Does Interference Exist When Training a Once-For-All Network?

Jordan Shipard[1], Arnold Wiliem[2], and Clinton Fookes[1]

[1]Signal Processing, Artificial Intelligence and Vision Technologies (SAIVT), Queensland University of Technology, Australia
[2]Sentient Vision Systems, Australia
{jordan.shipard@hdr., c.fookes@}qut.edu.au, arnoldw@sentientvision.com

Abstract

The Once-For-All (OFA) method offers an excellent pathway to deploy a trained neural network model into multiple target platforms by utilising the supernet-subnet architecture. Once trained, a subnet can be derived from the supernet (both architecture and trained weights) and deployed directly to the target platform with little to no retraining or fine-tuning. To train the subnet population, OFA uses a novel training method called Progressive Shrinking (PS) which is designed to limit the negative impact of interference during training. It is believed that higher interference during training results in lower subnet population accuracies. In this work we take a second look at this interference effect. Surprisingly, we find that interference mitigation strategies do not have a large impact on the overall subnet population performance. Instead, we find the subnet architecture selection bias during training to be a more important aspect. To show this, we propose a simple-yet-effective method called Random Subnet Sampling (RSS), which does not have mitigation on the interference effect. Despite no mitigation, RSS is able to produce a better performing subnet population than PS in four small-to-medium-sized datasets; suggesting that the interference effect does not play a pivotal role in these datasets. Due to its simplicity, RSS provides a $1.9\times$ *reduction in training times compared to PS. A* $6.1\times$ *reduction can also be achieved with a reasonable drop in performance when the number of RSS training epochs are reduced. Code available at* https://github.com/Jordan-HS/RSS-Interference-CVPRW2022

1. Introduction

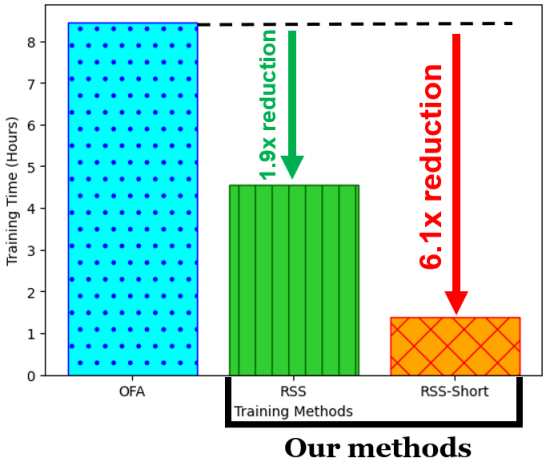
Deploying deep neural network models for real-world applications requires accurate and fast inference [37]. Generally, these accuracy and latency constrains are competing with one another, with only architecturally optimal networks being able to achieve both. Designing these optimal networks by hand is a challenging task and has led to the explosion of the neural architecture search (NAS) field.

Some initial NAS methods [1, 26, 40] were based on reinforcement learning approaches and proved the concept viable. Unfortunately, these approaches required extremely high computational resources as they train hundreds of architectures. Moreover, these only searched the architectures which optimise accuracy.

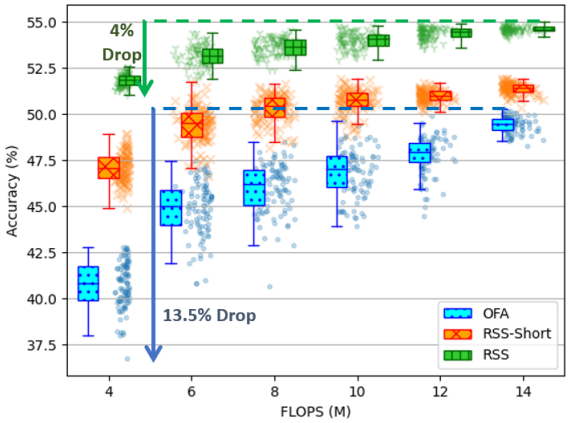
The goal of the approach eventually shifted to producing optimal networks with high accuracy and fast inference. To this end, two different approaches were developed: (1) the direct NAS methods [5, 8, 15, 23, 33]; and (2) the one-shot methods [2, 3, 11, 34]. The former constructs a continuous search space and utilises gradient descent for the search. Whilst, the later uses a discrete search space which only requires the neural network architecture search space to be trained once. This means the search becomes much faster and requires significantly fewer computational resources.

Previous one-shot methods [2, 3, 11, 34] use the trained search space as a guide for finding optimal networks. Once the optimal networks are found, they are trained from scratch before deployment. This presents a problem if we wish to deploy different network variants for various deployment platforms. To address this, the Once-For-All (OFA) method [4] trains a supernet wherein a sub-network/subnet architecture and its weights can be directly sampled from the supernet. Once sampled, a small amount of fine-tuning might be applied before the subnet is deployed; thus, side stepping the need for training the sampled subnets from scratch for each deployment platform.

To achieve its goal, OFA needs to train a large subnet population (2×10^{19} subnets) [4]. When attempting to train the subnet population using naive approaches they find the subnets interfere with each other, resulting in significant accuracy drops. This is taken to suggest that modifying the weights of one subnet could affect the performance of other subnets in the subnet population. However, this interference is not currently well understood. To address the interference, OFA proposes a novel training method, called Progressive Shrinking (PS). PS trains the largest architecture first (*i.e.,* the supernet) and then progressively samples and trains smaller subnet architectures. Recent work in [27] showed that it is possible to improve the accuracy of the



(a) Training time measured from the start of the first training epoch and the finish of the last epoch for OFA, RSS (proposed) and RSS-Short (proposed) methods. Refer to Tab. 1 for number of training epochs.



