GraViT-E: Gradient-based Vision Transformer Search with Entangled Weights

Rhea Sukthanker[1], **Arjun Krishnakumar**[1], **Sharat Patil**[1], **Frank Hutter**[1,2]
[1]University of Freiburg,
[2]Bosch Center for AI

Abstract

Differentiable one-shot neural architecture search methods have recently become popular since they can exploit weight-sharing to efficiently search in large architectural search spaces. These methods traditionally perform a continuous relaxation of the discrete search space to search for an optimal architecture. However, they suffer from large memory requirements, making their application to parameter-heavy architectures like transformers difficult. Recently, single-path one-shot methods have been introduced which often use *weight entanglement* to alleviate this issue by sampling the weights of the sub-networks from the largest model, which is itself the supernet. In this work, we propose a continuous relaxation of weight entanglement-based architectural representation. Our *Gradient-based Vision Transformer Search with Entangled Weights* (GraViT-E) combines the best properties of both differentiable one-shot NAS and weight entanglement. We observe that our method imparts much better regularization properties and memory efficiency to the trained supernet. We study three one-shot optimizers on the Vision Transformer search space and observe that our method outperforms existing baselines on multiple datasets while being upto 35% more parameter efficient on ImageNet-1k.

1 Introduction

One-shot neural architecture search (NAS) methods [18, 27, 4, 8] have recently gained a lot of popularity across multiple computer vision domains like classification [18], semantic segmentation [16, 21] and super resolution [5, 23]. Most of these methods adopt the weight sharing paradigm [24] for efficient search. Differentiable NAS methods based on weight-sharing have often been criticized for their *large memory consumption*, with the size of the required working memory increasing linearly with the number of operator choices. Hence these methods often (have to) perform search on smaller network proxies. One-shot NAS methods, such as SPOS [11], AutoFormer [3] and OFA [2, 22] alleviate this problem by using *weight-entanglement*, which further enforces sharing of network weights between the operator choices. These methods first train the supernet by sampling a subnetwork and updating its weights in every epoch. They then use black-box optimizers to select the best architecture from the trained supernet. The black-box search is based upon the surrogate performance of the architectures inherited from the supernet which could potentially have a poor rank correlation with the actual subnetwork performance, hence biasing the search. Motivated by this observation, we combine the continuous relaxation of the architecture space, which is widely studied for differentiable architecture search [19, 8, 4] with weight-entanglement from AutoFormer [3] to directly search for optimal architectures in the transformer search space in a differentiable manner.

We observe that our method (GraViT-E) obtains architectures which outperform the baseline models in terms of accuracy while maintaining parameter and FLOP efficiency. Further our proposed method can be integrated with any off-the-shelf one-shot optimizer and is general enough to be adapted to any new search space design.

6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans.

Our Contributions. We outline our main contributions below:

- We propose a search space for Vision Transformers [9] that combines weight-entanglement with the continuous relaxation of the architectural space as seen in gradient-based one-shot NAS methods.
- We study popular one-shot optimizers like DARTS [17], DrNAS [4] and GDAS [8] applied in conjunction with weight-entanglement. Our method Pareto-dominates the AutoFormer baseline models on CIFAR10, CIFAR100 and ImageNet-1k while improving the accuracy by 0.13% and 2.79 % and 1.5% respectively.

2 Related Work

Neural Architecture Search [10] aims at automating the design of network architectures. ENAS [20] was the first method to apply *weight-sharing*, which forced all the architectures in the search space to share the parameters by defining a computational super-graph, the sub-graphs of which represented individual models. Using this weight-sharing, ENAS reduced the computational cost of NAS by $1000\times$ compared to previous methods.

DARTS [19] combined one-shot architecture search with a gradient-based approach. It formulated the search as a bi-level optimization problem that trained the parameters of the super-net on the training set in the inner loop while training the *architectural parameters* of the super-net on the validation set in the outer loop. These architectural parameters are later used to *discretize* the super-net to get the most promising model. DARTS, however, had prominent *failure modes*, as outlined in Robust-DARTS [27]. Several other gradient-based one-shot optimization techniques have since been developed to improve upon DARTS, such as GDAS [8] and DrNAS [4]. GDAS samples one path through the entire super-graph in every training step using differentiable *Gumbel-softmax sampling*. DrNAS, on the other hand, re-frames the one-shot search as learning a Dirichlet distribution.

