Neural Architecture Search for Fracture Classification

Aloïs Pourchot, Kevin Bailly, Alexis Ducarouge, Olivier Sigaud

NEURAL ARCHITECTURE SEARCH FOR FRACTURE CLASSIFICATION

Aloïs Pourchot[1,2,⋆], *Kévin Bailly*[1], *Alexis Ducarouge*[2], *Olivier Sigaud*[1]

[1] Sorbonne Université CNRS UMR 7222, ISIR, F-75005 Paris, France
[2] Gleamer, 117 Quai de Valmy, 75010 Paris, France

ABSTRACT

The adoption by radiologists of deep-learning based solutions to the bone fracture problem has helped improved diagnostic performances and patient care. The base models behind these tools were initially designed to solve problems on natural images, favoring transfer learning between standard image datasets and sets of radiographs. Those architectures could yet be made more specific to radiographs using neural architecture search (NAS). Unfortunately, current NAS approaches do not benefit from transfer learning. In this paper, we introduce an efficient scheme to exploit transfer learning when performing NAS. Using our approach, we validate the architecture tailoring paradigm to radiographs. On a custom fracture classification task, we find a new model with improved performances and reduced computational overhead over its counterparts pre-trained on ImageNet.

***Index Terms*—** Neural Architecture Search (NAS), Medical Imaging, Fracture Classification

1. INTRODUCTION

Radiography is the main tool for fracture diagnosis and one of the most prevalent imaging modality worldwide. Yet, the reading of radiographs is a demanding task requiring medical expertise that gets scarcer as the volume of images to analyse increases. A promising direction to facilitate interpretation and reduce the prevalence of errors is to assist radiologists with a computer-aided diagnosis software. The recent advent of deep learning has made such software capable of improving the performance of radiologists [9] and even outperform experts on their own [4, 6].

These systems are usually built on top of existing object detection frameworks, and reuse long-established convolutional backbones such as ResNet or DenseNet, which were designed to perform well on natural images datasets such as CIFAR-10, ImageNet or COCO. Re-using off-the-shelf models allows them to rely on transfer learning through fine-tuning, a feature that is key to improve performance when working with scarce data sources [8]. Tailoring model architectures to radiographs using Neural Architecture Search (NAS) appears as a promising lead for improved performance, and may be a key step towards lightweight architectures, which are key to make these medical tools widely available through smartphones or embedded medical devices. Unfortunately, classical NAS approaches are hardly compatible with pre-training, as they involve training many different architectures for which no set of pre-trained weights are available in advance. This makes NAS algorithms impractical in scenarios where transfer learning is key.

In this paper, we insist on the importance of transfer learning on a fracture patch classification task. We then introduce a procedure that reconciles classical NAS approaches and transfer learning in a reasonable setting. Finally, we challenge the efficacy of the architecture fitting paradigm, and demonstrate using a plain genetic algorithm that it is possible to assemble architectures adapted to radiographs that are slightly better than architectures selected on ImageNet, whilst having less parameters and multiply-adds.

2. RELATED WORK

2.1. Deep Learning for Fracture Detection

Several deep learning based solutions to the fracture detection problem have been studied in the literature, but no technological breakthrough has clearly emerged so far. All the authors *de facto* reuse an off-the-shelf architecture designed and pre-trained on datasets of natural images, which they fine-tune on private annotated datasets of fractures. This work is based on previous studies of Gleamer [4, 6], in which a Mask-RCNN [7] model pre-trained on COCO, is assembled using the Detectron2 framework [23] and fine-tuned on a private internal dataset of 60,000 radiographs of patients with trauma gathered from 22 institutions and annotated by medical experts.

2.2. Neural Architecture Search

In the last decade, a paradigm shift was observed, from hand-designing features that can be fed to a machine learning algorithm, to hand-designing architectures of neural networks that can extract those features automatically. However, designing neural networks is itself a time-consuming task that requires domain-dependent expertise. The automatic arrangement

⋆ Corresponding author: alois.pourchot@gleamer.ai

of those architectures through Neural Architecture Search (NAS) [5], is an interesting way to mitigate this burden. Several approaches to NAS have been studied in the literature, from using deep reinforcement learning (RL) [27, 28], evolutionary algorithms [15, 14] or even fully differentiable approaches [10]. The inherent training cost of a single deep learning architecture is a strong limitation of this paradigm. Early NAS algorithms [14, 28, 27] suffered from impractical computational costs, requiring up to thousands of GPU days worth of computing.

