Training-free Design of Deep Networks as Ensembles of Clinical Experts

Tinghui Wu[1], Wuyang Chen[2*], Zijun Zhang[1,3*]

[1]Division of AI in medicine, Cedars-Sinai Medical Center, Los Angeles, 90048, California, USA.
[2]School of Computing Science, Simon Fraser University, Burnaby, 10700, British Columbia, Canada.
[3]Department of Computational Biomedicine, Cedars-Sinai Medical Center, Los Angeles, 90048, California, USA.

*Corresponding author(s). E-mail(s): wuyang@sfu.ca ; zijun.zhang@cshs.org;
Contributing authors: tinghui.wu@cshs.org;

Abstract

Artificial intelligence (AI) techniques such as deep learning hold tremendous potential for improving clinical practice. However, clinical data complexity and the need for extensive specialized knowledge represent major challenges in the current, human-driven model design. Moreover, as human interpretation of a clinical problem is inherently encoded in the model, the conventional single model paradigm is subjective and cannot fully capture the prediction uncertainty. Here, we present a fast and accurate framework for automated clinical deep learning, TEACUP (training-free assembly as clinical uncertainty predictor). The core of TEACUP is a newly developed metric that faithfully characterizes the quality of deep networks without incurring any cost for training of these networks. When compared to conventional, training-based approaches, TEACUP reduces computation costs by more than 90% while achieving improved performance across distinct clinical tasks. This efficiency allows TEACUP to create ensembles of expert AI models, mimicking the recommended clinical practice of using multiple human experts when interpreting medical data. By combining multiple perspectives, TEACUP provides more robust predictions and uncertainty quantification, paving the way for more reliable clinical AI.

Keywords: clinical artificial intelligence, deep learning, neural network architecture, automated machine learning, uncertainty quantification

1 Main

Deep neural networks (DNNs) have provided state-of-the-art tools for many machine learning tasks in clinical and biomedical applications, including disease classification [1, 2], medical imaging segmentation [3], medical device signal processing [4, 5], and genetic variant pathogenicity prediction [6, 7]. These clinical data modalities vastly differ from each other, making the construction of DNN models handling specific clinical datasets expertise-dependent. In routine clinical care, computed tomography (CT) scans, capturing detailed internal body structures in 3D, are typically modeled using 3D-convolutions, whereas electrocardiogram (ECG) recordings, which represent temporal dynamics of heart electrical activity in 1D, necessitate 1D-convolution models. Furthermore, the design of high-quality DNN models, even for the same clinical data modality, can vary significantly. For instance, different DNNs were designed to analyze and interpret ECG data to aid in the diagnosis of abnormal heart rhythms [4] versus asymptomatic left ventricular dysfunction[8]. Overall, designing clinical DNN models currently requires a deep knowledge of medicine, an expertise in deep learning techniques, and time to manually perform many trial and error analyses. Therefore, developing DNN design strategies that don't depend on empirics-driven model design would eliminate these limitations and offer more scalable and data-driven solutions for clinical artificial intelligence (AI).

Automated machine learning (AutoML) algorithms, unlike conventional machine learning [9–11], can overcome the dependence on empirics-driven expertise by automating the design of high-quality models. Conventionally, AutoML methods aim to provide a one-size-fits-all solution that can be applied to various datasets and prediction tasks, and are designed to work with both conventional machine learning and deep learning models. TPOT, a popular tool in this category, is a tree-based pipeline optimization tool that leverages feature preprocessors and model architectures to achieve optimal performance [12, 13]. AutoML methods tailored to DNNs have recently been used to design DNNs in specific domains, tasks, or types of data. For example, the Automated Modelling for Biological Evidence-based Research (AMBER) framework uses AutoML algorithms and Neural Architecture Search (NAS) to design DNN models that classify genomic sequences and identify disease-relevant genetic variants [14, 15].

However, all these AutoML strategies require model training to estimate the quality of the designed networks [16], which can compromise the efficiency and accuracy of the model search [17, 18]. DNN training is too time-consuming and computationally costly, routinely requiring hundreds [19] to dozens of thousands [20] of GPU hours; thus, the use of AutoML methods for clinical DNNs in the broader clinical and biomedical community has been prohibitive.

To overcome these obstacles, we present TEACUP (Training-free Assembly as Clinical Uncertainty Predictors), an automated deep learning framework for the rapid and accurate design of DNNs that tackle clinical tasks. TEACUP is modular and effectively separates the automated DNN design into evaluation and search phases (Fig.1A). During the evaluation phase, TEACUP leverages a novel composite training-free metric that characterizes a model's performance without any model training. TEACUP then optimizes the composite training-free metric to search for high-quality, task-specific DNN architectures, compatible with a wide range of searching algorithms.

Importantly, the TEACUP architecture ensemble quantifies prediction uncertainty (Fig. 1B), which is not feasible in the human-expert design or training-based AutoML design of neural networks, due to prohibitive computation costs. In clinical applications, uncertainty quantification can help clinicians make more informed and accurate decisions. For example, uncertain AI predictions can serve as a trigger for additional clinical investigations and more comprehensive, orthogonal clinical tests. On the contrary, highly confident AI predictions may reduce unnecessary testing or interventions. Indeed, multiple computational methods must agree on the same prediction before they can be considered clinically [21]. Thus, TEACUP uniquely allows clinicians to prioritize high-confidence AI predictions and to flag uncertain predictions for further clinical investigation. This key distinction leads to more accurate diagnoses, fewer unnecessary interventions, and improved patient outcomes [22].

TEACUP characterizes model performance across clinical tasks without training.

The core innovation of TEACUP is to develop and apply training-free methods that can faithfully evaluate a DNN's quality, without actually training the model. This is facilitated by previous extensive theoretical and empirical studies on natural images (e.g. CIFAR-100). However, clinical tasks harbor distinct challenges compared to natural images. We systematically introduced four training-free metrics: (1) Length distortion (Length), which measures the extent to which the network distorts an input curve [23–25]; (2) condition number of Neural Tangent Kernel (NTK), which indicates the trainability of a neural network [26–28]; (3) Synaptic flow (SynFlow), which is based on DNN parameter pruning and measures the sensitivity to changes in a specific parameter [29, 30]; and (4) Zero-shot Inverse Coefficient (ZiCo), which jointly considers the mean and standard deviation of a DNN's sample-wise gradients for high training convergence speed and generalization capacity [31] (Methods 5.2). Therefore, these four training-free metrics characterize the properties of a DNN in a complementary fashion and, taken together, provide a promising way of evaluating a DNN.

