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-tailed 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 the contrary. The first row shows two architectures—one, A1, 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 on (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 it’s performance on imbalanced ones.

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-

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Flops</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bal(1×)</td>
<td>imbal(100×)</td>
<td></td>
</tr>
<tr>
<td>Cifar10</td>
<td>A1</td>
<td>410</td>
<td>94.6</td>
</tr>
<tr>
<td></td>
<td>A2</td>
<td>407</td>
<td>94.7</td>
</tr>
<tr>
<td>Cifar100</td>
<td>A3</td>
<td>400</td>
<td>76.1</td>
</tr>
<tr>
<td></td>
<td>A4</td>
<td>179</td>
<td>75.0</td>
</tr>
</tbody>
</table>

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 on (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 it’s performance on imbalanced ones.
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 transfer poorly from balanced to imbalanced datasets.
2. Novel task. We construct the novel task to efficiently adapt a NAS super-network from balanced to imbalanced datasets.
3. Novel solution. We propose a simple and efficient solution—re-training 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]):

- Information augmentation that includes transfer learning (Wang, Ramanan, and Hebert 2017), 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 ensemble (Zhou et al. 2020). Different from all of the existing works, our work explores a new direction of performance improvement on long tail datasets via optimising 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 as improving its three main pillars (see survey [Elsken, Metzen, 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 a priori 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 $D = \{x_i, y_i\}$ 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 $n_i \leq n_j$ and $n_1 >> n_C$. We use $\phi$ 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 $\phi$ 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 $\frac{1}{\beta_{C}}$ where $\beta$ is a hyperparameter. Following previous works (Cao et al. 2019, Duggal et al. 2020), the re-weighting strategy is applied after a delay of a 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
Architecture rankings transfer poorly across data imbalance

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 $D_s, D_t$ 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 $A$ contains 135 architectures with exactly the same learning capacity (or Flops), but different architectural patterns (e.g. kernel sizes, layer connectivity). The architectures in $A$ are trained on the source and target datasets $D_s, D_t$ using loss function $CE$ on balanced datasets, and the re-weighted loss function $CE+DRW$ on imbalanced datasets. Following this, the architectures are ranked based on validation accuracy and the kendall Tau metric is computed between the rank orderings obtained on $D_s$ and $D_t$. 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) $D_s$ is CIFAR10 at three levels of imbalance ($1 \times, 50 \times, 100 \times$) and $D_t$ 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 $D$ via the following minimization

$$w_{a, D}, w_{c, D} = \min_{w_a, w_c} \mathbb{E} \left( L(\phi(w_c), \phi(a, w_a); D) \right).$$

Here the inner expectation is performed by sampling architectures $a$ from a search space $A$ via uniform, or attentive sampling.

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

$$a^*_D = \max_{a \in A} \text{Acc} \left( \phi(w_c), \phi(a, w_a); D \right).$$

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 $D_s, D_t$, we first train a super-network on $D_s$ by solving the following optimisation

$$w_{a, D_s}, w_{c, D_s} = \min_{w_a, w_c} \mathbb{E} \left( L(\phi(w_c), \phi(a, w_a); D_s) \right).$$

Our goal then is to adapt the optimal super-net weights $w_{a, D_s}, w_{c, D_s}$ found on $D_s$ to the target dataset $D_t$ which suffers from class imbalance. The most efficient adaptation procedure involves freezing the backbone, while adapting only the linear classifier on $D_t$ by minimizing the re-weighted loss $L_{DRW}$

$$w_{c, D_t} = \min_{w_c} \mathbb{E} \left( L_{DRW}(\phi(w_c), \phi(a, w_{a, D_s}); D_t) \right).$$

The resulting super-network contains backbone weights $w_{a, D_s}$ trained on $D_s$ and classifier weights $w_{c, D_t}$ trained on $D_t$. 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 $L_{DRW}$

$$w_{a, D_t}, w_{c, D_t} = \min_{w_a, w_c} \mathbb{E} \left( L_{DRW}(\phi(w_c), \phi(a, w_{a, D_s}); D_t) \right).$$

