

IMB-NAS: Neural Architecture Search for Imbalanced Datasets

Rahul Duggal¹, Sheng-Yun Peng¹, Hao Zhou, Duen Horng Chau¹

¹ Georgia Institute of Technology
 {rahulduggal, shengyun, polo}@gatech.edu, zhhoper@gmail.com

Abstract

Class imbalance is a ubiquitous phenomenon occurring in real world data distributions. To overcome its detrimental effect on training accurate classifiers, existing work follows three major directions: class re-balancing, information transfer, and representation learning. In this paper, we propose a new and complementary direction for improving performance on long tailed datasets—optimising the *backbone architecture* through neural architecture search (NAS). We find that an architecture’s accuracy obtained on a balanced dataset is not indicative of good performance on imbalanced ones. This poses the need for a full NAS run on long tailed datasets which can quickly become prohibitively compute intensive. To alleviate this compute burden, we aim to efficiently adapt a NAS super-network from a balanced source dataset to an imbalanced target one. Among several adaptation strategies, we find that the most effective one is to retrain the linear classification head with reweighted loss, while freezing the backbone NAS super-network trained on a balanced source dataset. We perform extensive experiments on multiple datasets and provide concrete insights to optimise architectures for long tailed datasets.

1 Introduction

The natural world follows a long tail data distribution wherein a small percentage of classes constitute the bulk of data samples, while a small percentage of data is distributed across numerous minority classes. Training accurate classifiers on imbalanced datasets has been an active research direction since the early 90s. Much of prior work (Kang et al. 2019; Zhou et al. 2020; Duggal et al. 2021a) centers on improving the performance (measured via accuracy) of a fixed backbone architecture such as ResNet-32. In this work, we take a complementary direction and aim to optimise the backbone architecture via neural architecture search. Indeed this is an important direction since prevalent practices demand that neural architectures be optimised to fit the size/latency constraints of tiny edge devices.

To optimise the backbone architecture, we rely on the recent work from Neural Architecture Search (NAS) (Guo et al. 2020) that optimises a neural network’s architecture primarily on datasets that are *balanced* across classes. This workflow naturally prompts the question: is the architecture optimised on a class balanced dataset also the optimal one for imbalanced datasets? Table 1 provides evidence to

Dataset	Model	Flops	Accuracy (%)	
			bal(1×)	imbal(100×)
Cifar10	A1	410	94.6	77.3
	A2	407	94.7	74.1
Cifar100	A3	400	76.1	39.4
	A4	179	75.0	43.0

Table 1: **Motivation.** We sample four architectures A1-A4 from the DARTS search space and train them on balanced (i.e. 1×) and imbalanced versions (i.e. 100×) of Cifar10 and Cifar100. **(Top)** Two similarly sized architectures (A1,A2) achieve similar accuracy on balanced Cifar10, but differ by 3% in presence of 100× imbalance. **(Bottom)** The larger architecture (A3) outperforms the smaller one (A4) on balanced Cifar100, but under performs by 3.6% in the presence of 100× imbalance. This suggests that an architecture’s performance on balanced datasets is not indicative of its performance on imbalanced ones.

the contrary. The first row shows two architectures—A1,A2—sampled from the DARTS search space (Liu, Simonyan, and Yang 2018) having similar size and accuracy on balanced Cifar10, but an accuracy gap of 3% in the presence of 100× imbalance. The second row compares a larger architecture A3 outperforms a smaller one A4 on balanced Cifar100. However, in the presence of 100× imbalance, the smaller architecture outperforms the larger one by more than 3%. These results and more in Sec 3.2, indicate that the optimal architecture on a balanced dataset may not be the optimal one for imbalanced datasets. This means each target imbalanced dataset requires its own NAS procedure to obtain the optimal architecture.