(b) Subnet population performance of the OFA, RSS (proposed) and RSS-Short (proposed). Both proposed variants have better top performing subnets with a lower variance in the population. More detailed results are reported in Section 6.2

Figure 1. Training time reduction and accuracy improvements of the proposed Random Subnet Sampling (RSS) compared to the Once-For-All (OFA) method. Results on CIFAR100 dataset [17].

subnets by reducing the search space. The work suggests the accuracy gained is due to a reduction in interference resulting from the reduced search space. Therefore, the limiting factor on improving subnet population performance appears to be related to the interference effect between subnets during training.

In this work, we take a second look at this effect. In particular we ask the following questions: (1) Does the interference effect exists? (2) If it exists, then by how much does it affect the subnet population's performance? (3) If any other factors impact the subnet population's performance? To examine these questions, this paper introduces a simple method dubbed Random Subnet Sampling (RSS), which randomly samples a single subnet to train at each epoch. Obviously, RSS does not have any mitigation on the interference effect. We compare RSS with OFA's PS method on four datasets: MNIST [20], Fashion-MNIST [35], CIFAR10 [17], and CIFAR100 [17]. To our surprise, the subnet population is able to better generalise and achieve higher accuracies than PS. Fig. 1 shows the main findings from this work.

Our findings suggest that interference between subnets has a minimal effect on the subnet population's performance. Instead, we argue that bias in the subnet selection scheme during training has a larger impact on performance. When a subnet architecture is sampled and trained more often than the others, the subnet tends to have significantly higher accuracy. On the other hand, the performance of rarely sampled subnet architectures tend to have significantly lower accuracy. This sampling bias is analogous to the bias introduced when training a neural network model with an imbalanced dataset. The proposed RSS method addresses this sampling bias by uniformly sampling the subnet architecture during each epoch.

We also observe that the interference effect becomes more apparent when combining multiple subnets gradients during a single update step. This corroborates recent findings by Xu *et al*. [36] in the Natural Language Processing (NLP) field.

Contributions - Our contributions are listed as follows,

1. In contrast to the recent belief, we show that the interference has minimal effect when training the subnet population.
2. Instead, we argue that bias in the subnet selection scheme during training plays a bigger role.
3. We also show that the interference effect becomes more pronounced when combining the gradients of multiple subnets in a single update step.
4. To show the above points, we propose a simple-yet-effective method called Random Subnet Sampling (RSS). The proposed RSS method outperforms Once-For-All's Progressive Shrinking (PS) method which suggests point (1). In addition, the reason why RSS has good performance is because it addresses the bias problem as stated in point (2) and it only trains a single subnet for each epoch, in line with point (3).

We continue our paper as follows. Related works are discussed in Section 2. The subnet population training problem formulation is presented in Section 3. We then introduce the proposed method in Section 4 and discuss subnet sampling bias during training in Section 5. Section 6 presents the experimental results. Finally, Section 7 discusses conclusions and the future direction of this work.

2. Related Work

One-Shot NAS - The goal of Neural Architecture Search (NAS) is to search for an optimal architecture in a large architecture search space. This search space can be continuous, in the case of direct NAS methods [5, 8, 15, 23, 33], or discrete, in the case of one-shot methods [2, 3, 11]. Additionally, the name 'one-shot' refers to the subnet population only requiring to be trained once; whereas previous methods used reinforcement learning and trained hundreds of individual networks during search [1, 26, 40]. One-shot methods, such as Once-For-All [4], train large discrete search spaces using weight sharing techniques [6, 12, 13, 39] removing the requirement to train every architecture. The relative accuracy of subnets in the subnet population is then used to find an optimal architecture. Once found, the architecture is trained from scratch as the previous shared weights are not fully optimised. As a further improvement, various methods were able to remove this need for retraining [4, 16, 22, 27, 30, 38], allowing subnets to be directly extracted from the subnet population. Despite its computational benefits, weight sharing is not a perfect solution as it is believed to introduce the problem of interference between subnets. In this work we look to further study this interference as it is currently under explored.

Interference Effect in One-Shot Training - Most of the current understanding around the effects of interference come from observations made during the application of specific methods. Guo *et al*. [11] states that the weights in the supernet are deeply coupled, although no further study to explain the underlying mechanism of coupling is provided. Cai *et al*. [4] state that randomly selecting subnets for training causes interference and accuracy drops; which Sahni *et al*. [27] echo and believe their compound heuristic is able to reduce interference and therefore improve accuracy. Liu *et al*. [34] state that the existence of unnecessary neurons and connections in the supernet negatively impacts training and leads to greater interference. Perhaps, the most closely related study of interference is from the field of Natural Language Processing (NLP) by Xu *et al*. [36]. They conduct a specific analysis of the interference effect during training a single-path one-shot NAS method. Their work finds that interference between subnets is caused by diverse gradient directions produced by multiple sampled subnets. Combining these diverse gradient directions would hamper the training progress. Furthermore, large architectural differences between sampled subnets contribute to the diverse gradient directions. Our work differs as we train a subnet population constructed from a computer vision-based neural network model, instead of a single-path subnet population for NLP. Despite these differences we find their findings to be consistent with our own.

