GRADSIGN: MODEL PERFORMANCE INFERENCE WITH THEORETICAL INSIGHTS

Anonymous authors
Paper under double-blind review

ABSTRACT

A key challenge in neural architecture search (NAS) is quickly inferring the predictive performance of a broad spectrum of networks to discover statistically accurate and computationally efficient ones. We refer to this task as model performance inference (MPI). The current practice for efficient MPI is gradient-based methods that leverage the gradients of a network at initialization to infer its performance. However, existing gradient-based methods rely only on heuristic metrics and lack the necessary theoretical foundations to consolidate their designs. We propose GradSign, an accurate, simple, and flexible metric for model performance inference with theoretical insights. The key idea behind GradSign is a quantity Ψ to analyze the optimization landscape of different networks at the granularity of individual training samples. Theoretically, we show that both the network's training and true population losses are proportionally upper bounded by Ψ under reasonable assumptions. In addition, we design GradSign, an accurate and simple approximation of Ψ using the gradients of a network evaluated at a random initialization state. Evaluation on seven NAS benchmarks across three training datasets shows that GradSign generalizes well to real-world networks and consistently outperforms state-of-the-art gradient-based methods for MPI evaluated by Spearman's ρ and Kendall's Tau. Additionally, we integrate GradSign into four existing NAS algorithms and show that the GradSign-assisted NAS algorithms outperform their vanilla counterparts by improving the accuracies of best-discovered networks by up to 0.3%, 1.1%, and 1.0% on three real-world tasks.

1 INTRODUCTION

As deep learning methods evolve, neural architectures have gotten progressively larger and more sophisticated (He et al., 2015; Ioffe & Szegedy, 2015; Krizhevsky et al., 2017; Devlin et al., 2019; Rumelhart et al., 1986; Srivastava et al., 2014; Kingma & Ba, 2017), making it increasingly challenging to design model architectures that can achieve state-of-the-art predictive performance manually. To alleviate this challenge, recent work has proposed several approaches to *automatically* discover statistically accurate and computationally efficient neural architectures. The most common approach is *neural architecture search* (NAS), which explores a comprehensive search space of potential network architectures based on compositions of basic network modules to discover architectures outperforming human-designed counterparts (Liu et al., 2018a; Zoph & Le, 2016; Pham et al., 2018). As another example, *network pruning* removes subsets of weights and activations from a network to reduce its computational cost while maintaining accuracy (Molchanov et al., 2019).

A key challenge in existing automated approaches is quickly assessing the predictive performance of a diverse set of candidate architectures to discover performant ones. We refer to this task as *model performance inference* (MPI). A straightforward approach for MPI is directly training candidate architectures on a dataset until convergence and recording the achieved training losses and validation accuracies (Frankle & Carbin, 2018; Chen et al., 2020a; Liu et al., 2018a; Zoph & Le, 2016). Though accurate, this approach is computationally prohibitive and cannot scale to large networks or datasets.

The current practice to efficient MPI is *gradient-based methods* that leverage the gradient information of a network at initialization to infer its predictive performance (Lee et al., 2018; Wang et al., 2020; Tanaka et al., 2020). Compared to directly measuring the accuracy of candidate networks on a training dataset, gradient-based methods are computationally more efficient since they only require evaluating

a mini-batch of gradients at initialization. However, existing gradient-based methods rely only on heuristic metrics and lack the necessary theoretical insights to consolidate their designs.

In this paper, we propose GradSign, a simple yet accurate metric for MPI with theoretical foundations. GradSign is inspired by analyzing the optimization landscapes of individual training samples. GradSign takes as inputs a mini-batch of sample-wise gradients evaluated at a random initialization point and outputs the statistical evidence of a network that highly correlates to its well-trained predictive performance measured by accuracy on the entire dataset.

Prior theoretical results (Allen-Zhu et al., 2019) show that for a sufficiently large neighborhood of a randomly initialized point, the optimization landscape is nearly convex and semi-smooth. To realize its potential for MPI, we generalize these results to *sample-wise* optimization landscapes and propose a quantity Ψ to measure the density of *sample-wise* local optima in the convex areas around a random initialization point. Additionally, we prove that both the training and population losses (i.e., generalization error) of a network are proportionally upper bounded by Ψ^2 under reasonable assumptions.

Based on our theoretical results, we design GradSign, an accurate and simple approximation of Ψ, and empirically show that GradSign generalizes to realistic setups that may violate our assumptions. In addition, our method uses only the sample-wise gradient information at a random initialization point to provide statistical evidence to infer Ψ, which makes it efficient to compute and easy to implement.

Extensive evaluation on seven NAS benchmarks (i.e., NAS-Bench-101, NAS-Bench-201, and five design spaces of NDS) across three datasets (i.e., CIFAR-10, CIFAR-100, and ImageNet16-120) shows that GradSign consistently outperforms existing SOTA gradient-based methods in all circumstances. Furthermore, we integrate GradSign into existing NAS algorithms and show that the GradSign-assisted variants of these NAS algorithms lead to more accurate architectures.

Contributions. This paper makes the following contributions:

- We provide a new perspective to view the overall optimization landscape of a network as a combination of sample-wise optimization landscapes. Based on this insight, we introduce a new quantity Ψ that provides an upper bound on both a network's training and population losses under reasonable assumptions.
- To infer Ψ, we propose GradSign, an accurate and simple estimation of Ψ. GradSign enables fast and efficient MPI using only the gradients of a network at initialization.
- We empirically show that our method could generalize reasonably well for modern network architectures by outperforming other SOTA gradient-based MPI methods. We additionally show that with the help of our method, SOTA NAS algorithms can be further improved.

2 Related Work

Table 1 summarizes existing approaches to inferring the statistical performance of neural architectures.

Sample-based methods assess the performance of a neural architecture by training it on a dataset. Though accurate, sample-based methods require a surrogate training procedure to evaluate each architecture. EconNAS (Zhou et al., 2020) mitigates the cost of training candidate architectures by reducing the number of training epochs, input dataset sizes, resolution of input images, and model sizes.

Theory-based methods leverage recent advances in deep learning theory, such as Neural Tangent Kernel (Jacot et al., 2018) and Linear Region Analysis (Serra et al., 2018), to assess a model's accuracy (Chen et al., 2020a; Mellor et al., 2021; Park et al., 2020). In particular, NNGP (Park et al., 2020) infers a model's performance by fitting its kernel regression parameters on a training dataset and evaluating its accuracy on a validation set, which alleviates the burden of training. As another example, Chen et al. (2020a) utilizes the kernel condition number proposed in Xiao et al. (2020), which can be theoretically proved to correlate to training convergence rate and generalization performance. However, this theoretical evidence is only guaranteed for extremely wide networks while having a specialized initialization mode. While the linear region analysis used in Mellor et al. (2021) and Chen et al. (2020a) is easy to implement, such technique is only applicable to networks with ReLU activations (Agarap, 2018).