Running a NAS procedure for each target dataset is computationally expensive and quickly becomes intractable in the presence of multiple target datasets. To overcome the compute burden of running NAS from scratch, we formalize the task of architectural rank adaptation from balanced to imbalanced datasets. Towards this task, Section 3.4 describes two intuitive rank adaptation procedures that either fine-tune the classifier only, or together with the backbone. Our comprehensive experiments reveal the key insight that the adaptation procedure is most affected by the linear clas-

sification head trained on top of the backbone. Armed with this insight, we propose to re-use a NAS super-net backbone trained on balanced data and re-train only the classification head to efficiently adapt a pre-trained NAS super-net for imbalanced data. This is extremely efficient since it involves training only a linear layer on top of the pre-trained super-network.

Overall, our contributions in this work are:

1. **New insight.** We show that architectural rankings transfers poorly from balanced to imbalanced datasets.
2. **Novel task.** We construct the novel task to efficient adapt a NAS super-network from balanced to imbalanced datasets.
3. **Novel solution.** We propose a simple and efficient solution—retraining the classifier head while freezing the backbone—to efficiently adapt a NAS super-network from balanced to imbalanced datasets.

2 Related Works

We cover relevant work from three related areas.

2.1 Overcoming long tail class imbalance

Prior work on tackling long tail imbalance can be divided into three broad areas (see survey (Zhang et al. 2021b)): class-rebalancing that includes data re-sampling (SMOTE (Chawla et al. 2002), ADASYN (He et al. 2008)), loss re-weighting (Kang et al. 2019; Cui et al. 2019; Duggal et al. 2020, 2021a), logit adjustment (Menon et al. 2020; Tian et al. 2020; Zhang et al. 2021a); Information augmentation that includes transfer learning (Wang, Ramanan, and Hebert 2017; Yin et al. 2019), data augmentation (Chu et al. 2020); and module improvement that encompasses methods in representation learning (Liu et al. 2019), classifier design (Wu et al. 2020), decoupled training (Kang et al. 2019) and ensembling (Zhou et al. 2020). Different from all of the existing works, our work explores a new direction of performance improvement on long tail datasets—that via optimizing the backbone architecture. This complements existing approaches and can work in tandem to further boost accuracy and efficiency on imbalanced datasets.

2.2 Neural architecture search

Prior work on architecture search can be categorized in improving its three main pillars (see survey (Elsken, Metzger, and Hutter 2019))—Search space design with the idea of incorporating a large diversity of architectures. Popular spaces include cell based spaces such as NASNets (Zoph et al. 2018), and recent spaces from the ShuffleNet (Zhang et al. 2018) and MobileNet (Howard et al. 2017) model families. The second pillar constitutes search strategy design to efficiently locate performant architectures from the search space. Popular strategies involve reinforcement learning (Baker et al. 2016; Zoph et al. 2018), evolutionary algorithms (Real et al. 2017; Duggal et al. 2021b) or gradient descent on continuous relaxations of the search space (Liu, Simonyan, and Yang 2018). The third pillar constitutes performance estimation strategies (Baker et al. 2017; Falkner,

Klein, and Hutter 2018) with the goal of cheaply estimating the goodness (in terms of accuracy or efficiency) of an architecture. All of the above works search optimal architectures on datasets that are fully balanced across all classes. Our experiments however show that the set of optimal architectures differ significantly from balanced to imbalanced datasets. This calls for developing new NAS methods or efficient adaptation strategies (e.g. this work) to search for optimal architectures on real world, imbalanced datasets.

2.3 Architecture transfer

We summarize prior work on evaluating robustness of architectures to distributional shifts in the training dataset. Neural Architecture Transfer (Lu et al. 2021) explore architectural transferability from large-scale to small-scale fine grained datasets. However, there are two limitations—the source and target datasets considered in this work are balanced across all classes and additionally this work assumes all target datasets are known apriori which is infeasible in many industry use-cases. NASTransfer (Panda et al. 2021) consider transferability between large-scale imbalanced datasets including ImageNet-22k which is a highly imbalanced dataset. Their approach is practically useful for very large datasets (e.g. ImageNet-22k) for whom direct search is prohibitive, however when it is feasible (e.g. on ImageNet) direct search typically leads to better architectures than proxy search. Differing from these, our work advocates to directly adapt a super-network pre-trained on fully balanced datasets (instead of proxies) to imbalanced ones. A key feature of such adaptation is efficiency—the compute required for such an adaptation needs to be much lesser than that for repeating the search on the target dataset.

