ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

CNNGen: A Generator and a Dataset for Energy-Aware Neural Architecture Search

Antoine Gratia[1], Hong Liu[2], Shin'ichi Satoh[2],
Paul Temple[3], Pierre-Yves Schobbens[1] and Gilles Perrouin[1]

1- NaDI, Faculty of Computer Science, University of Namur, Belgium
2 - Digital Content and Media Sciences Research Division,
National Institute of Informatics, Tokyo, Japan
3 - University of Rennes, IRISA, Inria, Rennes, France

Abstract.
Neural Architecture Search (NAS) methods seek optimal networks by exploring thousands of variants of a reference architecture. Yet, optimality is typically related to prediction performance, overlooking the environmental impacts of training. Thus, NAS search spaces are unfit for performance and energy consumption trade-offs. We contribute to energy-aware NAS with *(i)* a grammar-based Convolutional Neural Network generator (CNNGen) producing diverse architectures not based on a reference one; *(ii)* 1,300 available architectures obtained via CNNGen with their implementation, energy consumption and performance measurements; *(iii)* Three state-of-the-art predictors releasing the need for trained models for performance and energy estimation.

1 Introduction

Neural Architecture Search (NAS) seeks the best CNN architecture for a task [1], solving an optimization problem over reference designs to find top-performing ones. To validate optimization approaches, techniques like NASBench-101 [2] provide datasets that benchmark thousands of architectures via cell-based generation. However, NAS's resource-intensive nature significantly impacts the environment, focusing mainly on performance. We propose CNNGen(Section 2), a grammar-based CNN generator exploring the search space without needing a reference architecture. CNNGen automates model creation and training, offering complete descriptions and reporting performance and energy metrics, supported by three novel prediction models (Section 3).

2 CNN Generator (CNNGen)

CNNGen uses the Xtext context-free grammar framework [3] to generate CNN architectures. The sequence of grammar tokens describes the CNN's topology (*i.e.,* the succession of layers). Our grammar captures the CNN domain knowledge to produce valid architectures. Thus, CNNGen differs from other NAS methods like NASBench [2] Indeed, CNNGen produces architectures from scratch and not as variants of existing ones. CNNGen also comes with an editor allowing to specify architectures. From a valid sequence of grammar tokens,

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

CNNGen automatically transforms this specification into the model's code using the Keras Python framework. CNNGen's grammar covers popular architectures (*e.g.,* LeNet, AlexNet, or ResNet) but can also yield unknown ones, allowing for more diversity in the search space. Our companion repository[1] describes our grammar and shows some examples of generated architectures. CNNGen computes the following metrics: accuracy, training time (in seconds), number of epochs and training parameters, emissions (CO2-equivalents [CO2eq], in kg), emissions rate (Kg/s), and total energy consumed (Kw). We use Code Carbon[2] to calculate measurements related to energy consumption. We used CNNGen to produce a benchmark of $1,300$ randomly generated CNN models. Finally, for each generated model, we release its CNNGen grammar phrase (*i.e.,* the succession of grammar tokens), PNG image representation, and Python code (using Keras). CNNGen source code and our benchmark are available on our companion repository to ease reproducibility and reuse.

3 Predictors

NAS methods rely on predictors to optimize architectures [4–9]. Some require retraining the models on new data to predict efficiency metrics, while others estimate the performance of a given model solely based on its structure [4]. Considering the impact of training [10], being able to predict accurately the performance of models without having to train is a step toward more sustainability. Thus, we propose three prediction models for performance and energy consumption that do not require trained models.

Our first predictor builds a relation between the graphical representation of an architecture (*i.e.,* a PNG image generated with a call to the Keras function *plot_model* [3]) and its classification performance. This graphical representation contains the succession of layers (and their parameters) forming a CNN.

Our second predictor relies on the Python code generated by CNNGen to run a specific CNN model based on its grammar phrase. This predictor focuses on the different architectural layers and extracts their hyperparameters. The goal here is to learn correlations between the layers and their hyperparameters on the one hand and the architecture's performance on the other hand.

Our third predictor relies on the Regressor Decision Tree and uses four (manually defined) features: the number of layers in the model, the number of epochs, the flops, and the number of parameters of the model.

4 Experiments

We evaluate the characteristics of our CNNGen benchmark, composed of $1,300$ models, relative to NASBench-101. This comparison focuses on performance and

[1] https://doi.org/10.5281/zenodo.11109244

[2] https://codecarbon.io/

[3] https://keras.io/api/utils/model_plotting_utils/

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

energy consumption. In addition, we also evaluate the relevance of the predictors described in Section 3. Due to space limitations, we provide a companion repository [4] containing all evaluation results.

4.1 Experimental Settings

In our experiments, we randomly selected 1,300 architectures from CNNGen and trained them on CIFAR-10, CIFAR-100, and Fashion-MNIST datasets with 36 epochs[5]. The datasets are divided into 80% for training and 20% for testing, with 20% training data used for validation. After training the different models, we compared the predictors (see Section 3) in energy and performance prediction tasks. We also added the *Neural Predictor for Neural Architecture Search* [11] as a baseline. To evaluate the different predictors, our previously trained architectures are split into 80% for training and 20% for testing, with 10% of the training set reserved for validation. We mitigate randomness effects by repeating our training and evaluation on ten independent runs.