Lastly, our work is closely related to AutoFormer [3], which introduced *weight-entanglement* as a means of restricting memory consumption. Our approach performs a continuous relaxation of the weight entanglement search space by adding architectural parameters to it which allows the use of gradient-based one-shot optimizers.

3 Methodology

3.1 Weight Sharing and Entanglement

Weight Sharing. We first review *weight-sharing* [24], which is used in many one-shot optimization techniques. The search space in weight-sharing NAS methods consists of a single *supernet*, each edge of which holds several different *operators*, such as convolutional or pooling layers. Each of these operators has an associated weight matrix, and these weight matrices are shared: all (exponentially many) architectures that include a certain operator use the same shared weight matrix. That is, by updating one path of the supernet, exponentially many other paths are also (partially) updated.

Weight-sharing comes with a set of advantages and disadvantages. On one hand, it allows a large number of models to be trained and searched at the expense of training a single supernet [19]. This has drastically increased the speed and efficiency of NAS [1]. On the other hand, the memory requirements for training the supernet using differentiable optimizers like DARTS with weight sharing increases linearly with the number of choice of operators on each edge, making it infeasible to train very large models.

Weight Entanglement. While *weight-sharing* shares the operator parameters between different architectures, *weight-entanglement* goes one step further and shares the parameters between the different operators on a given edge of the supernet. E.g., the parameters of a smaller convolutional kernel are sampled from the largest convolutional kernel on the same edge. The weights are hence shared between different sets of operators as well as different architectures sampled from the supernet.

Weight-entanglement has several advantages. Primarily, the memory requirement for the entire supernet is exactly the same as for the largest architecture in the search space, making it feasible to

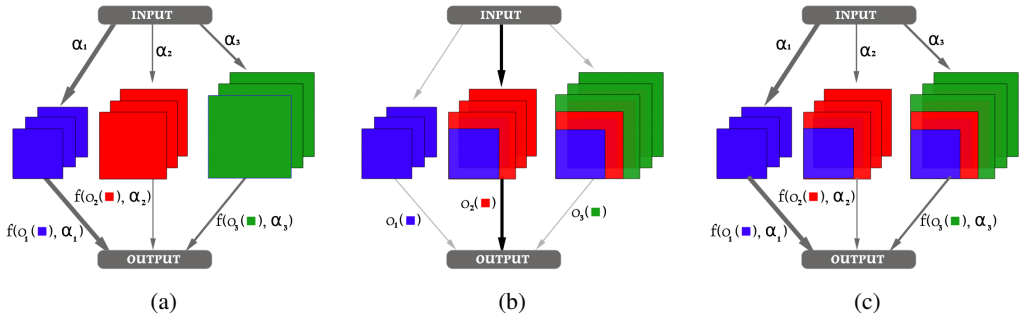

Figure 1: An overview of the our method (c), compared against gradient-based one-shot optimization (a) and weight entanglement (b). In one-shot optimization, the operators have different weights (represented here with square tensors of different colors) and an additional architectural *alpha* parameter for each operator. Weight entanglement samples the parameters of the smaller operators from the larger ones, and have no alpha parameters. Our method uses both weight entanglement and architectural parameters, which are then learned by the gradient-based one-shot optimizers.

explore much larger architectures than what standard weight-sharing permits. Weight-entanglement also imparts the supernet with several of the beneficial properties of the once-for-all networks [2, 22].

3.2 Our Approach

Our approach combines weight-entanglement with gradient-based one-shot NAS. The operators on any given edge reuse the parameters of the largest operator on the edge, as in weight-entanglement, but we also learn architectural parameters for each of these operators, as in gradient-based one-shot NAS. This allows us to apply one-shot architectural optimizers such as DARTS, GDAS and DrNAS to search spaces that leverage weight-entanglement (in our experiments, the AutoFormer search space [3]). See Figure 1 for an overview. Unlike in AutoFormer, we can now acquire the optimal architecture directly by discretizing the supernet using the learned architectural parameters.

Search Space. Our experiments were run on the AutoFormer search space [3]. This is a transformer-based search space that varies five factors in its building blocks: embedding dimension α_e (dimension of the patch embeddings which are fed into the transformer encoders) , *Q-K-V* dimension (dimensions of the query, key and value vectors of the Multi-head Self Attention layer), number of heads in the MSA α_h, MLP ratio α_r (the ratio of hidden dimension to the embedding dimension in the multi-layer perceptron), and network depth α_n (number of transformer encoder blocks stacked together) . The AutoFormer search space also fixes the ratio of *Q-K-V* dimension to the number of heads hence it is not a part of the search space.