3 Methodology

3.1 Notation

Assume $\mathcal{D} = \{x_i, y_i\}$ denotes the training dataset of images where y_i is the label for image x_i . Let n_j specify the number of training images in class j . After sorting the classes by cardinality in decreasing order, the long tail assumption specifies that if $i < j$, then $n_i \geq n_j$ and $n_1 \gg n_c$. We use ϕ to denote a deep neural network that is composed of a backbone $\phi(a, w_a)$ with architecture a , weights w_a and a linear classifier $\phi(w_c)$. The model ϕ is trained using a training loss and loss re-weighting strategy. On balanced datasets, we use the cross entropy loss (denoted as CE) to train a neural network. For imbalanced datasets we additionally incorporate the effective re-weighting strategy (Cui et al. 2019) that reweights samples from class j with $\frac{1-\beta}{1-\beta^{n_j}}$ where β is a hyperparameter. Following previous works (Cao et al. 2019; Duggal et al. 2020), the re-weighting strategy is applied after a delay of few training epochs which is denoted using the shorthand DRW.

3.2 Architecture ranking transfer: A motivating experiment

We study the impact of backbone architecture on imbalanced datasets using the following experiment. We construct an architecture search space \mathcal{A} by sampling all 149 Mega Flops

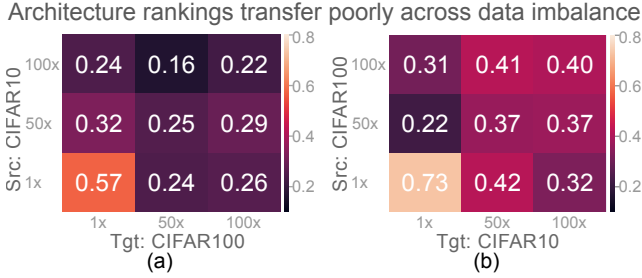


Figure 1: **Evaluating architectural transferability.** We train all 149 Mflop architectures from NATS-Bench on Cifar10, Cifar100 with $1\times, 50\times, 100\times$ imbalance and compute kendall tau correlation between the rank orderings on all datasets. We observe high correlation (bottom left cells) when both $\mathcal{D}_s, \mathcal{D}_t$ are balanced, and low correlation otherwise. This means that architectural rankings transfer poorly across data imbalance.

architectures from the NATS-bench search space (Dong et al. 2021). Overall \mathcal{A} contains 135 architectures with exactly the same learning capacity (or Flops), but different architectural patterns (e.g. kernel sizes, layer connectivity). The architectures in \mathcal{A} are trained on the source and target datasets $\mathcal{D}_s, \mathcal{D}_t$ using loss function CE on balanced datasets, and the re-weighted loss function CE+DRW on imbalanced ones. Following this, the architectures are ranked based on validation accuracy and the kendall Tau metric is computed between the rank orderings obtained on \mathcal{D}_s and \mathcal{D}_t . A high correlation means similar architectural rankings on both datasets, while a low correlation implies widely different rankings.

Figure 1 presents the outcomes on two scenarios: (1) \mathcal{D}_s is Cifar10 at three levels of imbalance ($1\times, 50\times, 100\times$) and \mathcal{D}_t is Cifar100 at the same imbalance levels; and (2) the opposite direction. There are two major observations—First, the high correlation in the bottom left square indicates that the architectural rankings transfer quite well across balanced datasets. Second, the low correlation for all other cells indicates low transferability across imbalanced datasets. This means the rank orderings on imbalanced datasets widely differs from that on balanced ones.