To rigorously evaluate the quality of these training-free metrics, we designed a comprehensive benchmark from clinical task and model architecture perspectives. First, from clinical task perspective, we performed two distinct clinical tasks with disparate application and data dimensions. Specifically, we used TEACUP to generate models from 1D ECG recordings for multi-classification of atrial fibrillation [15] and from 3D CT scan images to classify cancerous lung nodules [32] (see Methods 5.1 for details). Second, from model architecture perspective, we diversely sampled $n = 150$ models from the NAS-Bench-201 model space, one of the most popular and powerful deep residual convolutional model spaces widely adopted in deep learning [33] (see Supplementary Figure 1 for an illustration). NAS-Bench-201 model space provides a predefined set of neural network architectures, and we evaluated the corresponding performance metrics on each specific benchmark dataset. The sampled models correspond to approximately 1% of all possible models (Methods 5.3). We then compared the models ranked by training-free metrics to the performances of models that were fully trained to convergence. Specifically, we measured the Spearman correlation coefficient (ρ) of

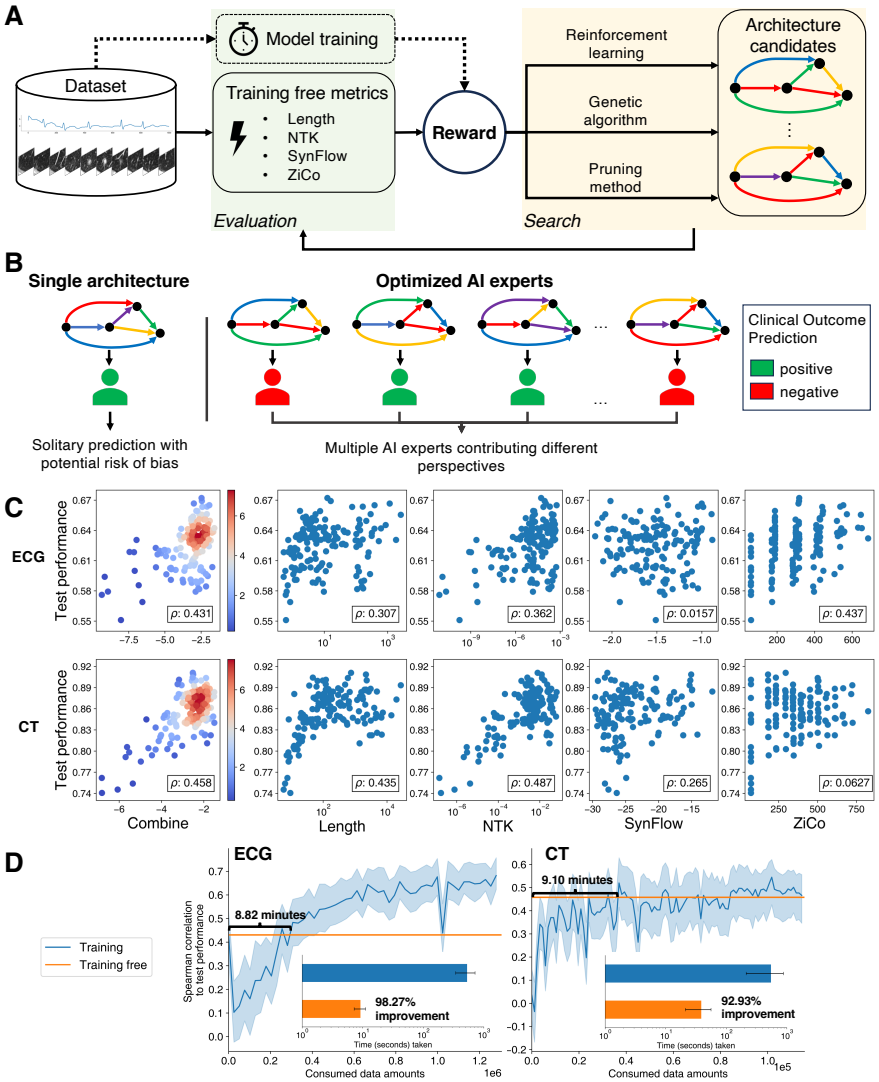


Fig. 1: TEACUP enables fast model evaluation on diverse clinical tasks with training-free metrics. **(A)** The workflow of the TEACUP framework. Instead of relying on traditional model training for DNN design, the reward of each sampled architecture is generated from the TEACUP metric, a weighted combination of calculated training-free metrics. Then, search algorithms optimize architectures based on rewards. **(B)** Illustration of the conventional single model architecture prediction and an ensemble of distinct models designed by our TEACUP. Compared to deep ensemble of AI experts, the single model often cannot fully capture the prediction uncertainty, posing potential risk of bias. **(C)** Spearman correlation between the test reward and training-free metrics. A composite TEACUP metric (left) is derived by combining all four metrics: Length, NTK, SynFlow and ZiCo. **(D)** Comparison of computation time costs between partial training and the TEACUP metric when they achieve equal Spearman correlations with the final trained DNN performance. For "Training", the Spearman correlation across the 150 benchmark architectures (y-axis) is calculated between the testing and the validation accuracy after trained for a certain amount of data (x-axis).

training-free metrics with the testing performance (F1 for ECG, AUC for CT) on ECG and CT datasets, respectively (Fig. 1C; Supplementary Table 1).

Importantly, no single existing training-free metric can consistently outperform other metrics across both ECG and CT datasets. Each of the four training-free metrics provided different levels of information when compared to their fully-trained model. While ZiCo achieved superior performance in ECG ($\rho = 0.437$), it failed to capture the trained performance in CT ($\rho = 0.063$). By contrast, NTK performed substantially better in CT ($\rho = 0.487$) than in ECG ($\rho = 0.362$).

Motivated by the different behaviors and complementary predictions of these four training-free metrics, we sought to develop a new composite training-free metric that can consistently characterize a DNN's quality across different clinical tasks. We refer to this new composite metric as the TEACUP metric that combines all four existing training-free metrics. We performed a multivariate linear regression analysis to find the most predictive linear combination of training-free metrics (Methods 5.4). Even though test rewards in ECG and CT datasets differ by a scale factor, our bootstrapped regression analysis revealed a surprisingly simple combination: TEACUP metric$=-0.5 \times \log(\text{Length}) + \log(\text{NTK}) + \log(\text{SynFlow}) + \log(\text{ZiCo})$(Supplementary Figure 2). As shown in Fig. 1C, the composite TEACUP metric that combines all training-free metrics, Length, NTK, SynFlow, and ZiCo, consistently achieved an overall high Spearman correlation with the performance of trained models ($\rho = 0.471$ for ECG, $\rho = 0.442$ for CT). We further demonstrated that all four training-free metrics, despite their various level of correlations to model performance in clinical tasks, were all informative to the final performance of the TEACUP metric (Supplementary Table 2).

TEACUP dramatically reduces computing time and costs.

We quantified the computing costs saved by employing the TEACUP metric versus training DNN models. To set up the comparison, we tested a commonly-used, naive baseline approach that assumes partially-trained models (i.e., only trained for the first few epochs) can predict DNN architecture generalization, therefore providing a fair comparison to our TEACUP metric performance. To determine how many training samples were consumed to achieve the same correlation as the TEACUP metric in each clinical dataset, we measured the correlation between the validation performance (F1 for ECG, AUC for CT) at each trained epoch and the test performance across $n = 150$ models. Notably, the correlation coefficient, when computed against the validation performance from each trained epoch across all $n = 150$ architectures, will be close to, but not reach 1. This is expected because even though there is a strong relationship between the validation and test rewards, they are not perfectly aligned.

As shown in Fig. 1D, on a single A100 GPU each model will take almost 10 minutes to train on ECG and CT scans to reach the same correlation to the model's test performance as the TEACUP metric, corresponding to training on roughly 300,000 samples in ECG and 35,000 samples in CT. In contrast, computing the training-free TEACUP metric takes less than 10 and 40 seconds to achieve a high correlation over 0.4 on ECG and CT, respectively. This represents a substantial reduction of over 90% in computing cost every time a DNN model is evaluated. Equivalently, using TEACUP framework on the scale of a previous milestone study that evaluated 12,800

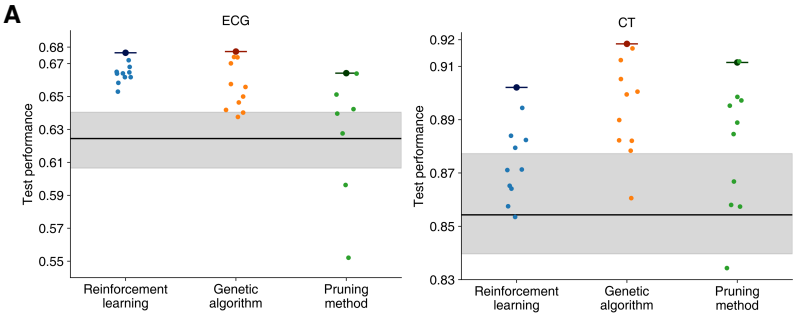
architectures [20], would be predicted to save over $7,300 in cloud computing costs and 1,792 hours per AutoML experiment [1] .