4.2 How does CNNGen compare to NASBench regarding performance and energy distribution?

CNNGen and NASBench use different methods to generate architectures: CNNGen is a grammar-based approach, while NASBench relies on predefined cell-based architectures. This difference influences the diversity of topologies generated by each method, with CNNGen potentially offering a wider range of architectural structures tailored for image classification tasks. To assess performance dispersion, we compare the accuracy distribution of architectures generated by CNNGen (1,300 architectures) and NASBench (423,000 architectures) when evaluated on a test set. CNNGen architectures exhibit a broader range of accuracies (40% to 68%) compared to NASBench (80% to 85%), as illustrated by Figure 1. We conclude that CNNGen can produce a diversely performing and energy-consuming set of models, which we see as a prerequisite for complex trade-offs and generalizable predictors. NASBench-101 does not include energy metrics and it is impossible to compute them because of the lack of information to retrain such networks and the associated costs.

Figure 2 depicts the energy consumption distribution over the 1,300 models. For a given dataset, these diversified models (from 5 to 250 layers) have varied energy consumption. Energy consumption is also logically dataset-dependent.

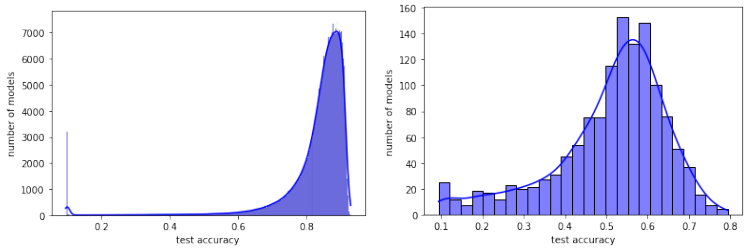
4.3 Performance of the energy/performance predictors

First, we want to compare the performance prediction (*i.e.,* accuracy) of our predictors before considering energy consumption prediction. We trained the four different predictors on the performance reported by the CNNGen architectures and we predict the performance for the remaining architectures. We use

[4] `https://doi.org/10.5281/zenodo.11109244`

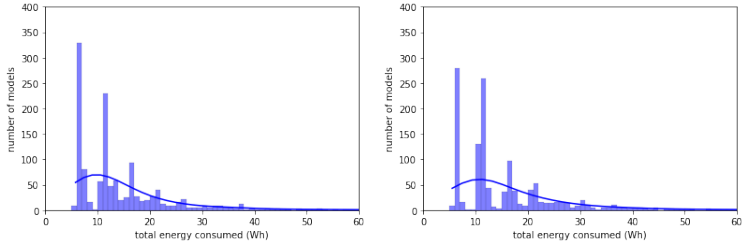
[5] This is one of the NASBench-101 settings, chosen to allow comparison.

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.



(a) (b)

Fig. 1: Distribution of accuracy of NASBench-101 models on CIFAR-10 (a) and 1,300 models generated with CNNGen on CIFAR-10 (b) on the test set after training models for 36 epochs.



(a) (b)

Fig. 2: Energy distribution for CIFAR-100 (a) and fashion-MNIST (b) over 1,300 models generated by CNNGen and trained for 36 epochs.

CIFAR-10, CIFAR-100, and F-MNIST to train and evaluate the CNNGen architectures. We report the mean average error (MAE) and Kendall's τ coefficient in Table 1 for each predictor, as well as the standard deviation. *Neural* is our baseline, *Img* is the predictor using information in the PNG image file, *Py_ code* uses information from the Python source code, and *DT* is the decision tree predictor using only 4 features. The *Py_ code* predictor performs better than the others, showing lower MAE and higher Kendall's τ coefficient. Interestingly, the *Img* performs comparably to our baseline (*i.e., Neural*).

Regarding the prediction of carbon emissions, using carbon emission data from CodeCarbon, we train our predictors on 80% of 1,300 CNNGen architectures and evaluate their prediction ability on the remaining architectures. Again, Table 2 reports the average and standard deviation over all the execution of the MAE and Kendall's tau coefficient for the different predictors. Note that, this time, since previous work did not report carbon emissions, we do not have any baseline. Our results show that the decision tree predictor outperforms others, exhibiting lower MAE and higher Kendall's *tau* coefficient across all datasets. Mann-Whitney tests confirm significant differences in prediction distributions,