To avoid the compute burden of performing a NAS run on every target imbalanced dataset, we develop efficient “adaptation” procedures to adapt a NAS super-net from balanced to imbalanced datasets. Before going into the details, in the next section we provide a brief overview of existing NAS methods.

3.3 Revisiting neural architecture search

We look at sampling based NAS methods that involve two steps. The first step involves training a super-network with backbone $\phi(a, w_a)$ and classifier $\phi(w_c)$ on a training dataset \mathcal{D} via the following minimization

$$w_{a,\mathcal{D}}^*, w_{c,\mathcal{D}}^* = \min_{w_a, w_c, a \sim \mathcal{A}} \mathbb{E} (\mathcal{L}(\phi(w_c), \phi(a, w_a); \mathcal{D})). \quad (1)$$

Here the inner expectation is performed by sampling architectures a from a search space \mathcal{A} via uniform, or attentive

sampling.

The second step involves searching the optimal architecture that maximizes validation accuracy via the following optimisation

$$a_{\mathcal{D}}^* = \max_{a \in \mathcal{A}} \text{Acc} (\phi(w_c), \phi(a, w_a); \mathcal{D}). \quad (2)$$

This maximization is typically implemented via evolutionary search or reinforcement learning. Next, we discuss efficient adaptation procedures to adapt a NAS super-net trained on a balanced dataset onto an imbalanced one.

3.4 Rank adaptation procedures

Given source and target datasets $\mathcal{D}_s, \mathcal{D}_t$, we first train a super-network on \mathcal{D}_s by solving the following optimisation

$$w_{a,\mathcal{D}_s}^*, w_{c,\mathcal{D}_s}^* = \min_{w_a, w_c, a \sim \mathcal{A}} \mathbb{E} (\mathcal{L}(\phi(w_c), \phi(a, w_a); \mathcal{D}_s)). \quad (3)$$

Our goal then is to adapt the optimal super-net weights $w_{a,\mathcal{D}_s}^*, w_{c,\mathcal{D}_s}^*$ found on \mathcal{D}_s to the target dataset \mathcal{D}_t which suffers from class imbalance. The most efficient adaptation procedure involves freezing the backbone, while adapting only the linear classifier on \mathcal{D}_t by minimizing the re-weighted loss \mathcal{L}_{RW}

$$w_{c,\mathcal{D}_t}^* = \min_{w_c, a \sim \mathcal{A}} \mathbb{E} (\mathcal{L}_{RW}(\phi(w_c), \phi(a, w_{a,\mathcal{D}_s}^*); \mathcal{D}_t)). \quad (4)$$

The resulting super-network contains backbone weights w_{a,\mathcal{D}_s}^* trained on \mathcal{D}_s and classifier weights w_{c,\mathcal{D}_s}^* trained on \mathcal{D}_t . Solving the above optimisation is extremely efficient since most of the network is frozen while only the classifier is trained. On the other hand, one could also adapt the backbone by fine-tuning on the target dataset. This is achieved by minimizing the delayed re-weighted loss \mathcal{L}_{DRW}

$$w_{a,\mathcal{D}_t}^{**}, w_{c,\mathcal{D}_t}^* = \min_{w_a, w_c, a \sim \mathcal{A}} \mathbb{E} (\mathcal{L}_{DRW}(\phi(w_c), \phi(a, w_{a,\mathcal{D}_s}^*); \mathcal{D}_t)). \quad (5)$$

Here, the double star on w_{a,\mathcal{D}_t}^{**} indicates the weights were obtained via fine-tuning w_{a,\mathcal{D}_s}^* using one tenth of the original learning rate and one third the number of original training epochs. Also, recall that the delayed re-weighted loss \mathcal{L}_{DRW} is nothing but the unweighted loss \mathcal{L} in the first few epochs and the re-weighted loss \mathcal{L}_{RW} subsequently. Note that our second adaptation procedure is more compute intensive since the backbone is also adapted, but still much less intensive than running the full search on the target dataset.