Bias in One-Shot Training - The effect of bias during one-shot training has been raised recently by Chu *et al*. [7]. Their work observe an unfair bias in the previous one-shot methods' [23, 25] supernet training as the cause for the poor correlation between the subnet proxy performance and its corresponding standalone trained performance (*i.e.,* trained from scratch). Since the proxy performance is used as a guide when searching for optimal architectures, low correlation with the standalone trained performance will produce suboptimal architectures. To this end, they propose a strict fairness based training method. Despite bias being the core issue, their work only focus on the training fairness issue and does not extensively explore the bias itself. Different to their work, in this work we explain bias in the subnet sampling/selection during training of a OFA subnet population, to be analogous to bias in the data imbalance problem. This analogy allows us to borrow solutions developed for this problem to help us address the subnet sampling bias.

Data Imbalance - Real world data is not balanced [32] like the ones commonly used in scientific works. This data imbalance or bias is a well known problem within image classification and many solutions have been developed to address it. These include; Balanced sampling [10, 18, 32]; Hard mining [24, 28, 29, 31] which directly relate to ideas used by Wang *et al*. [30] for worst-up training; and focal loss [9, 21]. As mentioned, we propose that the subnet sampling bias can be explained from the data imbalance perspective.

3. Problem Description

We follow the problem definition presented in the original OFA work [4]. Let $\mathrm{arch}_i \in \mathcal{A}$ be the i-th subnet architecture. It is assumed that all subnet architectures in $\mathcal{A}$ can be derived/sampled from the supernet (the largest model amongst all). Let $\mathbf{W_o}$ be the weights of the supernet and $\mathbf{W}_{\mathrm{arch_i}}$ be the weights of arch_i. We derive $\mathbf{W}_{\mathrm{arch_i}}$ from $\mathbf{W_o}$ via a selection function $\mathrm{C}(\cdot)$. The subnet population training problem can then be formalised via a minimisation problem as,

$$\arg\min_{\mathbf{W_o}} \sum_{arch_i \in \mathcal{A}} \mathcal{L}_{val}(C(\mathbf{W_o}, arch_i)), \tag{1}$$

where $\mathcal{L}_{val}$ is the loss on the validation set. Essentially, the above equation aims at minimising the loss of every subnet from $\mathcal{A}$ on the validation set. In the case of the OFA, and for this paper, the size of $\mathcal{A}$ is 2×10^{19} subnets.

4. Subnet Population Training

Due to the large population size (*i.e.,* $|\mathcal{A}| = 2 \times 10^{19}$), directly minimising Eq. 1 is impractical. The general approach is to sample and train a subset $\tilde{\mathcal{A}} \in \mathcal{A}$. The subset $\tilde{\mathcal{A}}$ is sampled according to some selection scheme and set via $\mathrm{C}(\cdot)$. Any subnet that does not belong to $\tilde{\mathcal{A}}$ will still be indirectly trained due to its weights being shared with the subnets in $\tilde{\mathcal{A}}$. In this section we first briefly discuss PS, proposed in the original OFA paper [4], and then propose Random Subnet Sampling (RSS).

4.1. Progressive Shrinking

The main idea of the PS [4] is to train the subnets with respect to their Floating Point Operation (FLOP) size; from the largest one (*i.e.,* supernet) to the smaller ones progressively.

The supernet is required to be trained prior to using the progressive shrinking method. Once the supernet is trained, PS trains the subnet population in three stages by performing a controlled sampling on three network parameters: (1) kernel size (*e.g.,* 7x7, 3x3); (2) number of layers in each block (Depth); and (3) number of channels (Width).

In the first phase, named dynamic kernel training, only the kernel size is varied while the depth and width are kept at their maximum values. In this stage a single subnet is sampled for each update step. In the second phase, called dynamic depth training, the method varies both the number of layers and the kernel size, while width still remains at its maximum value. This stage samples two subnets and combines the gradient from both for each update step. Finally, in dynamic width training, the method varies all three parameters and combines the gradients from four sampled subnets during each training step.

4.2. Random Subnet Sampling

We propose Random Subnet Sampling (RSS) which is not designed to mitigate the interference effect. In contrast to the progressive shrinking method, RSS does not perform any controlled sampling. It samples the subnets by varying all three network parameters used in PS (*i.e.,* the kernel size, depth and width). We use uniform randomness to choose the value for each parameter. Unlike PS which samples subnets for each update step, or batch, RSS samples a single subnet for each epoch. We show later that per-epoch sampling is a more effective method than per-batch sampling. Algorithm 1 illustrates the proposed RSS method.

Algorithm 1 Pseudo code for the proposed Random Subnet Sampling (RSS) method. Kernel_settings, width_settings and depth_settings are sets of values. For instance, width_settings = {3, 4, 6}. $\mathrm{rand}(\cdot)$ is a sampling function that uniformly samples values from the input argument.

