Heuristic Architecture Search Using Network Morphism for Chest X-Ray Classification

Pavlo Radiuk[1] [0000-0003-3609-112X] and Hakan Kutucu[2] [0000-0001-7144-7246]

[1] Khmelnytskyi National University, 11, Instytuts'ka str., Khmelnytskyi, 29016, Ukraine
[2] Karabük University, Kılavuzlar/Karabük Merkez/Karabük, 78050, Turkey

[1]radiukpavlo@gmail.com, [2]yukselcelik@karabuk.edu.tr

Abstract. Nowadays, the demand for medical image computing is exceptionally high. This growth was mostly driven by the manual development of machine learning models, in particular neural networks. However, due to the constant evolution of domain requirements, manual model development has become insufficient. The present study proposes a heuristic architecture search that can be in an excellent service for the task of medical image classification. We implemented a novel approach called network morphism to the search algorithm. The proposed search method utilizes the enforced hill-climbing algorithm and functional-saving modifications. As a result of computational experiments, the search method found the optimal architecture in 28 GPU hours. The model formed by the found architecture achieved performance of 73.2% in validation accuracy and 84.5% in AUC on the validation dataset that is competitive to the state-of-the-art hand-crafted networks. Moreover, the proposed search method managed to find the architecture that contains four times fewer parameters. Besides, the model requires almost ten times less physical memory, which may indicate the practical usefulness of our method in medical image analysis.

Keywords: heuristic search, neural architecture search, network morphism, medical image classification, Chest X-Ray

1 Introduction

Over the past decade, there has been a dramatic increase in the manual development of deep convolutional neural network (DCNN) architectures that show significant results in image processing. The most recognized hand-crafted DCCNs are VGG [1], ResNet [2], and DenseNet [3]. These networks were customized to solve multi-classification tasks, e.g., the Imagenet classification benchmark [4] with 1000 classes.

In contrast, practical tasks usually require models that can perform well on various data, either small or large images, with two or many classes, based on numerous or few training data. For instance, state-of-the-art architectures [1–3] show excellent performance in image classification tasks with many classes. However, these networks contain a significant number of parameters that make them heavy in terms of memory and training time and, therefore, redundant for many practical tasks. Moreo-

ver, different cases may involve various features in the implemented models. Medical image processing is a representative example of a real challenge that requires both flexibility and reliability in classification tasks.

Automated machine learning (AutoML) and its branch neural architecture search (NAS) might be an excellent solution to the mentioned-above issues. To date, AutoML and NAS methods have been actively developed and employed in various fields. For instance, NAS allows deploying neural networks into classification and segmentation tasks with little domain expertise and small datasets. The most recognized NAS approaches employ search strategies based on genetic algorithms [5], reinforcement learning techniques [6], Bayesian optimization [7], gradient-based methods [8]. According to recent reviews [9,10], the most promising in terms of efficiency are architecture search methods based on genetic algorithms.

Despite significant advances in NAS, its impact on the practical tasks in healthcare is yet not clear. There is still much uncertainty about the efficiency of NAS approaches in medical screening. To address this issue, the present study examines the application of automated search methods to find optimal neural architecture medical image classification.

2 Related Works

Genetic or so-called evolutionary algorithms have significantly advanced and expanded the use of AutoML. In the field of NAS, genetic algorithms are designed to construct new sets of neural architectures, called population [11]. The algorithm starts with establishing underlying networks, either random or predefined. Each structure is then trained and evaluated based on the requirements of a particular task, such as image classification or segmentation. After that, the most suitable network may serve as a parent for further search. At the following stage, the algorithm creates offspring by implementing modifications (mutations) in the structure of the parent network. The algorithm ends when new adjustments cannot be applied to an offspring as it reaches the best performance in the task.

Evolutionary algorithms have become particularly popular due to the use of a vast search space, which has considerably improved NAS techniques. Even though genetic algorithms allow achieving decent results in NAS [9], neural models based on evolutionary search require to train an enormous number of architectures [12] that lead to significant time expenses [13]. Also, many real-world applications cannot afford high computational cost due to technical limitations. Thus, the development of methods that would be effective at computational and time constraints is a relevant task at present.

Over the past years, researchers have proposed various methods to solve primary drawbacks of any NAS methods: significant time costs and the considerable weight of an output network. For example, Veniat et al. [14] applied the gradient descent method to the budgeted learning function, which includes the maximum allowable cost. They could train a neural network capable of predicting well in less than 100 milliseconds on both CIFAR10 and CIFAR100 datasets. In [15], Li et al. presented a prun-

ing method to cut search space by including profile information about the output speed on the target dataset. This approach was able to provide an automated architecture search with an excellent compromise in speed and accuracy on the ImageNet dataset. In [16], Tan et al. suggested a scaling technique that uniformly expanded all dimensions using a compound coefficient. Also, the authors rescaled MobileNets and ResNet up to obtain a family of efficient models. This approach showed decent results in the relevant transfer training stands.

