GENNAPE: Towards Generalized Neural Architecture Performance Estimators

Keith G. Mills¹,²*, Fred X. Han², Jialin Zhang³, Fabian Chudak², Ali Safari Mamaghani¹, Mohammad Salameh², Wei Lu², Shangling Jui³, Di Niu¹

¹Department of Electrical and Computer Engineering, University of Alberta  
²Huawei Technologies, Edmonton, Alberta, Canada  
³Huawei Kirin Solution, Shanghai, China  

{fred.xuefei.han1, fabian.chudak, mohammad.salameh, jui.shangling}@huawei.com  
{zhangjialin10, robin.luwei}@hisilicon.com

Abstract

Predicting neural architecture performance is a challenging task and is crucial to neural architecture design and search. Existing approaches either rely on neural performance predictors which are limited to modeling architectures in a predefined design space involving specific sets of operators and connection rules, and cannot generalize to unseen architectures, or resort to Zero-Cost Proxies which are not always accurate. In this paper, we propose GENNAPE, a Generalized Neural Architecture Performance Estimator, which is pre-trained on open neural architecture benchmarks, and aims to generalize to completely unseen architectures through combined innovations in network representation, contrastive pre-training, and a fuzzy clustering-based predictor ensemble. Specifically, GENNAPE represents a given neural network as a Computation Graph (CG) of atomic operations which can model an arbitrary architecture. It first learns a graph encoder via Contrastive Learning to encourage network separation by topological features, and then trains multiple predictor heads, which are soft-aggregated according to the fuzzy membership of a neural network. Experiments show that GENNAPE pre-trained on NAS-Bench-101 can achieve superior transferability to 5 different public neural network benchmarks, including NAS-Bench-201, NAS-Bench-301, MobileNet and ResNet families under no or minimum fine-tuning. We further introduce 3 challenging newly labelled neural network benchmarks: HiAML, Inception and Two-Path, which can concentrate in narrow accuracy ranges. Extensive experiments show that GENNAPE can correctly discern high-performance architectures in these families. Finally, when paired with a search algorithm, GENNAPE can find architectures that improve accuracy while reducing FLOPs on three families.

Introduction

Understanding and predicting the performance of neural networks is crucial to automated neural architecture design and neural architecture search (NAS). In fact, the amount of time and computational resources required by a NAS scheme is critically dependent on how evaluation is performed. Early methods (Zoph and Le 2017) rely on training individual architectures from scratch via intensive GPU usage. Weight-sharing methodologies like DARTS (Liu, Simonyan, and Yang 2019) propose supernet models that superimpose the features of all architectures in a search space, substantially reducing the cost of search. DARTS further reduces this cost by searching for an architecture on a small image classification dataset like CIFAR-10 (Krizhevsky, Hinton et al. 2009) before transferring the found architecture to a larger benchmark dataset, like ImageNet (Russakovsky et al. 2015). In contrast, Once-for-All (OFA) (Cai et al. 2020) employs strategies to effectively pre-train a re-usable supernet on ImageNet prior to performing search. Newer methods have seen the introduction of Zero-Cost Proxies (Abdelfattah et al. 2021) which estimate performance from forward pass gradients.

Advancements in neural predictors (Tang et al. 2020; Wen et al. 2020) allow for faster inference time, which significantly lowers the architecture evaluation cost incurred in NAS. However, one major trade-off associated with using neural predictors is a lack of generalizability. The input to a neural predictor is usually a vector encoding of the architecture, whose scope is confined to the designated search space the predictor is meant to operate in. For example, to facilitate architecture search, Once-for-All, SemiNAS (Luo et al. 2020) and BANANAS (White, Neiswanger, and Savani 2021) design specialized predictors, each performing in an individually defined search space. Such a restriction seriously limits the scope that a search algorithm (Mills et al. 2021a,c) can explore, regardless of how effective it is.

Furthermore, as a separate new predictor must be trained for each new search space created, there is an implied additional heavy cost associated with the laborious and resource-intensive task of labelling the performance of a suitable amount of neural architecture samples in this search space, since these architecture samples must be trained from scratch for evaluation. Ideally, a generalized neural predictor would not be subject to these constraints. Not only would it be able to receive input from multiple search spaces, but it would learn an architecture representation that is robust enough to accurately estimate and rank the performance of...
networks from unseen search spaces.