Here, the double star on $w_{a, D_t}$ indicates the weights were obtained via fine-tuning $w_{a, D_t}$ 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 $L_{DRW}$ is nothing but the unweighted loss $L$ in the first few epochs and the re-weighted loss $L_{RW}$ 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 $L_{DRW}$. This is achieved via the following minimization

$$w_{a, D_t}, w_{c, D_t} = \min_{w_a, w_c} \mathbb{E} \left( L_{DRW}(\phi(w_c), \phi(a, w_a); D_t) \right).$$

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.
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 $\mu$ 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 epoch 100. For searching the best sub-net, 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 $D_s$, our goal is to efficiently adapt it to the target dataset $D_t$ following which, the best sub-net is searched in the adapted super-net. Table 2 illustrates the results for the case when $D_s$ is Cifar10, and $D_t$ is Cifar100 with varying levels of imbalance. The first row (i.e. P0) refers to the case when the best sub-nets obtained on $D_s$ are re-trained on $D_t$. 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 $D_t$. 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.

<table>
<thead>
<tr>
<th>Adj</th>
<th>Eqn Description</th>
<th>Imbalance Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>(3) No adaptation.</td>
<td>50x 100x 200x 400x</td>
</tr>
<tr>
<td>P1</td>
<td>(4) Freeze backbone, retrain classifier on $D_t$.</td>
<td>50x 100x 200x 400x</td>
</tr>
<tr>
<td>P2</td>
<td>(5) Finetune backbone and retrain classifier on $D_t$.</td>
<td>50x 100x 200x 400x</td>
</tr>
<tr>
<td>P3</td>
<td>(6) Re-train backbone and classifier on $D_t$.</td>
<td>50x 100x 200x 400x</td>
</tr>
</tbody>
</table>

Table 2: Summarizing rank adaptation procedures.
Comparing the compute cost of adaptation procedures

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

<table>
<thead>
<tr>
<th>Adapt. procedure</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wall-Clock Time (seconds)</td>
<td>3,725</td>
<td>4.98x less</td>
<td>9,196</td>
</tr>
<tr>
<td>0</td>
<td>2000</td>
<td>6000</td>
<td>8000</td>
</tr>
<tr>
<td>0</td>
<td>2000</td>
<td>6000</td>
<td>8000</td>
</tr>
</tbody>
</table>

Table 4: **Ablating on** the number of backbone fine-tuning epochs with P2 while adapting from Cifar10-1× to Cifar100-100×. 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.

<table>
<thead>
<tr>
<th>Proc</th>
<th>Epochs</th>
<th>Imbalance Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100×</td>
<td>200×</td>
</tr>
<tr>
<td>P0</td>
<td>40.83</td>
<td>36.3</td>
</tr>
<tr>
<td>P1</td>
<td>41.93</td>
<td>36.76</td>
</tr>
<tr>
<td>P2</td>
<td>40.06</td>
<td>35.46</td>
</tr>
<tr>
<td>P3</td>
<td>41.43</td>
<td>36.70</td>
</tr>
</tbody>
</table>

Table 5: **Ablating on the loss** used to train the NAS super-net on Cifar100-100×. 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.

<table>
<thead>
<tr>
<th>Adj</th>
<th>Imbalance Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>100×</td>
<td>200×</td>
</tr>
<tr>
<td>P0</td>
<td>64.1</td>
</tr>
<tr>
<td>P1</td>
<td>85.2</td>
</tr>
<tr>
<td>P2</td>
<td>65.2</td>
</tr>
<tr>
<td>P3</td>
<td>65.0</td>
</tr>
</tbody>
</table>

Table 6: **Dissecting the overall accuracy** on Cifar100-\{100, 400\}× 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.

<table>
<thead>
<tr>
<th>Adj</th>
<th>Imbalance Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>100×</td>
<td>200×</td>
</tr>
<tr>
<td>P0</td>
<td>High</td>
</tr>
<tr>
<td>P1</td>
<td>64.1</td>
</tr>
<tr>
<td>P2</td>
<td>85.2</td>
</tr>
<tr>
<td>P3</td>
<td>65.0</td>
</tr>
</tbody>
</table>

Dissecting the overall accuracy improvement. To analyze which classes contribute to an increase accuracy, Table. 6 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× and 400× 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 if 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


