AIO-P: Expanding Neural Performance Predictors beyond Image Classification

Keith G. Mills[1,2*], Di Niu[1], Mohammad Salameh[2], Weichen Qiu[1], Fred X. Han[2], Puyuan Liu[2], Jialin Zhang[3], Wei Lu[2], Shangling Jui[3]

[1]Department of Electrical and Computer Engineering, University of Alberta
[2]Huawei Technologies, Edmonton, Alberta, Canada
[3]Huawei Kirin Solution, Shanghai, China
{kgmills, dniu, wqiu1}@ualberta.ca
{mohammad.salameh, fred.xuefei.han1, puyuan.liu, jui.shangling}@huawei.com
{zhangjialin10, robin.luwei}@hisilicon.com

Abstract

Evaluating neural network performance is critical to deep neural network design but a costly procedure. Neural predictors provide an efficient solution by treating architectures as samples and learning to estimate their performance on a given task. However, existing predictors are task-dependent, predominantly estimating neural network performance on image classification benchmarks. They are also search-space dependent; each predictor is designed to make predictions for a specific architecture search space with predefined topologies and set of operations. In this paper, we propose a novel All-in-One Predictor (AIO-P), which aims to pretrain neural predictors on architecture examples from multiple, separate computer vision (CV) task domains and multiple architecture spaces, and then transfer to unseen downstream CV tasks or neural architectures. We describe our proposed techniques for general graph representation, efficient predictor pretraining and knowledge infusion techniques, as well as methods to transfer to downstream tasks/spaces. Extensive experimental results show that AIO-P can achieve Mean Absolute Error (MAE) and Spearman's Rank Correlation (SRCC) below 1% and above 0.5, respectively, on a breadth of target downstream CV tasks with or without fine-tuning, outperforming a number of baselines. Moreover, AIO-P can directly transfer to new architectures not seen during training, accurately rank them and serve as an effective performance estimator when paired with an algorithm designed to preserve performance while reducing FLOPs.

Introduction

Performance evaluation of neural network models is resource and time consuming. Several factors contribute to its expensiveness, such as task complexity, training dataset size, architecture topology, and training time. It is yet a main component and the bottleneck in Neural Architecture Search (NAS) (Elsken et al. 2019). Early NAS approaches train sampled architectures to completion during search (Zoph and Le 2017) while later approaches adopt weight-sharing supernet approaches (Pham et al. 2018; Liu, Simonyan, and Yang 2019; Changiz Rezaei et al. 2021; Mills et al. 2021a,c), which reduce the computational burden but do not eliminate it. Advanced supernet schemes like Once-for-All (OFA) (Cai et al. 2020) and BootstrapNAS (Munoz et al. 2022) introduce progressive shrinking to train a reusable supernet. Specifically, OFA supernets are robust enough that individual architectures can be sampled for immediate evaluation on ImageNet (Russakovsky et al. 2015).

Zero-Cost Proxies (ZCP) (Abdelfattah et al. 2021) are a recent development aiming to correlate performance with gradient statistics and thus can generalize to any type of network. However, the efficacy of ZCP methods depend on the architecture and task and may not be always reliable. Other recent schemes like NAS-Bench-301 (Zela et al. 2022), SemiNAS (Luo et al. 2020), TNASP (Lu et al. 2021b) and WeakNAS (Wu et al. 2021) develop neural predictors that estimate architecture performance from network topology and operation features. However, they model customized architectures in specific search spaces, e.g., NAS-Bench-101 (Ying et al. 2019) and 201 (Dong and Yang 2020) for common benchmark tasks like CIFAR image classification (Krizhevsky, Hinton et al. 2009), and cannot be directly transferred to other challenging tasks like pose estimation or segmentation or to architectures with new types of topologies/connections.

In this paper, we propose All-in-One Predictor (AIO-P), a multi-task neural performance predictor which achieves cross-task and cross-search-space transferability via predictor pretraining and domain-specific knowledge injection. AIO-P uses Computational Graphs (CG) to represent neural architectures, which is lower-level information extracted from TensorFlow execution and thus can model general types of architectures. We make the observation that many CV architectures consist of a *body* (e.g., ResNet) that performs feature extraction and a *head* that uses extracted features to generate task-specific outputs. Figure 1 illustrates how architectures can be constructed for different tasks by combining various types of bodies and heads. Based on such network representations, we introduce an effective transfer learning scheme to first pretrain AIO-P on image classification benchmarks, then infuse domain knowledge from other

[*]Work done during an internship at Huawei.

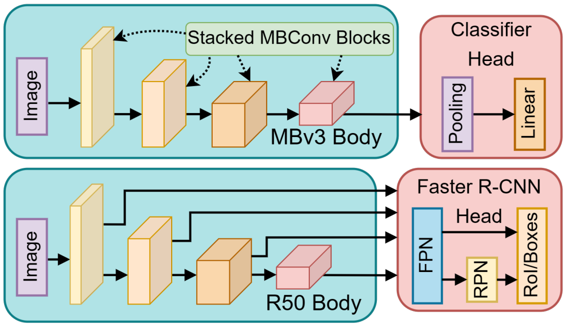

Figure 1: Network bodies from classification search spaces can pair with heads for different tasks. A simple classification head has pooling and a linear layer. Faster R-CNN (Ren et al. 2015) uses a Feature Pyramid Network (FPN) with feature maps of different sizes. FPN feeds a Regional Proposal Network (RPN) and a Region of Interest module (RoI) to estimate bounding boxes.

tasks or architecture spaces efficiently, and finally transfer to a downstream task with minimum or no fine-tuning. Specifically, we propose the following techniques to achieve transferability to downstream tasks:

First, we introduce K-Adapters (Wang et al. 2021), originally used to inject domain knowledge into language models, into a GNN predictor pretrained on network benchmarks for image classification (IC) such that the predictor can infuse knowledge from segmentation/detection tasks or other network topologies. We then transfer the learned model to perform predictions on potentially unseen downstream tasks or architectures.