Input: kernel_settings, width_settings, depth_settings, n_epochs
Output: Trained subnet population

1: **while** $i \leq n\_epochs$ **do**
2: $\quad \mathbf{subnet\_k} \leftarrow \mathrm{rand}(kernel_{settings})$
3: $\quad \mathbf{subnet\_e} \leftarrow \mathrm{rand}(width_{settings})$
4: $\quad \mathbf{subnet\_d} \leftarrow \mathrm{rand}(depth_{settings})$
5: $\quad subnet \leftarrow \mathrm{C}(\mathbf{subnet\_k}, \mathbf{subnet\_e}, \mathbf{subnet\_d})$
6: $\quad$ train $subnet$ for one epoch
7: **end while**

The kernel, width and depth settings are one dimensional vectors with the length prescribed as follows.

The length of the kernel settings vector ($\mathbf{subnet\_k}$) and width settings vector ($\mathbf{subnet\_e}$) is equal to the maximum depth setting multiplied by the number of blocks in the supernet. As an example, if the supernet has five blocks, with the maximum depth setting being four. Our kernel and width settings vectors would therefore contain 20 values (i.e. 5 blocks $\times$ 4 layers = 20). The length of the depth settings vector ($\mathbf{subnet\_d}$) is equal to the number of blocks. As we see from Algorithm 1, the values of these three setting vectors are randomised between lines 2-3. The subnet is then derived by passing the sampled settings into the selection function $\mathrm{C}(\cdot)$ and trained. More specifically, we follow OFA [4] to derive the subnet and its weights from the supernet by using the sampled settings.

5. Subnet Sampling Bias

Different sampling strategies between PS and the proposed RSS can be studied from the sampling bias perspective. We argue that Eq. 1, bears similarities to the standard classification problem [19]. To train a classification model, we can solve the following minimisation problem.

$$\min_{\mathbf{W}^*} \sum_{i=1}^{M} \mathcal{L}_{train}(\mathbf{W}^*, \mathbf{x_i}, y_i), \tag{2}$$

Where $\mathbf{x_i}$ and y_i are the i-th data point and its corresponding class label. $\mathbf{W}^*$ is the neural network weights and $\mathcal{L}_{train}$ is the training loss.

Similar to Eq. 1, the above minimisation problem aims at reducing the loss for each data point in the training set, $(\mathbf{x_i}, y_i)_{i=1}^{M}$. The difference for Eq. 1 is that instead of summing the loss over all the data points, it sums the loss over all the subnet architectures.

Linking the subnet population training problem presented in Eq. 1 with the classification problem in Eq. 2 is appealing as we can use the tools/experience developed in the classification problem into this field. For instance, bias in the data is one of the big themes in the classification field [32]. When training a network with an imbalanced training set (*i.e.,* some classes have significant higher number of training data than others), the network will tend to have stronger confidence scores towards these large classes.

Using this new perspective, we argue that PS is a biased selection scheme, as it initially trains the supernet before training the largest to smallest subnets. As described in Section 4.1, PS has three phases which progressively varies the three network parameters. As an example, during dynamic kernel training the width and depth are set at their maximum values. This results in only larger subnets being trained during this stage. Overall this could lead to the effect obsereved by Chu *et al.* [7], with the larger subnets (especially the supernet) performing better than smaller subnets, simply due to more training.

Unlike PS, RSS randomly samples subnets for the entire training duration. This means, the subnet sampling is not biased towards the larger subnets. We will show in our results that the subnet sampling bias does indeed play a significant role in the overall performance; and that this role is similar to the data sampling bias in the classification problem.

6. Experimental Results

We contrast between the proposed RSS method and PS in this section. If the interference effect has significant contribution, then RSS will have significant lower performance than PS. First, the experimental set-up and the datasets are discussed. Then, we present our main results. Finally, we show additional ablation experiments.

6.1. Experiment Setup

Datasets and Methods - The datasets used for comparison are MNIST [20], Fashion-MNIST (FMNIST) [35], CIFAR10 [17], CIFAR100 [17]. These datasets were used to cover a range of dataset complexities with MNIST being the least complex and CIFAR100 being the most complex.

We compare three training methods as follows.

OFA [4]- The progressive shrinking method is used.

RSS- The proposed random subnet sampling with the same number of training epochs as OFA.

RSS-short- The proposed RSS with the same number of training epochs as only OFA's supernet training.The training epochs for each method are shown in Tab. 1.

All training was conducted on a single Nvidia RTX3080 Graphics Processing Unit (GPU) with a initial learning rate of 0.01 using cosine learning rate decay; batch size of 64; momentum of 0.9; weight decay of $3e^{-5}$ and used cross-entropy loss, unless otherwise specified.

Subnet Population Architecture - In this work, the MobileNetV3 [14] is used as the base architecture. From our empirical observations (provided in supplementary material), our findings are still consistent when a different base architecture is used.

The subnet derivation from the base architecture is kept consistent with the settings used by OFA. The subnet population is defined along three mutable dimensions, the kernel size, layer width (number of channels), and layer depth (number of layers for each block). The possible settings for each dimension are shown below.

- Kernel $\in \{3 \times 3, 5 \times 5, 7 \times 7\}$.
- Width $\in \{3, 4, 6\}$.
- Depth $\in \{2, 3, 4\}$.

The supernet consists of five configurable blocks, shown in Fig. 2. All subnets are unique combinations of the kernel, width and depth settings. The kernel is set layer by layer and controls the active kernel size. The width, or expansion ratio, is also set layer by layer and is a multiplier for the base channel count of each layer. The depth is set for each block and controls the number of active layers in that block. We follow OFA [4] for transforming supernet weights to derive the prescribed subnet.