Our final and most compute intensive procedure involves directly searching on the target dataset via \mathcal{L}_{DRW} . This is achieved via the following minimization

$$w_{a,\mathcal{D}_t}^*, w_{c,\mathcal{D}_t}^* = \min_{w_a, w_c, a \sim \mathcal{A}} \mathbb{E} (\mathcal{L}_{DRW}(\phi(w_c), \phi(a, w_a); \mathcal{D}_t)). \quad (6)$$

The three adaptation procedures and their associated compute costs are summarized in Table 2.

4 Experiments

We begin this section by answering which rank adaptation procedure works best, both in terms of efficiency of the procedure and the accuracy of the resulting networks. We then perform an extensive ablation study to uncover the effect of different design choices.

Adj	Eqn	Description
P0	(3)	No adaptation.
P1	(4)	Freeze backbone, retrain classifier on \mathcal{D}_t .
P2	(5)	Finetune backbone and retrain classifier on \mathcal{D}_t .
P3	(6)	Re-train backbone and classifier on \mathcal{D}_t .

Table 2: Summarizing rank adaptation procedures.

4.1 Implementation details

We implement our methods using Pytorch on a system containing 8 V100 GPUs. Other details are as follows:

Datasets. We construct imbalanced versions of Cifar-10 and Cifar-100 by sub-sampling from their original training splits (Cui et al. 2019). The c^{th} class in the resulting datasets contains $n_c = n\mu^c$ examples where n is the original cardinality of class c , and $\mu \in [0, 1]$. We select μ such that the imbalance ratio—which is defined as the ratio between the number of examples in the largest and smallest class—is $50\times$ to $1000\times$.

Sub-network training strategies. We train a network on balanced Cifar-10/100 for 200 epochs with an initial learning rate of 0.1 decayed by 0.01 at epochs 160 and 180 using the cross entropy loss. On imbalanced versions, we introduce effective re-weighting (Cui et al. 2019) at epoch 160 and refer to this strategy as delayed re-weighting or DRW-160 (Cao et al. 2019).

Neural Architecture Search We train a super-network for 600 epochs with an initial learning rate of 0.1, decayed by 0.01 at epochs 400 and 500. On imbalanced datasets, re-weighting is applied at epoch 400. For searching the best subnet, we follow (Guo et al. 2020) and use an evolutionary search with 20 generations, population of 50, crossover number 25, mutation number 25, mutate probability 0.1 and top-k of 10.

Adaptation Strategies To adapt a super-network, we fine-tune it for 200 epochs with an initial LR of 0.01, decayed by 0.01 at epoch 100. In case of procedure P1, we introduce re-weighting at epoch 1. For P2, we delay the re-weighting to epoch 100. For P3, we follow the NAS strategy detailed above.

4.2 Baseline and Paragon for IMB-NAS

Given a NAS super-network trained on a source dataset \mathcal{D}_s , our goal is to efficiently adapt it to the target dataset \mathcal{D}_t following which, the best sub-net is searched in the adapted super-net. Table 3a illustrates the results for the case when \mathcal{D}_s is Cifar10, and \mathcal{D}_t is Cifar100 with varying levels of imbalance. The first row (i.e. P0) refers to the case when the best sub-nets obtained on \mathcal{D}_s are re-trained on \mathcal{D}_t . This serves as our lower bound or *baseline*. The last row (i.e. P3) refers to the case when the NAS super-net is trained on \mathcal{D}_t . This serves as the upper bound or the *paragon* of accuracy. Our two adaptation procedures (P1, P2) in the middle rows are highlighted yellow when they outperform the baseline, and the better among the two is bolded.

