIESZKA MIKOŁAJCZYK
Faculty of Electrical and Control Engineering, Gdańsk University of Technology, 80-233 Gdańsk, Poland

Corresponding authors: Arkadiusz Kwasigroch (arkadiusz.kwasigroch@pg.edu.pl) and Michał Grochowski (michal.grochowski@pg.edu.pl)

This work was supported by the Ministry of Science and Higher Education in the years 2017–2021, under Diamond Grant DI2016020746.

ABSTRACT Deep neural networks have achieved great success in many domains. However, successful deployment of such systems is determined by proper manual selection of the neural architecture. This is a tedious and time-consuming process that requires expert knowledge. Different tasks need very different architectures to obtain satisfactory results. The group of methods called the neural architecture search (NAS) helps to find effective architecture in an automated manner. In this paper, we present the use of an architecture search framework to solve the medical task of malignant melanoma detection. Unlike many other methods tested on benchmark datasets, we tested it on practical problem, which differs greatly in terms of difficulty in distinguishing between classes, resolution of images, data balance within the classes, and the number of data available. In order to find a suitable network structure, the hill-climbing search strategy was employed along with network morphism operations to explore the search space. The network morphism operations allow for incremental increases in the network size with the use of the previously trained network. This kind of knowledge reusing allows significantly reducing the computational cost. The proposed approach produces structures that achieve similar results to those provided by manually designed structures, at the same time making use of almost 20 times fewer parameters. What is more, the search process lasts on average only 18h on single GPU.

INDEX TERMS Deep learning, convolutional neural network, neural architecture search, network morphism, malignant melanoma.

I. INTRODUCTION

Deep learning has achieved great success in various fields, such as image processing, natural language processing, image generation, speech generation, and many more. These methods allow processing of high-dimensional data with mere preprocessing, unlike classic methods that involve complex preprocessing [1]. Deep learning-powered technology has influenced many aspects of our modern society. Language translation, text to speech, or speech to text systems are just small examples of applications based on deep neural networks that are widely used by millions of users.

The first successes of deep learning methods took place in the field of image processing [2]. Currently, the deep networks are used for many image processing tasks, such as image classification (e.g. classification of ImageNet dataset [3]), object detection (e.g. Yolo [4], Faster R-CNN (Region-based Convolutional Neural Networks) [5]), scene segmentation [6], depth estimation [7], and many more. They are deployed in many domains, such as autonomous cars [8], medical applications [9], surveillance [10], automatic content labeling on web services, image synthesis [11], [12], or data generation [13]. The number of applications is huge and still growing up. Deep learning-based methods highly outperform classical methods that require preparations of both the feature extraction algorithms and the classifier. In the classic approach, one needs to put effort to choose the proper methods, which involves having domain expertise and knowledge of many image processing algorithms. Moreover, those methods tend to have poor generalization abilities, for example, they are often not robust to light changing conditions, different size and positions of objects [14]. Deep learning methods have helped to overcome these problems and to achieve impressive results in image processing tasks in various conditions [1]. To handle these tasks, many architectures were proposed by the researchers in recent years.

The associate editor coordinating the review of this manuscript and approving it for publication was Donato Impedovo.

Over time, state-of-the-art structures have achieved better and better results. The most popular architectures include VGG (Visual Geometry Group) [15], DenseNet [16], ResNet [17], WideResnet [18], or Efficient Net [19]. The very first architectures were plain, chain-structured networks, consisting of many layers in a pipeline. Next, the researchers discovered that so-called skip connections are advantageous for more effective training and achieving better results. The next proposed architectures consisted of a much more parallel path of signals within the neural network such as in Inception or DenseNet networks. The progress in image processing was made not only due to better architecture design but also due to the advance in other areas such as optimization (Adam [20], SGDR [21]), regularization (batch normalization [22], dropout [23]), activations function (Leaky ReLU [24], Scaled Exponential-Regularized Linear Units (SERLU)[25]), and data augmentation (Cutout [26], style-transfer based image generation [27]).

The mentioned architectures are considered as universal structures and are used as the off-the-shelf solution in numerous problems. However, the price that needs to be paid for their universality is the excessively complex structure of the network and an overly large number of network parameters to be found. Such structures are usually difficult to train in cases when only a small dataset is available.

There are many regularization techniques to overcome the problem of small datasets. The most popular is to use transfer learning that enables using the knowledge gathered on another dataset (usually large Imagenet dataset) [28]. In this approach, the weights of another network trained on large datasets are used as the initial weights during the training on the target dataset. The results achieved by this method are satisfactory, but the problem is that these architectures are naturally predisposed to classification tasks with many classes. For example, they are adapted to distinguish between different classes such as dog breeds and household devices. However, in many practical tasks, there is no need for such complex feature extraction and more specific feature extractors are needed – for example, to classify medical images.

In order to tackle those problems, there is a need to design architectures that are better suited to solve a particular task. The structure design process should take into account the difficulty of the problem to be solved, the size of the dataset available, implementation constraints, learning time, and many more. Proper architecture selection is an error-prone and laborious process that involves intuition and expertise. Each new-designed network needs to be validated. Unfortunately, the training of a deep neural network lasts very long and therefore the number of architectures that can be examined is limited.

The field of research called Neural Architecture Search (NAS) [29] tries to remedy this problem. NAS is a family of methods that try to find, in an automated fashion, an optimal network structure with respect to the existing limitations and available dataset.

Early works on the automated design of deep neural structures achieved decent results but the accompanying computational cost was unacceptably high. The search process took thousands of GPU-hours, hence such algorithms were not suitable for practical use by average users [30], [31]. The appearance of new methods, such as network morphism operations [32] and gradient-based neural architecture search [33], allowed to significantly reduce the computational time of this process.

