SuperNet in Neural Architecture Search: A Taxonomic Survey

Stephen Cha¹, Taehyeon Kim², Hayeon Lee³ and Se-Young Yun⁴
¹²⁴OSI Lab, KAIST
³⁴MLAI Lab, KAIST
stephencha@pitt.edu, {potter32, hayeon926, yunseyoung}@kaist.ac.kr,

Abstract

Deep Neural Networks (DNN) have made significant progress in a wide range of visual recognition tasks such as image classification, object detection, and semantic segmentation. The evolution of convolutional architectures has led to better performance by incurring expensive computational costs. In addition, network design has become a difficult task, which is labor-intensive and requires a high level of domain knowledge. To mitigate such issues, there have been studies for a variety of neural architecture search methods that automatically search for optimal architectures, achieving models with impressive performance that outperform human-designed counterparts. This survey aims to provide an overview of existing works in this field of research and specifically focus on the supernet optimization that builds a neural network that assembles all the architectures as its sub models by using weight sharing. We aim to accomplish that by categorizing supernet optimization by proposing them as solutions to the common challenges found in the literature: data-side optimization, poor rank correlation alleviation, and transferable NAS for a number of deployment scenarios.

1 Introduction

Designing neural architectures has been essential in a variety of visual recognition tasks. In general, such architectures are being hand-engineered using expert knowledge and heuristics [Krizhevsky et al., 2012; He et al., 2016; Howard et al., 2017] due to vanishing/exploding gradient issues. ResNet [He et al., 2016] utilizes skip connections where outputs of a layer jump over to a deeper layer to alleviate such issues while there still remain labor-intensive jobs that require expert knowledge to deploy them into real-world problems. Neural Architecture Search (NAS) lessens such issue by automating the process of finding such architectures. During search time, the optimal architecture is identified with certain evaluation criterion.

With the rapid growth of interest in NAS, there has been a lot of recent literature based on Reinforcement Learning (RL) [Zoph and Le, 2016; Baker et al., 2016], Evolutionary Search (ES) [Real et al., 2017; Liu et al., 2017; Elsken et al., 2018] and Bayesian Optimization (BO) [Shi et al., 2019; Eriksson et al., 2021; Liu et al., 2021] despite their expensive time costs for searching processes. Early RL and ES studies can require repetitive trial-and-error that results in over thousands of GPU days for search time because both approaches train numerous architectures by directly sampling the candidates from a large search space. As a result, this makes such NAS methods infeasible for most users.

The supernet approach lightens such computational burden [Pham et al., 2018]. A supernet is a network that contains all possible architectures for the search process. As the name suggests, supernet-based methods first train a supernet which embodies the entire search space. The method then takes a subnet of the supernet as the sampled architecture. Unlike earlier approaches that requires repetitive training, supernet-based NAS eliminates that requirement. This approach reduces inference time from thousands of GPU days by orders of magnitude.

This paper categorizes supernet optimization with three common problems found throughout the literature (Figure 1):

• Data-Side Issue. A common avenue to optimize a machine learning algorithm is to address its data-related...
Figure 1: A visualization of problems found in supernet optimization literature. On the upper left, we provide a diagram of a supernet cell where the circles represent feature maps and rectangles denote candidate operations (e.g., convolution, pooling, skip connection). In the upper right, we illustrate how a supernet can be constructed by connecting repeating cell modules. In the lower half, we illustrate the three challenges we present in this survey. We discuss multi-resolution, class noise, and out-of-distribution for data-side issues. Multi-resolution refers to the case where the images are of various sizes. Class noise refers to the presence of mislabelled data in the supervised case. Out-of-distribution refers to the mismatch of training data between the training and live stage. Poor rank correlation is when sub-networks sampled from a supernet perform significantly worse than standalone accuracy. The line labelled as ideal refers to perfect correlation. I.e., the predicted accuracy coincides with the standalone accuracy. For transferability, specialized networks that require no extra training can be sampled from a large network and be directly deployed to specific hardware settings.

issues. In image classification, the resolution size is of concern. Most supernet optimization methods are trained to perform on ImageNet for $224 \times 224$ pixels [Zhong et al., 2018; Guo et al., 2020; Liu et al., 2018; Xie et al., 2018; Xu et al., 2019; Chu et al., 2021; Cai et al., 2019; Yu et al., 2020; Wang et al., 2021b; Zhang et al., 2021]. Given the prevalence of high resolution images in real life settings, research on supernet optimization methods that can handle such images are of interest. Another issue is data labelling. Supervised learning requires ground truth labels which are laborious to provide. In addition, most machine learning algorithms are built under the i.i.d. assumption (independent identically distributed). Experiments in the deep learning literature usually deal with clean data using CIFAR10/100 and ImageNet. However, in real life settings, there can be noise in the input data and labels. This can hinder model performance for supernet optimization. Such cases are referred to as Out-of-Distribution (OoD). Thus, there is a need for developing NAS methods that are robust to OoD data.