Other researchers moved in a slightly different direction. In [17], Wei et al. systemized preserved functions and introduced a set of parametric operations that could enhance the morphing of any continuous nonlinear activation neurons. Their approach showed decent results on the CIFAR10 and CIFAR100 datasets. In [18], Elsken et al. evolved network morphism by combining it with a simple hill-climbing search algorithm. Furthermore, the authors conducted optimization runs by cosine annealing after each operation. This method achieved a stunning 94% validation accuracy in only 12 hours on a single GPU on the CIFAR10 dataset. Gordon et al. [19] presented a novel framework called MorphNet that could iteratively shrink and expand a neural network. Their method was adaptable to specific resource constraints and could improve network performance on different data sets.

Several studies have addressed the efficiency of NAS on different medical datasets. Gessert et al. [20] proposed an efficient NAS by subsequently transferring low-dimensional data to high-dimensional one for OCT image segmentation. The authors achieved an 87.5% reduction in search time on one-dimensional data, compared to two-dimensional data. In [21], the authors proposed a NAS framework based on particle swarm optimization technique that could temporally evolve and finally converged to a feasible optimal architecture. The framework ensured robust architecture design having been trained on a single GPU card and tested on the volumetric fMRI data. Kwasigroch et al. [22] employed network morphism operations to the evolution strategy in order to diversify the exploring network without reducing validation accuracy. Such an approach provided a significant reduction in computational cost on the skin lesion dataset.

From the analysis of the literature, we assumed that network morphism could be an excellent solution to reduce the running time of the search method and facilitate the output model. Therefore, in this work, we apply network morphism to a genetic search for Chest X-ray classification.

3 The Problem Statement

The main goal of the present research is to investigate the efficiency of the network morphism approach in the medical classification task. To achieve the goal, the following tasks are due to be resolved:

1. To consider network morphism and adjust it to a genetic algorithm.
2. To select an appropriate dataset of medical images for the multiclassification task.
3. To construct a CNN as a baseline architecture and investigate its impact on the output architecture.

4. To implement a heuristic algorithm based on network morphism to search for an optimal neural architecture.
5. To evaluate an optimal architecture found automatically with objective metrics and compare it with state-of-the-art manual networks on the medical image dataset.

In this paper, we address the issue of searching for the optimal neural network architecture for the target data set $D = \{D_{tran}, D_{test}, D_{val}\}$, where D_{tran} is a training dataset, D_{test} stands for a test dataset, and D_{val} is a validation dataset. Let us consider the target problem as a two-level multipurpose optimization problem. The function of multiobjective bilevel optimization can be presented in a general form as follows:

$$a_{opt} = \min_{A \in \mathbf{A}} \left\{ L_{val}\left(A, w^*\right), C\left(A\right) \right\} \tag{1}$$

subject to

$$w^* \in \arg\min_{w \in \mathbf{W}} \left\{ L_{tran}\left(w, a\right) \right\},$$

where a_{opt} represents the final architecture among optimized ones A from the search space of all possible architectures of $\mathbf{A}$, $C\left(A\right)$ is the function of the complexity of the optimized architecture and other related values, w stands for the weights of the neural network from the weight space of $\mathbf{W}$, L_{tran} and L_{val} are loss functions on D_{tran} and D_{val}, respectively.

Below, we describe the proposed NAS method that was designed and assessed step by step by the conceptual framework from [23].

4 Heuristic Architecture Search Algorithm

In this study, we investigate a heuristic architecture search inspired by network morphism operators proposed in [18]. The morphism allows modifying a neural network without losing the obtained information about network structure and its hyperparameters. Created by the use of morphism, neural architectures can achieve performance measures as their predecessors but concurrently show encouraging computational capabilities. Fig. 1 illustrates the idea behind network morphism.