In this paper we propose a Generalized Neural Architecture Performance Estimator (GENNAPE), to perform a feasibility check on pretraining a robust and generalizable neural predictor based only on open neural network benchmarks and transferring it to a wide range of unseen architecture families for accurate accuracy ranking. GENNAPE accepts input architectures from any search space through a generic Computation Graph representation, and performs a two-stage process of graph encoding and cluster-based prediction to achieve transferability to a new neural network design space under no or minimum fine-tuning. In designing GENNAPE, we make the following contributions:

First, we propose a generic embedding scheme based on Computation Graph (CG) representations of any neural networks and a self-supervised Contrastive Learning (CL) loss as the starting point of our predictor. Specifically, we represent an input neural network as a graph of atomic neural operations, e.g., convolutions, pooling, activation functions, rather than macro units like MBCConv blocks (Howard et al. 2019) that are specific to architecture families. This allows us to feed any neural network into our predictor. We then pass the CG through an encoder to obtain a fixed-length embedding. The graph encoder is trained by a carefully designed CL loss based on Laplacian Eigenvalues and dropout data augmentation, to distinguish graphical features between architecture families.

Second, we propose to learn multiple prediction heads, each being an MLP and focusing on learning the characteristics of a local cluster of architectures in the training family. We use Fuzzy C-Means (FCM) clustering to partition the training family (in our case, NAS-Bench-101 due to its abundance of labelled samples) into soft clusters, where a given input architecture can find soft memberships among the clusters. Based on the FCM partitions, we construct a Multi-Layer Perceptron-Ensemble (MLP-E) that makes prediction for an architecture by soft aggregating the MLP heads according to the architecture’s membership.

We verify the performance of GENNAPE on a wide range of public NAS families. Specifically, we pretrain our predictor on NAS-Bench-101, and treat the search spaces of NAS-Bench-201, NAS-Bench-301, ProxylessNAS (Cai, Zhu, and Han 2019), OFA-MobileNetV3, OFA-ResNet as unseen target families. We show that our predictor is able to achieve over 0.85 SRCC and over 0.9 NDCG on all of these target families with minimal fine-tuning. When integrated into a downstream NAS search algorithm, GENNAPE further improves high-accuracy architectures from NAS-Bench-101, NAS-Bench-201 and OFA-ResNet on ImageNet120.

To stress test the generalizability of GENNAPE, we further introduce three new challenging architecture benchmark sets: HiAML, Inception and Two-Path. These families contain certain properties, such as narrow accuracy range and ties, which make it difficult for a predictor to accurately learn architecture rankings, and thus serve as challenging benchmarks for future NAS research. We show that GENNAPE can obtain over 0.78 NDCG on all three families. Finally, we open-source these new benchmarks to facilitate further research on generalizable neural predictors.

### Related Work

Neural predictors rely on benchmark datasets, namely NAS-Bench-101 (Yang et al. 2019), 201 (Dong and Yang 2020) and 301 (Zela et al. 2022), which allow for performance querying. The downside to these methods is that every architecture must be trained from scratch, possibly multiple times. Arguably, supernet-based methods like OFA (Cai et al. 2020) fall under a unique category of benchmark, where sampled architectures can be immediately evaluated. However, this is still less resource intensive than training the network from scratch. Therefore, while benchmarks provide a rich repository of architecture statistics, an upfront resource cost is incurred, motivating the need for generalized predictors that can perform well on unseen families.

One method of low-resource neural prediction are Zero-Cost Proxy (ZCP); Abdelfattah et al. (2021) outline many variants. ZCP schemes involve gradient calculations as well as parameter salience. As Table 1 shows, while ZCP methods may achieve good results on some architecture families, their performance is inconsistent across families. In addition, ZCP methods require network instantiation to perform forward passes to compute corresponding measures, while neural predictors do not.

Contrastive Learning (CL) is based on distinguishing pairs of similar and dissimilar objects. This approach has been recently used successfully in image classification (Chen et al. 2020). Specifically, CL classifiers improve transferability on unseen data. Furthermore, fine-tuning on a very small fraction of the labeled data from a new dataset is enough to produce state of the art classifiers. Khosla et al. (2020) further extend this approach to the case when image labels are known during training. We focus on applying CL methods for finding vector representations of graphs.

Some deep learning schemes use Fuzzy C-Means clustering (FCM) to design interpretable models. Yeganejou, Dick, and Miller (2020) substitute the last MLP for an FCM clustering layer and classify images according to the highest membership value. By contrast, GENNAPE incorporates FCM into an MLP ensemble, using the membership values to perform a weighted sum of MLP heads.

