Evolved Optimizer for Vision

Xiangning Chen[1,2*], **Chen Liang**[1], **Da Huang**[1], **Esteban Real**[1], **Yao Liu**[1†], **Kaiyuan Wang**[1], **Cho-Jui Hsieh**[2], **Yifeng Lu**[1], **Quoc V. Le**[1]

[1]Google [2]Department of Computer Science, UCLA

Abstract We present an optimizer, Hero-Lion (*EvoLved Sign Momentum*), discovered by evolutionary search from basic math operations in the AutoML-Hero project. It keeps track of only the momentum and leverages the sign operation to calculate the update to the weights. Despite the simplicity, Hero-Lion outperforms the commonly used optimizer, such as AdamW, AdafactorW, and SGD with momentum, for training a variety of architectures on different tasks. Notably, it improves the accuracy of Vision Transformer for up to 2% when trained from scratch on ImageNet. When used in pre-training with larger data and model sizes, Hero-Lion still outperforms AdamW and AdafactorW, and can save 2-5x compute. On JFT-300M, ViT-L/16 trained by Hero-Lion matches the accuracy of the previous ViT-H/14 trained by AdamW. By replacing AdafactorW with Hero-Lion, we improve the ImageNet accuracy of ViT-G/14, pre-trained on JFT-3B, from 90.45% to 90.71%. Besides, Hero-Lion improves the contrastive pre-training of multi-modal Transformers by achieving $\sim$1% gain of ImageNet zero-shot accuracy.

1 Introduction

Optimization algorithms play a fundamental role in training neural networks. Despite a tremendous number of new optimizers introduced in recent years [2, 5, 26, 49], Adam [22], proposed in 2014, and its variant with decoupled weight decay, AdamW [27], are still the de facto standard optimizers for most deep neural networks, especially the recently proposed convolution-free or hybrid vision models, e.g., Vision Transformer (ViT) [12], MLP-Mixer [41], CoAtNet [11], etc.

Besides handcrafted optimizers, the learning-to-optimize approach proposes to automatically discover optimizers by training parameterized models, e.g., neural networks, to output the updates [1, 23, 28, 43]. However, those black-box optimizers often fail to generalize to practical learning tasks that train on larger models with longer training steps. Neural Optimizer Search [3] applies reinforcement learning to discover new optimizers represented as binary trees composed from a selected subset of operands and operators. However, the restricted search space limits the potential of the search. For example, it cannot change how the estimated first or second moments are updated.

In this paper, we present an optimizer discovered in the AutoML-Hero project that leverages the evolutionary search to discover novel machine learning algorithms from basic math operations. Our proxy task trains a ViT [12] on ImageNet and we warm start the search with AdamW. The evolutionary search discovers a simple and effective learning algorithm: Hero-Lion (short for *EvoLved Sign Momentum*) that only tracks the momentum and does not apply the adaptive learning rate like AdamW. It achieves superior generalization across various architectures (ViT, MLP-Mixer, CoAtNet and ResNet), datasets (ImageNet, JFT, image-text pairs), and learning tasks (supervised and contrastive). The main focus of this paper is to evaluate the discovered optimizer. We defer the detailed discussion of the search method and the results in other domains to a more comprehensive version later. Notably, Hero-Lion achieves up to 5x computational cost reduction when pre-training ViT on JFT-300M (Figure 1) and improve the accuracy of ViT-B/16 for 2% when trained from scratch

*Work done as a student researcher at Google Brain.
†Work done while at Google.

on ImageNet (Table 2). Besides, the gap between our optimizer and AdamW tends to enlarge on bigger models and more challenging benchmarks (e.g., ImageNet A [17]). Our approach scales to the largest ViT-G/14 and improves its ImageNet accuracy from 90.45% to 90.71% (Table 1). Apart from the supervised setting, we also observe that Hero-Lion enhances the contrastive pre-training of multi-modal Transformer by bringing ~1% improvement on zero-shot accuracies (Table 3).

```
def train(w, g, m, v):
    g2 = square(g)
    m = interpolate(g, m, 0.9)
    v = interpolate(g2, v, 0.999)
    v_sqrt = sqrt(v)
    update = m / v_sqrt
    wd = w * λ
    update = update + wd
    update = update * lr
    return update, m, v
```