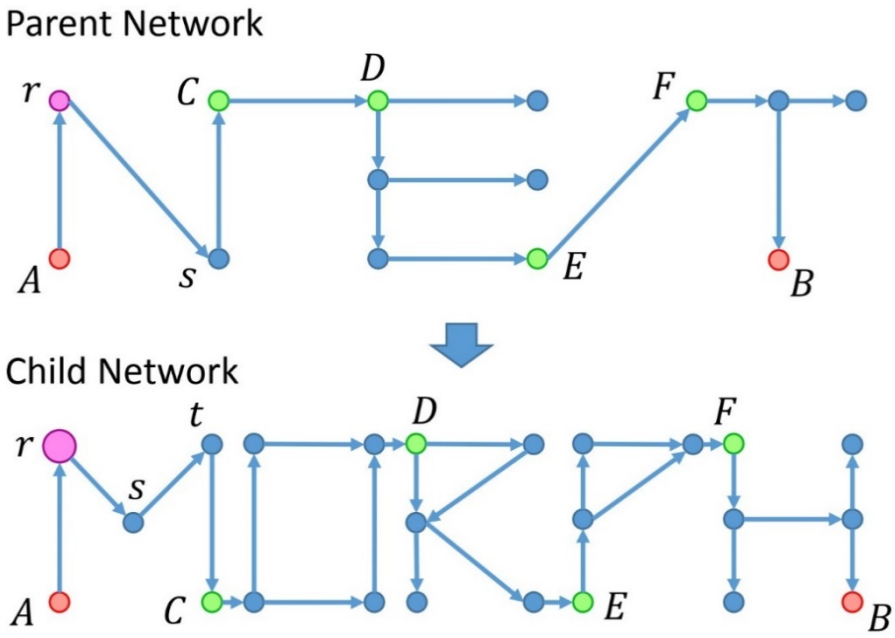

Fig. 1. The procedure of network morphism [17]. According to the algorithm, the child network will inherit all knowledge (A–F) from the parent network while maintaining the network function. A combination of morphing modifications can provide diverse network morphism.

In contrast, our paper applied a morphism algorithm to the classification task with large medical images. In this regard, we implemented various functional-saving operations into the algorithm to achieve the best training performance.

4.1 Functional-Saving Modifications

Network morphism allows avoiding training of each new network from scratch. Thereby, researchers can test many models in a short time. In this study, we applied several modifications to ensure model efficiency. Concatenation operations and add nodes serve as in [18]. Convolutional layers were expanded by the morphism functions, according to [19]. However, in contrast with previous studies, we implemented a combination of an additional operational layer with an extra node to provide skip connection in training. Also, we applied a new modification by inputting noise into the weights of the newly generated network after all previously mentioned modifications. Besides the introduced operations, symmetry disruption in the neural network parameter sequence can also lead to significant performance improvements in the NAS algorithm.

4.2 The Enforced Hill-Climbing Search

In this section, we briefly describe a heuristic search that executes the task (1). As a search strategy, we utilized a genetic algorithm called the enforced hill-climbing search [23]. Fig. 2 depicts the scheme of the search.

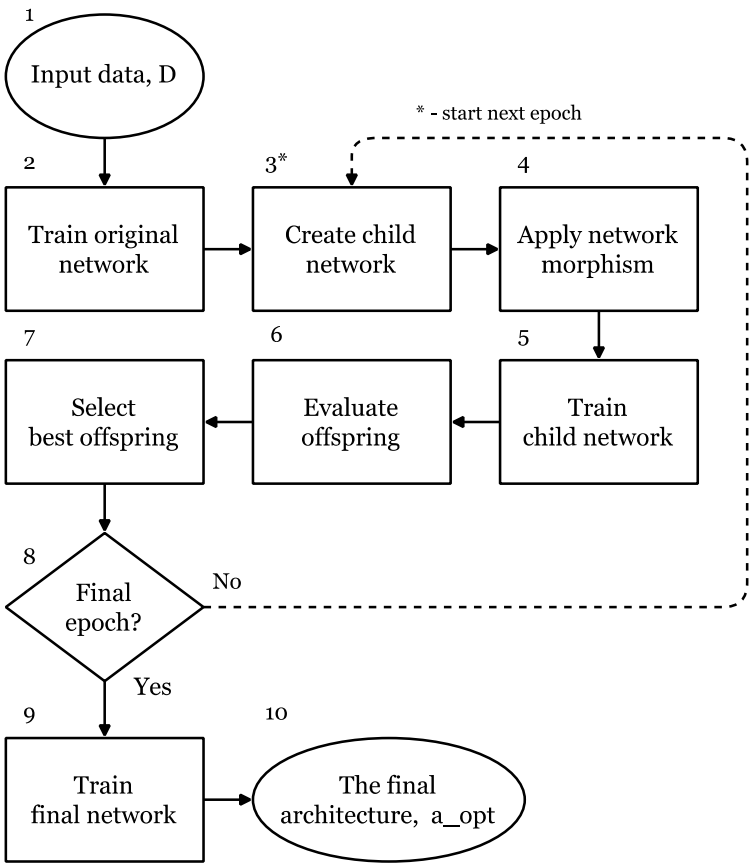

Fig. 2. The scheme of a genetic algorithm based on the enforced hill-climbing and enhanced with network morphism.