TEACUP improves the accuracy of automated model design.

We systematically evaluated three distinct searching algorithms to optimize neural architectures using the TEACUP metric as optimization objectives. In particular, we implemented a reinforcement learning-based controller network [14, 20] and a probabilistic model building genetic algorithm [34] to maximize the TEACUP metric as a reward, and a pruning-based algorithm that iteratively removes the edge with the minimum contribution to the TEACUP metric from a super-net [28]. Given that these three algorithms have distinct underlying assumptions for DNN architectures and cover a wide range of neural architecture search methods, their performance would be indicative of the general utility of the TEACUP metric in automated DNN design. TEACUP removed heavy computation costs in conventional AutoML methods and allowed us to efficiently perform 10 independent runs for each search algorithm on each clinical task, which enabled us to account for the stochasticity of the search algorithms. This would not have been feasible using conventional AutoML methods for DNN, given the cost and time required to finish these tasks. In contrast, all three search algorithms were efficient in optimizing the TEACUP metric, and converged within 6 GPU hours across all runs with TEACUP (Methods 5.5; Supplementary Figure 3).

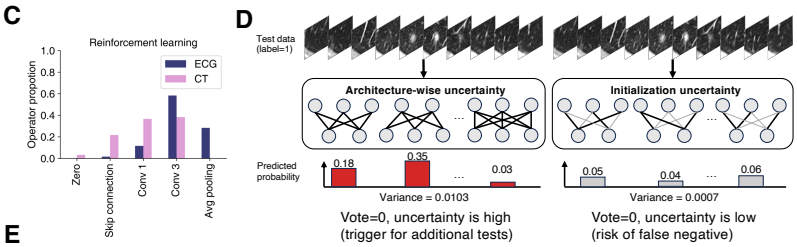
The improved TEACUP metrics successfully translated to high-performance DNN models. To examine the quality of DNN models built using TEACUP, we first compared TEACUP-optimized models to randomly sampled models from the NAS-Bench-201 model space, then expanded the comparison to state-of-the-art AutoML methods. TEACUP-optimized models exhibited significantly higher testing performance in both ECG and CT datasets, with the sole exception of pruning-based search method in ECG which failed to improve performance (Fig. 2A). Nonetheless, in CT dataset, pruning-based algorithm searched for models (test AUC=0.88±0.02) significantly better than the randomly sampled benchmark (test AUC=0.85±0.03; Fig.2B). This demonstrates that different search algorithms have varying performances across clinical datasets. Indeed, we found that reinforcement learning-based controller network achieved superior searched model performance in the ECG dataset, while genetic algorithm worked the best in the CT dataset. Furthermore, these TEACUP-optimized models are significantly more accurate than previously published human-designed baselines (Fig. 2B). In ECG [15], TEACUP models achieved an average testing F1 of 0.66±0.01, significantly higher than XGBoost (0.44±0.02) [35] and Wide ResNet (0.57±0.01) [36]. Similarly in CT [32], TEACUP models performed better (testing AUC of 0.89±0.01) than ResNet-18 (0.88±0.02) and a substantially more parameterized ResNet-50 (0.87±0.02) [37]. Even compared with training-based AutoML methods [14, 38–41], TEACUP remained highly competitive and matched the performance among state-of-the-art AutoML methods. Additionally, we ensembled the TEACUP models into the TEACUP ensemble (Fig. 2A), which achieved the highest test performance among existing AutoML methods (test F1=0.68 on ECG, test AUC=0.92 on CT).

[1] We use one cloud A100 on AWS (https://aws.amazon.com/ec2/instance-types/p4/) as an example: $4.1 per hour × 0.14 hours × 12800 ≈ $7347.



B

Dataset	ECG		CT	
	Method	Result (F1)	Method	Result (AUC)
Training-free searching method	Benchmark	0.62±0.02	Benchmark	0.85±0.03
	Reinforcement learning	0.66±0.01	Reinforcement learning	0.87±0.01
	Genetic algorithm	0.65±0.01	Genetic algorithm	0.89±0.02
	Pruning method	0.62±0.04	Pruning method	0.88±0.02
Fixed architecture training	Wide ResNet (WRN)	0.57±0.01	ResNet-18	0.88±0.02
	XGBoost	0.44±0.02	ResNet-50	0.87±0.02
Training-based searching method	DenseNAS random	0.58±0.01	Auto-sklearn	0.91±0.01
	DARTS GAEA	0.66±0.01	AutoKeras	0.87±0.03
	AMBER	0.67±0.02		



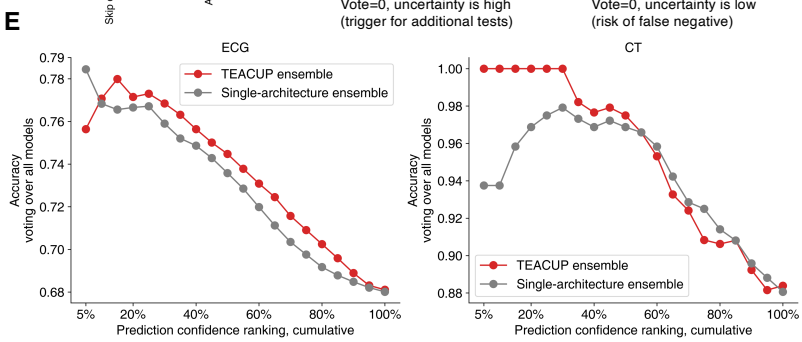


Fig. 2: TEACUP generates high-performance deep ensembles for clinical uncertainty quantification. **(A)** Performance of ensemble prediction (larger dot with a dash) and optimized architectures (dot) compared with benchmark results (gray area). The horizontal line is the median, and the gray area is the upper to lower quartile region, for the performance of 150 benchmark architectures, respectively. **(B)** Performance comparison between our approach and previous works. As ECG[15] and CT[32] were benchmarked by different methods previously, we sought to keep consistent with previous works in terms of evaluation metrics and methods used. **(C)** Distributions of optimized architectures by reinforcement-learning search algorithm on two datasets. **(D)** Comparison of a false negative test case over two uncertainty quantification methods: different architectures vs. single architecture trained from different weight initialization. Despite both methods yielding incorrect predictions, the larger architecture-wise uncertainty suggests the need for additional tests. **(E)** Across testing samples of different confidence, the ensemble of TEACUP-optimized models outperforms the ensemble of a single architecture trained from random initialization.

The automated TEACUP-optimized DNNs not only achieved strong accuracy, but also discovered task-specific patterns of DNN structures that are aligned with characteristics of different tasks and clinical data modalities (Fig. 2C; Supplementary Figure 4). In the case of ECG, more edges are allocated with pooling layers and convolutions with the larger kernel size. In contrast, skip connections and the small convolution kernel are preferred in CT. These two different groups of architectures are highly aligned with the characteristics of the two clinical datasets. In ECG, the long sequence of the heartbeat signal (1000 in length) requires a larger receptive field, which can be achieved by pooling and large kernels. However, 3D images of CT scans have a much smaller spatial resolution ($28 \times 28 \times 28$), and using small convolutions will achieve sufficient receptive fields.