		Imbalance Ratio			
		50×	100×	200×	400×
baseline	P0	45.80	40.83	36.30	32.80
	P1	45.06	41.93	36.76	33.70
	P2	44.86	41.86	36.70	33.46
paragon	P3	45.93	41.53	37.03	33.40

(a) Cifar10-1× \rightarrow Cifar100- $\{50, 100, 200, 400\}\times$

		Imbalance Ratio			
		100×	200×	400×	800×
baseline	P0	75.96	68.96	63.26	56.90
	P1	75.93	69.70	63.80	58.23
	P2	75.86	69.26	63.70	58.03
paragon	P3	76.03	70.23	63.96	57.70

(b) Cifar100-1× \rightarrow Cifar10- $\{100, 200, 400, 800\}\times$

Table 3: **Comparing rank adaptation strategies.** Given a NAS super-net trained on \mathcal{D}_s , we adapt it to \mathcal{D}_t and search the optimal sub-nets. These are retrained from scratch on \mathcal{D}_t and the average validation accuracy is presented. Note that sub-nets obtained via P1/P2 outperform P0 for high imbalance ratios (shaded yellow) and typically P1 outperforms P2 (the winner is bolded). Results averaged over three seeds.

Observe from Tables 3a,3b that both adaptation procedures comprehensively outperform the baseline at higher levels of imbalance. This means that the architectures searched on \mathcal{D}_s can no longer be assumed as the optimal ones on imbalanced target datasets. Interestingly, between P1 and P2, we find that P1 consistently outperforms P2. This is surprising since P2 also adapts the NAS backbone on the target data whereas P1 re-uses the backbone from the source dataset. We hypothesize this occurs because, class imbalance is much larger an issue for searching the NAS backbone than the domain difference between Cifar10 and Cifar100.

Overall, we find that P1 and P2 achieve very close accuracy to the paragon (P3) while avoiding much of the compute burden of P3 as illustrated in the next section.

4.3 Dissecting the performance adaptation

In this section, we analyze different aspects of procedures P1-P3 by applying them to adapt a NAS super-net pre-trained on Cifar10-1x onto Cifar100-100x.

Comparison on training cost. We measure the wall-clock training time on a single V100 GPU as a proxy for training cost. The amortized training cost over three runs is presented in Fig 2. It takes P1 2000 seconds to adapt a NAS super-network from cifar10-1× to cifar100-100×. In comparison P2 consumes 2×, and P3 consumes 5× more time. These results demonstrate that not only P1 can successfully adapt a super-net to improve accuracy, it is also very efficient.

Impact of fine-tuning the backbone with P2. In procedure P2, we adapt the NAS backbone via fine-tuning on the target

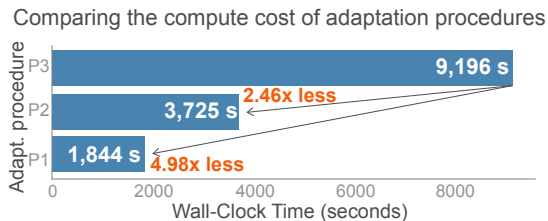


Figure 2: **Comparing the compute cost** of adapting a NAS super-net trained on Cifar10-1 \times onto Cifar100-100 \times . The y-axis plots the wall clock time spent on a single V100 GPU. Observe that P2 and P3 consume 2 \times and 5 \times the cost of P1. Results averaged over three seeds.

Proc	Epochs	Imbalance Ratio		
		100 \times	200 \times	400 \times
-	-	40.83	36.3	32.80
P0	-	41.93	36.76	33.70
P1	-	40.06	35.46	32.56
P2	50	41.43	37.26	33.00
	100	41.40	36.63	32.86
	150	41.86	36.70	33.46
P3	-	41.53	37.03	33.4

Table 4: **Ablating on** the number of backbone fine-tuning epochs with P2 while adapting from Cifar10-1 \times to Cifar100- $\{100, 200, 400\}\times$. Coinciding the freezing of the backbone with the loss re-weighting at epoch 100 typically outperforms the baseline. Generally more fine-tuning epochs are better. Results averaged over three seeds.

dataset \mathcal{D}_t . One may wonder, can the backbone be frozen after the loss re-weighting is applied? The intuition being that re-weighting mainly helps adapt the classification boundary while negatively affecting the representation learned by the backbone (Kang et al. 2019). To answer this, Table 4 presents an ablation on the number of epochs spent on fine-tuning the NAS backbone with P2. Observe that too few fine-tuning epochs (e.g. 50) leads to low sub-net accuracy. At the other end, fine-tuning for 100 epochs is sufficient to improve the sub-net accuracy beyond the paragon (P3). This means that one could further lower the compute burden of P2 by freezing the backbone once loss re-weighting is applied at epoch 100.