The algorithm begins with a data *D* input. Then, a pre-trained original neural network serves as a baseline architecture for climbing a so-called hill. In response to the application of functional-saving modifications, the original network produces a certain number of descendants. These modifications ensure that each descendant performs the same as its parent network. Moreover, due to the genetic algorithm, the descendants suit better for training than their parents. Each descendant is trained on a small number of epochs and then evaluated by a precision measure – the best descendent moves

to the next step, where they form new descendants. The search procedure ends after a certain number of epochs. The algorithm presents the so-called best architecture a_{opt}, which then serves for final long-term training. Fig. 3 demonstrates a heuristic architecture search based on enforced hill-climbing algorithm inspired by [18]:

```
function HAS( model_0 , n_steps , n_neigh , n_morph , epoch_neigh , epoch_final , l_rate )
    # model_0 – initial model, n_steps – number of hill-climbing steps;
    # n_neigh – number of neighbors, n_morph – number of func.-saving
    # modifications, epoch_neigh – number of epochs to train each neighbor,
    # epoch_final – number of epochs for final training,
    # l_rate – value of learning rate during model optimization.

    model_best := model_0
    # start enforced hill climbing
    for i:= 1,n_steps  do
        # get n_neigh neighbors of model_0 by applying n_morph func.-saving
        # modifications to model_best
        for j:= 1,n_neigh - 1  do
            model_j := ApplyFuncSav(model_best, n_morph)
            # train the model for several epochs on the training dataset
            model_j := Train(model_j, epoch_neigh, l_rate)
        end for
        # in case the last neighbor is the best
        model_n_neigh := Train(model_best, epoch_neigh, l_rate)
        # receive the best model on the validation dataset
        model_best := argmax_{j=1,...,n_neigh} { performance_val(model_j) }
    end for
    # train the final model both on training and validation datasets
    model_best := Train(model_best, epoch_neigh, l_rate)
    return model_best
end function
```

Fig. 3. The pseudo-code of heuristic architecture search with network morphism

The algorithm does not have to choose a new model at each iteration, and can also hold the one from the previous step if others do not improve. Consequently, the current best model at iteration may also be considered a child network. It is assumed that $epoch_{neigh}$ should be small, as the algorithm is forced to train numerous networks. Thus, to check the possibility of overfitting, we conducted two experiments with small and large numbers of epochs.

The algorithm presented above is a straightforward implementation of the hill-climbing method. It can be interpreted as a simple genetic algorithm with one organ-

ism, population size of n_{neigh}, and without crossover. While climbing, the functional-saving modifications of network morphism serve as mutations. Besides, the selection part considers only members of the population with the best performance as a parent for the next generation.

5 Implementation Details

5.1 Benchmark Dataset and Data Augmentation

To evaluate the proposed heuristic search, we chose the CheXpert benchmark dataset [25]. The whole dataset comprises 224,316 chest radiographs with a size of 320×320 pixels excluded from 65,240 patients. The images are labeled for 14 diseases as negative, positive, or uncertain. In the original dataset, positive and negative cases were marked as ones, dubious images – as zeros. Fig. 4 presents an example of the CheXpert dataset.

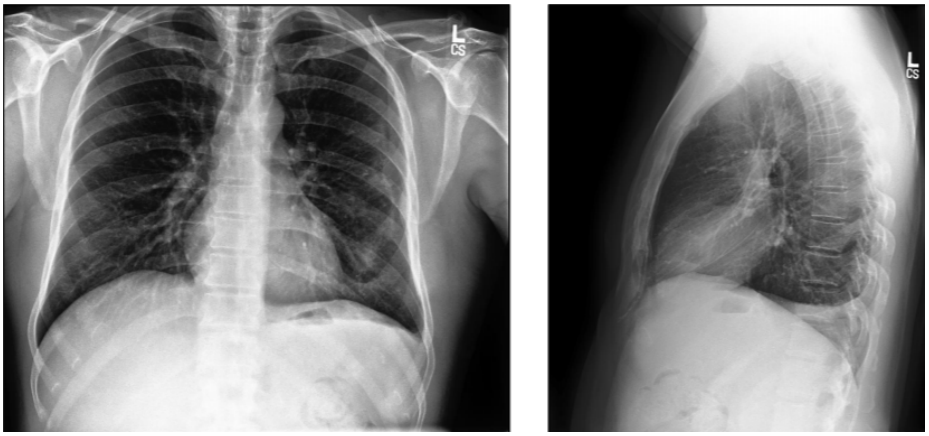

Fig. 4. CheXpert dataset sample [25].

For the experiment, we formed a subgroup of five diseases, namely atelectasis, cardiomegaly, consolidation, edema, and pleural effusion. Therefore, the number of classes was set to five. We split the subset into 70% training, 20% testing, and 10% validation images. Furthermore, several data augmentation techniques, such as random flips, translations, and rotations, were applied to the pictures of the training dataset.