---

1https://github.com/Ascend-Research/GENNAPE

| Family | Synflow | Jacov | Fisher | Snip | FLOPs |
|--------|---------|-------|--------|------|-------|
| NB-201 | 0.823   | 0.859 | 0.687  | 0.718| 0.001 |
| NB-301 | -0.210  | -0.190| -0.305 | -0.336| 0.578 |
| PN     | 0.086   | -0.022| 0.677  | 0.731| 0.689 |
| OFA-MBv3| 0.648  | 0.035 | 0.602  | 0.649| 0.614 |
| OFA-RN | 0.724   | -0.094| 0.651  | 0.752| 0.785 |
| HiAML  | 0.154   | -0.010| 0.102  | 0.168| 0.277 |
| Inception | 0.085 | 0.177 | -0.152 | 0.061| 0.412 |
| Two-Path | 0.227  | -0.071| 0.042  | -0.027| 0.333 |

Table 1: Spearman Rank Correlation coefficients for four Zero-Cost Proxy methods and a simple FLOPs-based predictor on all target families. Best results in bold.
Table 2: Architecture families in terms of dataset, number of CGs and accuracy range. Horizontal line demarcates existing benchmark families.

| Family      | Dataset       | #CGs | Acc. Range [%]          |
|-------------|---------------|------|-------------------------|
| NB-101      | CIFAR-10      | 50k  | [10.00, 94.09]          |
| NB-201      | CIFAR-10      | 1.1k | [46.25, 93.37]          |
| NB-301      | CIFAR-10      | 10k  | [88.28, 94.67]          |
| PN          | ImageNet      | 8.2k | [71.15, 77.83]          |
| OFA-MBv3    | ImageNet      | 7.5k | [73.56, 78.83]          |
| OFA-RN      | ImageNet      | 10k  | [75.25, 79.94]          |
| HiAML       | CIFAR-10      | 4.6k | [91.11, 93.44]          |
| Inception   | CIFAR-10      | 580  | [89.08, 94.03]          |
| Two-Path    | CIFAR-10      | 6.9k | [85.53, 92.34]          |
| ProxylessNAS| ImageNet      | 8.2k | [71.15, 77.83]          |
| Once-for-All-MobileNetV3 | ImageNet | 7.5k | [73.56, 78.83] |
| Once-for-All-ResNet | ImageNet | 10k | [75.25, 79.94] |

†We only use the 4096 architectures that do not contain the ‘none’ operation.

Figure 1: The architecture backbone of HiAML, containing 4 stages. Each stage contains 2 identical blocks.

Figure 2: Accuracy [%] and FLOPs [10e6] histograms for the new benchmarks. Accuracy binning interval is 0.01%.

Architectures Families

We describe the neural network families used in this paper. Table 2 provides an overall summary of each family.

Public NAS Benchmark Families

NAS-Bench-101 (NB-101) is one of the first and largest benchmarks for NAS. It consists of 423k unique architectures, individually evaluated on CIFAR-10. The architectures are cell-based, where each cell is a Directed Acyclic Graph (DAG) containing operations, stacked repeatedly to form a network. We sample 50k random architectures from this family to form our CG training family.

NAS-Bench-201 (NB-201) and NAS-Bench-301 (NB-301) are two additional benchmarks. Like NB-101, architectures consist of a fixed topology of cells, except they follow the DARTS search space. Additionally, NB-201 only contains 15.6k evaluated architectures, while NB-301 is a surrogate predictor for the DARTS search space. Therefore, we treat both as test families.

ProxylessNAS (PN) (Cai, Zhu, and Han 2019) and Once-for-All-MobileNetV3 (OFA-MBv3) (Cai et al. 2020) are based on the MobileNet (Howard et al. 2019) architecture families, with PN and OFA-MBv3 implementing versions 2 and 3, respectively. Once-for-All-ResNet (OFA-RN) is based on the classical ResNet-50 (He et al. 2016) topology. All three evaluate on ImageNet. Architectures consist of searchable, macro features where the number of blocks is variable. We refer the reader to Mills et al. (2021b) for further details regarding PN, OFA-MBv3 and OFA-RN.

Introduced NAS Benchmark Families

We introduce the key attributes of the new network families and provide additional information in the supplementary. For all families, we define an operator as a bundle of primitive operations, e.g., Conv3x3-BN-ReLU. A block is a set of operators with different connection topologies, while a stage contains a repetition of the same block type and the backbone determines how we connect blocks to form distinct networks. For these network families, accuracy values are rounded to four decimal places. That is, we would represent 91.23% as a terminating 0.9123. Figure 2 provides accuracy and FLOPs histograms.