Table 1: A summary of existing methods for model performance inference. The right four columns show (1) whether a method is based on theoretical results, (2) whether a method avoids expensive training process, (3) whether a method is applicable to different model architectures, and (4) whether a method is applicable across different tasks.

	Methods	**Theoretical Insight**	**Training Free**	**Model Independent**	**Task Independent**
Sample-Based	EconNAS	✗	✗	✓	✓
Theory-Based	NNGP,TE-NAS, NASWOT	✓	✓	✗	✓
Learning-Based	Neural Predictor, One-Shot-NAS-GCN	✗	✗	✓	✗
Gradient-Based	Snip, Grasp, Synflow Fisher	✗	✓	✓	✓
	GradSign	✓	✓	✓	✓

Learning-based methods train a separate network (e.g., graph neural networks) to predict a network's accuracy (Liu et al., 2018a; Luo et al., 2020; Dai et al., 2019; Wen et al., 2020; Chen et al., 2020b; Siems et al., 2020). Though these learned models can achieve high accuracies on a specific task, this approach requires constructing a training dataset with sampled architectures for each downstream task. As a result, existing learning-based methods are generally task-specific and computationally expensive.

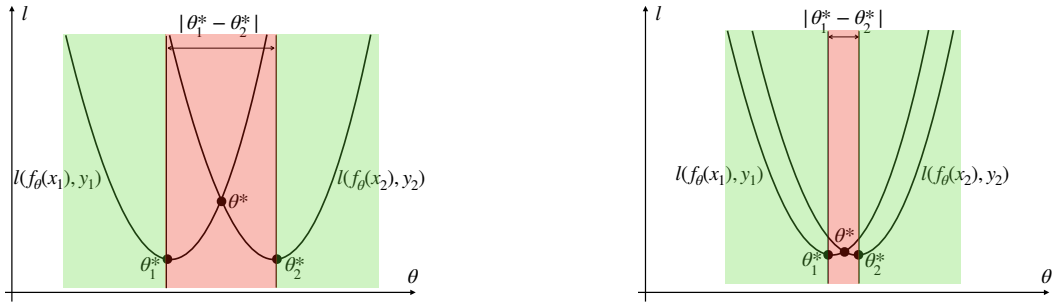
Gradient-based methods infer the statistical performance of a network by leveraging the gradient information at initialization, which can be easily obtained using an automated differentiation tool in today's ML frameworks, such as PyTorch (Paszke et al., 2017) and TensorFlow (Abadi et al., 2016). For example, Snip (Lee et al., 2018) and Grasp (Wang et al., 2020) use a mini-batch of training samples and their gradients to calculate their metrics, while Synflow (Tanaka et al., 2020) is sample-free. An alternative stream of work (Turner et al., 2019; 2021; Theis et al., 2018) uses approximated second-order gradients, known as empirical Fisher Information Matrix (FIM), at a random initialization point to infer the performance of a network. Empirical FIM (Martens, 2014) is a valid approximation of a model's predictive performance only if the model's parameters are a Maximum Likelihood Estimation (MLE). However, this assumption is invalid at a random initialization point, making FIM-based algorithms inapplicable. The key difference between GradSign and existing gradient-based methods is that our method is based on a fine-grained analysis of sample-wise optimization landscapes rather than heuristic insights. In addition, GradSign also provides the first attempt for MPI by leveraging the optimization landscape properties contained in sample-wise gradient information, while prior gradient-based methods only focus on gradients evaluated in a full batch fashion.

2.1 Neural Architecture Search

Recent work has proposed several algorithms to explore a NAS search space and discover highly accurate networks. RS (Bergstra & Bengio, 2012) is one of the baseline algorithms that generates and evaluates architectures randomly in the search space. REINFORCE (Williams, 1992) moves a step forward by reframing NAS as a reinforcement learning task where accuracy is the reward and architecture generation is the policy action. Given limited computational resources, BOHB (Falkner et al., 2018) uses Bayesian Optimization (BO) to propose candidates while uses HyperBand(HB) (Li et al., 2017) for searching resource allocation. REA (Real et al., 2019) uses a simple yet effective evolutionary searching strategy that achieves state-of-the-art performance. GradSign is complementary to and can be combined with existing NAS algorithms. We integrate GradSign into the NAS algorithms mentioned above and show that GradSign can improve the search procedure of NAS algorithms on various real-world tasks.

2.2 Optimization Landscape Analysis

Inspired by the fact that over-parameterized networks always find a remarkable fit for a training dataset (Zhang et al., 2016), optimization landscape analysis has been one of the main focuses in deep learning theory (Brutzkus & Globerson, 2017; Du et al., 2018; Ge et al., 2017; Li & Yuan, 2017; Soltanolkotabi, 2017; Allen-Zhu et al., 2019). Even though existing theoretical results for optimization landscape analysis rely on strict assumptions on the landscape's smoothness, convexity, and initialization point, we can leverage theoretical insights to guide the design of GradSign. In addition, SGD-based optimizers trained from randomly initialized states hardly encounter non-



(a) Optimization landscape with **sparser** sample-wise local optima corresponding to **worse** $J(\theta^*)$.

(b) Optimization landscape with **denser** sample-wise local optima corresponding to **better** $J(\theta^*)$.

Figure 1: Illustration of our theoretical insight that denser sample-wise local optima indicate lower training losses. As the distances ($|\theta_1^* - \theta_2^*|$, shown in red) between the local optima across samples reduce, there is a higher probability that the gradients of different samples have the same sign at a random initialization point, shown as the green areas.

smoothness or non-convexity in practice for varieties of architectures (Goodfellow et al., 2014). Furthermore, Allen-Zhu et al. (2019) provides theoretical evidence that for a sufficiently large neighborhood of a randomly initialized point, the optimization landscape is nearly convex and semi-smooth. Different from existing optimization landscape analyses depending on objectives evaluated across a mini-batch of training samples, we propose a new perspective that decomposes a mini-batch objective into the aggregation of sample-wise optimization landscapes. To the best of our knowledge, our work is the first attempt to inferring models' performance by leveraging sample-wise optimization landscapes.

