Beyond Graph-Based Modeling for Hierarchical Neural Architecture Search

Lum Birinxhiku[1] **Danny Stoll**[1] **Simon Schrodi**[1] **Frank Hutter**[1]

[1]University of Freiburg

Abstract Neural Architecture Search (NAS) seeks to automate the discovery of well-performing neural architectures. Recently, a hierarchical approach to NAS (hNAS) has been shown to allow for high search space expressiveness and efficient searching. However, BOHNAS, the current best strategy for hNAS requires the conversion of sampled networks from their native string representations to graphs, complicating the extension of hierarchical search spaces to include not only architectures, but also other pipeline components, such as hyperparameters, learning rate schedules and data augmentation. In this work, we introduce hNASK, a string kernel that operates on such string representations, is able to take advantage of the hierarchical structure of the search space and preserves the performance of BOHNAS on the performed architecture search experiments. As such, this kernel opens the door for future work in hNAS without being constrained to graph-based modeling of search spaces. Code is available at `https://github.com/automl/hnas_with_string_kernels`.

1 Introduction

The field of Neural Architecture Search (NAS) is concerned with the automatic discovery of neural architectures [1]. The difficulty in solving this problem arises due to the potential size of the search space and the constraints of time and compute resources. Researchers also try to find ways of defining search spaces with high expressive power that can be searched efficiently [2].

Considering that traditional search spaces are lacking in expressive power, Schrodi et al. [3] have proposed a way of constructing hierarchical search spaces where sampled networks are formulated as strings. For such spaces, they propose a search strategy, BOHNAS, and show promising results.

However, their way of defining the search space only allows architecture search and offers no way to also specify other pipeline components, such as hyperparameters [4, 5], learning rate schedules [6, 7], or data augmentation [8, 9]. The main challenge to adding such functionality is that both kernels currently used, the hWL [3] and the WL kernel [10, 11], require an explicit conversion of sampled networks into graphs. The need for this conversion increases the modeling difficulty for researchers, due to the more restricted expressive capabilities of graphs compared to simple strings.

To facilitate future developments which extend the approach into more of a neural pipeline search, there is a need for string kernels which would take as input sampled networks in their native string representation and are then able to perform similar computations to the graph based kernels, while also allowing for more information to be encoded in the strings, which could then be used to improve the search performance.

Our contributions. We summarize our key contributions below:

1. We present a string based kernel which can be used with the BOHNAS approach - the Neural Architecture String Kernel (NASK).
2. We show that hierarchical NASK (hNASK) is able to take advantage of the hierarchical nature of the search space to improve the search performance.

 © 2024 the authors, released under CC BY 4.0

3. We show that hNASK achieves similar performance to the hWL kernel across all considered datasets.

4. We adhere to the NAS best practice checklist [12] and provide code at `https://github.com/automl/hnas_with_string_kernels` under an MIT License.

2 Existing kernels

The BOHNAS approach can be modified to use kernels that do not rely on graphs, but instead are string based and take as input two strings, each containing the algebraic representation of a network. There already exist string kernels which are used for example in text classification and in bioinformatics. Gärtner [13] gives examples of such kernels and separates them into two categories - kernels which operate on a character sequence level and base their work on common character subsequences and kernels that use a 'bag-of-words' representation of their inputs for their computations. Common subsequence character based kernels are inappropriate here due to us dealing with words/symbols which can be related to each other even when they are not in proximity in the string. A bag-of-words kernel could work, provided it is able to take into account any structural information inherent to the problem, present in the given strings. We choose to construct such a kernel.

3 A string kernel for BOHNAS

To simplify future extensions in search space expressiveness, we seek to avoid needing to convert sampled networks from their native string representations into graphs. Searching such spaces using BOHNAS then requires the use of a kernel taking such string representations as input.

In Section 3.1 we describe how the proposed kernel NASK works. Then, in Section 3.2 we consider how the hierarchical nature of the search space can be used to improve its searching performance.