• Poor Rank Correlation Issue. When a supernet-based NAS method yields an architecture, it evaluates the sampled network with a proxy performance metric. It repeats this process and outputs a list of architectures ranked by their predicted performance. However, the estimated accuracies have low correlation with the standalone accuracies. Many works address this issue [Bender et al., 2018; Guo et al., 2020; Chu et al., 2021; Li et al., 2021; Zhao et al., 2021; Peng et al., 2021].

• Transferability. Transferability for a number of deployment scenarios across different hardware settings is of interest for applications of NAS. Despite its increasing demand, many works do not take hardware setting transferability into consideration during the search process [Zoph and Le, 2016; Real et al., 2017; Liu et al., 2018; Bender et al., 2018]. Supernets are no exception to high compute cost because they require retraining after sampling a sub network [Liu et al., 2018; Bender et al., 2018]. To accommodate for all different hardware settings, users would have to repeat the cost-heavy process repeatedly for each setting.

Organization. This survey is structured as follows. In Section 2, we give preliminaries to provide necessary background
information for supernet optimization. In Section 3, we go over the optimization strategies that tackle the aforementioned three issues in supernet optimization. In Section 4, we compare the performances of different architectures found by human experts and NAS methods. In Section 5, we provide insights on potential future directions for supernet-based NAS for image classification. And lastly, we conclude our paper in Section 6.

2 Preliminaries

2.1 Neural Architecture Search (NAS)

Neural Architecture Search (NAS) automatically finds an architecture for a deep learning task while satisfying accuracy constraints. The NAS process can be broken down into three parts: the search space, the search strategy, and the performance estimation strategy [Elsken et al., 2019].

- **Search space.** The search space embodies the possible architectures for the given task. Introducing inductive bias when defining the search space is inevitable. However, works involving meta-learning [Lee et al., 2021] constructs an embedded search space that circumvents such human bias. The earliest example of a search space would be a sequential layer of operations.

- **Search strategy.** The search strategy determines how the architectures are sampled from the search space. There are two competing metrics here: architecture performance and search time. Finding high performing architectures may take a long time. A potential navigation method for this would be probing the Pareto frontier; the set of solutions where improving one objective will worsen another objective. Reinforcement learning [Zoph and Le, 2016] and evolution [Real et al., 2017] are other examples of search strategies.

- **Performance estimation strategy.** The performance estimation strategy seeks to measure how well the searched architecture does on new unseen data. The naive way to do this is to predict on the validation set. Of course, full-training is computationally taxing and not the most practical way to do things. Works in the NAS literature have found many different proxies for performance estimation.

2.2 Supernet Optimization

We remark that the terms supernet and weight-sharing are at times used interchangeably and circularly in the NAS literature. For the purposes of this survey, we will distinguish and define those terms.

A **supernet** is a neural network that contains all possible architectures to be sampled, i.e., it serves as the search space. Architectures are found by taking subnets from the supernet. We can represent a supernet as a Directed Acyclic Graph (DAG) where the nodes are feature maps and the edges are operations (e.g., convolution, pool, zero). We refer to figure 2 for an illustration of a supernet.

A key feature of supernet-based NAS would be **weight-sharing** [Pham et al., 2018]. Weight-sharing is done by first training a single large supernet. An RL-based controller [Pham et al., 2018] then samples a subnet from the supernet. As a result, all sampled child networks share weights from the supernet. The introduction of weight-sharing has reduced GPU days for the search process from thousands of GPU days down to a few GPU days. [Zoph and Le, 2016; Real et al., 2017; Zoph et al., 2018]. The one-shot approach [Bender et al., 2018] extends the weight-sharing technique. The authors train a single large supernet which is just a neural network trained using SGD with Momentum. During training, a linearly scheduled dropout strategy [Hinton et al., 2012] is used to prevent co-adaptation among operations (edges). The dropout rate starts at 0 and then linearly increases to \( r^{1/k} \) where \( 0 < r < 1 \) is a hyperparameter and \( k \) is the number of incoming paths to a given operation. When applying dropout, a random subset of the operations are zeroed out. After training, subnets are randomly sampled from the trained supernet following a uniform distribution (other search methods such as evolution or reinforcement learning can be used instead). The sampled subnet is then evaluated on a held-out validation set. The resulting output is a list of candidate architectures ranked by one-shot accuracy (based on validation set). The best performing architectures can be selected to be retrained from scratch.

