Generating Neural Network Architectures with Conditional Graph Normalizing Flows

Lichuan Xiang[1] **Łukasz Dudziak**[2] **Abhinav Mehrotra**[2] **Mohamed S. Abdelfattah**[3]
Nicholas D. Lane[2,4] **Hongkai Wen**[1,2]

[1]University of Warwick, UK
[2]Samsung AI Center Cambridge, UK
[3]Cornell University, USA
[4]University of Cambridge, UK

Abstract Neural Architecture Search (NAS) is gaining popularity in automating designing deep neural networks for various tasks. A typical NAS pipeline starts with a manually designed search space, narrowed down to promising subspaces through NAS, allowing the discovery of high-performance models. However, those models are still strictly included in the initial manually-specified search space, whose quality may be a limiting factor for the final NAS performance. A prohibitively ample search space may make NAS significantly more time-consuming, while a smaller search space may fail to include top-performing models. This paper develops new generative methods based on graph-normalizing flows that can generate high-performing neural network architectures that do not belong to a given search space. We leverage information about known high-performing reference models from prior knowledge, for example, manually designed state-of-the-art models, to condition our generative model to generate high-performing architectures with similar characteristics. We show that our approach can discover better architectures beyond the scope of well-studied NAS search spaces.

1 Introduction

Automated search space design methods can be seen as mappings from one search space to another, $S \rightarrow A$, where typically $|A| \ll |S|$ and $A \subset S$. Compared to performing NAS directly on S, they are more efficient due to the relaxed objective in the first stage, effectively resulting in hybrid systems combining coarse and fine-grained searching. However, existing approaches that focus on pruning or evolving a large initial search space S often rely on repeating a variation of NAS process multiple times to provide feedback for updating S. This is also subject to bias, e.g. when assessing the fitness of different subspaces, and will almost certainly result in non-negligible additional cost Zhou et al. (2021); Ci et al. (2021).

In contrast to the existing methods, our goal is to design a method of automatically creating search spaces A, s.t. (possibly) $S \cap A = \emptyset$, that all architectures in A mainly satisfy our user conditions. We can obtain those conditions from a seed search space that allows us to incorporate any prior knowledge efficiently in the form of labelled architectures. The sampling mechanism is realized and learned using a combination of a Graph Variational Autoencoder (G-VAE), a Conditional Continuous Normalizing Flow (CCNF), and Zero-Cost (ZC) proxies Abdelfattah et al. (2021), which all are adapted and/or extended to suit our needs. In summary, the contributions of our work are:

- We perform semi-supervised learning of a G-VAE to have a reversible encoding of graphs to a latent space that preserves the clustering of the ZC space.
- We incorporate a CCNF model to learn to navigate the latent space of the G-VAE efficiently, resulting in an efficient search space design mechanism supporting user-defined conditioning.

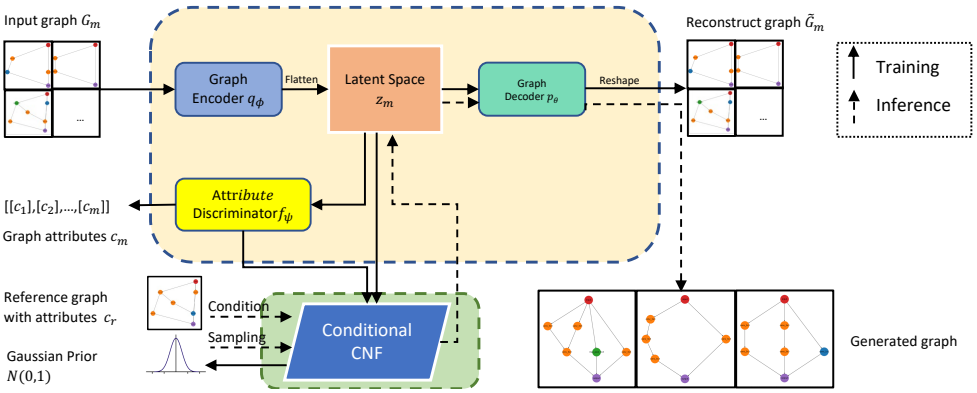

Figure 1: Our approach performs semi-supervised learning of a G-VAE to have a reversible encoding of graphs to a latent space using triplet loss on ZC scores. A CCNF model is then used to navigate the learned latent space and generate novel architectures given reference and user-defined conditions.