5.2 Baseline Architecture

Here, we describe the baseline architecture, which serves as an original network for further optimization. According to the recent comprehensive overviews [26,27], CNNs are the type of neural architectures that suite the best for the classification of medical images. Therefore, this work is devoted to applying heuristic architecture

search only to CNNs. Hence, guided by [28,29], a small baseline convolutional architecture was represented as follows

$$\text{Input} \rightarrow \left\{16 \cdot \ 2^{i-1} \times \text{ConvL}_i \rightarrow \ \text{MPL}_i\right\}_{i=1}^{3} \rightarrow 128 \times \text{ConvL} \rightarrow \ \text{SML}\,, \tag{2}$$

where ConvL stands for convolutional layer, from 16 to 128 filters of size 3×3 and stride 1, ReLU is an activation function, MPL stands for MaxPool layer of size 2×2 and stride 2, and SML represents SoftMax function for the probability distribution of the output result.

The large-scale convolutional architecture was set as

$$\begin{aligned} \text{Input} \rightarrow \ & \left\{16 \cdot \ 2^{i-1} \times \text{ConvL}_i \rightarrow \ \text{ReLU}_i \rightarrow \ \text{MPL}_i\right\}_{i=1}^{3} \\ & \rightarrow 128 \times \text{ConvL} \rightarrow \ \text{BN} \rightarrow \ \text{ReLU} \rightarrow \ \text{SML}, \end{aligned} \tag{3}$$

where BN represents batch normalization. All other parts of architecture (2) denote the corresponding elements in the network (1).

5.3 Parameters Setup

In this section, we describe the setup of the training parameters. In the hill-climbing algorithm, we set the number of search epochs of 10, the number of organisms in the population of 10, the number of epochs for training each element of 5, the number of mutations of 5. We also limited the possible size of the model. If the sample model became too large, the agent would reject the investigated architecture and choose another one.

For the training procedure, we employed Adam optimization method with the learning rate of 10^{-3}, the weight decay of $0.5 \cdot 10^{-3}$, the momentum of 0.9, and a batch size of 256. According to experimental results in [30], this setup of training parameters can assure excellent model approximation in training. The original network was pre-trained by ten epochs on one fold of the training dataset, while the final architecture was optimized by one hundred runs on each fold from scratch. Overall, the training was performed five times, and the milestone results were averaged.

All experiments were performed in Python v3.6, using the TensorFlow v.1.13 backend [31]. The hardware setup consists of 8 core Ryzen 2700 and a single NVIDIA GeForce GTX1080 GPU with 8 GB memory. The working code is open-sourced and available by [32].

5.4 Evaluation Criteria

In this study, we evaluate the output architecture and compare it with other networks by several statistical measures, which are recall (REC), precision (PREC), accuracy (ACC), and area under the curve (AUC). Let us consider the number of real positive (P) and real negative (N) cases in the data. As it is known from the theory of statistics [33], the classification results are distributed as true positive (TP), true negative (TN),

false positive (FP) and false negative (FN) cases. Thus, the evaluation metrics used in this study are as follows

$$\mathrm{REC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{4}$$

$$\mathrm{PREC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{5}$$

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{6}$$

For binary classification, AUC is set as

$$A = \frac{1}{2}\left(\frac{\mathrm{FP}_2}{\mathrm{FP}_2 - \mathrm{TN}_2} - \frac{\mathrm{FP}_1}{\mathrm{FP}_1 - \mathrm{TN}_1} \right) \times \left(\frac{\mathrm{TP}_2}{\mathrm{TP}_2 - \mathrm{FN}_2} - \frac{\mathrm{TP}_1}{\mathrm{TP}_1 - \mathrm{FN}_1} \right).$$

In the case of five classes, AUS is as follows

$$\mathrm{AUC}_5 = \sum_{i=1}^{5} \mu(c_i)\, p(c_i), \tag{7}$$

where $\mu(c_i)$ stands for AUC under the ROC curve of class c_i, $p(c_i)$ is the prior probability of class c_i.

Also, the different architectures were evaluated and compared by network size, number of parameters, number of training epochs, and training time.

6 Experiments and Results

In this section, we describe how the initial type of network can influence the results of the heuristic architecture search. Also, we compare the result of our approach with two different numbers of epochs to the set of hand-crafted architectures.

6.1 Impact of a Baseline Architecture on the Search Process

The conducted experiments revealed that selecting a small original network lead to a lengthy search process. Moreover, small network (2) required numerous mutations, and thus, the time to evolve into an optimal architecture. In contrast, a large-scale architecture (3) could limit the search space with large structures, neglecting smaller ones that might also be suitable for search.