3.1 NASK

The proposed string kernel NASK computes its result by first embedding the two input network strings into two vectors containing non-negative values and then computing their cosine similarity.

To show how the embedding of a network into a vector is done, we use the following example network:

(Sequential(Residual(conv)(id)(conv))(Residual(conv)(id)(conv))(fc))

First the network is decomposed into parts according to three categories: operator (the operations done), operands (the sub-networks on which the operation is done), and the joined operator and operands string.

For the example network given above, there are in total 9 parts present:

The operator parts are: "id", "fc", "conv", "Sequential", "Residual"
The operand parts are:

"(conv)(id)(conv)", "(Residual(conv)(id)(conv))(Residual(conv)(id)(conv))(fc)"

The joined operator-operand parts are:

"Residual((conv)(id)(conv))",
"Sequential((Residual(conv)(id)(conv))(Residual(conv)(id)(conv))(fc))"

The network is then embedded into three vectors, each in $\mathbb{R}^9$ and in which each part is mapped to the same position based on the part length in characters (increasing) with values referring to the frequency how many times that part is present in the network in that capacity. In our example:

$$\begin{aligned}
&\text{Operator frequencies:} && < 2, 1, 4, 1, 2, 0, 0, 0, 0 > \\
&\text{Operand frequencies:} && < 0, 0, 0, 0, 0, 2, 0, 1, 0 > \\
&\text{Joined operator-operand frequencies:} && < 0, 0, 0, 0, 0, 0, 2, 0, 1 >
\end{aligned}$$

Finally, the vectors are multiplied by their respective non-negative weight and are added together to give the result vector representing the network.

3.2 Hierarchical NASK (hNASK)

As given, NASK considers the appearances of parts without regard for location. Furthermore, the operand and joined operator-operand parts can be very specific when the network is deep, causing less likelihood for matches between the networks given as input. This can lead to results which provide little information.

Using the same approach as hWL, we try to overcome such limitations by computing the kernel multiple times, for example over all hierarchy levels of the input networks. Then, we weigh the individual kernel results by their respective non-negative weight and sum them into a final kernel result, preserving the properties needed for use in a GP.

For larger and deeper networks, by computing the kernel over multiple hierarchy levels and combining the results (hNASK) it is expected that the final result will better take into account location information of the parts and also better take into account structure similarities from the higher levels of the networks. Indeed, surrogate experiments in Section 4.1 show this to be the case.

4 Experiments

We compare the performance of hNASK as proposed to the WL and hWL graph kernels in multiple experiments. Matching time and compute resources, we use the same experiment setup, data and hyperparameter learning process as Schrodi et al. [3].

The datasets used are also kept the same and they are CIFAR-10 [14], CIFAR-100 [14], ImageNet16-120 [15], CIFARTile [16], and AddNIST [16].

4.1 Surrogate experiments

The purpose of these experiments is to compare the performance of the GP surrogate [10] across kernels. Initially we compare the proposed string kernel using only one hierarchy level (the full network description), using four hierarchies (the most-detailed ones) and using all seven available hierarchy levels of the networks. Then, we compare the best performing version to the hWL and WL kernels.

The results in Figure 1 show hNASK performs better than the NASK versions with limited or no hierarchy use, particularly as the number of training samples is smaller. When compared to hWL, hNASK has a worse performance when the number of training samples is smallest, but quickly catches up and surpasses it, continuing to improve as more training samples are made available.

4.2 Activation function search experiment

We perform neural architecture search with the goal of finding better performing activation functions using the same setup as Schrodi et al. [3] and Ramachandran et al. [17]. The results of the experiment are given in Table 1. They show that hNASK and hWL achieve very similar test and validation error scores.