HiAML is a custom family of networks inspired by NAS-Bench-101. Networks are built using 14 pre-defined operation block structures found by GA-NAS (Changiz Rezaei et al. 2021). Specifically, a pair of identical blocks form a stage, and there are 4 stages per network. Since August 2021, HiAML networks have been in use as the backbone feature extractor in Huawei’s mobile Facial Landmark Detection application, i.e., the family is not limited to Image Classification. HiAML networks are efficient enough to run entirely on the less powerful Ascend Tiny Core, which is a part of Da Vinci Architecture 2.0 in Huawei’s Neural Processing Unit (NPU). Figure 1 provides a backbone illustration of HiAML. As shown in Table 2, we observe that HiAML has the narrowest performance profile among all families considered, as 4.6k architectures fit into an accuracy range of only 2.33%. As a comparison, OFA-MBv3 has the
second smallest accuracy range of 4.25%. This characteristic makes accurate prediction on HiAML networks very difficult as there are many ties as Figure 2(a) shows.

**Inception** takes inspiration from the Inception-v4 (Szegedy et al. 2016) classification networks in literature. Our custom Inception networks consist of a single-path backbone with 3 stacked stages, and inside each stage, there is a repetition of 2 to 4 blocks. Like the original Inception, each block may contain up to 4 branching paths. The main advantage of Inception is that channels are divided among the paths, leading to lower computation costs. The backbone follows a single-path topology with 3 stages. Currently, Inception networks serve as feature extractors for Huawei’s Facial Recognition framework.

**Two-Path** networks are designed to be more lightweight than other families. Like the other benchmarks, Huawei utilizes Two-Path networks for real-world applications, specifically multi-frame Super Resolution and 4k LivePhoto applications. As the name suggests, the backbone consists of two branching paths. On each path, there are 2 to 4 blocks, each a sequence of up to 3 operators. One may view the Two-path family as the complement of the Inception family. An Inception network has a single-path backbone with branching paths inside blocks. But a Two-Path network has two branching paths of blocks, while each block is a single path of operators.

**Methodology**

Figure 3 visualizes our scheme. A predictor that generalizes to arbitrary architecture families should accept a generalizable neural network representation as input. We consider the Computation Graph (CG) of a network structure based on the graph structures deep learning libraries like TensorFlow (Abadi et al. 2016) and PyTorch (Paszke et al. 2019) use to facilitate backpropagation. Specifically, nodes refer to atomic operators in a network, e.g., Convolution, Linear layers, Pooling layers, non-linearities and tensor operations like mean, add and concat. Edges are directed and featureless.

As graphs can contain an arbitrary number of nodes and edges, it is necessary to process them into a common, fixed-length format prior to prediction. Therefore, in this section, we introduce our Contrastive Loss-based (CL) graph embeddings scheme, the Fuzzy C-Means (FCM) ensemble that forms the GENNAPE predictor and describe a FLOPs-based accuracy transform.

**Embeddings from Contrastive Learning**

We use Contrastive Learning (CL) to learn vector representations of CGs by adapting the two key components of SimCLR (Chen et al. 2020). The first is data augmentation, where perturbations produce two different copies of every
image in a batch. The second is the CL loss function, which aims to identify positive and negative pairs within a batch. Unlike images, random perturbations of CGs are unlikely to belong to the same architecture family. Therefore, to generate embedding variation, we adopt dropout augmentation from Gao, Yao, and Chen (2021).

Consider a batch of \( N \) CGs. We use dropout and apply a graph encoder twice to make 2 copies of each CG. We then use an MLP head to project these embeddings of length \( e \) into a lower dimensional space \( p \) to optimize the CL loss. Let \( I = \{ h_1, h_2, \ldots, h_{2N} \} \subset \mathbb{R}^e \) be the vector representations, and let \( z_i = \text{proj}(h_i) \in \{ ||z|| = 1 : z \in \mathbb{R}^p \} \) be their projections. We define the cosine similarity \( \text{sim}(z_i, z_j) = z_i \cdot z_j / \tau \), with temperature \( \tau > 0 \) and \( \cdot \) as the dot product. Then, our CL loss function is

\[
\mathcal{L}_{CL} = - \sum_{i \in I} \sum_{\ell \neq i} \alpha^{(i)}_\ell \log \frac{\exp(\text{sim}(z_i, z_\ell))}{\sum_{r \neq i} \exp(\text{sim}(z_i, z_r))},
\]

where \( \alpha^{(i)}_\ell \geq 0 \) and \( \sum_{\ell \neq i} \alpha^{(i)}_\ell = 1 \). For each computational graph \( i \), we calculate the convex combination \( \alpha^{(i)}_\ell \) using the spectral distance of the underlying undirected graph as a similarity measure. We use topological properties to group architectures in the same family or separate architectures from different families, because the arrangement of operations differs for each family. Therefore, given a computation graph, let \( G \) be its underlying undirected graph. The normalized Laplacian of \( G \) encodes important connectivity properties. Furthermore, it allows us to define a (pseudo) distance between computation graphs by evaluating the Euclidean distance between the corresponding \( q = 21 \) smallest eigenvalues (Dwivedi and Bresson 2021). We use \( \sigma(g_1, g_2) \) to denote the distance between computation graphs \( g_1, g_2 \) and define \( \alpha^{(i)}_* \) to be the softmax of \( \sigma(i, *) \).