3 Optimization Strategies

In this section, we survey supernet optimization methods with respect to the following optimization strategies used to address the aforementioned issues.

- **Data-Side Optimization.** Corresponding to data-side issues, we survey supernet-based NAS methods that are tailored for high resolution images [Ding et al., 2021], randomly labeled data [Zhang et al., 2021], and Out-of-Distribution data [Bai et al., 2021].

- **Poor Rank Correlation Alleviation.** Corresponding to the poor rank correlation issue, alleviation can be achieved by either navigating or factorizing the search space. Navigation is the intuitive solution because taking a subnet from the supernet is essentially selecting a path in the supernet[Bender et al., 2018;
Factorization can be done by doing a hierarchical partitioning of the supernet [Zhao et al., 2021] or breaking down the search space into blocks [Li et al., 2020; Li et al., 2021]. Branching away from a search space oriented approach, a different work [Peng et al., 2021] attributes the low correlation to supernet training consistency shift and proposes “a nontrivial supernet-II model”.

- **Transferable NAS.** We survey works that require no additional training after sampling a subnet from its respective supernet [Cai et al., 2019; Yu et al., 2020]. That is, works that yield subnets that are instantly deployable on various hardware settings [Cai et al., 2019].

### 3.1 Data-Side Optimization

A lot of works in supernet optimization focus on improving the architecture or the search phase itself. Works such as OFA [Cai et al., 2019] and BigNAS [Yu et al., 2020] seek to yield architectures that can be instantly deployed for tasks (these are mentioned in section 3.3). Methods such as ENAS [Pham et al., 2018] and DARTS [Liu et al., 2018] focus on searching architectures efficiently (quickly). However, the actual architecture and search phase are not the only avenues for improving a machine learning algorithm. Data is another route for optimization.

The issue of data is essential to any machine learning task. In this section, we survey supernet-based NAS with respect to data-side optimization. Given that the task is image classification, the image resolution is of concern. We cover High Resolution NAS (HR-NAS) [Ding et al., 2021] which is designed for high resolution images. For supervised learning, labels are integral. However, labelling high volumes of data is laborious and can be a bottleneck in the machine learning pipeline.

We explore Random Label NAS (RLNAS) [Zhang et al., 2021] which removes ground truth labels and only uses random labels for the search. In addition, most NAS methods are fed input signals with clean data. There is a need for NAS methods that are robust against noisy data. We also look into NAS that is robust against Out-of-Distribution (OOD) data [Bai et al., 2021].

### High Resolution Images

Image resolution sizes in image classification tasks tend to be on the smaller side (about $224 \times 224$). Naturally, computational requirements increase as image resolution goes up. So a natural concern is to develop methods that handle high resolution images. High Resolution NAS (HR-NAS) searches for architectures designed to take in high resolution images.

HR-NAS introduces a light-weight Vision Transformer (ViT) [Dosovitskiy et al., 2020] that also incorporates convolution. The ViT consists of a projector, an encoder, and a decoder. For the projector, instead of using a sinusoidal encoding, the authors project the input feature with a simple normalized 2D positional map. For the encoder, the input feature is transformed into a set of $n$ tokens where each is a lower-dimensional semantic embedding with positional information. Then the tokens are fed as queries, keys, and values into the ViT.

The authors introduce a multi-branch search space inspired by HRNet [Wang et al., 2020]. This search space contains both multi-scale features and global contexts while maintaining high resolution representations throughout the network. The supernet in HR-NAS is a multi-branch network where each branch is a chain of search blocks operating at different resolutions. Each search block combines a MixConv and a Transformer [Vaswani et al., 2017]. The search is based on progressive shrinking that discards some convolutional channels and Transformer queries during training.

### Randomly Labeled Data

Random Label NAS (RLNAS) [Zhang et al., 2021] abandon ground truth labels and adopts random labels during searching. The authors of RLNAS decouple two convergence-based optimization steps.

First, they train a supernet with random labels:

$$W^* = \arg\min_{W} \mathbb{E}_{\alpha \sim \Gamma(A)} \mathcal{L}(\alpha, W, R), \tag{1}$$

where $W$, $\alpha$, $A$, $R$, $\mathcal{L}$, $\Gamma(A)$ denote the weights, architecture, architecture space, random labels, loss function, and a uniform distribution of $\alpha \in A$, respectively.