We conducted two representative experiments to investigate the above-mentioned hypothesis. In the first run, we checked the small original network (2), in the second – the large one (3). Fig. 5 shows a comparison of the results of two possible original architectures.

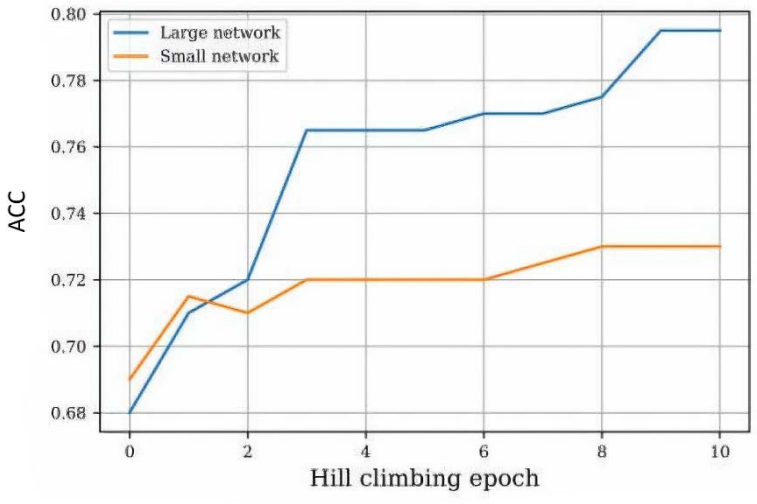

Fig. 5. The efficiency of a baseline architecture depending on its size.

According to Fig. 5, a large-scale network provides higher model accuracy.

6.2 The Final Network Performance

In this section, we examine the final architecture optimized by our NAS algorithm. We employed three hand-crafted convolutional networks commonly used in medical image tasks. These networks are VGG19 [1]. Inception v4 [2] and DenseNet [3]. The final network generated by the search algorithm can be observed via [32].

We trained the networks on the selected dataset and compared them with the final architecture of our algorithm. Besides, we conducted two separate experiments with different numbers of epochs to investigate the probability of overfitting. All networks were evaluated by metrics (4)–(7). The hyperparameters remained the same. Table 1 presents averaged evaluation results of k-fold validation ($k = 5$).

Table 1. Comparison of recognized hand-crafted neural networks with NAS architecture.

Network	REC	PREC	ACC	AUC
VVG19 [1]	0.723	0.631	0.627	0.747
Inception v4 [2]	0.739	0.646	0.715	0.815
DensNet [3]	0.812	0.742	0.727	0.838
The final NAS architecture, 1st experiment	0.764	0.701	0.731	0.842
The final NAS architecture, 2nd experiment	0.758	0.699	0.732	0.845

According to Table 1, the architecture found by our heuristic search demonstrates competitive results to state-of-the-art models in the classification tasks. The NAS architecture comprises various branches such as concatenates, skip nodes, adds. In contrast, hand-crafted architectures usually lack additional layers as it requires numerous experiments. While most state-of-the-art models have a regular structure, that is, they consist of multiple blocks that are repeated in the architecture, the NAS architecture has a structure without a noticeable repeating pattern. This approach allows for creating flexible architectures for different datasets. Table 2 reveals more details of the comparison.

Table 2. Comparison of networks by computational costs and weight of trained models.

Network	Training time (GPU hours)	Number of training epochs	Number of parameters (mil.)	Network size (MB)
VVG19 [1]	7.4	33	134.2	417
Inception v4 [2]	9.6	27	24.5	314
DensNet [3]	13.7	41	25.8	285
The final NAS architecture, 1st experiment	9.1	62	6.2	29
The final NAS architecture, 2nd experiment	7.8	21	6.1	29

The proposed NAS algorithm allowed finding the optimal architecture in 28 GPU hours, while the manual search can require weeks of tedious attempts and experiments. Furthermore, the final model fewer number of parameters with sufficiently high accuracy compared to the state-of-the-art. Thus, the model requires less physical memory and could ensure efficient use in practice.

7 Discussion and Conclusion

In this study, we proposed a heuristic algorithm of neural architecture search in medical image analysis. To investigate the issue of medical image classification, we employed the CheXpert benchmark dataset and considered a multiclassification task. The core of the search method was the enforced hill-climbing algorithm enhanced with network morphism.

Firstly, we composed and examined two hand-crafted CNNs as baseline architecture. As a result of numerical experiments, the large-scale network turned out to be a better solution. Secondly, we implemented a heuristic search with network morphism modifications in order to find optimal neural architecture. To check if the final network was subjected to overfitting, we conducted two separate experiments with numbers of epochs 21 and 69. As a result, even taking into account the almost three-fold difference between the number of epochs, both statistical indicators and computational costs and weight were approximately equal. This outcome could occur due to either

the use of one organism within the algorithm or lack of crossover. The authors aim to investigate this knowledge gap in future work.

