PSEUDO-INVERTED BOTTLENECK CONVOLUTION FOR DARTS SEARCH SPACE

Arash Ahmadian[1], Yue Fei[1], Louis S.P. Liu[1], Konstantinos N. Plataniotis[1], Mahdi S. Hosseini[2]

[1]The Edward S. Rogers Sr. Department of Electrical & Computer Engineering, University of Toronto, Canada
[2]Computer Science and Software Engineering (CSSE), Concordia University, Canada

ABSTRACT

Differentiable Architecture Search (DARTS) has attracted considerable attention as a gradient-based Neural Architecture Search (NAS) method. Since the introduction of DARTS, there has been little work done on adapting the action space based on state-of-art architecture design principles for CNNs. In this work, we aim to address this gap by incrementally augmenting the DARTS search space with micro-design changes inspired by ConvNeXt and studying the trade-off between accuracy, evaluation layer count, and computational cost. To this end, we introduce the Pseudo-Inverted Bottleneck conv block intending to reduce the computational footprint of the inverted bottleneck block proposed in ConvNeXt. Our proposed architecture is much less sensitive to evaluation layer count and outperforms a DARTS network with similar size significantly, at layer counts as small as 2. Furthermore, with less layers, not only does it achieve higher accuracy with lower GMACs and parameter count, GradCAM comparisons show that our network is able to better detect distinctive features of target objects compared to DARTS.

1. INTRODUCTION

Since the introduction of Vision Transformers (ViTs) by *Dosovitskiy et al.* [1], a new class of research has emerged, pushing the boundaries of Transformer-based architectures on a variety of computer vision tasks [2, 3, 4, 5]. These advances make it seem inevitable that ViTs would overtake conventional Convolutional Neural Networks (CNNs). Recently, *Liu et al.*'s ConvNeXt [6] has sparked a resurgence in further exploring the architectural designs of CNNs in image recognition. Specifically, they argued that by adapting components from Transformers into the standard ResNet backbone [7], the trained models can match or outperform state-of-the-art ViTs in image classification, objection detection, and segmentation. If CNNs can still be improved by design elements that were previously overlooked, this begs the question: *Can we apply the same Transformer principles to a Neural Architecture Search (NAS) framework to improve its performance?*

NAS has historically seen immense success on large-scale image classification prior to ViTs [8, 9, 10] as it alleviates the task of manually designing for the optimal neural network architecture. Early works of NAS employed Reinforcement Learning [11], Evolutionary Search [12], and Bayesian Op-

timization [13] while more recent works have shifted to the One-Shot NAS paradigm [14], which leverages weight-sharing of models within a supernet to reduce computation time.

One popular branch stream of NAS is Differentiable Architecture Search (DARTS) [15]. This method relaxes the search space from discrete to continuous by attributing weights to operations sampled from set and using a *Softmax* function to choose the best candidate. This enables end-to-end training using common optimizers such as Stochastic Gradient Descent. Many works have investigated ways of improving the NAS operation space using methods such as: increasing the granularity of operations by breaking down search units across input convolution channels [16], grouping similar operations to combat the effects of multi-collinearity [17], creating more expressive operations by replacing the DFT matrices in convolution's diagonalization with K-matrices [18], and reducing the operation set [19]. In this work, we investigate optimizations to the search space through a different set of lens by drawing inspiration from ConvNeXt.

We start with the second-order DARTSV2 cell (vanilla) structure and incrementally augment the search operations by adapting design elements from ConvNeXt. For each stage, we conduct search and evaluation phases on CIFAR-10 [20] using the same training setup and hyper-parameters as DARTS [15]. In our experiments, we encountered a large increase in parameter count when directly adopting the ConvNeXt convolution block with hindering performances. To combat this, we also propose a new Pseudo-Inverted Bottleneck structure to incorporate an inverted bottleneck while minimizing model size. Our proposed architecture is much less sensitive to evaluation layer count and achieves better test error than the original DARTSV2 with comparable parameter count and computations. We further demonstrate its effectiveness by performing a GradCAM [21] analysis, showing that it is able to capture prominent image features at 10 layers vs. a 20-layer DARTSV2.

Our contributions are summarized as follows:

[**C1.**] We present an incremental experiment procedure to evaluate how design components from ConvNeXt impact the performance of DARTS by redesigning its search space.

[**C2.**] We introduce a Pseudo-Inverted Bottleneck block to implement an inverted bottleneck structure while minimizing model footprint and computations. This outperforms vanilla DARTSV2 with lower number of layers, parameter count, and GMACs.