Many deep learning solutions are tested on benchmark datasets. Although it is good for the machine learning community, due to standardized testing workflow, there could be the problem that proposed solutions could overfit to datasets used in practice that are usually of much smaller size. Moreover, the problem with benchmarks is that they are well prepared i.e. they contain very large number of good-quality images that are equally distributed between classes. On the contrary, the raw, real-life datasets being in use in industry, business, economy, laboratories and so on, are often of different quality and scales, unbalanced, with noisy labels.

In this work, we deployed one of the neural architecture search methods to the non-benchmark, medical problem of malignant melanoma detection [34]. The applied dataset has features that are often met in practical solutions. The classes are not equally balanced, moreover examples from different classes are very similar, and their proper classification may be a problem even for skilled specialists. Because of that, the network has to have high accuracy and high generalization ability. The process of single network training takes a few hours, which makes a manual design of a new architecture difficult.

The remainder of the paper is organized as follows. The related works are overviewed in Section II, while in Section III, network morphism methods are presented that allow the expansion of networks. The description of the implemented search algorithm is given in Section IV. Section V brings a brief overview of the malignant melanoma detection task. The implementation details are provided in section VI. The results are discussed in Section VI and concluded in Section VII.

II. RELATED WORK

The research on neural architecture search had been conducted before the interest in deep learning emerged. Early systems involved random search, grid search [35], and evolutionary algorithms [36], [37] to find the proper architecture of a classic (shallow), fully connected neural network. With the growth of the size of neural networks, many difficulties have arisen due to long training time which limit the possibility of testing new architectures. Not only the training times become longer, but also the search space larger. With the grow of network structures, the number of hyperparameters describing the architecture increases significantly.

Each NAS framework can be described by three elements: search space, search method and performance evaluation strategy [29]. The search space defines which architecture

types could be found during the process. The search strategy defines the way the search space should be explored. The performance evaluation strategy defines the way the performance of the proposed network is estimated.

The assumed search space has a crucial impact on the search process. If the search space is too small it leads to poor performance, while on the other hand, if it is too big it could significantly extend the search time. The search space can be described by such factors as the number of layers, number of neurons within the layers, type of layers, activation function, etc. In most cases, the search space is conditional, which means that some hyperparameters have an influence on the total number of hyperparameters e.g. the increase of a number of layers will increase the number of hyperparameters describing those layers.

Two types of search space can be distinguished: the network-based search space, and the cell-based search space. The network-based approach explores the whole architecture, whereas the cell-based approach finds just the cells that are then stacked to solve a given problem [38]. The number of cells in the stack depends on the task being solved.

Currently, numerous search methods are used to explore the search space including random search, grid search [35], evolutionary algorithms [28], [35], Bayesian optimization [36], [37], reinforcement learning (RL), and the gradient methods [33], [42]–[44], which are nowadays gaining ground and popularity.

The performance evaluation strategy is the way in which the performance of the neural network is estimated. The simplest way is to train each network until convergence and then measure the validation accuracy. Although it provides an accurate estimate of the network architecture, this method is very time-consuming. Many methods have been proposed to speed up the process of network evaluation during the architecture search. Lower fidelity estimates involve network evaluation based on, for instance, limited training time [45], the limited size of dataset [46], or reduced size of photos in the dataset [47]. The learning curve extrapolation strategy allows accelerating the search process by rejecting structures at an early stage of training based on the prediction of their performance [48]. Another approach to speed up the structure performance estimation is to use weight inheritance or function preserving transformation [32], instead of training the network from scratch.

The NAS field has attracted a lot of attention in recent years. The popular approaches relied on either evolutionary algorithms or reinforcement learning. Authors of [30] proposed the reinforcement learning algorithm to design deep network architectures with the use of vast computational power. The method involves training of the Recurrent Neural Network (RNN) based on Long Short Term Memory (LSTM) [49] cells to generate the new architectures. The RNN is trained using the REINFORCE [50] algorithm. Each network proposed by the RNN controller is trained for 50 epochs on CIFAR10 dataset. The method is computationally expensive and involves the use of vast computational resources (800 GPUs). During the exploration of search space, the algorithm proposed and trained 12 800 architectures, in total. The computational complexity of the method makes it impossible to use in practice by individual researchers, academic research teams, and even small companies.

Authors of [31] applied an evolutionary approach to explore the architecture space for the task of CIFAR10 classification. That method evolves the population of models with organisms being the neural network structures. The evolutionary approach chooses individuals and performs mutations that alter their structure. The search algorithm runs for 10 days and utilizes 250 GPUs.

Although both methods achieve promising results on CIFAR10 dataset, their success relies on an extremely high computational demand. This is caused by the fact that those methods sample many architectures that are then trained from scratch. Therefore, there was a need for a method that allows incorporating the experience from previous training into new architectures.

One of the solutions to this problem is so-called network morphism [32], [51]. This is a family of methods that enable expanding a network by new elements (e.g. filters or layers) while preserving its performance. The new architecture gets the same accuracy, validation loss, or other selected statistical measures. The extended network has the same performance as the parent network but it has a bigger capacity. This means that the extended network has the ability to fit a wider variety of functions. As a result, the network can learn more complex relationships in the data during further training. This approach leads to significant decrease in the search time, as it utilizes the knowledge acquired during the training of previous network architecture to form new, more efficient architectures.

The authors of [52] propose a reinforcement learning framework that takes advantage of the network morphism operation to design the network structure. They use the reinforcement learning approach to train the RNN agent that explores the architecture space by applying network preserving transformations. The methods are limited to transformations that either insert a layer or make the layer wider. That approach allows designing only plain chain-structured networks. The utilization of knowledge from previously trained networks makes the search process more efficient compared with the methods in which the networks are trained from scratch. The methods allow for a huge reduction of computational requirements from a few thousand of GPU-days to only 5 GPU-days.