TEACUP enables robust ensemble learning and uncertainty quantification.

The multiple accurate, distinct architectures efficiently optimized by TEACUP present an unprecedented opportunity for uncertainty quantification, which is impossible to achieve using conventional single-model paradigms given the prohibitive DNN training cost. In the context of DNNs, different types of neural networks can make incorrect clinical predictions for the same patient persistant to an architecture, even when starting from different weight initializations, suggesting that each neural network is sensitive to different aspects of the data [42].

We hypothesized that DNNs of different architectures optimized by TEACUP capture different perspectives of clinical data patterns and thus would allow us to prioritize high-confidence, high-accuracy predictions over uncertain, less-accurate predictions. To rank predictions based on their confidence, we computed their variance across different TEACUP-optimized models as a measure of uncertainty quantification (Method 5.7). Intuitively, neural network models with distinct architectures (i.e. different "experts") that make the same prediction from clinical samples are more confident and thus likely to be accurate. For comparison, we followed the same uncertainty computation across different random initializations of a single model architecture, which already goes beyond the conventional single model regime by considering multiple random weight initialization. In the test CT example illustrated in Fig. 2D, even though the predictions from both schemes are incorrect (i.e. false negative), the architecture-wise uncertainty calculated by the TEACUP ensemble is high, flagging this example for additional tests; whereas the initialization uncertainty calculated by the single-architecture ensemble is low, indicating over-confidence in its incorrect prediction.

We further chose the architecture that occurred most frequently across all TEACUP search runs as the single model architecture baseline for ECG and CT, respectively (Method 5.7). This represents a strong and fair baseline, because the DNN architecture that is consistently searched as optimal by all three search algorithms is more likely to fit the corresponding clinical data. Consequently, this allowed us to interrogate the influence of assembling distinct architectures versus only weight initialization variability in an optimal but fixed architecture.

Deep ensembles of TEACUP-optimized distinct architectures showed superior performance in prioritizing high-confidence clinical predictions, even when compared

against the strong baseline of the most consistently-searched single architecture ensemble. Although both methods were highly competitive in ECG and CT when evaluating all test samples (rightmost dot, 100% test data; Fig. 2E), the TEACUP ensemble achieved a superior accuracy compared to the single-architecture ensemble when we move to the most high-confidence predictions, (leftmost dots, Fig. 2E). For example, the TEACUP ensemble achieved perfect accuracy (100%) in 55% of top-ranked confident predictions in CT; whereas the single-model ensemble achieved an accuracy of only 94%, highlighting a 6% accuracy gap. Thus, the TEACUP ensemble more effectively ranks and prioritizes high-confidence, high-accuracy predictions.

In summary, TEACUP is a fast, accurate, and universal automated deep learning framework for clinical tasks. By using training-free metrics, we can estimate a neural network's performance without expending large computation resources on model training, reducing computation cost by more than 90% per model evaluation. The composite metric we developed is compatible with a wide range of search algorithms. We demonstrated improved training-free metrics using three search algorithms. TEACUP generated multiple, optimal models with distinct architectures that enabled architecture-wise ensemble learning, which in turn allowed us to quantify DNN uncertainty for clinical data for the first time. As such, the neural uncertainty enabled by our TEACUP framework presents a major paradigm shift to current deep learning-based clinical prediction methods, and we expect that it will be of significant clinical utility.

2 Data Availability

The benchmarking datasets are publicly available. ECG is precompiled in https://nb360.ml.cmu.edu/. CT scan is part of the MedMNIST dataset in https://medmnist.com/.

3 Code Availability

Our code and analysis can be found in GitHub: https://github.com/zhanglab-aim/TEACUP.

4 Acknowledgements

We acknowledge all members of the Zhang lab for helpful discussions. This work is supported by a Winnick family foundation award, a PCF challenge award, and a Cedars-Sinai institutional start-up to ZZ.

References

[1] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., et al.: Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv preprint arXiv:1711.05225 (2017)

[2] Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. nature **542**(7639), 115–118 (2017)

[3] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, pp. 234–241 (2015). Springer

[4] Hannun, A.Y., Rajpurkar, P., Haghpanahi, M., Tison, G.H., Bourn, C., Turakhia, M.P., Ng, A.Y.: Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. Nature medicine **25**(1), 65–69 (2019)

[5] Acharya, U.R., Oh, S.L., Hagiwara, Y., Tan, J.H., Adeli, H.: Deep convolutional neural network for the automated detection and diagnosis of seizure using eeg signals. Computers in biology and medicine **100**, 270–278 (2018)

[6] Quang, D., Chen, Y., Xie, X.: Dann: a deep learning approach for annotating the pathogenicity of genetic variants. Bioinformatics **31**(5), 761–763 (2014)

[7] Zhou, J., Troyanskaya, O.G.: Predicting effects of noncoding variants with deep learning–based sequence model. Nature methods **12**(10), 931–934 (2015)

[8] Attia, Z.I., Kapa, S., Lopez-Jimenez, F., McKie, P.M., Ladewig, D.J., Satam, G., Pellikka, P.A., Enriquez-Sarano, M., Noseworthy, P.A., Munger, T.M., *et al.*: Screening for cardiac contractile dysfunction using an artificial intelligence–enabled electrocardiogram. Nature medicine **25**(1), 70–74 (2019)

[9] Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. The Journal of Machine Learning Research **20**(1), 1997–2017 (2019)

[10] Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., Wang, X.: A comprehensive survey of neural architecture search: Challenges and solutions. ACM Computing Surveys (CSUR) **54**(4), 1–34 (2021)

[11] He, X., Zhao, K., Chu, X.: Automl: A survey of the state-of-the-art. Knowledge-Based Systems **212**, 106622 (2021)

[12] Olson, R.S., Moore, J.H.: Tpot: A tree-based pipeline optimization tool for automating machine learning. In: Workshop on Automatic Machine Learning, pp. 66–74 (2016). PMLR

[13] Romano, J.D., Le, T.T., Fu, W., Moore, J.H.: Tpot-nn: augmenting tree-based automated machine learning with neural network estimators. Genetic Programming and Evolvable Machines **22**, 207–227 (2021)

[14] Zhang, Z., Park, C.Y., Theesfeld, C.L., Troyanskaya, O.G.: An automated framework for efficiently designing deep convolutional neural networks in genomics. Nature Machine Intelligence **3**(5), 392–400 (2021)

[15] Tu, R., Roberts, N., Khodak, M., Shen, J., Sala, F., Talwalkar, A.: Nas-bench-360: Benchmarking neural architecture search on diverse tasks. Advances in Neural Information Processing Systems **35**, 12380–12394 (2022)

[16] Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameters sharing. In: International Conference on Machine Learning, pp. 4095–4104 (2018). PMLR

[17] Yu, K., Ranftl, R., Salzmann, M.: An analysis of super-net heuristics in weight-sharing nas. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(11), 8110–8124 (2021)

[18] Wang, J., Bai, H., Wu, J., Shi, X., Huang, J., King, I., Lyu, M., Cheng, J.: Revisiting parameter sharing for automatic neural channel number search. Advances in Neural Information Processing Systems **33**, 5991–6002 (2020)

[19] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2820–2828 (2019)

[20] Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)

[21] Richards, S., Aziz, N., Bale, S., Bick, D., Das, S., Gastier-Foster, J., Grody, W.W., Hegde, M., Lyon, E., Spector, E., *et al.*: Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the american college of medical genetics and genomics and the association for molecular pathology. Genetics in medicine **17**(5), 405–423 (2015)

[22] Kompa, B., Snoek, J., Beam, A.L.: Second opinion needed: communicating uncertainty in medical machine learning. NPJ Digital Medicine **4**(1), 4 (2021)