Step 1: Initialize (AdamW)

```
def train(w, g, m, v):
    g = clip_by_global_norm(g, 1.0)
    g = arcsin(g)
    m = interpolate(g, v, 0.9)
    m2 = square(m)
    v = interpolate(g, m, 1.1)
    m_abs = sqrt(m2)
    update = m / m_abs
    wd = w * λ
    update = update + wd
    update = update * lr
    return update, m, v
```

Step 2: Evolve

```
def train(w, g, m, v):
    update = interpolate(g, m, 0.9)
    update = sign(update)
    m = interpolate(g, m, 0.99)
    wd = w * λ
    update = update + wd
    update = update * lr
    return update, m, v
```

Step 3: Simplify (Hero-Lion)

2 Method

Inspired by AutoML-Zero [34], we represent a training algorithm as a program, and the search space consists of basic mathematical operations. We apply regularized evolution [33], and warm start the search by initializing the population with AdamW [22, 27], an exceedingly popular optimizer used in training state-of-the-art vision models. The proxy task is to train a small ViT on ImageNet-1K for ~50K steps and we treat the validation accuracy as the fitness. As shown in Step 1, AdamW is represented as a function that takes in the weights (`w`), gradient (`g`), the estimated first (`m`) and second (`v`) moments as inputs. It returns the update (`update`) to the weights, the new estimated first and second moments that will be forwarded to the next training step. In the program, the `interpolate(x, y,` β`) = (1 -` β`)x +` β`y` is the linear interpolation function, λ and lr are constants that represent the weight decay factor and the peak learning rate. For simplicity, we omit the bias correction term and the ϵ added to the gradient magnitude estimate for numerical stability.

The evolutionary search produces the program in Step 2, and we simplify it to get the final algorithm (Hero-Lion). The statement using `arcsin` is removed since we observe no difference in quality without it. We also omit the universal gradient clipping and merge the two `interpolate` operations in Step 3. The derived algorithm entirely abandons the adaptive learning rate for different weights, and the update magnitude is uniform by simply taking the `sign` operation. It only tracks the momentum using exponential moving average (EMA) of the gradients thus cuts the memory requirement of AdamW by half, which can be beneficial when training extremely giant models [36, 46]. The decay factor used to track the momentum is 0.99 instead of the commonly used default value 0.9 in AdamW, and it interpolates the momentum and the current gradient using 0.9 before applying the `sign` operation. Our intuition is that it remembers longer gradient history for momentum estimate but focuses more on the current gradient (see Section 3.2 for an ablation). We manually add a bias correction term (for mathematical soundness, has minor effects) and show the full optimizer in Algorithm 2 (in the appendix). More details of the search space and search method are deferred to the comprehensive version.

3 Evaluation

We perform extensive experiments spanning various datasets, architectures, and learning tasks. We mainly compare our searched optimizer to AdamW [22, 27] (or AdafactorW [36] when memory is a bottleneck) as it is exceedingly popular. The results of SGD with momentum is only included

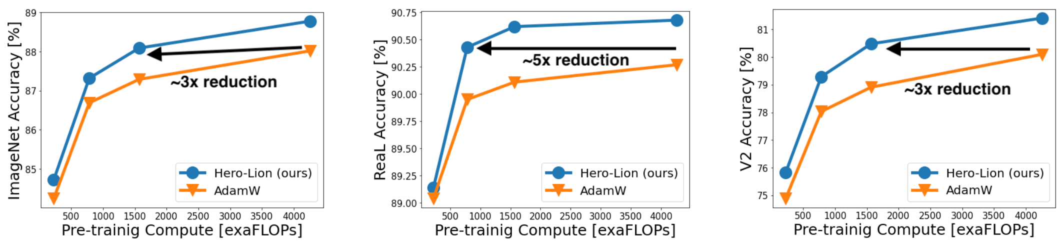

Figure 1: ImageNet (**Left**), ReaL (**Middle**), and V2 (**Right**) accuracy vs. pre-training cost when training ViTs on JFT-300M. Models are fine-tuned on ImageNet-1K at resolution 384^2 (392^2 for ViT-H/14). See Table 4 (in the appendix) for the detailed numbers.

Table 1: Model Performance when pre-trained on JFT. Two giant ViTs are pre-trained on JFT-3B and smaller ones are pre-trained on JFT-300M.