In their further work [53], the authors expanded the available network morphism operations to allow building multi-branch neural networks. The proposed function preserving operations allow replacing a single layer with a multi-branch motif. Then, those motifs can be further transformed by other modifications. The introduction of multi-branch structures leads to better performance and significant reduction in the number of parameters describing the network.

Our work has been inspired by [54], which uses the hill-climbing algorithm with the network morphism transformation to search for the architectures. The method enables designing multi-branch neural networks, but the network transformation operators differ from those used in [53]. The search algorithm applied in this work could be interpreted as a simple evolutionary algorithm with no crossover operation. It incrementally expands the initial network applying network morphism operations. The algorithm applies mutations (network morphism operation) to the best organism from the previous iteration. Then, each generated network is trained for small number of epochs. The best organism passes to the next generation as a parent and the process repeats. The proposed method allows finding competitive architectures in less than 1 GPU-day on CIFAR10 task. The hill-climbing algorithm can be interpreted as a simple evolutionary algorithm with only the mutation operator. Therefore, terms related to this field e.g. organism, offspring, or parent, are used further in the paper.

III. NETWORK MORPHISM

The network morphism is a family of methods that allow expanding the size of a neural network without loss of the acquired experience [32]. It can significantly accelerate the neural architecture search algorithm. Early NAS algorithms proposed a large number of architectures and trained them from scratch. It was a wasteful process, as the experience acquired during the training was not transferred to other networks trained later. In contrast, the network morphism enables transferring the knowledge from the original network to the bigger one by applying a special way of weights initialization in the extended network. This is realized by so-called function preserving transformations that produce network g(·) by modifying network f(·) in the way which satisfies the equation:

$$\forall x, \quad f(x) = g(x) \tag{1}$$

where x is the input of the network.

Through such an approach, the expanded network contains all of the experience acquired by the smaller one, but it has a higher capacity; therefore, its performance can be improved in further training. This type of initialization allows to reuse the information from the previously trained networks and save time.

The network morphism operation can be specified as follows [51]: given the original (initial) network represented by the function $y = f(x; \theta)$, where x is the input, y is the output, and θ represents the parameters of the network, the task is to choose a new set of parameters θ' for an extended network g such that:

$$\forall x, \quad f(x; \theta) = g(x; \theta') \tag{2}$$

In this work, we employ the layer addition operation (Net2WiderNet), the layer extension operation [32], and the operations that allow constructing multi-branch architectures [53].

Net2WiderNet enables to expand any layer by additional units (e.g. filters in a convolutional layer, or neurons in a fully connected layer). The method involves replicating weights of randomly chosen units from the layer being expanded. The number of weights of each unit in the next layer is increased to take into account units inserted in the previous layer. Moreover, the weights in the next layer are multiplied by the scaling factors, taking into account additional units in the previous layer in order to perform the same function as the original network.

Net2DeeperNet modification allows adding a layer anywhere within the network. It is realized by initializing the weights of the inserted layer as either the identity matrix or identity filters. Unlike the Net2WiderNet method, the weights of other layers are not modified. The newly added layer has to have the same size as the previous layer (e.g. the same number of filters).

To introduce branch connections during NAS algorithm operation, the methods described in [53] were used. Nowadays, multi-branch neural networks are extensively used in the deep learning community. The most popular architectures that use this kind of connections include Inception, Resnet, and Densenet. In those networks, the signal is often distributed to several branches. Each branch conducts some computations, then the signals from the branches are aggregated using one of the merge schemes (add or concatenation).

The function preserving operations can transform each layer into an equivalent multi-branch motif with either add or concatenation merge scheme.

To transform a layer C(·) to the equivalent motif with add merge scheme, the layer has to be replicated. To preserve the function performed by a single layer, the outputs of the original and replicated layers have to be scaled by factor 0.5 before adding, that yields:

$$0.5 * C(x) + 0.5 * C(x) = C(x) \tag{3}$$

which gives the same result as from the original layer.

To transform the layer in an equivalent motif with concatenation merge scheme, the layer is split into two parts. Each part contains half of the units. The branches are joined together using concatenation that results in the operation that is equivalent to that performed by the layer before modification.

Although after modification, those multi-branch motifs act as a single layer, those structures could be further expanded by modifications inside the motifs e.g. by inserting a layer in one of the branches and making that layer wider. It is also possible to create other branched motifs inside the existing ones. The way the add and concatenation merge schemes are constructed is outlined in Fig. 1.

The skip connections that are a special type of multi-branch motifs are constructed by using Net2DeeperNet and the add merge scheme. We propose the method that first adds a new layer, and next adds the skip connection to this layer in a similar way as in the add merge scheme. The output of the

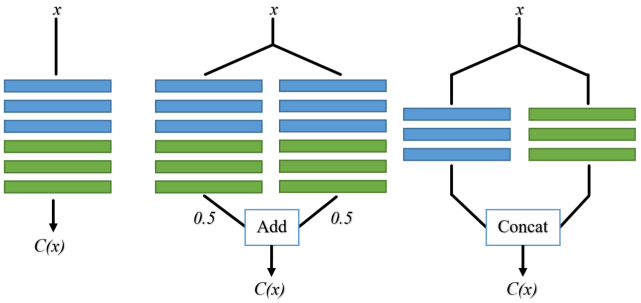

FIGURE 1. Construction of the add and concatenation motifs that are equivalent to a single layer.

identity layer and the skip connection are summed up with 0.5 weights.