Evaluation Protocol - OFA [4] evaluate their training methods performance according to the top performing subnets at various latency constraints. However, improving the performance of top-performing subnets may not correlate to a better performing subnet population. We instead wish to evaluate the average performance of the overall subnet population. This is done by randomly sampling a group of subnets according to the specific architectural size measurement of Mega FLoating-point Operation Per Seconds (MFLOPs). The performance of all sampled subnets is then recorded on the test set. For all datasets, we use the following architecture sizes of evenly spaced MFLOPs values $\{4, 6, 8, 10, 12, 14\}$. Subnets of size 6-12 MFLOPs can be found by simply randomising all settings until the resulting subnet is within the required MFLOP range, ± 0.5 in our case. Subnets with size 4 and 14 MFLOPs are rare and challenging to sample in this manner. As such, to find these subnets, the width setting is locked at its minimum value to aid sampling of 4 MFLOP subnets; and locked at its maximum to aid sampling of 14 MFLOP subnets. We use the wall clock to measure the training time for each method from the start of the first epoch to the conclusion of the last epoch.

	Training Epochs		
Dataset	OFA	RSS-Short	RSS
MNIST	10(s)+27(ps)	10	37
F-MNIST	25(s)+57(ps)	25	82
CIFAR10	180(s)+410(ps)	180	590
CIFAR100	180(s)+410(ps)	180	590

Table 1. Training epochs for each method on each dataset where (s) denotes supernet training and (ps) denotes progressive shrinking training. For CIFAR10 and CIFAR100 we use the same training protocol as detailed by Cai *et al.* [4]. Training on MNIST and F-MNIST are scaled versions of the same CIFAR10 and CIFAR100 training with less epochs.

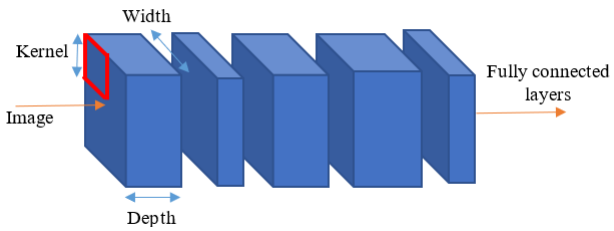
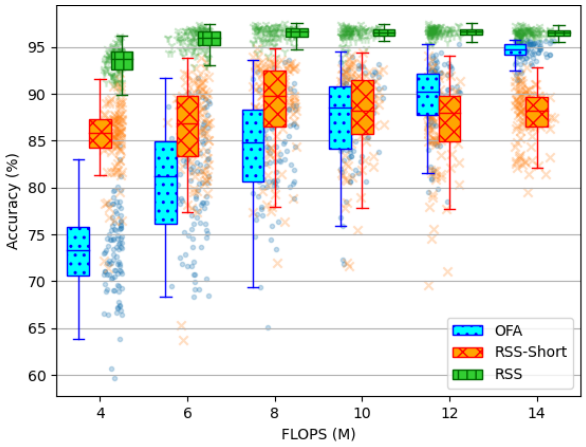
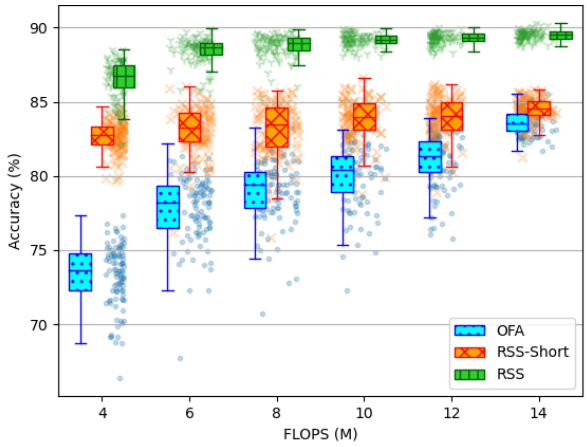


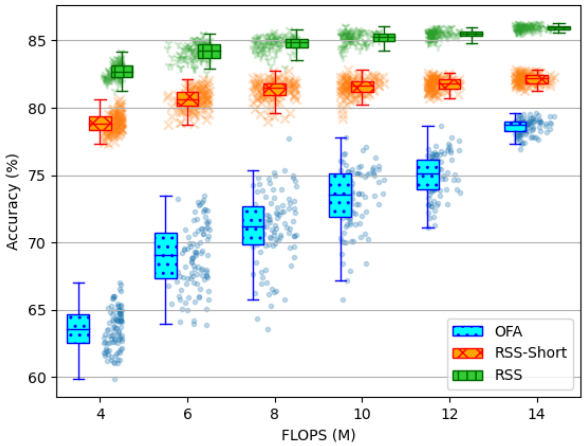
Figure 2. Diagram showing the MobileNetV3 [14] base architecture used to sample subnets with kernel $\in$ {3x3,5x5,7x7}, width $\in$ {3,4,6} and depth $\in$ {2,3,4}.