We evaluated the final architecture found automatically with an objective statistical metrics and compared it with recognized hand-crafted networks on CheXpert dataset. Our search algorithm managed to find the optimal architecture in total in 28 GPU hours. The optimized architecture in the second experiment achieved the performance of 73.2% in accuracy and 84.5% in AUC on the validation dataset, yielding the competitive results to the state-of-the-art hand-crafted networks. Moreover, the optimized model contained four times fewer parameters and required almost ten times less physical memory compared to other networks. In summary, these results could indicate the practical usefulness of the proposed heuristic search method.

Further work needs to be done to determine the influence of sensitivity and specificity on the proposed heuristic search. A reasonable approach to tackle this issue could be the inclusion of diverse medical datasets of both X-Ray images and computer tomography scans. Further research should also focus on the optimization of hyperparameters within the heuristic architecture search.

References

1. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. Paper presented at the 3rd International Conference on Learning Representations (ICLR-2015), San Diego, CA, USA, 7–9 May 2015
2. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-2017), San Francisco, California, USA, 4–10 February 2017. pp. 4278–4284. AAAI Press (2017). doi:10.5555/3298023.3298188
3. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2017), Honolulu, HI, USA, 21–26 July 2017. pp. 2261–2269. IEEE Inc. (2017). doi:10.1109/CVPR.2017.243
4. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2009), Miami, FL, USA, 20–25 June 2009. pp. 248–255. IEEE Inc. (2009). doi:10.1109/CVPR.2009.5206848
5. Angeline, P.J., Saunders, G.M., Pollack, J.B.: An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans. Neural Networks. 5(1), 54–65 (1994). doi:10.1109/72.265960
6. Bello, I., Zoph, B., Vasudevan, V., Le, Q. V: Neural optimizer search with reinforcement learning. In: Precup, D. and Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning (ICML-2017), Sydney, Australia, 6–11 August 2017. vol. 70, pp. 459–468. JMLR.org (2017). doi:10.5555/3305381.3305429
7. Kandasamy, K., Neiswanger, W., Schneider, J., Póczos, B., Xing, E.P.: Neural architecture search with bayesian optimisation and optimal transport. In: Bengio, S., Wallach, H.M. (eds.) Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS-2018), Montreal, QC, Canada, 3–8 December 2018. pp. 2020–2029. Curran Associates Inc. (2018). doi:10.5555/3326943.3327130

8. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable architecture search. Paper presented at the 7th International Conference on Learning Representations (ICLR-2019), New Orleans, LA, USA, 6–9 May 2019
9. Kerschke, P., Hoos, H.H., Neumann, F., Trautmann, H.: Automated algorithm selection: Survey and perspectives. Evol. Comput. 27(1), 3–45 (2019). doi:10.1162/evco_a_00242
10. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. J. Mach. Learn. Res. 20(55), 1–21 (2019)
11. Radiuk, P.M.: Neuroevolution of convolutional neural networks for the classification of lung cancer images. Her. Khmelnytskyi Natl. Univ. 267(6), 188–192 (2018). doi:10.31891/2307-5732-2018-267-6(2)-188-192
12. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: From architectures to learning. Evol. Intell. 1, 47–62 (2008). doi:10.1007/s12065-007-0002-4
13. Chugh, T., Sindhya, K., Hakanen, J., Miettinen, K.: A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. Soft Comput. 23(9), 3137–3166 (2019). doi:10.1007/s00500-017-2965-0
14. Veniat, T., Denoyer, L.: Learning time/memory-efficient deep architectures with budgeted super networks. In: Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR-2018), Salt Lake City, UT, USA, 18–23 June 2018. pp. 3492–3500. IEEE Inc. (2018). doi:10.1109/CVPR.2018.00368
15. Li, X., Zhou, Y., Pan, Z., Feng, J.: Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In: Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR-2019), Long Beach, CA, USA, USA, 15–20 June 2019. pp. 9137–9145. IEEE Inc. (2019). doi:10.1109/CVPR.2019.00936
16. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K. and Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning (ICML-2019), Long Beach, California, USA, 10–15 Jun 2019. vol. 97, pp. 6105–6114. PMLR Inc. (2019)
17. Wei, T., Wang, C., Rui, Y., Chen, C.W.: Network morphism. In: Balcan, M.F., and Weinberger, K.Q. (eds.) Proceedings of the 33rd International Conference on Machine Learning (ICIML-2016), New York City, NY, USA, 19–24 Jun 2016. vol. 48, pp. 564–572. JMLR.org (2016). doi:10.5555/3045390.3045451
18. Elsken, T., Metzen, J.-H., Hutter, F.: Simple and efficient architecture search for convolutional neural networks. Paper presented at the 6th International Conference on Learning Representations (ICLR-2018), Vancouver, BC, Canada, 30 April – 3 May 2018
19. Gordon, A., Eban, E., Nachum, O., Chen, B., Wu, H., Yang, T., Choi, E.: MorphNet: Fast & simple resource-constrained structure learning of deep networks. In: Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR-2018), Salt Lake City, UT, USA, 18–23 June 2018. pp. 1586–1595. IEEE Inc. (2018). doi:10.1109/CVPR.2018.00171
20. Gessert, N.T., Schlaefer, A.: Efficient neural architecture search on low-dimensional data for OCT image segmentation. In: Proceedings of Conference on Medical Imaging with Deep Learning (MIDL-2019), London, The UK, 8–10 July 2019. pp. 1–5. Medizintechnische Systeme E-1 Institute (2019). doi:10.15480/882.2731
21. Qiang, N., Ge, B., Dong, Q., Ge, F., Liu, T.: Neural architecture search for optimizing deep belief network models of fMRI data. In: Li, Q., Leahy, R., Dong, B., and Li, X. (eds.) Proceedings of the 1st International Workshop on Multiscale Multimodal Medical Imaging (MMMI-2019), Shenzhen, China, 13 October 2019. vol. 11977, pp. 26–34. Springer International Publishing (2020). doi:10.1007/978-3-030-37969-8_4