IV. SEARCH ALGORITHM

This section provides the general idea of the method we used, while the implementation details are provided in Section VI. We make use of the NAS approach similar to [54]. We apply the same search strategy, but with different kind of network morphism operations. Moreover, our approach is used to search for networks that process much bigger images $-224 \times 224 \times 3$ instead of $32 \times 32 \times 3$ processed in the cited paper. The implemented search strategy is based on a greedy algorithm called the hill-climbing algorithm. The algorithm allows faster exploration of the search space than a genetic algorithm does. Moreover, the algorithm does not use crossover operation, that could be very difficult to apply in our approach. The scheme of this algorithm is shown in Fig. 2.

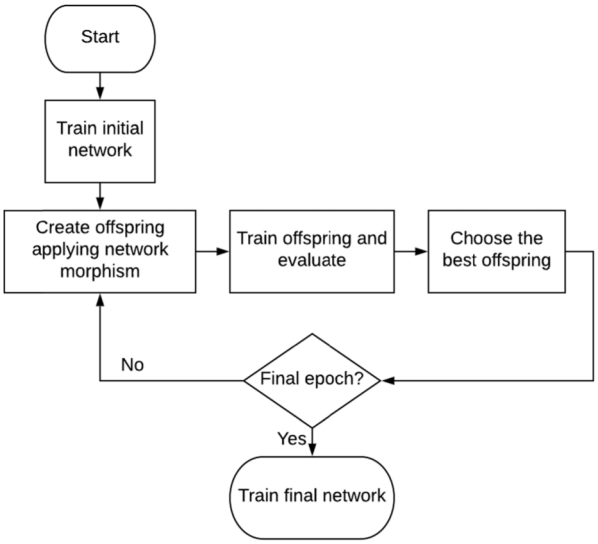

FIGURE 2. Scheme of the search algorithm.

The first step of the framework is to train a small network for a certain number of epochs. Then, this network becomes the parent in the first iteration of the hill-climbing algorithm. At the beginning of each iteration, offspring are produced by applying one or more function-preserving operations to extend the structure of the parent. Next, each offspring is trained for a certain number of epochs. Due to the network morphism, training of each offspring is not performed from scratch, but it begins from a point at which the parent finishes its training. For example, if the parent finishes training with 70% accuracy, each offspring starts from the point when it already has 70% accuracy. The training enables to exploit an increased capacity resulted from introducing extra elements to the structure.

The following modifications are applied during the search process:

- inserting a layer [32],
- adding filters to the existing layer [32],
- adding the add merge scheme [53],
- adding the skip merge scheme,
- adding the concatenation merge scheme [53].

For every iteration, the function preserving operations are sampled by the algorithm. Then it draws the place where the operation has to be performed e.g. which layer to expand or where to place the additional layer. At the end, the final network proposed by the algorithm undergoes further training.

V. CASE STUDY

We decided to evaluate the neural architecture search algorithm on the task of skin lesion classification. This task involves distinguishing between benign and malignant skin lesions.

The classical method of skin cancer detection involves the examination of a skin lesion by a skilled specialist. The decision of whether the lesion is benign or malignant is made based on specific properties of the lesion such as symmetry, border, color, and differential structures [55], [56]. The main problem is that the rules how to classify skin lesions are not precisely defined. This means that the distinction between benign and malignant lesions is very ambiguous and may lead to different diagnoses given by different physicians. Such inconsistency in diagnosis makes the problem of automatic classification much more difficult than in standard tasks where people have almost perfect accuracy, for instance when distinguishing between cars and planes.

In other words, the difficulty of the task comes from high intra-class variance and high inter-class similarity. That means the examples from two different classes can be very similar to each other. Note the similarity between the two lesions in Fig. 3. These lesions look very similar, but they represent two different classes – benign (top) and malignant (bottom). In fact, the benign lesion can attract the attention of an inexperienced patient, as it meets the rules of a malignant lesion, such as not sharp and asymmetric border, visible structures inside the lesion, and inconsistent color.

The publicly available dataset is provided by the International Skin Imaging Collaboration [34]. It consists of high-quality dermoscopic images collected from clinics across Europe, Australia and the United States, acquired from patients of various age and sex. The images are annotated by

FIGURE 3. Examples of benign (top) and malignant (bottom) lesions.

high-skilled experts into benign and malignant moles. The dataset consists of about 12500 benign instances and only 1100 malignant instances. This disproportion between the classes makes proper training of classification systems more difficult. Moreover, the small number of examples leads to a smaller validation set and the resulting noisy estimate of neural network performance.

The abovementioned issues make the present problem differs from the problems tested on benchmark dataset in which a large number of equally distributed examples are almost always provided.

VI. IMPLEMENTATION DETAILS

A. DATA PREPROCESSING

The data preprocessing is performed the same way as in our previous work [57]. Using the provided masks, the lesions were extracted from the pictures. In order to improve the training process, the dataset was normalized to obtain zero mean and unit variance. The size of the images was changed to 224 × 224px to fit the input of the neural network.

The standard train-validation-test scheme was applied. The division was performed randomly. Both the test set and the validation set contained 200 examples, with 100 examples per class. The remaining part of the dataset became the training data. To equalize the number of examples of each class in the training set, the upsampling was performed.

The dataset was augmented by numerous modifications, such as rotation, width and height shift, horizontal and vertical flip, and zooming. The data augmentation was applied online during the training, before passing the images to the neural network input.