- We evaluate our method on popular NAS search spaces, including NAS-Bench-101 Ying et al. (2019), NAS-Bench-201 Dong and Yang (2020) (acting as seed search spaces), and also explored generalizability to diverse tasks by transferring to NAS-Bench-360 Tu et al. (2022). Our approach presents successful extrapolation to generate novel architectures beyond original search spaces.

2 Related Work

NAS Search Space Design: The automated optimization of NAS search spaces has been a focus of the NAS community Radosavovic et al. (2020); Zhou et al. (2021); Hu et al. (2021, 2020). RegNet Radosavovic et al. (2020) and NSE Ci et al. (2021), for example, refine NAS search spaces by constraining the design space or evolving it, respectively. AutoSpace Zhou et al. (2021) employs evolutionary algorithms to optimize from an open architecture space. Despite methodological differences, these approaches primarily aim to prune the initial NAS search space to contain high-performing models.
Flow-based Generative Models: Normalizing flows Papamakarios et al. (2021) can map complex to simple distributions and have seen successful application across various tasks, such as image editing Abdal et al. (2021) and point cloud generation Yang et al. (2019). This work leverages the efficiency and invertibility of Continuous Normalizing Flows (CNF)Grathwohl et al. (2018), based on neural ODEs Chen et al. (2018), and conditional-CNF (CCNF) models, which can control the properties of the output by concatenating parameters to the embeddings. For example, StyleFlow Abdal et al. (2021) and SRFlow Lugmayr et al. (2020) use CCNF for attribute-semantic image edits and high-resolution image generation.

3 Method

Unlike the NAS search space design work mentioned above, our approach is fundamentally different: instead of pruning or shrinking an initial search space, we generate optimized architectures given a reference model and the desired conditions, e.g. zero-cost scores, while in practice, the generated architectures may lie beyond the scope of the search space associated with the reference. Moreover, our work resembles CCNF's use to generate new deep-learning models given a reference and different conditions. However, to our knowledge, we are the first to show that employing CCNF generates optimized neural architectures. The overview of our method is presented in Figure 1, and the following sections describe each part in detail.

3.1 Semi-supervised latent-feature and graph pseudo attributes discriminative model

The Variational AutoEncoder (VAE) clusters samples in latent space based on observations, though our graph representations currently fall short in reflecting network performance. To improve this, we introduce auxiliary tasks in G-VAE training to better capture zero-cost score information.

Computing zero-cost scores for every graph in a vast design space is infeasible. Thus, we focus on a subset with corresponding zero-cost scores. Represented as $p_a(G, c)$ and $p_u(G)$, these labelled and unlabeled subsets, respectively, form empirical distributions. We aim to develop semi-supervised learning models that leverage these distributions, thereby improving attribute prediction performance.

During G-VAE training, we introduce a triplet margin loss as an auxiliary task. This loss function is designed to regularize the latent space, thus facilitating the clustering of models with similar distance relationships in the graph attribute space. The triplet margin loss function computes the loss by iterating over selected triplets and measuring the difference between the anchor-positive and anchor-negative distances. The overall loss is then calculated as the sum of the rectified differences divided by the total number of triplets. This loss function strives to minimize the distance between the anchor and positive samples while maximizing the distance between the anchor and negative samples. The margin parameter, m, enforces a minimum separation between these two distances, thereby preserving the desired distance relationships throughout the learning process.

As the triplet margin loss guides the G-VAE to cluster graphs with similar zero-cost scores together in latent space, we can simultaneously train a separate attribute predictor based on latent space representation, z, in a semi-supervised manner. This model is designed to predict an attribute, c, facilitating a more cost-efficient attribute evaluation for all possible graphs. This approach contrasts with traditional methods that demand extensive resources, such as creating actual network instances and measuring attributes through numerous forward and backward passes.