3 Theoretical Foundations

3.1 Insights

Conventional optimization landscape analyses focus on objectives across a mini-batch of training samples and miss potential evidence hidden in the optimization landscapes of individual samples. By decomposing a mini-batch objective into the summation of *sample-wise* objectives across individual samples in a mini-batch, we can distinguish better local optima as illustrated in Figure 1. Both Figure 1a and Figure 1b reach a local optimum at θ^* for the mini-batch objective $J = \frac{1}{2}(l(f_{\theta^*}(x_1), y_1) + l(f_{\theta^*}(x_2), y_2))$. However, the optimization landscape in Figure 1b contains a better local optimum θ^* (i.e., a lower J). This can be distinguished by analyzing the relative distance between local optima across training samples (i.e., $|\theta_1^* - \theta_2^*|$ in Figure 1).

For a mini-batch with more than two samples, we use a sample-wise local optima density measurement Ψ defined in Section 3.2 to represent the overall closeness of sample-wise local optima. Intuitively, as the distances between the local optima across samples reduce (shown as the red areas in Figure 1), there is a higher probability that the gradients of different samples evaluated at a random initialization point have the same sign (shown as the green areas in Figure 1). Driven by this insight, we propose GradSign to infer the sample-wise local optima density Ψ statistically. The design of GradSign is based on our theoretical results that a network with denser sample-wise local optima has lower training and generalization losses under reasonable assumptions. We introduce the notations and assumptions in Section 3.2, provide a formal derivation of our theoretical results in Section 3.3, and present GradSign in Section 4.

3.2 Preliminaries

We use $\mathcal{S} = \{(x_i, y_i)\}_{i \in [n]}$ to denote training samples, where each $x_i \in \mathbb{R}^d$ is a feature vector, and y_i is the corresponding label. We use $l(\hat{y}_i, y_i)$ to represent a loss function where $\hat{y}_i$ is the prediction of our model. We use $f_\theta(\cdot) : \mathbb{R}^d \to \mathbb{R}^o$ to denote the model parameterized by θ and use $\theta_0 \in \mathbb{R}^m$ to denote random initialized parameters where m is the number of model parameters. Bold font constant denote a constant vector such as $\mathbf{0} = [0, 0, \cdots, 0]$ whose dimension depends on the corresponding situation. $\mathcal{D}$ denotes the underline data distribution, which is the same for training and testing.

Our theoretical results rely on the assumption that there exists a neighborhood Γ_{θ_0} for a random initialization point θ_0 in which sample-wise optimization landscapes are almost convex and semi-smooth. Note that our assumption is weaker than the ones stated in prior work (Allen-Zhu et al., 2019). We use $\{\theta_i^*\}_{i\in[n]} \in \Gamma_{\theta_0}$ to denote a local optima in the convex areas attached to the i-th sample near the initialization point θ_0. Under this assumption, the overall optimization landscape is also convex and semi-smooth within the neighborhood Γ_{θ_0} as additive operations preserve both. We use θ^* to denote a local optimum within Γ_{θ_0} for the mini-batch optimization landscape. Note that θ^* always lie in the convex hull of $\{\theta_i^*\}_{i\in[n]}$. Additionally, we assume that only gradient-based optimizers [1] are used during training. Thus the optimizer eventually converges to θ^*.

Finally, our theoretical analysis assumes that every Hessian in the set $\{\nabla^2 l(f_\theta(x_i), y_i) | \forall i \in [n], \theta \in \Gamma_{\theta_0}\}$ is almost diagonal as in the Neural Tangent Kernel (NTK) regime (Jacot et al., 2018). Section 5 shows that our method generalizes well to real-world networks that may violate this assumption.

Sample-wise local optima density. We use *sample-wise local optima density* to represent the relative closeness of $\{\theta_i^*\}_{i\in[n]}$. Given a dataset $\mathcal{S} = \{(x_i, y_i)\}_{i\in[n]}$, an objective function $l(\hat{y}_i, y_i)$, and a model class $f_\theta(\cdot)$, we use $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$ to measure the average distance between the local optima across samples $\{\theta_i^*\}_{i\in[n]}$ near a random initialization point θ_0:

$$\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot)) \quad = \quad \frac{\sqrt{\mathcal{H}}}{n^2} \sum_{i,j} \|\theta_i^* - \theta_j^*\|_1 \tag{1}$$

$\mathcal{H} \in \mathbb{R}$ is a smoothness upper bound: $\forall k \in [m], i \in [n], [\nabla^2 l(f_\theta(x_i), y_i)]_{k,k} \leq \mathcal{H}$. This upper bound always exists due to the smoothness assumption. Intuitively, $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$ can be interpreted as the mean Manhattan distance with respect to each pair of $\{\theta_i^*\}_{i\in[n]}$ normalized by the inverse of the square root of the smoothness upper bound. The denser $\{\theta_i^*\}_{i\in[n]}$ are, the smaller $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$ is. In an ideal case, $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot)) = 0$ when all local optima are located at the same point.

3.3 Main Results

We show the local optimum property of sample-wise optimization landscapes in the following lemma.

Lemma 1 *There exists no saddle point in a sample-wise optimization landscape and every local optimum is a global optimum.*

Using Lemma 1, we can draw a relation between the training error $J = \frac{1}{n}\sum_i l(f_\theta(x_i), y_i)$ and $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$ using the following theorem.

Theorem 2 *The training error of a network on a dataset $J = \frac{1}{n}\sum_i l(f_\theta(x_i), y_i)$ is upper bounded by $\frac{n^3}{2}\Psi^2_{\mathcal{S},l}(f_{\theta_0}(\cdot))$, and the bound is tight when $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot)) = 0$.*

Finally, we show that $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$ also provides an upper bound for the generalization performance of a network measured by population loss.

Theorem 3 *Given that $Var_{(x_u,y_u)\sim\mathcal{D}}[\|\theta^* - \theta_u^*\|_1^2]$ is bounded by σ^2 where θ_u^* is a local optimum attached to the convex area near θ_0 for $l(f_\theta(x_u), y_u)$. With probability $1-\delta$, the true population loss is upper bounded by $\frac{n^3}{2}\Psi^2_{\mathcal{S},l}(f_{\theta_0}(\cdot)) + \frac{\sigma}{\sqrt{n\delta}}$.*

A formal proof of all theoretical results is available in Appendix A.1.

Main takeaways: The key takeaway of our theoretical results is that $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$ closely relates to an upper bound for the training and generalization performance of a network. Albeit theoretically sound, $\Psi_{\mathcal{S}l}(f_{\theta_0}(\cdot))$ is intractable to be directly measured. Instead, we derive a simple yet accurate metric to reflect $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$, which we present in the next section.