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

	CIFAR-10	**CIFAR-100**	**F-MNIST**
Kτ			
Neural	$1.0e^{-1} \pm 3.1e^{-2}$	$1.3e^{-1} \pm 2.2e^{-2}$	$7.8e^{-2} \pm 2.8e^{-2}$
Img	$6.7e^{-2} \pm 2.1e^{-2}$	$4.7e^{-2} \pm 2.8e^{-2}$	$2.4e^{-2} \pm 1.4e^{-2}$
Py_code	$2.6e^{-1} \pm 2.9e^{-2}$	$3.4e^{-1} \pm 4.4e^{-2}$	$1.8e^{-1} \pm 3.7e^{-2}$
DT	$1.8e^{-1} \pm 1.1e^{-1}$	$1.5e^{-1} \pm 4.2e^{-2}$	$1.0e^{-2} \pm 7.6e^{-2}$
MAE			
Neural	$1.4e^{-1} \pm 1.5e^{-2}$	$8.9e^{-2} \pm 8.9e^{-3}$	$1.6e^{-1} \pm 1.9e^{-2}$
Img	$1.1e^{-1} \pm 6.9e^{-3}$	$8.2e^{-2} \pm 2.6e^{-3}$	$1.0e^{-1} \pm 1.1e^{-2}$
Py_code	$1.0e^{-1} \pm 9.5e^{-3}$	$6.8e^{-2} \pm 3.0e^{-3}$	$1.1e^{-1} \pm 1.2e^{-2}$
DT	$1.0e^{-1} \pm 7.3e^{-3}$	$7.6e^{-2} \pm 2.0e^{-3}$	$1.0e^{-1} \pm 3.8e^{-3}$

Table 1: Kendall τ and MAE for performance prediction

especially when comparing the decision tree predictor to others, with reported p-values of $1.8e^{-46}$. These findings suggest the potential of our predictors, particularly the decision tree predictor that considers only a few features, as proxies for carbon emissions prediction.

	CIFAR-10	**CIFAR-100**	**F-MNIST**
Kτ			
Img	$4.2e^{-1} \pm 3.6e^{-1}$	$4.7e^{-1} \pm 3.2e^{-1}$	$6.0e^{-1} \pm 1.8e^{-2}$
Py_code	$3.9e^{-2} \pm 9.1e^{-2}$	$1.2e^{-1} \pm 1.6e^{-1}$	$6.1e^{-2} \pm 1.8e^{-1}$
DT	$7.9e^{-1} \pm 2.1e^{-2}$	$7.7e^{-1} \pm 7.6e^{-3}$	$8.3e^{-1} \pm 1.3e^{-2}$
MAE			
Img	$1.0e^{-3} \pm 4.6e^{-4}$	$1.0e^{-3} \pm 2.5e^{-4}$	$1.2e^{-3} \pm 2.1e^{-4}$
Py_code	$1.5e^{-2} \pm 7.5e^{-3}$	$1.5e^{-2} \pm 6.7e^{-3}$	$2.0e^{-2} \pm 8.1e^{-3}$
DT	$3.3e^{-4} \pm 3.7e^{-5}$	$4.0e^{-4} \pm 4.5e^{-5}$	$4.4e^{-4} \pm 2.6e^{-5}$

Table 2: Kendall τ and MAE for energy prediction

5 Conclusion

In this paper, we introduce CNNGen, a context-free grammar-based architecture generator designed to tackle the challenges of generating diverse Convolutional Neural Network (CNN) architectures for image classification tasks within Neural Architecture Search (NAS). While previous approaches focused on performance, such as accuracy or latency, CNNGen also monitors environmental impacts such as energy consumption of these models, paving the way for greener NAS methods. CNNGen is easily extendable and reusable. We put effort into offering

[6]Results of statistical tests are available on the companion repository.

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

exhaustive data with our generated CNN architectures (grammar phrase, PNG image, Python code, and measures). Along with CNNGen and its 1,300 architectures, we propose 3 different performance predictors. While our code-based predictor is appropriate for performance prediction, a simple Decision Tree predictor using 4 features can handle energy consumption with a high accuracy. In the future, we would like to use our predictors in the context of NAS methods.

6 Acknowledgement

We would like to thank Valentin Delchevalerie for his help proofreading this paper. This work was supported by Service Public de Wallonie Recherche under grant n° 2010235 – ARIAC by DIGITALWALLONIA4.AI. Gilles Perrouin is an FNRS Research Associate.

References

[1] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. ACM, 2019.

[2] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-bench-101: Towards reproducible neural architecture search. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7105–7114, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[3] Lorenzo Bettini. *Implementing Domain Specific Languages with Xtext and Xtend - Second Edition*. Packt Publishing, 2nd edition, 2016.

[4] Xiangning Xie, Xiaotian Song, Zeqiong Lv, Gary G. Yen, Weiping Ding, and Yanan Sun. Efficient evaluation methods for neural architecture search: A survey, 2023.

[5] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[6] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. Few-shot neural architecture search. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12707–12718. PMLR, 18–24 Jul 2021.

[7] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 550–559. PMLR, 10–15 Jul 2018.

[8] Mohamed S. Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D. Lane. Zero-Cost Proxies for Lightweight NAS. In *International Conference on Learning Representations (ICLR)*, 2021.

[9] Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance deep image recognition. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021*, 2021.

[10] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics.

[11] Wei Wen, Hanxiao Liu, Hai Li, Yiran Chen, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. *CoRR*, abs/1912.00848, 2019.