2. METHODOLOGY

Our approach to modernizing the DARTS operation set involves incrementally making micro-changes to the design of the separable convolution block used within DARTS. However, not all changes proposed in ConvNeXt can be transferred to DARTS. (1) *Changing the stage compute ratio* to match that of the Swin Transformer [3] is not applicable as it would require major restructuring of the DARTS framework (*i.e.* changing the placement of reduction cells) which is beyond our scope of updating the operation set. (2) *Modifying the stem cell* to mimic the "patchify" operation in Swin is not applicable since a $4\times$ downsampling is too aggressive for the 32×32 images in CIFAR-10. With every change, we search for a cell structure (or *genotype*), under hyper-parameter settings described in Section 4 and evaluate on different layer counts (1). We compare the highest achieved accuracies and corresponding GMACs. Below we present this exploration step by step.
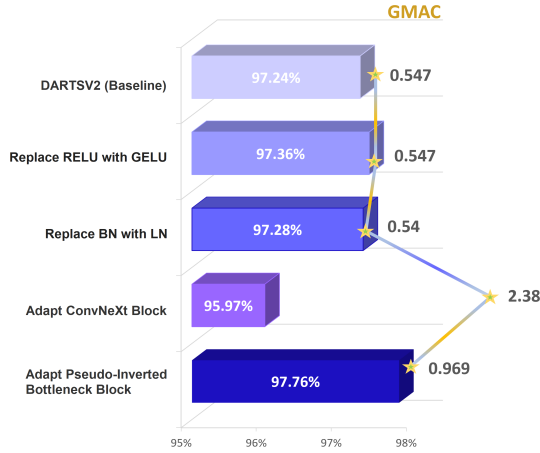

Fig. 1: Roadmap of the incremental augmentations described in Section 3, along with their corresponding accuracies and methodologies.
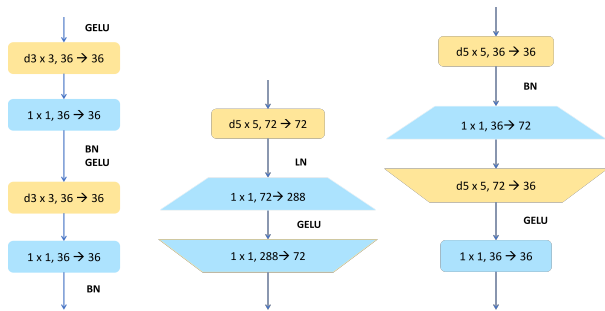

Fig. 2: Convolution Blocks : **(a)** DARTS Separable Convolution Block; **(b)** Inverted Bottleneck ConvNeXt Convolution Block ($C_{inv} = C \times 4$); **(c)** Pseudo-Inverted Bottleneck Cell ($C_{inv} = C \times 2$)

Replacing ReLU with GELU We replace the widely used ReLu [22] activation with GELU [23] which provides an approximation of the former with the key distinction that a small portion of negative signals are let through to the next layer. *This boosts the accuracy by* 0.12% *and from now on we use GELU instead of ReLU.*

Replacing BatchNorm with LayerNorm There have been multiple attempts to develop an alternative to normalization however it remains a key ingredient in modern NN design [24]. In ConvNeXt, replacing BN with LN slightly improves the accuracy of the network. We replace BatchNorm [25] with LayerNorm [26] in our separable convolution operation. Initially, this results in minor degradation in accuracy. We also experiment with retaining LN and adding the various micro-changes proposed in this section. We did not achieve a performance boost from LN in any setting. *We will use BN instead of LN.*

Adapting the ConvNeXt Block Vanilla DARTS uses depthwise separable convolution as popularized by Xception [27]. The stacked topology used in DARTS is depicted in Fig. 2a. However, the inverted bottleneck popularized by MobileNetV2 [28] has made its way to multiple modern networks [8, 29] and thus warrants exploration in the DARTS framework. We implement the ConvNeXt block structure in Fig. 2b. It consists of three key changes: (1) Reducing the number of activation and normalization functions, (2) Adapting to an inverted bottleneck structure, and (3) Moving up the depthwise separable conv layer to facilitate training with large kernel sizes. However, directly adapting the ConvNeXt block significantly increases the number of parameters and GMACs while sharply decreasing accuracy.

