Exploring NAS spaces with C-BRED

Matteo Spallanzani, IIS, D-ITET, ETH Zürich
Thorir Mar Ingolfsson, IIS, D-ITET, ETH Zürich

1 Introduction

Deep neural networks (DNNs) are amongst the most effective machine learning systems available nowadays. Indeed, DNNs are critical components of automated decision making systems and scientific tools that were unthinkable just a decade ago [1, 2, 3]. These breakthroughs have been due both to the technological context (fast, energy-efficient, programmable, parallel computing devices such as graphics processing units (GPUs), as well as large annotated data sets) and to the advancements in the design of network architectures. The evolution of convolutional neural network (CNN) architectures is emblematic of the latter [4, 5, 6, 7, 8, 9, 10, 11].

Assessing the trade-offs between a DNN's accuracy and its computational requirements requires considerable expertise, experimentation and, ultimately, time. For this reason, the DNN research community has devoted considerable efforts to neural architecture search (NAS) [12]. NAS is the deep-learning-specific variant of the standard statistical problem of model selection: it encompasses techniques to describe entire families of network architectures in some parametric form, and then select those members which are most likely to deliver the desired accuracy. The families of network architectures are known as **NAS spaces** (or simply *search spaces*), whereas the selection techniques are known as **NAS algorithms** (or simply *search algorithms*).

NAS spaces are usually designed to include a large number of networks to increase the likelihood of including members with good performance. However, this over-parametrisation implies that search algorithms will require considerable amounts of time to identify suitable candidates. For this reason, techniques that are capable of reducing the size of over-parametrised NAS search spaces are important to accelerate the convergence of NAS search algorithms. To increase the likelihood that the search algorithms will yield networks with good performance, these *sub-space selection algorithms* should return collections of architectures whose quality is in some sense superior to the quality of the architectures enclosed in the entire space.

In a previous project, we designed and implemented clustering-based reduction (C-BRED), a new algorithm to select high-quality sub-spaces from given NAS spaces. In this project, we will refactor the implementation of C-BRED to extend its scope and explore its capabilities more thoroughly.

2 Clustering-based reduction of NAS spaces

2.1 An overview of NAS spaces and algorithms

The parametrisation of NAS spaces is usually defined in terms of macroscopic structural characteristics such as the number of layers and their mutual connectivity. Some DNN search spaces are parametrised in terms of *cells*, i.e., sub-networks including a few layers which are replicated and stacked on top of each other to compose the candidate networks. The NAS-Bench-101 (NB101) and NAS-Bench-201 (NB201) data sets are NAS spaces designed for the benchmarking of NAS algorithms, and are examples of such cell-based spaces [13, 14].

Researchers in the NAS community have proposed search algorithms based on different paradigms: from evolutionary algorithms [15], through reinforcement learning [16] and differentiable algorithms [17, 18], to randomised algorithms [19] and even graph-based Bayesian methods [20, 21]. Several search algorithms have also been proposed to satisfy constraints imposed by the target computing platform [22, 23].

2.2 Computational graphs

Computational graphs provide a convenient way to describe computer programs since operands and operations can be represented as nodes, whereas read and write operations can be represented as arcs. Modern deep learning frameworks such as TensorFlow [24] and PyTorch [25] are built around the dataflow programming model, which is based on the computational graph abstraction [26, 27].

2.3 Clustering

Clustering can be considered a sub-field of unsupervised machine learning. It encompasses a collection of techniques to partition a given family of data points into disjoint subsets (the *clusters*) based on some similarity measure: points which are close under the chosen metric should be mapped to the same subset [28, 29, 30].

In some cases, it is hard to give explicit definitions of distances. Graphs are an example of objects for which defining explicit distances is non-trivial. In these cases, we can use kernel functions to measure object similarity [31, 32].

2.4 DNN statistics

From the user's perspective, the most important property that a DNN must satisfy is delivering the required functionality. This consideration makes *statistical accuracy*, or simply *accuracy*, the most important metric to evaluate a DNN.