Model	ViT-L/16_{512}		ViT-H/14_{518}		ViT-g/14_{518}		ViT-G/14_{518}	
#Params	305.18M		633.47M		1.04B		1.88B	
Optimizer	AdamW	Hero-Lion	AdamW	Hero-Lion	AdamW	Hero-Lion	AdafactorW	Hero-Lion
ImageNet	87.76 [12] / 87.72	**88.50**	88.55 [12] / 88.55	**89.09**	90.25	**90.52**	90.45 [46]	**90.71 / 90.71***
ReaL	90.54 [12] / 90.46	**90.91**	90.72 [12] / 90.62	**91.02**	90.84	**91.11**	90.81 [46]	**91.06 / 91.25***
V2	79.80	**81.13**	81.12	**82.24**	83.10	**83.39**	83.33 [46]	**83.54 / 83.83***
A	52.72	**58.80**	60.64	**63.78**	-	-	-	-
R	66.95	**72.49**	72.30	**75.07**	-	-	-	-

* We observe overfitting in the fine-tuning process, and therefore report both the last and the oracle results here.

for ResNet since it performs much worse than AdamW when optimizing other architectures (e.g., 71.5% vs. 74.6% ImageNet accuracy reported in Chen et al. [9]). For evaluation, we report the results on various benchmarks including the ImageNet-1K validation set, ImageNet ReaL [6], ImageNet V2 [35], ImageNet A [17], and ImageNet R [16]. The image resolution is 224^2 by default if not specified by the subscript in the model name.

Hyperparameter tuning. For a fair comparison, we only tune the peak learning rate lr and weight decay λ for both AdamW and our Hero-Lion in the log scale, i.e., $lr \in \{1e-4, 3e-4, 1e-3, 3e-3, \ldots\}$, $\lambda \in \{0.1, 0.3, 1.0, 3, 0, \ldots\}$. Other hyperparameters are fixed throughout the paper, e.g., $\beta_1 = 0.9$, $\beta_2 = 0.999$ for AdamW and $\beta_1 = 0.9$, $\beta_2 = 0.99$ for Hero-Lion. Please see Table 5 (in the appendix) for the detailed settings.

3.1 Main Results

Train from scratch on ImageNet-1K. Following previous works [11, 12, 41], we train all models except ResNet for 300 epochs with 4,096 batch size. For ResNet-50, we use batch size 1,024 and train it for 90 epochs. The learning rate schedules are cosine decay with 10K steps warmup.

As shown in Table 2, Hero-Lion significantly outperforms AdamW on various architectures. Empirically, *the improvement enlarges on models with larger capacity*. For instance, 1.96% vs. 0.58% accuracy increase for ViT-B/16 and ViT-S/16, respectively. Besides, *the gap tends to become larger with fewer inductive biases inserted*. When augmentations, i.e, RandAugment [10] and Mixup [48], are already applied (to encourage translation, rotation equivalence, etc.), the gain of Hero-Lion over AdamW is reduced. But still, our optimizer outperforms AdamW for 0.42% when training CoAtNet-3, although a strong regularization is included in the training recipe [11] (RandAugment with two layers and magnitude 15, Mixup of 0.8, and stochastic depth [18] of 0.7).

Pre-train on ImageNet-21K. We pre-train ViT-B/16 and ViT-L/16 on ImageNet-21K for 90 epochs with 4,096 batch size, resulting in a total of ~280K steps. The fine-tuning settings are the same as Dosovitskiy et al. [12]: momentum SGD with 20K steps and batch size 512.

Table 2 shows that Hero-Lion still beats AdamW after the training set is enlarged for ~10 times. Their gaps on larger models are consistently bigger: +0.52% vs. +0.33% (ImageNet), +0.57% vs. +0.23% (ReaL), and +0.74% vs. +0.25% (V2) for ViT-L/16 and ViT-B/16, respectively.

Table 2: Model performance on ImageNet, ImageNet ReaL, and ImageNet V2. All results are averaged from three independent runs.