B. BASELINE

The hand-crafted networks presented in our previous research [57] were used as a baseline to compare the NAS method. Although the results described in that paper were evaluated using 5-fold cross-validation, in this work we decided to evaluate only one-fold, as the task was to compare automated against manual architecture design and this comparison can be effectively performed on one-fold. Moreover, 5-fold validation leads to a significant increase in the algorithm runtime. We tested 6 manually designed architectures from the family of VGG networks (VGG8, VGG11, VGG16), with different regularization methods applied (transfer learning, dropout, batch normalization). The networks are enumerated in Table 1, while their detailed description is provided in [57].

TABLE 1. Manually designed architectures of networks – The reference networks for testing the nas algorithm.

Network	A	B	C	D	E	F
Network	VGG 8			VGG 11	VGG 16	
Transfer learning						X
Dropout		X		X	X	X
Batch normalization			X			

C. THE NEURAL ARCHITECTURE SEARCH SYSTEM

The neural architecture search system involves the following three steps: training of the initial network, the search process, and training of the generated architecture. All the training processes are performed using Stochastic Gradient Descent with Warm Restarts (SGDR) [21]. This method involves cosinusoidal decay of the learning rate, and restart after every chosen number of epochs. The batch size was set to 8 and remained the same through all steps of the framework.

In order to reduce the computational demands of the algorithm, the advantage was taken of the lower fidelity estimates based on training on the one-third of the dataset. The initial network training and the search process were performed on the fraction of the dataset, whereas the final training was performed on the full dataset.

1) INITIAL NETWORK

The initial network has a plain, chain-structured feedforward architecture. That is Input – 5 * [Conv128 – MP] – Conv128 – Sigmoid. The images of size 224 × 224 × 3 are provided to the input. Conv stands for a convolutional layer, and the number is the number of filters. Each convolutional layer is followed by batch normalization [22] and ReLU layers,

which are omitted here for better readability. Each convolutional layer has a 3×3 filter size and stride set to 1. In order to preserve the size of the output feature maps padding is applied. The L2 regularization is applied to each layer with the value of 0.0005. MP stands for MaxPooling layer, with a 2×2 sliding window with stride set to 2. The initial network was trained with standard binary cross-entropy loss, for 60 epochs. The learning rate of the SGDR optimizer decays cosinusoidally from 0.01 to 0, with restarts after every 10 epochs.

2) NAS ALGORITHM

In this study, the word epoch is used to refer to the training of the neural network, while the word iteration refers to the iterations in the hill-climbing algorithm.

The architecture exploration starts after the initial network is trained. The hill-climbing algorithm runs for 15 iterations. It is a compromise between the quality of network performance, the size of the gained network, i.e. the number of its parameters and the time of algorithm running. More iterations do not increase the effectiveness of the classification but do increase the number of network parameters. For example, 10 iterations of the algorithm resulted in 1.489M parameters of the network; 15 iterations −1.932M; 20 iterations −2.372M and 25 iterations resulted in 5.327M parameters of the network. At each iteration, five offspring are created by applying two random function preserving operations to the parent structure. Furthermore, the parent structure passes, without modification, to the next iteration as the sixth offspring.

Each network preserving operation operates on 3×3 filters and with 0.0005 weight decay applied in the first training. The Net2WiderNet method doubles the number of filters in the layer. At every iteration, each organism is trained for 8 epochs, with the SGDR algorithm with the learning rate decayed from 0.005 to 0. At the end of an iteration, each offspring is evaluated and the best organism passes to the next epoch. If there is no improvement in the next iteration the parent does not change.

3) FINAL TRAINING

The best model from the last iteration of the hill-climbing algorithm is trained for a longer period of time. The training is performed for 200 epochs, with the SGDR algorithm with the learning rate being decayed from 0.005 to 0 after every 25 epochs.

VII. EXPERIMENTAL RESULTS

We performed several experiments to show the efficiency of the proposed approach. All results presented in the tables were obtained by evaluating the networks on the test set. Our first attempt was to perform validation accuracy based neural architecture search. The offspring selected as the parent in the next iteration was the one with the best accuracy (ACC) score on the validation set. In order to avoid statistical uncertainty of the achieved results, the algorithm was run four times and then the average values were taken. The results obtained are presented in Table 2.

TABLE 2. The search based on validation accuracy.

Run	ACC	AUC	Parameters
1 run	72.0	0.792	1 527 873
2 run	70.0	0.779	2 228 609
3 run	72.5	0.805	2 083 841
4 run	74.5	0.827	1 342 593
AVG	72.25	0.801	1 795 729

FIGURE 4. Comparing two search approaches: by validation loss and by validation accuracy.

During the experiments, we frequently observed the situation where more than one organism achieved the same validation accuracy. Moreover, the model that achieved a much better validation loss (binary crossentropy) often did not become the parent network, because it achieved worse validation accuracy score.

Although in many situations model selection is performed based on the validation accuracy score, it is not a feasible method in problems where a small validation set is provided. The accuracy measure is a discrete value that can take on a finite set of values. The number of elements of the set is equal to the number of examples in the validation set. Therefore, a lower number of validation examples may lead to situations when more than one model achieves the same validation accuracy. This, in turn, causes problems with selecting the best one. For the reasons stated above, we decided to select the best offspring based on the binary cross-entropy validation loss. The validation loss is a continuous value. Moreover, it carries much more information about the network performance than the validation accuracy, which only informs on a fraction of time the classification was correct. The validation loss takes into account also the certainty of prediction. During the training and architecture search, if the validation accuracy saturates, further training and the decrease of validation loss push the classes apart from each other, which results in