[23] Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., Ganguli, S.: Exponential expressivity in deep neural networks through transient chaos. In: Advances in Neural Information Processing Systems, pp. 3360–3368 (2016)

[24] Hanin, B., Jeong, R., Rolnick, D.: Deep relu networks preserve expected length. arXiv preprint arXiv:2102.10492 (2021)

[25] Chen, W., Huang, W., Du, X., Song, X., Wang, Z., Zhou, D.: Auto-scaling vision transformers without training. arXiv preprint arXiv:2202.11921 (2022)

[26] Jacot, A., Gabriel, F., Hongler, C.: Neural tangent kernel: Convergence and generalization in neural networks. In: Advances in Neural Information Processing Systems, pp. 8571–8580 (2018)

[27] Xiao, L., Pennington, J., Schoenholz, S.S.: Disentangling trainability and generalization in deep learning. arXiv preprint arXiv:1912.13053 (2019)

[28] Chen, W., Gong, X., Wang, Z.: Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In: International Conference on Learning Representations (2021)

[29] Tanaka, H., Kunin, D., Yamins, D.L., Ganguli, S.: Pruning neural networks without any data by iteratively conserving synaptic flow. Advances in neural information processing systems **33**, 6377–6389 (2020)

[30] Abdelfattah, M., Mehrotra, A., Dudziak, L., Lane, D.N.: Zero-cost proxies for lightweight nas. In: International Conference on Learning Representations (2021)

[31] Li, G., Yang, Y., Bhardwaj, K., Marculescu, R.: Zico: Zero-shot nas via inverse coefficient of variation on gradients. arXiv preprint arXiv:2301.11300 (2023)

[32] Yang, J., Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., Pfister, H., Ni, B.: MedMNIST v2 - A large-scale lightweight benchmark for 2D and 3D biomedical image classification. Scientific Data **10**(1), 41 (2023) https://doi.org/10.1038/s41597-022-01721-8 . Number: 1 Publisher: Nature Publishing Group. Accessed 2023-05-31

[33] Dong, X., Yang, Y.: Nas-bench-201: Extending the scope of reproducible neural architecture search. arXiv preprint arXiv:2001.00326 (2020)

[34] Zhang, Z., Lamson, A.R., Shelley, M., Troyanskaya, O.: Interpretable neural architecture search and transfer learning for understanding crispr–cas9 off-target enzymatic reactions. Nature Computational Science **3**(12), 1056–1066 (2023)

[35] Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)

[36] Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)

[37] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[38] Fang, J., Sun, Y., Zhang, Q., Li, Y., Liu, W., Wang, X.: Densely connected search space for more flexible neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10628–10637 (2020)

[39] Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)

[40] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. Advances in neural information processing systems **28** (2015)

[41] Jin, H., Song, Q., Hu, X.: Auto-keras: An efficient neural architecture search system. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1946–1956 (2019)

[42] Pedersen, C., Tesileanu, T., Wu, T., Golkar, S., Cranmer, M., Zhang, Z., Ho, S.: Reusability report: Prostate cancer stratification with diverse biologically-informed neural architectures. arXiv preprint arXiv:2309.16645 (2023)

[43] Clifford, G.D., Liu, C., Moody, B., Lehman, L.-w.H., Silva, I., Li, Q., Johnson, A.E., Mark, R.G.: AF Classification from a Short Single Lead ECG Recording: the PhysioNet/Computing in Cardiology Challenge 2017. Computing in cardiology **44**, 10–224892017065469 (2017). Accessed 2023-10-30

[44] The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): A Completed Reference Database of Lung Nodules on CT Scans. Medical Physics **38**(2), 915–931 (2011) https://doi.org/10.1118/1.3528204 . Accessed 2023-11-07

[45] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[46] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)

[47] Burkholz, R., Dubatovka, A.: Initialization of relus for dynamical isometry. In: Advances in Neural Information Processing Systems, pp. 2385–2395 (2019)

[48] Hayou, S., Doucet, A., Rousseau, J.: On the impact of the activation function on deep neural networks training. arXiv preprint arXiv:1902.06853 (2019)

[49] Shin, Y., Karniadakis, G.E.: Trainability of relu networks and data-dependent initialization. Journal of Machine Learning for Modeling and Computing **1**(1) (2020)

[50] Jacot, A., Gabriel, F., Hongler, C.: Neural tangent kernel: Convergence and generalization in neural networks. In: Advances in Neural Information Processing Systems 31, (2018)

[51] Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., Pennington, J.: Wide neural networks of any depth evolve as linear models under gradient descent. In: Advances in Neural Information Processing Systems, pp. 8572–8583 (2019)

[52] Chizat, L., Oyallon, E., Bach, F.: On lazy training in differentiable programming (2019)

[53] Lee, N., Ajanthan, T., Torr, P.H.: Snip: Single-shot network pruning based on connection sensitivity. arXiv preprint arXiv:1810.02340 (2018)

[54] He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)

[55] Wang, C., Zhang, G., Grosse, R.: Picking winning tickets before training by preserving gradient flow. arXiv preprint arXiv:2002.07376 (2020)

5 Methods

5.1 Data

To demonstrate the performance of our method, we study two benchmarks for machine learning tasks routinely used in clinical settings, covering 1D and 3D medical data:

1. ECG (Electrocardiogram): The dataset [15] is originally collected for the 2017 PhysioNet Challenge [43] for heartbeat categorization, with an average 30-second ECG recording stored at 300 Hz. Processed by 1,000 ms sliding window and 500 ms stride, there are 261,740 training data, 33,281 validation data, and 33,494 testing data in total. All the data is labeled from one of the four classes: normal, disease (atrial fibrillation, AF), other, or noisy rhythm. The macro-averaged F1 score is used to evaluate training performance.
2. NoduleMNIST3D: This lung nodule dataset collects images from thoracic CT scans. The dataset is used in a benchmark paper [32]. Originally from [44], a total of 1,633 three-dimensional 28×28×28 images with spacing 1mm×1mm×1mm are cropped from the center of spatially normalized thoracic CT scans. The data is turned into a binary classification task using malignancy levels 1 and 2 as negative, 4 and 5 as positive, ignoring malignancy level 3. With a splitting rate of 7:1:2, there are 1,158 training data, 165 validation data, and 310 testing data. The area under the curve (AUC) is used as an evaluation metric.

5.2 Training-free Metrics

The bottleneck of accelerating AutoML for medical problems is the costly and unstable DNN training, for each sampled DNN during the architecture search. To avoid such overhead, we aim to replace the DNN training with indicators or metrics that exhibit strong correlations with DNNs' performance on healthcare benchmarks while not requiring training. Inspired by the recent development of deep learning theory [24, 27]

and AutoML methods [20, 39], we introduce four training-free metrics. These four metrics are complementary to each other in characterizing different aspects of a DNN's properties:

1. Condition number of Neural Tangent Kernel (NTK): Training deep networks requires optimizing high-dimensional non-convex loss functions. In practice, gradient descent often finds the global or good local minimum. However, many expressible networks are not easily learnable. For example, a deep stack of convolutional layers (e.g. Vgg [45]) is much harder to train than networks with skip connections (ResNet [37], DenseNet [46], etc.), even the former could equip a larger number of parameters. The trainability of a neural network studies how effective it can be optimized by gradient descent [47–49]. To characterize the training dynamics of wide networks, Neural tangent kernel (NTK) is proposed [50–52]. NTK controls the training dynamics of linearized DNNs and can be treated as the measurement of "sample-wise similarity" by the inner product of network's gradients across different input data. NTK is defined as:
$$\hat{\Theta}(\boldsymbol{x}, \boldsymbol{x}') = J(\boldsymbol{x})J(\boldsymbol{x}')^T, \tag{1}$$
where $J(\boldsymbol{x})$ is the Jacobian evaluated at input samples $\boldsymbol{x}$. Inspired by [27, 28], we measure the trainability of networks by studying the spectrum and conditioning of $\hat{\Theta}$, and use the empirical condition number of NTK to represent trainability:
$$\hat{\kappa} = \frac{\lambda_{\max}(\hat{\Theta}(\boldsymbol{x}, \boldsymbol{x}))}{\lambda_{\min}(\hat{\Theta}(\boldsymbol{x}, \boldsymbol{x}))}. \tag{2}$$
$\boldsymbol{x}$ are drawn from a training dataset. $\hat{\kappa}$ is calculated at the network's initialization. A small NTK condition number indicates a smooth loss landscape and favors faster DNN training convergence.
2. Length distortion (Length): Intuitively, a complex network can propagate a simple input into a complex manifold at its output layer, thus likely to possess a strong learning capacity. In our work, we also study the manifold complexity of mapping a simple circle input through the network. We define the network as $\mathcal{M}$, its input-output Jacobian $\boldsymbol{v}(x) = \partial_x \mathcal{M}(x)$ at an input x, and $\boldsymbol{a}(x) = \partial_x \boldsymbol{v}(x)$. Length distortion measures the norm of a DNN's input-output Jacobian. The Length Distortion in Euclidean space is defined as $\mathcal{L}^E = \frac{\text{length}(\mathcal{M}(x))}{\text{length}(x)}$. It measures when the network takes a unit-length input, what is the length of the output curve. Since the ground-truth function we want to estimate (using $\mathcal{M}$) is usually very complex, one may also expect that networks with better performance should also generate longer outputs. We will calculate expected complexities over a certain number of xs randomly sampled. A large length distortion indicates a high sensitivity of DNN's output to changes of its input, and thus potentially indicates a larger model capacity in learning complicated input-output mappings. The original NTK is negatively correlated with the model performance; for simplicity in this work, we further transform to the negative absolute value of NTK so they positively correlate with the model performance.
3. Synaptic flow (SynFlow): Originally, [53] proposed performing parameter pruning based on a saliency metric computed at initialization using a single minibatch of data. This saliency criteria approximates the change in loss when a specific

parameter is removed. Tanaka et al. [29] generalized these so-called synaptic saliency scores and proposed a modified version (SynFlow) which avoids layer collapse when performing parameter pruning. Synaptic flow measures the product of a DNN parameter and its gradient, indicating the sensitivity of the loss with respect to changes in a specific DNN parameter (i.e. larger SynFlow indicates more important parameters). Similar to NTK, we transform to the negative absolute value of SynFlow.

$$\text{SynFlow} : \mathcal{S}(\theta) = \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta \tag{3}$$

$\mathcal{L}$ is the loss function of a neural network with parameters $\boldsymbol{\theta}$, $\theta \in \boldsymbol{\theta}$. S is the per-parameter saliency. We extend these saliency metrics to score an entire neural network by summing over all $|\boldsymbol{\theta}|$ parameters in the model: $\mathcal{S}_n = \sum_i^{|\boldsymbol{\theta}|} \mathcal{S}(\theta_i)$.

4. Zero-shot Inverse Coefficient (ZiCo): In [31], Li et al. found that networks with high training convergence speed and generalization capacity should have high absolute mean values and low standard deviation values for the gradient with respect to the parameters across different training samples and batches. Therefore, ZiCo jointly considers both absolute mean and standard deviation values. ZiCo prefers high absolute mean values and low standard deviation values of a DNN's sample-wise gradients. It implicitly improves the convergence and generalization of DNNs.

$$ZiCo = \sum_{l=1}^{L} \log \left(\sum_{\theta \in \boldsymbol{\theta}_l} \frac{\mathbb{E}\left[|\nabla_\theta \mathcal{L}\left(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}\right)|\right]}{\sqrt{\text{Var}\left(|\nabla_\theta \mathcal{L}\left(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}\right)|\right)}} \right). \tag{4}$$

L stands for the depths of the network. $\boldsymbol{\theta}$ denotes the set of all initialized parameters of a given network; $\boldsymbol{\theta}_l$ denote the parameters of the l^{th} layer of the network, and θ represents each element in $\boldsymbol{\theta}_l$; $\boldsymbol{x}$ and $\boldsymbol{y}$ are input samples and corresponding labels from the training set. $\mathbb{E}$ and Var are taken over the batch of input samples.

Instead of calculating the loss or error of a neural network after training, we measure these training-free metrics at the network's initialization to characterize its performance. The core benefit of our training-free metrics is to achieve a strong correlation with the quality of designed networks with minimal computation costs. Note that for all four metrics, we use their expectations taken over batches of samples drawn from a training dataset $\boldsymbol{x} \sim D_{\text{train}}$ and random Kaiming normal initializations $\boldsymbol{\theta} \sim \mathcal{N}(0, \frac{2}{N_l})$ (N_l is the width at layer l) [54].

5.3 Model Space

We build our model space based on NAS-Bench-201 [33]. NAS-Bench-201 provides a standard cell-based search space (containing 15,625 architectures). The macro skeleton of each architecture candidate begins with a convolution layer as a stem, followed by three blocks of cells with two intermediate residual blocks connecting them. Then, global average pooling is applied to flatten the output into one dimension, followed by a fully connected layer for binary or multi-classification output, depending on the application. Each block contains 5 cells, with each cell having 4 nodes (feature maps) that form a directed acyclic graph, where each edge is associated with an operation selected from *none* (*zero*), *skip connection*, *conv*1×1, *conv*3×3, and *average pooling* 3×3, resulting

in a total of $5^6 = 15,625$ candidates. We adapt it to our problems by changing the convolution and pooling layer to 1D and 3D kernels. To obtain the benchmark results, we sample 150 architectures, which is approximately $150/15,625 \approx 1\%$.

5.4 Training-free Metrics Combination

To improve the robustness of training-free metrics across different clinical datasets, we seek to ensemble different combinations of the four training-free metrics. We employ a bootstrap method ($B = 2000$) over the 150 benchmark data points from the two distinct datasets. Within each bootstrap iteration, a $80:20$ train-test split is applied separately on both the ECG and CT datasets. Then, after merging training sets from both datasets, a multivariate linear ridge regression is conducted to compute the weighted aggregate of training-free metrics. The multivariate regression regresses the logarithm of all four training-free metrics to the standardized model test performance of both datasets.

To further explore the potential of using a reduced number of training-free metrics, we repeat the bootstrap procedure while retaining only three out of the four training-free metrics. Spearman correlation coefficients between the model performance and weighted aggregate of training-free metrics across ECG and CT test sets are utilized to evaluate performance consistency. Because the rank correlation remains unchanged for linear rescale of the combined training-free metric, for simplicity, we divide the weights derived from the multivariate linear ridge regression by the minimum absolute coefficient to achieve a more interpretable overall proportion.

5.5 Architecture Search Methods

The overall pipeline of our method works as a modular framework of evaluation phase and search phase. Given a large model space (which includes over 10K different DNN structures in our work), we equip popular neural architecture search algorithms with our training-free metrics, i.e., these search algorithms will use the TEACUP metric as a reward to evaluate architectures when they explore the model space. For each architecture explored by the search algorithm, we randomly initialized the network and calculated the four training-free metrics on the network's initialization with mini-batches of training data (batch size= 8 with 4 mini-batches), without any gradient descent steps. That means our search is still data-dependent and metrics are aware of different tasks. The final searched network architecture will be evaluated by training.