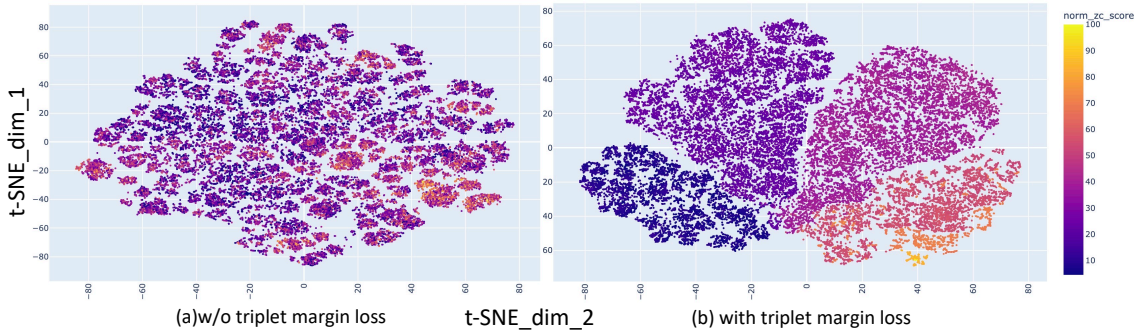

Figure 2: Comparison on Graph-VAE latent space with**(right)** and without **(left)** reference triplet margin loss. With triplet margin loss, latent space tends to cluster graphs with similar attributes well, which allows the semi-supervised attributes discriminator to achieve accurate prediction and helps CCNF better sample each subset of design space under different conditions. The latent space is projected to 2-dimension by using t-SNE for visualization

3.2 Flow-based graph generation from zero-cost score

Leveraging the previously trained graph Variational AutoEncoder (VAE) and the graph attribute discriminator, we can obtain the latent representation, z_i, and the estimated attribute, $\hat{c}_i$, for any graph, G_i, within the design space. This allows our Conditional Continuous Normalizing Flows

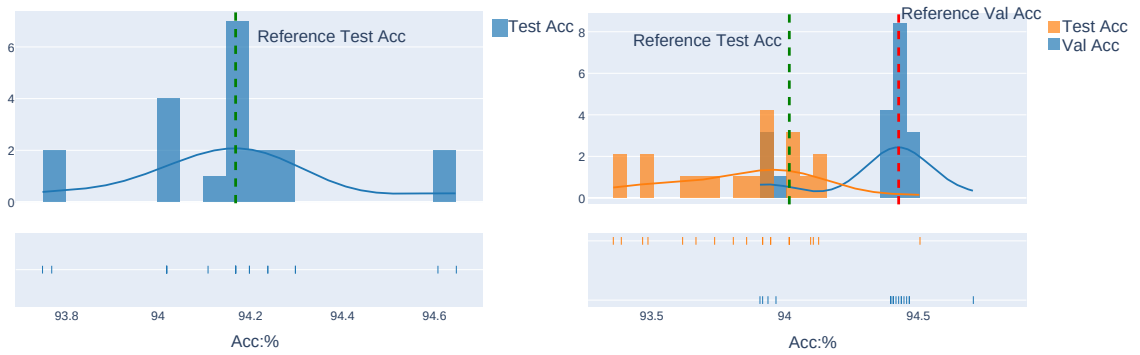

Figure 3: **(left)**Sampled models trained accuracy on CIFAR-10 in NB201-like Design Space. **(right)** Sampled models trained accuracy on CIFAR-10 in NB101-like Design Space

(CCNF) to model the conditional distribution on the latent representation by sampling any graphs from the design space. During this process, the CCNF model lets us sample random variables from a standard normal distribution, $\mathcal{N}(0, 1)$, with the target attribute as the conditioning factor. This enables the generation of graphs similar to the target attributes.

4 Evaluation

This section evaluates our method's ability to generate novel architectures beyond the original seed search space. Specifically, we focus on testing whether we can find more accurate and/or efficient models than the best ones in any of the considered seed search spaces (subsections 4.3. We also test how sampled search spaces behave when tested on the diverse tasks introduced in the recent NAS-Bench-360 Tu et al. (2022).