Second, the authors run an evolution algorithm that searches the optimal architecture based on a convergence-based metric $\text{Convergence}(\cdot)$ as fitness:

$$\alpha^* = \arg\max_{\alpha \in A} \text{Convergence}(\alpha, W^*). \tag{2}$$

The random labels follow a uniform distribution and the convergence-based metric is an angle-based model evaluation metric where it takes the angle between initial model weights and trained ones.

### Out-of-Distribution Generalization

NAS for image classification mostly works with clean datasets such as CIFAR10/100, ImageNet, etc. However, clean data is scarce in real world settings. In addition, data distribution shifts are often found in real world settings. For example, a dog breed identifier might be given a photo of a cat due to user error. Thus, there is a need for supernet optimization to be robust to Out-of-Distribution (OOD) cases. Out-of-Distribution refers to the case where a machine learning algorithm is fed data that is not from the distribution it was trained on. I.e., there is a distribution shift. There are mainly two kinds of OOD shifts: covariate shift and semantic shift. Covariate shift refers to a change in the input distribution. Semantic shift refers to a change in the output/label distribution.

The authors of NAS-OOD [Bai et al., 2021] aim to develop a method that is robust to OOD data. They carry this out by jointly optimizing a conditional generator to synthesize OOD examples during training. To elaborate, let $\alpha$, $\omega$, $\theta_G$ denote the parameters for the architecture topology, the supernet, and the conditional generator $G(\cdot, \cdot)$, respectively. The conditional generator $G(x, \cdot)$ takes data $x$ and domain labels $k$ as inputs. NAS-OOD is carried out by a minimax optimization. First, the conditional generator $G$ is learned to generate novel domain data by maximizing the validation loss, serving as an adversarial attack. Next, the validation loss is minimized by...
optimizing the architecture variables in $\alpha$ on the generated OOD images. The constrained minimax optimization problem can be formulated as:

$$\min_{\alpha} \max_{G} \mathcal{L}_{\text{val}}(\omega^*(\alpha), \alpha, G(x, \tilde{k}))$$

s.t. $\omega^*(\alpha) = \arg \min_{\omega} \mathcal{L}_{\text{train}}(\omega, \alpha, x)$,

where $G(x, \tilde{k})$ is the generated data from the original input data $x$ on the domain labels $\tilde{k}$. The optimization is carried out by gradient ascent for $\theta_G$ and gradient descent for $\omega$ and $\alpha$. In addition, the authors apply gradient descent to $\theta_G$ with respect to an auxiliary loss in order to improve the generator’s consistency and semantic information preservation when training.

### 3.2 Poor Rank Correlation Alleviation

In this section, we will survey methods that aims to alleviate poor rank correlation. After supernet optimization, a subnet needs to be sampled and evaluated. This process repeats and returns a list of ranked architectures. Unfortunately, supernet-based NAS suffers from poor rank correlation. I.e., model accuracy predictions are poorly correlated to the co-adaptation among the weights. I.e., when the shared weights have highly correlated behavior. Rank correlation is measured by three metrics: Kendall Tau ($\tau$) [Kendall, 1938], Spearman Rho ($\rho$), and Pearson R ($R$).

Poor rank correlation alleviation is usually accomplished by navigating or factorizing the search space. Navigation is carried out by selecting a path [Bender et al., 2018; Guo et al., 2020; Chu et al., 2021; Chu et al., 2020] in the supernet while search space factorization is done by block-wise [Li et al., 2021] or hierarchical partitioning [Zhao et al., 2021]. Disjoint from the aforementioned approaches, there is research that focuses on training consistency shift in the supernet [Peng et al., 2021].

#### Path Selection

Since architectures are sampled by taking subnets (paths) of the supernet, a natural approach is to smartly choose paths. I.e., this avenue of research is focused on navigating through the supernet search space. [Bender et al., 2018] utilizes a linearly scheduled dropout strategy during training. After learning the parameters, subnets are sampled following a uniform distribution. Aiming to improve upon [Bender et al., 2018], Single-Path One-Shot NAS (SPOS) [Guo et al., 2020] abstracts the supernet as a collection of choice modules. During training, SPOS randomly picks a subnet based on a uniform distribution and evaluates its validation accuracy.

Under the choice module paradigm, FairNAS [Chu et al., 2021] raises the issue of biased supernets, i.e., parameters are not trained evenly. FairNAS aims to make sure the parameters of each choice module are updated the same number of times. For example, if there are $m$ choice modules for each layer, FairNAS takes a uniform sampling without replacement of the choice modules, i.e., the $m$ choice module indices are permuted. Then, a model is built and evaluated based on the sampled index permutation. The search is accomplished via a genetic algorithm, NSGA-II [Deb et al., 2002].