Second, we propose an efficient learning scheme to train AIO-P, and especially adapters, by a pseudo-labeling scheme for task-specific network performance. In order to reduce the high cost associated with labeling each individual architecture's performance on a given task, we propose to train a weight-sharing task head for all body architectures in an entire search space, e.g., all variants of MobileNetV3 (Howard et al. 2019) in OFA. After training the shared head, it can pair with any architecture body and fine-tune for a few minutes to obtain a pseudo-label that is positively correlated with the true label seen when individually training the architecture for several hours. Moreover, we further propose a latent representation sampling technique to enhance the correlations of pseudo labels to true performance labels.

Third, since performance metrics and their distributions differ by task, we use several scaling techniques such as standardization and FLOPs-based transform to rescale labels when adapting AIO-P to a downstream task. AIO-P learns a unitless understanding of architecture performance that can then revert into a task-appropriate metric like Average Precision (AP) or mean Intersection over Union (mIoU).

Through extensive experiments, we demonstrate that AIO-P pretrained on one or two tasks in addition to classification is able to transfer to predict neural network performance for a diverse range of CV tasks including 2D Human Pose Estimation (HPE), Object Detection (OD), Instance Segmentation (IS), Semantic Segmentation (SS), and Panoptic Segmentation (PS). AIO-P consistently achieves a Mean Absolute Error (MAE) below 1% on task-specific metrics and a Spearman's Rank Correlation Coefficient (SRCC) above 0.5, outperforming both ZCP and GNN baselines under zero-shot transfer and minimum fine-tuning settings. In addition, AIO-P is able to correctly rank architectures in foreign model zoos whose body networks are different from those observed in training, including DeepLab (Chen et al. 2018) and TF-Slim (Guadarrama and Silberman 2016). Finally, by pairing AIO-P with a search algorithm, we can optimize a proprietary facial recognition model to preserve performance while reducing FLOPs by over 13.5%. We open-source[1] our data, code, and predictor design to advance research in this field.

Related Work

Benchmark datasets and neural predictors provide a quick avenue for performance estimation. Arguably, the most significant difference is how performance is queried. For example, NAS-Bench-101 and 201 contain 423k and 15.6k architectures, respectively. They store individual architecture performances on a look-up table. By contrast, NAS-Bench-301 operates on the DARTS search space, which contains 10^{18} architectures which are too many to evaluate individually. Instead, they train a neural predictor (Zela et al. 2022; Luo et al. 2020; Lu et al. 2021b; Wu et al. 2021). In both cases, architecture configurations define the keys to the table or predictor input, and serve to showcase the limitations of these approaches. Specifically, these configurations are for micro, cell-based NAS, where a network is built by stacking identical cell structures. These configurations generally assume details like latent representation sizes and the number of cells in the network to be constant.

On the other hand, OFA networks (Cai et al. 2020) use macro search spaces where networks are built by individually selecting and then stacking pre-defined blocks. It searches over the number of blocks in the network, kernel size and channel expansions (Mills et al. 2021b) within a block. Regardless, both approaches assume the stem and head of the network to be a nonsearchable fixed structure. This is acceptable when considering one task. They also abstract multiple operations into pre-defined sequences, e.g., MBConv block, that is specific to the search space and network body and thus may not exist in the head. By contrast, we aim to predict performance across various tasks with different heads and data sizes. Thus, we require a more robust and generalizable data format. AIO-P uses Compute Graphs (CG) as input, which incorporates full network topological details, latent tensors size, and node features.

A few approaches appear in the literature regarding NAS for CV tasks other than IC. For example, Ding et al. (2022) use NAS on nine tasks, including IC and SS. They perform a search to find architectures that provide high-performance on multiple objectives. Auto-DeepLab (Liu et al. 2019) reconfigure DARTS to perform variable upsampling/down-

[1]https://github.com/Ascend-Research/AIO-P

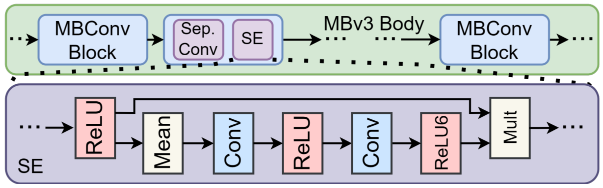

Figure 2: Example CG subgraph of a Squeeze-and-Excite (SE) module (Hu, Shen, and Sun 2018) in MBv3. We store properties like kernel size and channels as node features.

sampling and search for a good SS architecture. Next, TransNAS-Bench-101 (Duan et al. 2021) train architectures from separate micro and macro search spaces on various tasks for several hours and demonstrate the performance of multiple search algorithms. In contrast, we train architectures using the hyperparameters of Detectron2 (Wu et al. 2019) and Zhou et al. (2017b). We develop a novel shared head approach to build a generalizable performance predictor that learns on data from multiple CV tasks and predicts the performance of other downstream tasks.

Methodology

We cast performance prediction as a supervised learning regression problem where each data sample belongs to a given task domain t. Each task t contains instances (A_i^t, y_i^t) where A_i is an architecture, and y_i is its performance label denoted by a metric value on t. For example, y_i^t is the *accuracy* value for an Image Classification (IC) architecture, or *average precision* for Object Detection (OD). The goal is a neural predictor that can generalize across many tasks and provide accurate performance estimations. In this section, we describe how AIO-P represents architectures as Computational Graphs and uses K-Adapters to learn task transferable knowledge. Furthermore, we describe how shared task heads and pseudo-labeling let us form a large dataset of training instances. Finally, we discuss label scaling techniques for making accurate predictions across different performance distributions.

Network Representation

A CV architecture usually consists of a 'body' that performs feature extraction on an input and a 'head' that uses extracted features to make predictions. The 'body' structure comes from a given search space while the head comes from a specified task t.