Evaluating the accuracy of all the networks in a given NAS space can translate into unfeasible computational and time requirements. To see this, we make two considerations. First, it is reasonable to suppose that a DNN will deliver higher accuracy after having been trained; therefore, we are interested

in measuring accuracy after training has taken place. Second, it is unlikely that training the same architecture twice will yield identical results due to the stochasticity inherent to the training process (both the initial value of the parameter and the direction of the update steps are non-deterministic); therefore, we should perform multiple training experiments for a single DNN architecture in order to thoroughly assess its quality.

Note that the cost of measuring accuracy is mostly related to the cost of obtaining a single measurement. Training-free (TF) statistics are metrics to evaluate DNNs that can be measured running a single or even no iterations of the stochastic gradient descent (SGD) algorithm. Existing research has identified a few candidate TF statistics that have a desirable property: given a target network, their distributions seem to correlate with the network's accuracy distribution [33, 34, 35, 36, 37]. The hope is that these TF statistics can be used as cheap-to-compute proxies for accuracy.

2.5 Evaluating network spaces

Using statistical jargon, NAS spaces can be considered *populations* whose individuals are network architectures. Populations are typically evaluated by analysing how the values of selected observable variables distribute over its individuals. This probabilistic approach to evaluate populations of networks has recently been applied to the analysis of network spaces [38, 39]. Assuming that good TF statistics correlate with accuracy, we can analyse the distribution of TF statistics inside a NAS space (or a subset of its) to estimate the quality of its enclosed architectures.

2.6 Putting the pieces together: C-BRED

C-BRED combines similarity measures between computational graphs, clustering algorithms, and TF statistics to achieve the desired reduction of NAS spaces.

First, given a similarity measure and the computational graphs of the target NAS space, it computes a distance (i.e., dissimilarity) or similarity matrix between all the architecture pairs. Then, it provides the similarity measure to the clustering algorithm; to make the clustering process more robust, C-BRED uses cross-validation and other techniques to identify the partition which is most stable with respect to perturbations. Finally, clusters are compared by looking at the TF statistic distributions of their enclosed architectures, and the most promising sub-space is returned.

3 Project plan

This project will consist of two parts: the development of a modular software tool to explore the application of C-BRED to arbitrary NAS spaces; and an experimental investigation of the impact of such parameters on the performance of C-BRED, as measured on the NB101 and NB201 spaces.

3.1 Software development

The first part of the project will deal with the design, implementation, and testing of the C-BRED software tool.

First, during the design stage, we will analyse the C-BRED flow to identify its elementary components and define a composable system of primitives; this system of primitives should be independent of the specific graph distances, clustering algorithms and cluster selection procedures used. Then, during the implementation and testing steps, you will turn the software project into properly tested code; we will have regular code reviews to ensure that the delivered code can be easily reused by other researchers. In addition to C-BRED's functional requirements, the new codebase should support the following visualisation functionalities:

- scatterplots supporting both numerical and categorical colour codes; in conjunction with dimensionality reduction techniques, this feature will allow users to visually inspect NAS spaces by colouring each point (i.e., each network) with the corresponding value of a chosen TF statistic (numerical) or the identifier of its enclosing cluster (categorical);
- histograms and estimated densities of one-dimensional distributions; this feature will provide users with graphical devices to evaluate the clusters created by C-BRED.

3.2 Experimental evaluation

As a system-level test, during the second part of the project, we will apply the new C-BRED software tool to the NB101 and NB201 NAS spaces.

In particular, we are interested in evaluating both the absolute quality of the selected sub-spaces and their relative quality with respect to sibling clusters, under different hyper-parameter configurations. The quality of a cluster can be quantified by the distribution of the accuracy statistic over its enclosed networks: example metrics include the mean of this distribution, its variance, and the distribution of the minimum accuracy achieved by N-shot random search over the cluster. The space of hyper-parameter configurations is defined by the Cartesian product of the following degrees of freedom:

- the graph distance;
- the clustering procedure;
- the cluster selection procedure.

References

[1] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, T. Guez, A. Hubert, L. Baker, A. Lai, M. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering

the game of Go without human knowledge," *Nature*, vol. 550, pp. 354–359, 2017.

[2] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, pp. 350–354, 2019.