The aforementioned methods fall under single path sampling, i.e., methods that sample only one architecture (subnet). An extension to this would be multi-path sampling, i.e., supernet optimization methods that sample multiple architectures. The authors of MixPath [Chu et al., 2020] find that a simple superposition of feature vectors sampled from multiple paths incurs very dynamic statistics causing training instability. They find that feature vectors from multiple paths are nearly multiples of those from a single-path. The solution to this is to scale the vectors down to the same magnitude creating stability. The authors accomplish said stability by proposing a novel Shadow Batch Normalization (SBN). MixPath is a two-stage pipeline: train supernet with SBN and searching for high performing models via NSGA-II.

#### Block-Wise Partitioning

It has been found that the rank correlation can vary depending on the search space itself [Zhang et al., 2020b]. For example, on the NAS-Bench-201 search space [Dong and Yang, 2020], the correlation can be as high as 0.7, whereas architectures sampled from DARTS-PTB [Liu et al., 2018] can perform worse than random search.

In an attempt to alleviate the co-adaptation, [Zhang et al., 2020a] proposed to reduce the weight sharing density (the number of architectures sharing the weight of one operator) by downsampling the search space. However, [Zhang et al., 2020b] finds that the performance improvement is only significant when the downscale factor is 64 on NAS-Bench-201 (only 244 different architectures).

Instead of shrinking the entire search space, other works [Li et al., 2020; Moons et al., 2021] partition the search space into blocks to improve the rank correlation. This allows the original search space size to remain intact. The current state-of-the-art (SOTA) NAS architecture on ImageNet for image classification is BossNet-T1+[Li et al., 2021]. The authors of Block-wise Self-supervised Neural Architecture Search (BossNAS) achieve SOTA results alleviating the rank correlation issue by using block-wise weight-sharing.

We give a brief primer on block-wise partitioning. Let $\mathcal{N}$ be the supernet. We break $\mathcal{N}$ into $N$ blocks by the supernet’s depth and have:

$$\mathcal{N} = \mathcal{N}_N \circ \cdots \circ \mathcal{N}_{i+1} \circ \mathcal{N}_i \circ \cdots \circ \mathcal{N}_1,$$

where $\mathcal{N}_{i+1} \circ \mathcal{N}_i$ denotes that the $(i+1)$-th block is connected to the $i$-th block in the supernet. Then each block is learned.
separately by optimizing:

\[ W_i^* = \min_{W_i} \mathcal{L}_{\text{train}}(W_i, A_i; X, Y), \quad i = 1, 2, \ldots, N, \quad (5) \]

where \( \mathcal{L}_{\text{train}}, W_i, A_i, X, Y \) denote training loss, weights of the \( i \)-th block, search space of the \( i \)-th block, input data, and ground truth labels respectively.

What inspired the authors of BossNAS to use block-wise weight-sharing were other works such as DNA [Li et al., 2020] and DONNA [Moons et al., 2021] which achieve high correlation and high efficiency. However, the authors of BossNAS argue that those prior works have limited application on search spaces with disparate candidates, such as CNNs and Transformers [Li et al., 2021]. BossNAS trains each block separately before searching among all blocks in a linear combination of each block’s validation loss:

\[ \alpha^* = \{\alpha_i\}^* = \arg \min_{\substack{\alpha_i \in A}} \sum_{i=1}^{N} \lambda_i \mathcal{L}_{\text{val}}(W_i^*, \alpha_i; X_i, Y_i) \]

\[ \text{s.t. } W_i^* = \min_{W_i} \mathcal{L}_{\text{train}}(W_i, A_i; X_i, Y_i) \quad (6) \]

where \( \alpha^* \) is the optimal architecture, \( \{\alpha_i\}^* \) is the same optimal architecture as a sequence of blocks, \( \lambda_i \) is the weight of the \( i \)-th block, \( A \) is the search space, \( X_i \) and \( Y_i \) are the training data and labels for the \( i \)-th block respectively.

On the MBCov Search Space, BossNAS produces strong rank correlations of \( \tau = 0.65, \rho = 0.78 \), and \( R = 0.85 \). In contrast, DARTS [Liu et al., 2018], an early supernet-based method shows abysmal correlation \( \tau = 0.08, \rho = 0.14 \) and \( R = 0.06 \).