Model	#Params	Optimizer	Augmentation (RandAug + Mixup)	ImageNet	ReaL	V2
Train from scratch on ImageNet-1K						
ResNet-50	25.56M	SGD	✗	76.22	82.39	63.93
		AdamW		76.34	**82.72**	**64.24**
		Hero-Lion		**76.45**	**82.72**	64.02
Mixer-S/16	18.53M	AdamW	✗	69.26	75.71	55.01
		Hero-Lion		**69.92**	**76.19**	**55.75**
Mixer-B/16	59.88M	AdamW	✗	68.12	73.92	53.37
		Hero-Lion		**70.11**	**76.60**	**55.94**
ViT-S/16	22.05M	AdamW	✗	76.12	81.94	63.09
		Hero-Lion		**76.70**	**82.64**	**64.14**
		AdamW	✓	76.97	83.35	64.59
		Hero-Lion		**77.49**	**83.68**	**65.31**
ViT-B/16	86.57M	AdamW	✗	75.48	80.64	61.87
		Hero-Lion		**77.44**	**82.57**	**64.81**
		AdamW	✓	79.65	84.65	67.39
		Hero-Lion		**80.11**	**85.82**	**68.50**
CoAtNet-1	42.23M	AdamW	✓	83.64	-	-
		Hero-Lion		**84.07**	-	-
CoAtNet-3	166.97M	AdamW	✓	84.45	-	-
		Hero-Lion		**84.87**	-	-
Pre-train on ImageNet-21K						
ViT-B/16$_{384}$	86.86M	AdamW	✗	84.12	88.61	73.81
		Hero-Lion		**84.45**	**88.84**	**74.06**
ViT-L/16$_{384}$	304.72M	AdamW	✗	85.07	88.78	75.10
		Hero-Lion		**85.59**	**89.35**	**75.84**

Pre-train on JFT. To push the limit of Hero-Lion, we perform extensive experiments based on the JFT [39] dataset. We follow the settings of Dosovitskiy et al. [12] and Zhai et al. [46] for both pre-training and fine-tuning. Figure 1 shows the fine-tuning accuracy under different pre-training budgets spanning three scales of ViT (ViT-B/16, ViT-L/16, and ViT-H/14) on JFT-300M. Our optimizer enables the ViT-L/16 to match the performance of ViT-H/14 trained by AdamW on ImageNet and ImageNet V2 but with ~3x less pre-training cost. On ImageNet ReaL, the compute saving increases to ~5x: ViT-L/16 pre-trained for 7 epochs (~500K steps) by Hero-Lion surpasses the ViT-H/14 pre-trained for 14 epochs (~1M steps) by AdamW. Another evidence is that Zhai et al. [46] pre-train ViT-L/16 for ~4M steps to achieve the accuracy of 88.0% (ImageNet), 90.3% (ReaL) and 79.5% (V2), underperforming the same model with ~4x fewer steps by Hero-Lion.

We further fine-tune with higher resolution and Polyak averaging [31] in Table 1. Our obtained ViT-L/16 achieves a 0.78% accuracy gain on ImageNet compared to the AdamW trained counterpart and matches the previous ViT-H/14 results. The gap between Hero-Lion and AdamW is larger on more challenging benchmarks (mainly for real-world robustness): +1.33% (V2), +6.08% (A), +5.54% (R) for ViT-L/16 and +1.12% (V2), +3.14% (A), +2.77% (R) for ViT-H/14, revealing its superior generalization performance. After scaling up the pre-training dataset to JFT-3B, the obtained ViT-g/14 beats the previous ViT-G/14 results [46] with ~2x fewer parameters and GFLOPs, and our ViT-G/14 achieves a 90.71% ImageNet accuracy.

Pre-train on Image-Text Dataset. To demonstrate the potential and effectiveness of Hero-Lion in other domains, we pre-train a series of multi-modal Transformers on the in-house image-text dataset introduced in LiT [47]. We control the seen image-text pairs as 1B [‡], and the locked ViT (pre-trained on JFT-3B) are exactly the same for Hero-Lion and AdafactorW [36]. The model name

[‡]We use 1B settings that is much fewer than the 18B seen image-text pairs in the LiT paper for a faster comparison.

Table 3: Zero-shot accuracies (%) on ImageNet, CIFAR-100, and Oxford-IIIT Pet. Every model sees 1B image-text pairs.

Model	Optimizer	ImageNet	CIFAR-100	Pet
LiT-B/32-B	AdafactorW	68.78	71.41	86.62
	Hero-Lion	**69.88**	**71.78**	**87.36**
LiT-B/16-B	AdafactorW	74.26	72.25	89.83
	Hero-Lion	**75.39**	**72.49**	**91.20**
LiT-g/14_{288}-L	AdafactorW	83.43	80.93	94.88
	Hero-Lion	**84.09**	**81.43**	**95.86**

in Table 3 specifies the size, e.g., LiT-B/16-B denotes using ViT-B/16 as the image encoder and BERT-base as the text encoder.