Several methods have since been suggested to alleviate those costs. In particular, a plethora of papers rely on weight-sharing (WS) [11], a computational trick which allows to train all the architectures of a search space at once. This WS mechanism is however poorly understood, and several recent works suggest that its efficacy in a realistic scenario is limited [12, 26, 24], due to restricted correlations with real evaluation scores, and bias in the super-net training that favors certain architectures at the expense of others. WS can still be particularly useful in scenarios calling for a quick access to decent weights for many different architectures. In this paper, rather than using a super-net to directly conduct NAS, we propose to exploit a super-net trained on natural images as a generator of pre-trained weights used to fine-tune architectures on our fracture classification task.

3. METHODS

In this section, we first introduce our dataset of fracture patches. We then describe our scheme to combine NAS and transfer learning from natural images. From there, we introduce the Once For All framework [2], and the genetic algorithm with which we perform NAS. Finally, we describe the individual training setup of architectures.

3.1. Fracture Patches Dataset

Due to the complex and wide variety of patterns that can appear on medical images, medical detection models often require an associated false positive classification network to improve their specificity [19]. We thus propose to challenge the architecture tailoring process on a patch classification task. We rely on the detection model mentioned in Section 2.1 to create a dataset of diverse and realistic patches. During inference, the detection model produces bounding boxes of interest, each coming with an associated score, ranging from 0 to 100, expressing the confidence of the network in the presence of a fracture within the suggested region.

We performed a forward pass on the training images of our private internal dataset and kept the predicted bounding boxes with a confidence above 10. This procedure generated 500,000 patches, which we split into a training, a validation and a test dataset, respectively containing 60%, 20% and 20% of the samples. Extracting patches solely from the detection training dataset is necessary for fair model comparisons as it ensures that the distribution of prediction localization is the same during training, validation and test of the classification task. Predicted bounding boxes that had an intersection over union (IoU) with a ground truth bounding box above 0.5 were considered positives. Samples with an IoU below 0.2 were deemed negatives. During training, boxes with intermediate IoUs between 0.2 and 0.5 were ignored so as not to confuse the classification networks, but were considered as negatives during testing.

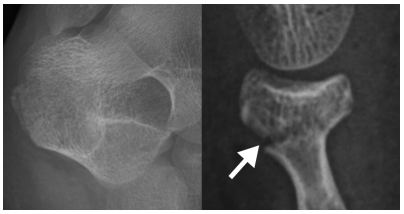

Fig. 1. Examples of patches created using the procedure described in Section 3.1. On the left, a false positive fracture patch located on a patient's heel. On the right, a patch containing a toe fracture, marked by a white arrow.

3.2. ImageNet pre-training

Fine-tuning architectures pre-trained on large image classification datasets such as ImageNet is a well known strong baseline when working with scarce data sources [8]. Although the extent to which this transfer learning is beneficial ultimately depends on the task at hands [8], we found that leveraging ImageNet pre-training was essential for our fracture classification task. As will transpire clearly in Section 4, the performance increase resulting from ImageNet pre-training is substantially greater than what is usually gained by optimizing an architecture to a problem.

Nonetheless, NAS algorithms commonly train architectures from scratch. This is reasonable, as the ImageNet dataset itself is used to evaluate NAS procedures. Besides, naively incorporating this pre-training further increases computational expenses, as each architecture needs to be first pre-trained, then fine-tuned on the task of interest. Instead of pre-training each individual architecture separately, we propose to exploit a pre-trained super-net. Indeed, by design, the weights of all associated architectures can be extracted from the sole training of this super-net. Although the individual performance obtained with shared pre-trained weights are likely worse than those of individual training, they provide a cheap and efficient proxy.

3.3. Search Space

Training super-nets is a long and minute process that requires clever tricks [2, 25]. To overcome this difficulty, we rely on publicly available assets. Few of the approaches described in the literature provide the code used for their development. The sole work additionally providing the weights of their

trained super-net that we could find was the Once For All (OFA) framework of [2]. The goal of the OFA approach is to quickly find neural architectures adapted to specific inference settings, a task in which WS particularly shines given its ability to very quickly evaluate architectures of interest. Each model of their search space can be divided into five stacks of several convolutional layers. The number of convolution layers in each stack varies between 2, 3 and 4. Individual layers are based on inverted residual blocks [17], and each layer has an expansion ratio chosen between 3, 4 and 6, and a kernel size, chosen between 3, 5 and 7. In total, an architecture is described by 45 parameters, each taking 3 possible values. For deeper understanding of the ins and outs of OFA, we refer the reader to the original paper.