**Hierarchical Partitioning**

The authors of few-shot NAS [Zhao et al., 2021] adopt a hierarchical partitioning approach. They argue that one-shot NAS using only one supernet may not be able to model the search space due to its limited capacity and co-adaptation of operations [Bender et al., 2018]. The hierarchical partitioning is done by splitting the connections in the supernet. In a supernet, each connection is a compound edge. i.e., all predefined candidate operations connect a pair of nodes (feature maps). In one-shot NAS, only one subnet is sampled from the supernet. For the case of few-shot, subnets are collected hierarchically when splitting compound edges. After sampling \( k \) subnets, they are trained separately.

Naturally, there is a trade-off between evaluation accuracy and inference time. The hierarchical splitting could be repeated so that the search space is partitioned exhaustively (i.e., brute force search), but that would be computationally infeasible. The authors find that only using 7 subnets establishes new state-of-the-art results on ImageNet: 80.5% top-1 accuracy at 600 MFLOPS and 77.5% top-1 accuracy at 238 MFLOPS.

The co-adaptation is found to be alleviated when measuring the rank correlation. The authors compared the correlation between the model’s predicted performance and the ground truth. The Kendall’s Tau scores for 1 sub-supernet (one-shot NAS), 6 sub-supernets, 36 sub-supernets, 216 sub-supernets, 1296 sub-supernets (the entire search space) are 0.013, 0.12, 0.26, 0.63, and 1.0 respectively. Naturally, more accurate predictions are made when more of the search space is incorporated.

**Supernet Training Consistency Shift**

The search space itself is not the only avenue for finding poor rank correlation causes. The authors of [Peng et al., 2021] empirically demonstrate that supernet training consistency shift hurts architecture ranking correlation. They break the consistency shift into two components: feature shift and parameter shift.

*Feature shift* refers to network instability due to input image perturbation. Let \( X_i, Y_i, \) and \( W_i \) denote the input, output, and weights of layer \( l \) respectively. Network instability refers to the disturbance in the loss \( \mathcal{L} \). By the chain rule for back propagation, we have:

\[ \frac{\partial \mathcal{L}}{\partial W_i} = \frac{\partial \mathcal{L}}{\partial Y_i} \frac{\partial Y_i}{\partial W_i} \]

This shows that architecture ranking preservation is highly dependent on the inputs \( X_i \). Since one-shot methods involve random path-sampling, the path that leads to layer \( l \) varies, and thus the input \( X_i \) varies.

*Parameter shift* refers to contradictory parameter updates for a given layer. During supernet training, a given layer \( l \) is always present in different paths throughout all iterations. However, the weights might have a contradictory update when iterating. Looking at the gradient descent update step: \( W_i^{t+1} \leftarrow W_i^t - \frac{\partial \mathcal{L}}{\partial W_i} \). There are two ways the rapidly varying \( W_i \) will hurt rank correlation. One, the loss is dependent on both the weights \( W_i \) and the descent \( W_i^{t+1} - \frac{\partial \mathcal{L}}{\partial W_i} \). i.e., stable weights can ensure a correct loss descent and guarantee an accurate architecture ranking, while erratic parameters could not achieve a correct ranking. Two, since the input \( X_i \) is generated by the network weights of the previous layers, varying parameters can also result in a feature shift, which further hurts architecture rank correlation.

The authors reduce the supernet training consistency shift by II-NAS: a nontrivial supernet-II model. They design a supernet-II model and a nontrivial mean teacher model [Tarlovinen and Valpola, 2017] to address feature shift and parameter shift respectively.
3.3 Transferable NAS

After supernet optimization, the sampled subnet requires post-processing: retraining, fine-tuning, etc. Training neural networks from scratch is computationally taxing, so there is interest in developing NAS methods that can yield architectures that can be instantly deployed for various tasks and hardware settings.

Specialized Networks for Specific Hardware Settings

Once-For-All (OFA) networks is a type of supernet that yields deployable sub-networks that require no further training for diverse tasks and hardware platforms [Cai et al., 2019]. The authors use progressive shrinking (PS) to train the OFA network. The idea is to initially train a large network and then progressively shrink it for fine-tuning with respect to 4 dimensions: width (channels), depth (layers), kernel size, and resolution. After training the full model, the authors set the filter size to be elastic, i.e., choose the kernel size to be 3, 5, or 7 at each layer while the depth and width remain the maximum values. The depth and width are then sequentially set to be elastic similarly. The resolution size remains elastic throughout the whole training process. For elastic kernel size, all kernels of different sizes share the same center. The concern that arises when kernels share the same center is that they need to play multiple roles (independent kernel and part of a large kernel) which can degrade performance. To address this, the authors use separate kernel transformation matrices for different layers. For elastic depth, if one wants a $D$ layer unit from an $N$ layer unit (such that $D < N$), the authors keep the first $D$ layers and skip the last $N - D$ layers. For elastic width, the authors introduce a channel sorting operation to support partial widths. If full width is $M$, then the top $K$ ($K < M$) important channels are retained. Importance is defined as computing the L1 norm of a channel’s weight (higher value meaning more important). The authors then use knowledge distillation [Hinton et al., 2015] after training the largest network.