4 GradSign

Inspired by the theoretical results derived above, we introduce GradSign, a simple yet accurate metric for model performance inference. The key idea behind GradSign is a simple and neat quantity to

[1] Gradient descent with infinitesimal step size

statistically reflect the relative value of $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$. Specifically,

$$\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot)) \propto C - \sum_k \sum_{i,j} P(\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k) = \text{sign}([\nabla_\theta l(f_\theta(x_j), y_j)|_{\theta_0}]_k)) \quad (2)$$

where C is a constant and k is the vector index. Detailed derivation is given in Appendix A.1. Using the above relation, we can infer $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$ by directly measuring the signs of the gradients for a mini-batch of training samples at a randomly initialized point instead of going through an end-to-end training process.

To enable more efficient calculation of $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$, we make a further simplification and use the following sample observation:

$$\sum_k |\sum_i \text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k)| \quad (3)$$

to infer the true probability $\sum_k \sum_{i,j} P(\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k) = \text{sign}([\nabla_\theta l(f_\theta(x_j), y_j)|_{\theta_0}]_k))$ whose relation is given by:

$$\frac{1}{n^2} \sum_{i,j} \mathbb{1}_{\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k) = \text{sign}([\nabla_\theta l(f_\theta(x_j), y_j)|_{\theta_0}]_k)} \quad (4)$$

$$\propto \quad |\sum_i \text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k)|^2 \quad (5)$$

The proof of this simplification is included in Appendix A.1. Given the above relationship, we formally state our algorithm pipeline as Algorithm 1. Note that a higher GradSign score indicates better model performance as we have an inverse correlation in Equation (2).

Algorithm 1: GradSign

Result: GradSign score τ_f for a function class f_θ
Given $\mathcal{S} = \{(x_i, y_i)\}_{i \in [n]}$, randomly select initialization point θ_0;
Initialize $g[n, m]$;
for $i = 1, 2, \cdots, n$ **do**
  for $k = 1, 2, \cdots, m$ **do**
    $g[i, k] = \text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k)$
  end
end
$\tau_f = \sum_k |\sum_i g[i, k]|$;
return τ_f

5 EXPERIMENTS

In this section, we empirically verify the effectiveness of our metric against SOTA gradient-based methods on three neural architecture search (NAS) benchmarks, including NAS-Bench-101 (Ying et al., 2019), NAS-Bench-201 (Dong & Yang, 2020) and NDS (Radosavovic et al., 2019)[2]. Theory-based, sample-based, and learning-based methods are excluded in our evaluation, as they either require further training processes or have strong assumptions not suitable for generic architectures.

Baselines. We compare GradSign against existing gradient-based methods, including snip (Lee et al., 2018), grasp (Wang et al., 2020), fisher (Turner et al., 2019), and Synflow. In addition, we also include two heuristic methods (i.e., *grad_norm* and *jacob_cov* used in Abdelfattah et al. (2021a)) as well as a one-shot metric NASWOT(Mellor et al., 2021). Since all gradient-based methods share a similar calculation pipeline (i.e., evaluating the gradients of a mini-batch at a random initialization point), we set the initialization mode and batch size to be the same across all methods to guarantee fairness. Experimental setup details are included in Appendix A.2. To align with the experimental setup of prior work (Abdelfattah et al., 2021b; Mellor et al., 2021), we use two criteria to evaluate the correlations between different metrics and test accuracies across approximately 20k networks:

Spearman's ρ (Daniel et al., 1990) characterizes the monotonic relationships between two variables. The correlation score is restricted in range [-1, 1], where $\rho = 1$ denotes a perfect positive monotonic

[2]All datasets have consented for research purposes and no identifiable personal information is included.

relationship and $\rho = -1$ denotes a perfect negative monotonic relationship. Following prior work, we use Spearman's ρ to evaluate gradient-based methods on NAS-Bench-101 and NAS-Bench-201.

Kendall's Tau. Similar to Spearman's ρ as a correlation measurement, Kendall's Tau is also restricted between [-1, 1]. While Spearman's ρ is more sensitive to error and discrepancies, Kendall's Tau is more robust with a smaller gross error sensitivity. We use Kendall's Tau to quantify the correlation between SOTA one-shot metric scores and model testing accuracies over the NDS search space.

5.1 NAS-Bench-101

NAS-Bench-101 is the first dataset targeting large-scale neural architecture space, containing 423k unique convolutional architectures trained on the CIFAR-10 dataset. The benchmark provides the test accuracy of each architecture in the search space, which we use to calculate the corresponding Spearman's ρ. We use a randomly sampled subset (≈ 4500) of the whole architecture space and a batch size of 64 in this experiment. Table 2 summarizes the results. GradSign significantly outperforms existing gradient-based methods and heuristic approaches and improves the Spearman's ρ score by 25% compared to the best existing method ($0.363 \rightarrow 0.449$).

Table 2: Performance of existed MPI methods (gradient-based + jacob_cov) on NAS-Bench-101 evaluated by Spearman's ρ.

Dataset	*grad_norm*	snip	grasp	fisher	Synflow	*jacob_cov*	GradSign
CIFAR10	0.263	0.189	0.315	0.3	0.363	0.324	**0.449**

5.2 NAS-Bench-201

NAS-Bench-201 is an extended version of NAS-Bench-101 with a different search space, containing 15,625 cell-based candidate architectures evaluated across three datasets: CIFAR-10, CIFAR-100 and ImageNet 16-120. The benchmark provides the test accuracies on the three datasets for all candidate architectures in the search space. We evaluate Spearman's ρ scores for GradSign and existing gradient-based methods. The experiments are conducted overall 15,265 architectures in NAS-Bench-201. The batch size is set to 64. The results on the three datasets are summarized in Table 3. GradSign consistently achieves the best performance across all three datasets and improves the Spearman's ρ scores by $\approx 4\%$ over the best existing approaches. This improvement is significant as the more Spearman's ρ approaches 1, the more difficult it can be further improved.

Table 3: Performance of existed MPI methods (gradient-based + jacob_cov) on NAS-Bench-201 evaluated by Spearman's ρ.

Dataset	*grad_norm*	snip	grasp	fisher	Synflow	*jacob_cov*	GradSign
CIFAR10	0.594	0.595	0.51	0.36	0.737	0.728	**0.765**
CIFAR100	0.637	0.637	0.549	0.386	0.763	0.703	**0.793**
ImageNet16-120	0.579	0.579	0.552	0.328	0.751	0.696	**0.783**

We further select 1000 architecture candidates randomly in the NAS-Bench-201 search space and visualize their testing accuracies and GradSign scores in Figure 2.