Training the NAS super-net with loss re-weighting. We observe that loss re-weighting generally results in improved super-net accuracy on imbalanced datasets. Does this mean the resulting sub-nets are better than the ones obtained from a super-net trained without loss re-weighting? We answer this question we train super-nets on Cifar100-100 \times with and without re-weighting. Then we search and train the best sub-nets which are presented in Table. 5. We find that there is no clear winner among the two NAS training approaches.

Dissecting the overall accuracy improvement. To analyze which classes contribute to an increase accuracy, Table. 6

Train Loss	Imbalance Ratio		
	100 \times	200 \times	400 \times
CE	41.36	37.5	33.9
CE-DRW	41.53	37.0	33.4

Table 5: **Ablating on the loss** used to train the NAS super-net on Cifar100-100 \times . Results presented are the validation accuracy of optimal sub-networks searched from corresponding super-networks. It is inconclusive if training the NAS super-net with re-weighted loss (CE-DRW) induces better sub-networks. Results averaged over three seeds.

Adj	Imbalance Ratio							
	100 \times				400 \times			
	High	Med	Low	All	High	Med	Low	All
P0	64.1	40.4	14.1	40.8	65.0	39.1	10.1	32.8
P1	65.2	41.6	15.0	41.9	66.3	41.0	10.2	33.7
P2	65.2	40.9	15.7	41.8	66.2	40.3	10.2	33.4
P3	65.0	41.5	14.1	41.5	66.2	39.5	10.5	33.4

Table 6: **Dissecting the overall accuracy** on Cifar100- $\{100, 400\}\times$ into the accuracy on classes containing many (i.e. > 100), medium (i.e. between 20-100) and few (i.e. < 20) examples per class. Sub-networks obtained via P1 and P2 outperform the baseline (P0) for all class categories (shaded yellow). Results averaged over three seeds.

dissects the overall accuracy (denoted by column “All”) into the accuracy obtained on classes containing Many (i.e. > 100), Medium (i.e. between 20-100) and Few (i.e. < 20) examples per class. For both 100 \times and 400 \times levels of imbalance, the architectures obtained via P1 and P2 outperform those obtained by P0 for all class categories. This means that indeed the architectures obtained via P1,P2 are able to learn better representations.

5 Conclusion

This work aims to improve performance on class imbalanced datasets by optimising the backbone architecture. Towards this goal, we discover that an architecture’s performance on balanced datasets is not indicative of its performance on imbalanced ones. This observation suggests re-running NAS on each target dataset. To overcome the prohibitive compute burden or re-running NAS, we propose to adapt a NAS super-net trained on balanced datasets onto imbalanced ones. We develop multiple adaptation procedures and find that re-training the linear classification head while freezing the NAS super-net backbone outperforms other adaptation strategies both in terms of efficiency of the adaptation and the accuracy of the resulting sub-networks.

References

Baker, B.; Gupta, O.; Naik, N.; and Raskar, R. 2016. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*.