After training the OFA network, the authors build neural-network twins to predict the latency and accuracy given a neural network architecture. The authors randomly sample 16K sub-networks with different architectures and image resolutions, then measure their accuracy on 10K validation images. The goal is to build an accuracy predictor fed with architectures as an input signal and accuracy as labels. The latency for a given architecture is predicted by utilizing a latency lookup table based on various target hardware platforms. When given a target hardware and latency constraint, the authors conduct evolutionary search based on the neural-network-twins to yield a specialized sub-network.

Mobile setting performance is of great interest when it comes to OFA. The authors compared ImageNet performance of OFA with MobileNetV3 [Howard et al., 2019] on 6 different mobile platforms: Samsung S7 Edge, Note8, Note10, Google Pixel1 & Pixel2, and LG G8. OFA consistently outperforms MobileNetV3 with the same latency constraints by training only once. Similar results were also found for specialized hardware accelerators: NVIDIA 1080TI & V100, Intel Xeon CPU, Jetson TX2, Xilinx ZU9EG FPGA, & ZU3EG FPGA.

BigNAS

BigNAS [Yu et al., 2020] aims to train a single supernet that yields subnets whose sizes range from 200 to 1000 MFLOPS without extra training or post-processing. In other words, the weights can be directly deployed after training. The authors coin such supernet as a big single-stage model. They select architectures using a simple coarse-to-fine selection method to find the most accurate model under resource constraints such as FLOPs, memory footprint and/or runtime latency budgets on different devices. When training the single-stage model the authors use two techniques [Yu et al., 2019]: the sandwich rule and inplace distillation. The sandwich rule, in each training step, given a mini-batch of data, samples the smallest child model, the biggest (full) child model and $N$ randomly sampled child models. It then aggregates the gradients from all sampled child models before updating the weights of the single-stage model. Here, “smallest” child means smallest input resolution, thinnest width, shallowest depth, and smallest kernel size. The motivation is to improve all child models in the search space simultaneously, by pushing up both the performance lower bound (smallest model) and the performance upper bound (full model).

4 Performance Comparison

Performance comparison can be done on multiple metrics: accuracy, latency, number of parameters, number of operations, search time, etc. On top of having a variety of different objectives, algorithms can be run on myriad environments and conditions: GPU models, number of GPUs, I/O overhead, etc. As a result, benchmarking NAS methods can be difficult. On top of this, NAS run on different search spaces can muddy the waters when benchmarking.

We provide a table listing different human and NAS designed architectures tasked to take on the ImageNet challenge. Supernet-based methods are highlighted and have their optimization strategy (poor rank correlation alleviation, transferability, data quality optimization) denoted. We also include top-1 accuracy, number of parameters, MACs, image resolution, and GPU days. Parameter and operation count are objective metrics that are only dependent on the algorithm. Image resolution is of concern because we are dealing with image classification; if a model performs well on a very low resolution, then it might be considered trivial. GPU days for search time would be heavily dependent on hardware specifications, but we include it because it has great practical importance. To offset high variance with respect to hardware, we include GPU settings for the models.

5 Future Directions

While this survey is about supernet optimization methods for image classification, the literature is lacking with regards to other computer vision tasks: semantic segmentation, object detection, etc. In other words, supernet research is primarily focused on achieving architectures that make the state-of-the-art on ImageNet.

Most works are concerned with maximizing classification accuracy, reducing the search time, and making the sampled
## 7 Acknowledgement

We would like to thank Ph.D. candidate Sangmook Kim (OSI Lab, KAIST) for giving feedback on clarity and conciseness.

### 6 Conclusion

Most NAS surveys focus on general NAS and puts less emphasis on supernet optimization. In addition, said surveys are structured by decomposing each method with respect to search space, search strategy, and performance estimation. This survey covers supernet optimization methods and adopts the structure from [Ren et al., 2020] by pairing challenges and solutions, and categorizing methods based on said pairs. This structure is used with the intention to build intuition on supernet NAS by thinking in terms of issues and respective fixes. We also provided a discussion on future directions with regards to research that is currently lacking in the supernet optimization literature.
on Computer Vision and Pattern Recognition, pages 1989–1998, 2020.