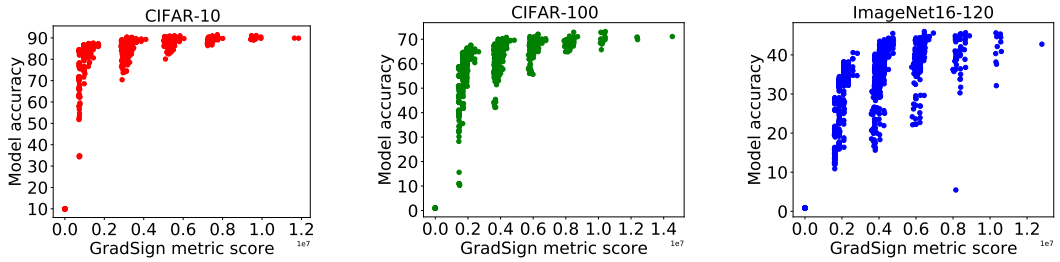


Figure 2: Visualization of model testing accuracy versus GradSign metric score on CIFAR10, CIFAR100, ImageNet16-120.

Figure 2 shows a highly positive correlation between the GradSign score and actual test accuracy of 1000 architectures. A higher GradSign score indicates higher confidence for the statistical performance of architecture. Note that the GradSign scores show a clustering pattern, which may correspond to different architecture classes in the NAS-Bench-201 search space.

5.3 NAS DESIGN SPACE (NDS)

NDS is a unified searching framework that includes five different design spaces: NAS-Net (Zoph et al., 2018), AmoebaNet (Real et al., 2019), PNAS (Liu et al., 2018a), ENAS (Pham et al., 2018), DARTS (Liu et al., 2018b). Each space contains $\sim 1k$ networks that are fully trained on the CIFAR-10 dataset. We include the performance of our method along with *grad_norm*, Synflow, and NASWOT on all five design spaces evaluated by Kendall's Tau and show the results in Table 4. GradSign significantly and consistently outperforms all other MPI methods in all five design spaces.

Table 4: Performance of existed MPI methods on five design spaces in NDS trained over CIFAR-10 evaluated by Kendall's Tau.

	DARTS	ENAS	PNAS	NASNet	Amoeba
grad_norm	0.28	-0.02	-0.01	-0.08	-0.10
Synflow	0.37	0.02	0.03	-0.03	-0.06
NASWOT	0.48	0.34	0.31	**0.31**	0.20
GradSign	**0.54**	**0.43**	**0.40**	**0.31**	**0.24**

5.4 ARCHITECTURE SELECTION

We evaluate whether GradSign can be directly used to select highly accurate architectures in a NAS search space. To pick a top architecture, we randomly sample N candidates in a NAS search space, choose the one with the highest GradSign score, and measure its validation/test accuracies (mean$\pm$std). We compare GradSign with Synflow, NASWOT, `Random`, and `Optimal`. `Random` uniformly samples architectures in the search space, while `Optimal` always chooses the best architecture across N candidates. The results[3] are summarized in Table 5. N in parenthesis indicates the number of architectures sampled in each run. All methods can generally find more accurate architectures with a high N. In addition to outperforming Synflow and NASWOT, GradSign ($N = 100$) can also find better networks even compared to NASWOT($N = 1000$). The results show that GradSign can directly identify accurate architectures besides highly correlating to networks' test accuracies.

Table 5: Mean $\pm$ std accuracy evaluated on NAS-Bench-201. All results are averaged over 500 runs. All searches are conducted on CIFAR-10 while the selected architectures are evaluated on CIFAR-10, CIFAR-100, and ImageNet16-120. N in parenthesis is the number of networks sampled in each run.

Methods	CIFAR-10		CIFAR-100		ImageNet16-120	
	Validation	Test	Validation	Test	Validation	Test
Synflow(N=100)	**89.83±0.75**	93.12±0.52	69.89±1.87	69.94±1.88	41.94±4.13	42.26±4.26
NASWOT(N=100)	89.55±0.89	92.81±0.99	69.35±1.70	69.48±1.70	42.81±3.05	43.10±3.16
NASWOT(N=1000)	89.69±0.73	92.96±0.81	69.98±1.22	69.86±1.21	**44.44±2.10**	**43.95±2.05**
GradSign(N=100)	**89.84±0.61**	**93.31±0.47**	**70.22±1.32**	**70.33±1.28**	42.07±2.78	42.42±2.81
`Random`	83.20±13.28	86.61±13.46	60.70±12.55	60.83±12.58	33.34±9.39	33.13±9.66
`Optimal(N=100)`	91.05±0.28	93.84±0.23	71.45±0.79	71.56±0.78	45.37±0.61	45.67±0.64

5.5 GRADSIGN-ASSISTED NEURAL ARCHITECTURE SEARCH

Besides evaluating GradSign on the Spearman's ρ and Kendall's Tau scores as prior work, we also integrate GradSign into various neural architecture search algorithms and evaluate how GradSign can assist neural architecture search on real-world tasks. Specifically, we integrate GradSign into four NAS algorithms: REA, REINFORCE, BOHB and RS. We design a corresponding method for each NAS algorithm that uses the GradSign scores of candidate architectures to guide the search.

[3]the results for NASWOT are referenced from their paper

Specifically, we integrate GradSign into each NAS algorithm by replacing the random selection of architectures with GradSign-assisted selection. We name these GradSign-assisted variants G-REA, G-REINFROCE, G-HB, and G-RS, and describe their algorithm details in Appendix A.

To evaluate how GradSign can improve the search procedure of NAS algorithms, we run each algorithm with and without GradSign's assistance for 500 runs on NAS-Bench-201 and report the validation and test accuracies of the best-discovered architecture in each run. Following prior work (Mellor et al., 2021; Dong & Yang, 2020), all searches are conducted on the CIFAR-10 dataset with a time budget of 12000s while the performance is evaluated on CIFAR-10, CIFAR-100 and ImageNet16-120. The baselines also include A-REA (Mellor et al., 2021), a variant of REA that uses the NASWOT scores at the initial population selection phase.

Table 6 shows the results. The GradSign-assisted NAS algorithms outperform their counterparts by improving test accuracy by up to 0.3%, 1.1%, and 1.0% on the three datasets.