In LiT [47], the pre-trained image encoder is frozen so only the text encoder is optimized in a contrastive manner. Surprisingly, Hero-Lion still achieves significant performance gain although it is searched based on a pure vision model. As shown in Table 3, we improve the ImageNet zero-shot accuracies of LiT-B/32-B, LiT-B/16-B, and LiT-g/14-L for 1.10%, 1.13%, and 0.66% respectively. Our performance gain is consistent on CIFAR-100 and Oxford-IIIT Pet.

3.2 Ablation Study

A previous work [5] proposes the signSGD optimizer (and its momentum variant) to alleviate the communication bottleneck under the distributed training settings, although showing inferior accuracy on ImageNet. To ablate the effect of β_1 and β_2 in Hero-Lion, we compare with a simple update rule similar to the one proposed in the previous work: `m = interpolate(g, m, 0.9); update = sign(m)`. As shown in Figure 2 (left), the ablated optimizer performs in-between AdamW and Hero-Lion, but its convergence speed is slower as the accuracy catches up at the last stage. We further use it to pre-train a ViT-B/16 on JFT-300M and achieve an ImageNet accuracy of 84.39%, compared to 84.24% by AdamW and 84.72% by Hero-Lion (see Table 4 in the appendix). Those results validate the necessity of using two `interpolate` functions in Hero-Lion.

4 Related Work

Our work lies in the area of AutoML that includes learning to learn [1, 3, 28, 32, 43, 44, 45], neural architecture search [7, 8, 25, 29, 30, 33, 38, 42, 51], and hyperparameter optimization [4, 19, 20, 24, 37, 40], etc. Our work builds upon on a symbolic search space similar to AutoML-Zero [34] but aims at discovering optimizers that can be applied to challenging vision benchmarks. Other related works are efforts on handcrafted optimizers [5, 13, 15, 22, 27, 36, 49].

5 Conclusion

This paper proposes to automatically discover optimizers via evolutionary search. We represent optimizers by programs consisting of basic mathematical operations, and warm start the population by AdamW. Our discovered optimizer Hero-Lion only tracks the momentum and applies the sign operation to calculate the update to the weights, halving the memory consumption of AdamW. Despite its simplicity, Hero-Lion achieves extraordinary generalization across architectures, datasets and tasks. We leave the detailed search method and results in other domains to a more comprehensive version in the future.

6 Limitations and Broader Impact Statement

This paper is limited in the vision (and vision-language) domain, and we have not tried some self-supervision settings such as contrastive learning and masked image modeling. Potential social impacts include reducing the compute thus the energy cost required to train giant models.

7 Reproducibility Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   (b) Did you describe the limitations of your work? [Yes] In Section 6.

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In Section 6.

   (d) Have you read the ethics author's and review guidelines and ensured that your paper conforms to them? `https://automl.cc/ethics-accessibility/` [Yes]

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit version), an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [No] We provide detailed hyperparameter settings in the appendix, and we will release the code in the future.

   (b) Did you include the raw results of running the given instructions on the given code and data? [Yes] In Section 3.

   (c) Did you include scripts and commands that can be used to generate the figures and tables in your paper based on the raw results of the code, data, and instructions given? [No] We provide raw results for the figure in the appendix.

   (d) Did you ensure sufficient code quality such that your code can be safely executed and the code is properly documented? [N/A]

   (e) Did you specify all the training details (e.g., data splits, pre-processing, search spaces, fixed hyperparameter settings, and how they were chosen)? [Yes] In Section 3 and the appendix.

   (f) Did you ensure that you compared different methods (including your own) exactly on the same benchmarks, including the same datasets, search space, code for training and hyperparameters for that code? [Yes] We make fair comparisons between AdamW and our optimizer.

   (g) Did you run ablation studies to assess the impact of different components of your approach? [Yes] In Section 3.2.

   (h) Did you use the same evaluation protocol for the methods being compared? [Yes] We exactly follow the previous metrics and fine-tuning settings.

   (i) Did you compare performance over time? [Yes] Figure 2 (left) shows the ImageNet accuracy vs. training steps.

   (j) Did you perform multiple runs of your experiments and report random seeds? [Yes] All results are an average of three independent runs.

   (k) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] The variance is pretty small compared to the gap between AdamW and our optimizer.