[3] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, S. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Higly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, pp. 583–589, 2021.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS 2012)*, Neural Information Processing Systems, 2012.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, ICLR, 2015.

[6] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014)*, ICLR, 2014.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the 2015 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2015)*, IEEE, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, IEEE, 2016.

[9] F. Chollet, "Xception: deep learning with depthwise separable convolutions," in *Proceedings of the 2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, IEEE, 2017.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: inverted residuals and linear bottlenecks," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018)*, IEEE, 2018.

[11] M. Tan and Q. V. Le, "EfficientNet: rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, MLResearchPress, 2019.

[12] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: a survey," *Journal of Machine Learning Research*, vol. 20, pp. 1–21, 2019.

[13] C. Ying, A. Klein, E. Real, E. Christiansen, K. Murphy, and F. Hutter, "NAS-Bench-101: towards reproducible neural architecture search," in *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, ML Research Press, 2019.

[14] X. Dong and Y. Yang, "NAS-Bench-201: extending the scope of reproducible neural architecture search," in *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*, ICLR, 2020.

[15] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 2, pp. 99–127, 2002.

[16] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, ICLR, 2017.

[17] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018)*, IEEE, 2018.

[18] L. Hanxiao, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, ICLR, 2019.

[19] S. Xie, A. Kirillov, R. Girshick, and K. He, "Exploring randomly wired neural networks for image recognition," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV 2019)*, IEEE, 2019.

[20] K. Kandasamy, W. Neiswanger, J. Schneider, B. Póczos, and E. P. Xing, "Neural architecture search with Bayesian optimization and optimal transport," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS 2018)*, Neural Information Processing Systems, 2018.

[21] B. Ru, X. Wan, X. Dong, and M. Osborne, "Interpretable neural architecture search via Bayesian optimisation with Weisfeiler-Lehman kernels," in *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*, ICLR, 2021.

[22] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "MCUNet: tiny deep learning on IoT devices," in *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, Neural Information Processing Systems, 2020.

[23] E. Liberis, L. Dudziak, and N. D. Lane, "μNAS: constrained neural architecture search for microcontrollers," in *Proceedings of the 1st Workshop on Machine Learning and Systems (EuroMLSys '21)*, ACM, 2021.

[24] M. Abadi, P. Barham, J. Chen, J. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, G. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, USENIX, 2016.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: an imperative-style, high-performance deep learning library," in *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, Neural Information Processing Systems, 2019.

[26] J. B. Dennis, "First version of a data flow procedure language," in *Programming Symposium*, Springer, 1974.

[27] W. M. Johnston, J. R. Paul Hanna, and R. J. Millar, "Advances in dataflow programming languages," *ACM Computing Surveys*, vol. 36, pp. 1–34, 2004.

[28] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, 1967.

[29] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: analysis and an algorithm," in *Proceedings of the 14th International Conference on Neural Information Processing Systems (NIPS '01)*, Neural Information Processing Systems, 2001.

[30] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, pp. 395–416, 2007.

[31] T. Hofmann, S. B., and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, pp. 1171–1220, 2008.

[32] N. M. Kriege, F. D. Johansson, and C. Morris, "A survey on graph kernels," *Applied Network Science*, vol. 5, pp. 1–42, 2020.

[33] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: convergence and generalization in neural networks," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS 2018)*, Neural Information Processing Systems, 2018.

[34] C. Liu, P. Dollár, K. He, R. Girshick, A. Yuille, and S. Xie, "Are labels necessary for neural architecture search?," in *Proceedings of the 16th European Conference on Computer Vision (ECCV 2020)*, Springer, 2020.

[35] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on ImageNet in four GPU hours: a theoretically inspired perspective," in *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*, ICLR, 2021.

[36] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, MLResearchPress, 2021.

[37] E. Amid, R. Anil, W. Kotłowski, and M. K. Warmuth, "Learning from randomly initialized neural network features," 2022.

[38] I. Radosavovic, J. Johnson, S. Xie, W.-Y. Lo, and P. Dollár, "On network design spaces for visual recognition," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV 2019)*, IEEE, 2019.

[39] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, IEEE, 2020.