We train a CL encoder on NB-101 and use it to infer embeddings for all test families. A 2-dimensional representation of the embeddings is shown in Figure 5. We note two advantages of the CL encoder. First, it effectively separates each test family into distinct clusters with minimal overlap. By contrast, the CL encoder divides the training family, NB-101, into many smaller clusters. This is an important finding. While NB-101 contains many architectures, it only contains 3 candidate operations, meaning that most of the diversity amongst these architectures comes from topological differences, i.e., Laplacian eigenvalues, which play an important role in training our CL encoder. As such, each NB-101 cluster may reside closer to a given test family than others. Next, we exploit these observations to build a predictor ensemble.

**Soft-Clustering Ensemble**

We use MLP Ensembles (MLP-E) in GENNAPE to make predictions. Each head in the MLP-E is responsible for a region of the embedding space, although these regions can partially overlap with each other. That is because we use Fuzzy C-Means (FCM) (Bezdek, Ehrlich, and Full 1984) clustering to divide up the embedding space, and FCM represents data using continuous memberships with respect to all clusters. The output of the MLP-E is defined as,

\[
y_i^* = \sum_{j=1}^{C} U_{i,j} f_j(h_i),
\]

where \( h_i \) and \( y_i^* \) are the graph embedding and target prediction for a CG, of which there are \( N \) in total, \( C \) is the number of heads, \( f_j \) is a head and \( U_{i,j} \) is the membership for CG \( i \) across cluster \( j \). FCM membership values \( U \in (0, 1)^{N \times C} \) are positive numbers that sum to 1 for a given sample; \( \forall i \sum_{j=1}^{C} U_{i,j} = 1 \).

Like other distance-based clustering algorithms, FCM consists of centroids that are computed alongside membership across all data samples and \( N \) clusters is less than \( m \) is known as the fuzzification coefficient, and is typically a value of 2 or greater (Pedrycz and Song 2012), which controls the degree of overlap between the clusters. The FCM algorithm alternates between Equations 2 and 3 until either a pre-specified number of iterations has been reached, or the following stopping criteria is satisfied:

\[
\max_{i,j} |U_{i,j}(t + 1) - U_{i,j}(t)| \leq \epsilon,
\]

where \( t \) is a time parameterization of \( U \). Essentially, clustering convergence is reached once the maximum change in membership across all \( N \) data samples and \( C \) clusters is less than \( \epsilon = 1 - e^{-5} \).

When applying FCM to GENNAPE, we compute cluster centroids on the training family. For test families, we infer membership from existing centroids using Equation 2.
A simple MLP is not guaranteed to generalize to features of families it has not seen. However, we can aid generalizability by only having the MLP learn a specific region in the feature space and weighing its predictions based on the distance between the unseen features and the learnt region. Moreover, we can divide up the feature space with an MLP-E. Each head learns to specialize on a subset of the embedding features for the training family, as well as an association with nearby, unseen features, depending on memberships.

**FLOPs-based Features and Transforms**

Every network has an associated FLOPs value, or floating point operations required for the forward pass. GENNAPE uses FLOPs to transform the regression targets. While the downstream goal of any regressor is to estimate the accuracy of a neural network, GENNAPE estimates the following,

\[ y_i = \frac{A_i}{\log_{10}(F_i + 1) + 1}, \]

where \( y_i \) is the label, \( A_i \) is the accuracy of a network, \( F_i \) is the FLOPs, measured in gigaFLOPs and \( Z \) is the Z-Score. The inverse of this transformation can be applied to a prediction to obtain the corresponding accuracy prediction. We use the standardization statistics from the NB-101 training set for all test families, and we prune out NB-101 architectures whose accuracy is below 80% when implementing Equation 5. This leaves us with 49k architectures.

**Experimental Results**

In this Section, we describe our experimental procedure and results in terms of regression and correlation metrics. Finally, we apply GENNAPE to NAS and perform search.