To manage the number of learnable parameters, we introduce the Pseudo-Inverted Bottleneck block as depicted in Fig. 2. We add a depthwise convolution after the intermediate pointwise conv layer which reduces the number of channels. We keep the positions of the activation and normalization the same relative to the next layer based on the ConvNeXt block. This structure also inhibits the stacked architecture which has been shown to increase accuracy by $1-2\%$ when introduced to separable convolution-based operations in the DARTS framework [15] (which the vanilla inverted bottleneck does not have), as well as an inverted bottleneck structure.

We compare the number of weights per block to estimate the parameter size and computational complexity of both networks. Define C to be the input and output channel size, C_{inv} to be the inverted bottleneck channel size, and K to be the kernel size of the depthwise convolution. Similarly, define $F = C_{inv}/C$ to be the inverted bottleneck ratio for the first pointwise convolution. The total number of weights between the ConvNeXt block (1) and our Pseudo-Inverted Bottleneck block (2) are compared below:

$$2FC^2 + K^2C \qquad (1)$$

$$(F+1)C^2 + 2K^2C \qquad (2)$$

In practice, the dominant variable in both equations is the channel size C, which is initialized to 16 and doubled at each reduction cell. Additionally, the conv operation dominates both DARTSV2 and our searched genotypes. Thus, comparing the coefficients of the quadratic term C^2 provides an estimate for the difference in parameter size and computational complexity of these networks. Our Pseudo-Inverted Bottleneck block has approximately 0.63 times the number of weights as the ConvNeXt block. We further choose $F = 2$ in the final block topology after experimentation with various values in $\{1.5, 4.5\}$ since it achieved the best accuracy-GMAC trade-off. *The use of the Pseudo-Inverted Bottleneck block boosts the accuracy by* 0.4%.

3. EXPERIMENTS

Experimental Setup We present our hyperparameter settings and experimental setup next. Following the DARTS framework, we search with an initial channel size of 16, 4 nodes, 8 layers, 50 epochs, and a batch size of 64. We use the SGD optimizer coupled with a cosine-annealing learning rate scheduler (no restarts) [30], 0.0025 initial learning rate, $3e-4$ weight decay, and 0.9 momentum. As for the evaluation phase, we train for 600 epochs with a batch size of 96, cutout augmentation [31], path dropout with probability 0.2 and auxiliary towers with 0.4 weight. Other hyper-parameter settings remain the same as the search phase. Both our search and evaluation phases are performed on CIFAR-10.
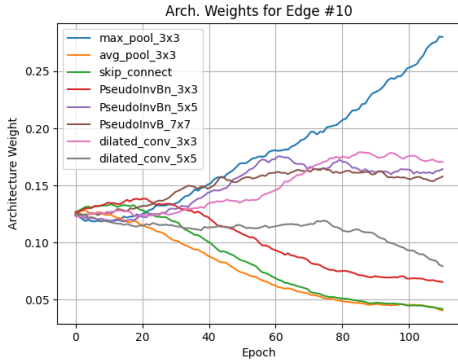

Fig. 3: Evolution of the architecture weights if searched for 115 epochs

Search Phase Our final operation set after the incremental changes described previously is comprised of the following 10 operations: *none*, *skip_connect*, *pseudo_inv_bn_3x3*, *pseudo_inv_bn_5x5*, *pseudo_inv_bn_7x7*, *dialated_conv_3x3*, *dialated_conv_5x5*, *conv_7x1_1x7*, *max_pool_3x3*, *avg_pool_3x3*. We argue that our genotype is trained to convergence with 50 epochs and avoids a common pitfall of falling back on skip-connections in later stages of training [32]. As depicted by Fig. 7, the decision boundary be-

tween the favored operation (in this case, pseudo_inv_bn_5x5) and skip-connection, is not crossed even very late into training. After searching with the mentioned hyperparameters and final operation set, we arrive at the genotype in Fig. 4.
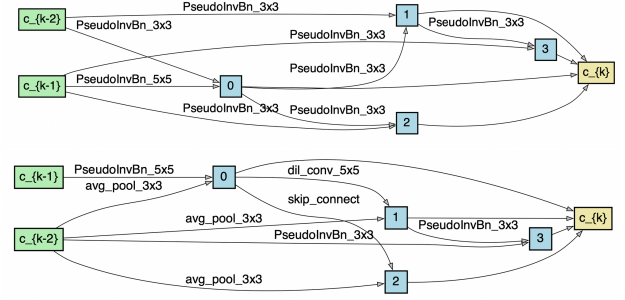