3.4. NAS Configuration

There is evidence in the literature that local search based NAS algorithms are quite efficient [3, 22]. Using a genetic algorithm to optimize the architecture follows naturally, especially considering the original results of [14, 15]. To perform the optimization, we opt for a plain $(1 + 1)$ evolution strategy which benefits from straight-forward parallelization [16]. At any time, to suggest an architecture, the algorithm considers the best model found so far, and creates a child candidate by applying random mutations. During this mutation process, each of the 45 parameters of the current best model is modified with probability p, which varies with the considered heuristic. Typically, p is set to the inverse of the dimension of the problem, such that on average a single parameter is modified. To further accelerate the process, we bootstrap the search from the best model found on ImageNet by the OFA authors, and set it as the first best architecture. Still, to prevent the optimization process from getting stuck around this model, we mimic brain storm optimization [18], by decaying the mutation probability from 1 to $\frac{1}{45}$ over the course of the optimization. Since we indirectly encourage the NAS approach to search for architectures in the neighbourhood of the best OFA model, we do not impose strong computational constraints on the found architectures, and optimize solely for the performance metric described in the next section. The optimization is performed using the Nevergrad framework [13].

3.5. Architecture Training and Evaluation

To reduce the computational cost of search, we follow the path of [20, 21] and evaluate models after five training epochs. We optimize architectures with standard stochastic gradient descent, using an initial learning rate of 0.02 decayed to 0 over the course of the five epochs using a cosine annealing scheme, a momentum of 0.9, batch-size 128 and a weight-decay of 10^{-4}. We additionally use exponential moving average of the weights, with a decay of 0.99. To evaluate the obtained models on the validation dataset, we report the performance of the underlying detection model obtained when replacing the scores of the original detection model with those of the newly trained model. Each patch is associated to the fracture score returned by the classification model, and we compute the area under the Free-ROC curve (AUFROC) [1] as the metric of interest. Images of our datasets are grouped in studies, with each study corresponding to a unique patient examination. The F-ROC curve is obtained by performing a study-wise average of the recall, as a function of the average number of false positives. A fracture is considered detected if it is accurately pinpointed on at least one of the images of the considered study. The AUFROC is the area under the resulting curve. To avoid having to work with varying intervals of definition, we choose to integrate the AUFROC between 0.02 and 0.5 false positives per image, as those two points cover most of the operating points of radiologists [4]. During the NAS run, we evaluate 100 architectures. Each individual training is performed on a cluster of four K80 GPUs, with 10 cluster running in parallel at all times.

4. RESULTS

We report the evolution of the validation scores of the models selected by the NAS algorithm in Figure 2. The total computing time amounts to around 450 hours. The influence of the brainstorm process introduced in Section 3.4 can be deducted from the performance evolution. The top left point, which corresponds to the best ImageNet model found by the OFA authors, provides a strong anchor point to the search. In the beginning of the run, the high mutation rate results in a performance drop, but several candidates of interest are found later on.

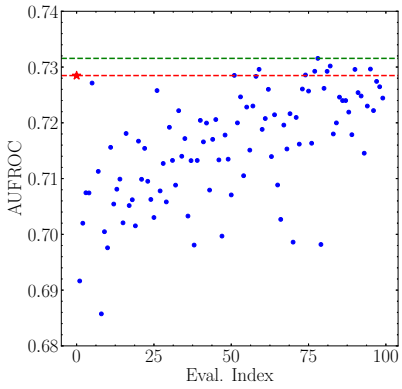

Fig. 2. Evolution of validation performance over the search. The red point marked by a star on the top left corresponds to the best architecture found on ImageNet by [2]. The red (resp. green) horizontal line indicates the corresponding score, (resp. the best found validation metric).