Combination Mixture Operation For each of the building blocks of the vision transformer, we single out one or more of the above factors it depends on. The main difference we observe from traditional gradient based optimization is that some operations depend on two or more $\alpha's$ which we call *combination mixop*. Consider the Multiheaded Self Attention (MSA) block, which depends upon the embedding dimension α_e and number of heads α_h. We define the architectural weights for this operation as the combination of these two choices, i.e., $\alpha_{e,h}$. We refer the reader to Appendix A.3 for further details on the *combination mixop*.

4 Experiments and Results

Image Classification on CIFAR10 and CIFAR100. As our first experiment, we study three gradient-based one-shot optimizers (DARTS, GDAS and DrNAS) on the AutoFormer search space. For each of these optimizers, we first train the supernet with the optimizer for search and then obtain the final architecture by discretizing the supernet; this architecture is then evaluated by finetuning as well as by retraining from scratch. The search and training is performed directly for CIFAR-10 and CIFAR-100 [15] are shown in Table 1 and Table 2.

	Accuracy		Params (MB)		FLOPS (G)	
Optimizer	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
AutoFormer	98.126 ± 0.170	80.130 ± 1.110	8.739	8.817	2.676	2.660
Ours + DARTS	98.243 ± 0.017	82.517 ± 0.454	8.631	9.456	2.647	2.860
Ours + DrNAS	$\mathbf{98.253} \pm 0.051$	82.503 ± 0.284	7.220	8.761	2.285	2.682
Ours + GDAS	98.153 ± 0.023	$\mathbf{82.92} \pm 0.181$	**6.317**	**5.845**	**2.055**	**1.902**

Table 1: Comparison of our methods with AutoFormer (patch size=2,image size=32)

Table 1 shows that the models from our methods Pareto-dominate the models found by AutoFormer, performing especially well in terms of model size and FLOPS. Interestingly, as Table 2 in Appendix B, our finetuned models are better than training from scratch indicating that our method imparts some regularization properties to the supernet, which require further investigation.

Further, as observed in Table 1 the models discovered by our method with the DARTS, DrNAS and GDAS optimizers achieve a better trade-off between error-rate, number of parameters and FLOPS than the Autoformer.

	CIFAR-10		CIFAR-100	
Optimizer	Finetune Accuracy	Retrain Accuracy	Finetune Accuracy	Retrain Accuracy
Ours + DARTS	98.25	98.03	82.98	79.67
Ours + DrNAS	98.29	97.96	82.59	81.08
Ours + GDAS	98.17	98.09	83.12	80.76

Table 2: Fine-tuning v/s retraining the best models for (patch size=2,image size=32)

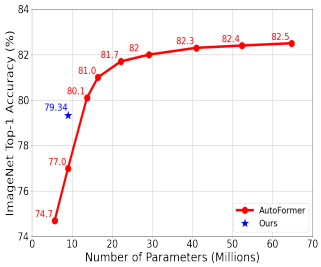

Figure 2: Our method compared to AutoFormer on ImageNet-1k

For our second experiment, we search for the best architecture on CIFAR10 and CIFAR100 (img size = 224, patch size = 16) Table 3 in Appendix C. We choose the best model that we obtained from across all the optimizers on CIFAR-100 (the model obtained using GDAS, in this case), and transfer its weights to a model which is then fine-tuned to perform classification on ImageNet-1k [7].

We achieve an accuracy of 79.34% on ImageNet-1k, compared to AutoFormer which achieves an accuracy of 74.7%. Further, as noted, from Figure 2, we obtain improvements of up to 1.5% on ImageNet-1k compared to AutoFormers of comparable model sizes.

5 Conclusion

We present a new framework GraViT-E to efficiently search for the architectural parameters of large models like transformers. Our method outperforms the AutoFormer search strategy on multiple datasets in terms of performance and parameter-efficiency. In the future, we plan to extend our work to support multiple search spaces (e.g. different transformer types [12] and convolutional networks [14, 13]) and multiple one-shot optimizers (e.g., PC-DARTS [26], SNAS [25], FairNAS [6]).