The search spaces we consider are from OFA. These include ProxylessNAS (PN) (Cai, Zhu, and Han 2019), MobileNetV3 (MBv3) (Howard et al. 2019) and ResNet-50 (R50) (He et al. 2016a). Architectures bodies from these spaces are pre-trained on ImageNet classification. Head designs vary with task complexity. For example, a typical IC head uses global average pooling and MLP layers to predict class labels for an entire image. By contrast, we consider tasks like HPE, OD, and segmentation, which upsample feature maps with different resolutions to make predictions. Specifically, HPE (Zheng et al. 2020) generates large

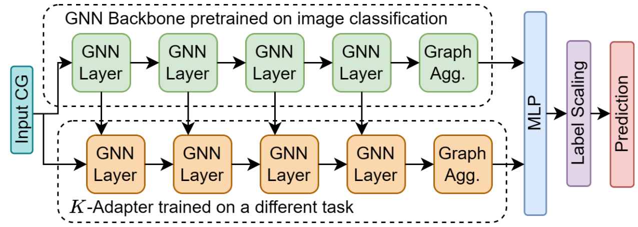

Figure 3: CG K-Adapter Diagram. We start with a graph encoder pre-trained on NAS-Bench-101 for IC and further extend the design with an adapter. The original encoder is frozen while we train the K-Adapter on a new search space and task, e.g., R50 on OD.

heat maps to estimate joint locations, while Semantic Segmentation (SS) (Liu et al. 2019) upsamples to predict class labels for every pixel in an image. OD (Ren et al. 2015) uses a Region Proposal Network (RPN) and Region of Interest Poolers (RoI) to estimate bounding box coordinates and classes, while Instance Segmentation (IS) replaces the bounding boxes with pixel-level masks for each instance of a class. Finally, Panoptic Segmentation (PS) (Kirillov et al. 2019) combines IS and SS to generate pixel masks for an image while differentiating individual instances of each class, e.g., masks for every person in a crowd will have different colors. We illustrate this breakdown of how search space bodies pair and interact with task heads in Figure 1 and provide finer details for each task head in the supplementary materials.

While prior neural predictors like White, Neiswanger, and Savani (2021) as well as Wen et al. (2020) adopt a customized encoding limited to predefined search spaces, e.g., NAS-Benchmarks on IC, we use a general encoding that can represent any neural network. Specifically, we derive Computational Graphs (CG) using the underlying graph structure that libraries like TensorFlow (Abadi et al. 2016) generate in the forward pass to execute backpropagation.

CGs are fine-grained graph structures where nodes denote individual atomic operations such as convolutions, linear layers, pooling, padding, addition, concatenation, etc. Node features describe properties like kernel sizes, strides, and channels. Featureless, directed edges denote the flow of latent data across the network, allowing for a task-transferable representation that incorporates the body and task head. Figure 2 illustrates how CGs represent one module of a MobileNetV3 (MBv3) body.

Finally, we note that our CG format can be extended to other network types, like those which perform Natural Language Processing tasks such as recurrent or attention-based models or even simple MLP models. However, such experiments are beyond the scope of this paper.

K-Adapters for Knowledge Infusion

AIO-P, as shown in Figure 3, starts with a GNN (Morris et al. 2019) regressor backbone. The stem consists of an encoding layer that transforms discrete node features, e.g., operation categories, into a continuous format. The GNN en-

	PN	MBv3	R50
#Architectures	215	217	215
Ground-Truth PCK	65.16%	65.22%	65.64%
Body Swap PCK	52.67%	43.62%	49.61%
SRCC	0.574	0.443	0.246
Sampling PCK	59.57%	61.58%	59.46%
SRCC	**0.659**	**0.576**	**0.375**

Table 1: Shared head performance distributions and SRCC on HPE PCK [%]. We compare performance obtained using body swapping and latent sampling to the ground-truth PCK from individually training architectures.

coder takes the computational graph as input, learns node embeddings using the adjacency matrix, and generates an overall graph embedding by aggregating node embeddings. The graph embedding is fed into an MLP, which outputs a performance prediction. In all our experiments, we pre-train the GNN regressor to predict classification accuracy on 50k NAS-Bench-101 CGs.

To predict performance on another downstream CV task, we extend this base GNN regressor with K-Adapters to infuse external knowledge from other tasks and datasets beyond image classification. To do this, we discard the original regressor MLP and freeze all remaining weights. We append the K-Adapter by pairing each existing GNN layer in the encoder with a new GNN "Adapter" layer. Each adapter layer accepts concatenated input from the previous adapter layer and the adjacent GNN layer. Formally, given the intermediate node embeddings for a graph x, we define the forward function to K-Adapter layer $GNN^{k,i}$ as

$$h_x^{k,i} = GNN^{k,i}(Concat[h_x^{k,i-1}, h_x^{b,i}]), \tag{1}$$

where $h_x^{k,i-1}$ and $h_x^{b,i}$ are the intermediate node embeddings produced by the previous K-Adapter layer and adjacent GNN layer in the backbone, respectively. Like the backbone, the K-Adapter produces an overall graph embedding. We concatenate both graph embeddings and feed them into a new MLP predictor.

Note that we can augment the original backbone with multiple K-Adapters, where each K-Adapter infuses the knowledge from a different task t. Hence, the overall predictor can generalize to different downstream tasks with different task head CG structures and labels.

Training K-Adapters Based on Latent Sampling

To train K-Adapters, we need datasets composed of architecture samples and their performance labels on tasks other than IC. We can obtain task-specific labels by sampling a body from a search space, e.g., OFA, coupling it with a task head, and training it. However, this is costly. Rather than having AIO-P learn on ground-truth labels obtained from individually trained architectures, we propose a pseudo-labeling method to efficiently obtain task-specific labels to train adapters, via a specially-trained task head that is *shared* amongst all network bodies in a search space.

Task	PN	MBv3	R50
HPE-LSP	215/2580	217/2862	215/2986
HPE-MPII	246/-	236/-	236/-
OD/IS/SS/PS	118/1633	118/1349	115/1399

Table 2: AIO-P K-Adapter dataset size, across each search space and task. '/' denotes architectures that we individually train (left) and those we label using a shared head (right).

A typical approach for training a shared task head is 'body swapping', which involves iteratively sampling pre-trained architecture bodies from a search space, e.g., OFA, attaching them to the shared head and training the head on a few batches of images. However, note that during body swapping, for each mini-batch of images, the head only samples a single body. To encourage randomness, rather than sampling body networks, we propose an approach where we directly sample the latent representations of the mini-batch of images from a distribution as if they are generated by sampled bodies. Yet, this will further mimic random body sampling per image instead of per mini-batch.