**NAS-Bench-101 Comparison**

We use NB-101 as the training family and treat all others as unseen test families. We split the NB-101 architectures into a training set with 80% of the data, and two separate validations sets that contain 10% each. We apply Equation 5 using the mean and standard deviation from the training partition to standardize the validation sets. The model trains for 40 epochs. We evaluate it on the first validation set at every epoch without applying the inverse of Equation 5 to track loss statistics on transformed labels. Once training is complete, we further evaluate the model on the second validation set, this time applying the inverse of Equation 5 on predictions to calculate performance compared to the ground truth accuracy. For each scheme, we train a model 5 times using different random seeds. The overall best model is the one that achieves the highest rank correlation on the second NB-101 validation set. We provide hyperparameters and other training details in the supplementary materials.

We consider several types of regressor variants in our experiments. ‘CL+FCM+T’ uses CL graph embeddings as well as the FCM MLP-E to predict performance labels transformed using Equation 5. ‘CL+T’ removes the FCM clustering ensemble and uses a single MLP while ‘CL+FCM’ keeps the clustering ensemble while removing Equation 5. Finally, ‘CL’ drops both Equation 5 and the FCM ensemble to predict accuracy directly using the CL embeddings. As baselines, we consider several GNN regressors such as \( k \)-GNN (Morris et al. 2019), Graph Convolutional Networks (GCN) (Welling and Kipf 2016) and Graph Isomorphism Networks (GIN) (Xu et al. 2019) and train them end-to-end on CGs. Finally, we also consider several related neural predictors, such as Neural Predictor for NAS (NPN) (Wen et al. 2020), BANANAS (White, Neiswanger, and Savani 2021) and TNASP (Lu et al. 2021). These predictors operate on the NB-101 family, however since they do not represent architectures using CGs, they are not transferable.

We consider two predictor performance metrics. The first is Mean Absolute Error (MAE), which measures predictor accuracy with respect to ground-truth labels. We report MAE as a percentage where lower is better. The second is Spearman’s Rank Correlation (SRCC) which judges a predictor’s ability to rank a population of architectures. SRCC ranges from -1 to 1 and in our case higher is better as that represents positive agreement with the ground-truth.

Table 3 enumerates our results for NB-101. We note the effectiveness of Equation 5, which allows us to achieve MAE scores below 1% on average. Moreover, the inclusion of CL and FCM further improve this result to a minimum of 0.59%. In terms of ranking correlation, CL+FCM+T achieves the second highest SRCC at 0.930. While NPN obtains a slightly higher SRCC, that is the limit of its capabilities, since NPN on NB-101 cannot perform predictions for architectures from other families. Finally, in terms of other GNN CG baselines, the \( k \)-GNN excels. While it does not obtain the best MAE or SRCC, it is distinctly better than both the GCN and GIN. Therefore, we continue to use it as a generalizable predictor baseline.

**Transferability Prediction Error**

When evaluating on test families, we consider two cases. The first is zero-shot transfer performance, where we measure a given metric on all architecture CGs in a family without fine-tuning the predictor. In the second case, we sam-

| Method          | MAE       | SRCC       |
|-----------------|-----------|------------|
| NPN            | 0.93 ± 0.003 | 0.934 ± 0.003 |
| BANANAS        | 1.40 ± 0.066 | 0.834 ± 0.002 |
| TNASP          | 1.23 ± 0.022 | 0.918 ± 0.002 |
| GCN            | 1.78 ± 0.063 | 0.732 ± 0.034 |
| GIN            | 1.72 ± 0.044 | 0.735 ± 0.035 |
| \( k \)-GNN     | 1.61 ± 0.080 | 0.814 ± 0.020 |
| CL             | 1.51 ± 0.176 | 0.874 ± 0.009 |
| CL+FCM        | 1.19 ± 0.124 | 0.896 ± 0.003 |
| CL+T          | 0.65 ± 0.080 | 0.921 ± 0.003 |
| CL+FCM+T      | 0.59 ± 0.011 | 0.930 ± 0.002 |

† Obtained by running their model using our 50k NB-101 architectures, using the same data splits.
ple 50 architectures at random, fine-tune for 100 epochs and then evaluate on the remaining, held-out architectures. We perform fine-tuning 5 separate times on different seeds, ensuring that we select the same 50 architectures per seed. Evaluating MAE performance on test families is applicable in the fine-tuning setting as that provides the predictor some information on the actual distribution of target labels. When fine-tuning using Equation 5, we use the mean and standard deviation information from the NB-101 training partition to standardize labels for the 50 fine-tuning samples, and to recover accuracy on the remaining samples at inference.

Table 4 provides test family MAE results for the \( k \)-GNN baseline and several CL-based regressors. Once again, we observe how regressors that use Equation 5 achieve the lowest MAE on 6 of 9 families, including all ImageNet families and Two-Path. The \( k \)-GNN achieves good performance on NB-201, likely because it has the second largest range after NB-101. On HiAML, Inception and Two-Path, the results are generally very close, showing how difficult it can be to predict accuracy for them.