(l) Did you use tabular or surrogate benchmarks for in-depth evaluations? [Yes] We evaluate on multiple benchmarks including ImageNet-1K, ImageNet ReaL, ImageNet V2, ImageNet A, and ImageNet R.

(m) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In Section 3 and the appendix.

(n) Did you report how you tuned hyperparameters, and what time and resources this required (if they were not automatically tuned by your AutoML method, e.g. in a NAS approach; and also hyperparameters of your own method)? [Yes] In Section 3, we tune the peak learning rate and the weigt decay in the log scale.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets…

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [No] All datasets are publicly available except the JFT and the image-text pairs dataset.

   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects…

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Acknowledgements. We would like to thank Xiaohua Zhai, Zihang Dai, Hanxiao Liu, Hieu Pham, and Boqing Gong for helpful discussions; the Google Brain team at large for providing a supportive research environment.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[2] Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 404–413. PMLR, 10–15 Jul 2018.

[3] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Neural optimizer search with reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 459–468. JMLR.org, 2017.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[5] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 560–569. PMLR, 10–15 Jul 2018.

[6] Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet?, 2020.

[7] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1554–1565. PMLR, 13–18 Jul 2020.

[8] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. DrNAS: Dirichlet neural architecture search. In *International Conference on Learning Representations*, 2021.

[9] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pre-training or strong data augmentations. In *International Conference on Learning Representations*, 2022.

[10] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. Randaugment: Practical automated data augmentation with a reduced search space. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624. Curran Associates, Inc., 2020.

[11] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 3965–3977. Curran Associates, Inc., 2021.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[13] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[14] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

[15] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1842–1850. PMLR, 10–15 Jul 2018.

[16] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8340–8349, October 2021.

[17] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15262–15271, June 2021.

[18] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 646–661, Cham, 2016. Springer International Publishing.

[19] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[20] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 240–248, Cadiz, Spain, 09–11 May 2016. PMLR.

[21] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[23] Ke Li and Jitendra Malik. Learning to optimize. In *International Conference on Learning Representations*, 2017.

[24] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1):6765–6816, jan 2017. ISSN 1532-4435.

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

[26] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[28] Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4556–4565. PMLR, 09–15 Jun 2019.

[29] Daiyi Peng, Xuanyi Dong, Esteban Real, Mingxing Tan, Yifeng Lu, Gabriel Bender, Hanxiao Liu, Adam Kraft, Chen Liang, and Quoc Le. Pyglove: Symbolic programming for automated machine learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 96–108. Curran Associates, Inc., 2020.

[30] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4095–4104. PMLR, 10–15 Jul 2018.

[31] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855, jul 1992. ISSN 0363-0129. doi: 10.1137/0330046.

[32] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[33] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33 (01):4780–4789, Jul. 2019. doi: 10.1609/aaai.v33i01.33014780.

[34] Esteban Real, Chen Liang, David So, and Quoc Le. Automl-zero: Evolving machine learning algorithms from scratch. In *International Conference on Machine Learning*, pages 8007–8019. PMLR, 2020.

[35] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5389–5400. PMLR, 09–15 Jun 2019.

[36] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604. PMLR, 10–15 Jul 2018.

[37] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[38] David So, Quoc Le, and Chen Liang. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886. PMLR, 2019.

[39] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[40] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 847–855, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450321747. doi: 10.1145/2487575.2487629.

[41] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24261–24272. Curran Associates, Inc., 2021.

[42] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable NAS. In *International Conference on Learning Representations*, 2021.

[43] Olga Wichrowska, Niru Maheswaranathan, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3751–3760. JMLR.org, 2017.

[44] Yuanhao Xiong, Li-Cheng Lan, Xiangning Chen, Ruochen Wang, and Cho-Jui Hsieh. Learning to schedule learning rate with graph neural networks. In *International Conference on Learning Representations*, 2022.

[45] Zhen Xu, Andrew M. Dai, Jonas Kemp, and Luke Metz. Learning an adaptive learning rate schedule, 2019.

[46] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2021.

[47] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18123–18133, June 2022.

[48] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[49] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18795–18806. Curran Associates, Inc., 2020.

[50] Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha C Dvornek, sekhar tatikonda, James s Duncan, and Ting Liu. Surrogate gap minimization improves sharpness-aware training. In *International Conference on Learning Representations*, 2022.