5.5.1 Reinforcement Learning (RL)

The policy agent maintains an internal state to represent the architecture search space, denoted as $\boldsymbol{\theta}^{\boldsymbol{A}}$. This internal state can be converted to a categorical distribution of the architectures ($\boldsymbol{\mathcal{A}}$) via softmax: $\boldsymbol{\mathcal{A}} = \sigma(\boldsymbol{\theta}^{\boldsymbol{A}})$.

Stopping Criterion: We stop the RL search when the average TEACUP metric stops increasing for 30 iterations (total iterations $T = 100$ in our work). We trained the RL agent with Adam optimizer and a learning rate as $\eta = 0.001$.

Architecture Sampling: In each iteration, the agent samples 10 architectures a_t from $\boldsymbol{\mathcal{A}}$.

Update: We update the RL agent via policy gradients.

$$\boldsymbol{\theta}_{t+1}^{\boldsymbol{A}} = \boldsymbol{\theta}_{t}^{\boldsymbol{A}} - \eta \cdot \nabla_{\boldsymbol{\theta}^{\boldsymbol{A}}} f(\boldsymbol{\theta}_{t}^{\boldsymbol{A}}) \quad t = 1, \cdots, T \tag{5}$$

$$f(\boldsymbol{\theta}_{t}^{\boldsymbol{A}}) = -\log(\sigma(\boldsymbol{\theta}^{\boldsymbol{A}})) \cdot (r_t - b_t) \tag{6}$$

$$b_t = \gamma b_{t-1} + (1-\gamma) r_t \quad (b_0 = 0, \gamma = 0.9) \tag{7}$$

r stands for reward, the TEACUP metric combined in Sec. 5.4, and b for an exponential moving average of reward for variance reduction.

Architecture Deriving: To derive the final searched network, the agent chooses the architecture that has the highest probability, i.e., $a^* = \text{argmax}_a \sigma(\boldsymbol{\theta}^{\boldsymbol{A}})(a)$.

5.5.2 Genetic Algorithm

We adopt a Bayesian probabilistic model building genetic algorithm as another search method. The genetic algorithm is first initialized with a population of 50 architectures by random sampling.

Stopping Criterion: We run the algorithm for a maximum of 100 iterations, and early-stop the searching when the average TEACUP metric stops increasing for 30 iterations.

Architecture Sampling: We model the distribution of the architectures $\boldsymbol{\mathcal{A}}$ as multinomial distributions

$$\boldsymbol{\mathcal{A}} \sim \text{Multinomial}(n = 1, p = \boldsymbol{\theta}), s.t. \sum_k \theta_k = 1 \tag{8}$$

That is, we draw $n = 1$ instance each time from $\{1, 2, .., k, .., K\}$ classes. The probability of each class θ_k follows a Dirichlet distribution as an uninformative prior:

$$\boldsymbol{\theta} \sim \text{Dir}(\mathbf{1}) \tag{9}$$

The posterior distribution of $\boldsymbol{\theta}$ will be updated using the survived architectures in each generation that further updates the sampled architectures.

Update: Following Zhang et al. [34], in each iteration, the population is updated by adding a set of new architectures (n=10 in our work) and popping out the same amount of oldest architectures (the one that stays in the population for the longest time). Our buffer keeps at most 5 sets of architectures, which is 50 architectures in total. At the end of each iteration, we calculate the mean reward r_{mean} in the buffer and only use the M architectures with a higher reward than r_{mean} for updating the probability $\boldsymbol{\theta}$ by generating the conjugate rule of Dirichlet-Multinomial:

$$p(\theta|\boldsymbol{\mathcal{A}}) = \text{Dir}(\{a > r_{mean}, \forall a \in \boldsymbol{\mathcal{A}}\}|\boldsymbol{\alpha}) \tag{10}$$

$$\alpha_k = \sum_{i=1}^{M} a_{i,k} + 1 \tag{11}$$

Architecture Deriving: To derive the final searched network, the best architecture from the population is selected, where the criterion is the same as we choose a_t (see above "Architecture Sampling").

5.5.3 Pruning

Inspired by recent works on pruning-from-scratch [53, 55], we also leverage a a pruning-by-importance model search mechanism [28] to quickly shrink the search possibilities and boost the search efficiency. Specifically, we start the search with a super-network $\mathcal{M}_0$ composed of all possible operators and edges, and iteratively prune operators with marginal importance (with respect to four training-free metrics). In the outer loop, for every round we prune one operator on each edge with the least contribution of four training-free metrics. In the inner loops, we measure and compare each operator's importance.

Stopping Criterion and Architecture Deriving: The outer-loop stops when the current supernet $\mathcal{M}_t$ is pruned to be a single-path network, i.e., there is only one candidate operator left on each edge.

Update: For the inner-loop, we measure the change of each four training-free metrics before and after pruning each individual operator, and assess its importance using the sum of ranks of the changes of four training-free metrics. We order all currently available operators in terms of their importance, and prune the lowest-importance operator on each edge.

For more details, please refer to the Algorithm 1 in [28].

5.6 Training Settings

We use a similar training setting as stated in the NAS-Bench-201[33] for both the benchmark 150 architectures and the evaluation of the optimized architecture generated from search methods. The SGD optimizer with a momentum of 0.9 is used for training on both datasets, while the weight decay of 10^{-7} and 0.005 is used for ECG and CT, respectively. A fixed cosine annealing learning scheduler initializes on different values: for CT, it ranges from 0.1 to 0 over 100 epochs, while for ECG, it ranges from 0.01 to 0. For the CT dataset, we train for 100 epochs without early stopping. For ECG, empirically we found that most models converge fast, therefore we train for 10 epochs and evaluate on the validation set every 100 step. We use the checkpoint that has the lowest validation loss as the final model.

5.7 Evaluation of Model Ensembles

We ensemble the predictions by voting and taking averages for ECG and CT, respectively, based on their reward metrics (F1 and AUC). For ECG, since it is a multi-class classification with an F1 score as the reward, we set the majority vote among 4 classes as the final prediction. For CT, since it is a binary classification with AUC as the reward, we take the average over all the model outputs.

To compare the disparities between ensemble over architecture-wise and initialization-wise predictions, we select the most common architecture from a pool of 30 optimized models obtained through all three searching algorithms. Subsequently,

we train the architecture for 10 times employing different random model initializations to form a robust control group. We refer to this baseline as "single-architecture ensemble", as opposed to ensembling over different architectures (referred to as "TEACUP ensemble").

Then, we analyze the prediction uncertainties for each test case by ranking them based on the variance among the predictions generated from different architectures or initializations. The test case exhibiting the lowest variance across the ten predictions indicates the least uncertainty, and hence the highest prediction confidence.

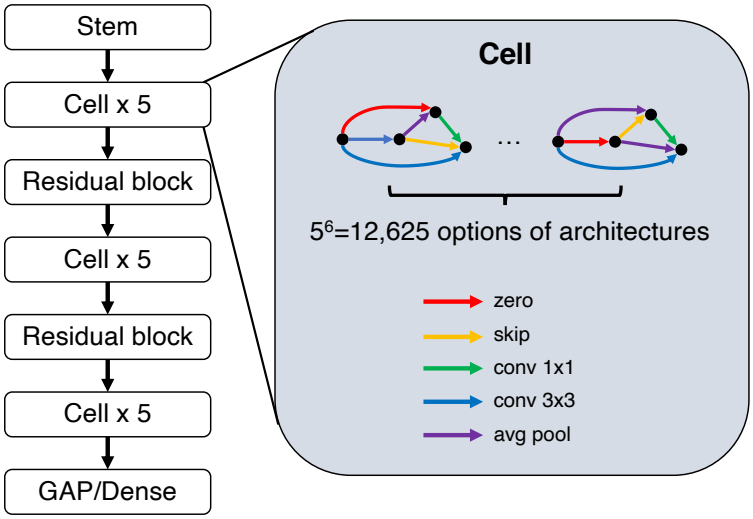
Finally, we compute the accuracy for the test cases that ranked from up to the top 5% prediction confidence to the entire dataset (100%). The majority voting among the 10 predictions is used when computing accuracy to avoid significant tremors when calculating AUC on a relatively small amount of data points.