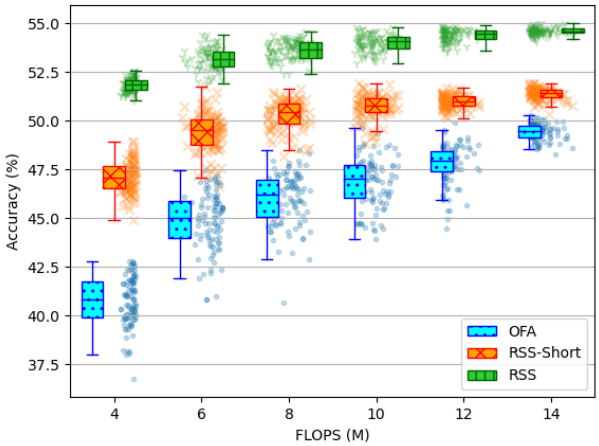
(a) MNIST Subnet Population Performance.



(b) FMNIST Subnet Population Performance.



(c) CIFAR10 Subnet Population Performance.



(d) CIFAR100 Subnet Population Performance.

Figure 3. Comparisons between OFA, RSS (proposed), and RSS-Short (proposed) subnet population performance across four datasets. RSS is able to consistently train a better performing subnet population for all datasets.

6.2. Main Results

Subnet Population Performance - The main results are presented in Fig. 3. Across all datasets, RSS achieves the best performing subnet population. The gap between RSS and OFA grows larger as the difficulty of the dataset increases (i.e., from MNIST to CIFAR). Furthermore, the range of best performing to worst performing subnets is significantly smaller for RSS methods. These results suggest that the interference effect between subnets during training is negligible.

In Section 6.3 we show RSS's consistent performance across different subnet sizes is due to both its unbiased subnet sampling scheme and reduction of interference. In contrast to RSS, OFA shows a significant bias towards larger subnets in all datasets. We attribute this to its initial supernet training and PS training procedure which starts from the larger subnets. Subnet sampling bias is further studied in Section 6.3.2.

Training Time - Fig. 4 shows the number of hours required to train each of the methods. Showing RSS and RSS-short are an average 1.9 and 6.1 times faster than OFA respectively. For RSS-short this is expected as it runs for nearly half the epochs of OFA; however, RSS and OFA run for the same number of epochs. This speed up is due to two factors: (1) RSS is more likely to train smaller subnets each epoch; and (2) RSS only trains a single subnet each epoch where OFA's PS trains two subnets during dynamic depth training and four during dynamic width training.

6.3. Why does RSS outperform OFA?

Our main results suggest that both proposed RSS variants outperform OFA. In this section we conduct further investigations by studying the two main differences between the methods: (1) the number and frequency of subnets sampled during training; and (2) the selection scheme for selecting subnets during training. We only present the results from CIFAR100 dataset for the study. We choose CIFAR100 as it offers more variations and complexity com-

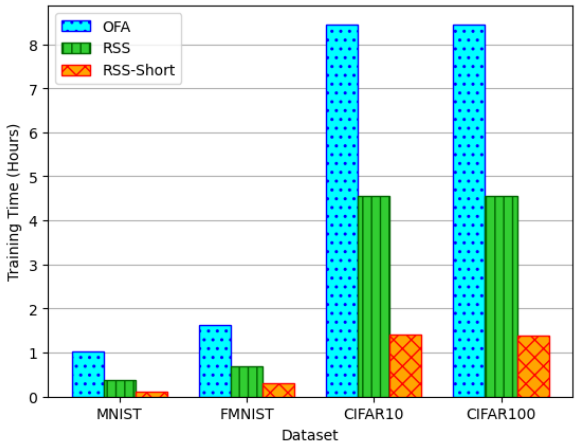


Figure 4. Comparisons of the training time, in hours, for each method as measured on a single Nvidia RTX3080 GPU.

pared to the other datasets. Furthermore, we only use RSS with the standard number of training epochs (590). Results (in supplementary material) on RSS-Short are similar; in addition to showing our results are not limited to specific hyperparemeter values.

6.3.1 Effect of Number of Subnets Sampled

The proposed RSS method samples a single subnet to train each epoch, whereas OFA samples and trains up to four subnets per update step. In this section, we modify the RSS subnet sampling to be similar to OFA, allowing us to compare subnet sampling methods on RSS. More specifically, we first change RSS's sampling from each epoch into sampling each batch (i.e., per update step). We then increase the number of sampled subnets for each batch from one to two. When sampling two or more subnets, we combine their gradients in each update step. Fig. 5 shows that despite training fewer subnets, per-epoch sampling produces a better performing subnet population than per-batch sampling. Indicating interference may be more severe in per-batch strategies. As an alternative explanation, we speculate there could be a benefit to allowing sampled subnets to train on the entire dataset instead of only a single batch. Future study is required to investigate this. The interference again appears to increase between sampling one and two subnets per batch. We conjecture this increased interference is likely due to the combining of diverse gradient directions, as also shown in [36].

6.3.2 Effect of Different Selection Schemes

The previous section investigated the number of subnets to sample. In this section, we study the subnet selection scheme. As mentioned in Section 5, the selection scheme can induce bias which skews the subnet population's performance towards the more frequently sampled subnets. In this section, we intentionally use biased subnet selection schemes to study the effects on the subnet population.

Single subnet selection - The simplest biased subnet selection scheme to test is training a single subnet. More specifically, we compare three variants of this biased selection scheme: (1) **smallest only**, with all settings at their minimum values; (2) **middle only**, where kernel=5, expand=4 and depth=3 for all blocks; and (3) **largest only**, where all settings are set to their maximum (i.e. the supernet). Fig. 6 shows that the subnet performance is skewed towards the size of the selected subnet. Furthermore, the performance drops off as soon as the subnet size deviates from the selected subnet size. Indicating excessive sampling of a subnet during training induces a strong bias in the subnet population's performance.