[51] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Table 4: Model performance when pre-trained on JFT-300 datasets. Those numbers correspond to Figure 1 in the main text. The fine-tuning resolution is 384^2 for ViT-B/16 and ViT-L/16, and 392^2 for ViT-H/14. Following Dosovitskiy et al. [12], Polyak averaging is not applied here.

Model	#Params	Epochs / Steps	Optimizer	ImageNet	ReaL	V2	A	R
ViT-B/16$_{384}$	86.86M	7 / 517,791	AdamW	84.24	89.04	74.89	27.39	53.71
			Hero-Lion	84.72	89.14	75.83	29.65	55.86
ViT-L/16$_{384}$	304.72M	7 / 517,791	AdamW	86.69	89.95	78.03	40.55	64.47
			Hero-Lion	87.32	90.43	79.29	47.13	68.49
		14 / 1,035,583	AdamW	87.29	90.11	78.91	42.56	64.34
			Hero-Lion	88.09	90.62	80.48	51.55	70.72
ViT-H/14$_{392}$	632.72M	14 / 1,035,583	AdamW	88.02	90.27	80.10	53.14	69.48
			Hero-Lion	88.78	90.68	81.41	58.21	73.09

A Detailed Algorithms

We manually include a bias correction term in Hero-Lion and show the full version in Algorithm 2, where θ_t is the models weights at step t, λ is the weight decay factor, η is the learning rate scheduler and η_t denotes the learning rate at step t, f is the target function to optimize. Empirically, the bias correction has minor effects for both AdamW and Hero-Lion.

Algorithm 1 AdamW Optimizer

```
given $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $\lambda$, $\eta$, $f$
initialize $\theta_0$, $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$
while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
    Update EMA of $g_t$ and $g_t^2$
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
    Bias Correction
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
    Update
    $\theta_t \leftarrow \theta_{t-1} - \eta_t(\hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) + \lambda\theta_{t-1})$
end while
return $\theta_t$
```

Algorithm 2 Hero-Lion Optimizer (ours)

```
given $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\lambda$, $\eta$, $f$
initialize $\theta_0$, $m_0 \leftarrow 0$, $t \leftarrow 0$
while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
    Bias Correction
    $\hat{m}_{t-1} \leftarrow m_{t-1} / (1 - \mathbb{1}_{t>1}\beta_2^{t-1})$
    Update
    $\hat{m}_t \leftarrow \beta_1 \hat{m}_{t-1} + (1 - \beta_1) g_t$
    $\theta_t \leftarrow \theta_{t-1} - \eta_t(\mathrm{sign}(\hat{m}_t) + \lambda\theta_{t-1})$
    Update EMA of $g_t$
    $m_t \leftarrow \beta_2 m_{t-1} + (1 - \beta_2) g_t$
end while
return $\theta_t$
```

B Hyperparameters

Hyperparameter settings are shown in Table 5, where lr denotes the peak learning rate, λ denotes the weight decay factor. The effective weight decay $lr * \lambda$ that actually applies to the weights is similar for AdamW and Hero-Lion. We fix all other hyperparameters in the optimizers, e.g., $\beta_1 = 0.9$, $\beta_2 = 0.999$ for AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.99$ for Hero-Lion. The fine-tuning settings are exactly the same as the original ViT [12, 46] and CoAtNet [11] paper. We use TPU V3 and V4 for all the experiments.

Table 5: Hyperparameter settings for the experiments. The effective weight decay $lr * \lambda$ is similar for AdamW and Hero-Lion.