4.1 Design Space

We use NAS-Bench-101 (NB101) and NAS-Bench-201 (NB201) as seed search spaces since they provide us with performance information about all networks, allowing us to perform detailed comparisons between the seed and optimized search spaces. NB101 is a search space of approximately 420k architectures, while NB201 contains roughly 15k. For NB101 settings, we consider a similar graph definition with five intermediate vertex nodes and remove the edge number constraint at 9, resulting in 37748736 different graphs as NB101-like design space. For NB201, the extended search space includes all possible adjacency matrices with six intermediate nodes resulting in 4831838208 possible graphs as NB201-like design space.

In both cases, optimized search spaces, sampled from the CCNF, are kept small – usually lower than 20 samples per a single reference model to make search space concise.

4.2 Performance on pseudo attributes discriminator

We analyse our pseudo attributes discriminator to evaluate if our semi-supervised task performed well on unseen graphs. Specifically, we used graphs from the original NB201 and NB101 as a test set which are not explicitly trained with graph-truth attributes in our training process. Considering NB101 is relatively large, we only sampled 4k graphs from it and obtained ground truth zero-cost values. For NB201, we used all possible 15284 graphs to create runnable networks from benchmarks. Additionally, we compared the zero-cost score with the percentage of the maximum zero-cost score we obtained in the training set, which is also the value we used to normalize our elements during the training process.

In NB201, the average predicted error in the zero-cost score is 0.91% of the maximum zero-cost score in the train set, while the maximum error is 9.65% which is lower than 10%. For NB101, we have a similar average error at 0.78%, while the maximum error is much lower at 5.26%.

4.3 Pushing the upper-bound of the seed search space

Considering the most straightforward case for seed search space, that uses a single best model from known search space. We take the best existing model from NB201 and NB101 as the seed search space. To compare the reference and sampled models fairly and avoid installation bias introduced by different implementations. We retrained the reference model and sampled models with the same random seeds and strictly followed the training procedure from the original paper.

In Fig.3, we can observe that our approach sampled model performs relatively similar to the reference model in NB201-like and NB101-like design space. Since we are exploring diverse graphs with similar attributes, means we are exploring the success reference model in latent space and finding better potential better models. Fig.3 shows that with only 20 sample sizes, we can push the upper bound of the accuracy from seed search space.

Dataset	Reference	Mean	Std	Best	Obj.
spherical	39.15	37.08	2.279	**39.63**	max
darcy-flow-5	0.033	0.027	0.006	**0.017**	min
psicov	3.20	3.32	0.272	**3.05**	min
cosmic	0.16	0.18	0.037	**0.14**	min
deepsea	39.74	39.58	0.375	**39.94**	max
ninapro	90.04	90.85	1.382	**92.58**	max
ecg	0.65	0.64	0.015	**0.67**	max
satellite	87.96	87.22	0.657	**88.04**	max

Table 1: Performance on NB360

Generalizability to diverse tasks. We transferred our search model to diverse tasks and compared them with the previous best model found in NB201. We can observe in Table.1 that by searching around the best reference model with generated search space at size 20, we consistently improved all tasks performed in NB360 compared to directly using the best network reported in NB201

5 Conclusion

In this paper, we propose a novel method to generate optimized neural network architectures given known reference models and user-defined conditions. Instead of pruning or evolving a pre-defined initial NAS search space, our approach firstly extends the original search space to a large extended search space and performs semi-supervised learning of a Graph VAE (G-VAE) to obtain a reversible encoding of graphs from the extended search space to a latent space. During the training of the G-VAE we incorporate a new Tiplet Margin Loss on the Zero-Cost proxies to regularize the latent space, preserving the desired clustering properties that models with similar performance tend to be close. We then employ a Conditional Continuous Normalizing Flow (CCNF) model to learn to efficiently sample from the latent space given a known strong-performing reference model and user-defined conditions and generate a set of novel architectures beyond the scope of the original search space. We show that our approach can discover better models than the very best ones in popular NAS search spaces and on diverse tasks, including NAS-Bench-101, NAS-Bench-201, and NAS-Bench-360.