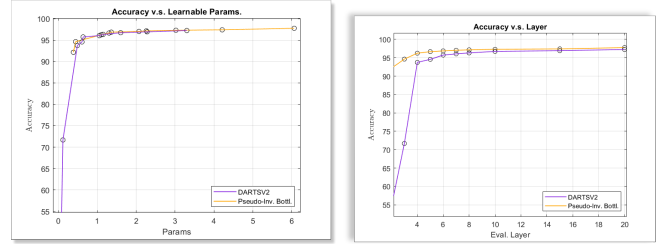**Fig. 4**: Proposed Genotype: **(a)** Normal cell; **(b)** Reduction cell

Fig. 5: Searched genotypes in comparison with DARTSV2: (a) Accuracy vs. Parameter count (b) Accuracy vs. Number of Evaluation Layers

Evaluation Phase We evaluate our final genotype at multiple evaluation layers to observe the effect of layer count on test accuracy and report the results in Table 1. We observe that the evaluation accuracy of our proposed genotype is significantly less affected by the evaluation layer count compared to DARTSV2. Specifically, at *10 layers*, we achieve a higher test accuracy compared to a 20 layer DARTSV2 network. Furthermore, at 2 layers, our architectures exceed the DARTSV2 genotype at 3 layers by over 20%, while at the same time maintaining similar GMACs. At 4 layers, we outperform the DARTSV2 genotype at 7 layers (to match the model size for a fair comparison) by 0.24%, while still maintaining lower GFLOPs. Fig. 6 presents a comparison between the Grad-CAM [21] visualizations produced from the last cell of each network for DARTSV2 at 20 layer, Our genotype at 10 and 20 layers. Our proposed genotype, in a 10 cell network, can effectively capture the prominent features of the classification. The increase in the number of cascaded cells leads to the gradual collapse of the heat-map boundaries, onto the outline of the object outperforming DARTS. We argue that this supports our claim that the proposed genotype, is inherently superior to that of DARTS.

Table 1: Performance comparison of different genotypes on CIFAR-10 dataset: Our genotype evaluated on 10 and 5 layers are highlighted to be compared with DARTSV2 genotype evaluated with 20 layers.

| Genotype	Eval. Layers	Test Acc. (%)	Params (M)	GMAC
DARTSV2	20	97.24	3.30	0.547
	15	96.93	2.28	0.408
	10	96.72	1.6	0.265
	8	96.32	1.15	0.207
	7	96.05	1.05	0.180
	6	95.73	0.635	0.153
	5	94.56	0.605	0.121
	4	93.74	0.487	0.090
	3	71.68	0.116	0.067
	2	54.52	0.082	0.035
Our Genotype	20	97.76	6.06	0.969
	15	97.40	4.21	0.724
	10	**97.29**	**3.02**	**0.470**
	8	97.15	2.26	0.369
	7	97.03	2.07	0.320
	6	96.86	1.36	0.275
	5	**96.65**	**1.30**	**0.218**
	4	96.24	1.10	0.166
	3	94.63	0.443	0.123
	2	92.15	0.385	0.067
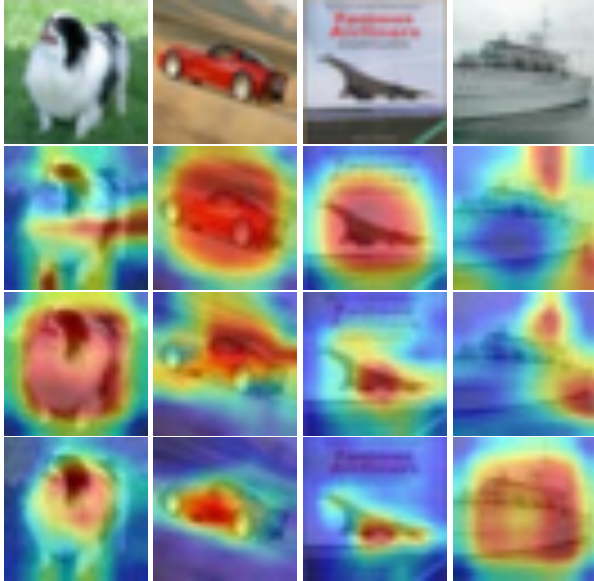
Fig. 6: GradCAM: The first row shows the 32×32 input images with labels: *dog, automobile, airplane, ship*; The second row shows DARTSV2 evaluated on 20 layers; Then third and fourth rows show our genotype evaluated on $\{10, 20\}$ layers, respectively. (Note: All of the images are up-sampled to 224×224 for better readability)
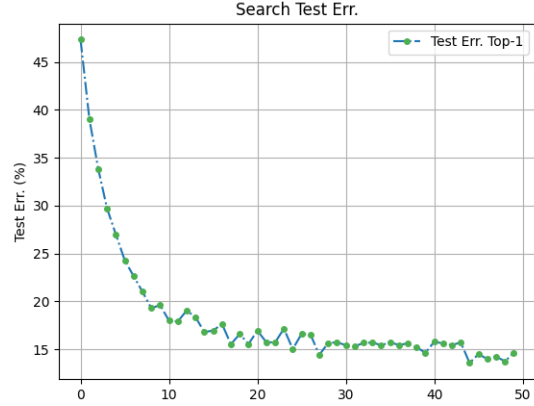