Two subnet selection - Fig. 6 shows that single subnet selection introduces a bias towards the selected subnet. In this part, we examine the effect of training two subnets. More specifically, we select either the smallest or the largest subnet for training. We compare three variants of selection schemes: (1) select the largest subnet for the first half of the total epochs, then select the smallest subnet the rest of the training (**Max then Min**); (2) select the smallest subnet for the first half of the total epochs, then select the largest subnet the rest of the training (**Min then Max**); and (3) alternating between smallest and largest subnets each epoch (**Alternating**). Unlike OFA which uses cosine learning rate decay, in this part of experiment, we use a constant learning rate throughout the training course. We report the results in Fig. 7.

Both **Min then Max** and **Max then Min** schemes skew their results towards the subnet most recently trained; however, it is clear that both subnet populations benefited from the training of an additional subnet. This is shown by **Max then Min** having better performing subnets at larger MFLOPs when compared to training the smallest subnet only in Fig. 6. The same can be observed for **Min then Max** at lower MFLOPs when compared to training the largest subnet only in Fig. 6. Interestingly, we do not see mirrored results between **Min then Max** and **Max then Min** schemes. This shows that the training sequence within the selection scheme is an important factor.

Another interesting observation is that the **Max then Min** scheme has lower performance than the **Min then Max** scheme. OFA has a similar strategy to the **Max then Min**. However, OFA uses a decreasing learning rate and a more controlled selection scheme which might help to mitigate this issue. A future study on this is warranted.

The **Alternating** scheme produces the best results with a more evenly trained subnet population and only slight bias towards the largest and smallest subnets. This further suggests that mitigating the subnet sampling bias during training is an important issue to address. Interestingly, despite never being trained, the subnets in ranges 8-10 MFLOPs

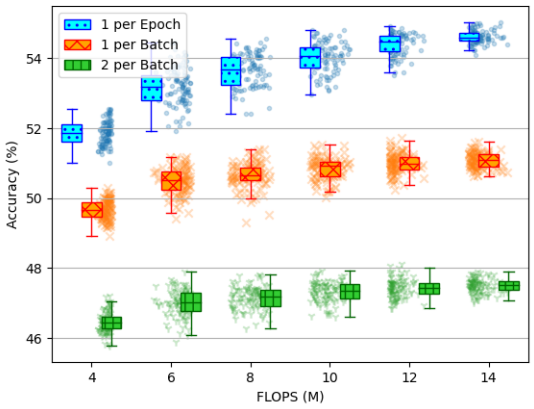


Figure 5. Effect of sampling different numbers of subnets during training. The 1 per epoch strategy is the default RSS method. When sampling 2 per batch the gradients are combined for the update step. As we can see, performance drops for per-batch strategies. The drop is more significant when more subnets are sampled per-batch (i.e., 2 per batch).

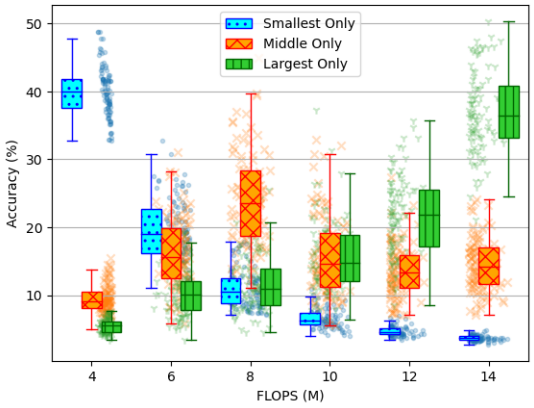


Figure 6. Resulting population performance from training a single subnet only. The **smallest only** uses only the smallest subnet which is constructed by settings all settings to their minimum. The **middle only** is when kernel=5, expand=4 and depth=3. The **largest only** is the supernet, when all settings are at their maximum. The results show a heavy bias towards the selected subnet.

perform nearly on par with the larger and smaller subnets. The **Alternating** scheme uses an alternating selection strategy, producing a less biased subnet population towards both trained subnet sizes. The proposed RSS is a step further by enlarging the subnet selection from two subnets to the whole subnet population. Moreover, RSS uses random sampling which enables each individual subnet to have the same chance to be selected for training. The results show there is benefit to sampling and training more subnets as RSS produces a better performing subnet population than the **Alternating** scheme and the other methods.

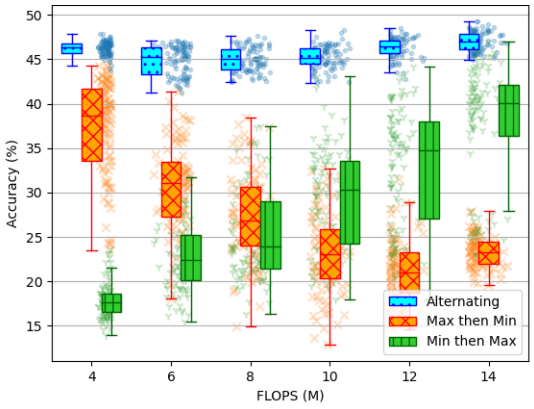


Figure 7. Resulting population performance from training the same two largest and smallest subnets in differing sequences, showing that training sequence does matter.