Table 6: Mean $\pm$ std accuracy evaluated over NAS-Bench-201. All results are averaged over 500 runs. To make a fair comparison across all the methods, the search is performed on CIFAR-10 dataset while the architectures' performance are evaluated over CIFAR-10, CIFAR-100 and ImageNet16-120. All the methods have a search time budget of 12000s. Note that the benchmark results might not match with the original paper as we have run all the experiments from start in a environment different from Dong & Yang (2020).

Methods	CIFAR-10		CIFAR-100		ImageNet16-120	
	Validation	Test	Validation	Test	Validation	Test
REA	91.08±0.45	93.85±0.44	71.59±1.33	71.64±1.25	44.90±1.20	45.25±1.41
A-REA	91.20±0.27	-	71.95±0.99	-	**45.70±1.05**	-
G-REA	**91.27±0.58**	**94.10±0.52**	**72.64±1.57**	**72.70±1.50**	**45.69±1.33**	**45.7±1.32**
RS	90.93±0.37	93.72±0.38	70.96±1.12	71.07±1.07	44.47±1.08	44.61±1.22
G-RS	**91.24±0.21**	**94.02±0.21**	**72.15±0.77**	**72.20±0.76**	**45.38±0.79**	**45.77±0.79**
REINFORCE	90.32±0.89	93.21±0.82	**70.03±1.75**	70.14±1.73	43.57±2.09	43.64±2.24
G-REINFORCE	**90.47±0.55**	**93.37±0.47**	**70.00±1.20**	**70.20±1.29**	**44.33±1.25**	**44.05±1.48**
BOHB	90.84±0.49	93.64±0.49	70.82±1.29	70.92±1.26	44.36±1.37	44.50±1.50
G-HB	**91.18±0.26**	**93.96±0.25**	**71.92±0.92**	**71.99±0.85**	**45.29±0.84**	**45.53±0.92**

6 Conclusion

In this paper, we propose a model performance inference metric GradSign and provide theoretical foundations to support our metric. Instead of focusing on full batch optimization landscape analysis, we move a step further to sample-wise optimization landscape properties, which give us additional information to uncover the quality of the local optima encountered on the optimization trajectory. We propose $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$ to quantitatively characterize the potential of a model $f_\theta(\cdot)$ at a random initialization point θ_0 based on our theory results. Finally, we design the GradSign metric to statistically infer the value of $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$ to give out our final score for model performance inference. Empirically, we have demonstrated that our method consistently achieves the best correlation with true model performance among all other SOTA gradient-based metrics. In addition, we also verified the practical value of our method in assisting existed NAS algorithms to achieve better results. Given that our metric is generic and promising, we believe that our work not only assists in accelerating MPI-related applications but sheds some light on optimization landscape analysis as well. Meanwhile, the effectiveness of our method may further reduce the energy cost introduced by modern NAS algorithms.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*, 2021a.

Mohamed S. Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D. Lane. Zero-Cost Proxies for Lightweight NAS. In *International Conference on Learning Representations (ICLR)*, 2021b.

Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pp. 242–252. PMLR, 2019.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Alon Brutzkus and Amir Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. In *International conference on machine learning*, pp. 605–614. PMLR, 2017.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *arXiv preprint arXiv:2007.12223*, 2020a.

Xin Chen, Lingxi Xie, Jun Wu, Longhui Wei, Yuhui Xu, and Qi Tian. Fitting the search space of weight-sharing nas with graph convolutional networks. *arXiv preprint arXiv:2004.08423*, 2020b.

Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Yangqing Jia, et al. Chamnet: Towards efficient network design through platform-aware model adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11398–11407, 2019.

Wayne W Daniel et al. Applied nonparametric statistics. 1990.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

Simon Du, Jason Lee, Yuandong Tian, Aarti Singh, and Barnabas Poczos. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. In *International Conference on Machine Learning*, pp. 1339–1348. PMLR, 2018.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pp. 1437–1446. PMLR, 2018.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Rong Ge, Jason D Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. *arXiv preprint arXiv:1711.00501*, 2017.

Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 5 2017. ISSN 1557-7317. doi: 10.1145/3065386. URL http://dx.doi.org/10.1145/3065386.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. *arXiv preprint arXiv:1705.09886*, 2017.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 19–34, 2018a.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018b.

Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Enhong Chen, and Tie-Yan Liu. Semi-supervised neural architecture search. *arXiv preprint arXiv:2002.10389*, 2020.

James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley. Neural architecture search without training, 2021.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.

Daniel S Park, Jaehoon Lee, Daiyi Peng, Yuan Cao, and Jascha Sohl-Dickstein. Towards nngp-guided neural architecture search. *arXiv preprint arXiv:2011.06006*, 2020.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

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

Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1882–1890, 2019.

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

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pp. 4558–4566. PMLR, 2018.

Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020.

Mahdi Soltanolkotabi. Learning relus via gradient descent. *arXiv preprint arXiv:1705.04591*, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

Hidenori Tanaka, Daniel Kunin, Daniel LK Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*, 2020.

Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. Faster gaze prediction with dense networks and fisher pruning. *arXiv preprint arXiv:1801.05787*, 2018.

Jack Turner, Elliot J Crowley, Michael O'Boyle, Amos Storkey, and Gavin Gray. Blockswap: Fisher-guided block substitution for network compression on a budget. *arXiv preprint arXiv:1906.04113*, 2019.

Jack Turner, Elliot J Crowley, and Michael FP O'Boyle. Neural architecture search as program transformation exploration. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 915–927, 2021.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *European Conference on Computer Vision*, pp. 660–676. Springer, 2020.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Lechao Xiao, Jeffrey Pennington, and Samuel Schoenholz. Disentangling trainability and generalization in deep neural networks. In *International Conference on Machine Learning*, pp. 10462–10472. PMLR, 2020.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pp. 7105–7114. PMLR, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. Econas: Finding proxies for economical neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11396–11404, 2020.