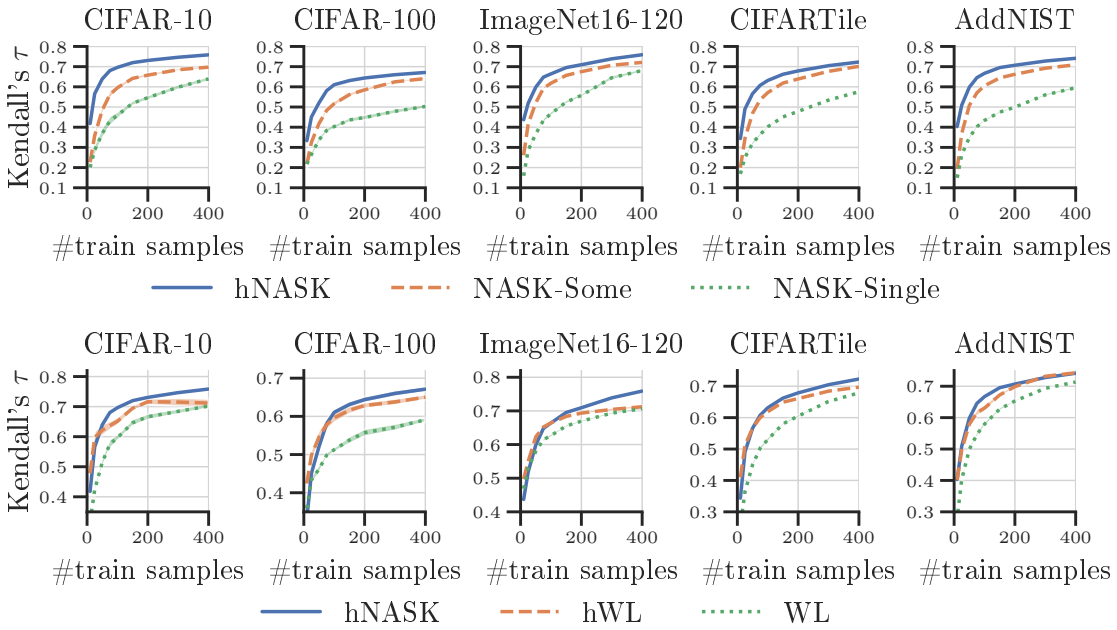

Figure 1: Mean Kendall's τ correlation with ±1 standard error achieved by a GP using the kernel.

Table 1: Test and validation errors [%] of the architectures with lowest training validation error for the activation function search experiment

Dataset	Training prot.	hNASK (val)	hWL (val)	hNASK (test)	hWL (test)
CIFAR-10	Act	7.05±0.06	7.19±0.02	8.63±0.01	8.61±0.03

4.3 Hierarchical NAS-Bench-201 search experiments

Next, we study hNASK used in Bayesian optimization on multiple datasets using Schrodi et al.'s modified hierarchical version [3] of the NAS-Bench-201 search space [18]. As Table 2 shows, hNASK and hWL have similar performance both in the validation and test errors, with hNASK having a slight edge on most datasets.

Table 2: Test and validation errors [%] of the architectures with lowest training validation error

Dataset Training protocol	CIFAR-10 NB201†	CIFAR-100 NB201†	ImageNet16-120 NB201†	CIFARTile NB201†	AddNIST NB201†
hNASK (val)	8.52±0.19	27.29±0.64	52.39±0.17	**27.14**±0.73	7.11±0.09
hWL (val)	8.55±0.33	27.44±0.37	52.51±0.74	29.48±0.81	6.91±0.18
hNASK (test)	8.92±0.23	28.10±0.54	53.63±0.67	**32.27**±0.97	6.49±0.11
hWL (test)	8.98±0.37	28.23±0.49	53.59±0.98	34.42±1.47	6.64±0.20

† includes hierarchical search space variants.