7. Conclusion

This work revisited the interference effect in an OFA based subnet population during training. Interference was believed to be the constraining factor in achieving good performance. To limit interference, OFA proposed progressive shrinking as a novel training method. However, we showed that interference is not the only constraining factor. Additionally, we must consider the subnet sampling bias of our selection scheme during training. Subnet sampling bias states that subnets which receive the most training will perform the best. We drew this conclusion by first connecting the subnet sampling bias to the data imbalance problem in general classification training. We then demonstrate this by proposing a simple-yet-effective training method called Random Subnet Sampling (RSS), which only mitigates the sampling bias, not interference. By mitigating only the bias, RSS trains a better performing subnet population than progressive shrinking on four small-to-medium datasets while also being 1.9 times faster. Furthering our experiments provided insight into both the interference and bias problems. We found interference to be more significant when combing the gradients of multiple sampled subnets. Additionally, we found the impact of bias depends on the subnet training sequence. In the future, we plan to test our hypothesis in large computer vision datasets such as the ImageNet dataset.

Acknowledgement

This research was partly supported by Sentient Vision Systems. Sentient Vision Systems is one of the leading Australian developers of computer vision and artificial intelligence software solutions for defence and civilian applications.

References

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations*, volume abs/1611.02167, 2017. 1, 3

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and Simplifying One-Shot Architecture Search. In *International Conference on Machine Learning*, pages 550–559. PMLR, July 2018. ISSN: 2640-3498. 1, 3

[3] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Smash: One-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*, volume abs/1708.05344, 2018. 1, 3

[4] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. 1, 3, 4, 5

[5] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019. 1, 3

[6] Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*, volume abs/1511.05641, 2016. 3

[7] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12239–12248, October 2021. 3, 4

[8] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Péter Vajda, and Joseph E. Gonzalez. Fbnetv3: Joint architecture-recipe search using predictor pretraining. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16271–16280, 2021. 1, 3

[9] Jianxiang Dong. Focal Loss Improves the Model Performance on Multi-Label Image Classifications with Imbalanced Data. *Proceedings of the 2nd International Conference on Industrial Control Network And System Engineering Research*, 2020. 3

[10] Andrew Estabrooks. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, pages 18–36, 2004. 3

[11] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020. 1, 3

[12] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, volume abs/1609.09106, 2017. 3

[13] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1135–1143, Cambridge, MA, USA, 2015. MIT Press. 3

[14] Andrew G. Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019. 5

[15] Shou-Yong Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Dsnas: Direct neural architecture search without parameter retraining. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12081–12089, 2020. 1, 3

[16] Sian-Yao Huang and Wei-Ta Chu. Ponas: Progressive one-shot neural architecture search for very efficient deployment. *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2021. 3

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 2, 5

[18] Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. In *AIME*, 2001. 3

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[20] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010. 2, 5

[21] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:318–327, 2020. 3

[22] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Loddon Yuille, Jonathan Huang, and Kevin P. Murphy. Progressive neural architecture search. In *ECCV*, 2018. 3

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. 1, 3

[24] Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. In *International Conference on Learning Representations Workshop*, volume abs/1511.06343, 2015. 3

[25] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient Neural Architecture Search via Parameter Sharing. *arXiv:1802.03268 [cs, stat]*, Feb. 2018. arXiv: 1802.03268. 3

[26] 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 *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2902–2911. JMLR.org, 2017. 1, 3

[27] Manas Sahni, Shreya Varshini, Alind Khare, and Alexey Tumanov. CompOFA: Compound once-for-all networks for faster multi-platform deployment. In *Proc. of the 9th International Conference on Learning Representations*, ICLR '21, May 2021. 1, 3

[28] Abhinav Shrivastava, Abhinav Kumar Gupta, and Ross B. Girshick. Training region-based object detectors with online hard example mining. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 761–769, 2016. 3

[29] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, and Francesc Moreno-Noguer. Fracking Deep Convolutional Image Descriptors. *arXiv:1412.6537 [cs]*,

Feb. 2015. arXiv: 1412.6537. 3

[30] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6414–6423, 2021. 3

[31] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2015. 3

[32] Gary M. Weiss and Foster J. Provost. The effect of class distribution on classifier learning. 2001. 3, 4

[33] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Péter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10726–10734, 2019. 1, 3

[34] Xin Xia, Xuefeng Xiao, Xing Wang, and Min Zheng. Progressive automatic design of search space for one-shot neural architecture search. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3525–3534, 2022. 1, 3

[35] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. 2, 5

[36] Jin Xu, Xu Tan, Kaitao Song, Renqian Luo, Yichong Leng, Tao Qin, Tie-Yan Liu, and Jian Li. Analyzing and Mitigating Interference in Neural Architecture Search. *arXiv:2108.12821 [cs]*, Aug. 2021. arXiv: 2108.12821. 2, 3, 7

[37] Xiaowei Xu, Yukun Ding, Sharon Hu, Michael Thaddeus Niemier, Jason Cong, Yu Hu, and Yiyu Shi. Scaling for edge inference of deep neural networks. *Nature Electronics*, 1:216–222, 2018. 1

[38] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, and Quoc V. Le. Bignas: Scaling up neural architecture search with big single-stage models. In *ECCV*, 2020. 3

[39] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2019. 3

[40] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, volume abs/1611.01578, 2017. 1, 3