Figure 3 provides a scatter plot of the number of trainable parameters of the architectures encountered over the run, against their validation score. Interestingly, no architecture was sampled with more parameters than the reference OFA

model. Among the sampled parameters, the first five act most on the number of parameters, as they dictate the number of convolutions appearing in each of the five blocks constituting the model. For the original OFA architecture, four out of those five parameters were set to their maximum value. This makes it hard to sample an architecture with more parameters in the early part of the brainstorm process where search is close to random. In the middle of the search, the algorithm could find an architecture with less convolution layers that performed better. This new architecture became the reference around which the next architectures were more closely sampled as the brainstorming faded away, which made sampling architectures with less parameters easier.

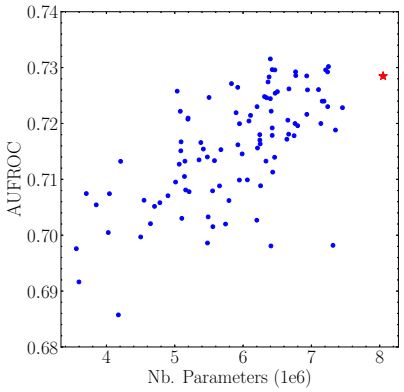

Fig. 3. Scatter plot of the validation performance against the number of parameters of the architectures seen during NAS. The red star marks the best OFA ImageNet architecture.

From the architectures found by the NAS, we extract the three with the best validation metrics and re-train them with the setting described in Section 3.5, but increasing the number of epochs to 20. Out of the three, our final architecture is the one with the best validation performance after this longer training. We refer to it as FractNet. We also perform this re-training for the best ImageNet OFA model. Additionally, we train with the same setting ResNet-152 and DenseNet-161 baselines, which are both used in the current Gleamer pipeline, and models from the EfficientNet family [21], which are state-of-the-art lightweight models, obtained using NAS on the ImageNet dataset over a different search space than OFA. All of those additional networks were pre-trained on ImageNet. To illustrate the relevance of pre-training, we further gather the performances of the OFA model and our FractNet, when training from scratch under the same setting.

Table 1 reports the final test metrics, as well as the number of parameters and the number of multiply-adds for the processing of a single image for each architecture and training of interest. The results first show that transfer learning is key to the problem, and that performing NAS without pre-training would be pointless. Secondly, our final FractNet model performs slightly better than the best ImageNet-selected OFA network, whilst having around 18% less parameters. This indicates that the architecture tailoring process has potential for improving performances in computationally constrained settings. Thirdly, our FractNet model has about the same number of parameters as an EfficientNet-b1, whilst performing slightly better. This strengthens our stand that task-specific architectures can improve performances. Finally, we see that our FractNet model reaches promising results compare to much larger models such as ResNet-151 or DenseNet-161, which require 20 and 13 times as many multiply-adds operations per image processed respectively.

Network	AUFROC	#Params	#MAdds
DenseNet-161	0.773[0.769,0.777]	26.3M	7.7B
ResNet-152	0.768[0.764,0.772]	58.2M	11.7B
EfficientNet-b0	0.740[0.736,0.744]	4.0M	0.38B
EfficientNet-b1	0.748[0.744,0.752]	6.5M	0.57B
OFA	0.746[0.742,0.750]	7.8M	0.61B
FractNet	0.754[0.750,0.758]	6.4M	0.56B
OFA •	0.676[0.672,0.680]	7.8M	0.61B
FractNet •	0.674[0.670,0.678]	6.4M	0.56B

Table 1. For each model of interest, we report the test AUFROC with a 90% confidence interval estimated using bootstrap, the number of trainable parameters (#Params), and the number of multipy-adds operations for the inference of a single image (#MAdds). OFA refers to the best model found on ImageNet by the authors of the OFA framework [2]. FractNet refers to the model selected using our NAS approach. Models marked with a black bullet were trained without ImageNet pre-training.

5. CONCLUSION

In this paper, we considered the problem of achieving computationally efficient NAS in a specific context where data is scarce: the fracture classification paradigm. We have shown that exploiting transfer learning was key to alleviate data sparsity and drastically reduce computing times. To perform NAS without losing the benefits of pre-training, we have proposed to exploit a super-net trained on ImageNet as a generator of pre-trained weights used to fine-tune architectures on our fracture classification task. From there, we introduced a plain genetic NAS algorithm and performed NAS on the search space of computationally efficient architectures introduced by the authors of the OFA framework. With this approach, we have created the FractNet model, which obtains better performances on the fracture classification problem whilst reducing computational overheads. This validates both the relevance of the architecture tailoring process, as well as our introduced super-net pre-training protocol. As further work, one could try to replicate this scheme on other search spaces, such as the EfficientNet search space. It would also be interesting to explore whether the performance gap with respect to much larger architectures could be reduced through techniques such as model distillation, or by slightly increasing model capacity using the EfficientNet growing strategy.