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

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

A APPENDIX

A.1 PROOF

Lemma 1 Proof: For a single training sample (x_i, y_i), we minimize its objective function $l(f_\theta(x_i), y_i)$ with gradient descent:

$$\nabla_\theta l(f_\theta(x_i), y_i) = \frac{\partial l(f_\theta(x_i), y_i)}{\partial f_\theta(x_i)} \cdot f'_\theta(x_i) \tag{6}$$

At any local optimal point θ_i^*, we have $\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_i^*} = \mathbf{0}$, but $f'_\theta(x_i) \neq \mathbf{0}$ for conventional neural architectures with at least one dense layer [4]. Therefore, we show $\frac{\partial l(f_\theta(x_i), y_i)}{\partial f_\theta(x_i)}|_{\theta_i^*}$ must be equal to 0. For the commonly used objective functions, such as Mean Squared Error Loss and Cross Entropy Loss, this derivative is equal to $C(f_\theta(x_i) - y_i)$, where C is a non-zero constant. Hence we have $f_{\theta_i^*}(x_i) = y_i$ and $l(f_{\theta_i^*}(x_i), y_i) = 0$ at local optima θ_i^*, which makes θ_i^* also a global optima as it is impossible to obtain a lower loss value for this single sample. In addition, at local optima θ_i^*, we have:

$$\begin{aligned}
\nabla_\theta^2 l(f_\theta(x_i), y_i) &= \nabla_\theta\left(\frac{\partial l(f_\theta(x_i), y_i)}{\partial f_\theta(x_i)} \cdot f'_\theta(x_i)\right) && (7)\\
&= \frac{\partial^2 l(f_\theta(x_i), y_i)}{\partial f_\theta(x_i)^2} \cdot f'_\theta(x_i) f'_\theta(x_i)^\top + \frac{\partial l(f_\theta(x_i), y_i)}{\partial f_\theta(x_i)} \cdot f''_\theta(x_i) && (8)\\
&= C \cdot f'_\theta(x_i) f'_\theta(x_i)^\top && (9)
\end{aligned}$$

The above equation implies that $\nabla_\theta^2 l(f_\theta(x_i), y_i)$ is a positive semi-definite matrix, since $C > 0$. This concludes the proof of non existence of saddle points in a sample-wise optimization landscape. This result aligns with our convexity assumptions in Section 3.2.

Theorem 2 Proof: Recall that θ^* denotes a local optima of J which could be reached by a gradient-flow based optimizer start from θ_0. Since θ_0 is randomly sampled and $[\nabla^2 l(f_\theta(x_i), y_i)]_{k,k} \leq \mathcal{H}$, we have:

$$\begin{aligned}
J &= \frac{1}{n}\sum_i l(f_{\theta^*}(x_i), y_i) && (10)\\
&\leq \frac{1}{n}\sum_i \mathcal{H} \cdot \|\theta^* - \theta_i^*\|_2^2 && (11)\\
&\leq \frac{\mathcal{H}}{n}\sum_i \|\theta^* - \theta_i^*\|_1^2 && (12)\\
&\leq \frac{\mathcal{H}}{n}\sum_{i,j} \|\theta_i^* - \theta_j^*\|_1^2 && (13)\\
&\leq n^3 \Psi_{\mathcal{S},l}^2(f_{\theta_0}(\cdot)) && (14)
\end{aligned}$$

where Eq 10 $\rightarrow$ Eq 11 uses the basic property of the smoothness upper bound $\mathcal{H}$ and the fact that each local optimum θ_i^* satisfies $l(f_{\theta_i^*}(x_i), y_i) = 0$. Eq 11 $\rightarrow$ Eq 12 uses Jensen Inequality for square root operators. Eq 12 $\rightarrow$ Eq 13 is derived from the fact that for each dimension in $(\theta^* - \theta_i^*)$ we have:

$$|[\theta^* - \theta_i^*]_k| \leq \sum_j |[\theta_j^* - \theta_i^*]_k| \tag{15}$$

Otherwise, θ^* dose not lie in the convex hull of $\{\theta_i^*\}_{i\in[n]}$ which contradicts with our assumption stated in Section 3.2. The bound is tight when $\theta_i^* = \theta_j^*, \forall i, j \in [n]$.

Eq 10$\rightarrow$Eq 11: As we have $\nabla^2 l(f_\theta(x_i), y_i) \preceq \mathcal{H}\mathbb{I}$ and $l(f_{\theta_i^*}(x_i), y_i) = 0, \nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_i^*} = \mathbf{0}$, we could derive the following inequality:

$$\begin{aligned}
l(f_{\theta^*}(x_i), y_i) &\leq l(f_{\theta_i^*}(x_i), y_i) + \nabla_\theta l(f_{\theta_i^*}(x_i), y_i)^\top(\theta^* - \theta_i^*) + \frac{\mathcal{H}}{2}\|\theta^* - \theta_i^*\|_2^2 && (16)\\
&= \frac{\mathcal{H}}{2}\|\theta^* - \theta_i^*\|_2^2 && (17)
\end{aligned}$$

[4] The gradient values corresponding to the bias term of the last dense layer are always non-zero.

we thus have $\frac{1}{n}\sum_i l(f_{\theta^*}(x_i), y_i) \leq \frac{1}{2n}\sum_i \mathcal{H} \cdot \|\theta^* - \theta_i^*\|_2^2$.

Theorem 3 Proof: Let $\mathbb{E}_{(x_u,y_u)\sim\mathcal{D}}[l(f_\theta^*(x_u), y_u)]$ denotes the true population error. With probability $1-\delta$, we have:

$$\begin{aligned}
\mathbb{E}_{(x_u,y_u)\sim\mathcal{D}}[l(f_\theta^*(x_u), y_u)] &\leq \mathcal{H}\mathbb{E}_{(x_u,y_u)\sim\mathcal{D}}[\|\theta^* - \theta_u^*\|_1^2] && (18)\\
&\leq \frac{\mathcal{H}}{n}\sum_{i=1}^{n}\|\theta^* - \theta_i^*\|_1^2 + \frac{\sigma}{\sqrt{n\delta}} && (19)\\
&\leq n^3\Psi_{\mathcal{S},l}^2(f_{\theta_0}(\cdot)) + \frac{\sigma}{\sqrt{n\delta}} && (20)
\end{aligned}$$

where n and σ are constants, $\mathcal{S}$ is a training dataset, and $\mathcal{D}$ denotes its underlying data distribution. This implies that $\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot))$ is an accurate indicator for the true population loss. Eq 18 $\rightarrow$ Eq 19 uses Chebyshev's inequality, while Eq 19 $\rightarrow$ Eq 20 uses the same inequality derived in **Claim 2**.