- Baker, B.; Gupta, O.; Raskar, R.; and Naik, N. 2017. Accelerating neural architecture search using performance prediction. *arXiv preprint arXiv:1705.10823*.
- Cao, K.; Wei, C.; Gaidon, A.; Arechiga, N.; and Ma, T. 2019. Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss. In *Advances in Neural Information Processing Systems*.
- Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357.
- Chu, P.; Bian, X.; Liu, S.; and Ling, H. 2020. Feature space augmentation for long-tailed data. In *European Conference on Computer Vision*, 694–710. Springer.
- Cui, Y.; Jia, M.; Lin, T.-Y.; Song, Y.; and Belongie, S. 2019. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9268–9277.
- Dong, X.; Liu, L.; Musial, K.; and Gabrys, B. 2021. Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE transactions on pattern analysis and machine intelligence*.
- Duggal, R.; Freitas, S.; Dhamnani, S.; Chau, D. H.; and Sun, J. 2020. Elf: An early-exiting framework for long-tailed classification. *arXiv preprint arXiv:2006.11979*.
- Duggal, R.; Freitas, S.; Dhamnani, S.; Chau, D. H.; and Sun, J. 2021a. HAR: Hardness Aware Reweighting for Imbalanced Datasets. In *2021 IEEE International Conference on Big Data (Big Data)*, 735–745. IEEE.
- Duggal, R.; Zhou, H.; Yang, S.; Xiong, Y.; Xia, W.; Tu, Z.; and Soatto, S. 2021b. Compatibility-aware heterogeneous visual search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10723–10732.
- Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1): 1997–2017.
- Falkner, S.; Klein, A.; and Hutter, F. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, 1437–1446. PMLR.
- Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; and Sun, J. 2020. Single path one-shot neural architecture search with uniform sampling. In *European conference on computer vision*, 544–560. Springer.
- He, H.; Bai, Y.; Garcia, E. A.; and Li, S. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, 1322–1328. IEEE.
- Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- Kang, B.; Xie, S.; Rohrbach, M.; Yan, Z.; Gordo, A.; Feng, J.; and Kalantidis, Y. 2019. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*.
- Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.
- Liu, Z.; Miao, Z.; Zhan, X.; Wang, J.; Gong, B.; and Yu, S. X. 2019. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2537–2546.
- Lu, Z.; Sreekumar, G.; Goodman, E.; Banzhaf, W.; Deb, K.; and Boddeti, V. N. 2021. Neural architecture transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9): 2971–2989.
- Menon, A. K.; Jayasumana, S.; Rawat, A. S.; Jain, H.; Veit, A.; and Kumar, S. 2020. Long-tail learning via logit adjustment. *arXiv preprint arXiv:2007.07314*.
- Panda, R.; Merler, M.; Jaiswal, M. S.; Wu, H.; Ramakrishnan, K.; Finkler, U.; Chen, C.-F. R.; Cho, M.; Feris, R.; Kung, D.; et al. 2021. Nastransfer: Analyzing architecture transferability in large scale neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 9294–9302.
- Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y. L.; Tan, J.; Le, Q. V.; and Kurakin, A. 2017. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, 2902–2911. PMLR.
- Tian, J.; Liu, Y.-C.; Glaser, N.; Hsu, Y.-C.; and Kira, Z. 2020. Posterior re-calibration for imbalanced datasets. *Advances in Neural Information Processing Systems*, 33: 8101–8113.
- Wang, Y.-X.; Ramanan, D.; and Hebert, M. 2017. Learning to model the tail. *Advances in neural information processing systems*, 30.
- Wu, T.-Y.; Morgado, P.; Wang, P.; Ho, C.-H.; and Vasconcelos, N. 2020. Solving long-tailed recognition with deep realistic taxonomic classifier. In *European Conference on Computer Vision*, 171–189. Springer.
- Yin, X.; Yu, X.; Sohn, K.; Liu, X.; and Chandraker, M. 2019. Feature transfer learning for face recognition with under-represented data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5704–5713.
- Zhang, S.; Li, Z.; Yan, S.; He, X.; and Sun, J. 2021a. Distribution alignment: A unified framework for long-tail visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2361–2370.
- Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6848–6856.
- Zhang, Y.; Kang, B.; Hooi, B.; Yan, S.; and Feng, J. 2021b. Deep long-tailed learning: A survey. *arXiv preprint arXiv:2110.04596*.
- Zhou, B.; Cui, Q.; Wei, X.-S.; and Chen, Z.-M. 2020. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9719–9728.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697–8710.