6. REFERENCES

[1] A. I. Bandos, H. E. Rockette, T. Song, and D. Gur. Area under the free-response roc curve (froc) and a related summary index. *Biometrics*, 65(1):247–256, 2009.

[2] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han. Once-for-all: Train one network and specialize it for efficient deployment. In *Proc. of ICLR*. OpenReview.net, 2020.

[3] T. Den Ottelander, A. Dushatskiy, Virgolin, et al. Local search is a remarkably strong baseline for neural architecture search. In *EMO*, pages 465–479. Springer, 2021.

[4] L. Duron, A. Ducarouge, A. Gillibert, et al. Assessment of an AI aid in detection of adult appendicular skeletal fractures by emergency physicians and radiologists: a multicenter cross-sectional diagnostic study. *Radiology*, 300(1):120–129, 2021.

[5] T. Elsken, J. H. Metzen, and F. Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. In *Proc. of ICLR*. OpenReview.net, 2019.

[6] A. Guermazi, C. Tannoury, A. J. Kompel, et al. Improving radiographic fracture recognition performance and efficiency using artificial intelligence. *Radiology*, page 210937, 2021.

[7] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *Proc. of ICCV*, pages 2980–2988. IEEE Computer Society, 2017.

[8] S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better? In *Proc. of CVPR*, pages 2661–2671. Computer Vision Foundation / IEEE, 2019.

[9] R. Lindsey, A. Daluiski, Chopra, et al. Deep neural network improves fracture detection by clinicians. *Proceedings of the National Academy of Sciences*, 115(45):11591–11596, 2018.

[10] H. Liu, K. Simonyan, and Y. Yang. DARTS: differentiable architecture search. In *Proc. of ICLR*. OpenReview.net, 2019.

[11] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In *ICML*, pages 4095–4104. PMLR, 2018.

[12] A. Pourchot, A. Ducarouge, and O. Sigaud. To share or not to share: A comprehensive appraisal of weight-sharing. *arXiv preprint arXiv:2002.04289*, 2020.

[13] J. Rapin and O. Teytaud. Nevergrad. https://github.com/facebookresearch/nevergrad, 2018.

[14] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In *Proc. of AAAI*, pages 4780–4789. AAAI Press, 2019.

[15] E. Real, S. Moore, A. Selle, et al. Large-scale evolution of image classifiers. In D. Precup and Y. W. Teh, editors, *Proc.*, volume 70 of *Proceedings of Machine Learning Research*, pages 2902–2911. PMLR, 2017.

[16] I. Rechenberg. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie*, pages 83–114. Springer, 1978.

[17] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. of CVPR*, pages 4510–4520. IEEE Computer Society, 2018.

[18] Y. Shi. Brain storm optimization algorithm. In *International conference in swarm intelligence*, pages 303–309. Springer, 2011.

[19] K. Suzuki. Overview of deep learning in medical imaging. *Radiological physics and technology*, 10(3):257–273, 2017.

[20] M. Tan, B. Chen, R. Pang, et al. Mnasnet: Platform-aware neural architecture search for mobile. In *Proc. of CVPR*, pages 2820–2828. Computer Vision Foundation / IEEE, 2019.

[21] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proc. of ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019.

[22] C. White, S. Nolen, and Y. Savani. Local search is state of the art for nas benchmarks. *arXiv preprint arXiv:2005.02960*, 2020.

[23] Y. Wu, A. Kirillov, F. Massa, et al. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[24] A. Yang, P. M. Esperança, and F. M. Carlucci. NAS evaluation is frustratingly hard. In *Proc. of ICLR*. OpenReview.net, 2020.

[25] J. Yu, P. Jin, H. Liu, et al. Bignas: Scaling up neural architecture search with big single-stage models. In *Proc. of ECCV*, pages 702–717. Springer, 2020.

[26] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann. Evaluating the search phase of neural architecture search. In *Proc. of ICLR*. OpenReview.net, 2020.

[27] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *Proc. of ICLR*. OpenReview.net, 2017.

[28] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proc. of CVPR*, pages 8697–8710. IEEE Computer Society, 2018.