### Unseen Architecture Ranking Performance

GENNAPE consists of a weighted average of predictions from several models. The first two are the aforementioned ‘CL+FCM+T’ and ‘CL+T’ regressors. We also include two pairwise classifiers, that, instead of estimating accuracy, predict which CG in a pair has higher accuracy. We label these as ‘CL+Pairwise+FCM’ and ‘CL+Pairwise’, differentiated by whether they have FCM and an MLP ensemble. We also include the \( k \)-GNN and the FLOPs predictor from Table 1. For zero-shot performance, GENNAPE weighs the output of all predictors equally. When fine-tuning, we calculate weights by taking the softmax of the Kendall’s Tau (KT) result each individual predictor obtains on the fine-tuning samples. The supplementary materials contains details and an ablation study of these components.

Table 5 provides SRCC results across all test families. We observe how GENNAPE achieves SRCC values above 0.85 on all public benchmarks. In fact, the only public family GENNAPE does not achieve zero-shot SRCC above 0.5 on is NB-301, however, at 0.3214 this is still much higher than the \( k \)-GNN and any of the ZCPs from Table 1. Moreover, fine-tuning greatly improves the performance of GENNAPE on NB-301. Meanwhile, the \( k \)-GNN does achieve higher zero-shot SRCC on OFA-RN, but the improvement is less than 0.1 SRCC. On all other public benchmarks it fails to achieve over 0.5 SRCC in the zero-shot setting.

### Table 4: Mean Absolute Error [%] results across all families and regressors. Lower is better. Best result in bold. Results averaged across 5 random seeds.

| Family   | \( k \)-GNN | CL  | CL+T  | CL+FCM+T |
|----------|-------------|-----|-------|----------|
| NB-101   | 1.61        | 1.51| 0.65  | 0.59     |
| NB-201   | **1.80**    | 2.81| 1.93  | 2.05     |
| NB-301   | 0.46        | 0.85| 0.43  | 0.57     |
| PN       | 0.56        | 0.40| 0.35  | 0.33     |
| OFA-MBv3 | 0.61        | 0.35| 0.31  | 0.27     |
| OFA-RN   | 0.40        | 0.44| 0.45  | 0.37     |
| HiAML    | **0.35**    | 0.45| 0.40  | 0.39     |
| Inception| 0.60        | 0.82| 0.71  | 0.86     |
| Two-Path | 0.80        | 1.11| **0.71**| 0.72     |

### Table 5: Spearman Rank Correlation results across test families in the zero-shot transfer and fine-tuning (w/ FT) contexts. Higher is better. Best results in bold. Fine-tuning results averaged across 5 random seeds.

| Family   | \( k \)-GNN | GENNAPE   |
|----------|-------------|-----------|
| NB-201   | 0.4930      | **0.8146**|
| w/ FT    | 0.8606 ± 0.0245 | **0.9103 ± 0.0114** |
| NB-301   | 0.0642      | **0.3214**|
| w/ FT    | 0.8584 ± 0.0290 | **0.8825 ± 0.0134** |
| PN       | 0.0703      | **0.8213**|
| w/ FT    | 0.7559 ± 0.0621 | **0.9506 ± 0.0039** |
| OFA-MBv3 | 0.4345      | **0.8660**|
| w/ FT    | 0.6862 ± 0.0253 | **0.9449 ± 0.0015** |
| OFA-RN   | **0.5721**  | 0.5115    |
| w/ FT    | 0.9102 ± 0.0146 | **0.9114 ± 0.0063** |
| HiAML    | -0.1211     | **0.4331**|
| w/ FT    | 0.4300 ± 0.0507 | 0.4169 ± 0.0479 |
| Inception| -0.2045     | **0.4249**|
| w/ FT    | 0.3340 ± 0.0793 | **0.5524 ± 0.0166** |
| Two-Path | 0.1970      | **0.3413**|
| w/ FT    | 0.3694 ± 0.0406 | **0.4875 ± 0.0311** |

S SRC measures correlation over all \( N \) samples and assigns equal weight to each. However, the concerns of NAS relate to finding high-performance architectures. Therefore, for further evaluation of GENNAPE, we consider another metric that assigns disproportionate importance. Following AceNAS (Zhang et al. 2021), we adopt Normalized Discounted Cumulative Gain (NDCG), an Information Retrieval (IR) metric where queried objects are assigned a relevance score. In IR, it is more important to properly order objects with higher relevance. This goal is analogous to the concerns of NAS. Accuracy is relevance.