[Li et al., 2021] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Boss-nas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. arXiv preprint arXiv:2103.12424, 2021.

[Liu et al., 2017] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. arXiv preprint arXiv:1711.00436, 2017.

[Liu et al., 2018] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055, 2018.

[Liu et al., 2021] Jihao Liu, Ming Zhang, Yangting Sun, Boxiao Liu, Guanglu Song, Yu Liu, and Hongsheng Li. Fnas: Uncertainty-aware fast neural architecture search. arXiv preprint arXiv:2105.11694, 2021.

[Lu et al., 2021] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kateryna Deb, and Vishnu Naresh Boddeti. Neural architecture transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(9):2971–2989, 2021.

[Moons et al., 2021] Bert Moons, Parham Noorzad, Andrii Skliar, Giovanni Mariani, Dushyan Mehta, Chris Lott, and Tijmen Blankevoort. Distilling optimal neural networks: Rapid search in diverse spaces. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 12229–12238, 2021.

[Pasunuru and Bansal, 2019] Ramakanth Pasunuru and Mohit Bansal. Continual and multi-task architecture search. arXiv preprint arXiv:1906.05226, 2019.

[Peng et al., 2021] Jiefeng Peng, Jiqi Zhang, Changlin Li, Guangrun Wang, Xiaodan Liang, and Liang Lin. Pi-nas: Improving neural architecture search by reducing supernet training consistency shift. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 12354–12364, 2021.

[Pham et al., 2018] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In International Conference on Machine Learning, pages 4095–4104. PMLR, 2018.

[Real et al., 2017] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In International Conference on Machine Learning, pages 2902–2911. PMLR, 2017.

[Real et al., 2019] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In Proceedings of the aaai conference on artificial intelligence, volume 33, pages 4780–4789, 2019.

[Ren et al., 2020] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. arXiv preprint arXiv:2006.02903, 2020.

[Shi et al., 2019] Han Shi, Renjie Pi, Hang Xu, Zhenguo Li, James T Kwok, and Tong Zhang. Bridging the gap between sample-based and one-shot neural architecture search with bonas. arXiv preprint arXiv:1911.09356, 2019.

[Tarvainen and Valpola, 2017] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. arXiv preprint arXiv:1703.01780, 2017.

[Vaswani et al., 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

[Wang et al., 2020] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual understanding. IEEE transactions on pattern analysis and machine intelligence, 2020.

[Wang et al., 2021a] Dilin Wang, Chengyue Gong, Meng Li, Qiang Liu, and Vikas Chandra. AlphaNet: Improved training of supernets with alpha-divergence. In International Conference on Machine Learning, pages 10760–10771. PMLR, 2021.

[Wang et al., 2021b] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6418–6427, 2021.

[Wortsman et al., 2022] Mitchell Wortsman, Gabriel Ibarco, Samir Yitzhak Gadre, Rebecca Roelof, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. arXiv preprint arXiv:2203.05482, 2022.

[Xie et al., 2018] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. arXiv preprint arXiv:1812.09926, 2018.

[Xu et al., 2019] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PCDarts: Partial channel connections for memory-efficient architecture search. arXiv preprint arXiv:1907.05737, 2019.

[You et al., 2020] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1999–2008, 2020.

[Yu and Huang, 2019] Jiahui Yu and Thomas S Huang. Universally-slimmable networks and improved training techniques. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 1803–1811, 2019.

[Yu et al., 2020] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In European Conference on Computer Vision, pages 702–717. Springer, 2020.

[Zhai et al., 2021] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. arXiv preprint arXiv:2106.04560, 2021.

[Zhang et al., 2020a] Yuge Zhang, Zeyun Lin, Junyang Jiang, Quanlu Zhang, Yujing Wang, Hui Xue, Chen Zhang, and Yaming Yang. Deeper insights into weight sharing in neural architecture search. arXiv preprint arXiv:2001.01431, 2020.

[Zhang et al., 2020b] Yuge Zhang, Quanlu Zhang, and Yaming Yang. How does supernet help in neural architecture search? arXiv preprint arXiv:2010.08219, 2020.

[Zhang et al., 2021] Xuyang Zhang, Pengfei Hou, Xiangyu Zhang, and Jian Sun. Neural architecture search with random labels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10907–10916, 2021.
[Zhao et al., 2021] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. Few-shot neural architecture search. In *International Conference on Machine Learning*, pages 12707–12718. PMLR, 2021.

[Zhong et al., 2018] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2423–2432, 2018.

[Zoph and Le, 2016] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[Zoph et al., 2018] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.