22. Kwasigroch, A., Grochowski, M., Mikołajczyk, A.: Neural architecture search for skin lesion classification. IEEE Access. 8, 9061–9071 (2020). doi:10.1109/ACCESS.2020.2964424
23. Radiuk, P.M., Hrypynska, N.V.: A framework for exploring and modeling neural architecture search methods. Paper presented at the 4th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2020), Lviv, Ukraine, 23–24 April 2020. CEUR-Workshop Proceedings, vol. 2604, pp. 1060–1074. CEUR-WS.org (2020)
24. Edelkamp, S., Schrödl, S.: Memory-restricted search. In: Edelkamp, S. and Schrödl, S.B.T.-H.S. (eds.) Heuristic Search: Theory and Applications. pp. 227–281. Morgan Kaufmann, San Francisco (2012). doi:10.1016/B978-0-12-372512-7.00006-7
25. Irvin, J., et al.: CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In: Proceedings of the 33d AAAI Conference on Artificial Intelligence 2019 (AAAI-2019), Honolulu, Hawaii, USA, 27 January – 1 February 2019. vol. 33(1), pp. 590–597. Association for the Advancement of Artificial Intelligence (2019). doi.10.1609/aaai.v33i01.3301590
26. Shen, J., Zhang, C.J.P., Jiang, B., Chen, J., Song, J., Liu, Z., He, Z., Wong, S.Y., Fang, P.-H., Ming, W.-K.: Artificial intelligence versus clinicians in disease diagnosis: A systematic review. JMIR Med. Informatics. 7(3), e10010 (2019). doi:10.2196/10010
27. Kim, M., Yan, C., Yang, D., Wang, Q., Ma, J., Wu, G.: Deep learning in biomedical image analysis. In: Feng, D.D. (eds.) Biomedical Information Technology. pp. 239–263. San Diego, Elsevier (2020). doi:10.1016/B978-0-12-816034-3.00008-0
28. Romanuke, V.V.: An attempt of finding an appropriate number of convolutional layers in CNNs based on benchmarks of heterogeneous datasets. Electr. Control Commun. Eng. 14(1), 51–57 (2018). doi:10.2478/ecce-2018-0006
29. Romanuke, V.V.: Smooth non-increasing square spatial extents of filters in convolutional layers of CNNs for image classification problems. Appl. Comput. Syst. 23(1), 52–62 (2018). doi:10.2478/acss-2018-0007
30. Radiuk, P.M.: Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. Inf. Technol. Manag. Sci. 20(1), (2017). doi:10.1515/itms-2017-0003
31. Abadi, M., et al.: TensorFlow: A system for large-scale machine learning. In: Keeton, K. and Roscoe, T. (eds.) Proceedings of 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI-2016), Savannah, GA, USA, 2–4 November 2016. pp. 265–283. USENIX Association (2016)
32. Heuristic NAS. GitHub, Inc. https://github.com/soolstafir/Heuristic-NAS (2020). Accessed 27 Apr 2020
33. Fawcett, T.: An introduction to ROC analysis. Pattern Recognit. Lett. 27(8), 861–874 (2006). doi:10.1016/j.patrec.2005.10.010