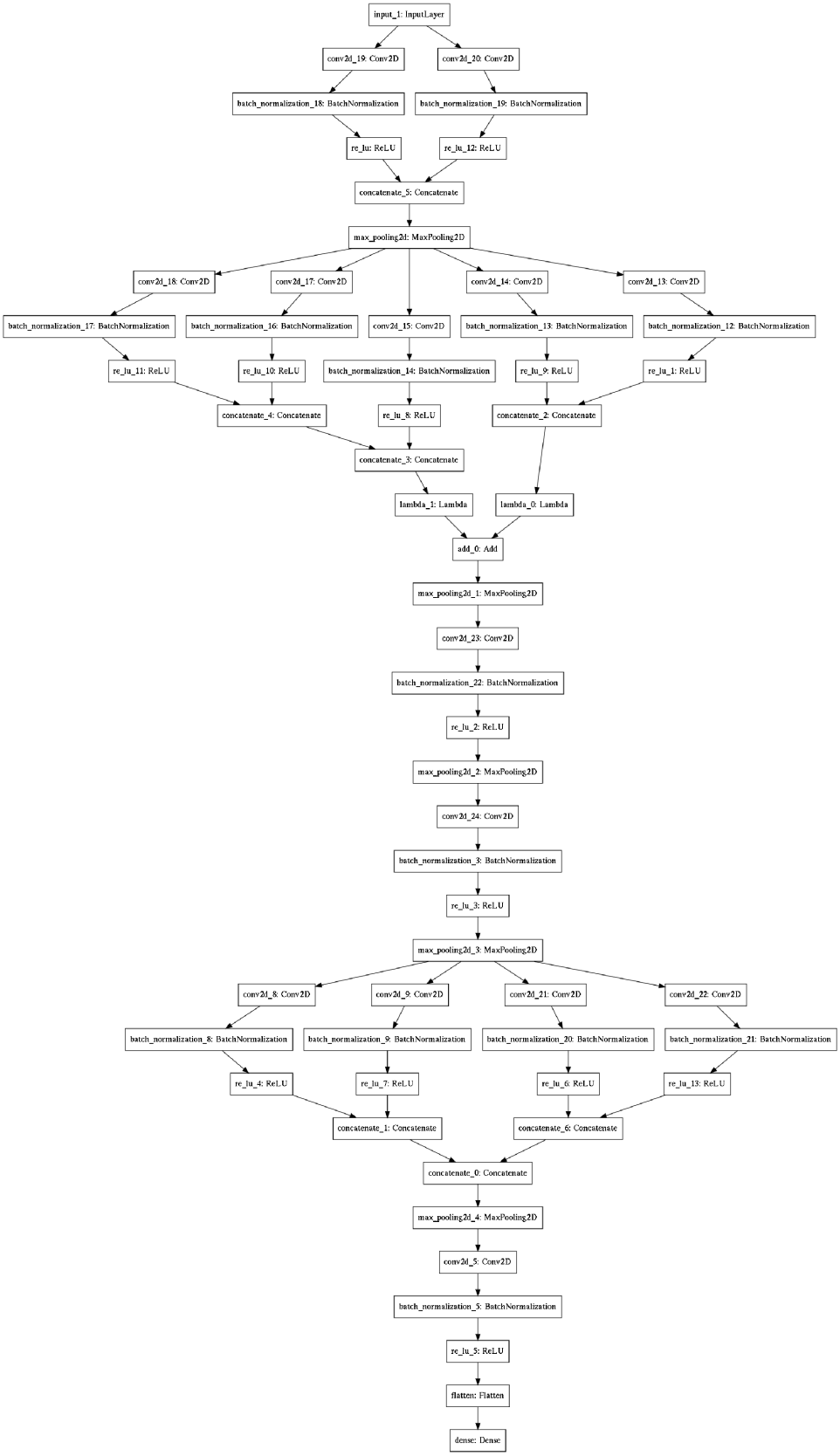

FIGURE 5. An example of generated network.

a more confident and reliable classifier [58]. Whereas this phenomenon is not a serious issue in tasks where big datasets are provided, it causes a problem when only small datasets are available. To illustrate the problem, the validation loss-based search was compared with that based on accuracy. The progress of these two search processes is shown in Fig. 4.

Note that the search by validation loss leads to a steady decrease of both the validation loss and the validation error. Whereas, in the case of search by accuracy, the validation accuracy decreases while the validation loss grows. Higher validation loss can lead to less robust classifier and lower AUC (Area Under Curve) score.

Based on that, we performed the NAS based on the validation loss. This approach leads to higher performance, as reported in Table 3.

TABLE 3. The search based on validation loss.

Run	ACC	AUC	Parameters
1 run	76.5	0.825	1 198 593
2 run	74.0	0.812	1 785 601
3 run	77.0	0.823	1 054 593
4 run	78.0	0.845	750 849
AVG	76.38	0.826	1 197 409

TABLE 4. Summary of the results.

Model	ACC	AUC	Parameters
Network A	70.58	0.803	30 652 545
Network B	74.75	0.827	30 652 545
Network C	75.08	0.841	30 659 587
Network D	69.42	0.780	35 971 843
Network E	67.83	0.748	41 456 449
Network F	75.75	0.847	41 456 449
Search based on loss	76.38	0.826	1 197 409
Search based on accuracy	72.63	0.810	1 795 729
Search based on loss - ensemble of 4 networks	77.00	0.843	7 182 916

The summarized results of human-designed and automatically designed networks are presented in Table 4. The architectures generated by the NAS algorithm have similar performance as those designed manually. Hoverer, the NAS algorithm generates architectures with significantly fewer parameters compared with those designed manually. What is important, the searching process was completed in a very short time, that on average took around 18 GPU-hours. Short searching time was achieved as a result of the applied method, but also due to the low fidelity performance estimation. The search based on the whole dataset yields the same result as those performed only on one-third of the dataset. We would like to emphasize that the average runtime of an algorithm is