Specifically, let x be an image, $\mathcal{S}$ be an architecture search space and $B \in \mathcal{S}$ be a body network. As OFA search spaces each contain roughly 10^{18} bodies, we take a subset of architectures $\mathcal{S}'$ and compute the mean $\vec{\mu}(x)$ and standard deviation $\vec{\sigma}(x)$ of their latent representations of image x, i.e.,

$$\vec{\mu}(x) = \mathbb{E}_{B \in \mathcal{S}'}[f_B(x)], \tag{2}$$

$$\vec{\sigma}^2(x) = \mathrm{Var}_{B \in \mathcal{S}'}[f_B(x)], \tag{3}$$

where f_B denotes the function given by network body B. We then sample a latent representation $\vec{z}(x) = \vec{\mu}(x) + \vec{\zeta} \circ \vec{\sigma}(x)$ where $\vec{\zeta}$ is a standard normal $\mathcal{N}(0,1)$ random vector and $\circ$ is element-wise multiplication. We denote this approach as 'latent sampling', where we train the shared task head by $\{(\vec{z}(x), y_x)\}$, using the sampled latent vector $\vec{z}(x)$ as input to the head and y_x as the ground-truth task label for x. We limit the size of $\mathcal{S}'$ and use a round robin strategy with binning to constantly swap new bodies into $\mathcal{S}'$. We provide procedural details on our round robin strategy, shared head hyperparameters and resource cost breakdown in the supplementary materials.

To assess the reliability of our pseudo-labelling approach, we form a ground-truth set by individually training several hundred architectures (with bodies sampled from three spaces) on the Leeds Sports Pose-Extended (LSP) dataset for Human Pose Estimation (HPE) and measure performance in terms of Percentage of Correct Keypoints (PCK). We obtain pseudo-labels by fine-tuning the same bodies when connected to shared heads trained using *body swapping* and *latent sampling*, respectively. We compare the PCK distributions and SRCC between the pseudo-labels and ground-truth labels. We do not use the pseudo-labeled architectures from this experiment to train AIO-P. They are only for comparison. Table 1 shows the results. We note that latent sampling achieves much better performance on average, relative to the ground truth, while body swapping lags by over 10%. Also, the SRCC we observe using the latent

	ProxylessNAS			MobileNetV3			ResNet-50		
Task	GNN	+Eqs. 4 & 5	AIO-P	GNN	+Eqs. 4 & 5	AIO-P	GNN	+Eqs. 4 & 5	AIO-P
LSP	27.27%	0.72%	**0.70%**	27.07%	0.74%	**0.52%**	21.47%	1.07%	**0.94%**
MPII	8.10%	**0.34%**	0.42%	8.91%	0.36%	**0.27%**	2.48%	1.11%	**1.03%**
OD	59.53%	1.15%	**0.63%**	59.56%	1.24%	**0.62%**	54.07%	0.88%	**0.60%**
IS	62.00%	0.93%	**0.52%**	61.76%	0.73%	**0.58%**	56.74%	0.64%	**0.48%**
SS	53.07%	0.71%	**0.50%**	53.40%	0.90%	**0.56%**	47.90%	0.52%	**0.44%**
PS	56.19%	0.76%	**0.50%**	56.18%	0.74%	**0.57%**	51.63%	0.52%	**0.45%**

Table 3: MAE [%] of AIO-P on three search spaces and six tasks in the zero-shot transfer setting (no fine-tuning), compared to GNN without and with rescaling by Eqs. 4 & 5. AIO-P adopts 2 K-Adapters, trained on LSP and OD. AIO-P uses Equation 5 and standardizes regression targets. Results averaged across 5 seeds.

	ProxylessNAS			MobileNetV3			ResNet-50		
Task	GNN	+Eqs. 4 & 5	AIO-P	GNN	+Eqs. 4 & 5	AIO-P	GNN	+Eqs. 4 & 5	AIO-P
LSP	0.593	0.561	**0.698**	0.259	0.418	**0.556**	-0.302	0.176	**0.261**
MPII	0.711	**0.767**	0.753	0.300	**0.764**	0.701	-0.315	0.446	**0.532**
OD	0.558	0.471	**0.781**	**0.645**	0.087	0.515	-0.489	0.645	**0.817**
IS	0.599	0.211	**0.831**	0.592	0.034	**0.602**	-0.493	0.495	**0.817**
SS	0.487	0.262	**0.735**	0.517	-0.367	**0.689**	-0.406	0.589	**0.660**
PS	0.562	0.119	**0.732**	**0.570**	-0.009	0.518	-0.455	0.599	**0.788**

Table 4: SRCC of AIO-P on three search spaces and six tasks. Same configurations as Table 3.

sampling approach is positive for all search spaces and more favorable than body swapping.

Label Scaling

In addition to having different heads, task performance distributions differ. PN architectures yield ~75% accuracy on ImageNet, and around 40% SS mIoU on COCO. Note that performance distributions differ amongst tasks, or even the same task between ground truth and pseudo-labels.

Therefore, we experiment with methods that scale the labels AIO-P learns from to add further task transferability. Specifically, we employ standardization,

$$\mathcal{Z}(y) = \frac{y - \mu}{\sigma}, \tag{4}$$

where μ and σ are the mean and standard deviation of the label y distribution, respectively. This approach fits a set of data into a normal distribution $\mathcal{N}(0, 1)$. Further, we incorporate scaling by FLOPs, or Floating Point Operations required to perform the forward pass of a network, as a divisor prior to standardization,

$$y_F = y \cdot (\texttt{Log}_{10}(F + 1) + 1)^{-1}, \tag{5}$$

where F are the FLOPs of the network with performance y, measured in GigaFLOPs (1e9), and the addition of 1 in the denominator assures it will be positive real number. In all cases, if we apply Equation 5 to labels, we then standardize them using Equation 4.

Results

In this Section, we describe our suite of tasks, experimental setup and results. We consider the computer vision tasks of 2D Human Pose Estimation (HPE), Object Detection (OD), Instance Segmentation (IS), Semantic Segmentation (SS) and Panoptic Segmentation (PS).

HPE predicts joint locations from an image. We measure 2D HPE performance using Percentage of Correct Keypoints (PCK) and consider two HPE datasets: MPII (Andriluka et al. 2014) and Leeds Sports Pose-Extended (LSP) (Johnson and Everingham 2011), which contain 22k and 11k images, respectively. We individually train networks for both datasets and train a shared head for LSP.