6 limitation

In this paper, our focus is to show that using G-VAE and CCNF is an efficient and effective way to generate desired search space that satisfied user input conditions. We extended the NB201 and NB101 design space by removing all hard constraints and allowing all possible graphs. It's indeed increasing the possible graphs that we can consider but it not covering some recent wildly accepted structure units like bottleneck convolution, which limit our G-VAE to produce a more effective network. In future work, we should design the design space more carefully which allows our G-VAE to explore more possible architectures which cover more existing search space.

7 Broader Impact Statement

After careful reflection, the authors have determined that this work presents no notable negative impacts on society or the environment.

8 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] [Sec.4.3],

   (b) Did you describe the limitations of your work? [Yes] [Sec.6]

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] [Sec.7]

   (d) Have you read the ethics author's and review guidelines and ensured that your paper conforms to them? `https://automl.cc/ethics-accessibility/` [Yes]

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...

   (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 version), an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes]

   (b) Did you include the raw results of running the given instructions on the given code and data? [Yes]

   (c) Did you include scripts and commands that can be used to generate the figures and tables in your paper based on the raw results of the code, data, and instructions given? [Yes]

   (d) Did you ensure sufficient code quality such that your code can be safely executed and the code is properly documented? [Yes]

   (e) Did you specify all the training details (e.g., data splits, pre-processing, search spaces, fixed hyperparameter settings, and how they were chosen)? [Yes]

   (f) Did you ensure that you compared different methods (including your own) exactly on the same benchmarks, including the same datasets, search space, code for training and hyperparameters for that code? [Yes]

   (g) Did you run ablation studies to assess the impact of different components of your approach? [Yes]

   (h) Did you use the same evaluation protocol for the methods being compared? [N/A]

   (i) Did you compare performance over time? [N/A]

   (j) Did you perform multiple runs of your experiments and report random seeds? [N/A]

   (k) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

   (l) Did you use tabular or surrogate benchmarks for in-depth evaluations? [Yes]

   (m) 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]

   (n) Did you report how you tuned hyperparameters, and what time and resources this required (if they were not automatically tuned by your AutoML method, e.g. in a NAS approach; and also hyperparameters of your own method)? [N/A]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets…

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [N/A]

   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. 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? [Yes]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

References

Abdal, R., Zhu, P., Mitra, N. J., and Wonka, P. (2021). Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *ACM Transactions on Graphics (ToG)*, 40(3):1–21.

Abdelfattah, M. S., Mehrotra, A., Dudziak, Ł., and Lane, N. D. (2021). Zero-cost proxies for lightweight NAS. In *International Conference on Learning Representations (ICLR)*.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Ci, Y., Lin, C., Sun, M., Chen, B., Zhang, H., and Ouyang, W. (2021). Evolving search space for neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6659–6669.

Dong, X. and Yang, Y. (2020). NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *International Conference on Learning Representations (ICLR)*.

Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2018). Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*.

Hu, Y., Liang, Y., Guo, Z., Wan, R., Zhang, X., Wei, Y., Gu, Q., and Sun, J. (2020). Angle-Based search space shrinking for neural architecture search. In *Computer Vision – ECCV 2020*, pages 119–134. Springer International Publishing.

Hu, Y., Wang, X., Li, L., and Gu, Q. (2021). Improving One-Shot NAS with Shrinking-and-Expanding supernet. *Pattern Recognit.*, 118:108025.

Lugmayr, A., Danelljan, M., Van Gool, L., and Timofte, R. (2020). Srflow: Learning the super-resolution space with normalizing flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 715–732. Springer.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. (2020). Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10428–10436.

Tu, R., Roberts, N., Khodak, M., Shen, J., Sala, F., and Talwalkar, A. (2022). NAS-bench-360: Benchmarking neural architecture search on diverse tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S., and Hariharan, B. (2019). Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4541–4550.

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., and Hutter, F. (2019). NAS-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning (ICML)*.

Zhou, D., Jin, X., Lian, X., Yang, L., Xue, Y., Hou, Q., and Feng, J. (2021). Autospace: Neural architecture search with less human interference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 337–346.