While our work initiates the study of continuous relaxation of the transformer search space for gradient-based one-shot optimizers, we still face some limitations. For now, our observations are based upon the study of only a lightweight Vision Transformer search space. Further we do not study different transformer applications like language modeling, semantic image segmentation, etc. How the weight entanglement scheme hampers or benefits the gradient based search also warrants further investigation. However, since our formulation can take advantage of the continually-improving set of methods for gradient-based one-shot NAS, we hope that our work will lead to the first generally-applicable robust and efficient architecture optimization method for transformers.

Acknowledgments

This research was partially supported by the following sources:

TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215; the German Federal Ministry of Education and Research (BMBF, grant RenormalizedFlows 01IS19077C); the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant number 417962828; the European Research Council (ERC) Consolidator Grant "Deep Learning 2.0" (grant no. 101045765). Robert Bosch GmbH is acknowledged for financial support. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the ERC. Neither the European Union nor the ERC can be held responsible for them.

Funded by the European Union

References

[1] G. Bender, P-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le. Understanding and simplifying one-shot architecture search. In *Proc. of ICML'18*, 2018.

[2] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[3] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12270–12280, 2021.

[4] X. Chen, R. Wang, M. Cheng, X. Tang, and C-J. Hsieh. Dr{nas}: Dirichlet neural architecture search. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=9FWas6YbmB3.

[5] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search. In *2020 25th International conference on pattern recognition (ICPR)*, pages 59–64. IEEE, 2021.

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

[7] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. of CVPR'09*, pages 248–255, 2009.

[8] X. Dong and Y. Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[10] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[11] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European conference on computer vision*, pages 544–560. Springer, 2020.

[12] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Brett Koonce. Mobilenetv3. In *Convolutional Neural Networks with Swift for Tensorflow*, pages 125–144. Springer, 2021.

[15] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[16] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 82–92, 2019.

[17] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJQRKzbA-.

[18] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

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

[20] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, 2018.

[21] Albert Shaw, Daniel Hunter, Forrest Landola, and Sammy Sidhu. Squeezenas: Fast neural architecture search for faster semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[22] Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. Hat: Hardware-aware transformers for efficient natural language processing. *arXiv preprint arXiv:2005.14187*, 2020.

[23] Yan Wu, Zhiwu Huang, Suryansh Kumar, Rhea Sanjay Sukthanker, Radu Timofte, and Luc Van Gool. Trilevel neural architecture search for efficient single image super-resolution. *arXiv preprint arXiv:2101.06658*, 2021.

[24] Lingxi Xie, Xin Chen, Kaifeng Bi, Longhui Wei, Yuhui Xu, Lanfei Wang, Zhengsu Chen, An Xiao, Jianlong Chang, Xiaopeng Zhang, et al. Weight-sharing neural architecture search: A battle to shrink the optimization gap. *ACM Computing Surveys (CSUR)*, 54(9):1–37, 2021.

[25] S. Xie, H. Zheng, C. Liu, and L. Lin. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rylqooRqK7.

[26] Y. Xu, L. Xie, X. Zhang, X. Chen, G. Qi, Q. Tian, and H. Xiong. PC-DARTS: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

[27] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gDNyrKDS.

A Appendix

A.1 Pareto Fronts

In Figure 3 we present the parameter-size and error Pareto front for AutoFormer v/s our models with different one-shot optimizers. We observe that GDAS pareto dominates other models for CIFAR100 and GDAS and DrNAS lie on the Pareto front for CIFAR10.

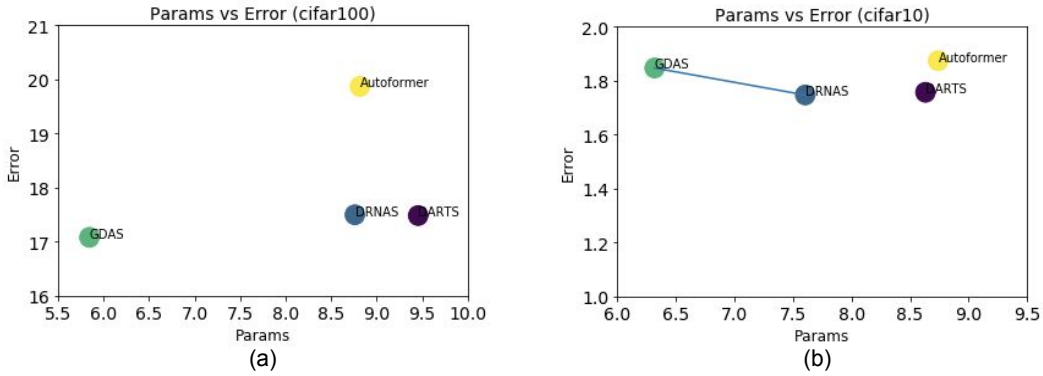

Figure 3: Parameter size-Error Pareto front of Autoformer and our models (a) CIFAR10 (b) CIFAR100