OD and IS measure performance in mean Average Precision (mAP), while SS uses mean Intersection over Union (mIoU). PS, a combination of IS and SS, uses Panoptic Quality (PQ), a balance of mAP and mIoU. These performance metrics are reported as percentages [%]. We consider the 2017 version of MS Common Objects in Context (COCO) (Lin et al. 2014) as our dataset for these tasks, as it contains 118k and 5k training and validation images, respectively. We use Detectron2 (Wu et al. 2019) to pair a given body with OD, IS SS, and PS to train on all four tasks simultaneously.

Table 2 enumerates of the number of architectures we have for each search space and task. We use pseudo-labeled architecture CGs to train AIO-P and reserve individually trained ones as held-out test samples. Additionally, we include a more extensive breakdown with performance and FLOPs distributions in the supplementary materials.

Training Procedure

Starting with the GNN backbone, we append two K-Adapter modules. We train these using the pseudo-labeled CGs for two task domains, OD and LSP-HPE. We separately apply Equation 5 and then standardize the labels of each K-

Space	Synflow	Jacov	Fisher	Grad Norm	Snip	FLOPs	AIO-P	AIO-P FT
PN-LSP	-0.004	-0.057	0.449	0.581	0.624	0.584	**0.698**	*0.668*
MBv3-LSP	**0.609**	0.029	0.129	0.426	0.466	0.562	0.556	*0.567*
R50-LSP	**0.639**	-0.071	0.515	0.581	*0.646*	0.263	0.261	0.264
PN-MPII	0.046	-0.008	0.538	0.733	*0.755*	0.735	0.753	**0.773**
MBv3-MPII	*0.736*	-0.014	0.203	0.679	0.691	*0.736*	0.701	**0.744**
R50-MPII	**0.865**	0.111	0.709	0.732	*0.849*	0.532	0.532	0.532
PN-SS	0.022	-0.023	0.050	0.141	-0.082	0.608	*0.735*	**0.849**
MBv3-SS	-0.309	0.042	0.022	0.040	0.188	0.445	*0.689*	**0.822**
R50-SS	-0.255	0.141	0.126	0.354	0.036	*0.661*	0.660	**0.677**

Table 5: SRCC between AIO-P, ZCP methods and a FLOPs-based predictor on three tasks. We also include 'AIO-P FT', i.e., fine-tuning on 20 held-out standardization samples. We bold the best result and italicize the second best.

Adapter task before training and freeze the weights of the GNN backbone.

To evaluate individually trained test set CGs for a task, we consider the zero-shot transfer context where we provide no data on a target task prior to inference. Similar to Lu et al. (2021a), we sample 20 random architectures from the test set to compute standardization parameters μ and σ and exclude these architectures from evaluation. We also consider a fine-tuning context where we use the same 20 random architectures to train AIO-P prior to inference. As we evaluate multiple methods using different random seeds, we ensure the set of 20 sampled architectures is the same for every seed value. Also, note that the set of body architectures we individually train on any task are disjoint from the ones we pseudo-label. We provide granular predictor details and training hyperparameters in the supplementary materials.

Zero-Shot Transfer Performance

We consider two predictor evaluation metrics, Mean Absolute Error (MAE) and Spearman's Rank Correlation Coefficient (SRCC). The former gauges a predictor's ability to make accurate estimations on individual labels, while the latter determines a predictor's capacity to correctly rank a population by performance. We report MAE as a percentage for each task metric, e.g., PCK for HPE and mIoU for SS, where lower is better. SRCC falls in $[-1, 1]$ and indicates agreement with ground-truth ordering, so higher is better.

Tables 3 and 4 list our results for AIO-P in terms of MAE and SRCC, respectively. AIO-P achieves the best MAE performance in the majority of scenarios, including every task on MBv3 and R50. The sole exception is a GNN using standardization and FLOPs scaling for MPII by less than 0.1% MAE, while the unmodified GNN fails to generalize to different task performance distributions.

For SRCC, AIO-P consistently achieves the best correlation on all R50 tasks and in at least half for PN and MBv3. While both variants of the GNN obtain the high SRCC on at least one search space and task pair, they are overall very inconsistent as the GNN with re-scaled labels achieves negative SRCC on MBv3 and the unmodified GNN fails on R50.

Next, Table 5 compares the ranking performance of AIO-P with Zero-Cost Proxies (ZCP) and FLOPs. Without fine-tuning, AIO-P achieves high correlation performance for all search spaces on MPII and SS, as well as PN and MBv3 for LSP, demonstrating high generalizability. The most competitive ZCP method is Synflow (Tanaka et al. 2020), but only for HPE tasks with MBv3 and R50 as it fails on PN and in the SS context. Grad Norm achieves positive SRCC across all search spaces and tasks but never the best performance. FLOPs are also highly correlated with performance in all settings, except LSP for R50. Unlike ZCP, the advantage of AIO-P in this context is its ability undergo fine-tuning to boost performance, as shown for the MPII and SS tasks. Next, we compare Eqs. 4 and 5 with another fine-tuning-based method for intertask prediction.

Task	GNN	AIO-P +AdaProxy	AIO-P w/o Eq. 5	AIO-P
MPII	0.33%	0.35%	*0.29%*	**0.25**%
Inst. Seg.	0.66%	*0.47%*	**0.39**%	0.58%
Pan. Seg.	0.64%	*0.51%*	**0.46**%	0.61%

Table 6: MAE of AIO-P on MPII, IS and PS for the MBv3 search space (with fine-tuning). A single K-Adapter was trained on SS for AIO-P variants. We bold the best result and italicize the second best.

Task	GNN	AIO-P +AdaProxy	AIO-P w/o Eq. 5	AIO-P
MPII	*0.680*	0.574	0.650	**0.750**
Inst. Seg.	0.547	*0.677*	**0.747**	0.585
Pan. Seg.	0.568	*0.627*	**0.685**	0.512

Table 7: SRCC of AIO-P on MPII, IS and PS for the MBv3 search space (with fine-tuning). Same experimental setup as Table 6.