5 Limitations

The proposed kernel is able to take advantage of the information encoded in the string representations of networks both at a single hierarchy and over multiple ones, performing similarly to the previously used hWL kernel in the searching experiments. However, looking at the surrogate

experiments in Figure 1, the kernel seems to have a weaker performance in the initial stages when a very small number of training samples is available. One aspect that could be changed to improve this is in how the part strings are compared to each other. Allowing partial matches which would contribute fractionally to the part frequencies in the embedding vector could allow for more information to be collected into the embedding vectors. Additionally, while the kernel is appropriate for the current structure of strings, as this structure is extended to allow for more kinds of information (e.g. hyperparameters), the kernel might have to be adjusted, for example by adding new kinds of parts extracted and embedded into the intermediate vectors.

6 Broader impact statement

After careful reflection, the authors have determined that this work presents no notable negative impacts to society or the environment. While neural architecture search has a high carbon footprint if executed inefficiently, this work aims to make it more efficient, thereby reducing this carbon footprint.

7 Conclusion

We introduced a new string based kernel hNASK, which is compatible with the BOHNAS approach for hierarchical NAS and is able to leverage hierarchical information to improve its performance. In the Bayesian optimization experiments hNASK performed similarly or better than hWL on all the tested datasets, while in the surrogate experiments it had a relatively weaker performance when a very small number of training samples was available, but a better performance as their count increased. Having this kernel simplifies future work in extending the expressiveness of hierarchical search spaces by removing the need for a conversion of the sampled networks from their native string representation into graphs, as previously required by the hWL kernel. Furthermore, we offered suggestions about how adjustments can be done to the kernel to accommodate such possible future enhancements.

Acknowledgements. We acknowledge funding by the European Union (via ERC Consolidator Grant DeepLearning 2.0, grant no. 101045765). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] T. Elsken, J. Metzen, and F. Hutter. Neural Architecture Search: A survey. *JMLR*, 20(55):1–21, 2019.

[2] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

[3] S. Schrodi, D. Stoll, B. Ru, R. Sukthanker, T. Brox, and F. Hutter. Construction of hierarchical neural architecture search spaces based on context-free grammars. In *Advances in Neural Information Processing Systems*, 2023. URL `http://lmb.informatik.uni-freiburg.de/Publications/2023/SB23`.

[4] F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In C. Coello, editor, *Proceedings of the Fifth International Conference on Learning and Intelligent Optimization (LION'11)*, volume 6683 of *Lecture Notes in Computer Science*, pages 507–523. Springer-Verlag, 2011.

[5] S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient Hyperparameter Optimization at scale. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*, volume 80, pages 1437–1446. Proceedings of Machine Learning Research, 2018.

[6] Yasusi Kanada. Optimizing neural-network learning rate by using a genetic algorithm with per-epoch mutations. In *2016 international joint conference on neural networks (IJCNN)*, pages 1472–1479. IEEE, 2016.

[7] Zhen Xu, Andrew M Dai, Jonas Kemp, and Luke Metz. Learning an adaptive learning rate schedule. *arXiv preprint arXiv:1909.09712*, 2019.

[8] Teppei Suzuki. Teachaugment: Data augmentation optimization using teacher knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10904–10914, 2022.

[9] Xi Peng, Zhiqiang Tang, Fei Yang, Rogerio S Feris, and Dimitris Metaxas. Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2226–2234, 2018.

[10] B. Ru, X. Wan, X. Dong, and M. Osborne. Interpretable Neural Architecture Search via Bayesian Optimisation with Weisfeiler-Lehman kernels. In *Proceedings of the International Conference on Learning Representations (ICLR'21)*, 2021. URL `https://openreview.net/forum?id=j9Rv7qdXjd`. Published online: `iclr.cc`.

[11] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[12] R. Garnett, F. Hutter, M. Lindauer, and J. Gardner. AutoML-Conf LatexTemplate. `https://github.com/automl-conf/LatexTemplate`, 2024.

[13] Thomas Gärtner. A survey of kernels for structured data. *ACM SIGKDD explorations newsletter*, 5(1):49–58, 2003.

[14] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[15] P. Chrabaszcz, I. Loshchilov, and F. Hutter. A downsampled variant of ImageNet as an alternative to the CIFAR datasets. *arXiv:1707.08819 [cs.CV]*, 2017.