A.2 Experimental Details

We use the official source code of AutoFormer available at https://github.com/microsoft/Cream/tree/main/AutoFormer for all the AutoFormer experiments on CIFAR10 and CIFAR100. We closely follow the AutoFormer training pipeline and search space design. AutoFormer searched on three transformer sizes Autoformer-T (tiny), AutoFormer-S (small) and Autoformer-B (base). We currently restrict ourselves to Autoformer-T. We use a 50%-50% train and validation split for the CIFAR10 and CIFAR100 dataset for the bi-level optimization.

A.3 Optimization Details.

We assign an architectural parameter α to each of these variables, α_d for embedding dimension, α_h for number of heads, α_r for MLP-ratio and α_n for the number of transformer encoder blocks to make the application of one-shot optimizers on the AutoFormer search space feasible. Next, for each of the building blocks of the vision transformer, we single out one or more of the above factors it depends on. We observe two categories of *mixture* operations, first which depend on only one of the above factors (*single mixop*) and a second one which is based upon combination of two (or more) $\alpha's$ which we call *combination mixop*. In *single mixops*, we use a mixture operation which is compatible with the specific one-shot optimizer that we use. For the *combination mixop*, let us consider the Multiheaded Self Attention (MSA) block, which depends upon the embedding dimension α_e and number of heads α_h. We define the architecture weights for this operation as the combination of these two choices, i.e., $\alpha_{e,h}$, which is defined as below. The mixture operation for MSA is then defined as in Equation (1), where $\mathcal{O}$ is the cross-product of the two sets of operation choices (here embedding dimension and number of heads). The formulation ensures that the ranking of the possible independent choices (α_e and α_h) and the choice for the combination operation ($\alpha_{e,h}$) is not conflicting with each other.

$$
\begin{aligned}
embedding_alphas &= \{\alpha_{e_1}, \alpha_{e_2}, ...\alpha_{e_n}\} \\
head_alphas &= \{\alpha_{h_1}, \alpha_{h_2}, ...\alpha_{h_m}\} \\
\alpha_{e_i,h_j} &= softmax(\alpha_{e_i}) * softmax(\alpha_{h_i})
\end{aligned}
$$

$$
MSA_mix(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} softmax(\alpha_{e_i,h_j})\mathcal{O}_{e_i,h_j}(x) \tag{1}
$$

B Finetuning v/s retraining

In Table 2 we compare the accuracy of the obtained models from search on CIFAR100 and CIFAR10 in terms of their retraining and finetuning accuracies. Surprisingly inheriting the weights from the supernetwork and finetuning obtains much better accuracies compared to retraining from scratch. We

hypothesize this to be the regularization effect imparted by the supernet training scheme and needs to be studied further.

C CIFAR10 and CIFAR100 with image size=224 and patch size=16

For architectures to be transferrable to ImageNet we need to perform train and search on CIFAR100 and CIFAR10 with the 32×32 image padded to 224×224 and further using a patch size of 16×16. The results obtained on CIFAR10 and CIFAR100 on these patch sizes is as presented in Table 3. We observe that similar to the 2×2 patch size studies in Section 4. GDAS performs the best on CIFAR100 and its weights are then transferred to ImageNet-1k.

	Accuracy		Params (MB)		FLOPS (G)	
Optimizer	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
Ours + DARTS	98.043 ± 0.005	82.116 ± 0.167	**7.641**	9.969	**1.756**	2.214
Ours + DrNAS	$\mathbf{98.170} \pm 0.053$	81.826 ± 0.172	9.024	9.854	2.028	2.192
Ours + GDAS	97.810 ± 0.037	$\mathbf{82.786} \pm 0.079$	8.322	**9.048**	1.871	**2.032**

Table 3: Comparison of our methods with AutoFormer (patch size=16,image size=224)