We list our findings in Table 6. GENNAPE clearly outperforms the \( k \)-GNN baseline in both contexts. The sole exception is OFA-R50, likely due to NB-101 architectures being partially based on ResNets in terms of operation choice.
We apply this routine to the best CIFAR-10 architectures and fine-tuning (w/ FT) context. Values are reported in the range \([0, 1]\) and higher is better. Best results in bold. Fine-tuning results averaged across 5 random seeds.

| Model       | Dataset   | FLOPs    | Top-1 Acc. (%) |
|-------------|-----------|----------|---------------|
| NB-101-Best | CIFAR-10  | 11.72G   | 94.97         |
| NB-101-Search | CIFAR-10  | 9.49G    | 95.05         |
| NB-201-Best | CIFAR-10  | 313M     | 93.27         |
| NB-201-Search | CIFAR-10  | 283M     | 93.62         |
| OFA-RN-Input | ImageNet120 | 12.13G | 80.62         |
| OFA-RN-Search | ImageNet120 | 9.46G   | 81.08         |

Table 7: Evaluation results for architectures found using CL+FCM+T predictors and our CG-based search algorithm.

120 classes of ImageNet to reduce carbon footprint. We use the fine-tuning CL+FCM+T predictors in these experiments. Search takes less than 3 hours. Inference on our predictors is fast. The primary bottleneck stems from the search algorithm determining whether newly found CGs are feasible as classification neural networks.

Table 7 summarizes the results, demonstrating the power of GENNAPE to improve upon existing, high-performance architectures, in terms of accuracy and FLOPs, on three public families. Notably, the OFA-ResNet architecture found by search achieves 22% FLOPs reduction while improving accuracy by 0.4%. Moreover, these results demonstrate the utility of our CG framework, as the atomic operation mutations result in architectures that are not exact fits to the original search spaces. Using CGs, we can modify sections like the stem or head that are typically fixed, and introduce new operation sequences. For example, the NB-201 architecture we compare to and outperform was already the best on CIFAR-10. Therefore, a typical neural predictor for NB-201 would not be able to adequately process and predict performance for architectures found by our search algorithm.

**Conclusion**

In this paper, we introduce GENNAPE, or Generalized Neural Architecture Performance Estimators, to address issues present in neural predictor design. Namely, that neural predictors typically operate within a fixed search space, and are not generalizable to unseen architecture families. GENNAPE receives Computation Graphs representing arbitrary network architectures as input, before using a Contrastive Learning encoder to generate embeddings. The embeddings pass through an MLP Ensemble, and we compute the prediction using a weighted summation according to Fuzzy C-Means clustering memberships. We extensively test GENNAPE against a number of known neural predictors on NB-101 and show that it yields high MAE and SRCC performance. Experimental results demonstrate the generalizability of our scheme in zero-shot and fine-tuning contexts in terms of SRCC and NDCG@10. When applied to search, GENNAPE can improve upon existing, high-performance NB-101, NB-201 and OFA-ResNet architectures. Finally, we introduce three new challenge families: HiAML, Inception and Two-Path, as open-source benchmarks.
Welling, M.; and Kipf, T. N. 2016. Semi-Supervised Classification With Graph Convolutional Networks. In J. International Conference on Learning Representations (ICLR 2017).

Wen, W.; Liu, H.; Chen, Y.; Li, H.; Bender, G.; and Kindermans, P.-J. 2020. Neural Predictor for Neural Architecture Search. In European Conference on Computer Vision, 660–676. Springer.

White, C.; Neiswanger, W.; and Savani, Y. 2021. BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 10293–10301.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019.

Yeganejou, M.; Dick, S.; and Miller, J. 2020. Interpretable Deep Convolutional Fuzzy Classifier. IEEE Transactions on Fuzzy Systems, 28(7): 1407–1419.

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, 7105–7114.

Zela, A.; Siems, J. N.; Zimmer, L.; Lukasik, J.; Keuper, M.; and Hutter, F. 2022. Surrogate NAS Benchmarks: Going Beyond the Limited Search Spaces of Tabular NAS Benchmarks. In International Conference on Learning Representations.

Zhang, Y.; Yan, C.; Zhang, Q.; Zhang, L. L.; Yang, Y.; Gao, X.; and Yang, Y. 2021. AceNAS: Learning to Rank Ace Neural Architectures with Weak Supervision of Weight Sharing. arXiv preprint arXiv:2108.03001.

Zoph, B.; and Le, Q. 2017. Neural Architecture Search with Reinforcement Learning. In International Conference on Learning Representations.