Fine-Tuning Results

We now evaluate the transferability of AIO-P on downstream tasks if predictor fine-tuning is allowed based on a small number, i.e., 20, downstream architectures. As a comparison, we also evaluate AIO-P using a weight scaling technique proposed in AdaProxy (Lu et al. 2021a), which scales the final MLP layer of a predictor by minimizing the follow-

Task	OD	SS	LSP	OD+SS	OD+LSP
MPII	0.60%	0.39%	**0.28%**	*0.35%*	0.42%
IS	0.70%	*0.56%*	1.03%	0.61%	**0.52%**
PS	*0.75%*	0.82%	1.03%	0.81%	**0.50%**
MPII FT	**0.25%**	*0.26%*	0.27%	0.27%	*0.26%*
IS FT	0.49%	0.50%	**0.27%**	0.50%	*0.33%*
PS FT	0.50%	0.54%	**0.31%**	0.52%	*0.33%*

Table 8: MAE of different K-Adapter tasks on the PN search space. ‘FT’ indicates fine-tuning on 20 held-out target architectures. We bold the best result and italicize the second best.

Task	OD	SS	LSP	OD+SS	OD+LSP
MPII	**0.762**	0.660	0.717	0.727	*0.753*
IS	0.749	0.728	**0.929**	0.750	*0.831*
PS	0.650	0.669	**0.859**	0.672	*0.732*
MPII FT	**0.793**	*0.790*	0.764	0.779	0.773
IS FT	0.749	0.757	**0.915**	0.730	*0.894*
PS FT	0.670	0.671	**0.880**	0.647	*0.858*

Table 9: SRCC of different K-Adapter tasks on the PN search space. Same experimental setup as Table 8.

ing loss on fine-tuning samples:

$$\min_{\alpha,\vec{b}} |[(\alpha\vec{I}^T + \vec{b}^T) \circ \vec{w}^T]\vec{x} - y|^2 + \lambda|\vec{b}|,$$

where α is a scalar, $\vec{I}$ is an identity vector, $\vec{w}$ are the weights in the final layer, $\vec{b}$ is a sparsity vector and λ is a regularizer weight. During this minimization, all of the original predictor weights are frozen. For this experiment, we consider the MBv3 search space, train a K-Adapter on SS and evaluate on MPII, IS and PS.

Tables 6 and 7 list our results in terms of MAE and SRCC, respectively. We are able to obtain more accurate predictions in terms of MAE and SRCC using some form of standardization rather than AdaProxy. On MPII specifically, Eq. 5 overcomes the limitations of low inter-task correlation to produce the best MAE and SRCC, while just using standardization is enough to obtain the most accurate IS and PS predictions. Another advantage of our standardization and FLOPs-based transform approach is the absence of tunable hyperparameters, e.g., λ.

Ablation Study

We ablate the effect of different K-Adapter training tasks using PN as an example search space. Tables 8 and 9 show our MAE and SRCC findings, respectively. We see that using a double K-Adapter on OD and LSP helps to generalize MAE and SRCC performance across the downstream tasks. While the best results typically use just OD or LSP, LSP struggles to produce low MAE on IS and PS without fine-tuning. However, we overcome this hurdle when adding another K-Adapter for OD. First, we note that the best results use either OD or LSP, although SS still produces good results. Particularly, LSP struggles to produce low MAE on IS and PS without fine-tuning. Introducing another K-Adapter for OD overcomes this hurdle.

Task	GNN	+K-Adapter	+ Eq. 4	AIO-P
MPII	2.48%	55.30%	**0.39%**	*2.38%*
IS	56.74%	*2.38%*	**0.73%**	**0.73%**
PS	51.63%	7.39%	*0.89%*	**0.61%**
MPII FT	0.28%	*0.77%*	**0.15%**	1.09%
IS FT	0.61%	0.85%	**0.36%**	*0.45%*
PS FT	0.71%	0.61%	**0.40%**	*0.44%*

Table 10: MAE performance comparing the effect of K-Adapters as well as Eqs. 4 and 5 in the zero-shot and fine-tuning (FT) contexts. We consider the R50 search space and train a K-Adapter on SS. We bold the best result and italicize the second best.

Task	GNN	+K-Adapter	+ Eq. 4	AIO-P
MPII	-0.315	**0.708**	*0.418*	0.337
IS	-0.493	**0.669**	*0.361*	0.324
PS	-0.455	**0.633**	*0.399*	0.291
MPII FT	*0.700*	**0.738**	0.512	0.532
IS FT	0.611	0.687	*0.727*	**0.840**
PS FT	0.601	0.667	*0.762*	**0.811**

Table 11: SRCC performance comparing the effect of K-Adapters. Same experimental setup as Table 10.

Additionally, we compare the GNN backbone to using a single K-Adapter, using standardization (Eq. 4), and then our FLOPs scaling (Eq. 5). Tables 10 and 11 list our results for MAE and SRCC, respectively. We see that AIO-P with or without Eq. 5 can obtain top MAE performance in most scenarios in the zero-shot and fine-tuning contexts. However, that is not the case for SRCC, where Eq. 5 allows AIO-P to achieve correlation metrics above 0.8 with fine-tuning. While the normal GNN is not very competitive, adding a single K-Adapter without standardization can improve ranking performance significantly. However, the prediction error is still high due to differences in task performance distributions (reported in the Supplementary Materials). Overall, standardization and Eq. 5 allow AIO-P to strike a balance between prediction error and ranking correlation.

Transfer to Foreign Network Types

To further test the transferability of AIO-P to new architecture types not seen in training, we perform inference on several foreign ‘model zoos’. Each model zoo contains 10 or fewer architecture variants, e.g., Inception (Szegedy et al. 2017), EfficientNets (Tan and Le 2019), MobileNets and ResNets. Specifically, we include model zoos from the DeepLab repository (Chen et al. 2018) which focus on for semantic segmentation (SS) performance on multiple datasets. We also evaluate AIO-P on several image classification (IC) model zoos from TensorFlow-Slim (Guadarrama and Silberman 2016).