Model	Dropout	Stoch Depth	Augmentations	Optimizer	lr	λ
Train from scratch on ImageNet-1K						
ResNet-50	-	-	-	AdamW	$3e-3$	0.1
				Hero-Lion	$3e-4$	1.0
Mixer-S/16	-	0.1	-	AdamW	$1e-2$	0.3
				Hero-Lion	$3e-3$	1.0
Mixer-B/16	-	0.1	-	AdamW	$1e-2$	0.3
				Hero-Lion	$3e-3$	3.0
ViT-S/16	0.1	0.1	-	AdamW	$1e-2$	0.1
				Hero-Lion	$1e-3$	1.0
			RandAug: 2, 10	AdamW	$3e-3$	0.1
			Mixup: 0.2	Hero-Lion	$3e-4$	0.3
ViT-B/16	0.1	0.1	-	AdamW	$3e-3$	0.3
				Hero-Lion	$1e-3$	1.0
			RandAug: 2, 15	AdamW	$1e-3$	0.1
			Mixup: 0.5	Hero-Lion	$3e-4$	1.0
CoAtNet-1	-	0.3	RandAug: 2, 15	AdamW	$1e-3$	0.05
			Mixup: 0.8	Hero-Lion	$2e-4$	1.0
CoAtNet-3	-	0.7	RandAug: 2, 15	AdamW	$1e-3$	0.05
			Mixup: 0.8	Hero-Lion	$2e-4$	1.0
Pre-train on ImageNet-21K						
ViT-B/16	0.1	0.1	-	AdamW	$1e-3$	0.1
				Hero-Lion	$1e-4$	0.3
ViT-L/16	0.1	0.1	-	AdamW	$1e-3$	0.3
				Hero-Lion	$1e-4$	1.0
Pre-train on JFT-300M						
ViT-B/16	-	-	-	AdamW	$6e-4$	0.1
				Hero-Lion	$1e-4$	0.3
ViT-L/16	-	-	-	AdamW	$3e-4$	0.1
				Hero-Lion	$1e-4$	0.3
ViT-H/14	-	-	-	AdamW	$3e-4$	0.1
				Hero-Lion	$3e-5$	0.3
Pre-train on JFT-3B						
ViT-g/14 & ViT-G/14	-	-	-	AdafactorW	$8e-4$	0.03
				Hero-Lion	$3e-5$	0.3

Table 6: Training error L_{train} and landscape flatness $L^{\mathcal{N}}_{train}$ of ViT-B/16.

Optimizer	AdamW	Hero-Lion
ImageNet	75.48	77.44
ReaL	80.64	82.57
V2	61.87	64.81
L_{train}	0.61	0.75
$L^{\mathcal{N}}_{train}$	3.74	1.37

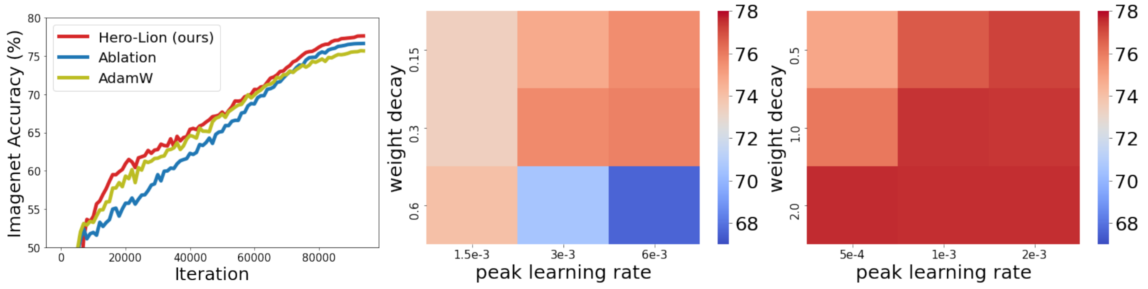

Figure 2: **Left**: Ablation for β_1 and β_2 in Hero-Lion. ImageNet accuracy (%) of ViT-B/16 when we vary lr and λ for AdamW (**Middle**) and our Hero-Lion (**Right**).

C Sensitivity to Hyperparameters

Apart from the generalization ability, the sensitivity to hyperparameters is also critical for the adoption of an optimizer in practice. Compared to AdamW, Hero-Lion does not need the hyperparameter ϵ for numerical stability. In Figure 2 (middle and right), we alter both lr and λ when training ViT-B/16 from scratch on ImageNet. As shown by the heatmaps, Hero-Lion is more robust to different hyperparameter choices compared to AdamW, benefiting its practical usage.

D Analysis of Loss Landscape

In this section, we try to understand the reason why our Hero-Lion optimizer achieves better generalization than AdamW from the lens of loss geometry. The convergence to a smooth landscape has been shown to benefit the generalization of deep neural networks [9, 14, 21, 50]. Following Chen et al. [9], we measure the landscape flatness at convergence by $L^{\mathcal{N}}_{train} = \mathbb{E}_{\epsilon\sim\mathcal{N}}[L_{train}(w+\epsilon)]$ (average over 1,000 random noises) in Table 6. We observe that the ViT-B/16 trained by AdamW enjoys a smaller training error L_{train}. However, Hero-Lion can enable ViT to converge to flatter regions, as it helps the model retain comparably lower error against Gaussian perturbations.