Dataset	Content	Mean	std	Spearman Correlation Coefficient								
				ECG					CT			
				Length	NTK	SynFlow	ZiCo	Test reward	Length	NTK	SynFlow	ZiCo
ECG	Length	93.28	296.41									
	NTK	1.78×10^{9}	1.29×10^{10}	0.920								
	SynFlow	1.53	0.31	0.591	0.567							
	ZiCo	304.97	146.18	0.477	0.452	0.170						
	Test reward	0.62	0.02	0.307	0.362	0.016	0.437					
CT	Length	1179.89	3192.46	0.969	0.899	0.563	0.432	0.280				
	NTK	1.07×10^{5}	7.36×10^{5}	0.838	0.838	0.489	0.367	0.273	0.866			
	SynFlow	23.88	4.15	0.700	0.686	0.671	0.200	0.051	0.666	0.624		
	ZiCo	349.57	168.77	0.463	0.439	0.169	0.994	0.427	0.417	0.352	0.187	
	Test reward	0.85	0.03	0.428	0.406	0.071	0.091	0.099	0.435	0.487	0.265	0.063

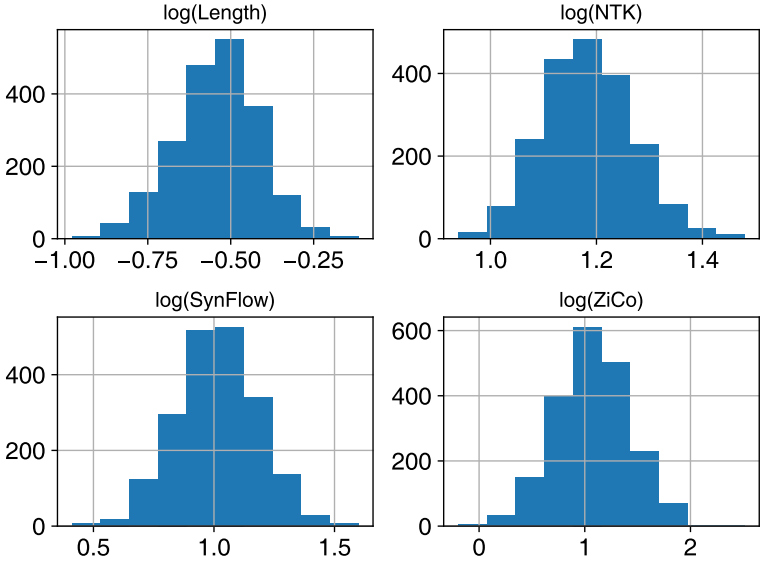
Supplementary Table 1: Training-free metric values, test rewards, and Spearman correlation coefficients across benchmark 150 architectures

Training-free metrics used	Spearman correlation with model performance		
	ECG	CT	Summation
All (Length + NTK + SynFlow + ZiCo)	0.41 ± 0.14	0.43 ± 0.16	**0.84**
NTK + SynFlow + ZiCo	0.38 ± 0.15	0.44 ± 0.16	0.82
Length + SynFlow + ZiCo	0.35 ± 0.15	0.28 ± 0.18	0.63
Length + NTK + ZiCo	0.45 ± 0.13	0.38 ± 0.17	0.83
Length + NTK + SynFlow	0.35 ± 0.15	0.48 ± 0.15	0.83

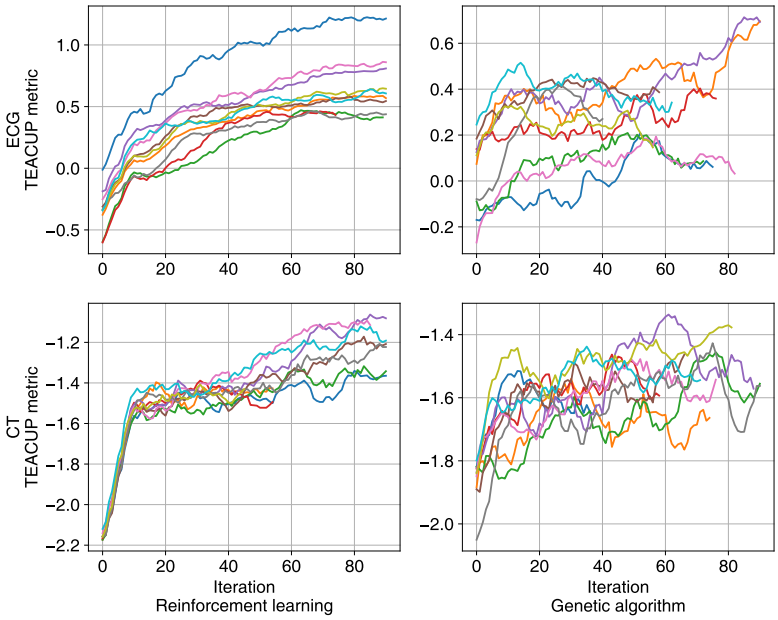
Supplementary Table 2: Bootstrap results ($B = 2000$) across different combinations of training-free metrics. Using all four training-free reach an overall best result.



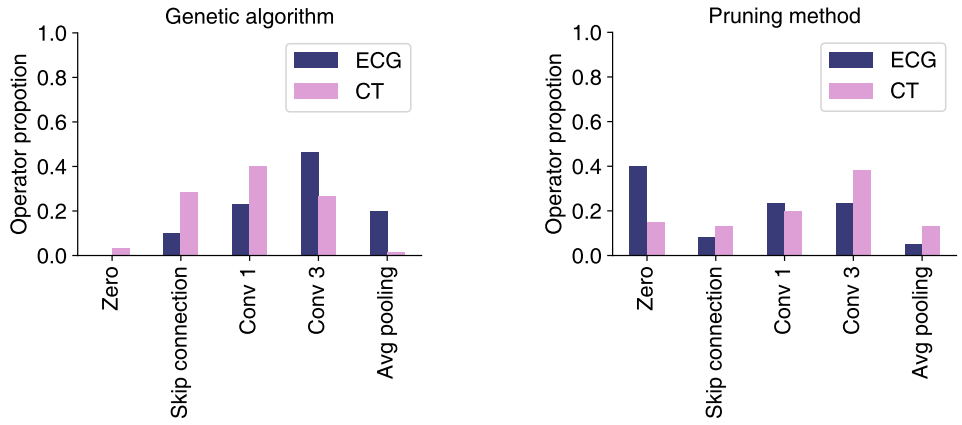
Supplementary Figure 1: Illustration of NAS-Bench-201 model space



Supplementary Figure 2: Bootstrap results of the coefficient distribution on the four log training metrics.



Supplementary Figure 3: Change of TEACUP metrics as iteration increases in reinforcement learning and genetic algorithm. Different colors represent different, independent search runs.



Supplementary Figure 4: The distribution of optimized architectures by our genetic algorithm (left; Sec. 5.5.2) and pruning method (right; Sec. 5.5.3).