Model Zoo	AIO-P w/o Eq. 5	AIO-P
DeepLab-ADE20k	0.127 $\pm$ 0.255	**0.991** $\pm$ 0.016
DeepLab-Pascal	0.392 $\pm$ 0.088	**0.939** $\pm$ 0.035
DeepLab-Cityscapes	0.572 $\pm$ 0.031	**0.925** $\pm$ 0.024
Slim-ResNets	-0.577 $\pm$ 0.183	**0.920** $\pm$ 0.106
Slim-Inception	-0.700 $\pm$ 0.316	**0.980** $\pm$ 0.040
Slim-MobileNets	-0.500 $\pm$ 0.000	**0.400** $\pm$ 0.535
Slim-EfficientNets	**1.000** $\pm$ 0.000	**1.000** $\pm$ 0.000

Table 12: SRCC of AIO-P on several 'model zoos'. Double horizontal line demarcates SS models from IC models.

DeepLab Semantic Segmentation We consider architectures on three different SS datasets: ADE20k (Zhou et al. 2017a), Pascal VOC (Everingham et al. 2015), and Cityscapes (Cordts et al. 2016). These are different SS datasets than the MS-COCO we use for training OFA-based architectures, so the results further demonstrate the generalizability of AIO-P across datasets, even for the same task. Specifically, we consider the following architectures per dataset:

1. **ADE20k** (5): MobileNetV2, Xception65 (Chollet 2017) as well as Auto-DeepLab-{S, M, L} (Liu et al. 2019).
2. **Pascal VOC** (6): MobileNetV2, MobileNetV2 w/ reduced depth, Xception65 and Auto-DeepLab-{S, M, L}.
3. **Cityscapes** (8): MobileNet{V2, V3-Small, V3-Large}, Xception65, Xception71 and Auto-DeepLab-{S, M, L}.

TensorFlow-Slim Image Classification We also consider several architectures that perform IC on ImageNet:

1. **ResNets** (6): ResNet-v1-{50, 101, 152} (He et al. 2016a) and ResNet-v2-{50, 101, 152} (He et al. 2016b).
2. **Inception** (5): Inception-{v1, v2, v3, v4} and Inception-ResNet-v2 (Szegedy et al. 2017)
3. **MobileNets** (6): MobileNet{V1, V1-0.5, V1-0.25, V2, V2-1.4} where '-X.Y' is a channel multiplier.
4. **EfficientNets** (Tan and Le 2019) (8): EfficientNet-{B0, B1, B2, B3, B4, B5, B6, B7}.

Using AIO-P pre-trained on ResNet-50 bodies and with a double K-Adapter trained on OD and LSP tasks, and then fine-tuned on SS (the task DeepLab performs), we investigate whether AIO-P can adequately rank the architectures in DeepLab and in TensorFlow-Slim, which contain new types of body networks other than ResNet-50.

As shown in Table 12, we note that AIO-P can achieve positive correlation inference on all SS and IC model zoos. In particular, we obtain perfect SRCC on EfficientNets. Moreover, this performance is superior to AIO-P when Eq. 5 is not present. This is because while standardization improves MAE performance, it does not affect architecture rankings, whereas FLOPs-based transformation does. These findings demonstrate the efficacy of AIO-P and Eq. 5 in ranking the performance of foreign networks with different connections/topologies.

	Full	Simple	Lighted	Dark	FLOPs
Base Pr	**96.3**%	**98.7**%	**97.9**%	96.5%	563M
AIO-P Pr	96.1%	**98.7**%	**97.9**%	**96.7**%	486M
Base Rc	**91.9**%	**98.3**%	**96.8**%	92.6%	563M
AIO-P Rc	91.1%	98.2%	96.6%	**93.2**%	486M

Table 13: Precision (Pr) and Recall (Rc) of a proprietary FR network found by pairing AIO-P with a search algorithm designed to preserve performance while reducing FLOPs.

Application to NAS

Finally, we apply AIO-P to NAS. Specifically, we use a predictor that achieves high SRCC on the bounding box task OD on R50, to optimize a proprietary neural network designed to perform Facial Recognition (FR) on mobile devices. We pair AIO-P with a mutation-based search algorithm that aims to preserve performance while reducing FLOPs. Although the architecture we optimize does not belong to any of the OFA search spaces we consider, like the aforementioned model zoos, AIO-P can estimate performance using the CG framework. Additionally, our mutation algorithm proposes edits to CGs that vary from swapping subgraphs of operation sequences to manually pruning the number of channels in a convolution node.

Table 13 demonstrates that we can maintain performance in most settings while reducing FLOPs by over 13.5%. In the 'dark' setting, where features can be hard to see, our model improves precision and recall by 0.2% and 0.6%, respectively. At most, we only lose 0.2% precision and 0.8% recall across other settings. Therefore, these findings demonstrate the efficacy of AIO-P in the NAS setting for any general neural network, not simply for well-known search spaces.

Conclusion

We propose AIO-P, or All-in-One Predictors, to broaden the scope of neural performance prediction tasks. AIO-P uses K-Adapters to infuse knowledge from different tasks and accepts Computational Graphs (CG) as input. CGs represent the body and head of an architecture by encoding all atomic operations as nodes with directed edges determined by the network forward pass. At the output, AIO-P incorporates target scaling techniques, including one based on FLOPs, to re-scale predictions into the appropriate task metric and ultimately obtain superior performance. To construct a suitable training set, we devise a shared head approach with latent sampling, which can pair with any architecture in the search space to produce a pseudo-label that is highly correlated with the true label. Experimental results show that AIO-P can obtain Mean Absolute Error and Spearman's Rank Correlations below 1% and above 0.5, respectively, when transferred to a wide range of downstream tasks. Moreover, AIO-P can directly transfer and adequately rank different networks in several foreign model zoos not seen in training for classification and semantic segmentation. Finally, we use AIO-P to optimize a proprietary facial recognition network to effectively preserve precision and recall while reducing FLOPs by over 13.5%.

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *OSDI*, 2016, 265–283. Savannah, GA, USA.

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

Andriluka, M.; Pishchulin, L.; Gehler, P.; and Schiele, B. 2014. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; and Han, S. 2020. Once for All: Train One Network and Specialize it for Efficient Deployment. In *International Conference on Learning Representations*.

Cai, H.; Zhu, L.; and Han, S. 2019. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *International Conference on Learning Representations*.