Algorithm Proof: Given that sample-wise local optima $\{[\theta_i^*], i \in [n]\}$ are contained in the convex area around θ_0. We derive the following property:

$$\text{sign}([\theta_i^* - \theta_0]_k) = \text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k) \quad (21)$$

Since θ_0 is a randomly chosen initialization point, without loss of generality, we assume θ_0 is sampled from a hypercube $[-a, a]$. Thus we have:

$$P(\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k) \neq \text{sign}([\nabla_\theta l(f_\theta(x_j), y_j)|_{\theta_0}]_k)) = \frac{|[\theta_i^*]_k - [\theta_j^*]_k|}{2a} \quad (22)$$

Where $P(\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k) \neq \text{sign}([\nabla_\theta l(f_\theta(x_j), y_j)|_{\theta_0}]_k))$ denotes the probability for $[\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k$ and $[\nabla_\theta l(f_\theta(x_j), y_j)|_{\theta_0}]_k$ having different signs. Notice that we have completely dropped the dependency for θ_i^* at this point and can simply infer from $\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k)$. To complete our proof:

$$\begin{aligned}
\Psi_{\mathcal{S},l}(f_{\theta_0}(\cdot)) &= \frac{\sqrt{\mathcal{H}}}{n}\sum_{i,j}\|\theta_i^* - \theta_j^*\|_1 && (23)\\
&= \frac{2a\sqrt{\mathcal{H}}}{n}\sum_k\sum_{i,j} P(\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k) \neq \text{sign}([\nabla_\theta l(f_\theta(x_j), y_j)|_{\theta_0}]_k)) && (24)\\
&\propto n^2 - \sum_k\sum_{i,j} P(\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k) = \text{sign}([\nabla_\theta l(f_\theta(x_j), y_j)|_{\theta_0}]_k)) && (25)
\end{aligned}$$

Simplification Proof:: for each i, j and a given k, we could estimate $P(\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k) = \text{sign}([\nabla_\theta l(f_\theta(x_j), y_j)|_{\theta_0}]_k))$ using:

$$\frac{|\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k) + \text{sign}([\nabla_\theta l(f_\theta(x_j), y_j)|_{\theta_0}]_k)|}{2} \quad (26)$$

which is valid (not equal to zero) only when $\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k) == \text{sign}([\nabla_\theta l(f_\theta(x_j), y_j)|_{\theta_0}]_k)$. Suppose we have p positive and n-p negative $\text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k)$ for n samples, since we only care about samples share the same sign, the original probability estimation is $\frac{(n-p)^2+p^2}{n^2} = \frac{1}{2} + \frac{(n-2p)^2}{2n^2}$. Thus we only need to measure the quantity $|n-2p|$ which simply equals to $|\sum_i \text{sign}([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k)|$.

A.2 EXPERIMENTS SETUP

The code we used during experimentation was mainly based on existed code base Abdelfattah et al. (2021b); Mellor et al. (2021)Abdelfattah et al. (2021b) which is under Apache-2.0 License.

The hardwares we used were Amazon EC2 C5 instances with no GPU involved and p3 instance with one V100 Tensor Core GPU.

As our methods is gradient-based which is training free, we don't need to split our dataset. For Spearman's correlation measurement on NAS-Bench-201, we set batch size to 64, which is used by most baselines. For the Kendall's Tau experiment, other accuracy comparison experience and the GradSign assisted algorithms, we used a batch size of 128, also to match the batch size in other baselines (NASWOT). We use Pytorch default parameter initialization for all architectures. Random seed in correlation experiments is set to 42 which is also randomly chosen. For accuracy experiments, our results are summarized over 500 runs whose random seed are chosen randomly for each run. For the correlation evaluation of each individual architecture, we only use one θ_0 for minimizing computational cost. Our approach can be easily generalized to an average of multiple θ_0s and can trade-off between efficiency and accuracy.

A.3 GRADSIGN ASSISTED NAS ALGORITHMS

Algorithm 2: G-REA

```
Result: Find the best performing architecture given the time constraint for 12000s
Population = [];
History = [];
population_size;
sample_size;
pool_size;
for 1, 2, ··· ,population_size do
    model = random.arch();
    model.acc, model.time_cost = eval(model);
    Population.append(model);
    History.append(model);
end
while not exceeding time budget do
    Sample = [];
    for 1, 2, ··· ,sample_size do
        Sample.append(random.choice(Population))
    end
    parent = max_acc(Sample);
    GradSign_pool = [];
    for 1, 2, ··· ,pool_size do                    /* GradSign assisted part */
        model = mutate_arch(parent);
        model.score = GradSign(model);
        GradSign_pool.append(model);
    end
    child = max_score(GradSign_pool);
    Population.append(child);
    History.append(child);
    Population.popleft();
end
return max_acc(History)
```

Algorithm 3: G-RS

Result: Find the best performing architecture given the time constraint for 12000s
History = [];
pool_size;
while *not exceeding time budget* **do**
  GradSign_pool = [];
  for $1, 2, \cdots$,*pool_size* **do** `/* GradSign assisted part */`
    model = random_arch();
    model.score = GradSign(model);
    GradSign_pool.append(model);
  end
  arch = max_score(GradSign_pool) arch.acc = eval(arch) History.append(arch);
end
return max_acc(History);

Algorithm 4: G-REINFORCE

Result: Find the best performing architecture given the time constraint for 12000s
History = [];
pool_size;
policy π_{θ_0};
Reward = [];
baseline;
while *not exceeding time budget* **do**
  arch = generate_arch(π_{θ_i});
  GradSign_pool = [];
  for $1, 2, \cdots$,*pool_size* **do** `/* GradSign assisted part */`
    child = mutate_arch(arch);
    child.score = GradSign(child);
    GradSign_pool.append(child);
  end
  arch = max_score(GradSign_pool);
  arch.acc = eval(arch);
  r= arch.acc;
  History.append(arch);
  Reward.append(r);
  baseline.update(r);
  $\theta_{i+1} = \theta_i + \nabla_\theta \mathbb{E}_{\pi_{\theta_i}}[r - \text{baseline}]$
end
return max_acc(History);
return

Algorithm 5: G-HB

Result: Find the best performing architecture given the time constraint for 12000s
Input: budgets $b_{\min}$ and $b_{\max}$, η;
$s_{\max} = \lfloor \log_{\eta} \frac{b_{\max}}{b_{\min}} \rfloor$;
score_list = [];
pool_size;
for $s \in \{s_{\max}, s_{\max-1}, \cdots, 0\}$ **do**
    config_space = [];
    set $n = \lceil \frac{s_{\max}+1}{s+1} \cdot \eta^{s} \rceil$;
    while *sizeof(config_space)* < *n* **do**
        GradSign_pool = [];
        for $1, 2, \cdots$,*pool_size* **do** `/* GradSign assisted part */`
            model = random_arch();
            if *model in score_list* **then**
                model.score = score_list[model];
            else
                model.score = GradSign(model);
                score_list[model] = model.score;
            end
            GradSign_pool.append(model);
        end
        arch = max_score(GradSign_pool);
        config_space.append(arch);
    end
    run SH on them with initial budget as $\eta^{s} \cdot b_{\max}$;
end
return best evaluated architecture;