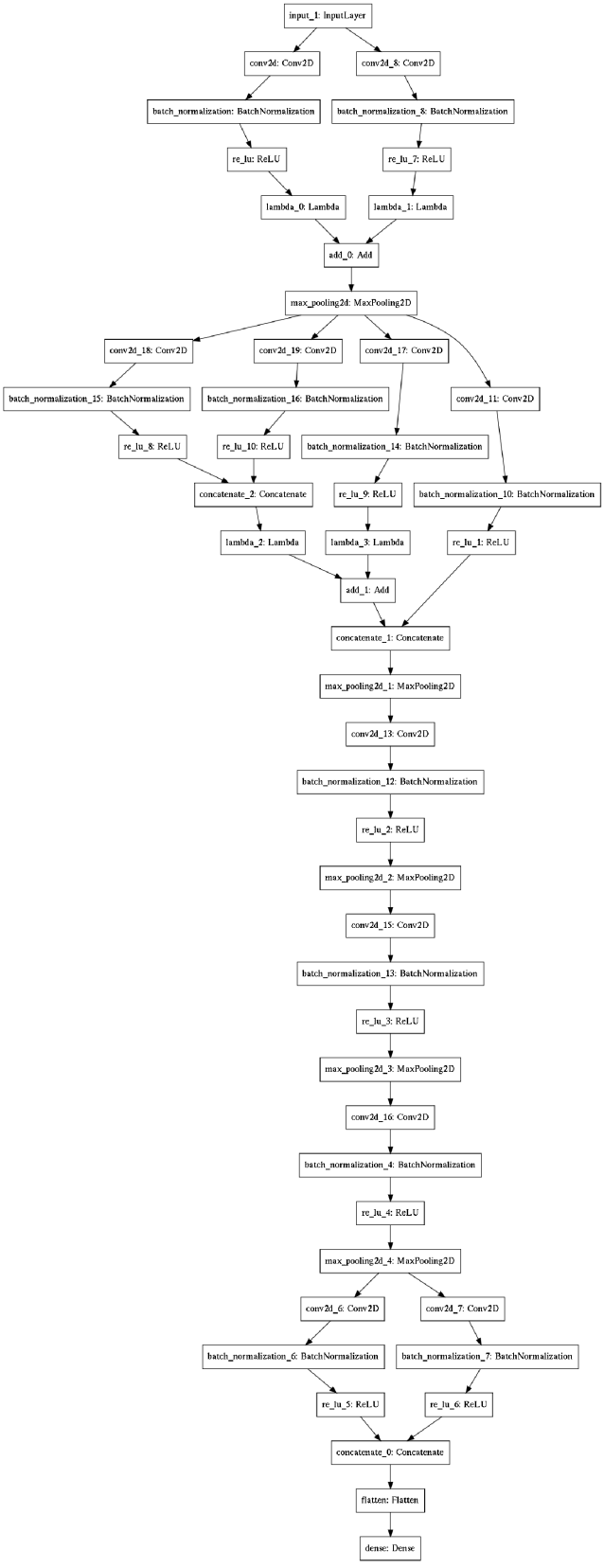

FIGURE 6. An example of generated structure.

very short compared with the training time of a single neural network that takes on average a few hours, not including the time to decide on the network structure. Examples of the generated architectures are presented in an Appendix.

We also applied the networks ensemble as it is a simple and cheap method to improve the performance of the classification algorithm. Since the search process was performed four times, the networks were already trained. The network ensemble improved the performance, both in accuracy and AUC score, achieving 77% accuracy and 0.843 AUC score. Note, that the sum of parameters of the networks composing the ensemble is still far less than the number of parameters designed manually.

VIII. CONCLUSION

In this paper, we presented the neural architecture search approach applied to designing a structure that solves the challenging task of skin lesion classification.

The deployment of the hill-climbing algorithm with function preserving modifications leads to competitive results. The network preserving transformations take advantage of the previously trained networks by reusing the weights from the previous training, which leads to significant computational cost reduction. Our experiments have shown that this approach enables producing structures with satisfactory performance.

We analyzed and showed that for small datasets, searching the structure with the performance function based on the validation loss leads to finding the efficient models while the searching based on validation accuracy worsen the efficiency of the architecture search process.

The networks generated by the algorithm perform as well as those designed manually, however, they have about 20 times fewer parameters. The search process is only few times longer than the training of a single network. However, note that the time of deployment of an effective architecture is not limited only to single training but to many pieces of training during a tedious try and error process.