Changiz Rezaei, S. S.; Han, F. X.; Niu, D.; Salameh, M.; Mills, K.; Lian, S.; Lu, W.; and Jui, S. 2021. Generative Adversarial Neural Architecture Search. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2227–2234. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Chen, L.-C.; Zhu, Y.; Papandreou, G.; Schroff, F.; and Adam, H. 2018. Encoder-Decoder With Atrous Separable Convolution For Semantic Image Segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 801–818.

Chollet, F. 2017. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1251–1258.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ding, M.; Huo, Y.; Lu, H.; Yang, L.; Wang, Z.; Lu, Z.; Wang, J.; and Luo, P. 2022. Learning Versatile Neural Architectures by Propagating Network Codes. In *International Conference on Learning Representations*.

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

Duan, Y.; Chen, X.; Xu, H.; Chen, Z.; Liang, X.; Zhang, T.; and Li, Z. 2021. TransNAS-Bench-101: Improving Transferability and Generalizability of Cross-Task Neural Architecture Search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5251–5260.

Elsken, T.; Metzen, J. H.; Hutter, F.; et al. 2019. Neural Architecture Search: A Survey. *J. Mach. Learn. Res.*, 20(55): 1–21.

Everingham, M.; Eslami, S. M. A.; Van Gool, L.; Williams, C. K. I.; Winn, J.; and Zisserman, A. 2015. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1): 98–136.

Guadarrama, S.; and Silberman, N. 2016. TensorFlow-Slim: A lightweight library for defining, training and evaluating complex models in TensorFlow. https://github.com/google-research/tf-slim. Accessed: 2022-08-15.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity Mappings in Deep Residual Networks. In *European Conference on Computer Vision*, 630–645. Springer.

Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. 2019. Searching For MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1314–1324.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-Excitation Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7132–7141.

Johnson, S.; and Everingham, M. 2011. Learning Effective Human Pose Estimation from Inaccurate Annotation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1465–1472. IEEE.

Kirillov, A.; Girshick, R.; He, K.; and Dollár, P. 2019. Panoptic Feature Pyramid Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6399–6408.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning Multiple Layers of Features From Tiny Images. *Technical Report*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C. L.; and Dollár, P. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, 740–755. Cham: Springer International Publishing.

Liu, C.; Chen, L.-C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A. L.; and Fei-Fei, L. 2019. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 82–92.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations (ICLR)*.

Lu, B.; Yang, J.; Jiang, W.; Shi, Y.; and Ren, S. 2021a. One Proxy Device is Enough for Hardware-Aware Neural Architecture Search. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5(3): 1–34.

Lu, S.; Li, J.; Tan, J.; Yang, S.; and Liu, J. 2021b. TNASP: A Transformer-based NAS Predictor with a Self-Evolution Framework. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 15125–15137. Curran Associates, Inc.

Luo, R.; Tan, X.; Wang, R.; Qin, T.; Chen, E.; and Liu, T.-Y. 2020. Semi-Supervised Neural Architecture Search. *Advances in Neural Information Processing Systems*, 33: 10547–10557.

Mills, K. G.; Han, F. X.; Salameh, M.; Changiz Rezaei, S. S.; Kong, L.; Lu, W.; Lian, S.; Jui, S.; and Niu, D. 2021a. L^2NAS: Learning to Optimize Neural Architectures via Continuous-Action Reinforcement Learning. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 1284–1293.

Mills, K. G.; Han, F. X.; Zhang, J.; Changiz Rezaei, S. S.; Chudak, F.; Lu, W.; Lian, S.; Jui, S.; and Niu, D. 2021b. Profiling Neural Blocks and Design Spaces for Mobile Neural Architecture Search. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 4026–4035.

Mills, K. G.; Salameh, M.; Niu, D.; Han, F. X.; Rezaei, S. S. C.; Yao, H.; Lu, W.; Lian, S.; and Jui, S. 2021c. Exploring Neural Architecture Search Space via Deep Deterministic Sampling. *IEEE Access*, 9: 110962–110974.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4602–4609.

Munoz, J. P.; Lyalyushkin, N.; Lacewell, C. W.; Senina, A.; Cummings, D.; Sarah, A.; Kozlov, A.; and Jain, N. 2022. Automated Super-Network Generation for Scalable Neural Architecture Search. In *International Conference on Automated Machine Learning*, 5–1. PMLR.

Pham, H.; Guan, M.; Zoph, B.; Le, Q.; and Dean, J. 2018. Efficient Neural Architecture Search via Parameters Sharing. In *International Conference on Machine Learning*, 4095–4104. PMLR.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*, 28.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3): 211–252.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. A. 2017. Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Thirty-first AAAI Conference on Artificial Intelligence*.

Tan, M.; and Le, Q. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, 6105–6114. PMLR.

Tanaka, H.; Kunin, D.; Yamins, D. L.; and Ganguli, S. 2020. Pruning Neural Networks Without Any Data by Iteratively Conserving Synaptic Flow. *Advances in Neural Information Processing Systems*, 33: 6377–6389.

Wang, R.; Tang, D.; Duan, N.; Wei, Z.; Huang, X.; Ji, J.; Cao, G.; Jiang, D.; and Zhou, M. 2021. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 1405–1418. Online: Association for Computational Linguistics.

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.

Wu, J.; Dai, X.; Chen, D.; Chen, Y.; Liu, M.; Yu, Y.; Wang, Z.; Liu, Z.; Chen, M.; and Yuan, L. 2021. Stronger NAS with Weaker Predictors. *Advances in Neural Information Processing Systems*, 34: 28904–28918.

Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.-Y.; and Girshick, R. 2019. Detectron2. https://github.com/facebookresearch/detectron2. Accessed: 2022-08-15.

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*.

Zheng, C.; Wu, W.; Chen, C.; Yang, T.; Zhu, S.; Shen, J.; Kehtarnavaz, N.; and Shah, M. 2020. Deep Learning-Based Human Pose Estimation: A Survey. *arXiv preprint arXiv:2012.13392*.

Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; and Torralba, A. 2017a. Scene Parsing Through ADE20K Dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5122–5130.

Zhou, X.; Huang, Q.; Sun, X.; Xue, X.; and Wei, Y. 2017b. Towards 3D Human Pose Estimation in the Wild: A Weakly-Supervised Approach. In *The IEEE International Conference on Computer Vision (ICCV)*.

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