Fig. 7: Search phase of our proposed genotype: Top 1% Test Error vs Epochs

4. CONCLUSION

In this work, we attempt to revise the DARTS search space. We incrementally augment the convolution operation with micro-changes inspired by ConvNeXt and propose the Pseudo-Inverted Bottleneck block to reduce the number of parameters used in the vanilla Inverted Bottleneck. Our proposed genotype's performance is much less sensitive to evaluation layer count compared to that of DARTSV2. It achieves a higher accuracy at a lower GMAC/ parameter count with 10 evaluation layers compared to DARTSV2 evaluated at 20 layers. Furthermore, we perform a GradCAM visualization on our genotype and compare it with that of DARTSV2.

Our network's high performance at lower layer counts, correspondingly with low GMACs and parameter count, makes it an attractive choice image processing applications as sharpening and blurring, as shallow networks suit these applications best. Consequently, a potential avenue for future work would be to explore the applications of our genotype/ Pseudo-Inverted Bottleneck block, to image processing tasks.

It is worth noting that our aim in this paper was not to combat the SOTA methods related to DARTS; but shedding light on the granularity of search space which is commonly shared across many DARTS variants in the literature. We hope our work initiates new ideas to investigate optimum search space designs in DARTS framework to build more robust and generalized models for representational learning problems.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *9th International Conference*

on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021, OpenReview.net, 2021.

[2] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablay-rolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," *CoRR*, vol. abs/2012.12877, 2020.

[3] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012–10022, October 2021.

[4] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 568–578, 2021.

[5] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 558–567, 2021.

[6] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[8] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[9] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," *CoRR*, vol. abs/2104.00298, 2021.

[10] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," *CoRR*, vol. abs/1905.02244, 2019.

[11] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *CoRR*, vol. abs/1611.01578, 2016.

[12] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," *ArXiv*, vol. abs/1711.00436, 2018.

[13] J. Liu, M. Zhang, Y. Sun, B. Liu, G. Song, Y. Liu, and H. Li, "Fnas: Uncertainty-aware fast neural architecture search," 2021.

[14] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," *CoRR*, vol. abs/1904.00420, 2019.

[15] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," *CoRR*, vol. abs/1806.09055, 2018.

[16] J. Mei, Y. Li, X. Lian, X. Jin, L. Yang, A. L. Yuille, and J. Yang, "Atomnas: Fine-grained end-to-end neural architecture search," *CoRR*, vol. abs/1912.09640, 2019.

[17] G. Li, X. Zhang, Z. Wang, Z. Li, and T. Zhang, "Stacnas: Towards stable and consistent optimization for differentiable neural architecture search," *ArXiv*, vol. abs/1909.11926, 2019.

[18] N. Roberts, M. Khodak, T. Dao, L. Li, C. R'e, and A. S. Talwalkar, "Rethinking neural operations for diverse tasks," in *NeurIPS*, 2021.

[19] X. Dong and Y. Yang, "Nas-bench-201: Extending the scope of reproducible neural architecture search," *CoRR*, vol. abs/2001.00326, 2020.

[20] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[21] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

[22] A. F. Agarap, "Deep learning using rectified linear units (relu)," *CoRR*, vol. abs/1803.08375, 2018.

[23] D. Hendrycks and K. Gimpel, "Bridging nonlinearities and stochastic regularizers with gaussian error linear units," *CoRR*, vol. abs/1606.08415, 2016.

[24] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *CoRR*, vol. abs/1602.07868, 2016.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[26] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[27] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *CoRR*, vol. abs/1610.02357, 2016.

[28] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018.

[29] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," 2018.

[30] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with restarts," *CoRR*, vol. abs/1608.03983, 2016.

[31] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *CoRR*, vol. abs/1708.04552, 2017.

[32] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, and Z. Li, "DARTS+: improved differentiable architecture search with early stopping," *CoRR*, vol. abs/1909.06035, 2019.