In order to make use of the networks generated during experiments, we applied an ensemble of a generated network, yielding even better classification results.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 2012, pp. 1106–1114.
[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009.
[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788.
[5] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[6] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," Dec. 2018, *arXiv:1812.01593*. [Online]. Available: https://arxiv.org/abs/1812.01593
[7] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6612–6619.
[8] Z. Chen and X. Huang, "End-to-end learning for lane keeping of self-driving cars," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1856–1860.
[9] C. Barata, M. E. Celebi, and J. S. Marques, "A survey of feature extraction in dermoscopy image analysis of skin cancer," *IEEE J. Biomed. Health Inform.*, vol. 23, no. 3, pp. 1096–1109, May 2019.
[10] X. Liu, W. Liu, T. Mei, and H. Ma, "A deep learning-based approach to progressive vehicle re-identification for urban surveillance," in *Proc. ECCV*, 2016, pp. 869–884.
[11] A. Mikolajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *Proc. Int. Interdiscipl. PhD Workshop (IIPhDW)*, May 2018, pp. 117–122.
[12] L. Gatys, A. Ecker, and M. Bethge, "A neural algorithm of artistic style," *J. Vis.*, vol. 16, no. 12, p. 326, Sep. 2016.
[13] X. Yi, E. Walia, and P. Babyn, "Generative adversarial network in medical imaging: A review," *Med. Image Anal.*, vol. 58, Dec. 2019, Art. no. 101552.
[14] M. Pietikäinen, A. Hadid, G. Zhao, and T. Ahonen, *Computer Vision Using Local Binary Patterns*. London, U.K.: Springer-Verlag, 2011.
[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–10.
[16] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.
[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[18] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, York, U.K., Sep. 2016, pp. 1–15.
[19] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Long Beach, CA, USA, 2019, pp. 6105–6114.
[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–15.
[21] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017.
[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, Lille, France, vol. 37, Jul. 2015, pp. 448–456.
[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jan. 2014.
[24] A. L. Maas, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013, vol. 30, no. 1, p. 3.
[25] G. Zhang and H. Li, "Effectiveness of scaled exponentially-regularized linear units (SERLUs)," Jul. 2018, *arXiv:1807.10117*. [Online]. Available: https://arxiv.org/abs/1807.10117
[26] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," Aug. 2017, *arXiv:1708.04552*. [Online]. Available: https://arxiv.org/abs/1708.04552
[27] A. Mikolajczyk and M. Grochowski, "Style transfer-based image synthesis as an efficient regularization technique in deep learning," in *Proc. 24th Int. Conf. Methods Models Autom. Robot. (MMAR)*, Aug. 2019, pp. 42–47.
[28] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proc. 27th Int. Conf. Artif. Neural Netw. (ICANN)*, Rhodes, Greece, vol. 11141, Oct. 2018, pp. 270–279.
[29] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, pp. 55:1-55:21, Mar. 2019.
[30] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–37.
[31] E. Real, "Large-scale evolution of image classifiers," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, Sydney, NSW, Australia, vol. 70, Aug. 2017, pp. 2902–2911.

[32] T. Chen, I. J. Goodfellow, and J. Shlens, "Net2Net: Accelerating learning via knowledge transfer," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, San Juan, Puerto Rico, May 2016, pp. 1–12.
[33] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–13.
[34] *ISIC Archive*. Accessed: Sep. 24, 2019. [Online]. Available: https://www.isic-archive.com
[35] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.
[36] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, Jun. 2002.
[37] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: From architectures to learning," *Evol. Intel.*, vol. 1, no. 1, pp. 47–62, Mar. 2008.
[38] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 8697–8710.
[39] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI*, vol. 33, pp. 4780–4789, Sep. 2019.
[40] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, "Towards automatically-tuned neural networks," in *Proc. 33rd Workshop Autom. Mach. Learn. (AutoML) 2016, Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, Jun. 2016, vol. 64, pp. 58–65.
[41] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, Atlanta, GA, USA, Jun. 2013, vol. 28, pp. 115–123.
[42] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–13.
[43] Y. Weng, T. Zhou, L. Liu, and C. Xia, "Automatic convolutional neural architecture search for image classification under different scenes," *IEEE Access*, vol. 7, pp. 38495–38506, 2019.
[44] Y. Weng, T. Zhou, Y. Li, and X. Qiu, "NAS-Unet: Neural architecture search for medical image segmentation," *IEEE Access*, vol. 7, pp. 44247–44257, 2019.
[45] A. Zela, A. Klein, S. Falkner, and F. Hutter, "Towards automated deep learning: Efficient joint neural architecture and hyperparameter search," Jul. 2018, *arXiv:1807.06906*. [Online]. Available: https://arxiv.org/abs/1807.06906
[46] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast Bayesian optimization of machine learning hyperparameters on large datasets," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Fort Lauderdale, FL, USA, vol. 54, Apr. 2017, pp. 528–536.
[47] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of ImageNet as an alternative to the CIFAR datasets," Jul. 2017, *arXiv:1707.08819*. [Online]. Available: https://arxiv.org/abs/1707.08819
[48] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–14.
[49] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[50] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.
[51] T. Wei, C. Wang, Y. Rui, and C. W. Chen, "Network morphism," in *Proc. 33nd Int. Conf. Mach. Learn. (ICML)*, New York City, NY, USA, vol. 48, Jun. 2016, pp. 564–572.
[52] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. 32nd AAAI Conf. Artif. Intell., (AAAI), 30th Innov. Appl. Artif. Intell. (IAAI), 8th AAAI Symp. Educ. Adv. Artif. Intell. (EAAI)*, New Orleans, LA, USA, Feb. 2018, pp. 2787–2794.
[53] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu, "Path-level network transformation for efficient architecture search," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, vol. 80, Jul. 2018, pp. 677–686.
[54] T. Elsken, J. H. Metzen, and F. Hutter, "Simple and efficient architecture search for convolutional neural networks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–14.
[55] F. Nachbar, W. Stolz, T. Merkle, A. B. Cognetta, T. Vogt, M. Landthaler, P. Bilek, O. Braun-Falco, and G. Plewig, "The ABCD rule of dermatoscopy: High prospective value in the diagnosis of doubtful melanocytic skin lesions," *J. Amer. Acad. Dermatol.*, vol. 30, no. 4, pp. 551–559, Apr. 1994.
[56] A. Mikołajczyk, A. Kwasigroch, and M. Grochowski, "Intelligent system supporting diagnosis of malignant melanoma," in *Proc. Polish Control Conf.*, 2017, pp. 828–837.
[57] M. Grochowski, A. Kwasigroch, and A. Mikołajczyk, "Selected technical issues of deep neural networks for image classification purposes," *Bull. Polish Acad. Sci., Tech. Sci.*, vol. 67, no. no. 2, pp. 363–376, 2019.
[58] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.