[16] Rob Geada, Stephen McGough, A Abarghouei Amir, Isabelle Guyon, and Sébastien Treguer. CVPR-NAS-Datasets (codebase for the "CVPR-NAS 2021 Unseen Data Track"), 2021.

[17] P. Ramachandran, B. Zoph, and Q. Le. Searching for activation functions. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*, 2018. Published online: `iclr.cc`.

[18] X. Dong and Y. Yang. NAS-Bench-201: Extending the scope of reproducible Neural Architecture Search. In *Proceedings of the International Conference on Learning Representations (ICLR'20)*, 2020. Published online: `iclr.cc`.

Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   (b) Did you describe the limitations of your work? [Yes] Yes, we include a limitations section.

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We include a broader impact section, the nature of the work has no direct societal impact.

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? `https://2022.automl.cc/ethics-accessibility/` [Yes] Yes.

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? [Yes] For all experiments, the same evaluation protocols were used across methods.

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [Yes] We specify that we used the same specifications as used the existing BOHNAS experiments.

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [Yes] Yes, all experiments were ran on multiple seeds, according to the available compute capabilities and time.

   (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [Yes] In our experiment results, we give the standard error of the observed results across the used seeds.

   (e) Did you report the statistical significance of your results? [No] No, but we provide the standard error values for the observed results, suggesting clear statistical significance.

   (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [Yes] We use surrogate benchmarks to judge the performance of our kernel compared to the existing kernels and also depending on the hierarchy information allowed.

   (g) Did you compare performance over time and describe how you selected the maximum duration? [Yes] In the surrogate experiments we give the performance of the kernels across a larger horizon. The values were chosen in accordance with previous experiments.

   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The amount and type of compute for all experiments is also similar as in the previously existing BOHNAS experiments.

   (i) Did you run ablation studies to assess the impact of different components of your approach? [Yes] We use ablation studies to judge the effect of having more hierarchy information available on the NASK performance.

3. With respect to the code used to obtain your results...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] Yes, in the provided code repository.

(b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes] Yes, we provide instructions on how to run the surrogate experiments, which can be used for such purposes.

(c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes] Yes, and we provide the commands needed to run the experiments.

(d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes] Yes, instructions on accessing the raw results are in the provided code repository.

(e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes] Yes, code for generating the figures and values for the tables is included in the provided code repository.

4. If you used existing assets (e.g., code, data, models)…

(a) Did you cite the creators of used assets? [Yes] We cited the creators of the datasets we used and also the relevant code for the search experiments.

(b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [N/A] No consent was required for the used datasets.

(c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] No such information is contained in the used datasets.

5. If you created/released new assets (e.g., code, data, models)…

(a) Did you mention the license of the new assets (e.g., as part of your code submission)? [Yes] We release our code under an MIT License as specified in the abstract.

(b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [Yes] The assets were linked in the abstract.

6. If you used crowdsourcing or conducted research with human subjects…

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] No such research was conducted.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] No such research was conducted.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] No such research was conducted.

7. If you included theoretical results…

(a) Did you state the full set of assumptions of all theoretical results? [N/A] No theoretical results were included.

(b) Did you include complete proofs of all theoretical results? [N/A] No theoretical results were included.

A Dataset licenses

Table 3: Licenses and URLs for the datasets used in the experiments

Dataset	License	URL
CIFAR-10 [14]	MIT	https://www.cs.toronto.edu/~kriz/cifar.html
CIFAR-100 [14]	MIT	https://www.cs.toronto.edu/~kriz/cifar.html
ImageNet-16-120 [15]	MIT	https://patrykchrabaszcz.github.io/Imagenet32/
CIFARTile [16]	GNU	https://github.com/RobGeada/cvpr-nas-datasets
AddNIST [16]	GNU	https://github.com/RobGeada/cvpr-nas-datasets