Searching for Efficient Multi-Stage Vision Transformers

Yi-Lun Liao, Sertac Karaman, Vivienne Sze
Massachusetts Institute of Technology
{ylliao, sertac, sze}@mit.edu

Abstract

Vision Transformer (ViT) demonstrates that Transformer for natural language processing can be applied to computer vision tasks and achieve comparable results to convolutional neural networks (CNN), which have been studied in computer vision for years. This naturally raises the question of how the performance of ViT can be advanced with design techniques of CNN. In this work, we propose to incorporate two techniques and present ViT-ResNAS, an efficient multi-stage ViT architecture designed with neural architecture search (NAS). First, we propose residual spatial reduction to decrease sequence lengths for deeper layers and utilize a multi-stage architecture. Second, we propose weight-sharing NAS with multi-architectural sampling. We enlarge a network and utilize its sub-networks to define a search space. A super-network covering all sub-networks is trained for fast evaluation of their performance. To efficiently train the super-network, we sample and train multiple sub-networks with one forward-backward pass. Lastly, evolutionary search is performed to discover high-performance network architectures. Experiments on ImageNet demonstrate the effectiveness of our ViT-ResNAS.

1. Introduction

Self-attention and Transformers [42], which originated from natural language processing (NLP), have been widely adopted in computer vision (CV) tasks, including image classification [2, 19, 29, 34, 48, 61], object detection [6, 34, 62], and semantic segmentation [45, 46]. Many works utilize hybrid architectures and incorporate self-attention mechanisms into convolutional neural networks (CNN) to model long-range dependence and improve the performance of networks. On the other hand, Vision Transformer (ViT) [10] demonstrates that a pure transformer without convolution can achieve impressive performance on image classification when trained on large datasets like JFT-300M [37]. Additionally, DeiT [40] shows that ViT can outperform CNN when trained on ImageNet [32] with stronger regularization. It is appealing to have powerful Transformers for CV tasks since it enables using the same type of neural architecture for applications in both CV and NLP.

A parallel line of research is to design efficient neural networks with neural architecture search (NAS) [4, 5, 13, 18, 35, 36, 38, 44, 50, 53, 55, 56, 63, 64]. Pioneering works use reinforcement learning to design CNN architectures. They sample many networks in a pre-defined search space and train them from scratch for a few epochs to approximate their performance, which requires expensive computation. To accelerate the process, weight-sharing NAS has become popular. Instead of training individual networks, weight-sharing NAS trains a super-network whose weights are shared across all networks in the search space. Once the super-network is trained, we can directly use its weights to approximate the performance of different networks in the search space. These methods successfully result in CNN architectures outperforming manually designed ones.

While CNN architectures have been studied in CV for years and optimized with NAS, recently ViT demonstrates superior performance over CNN in some scenarios. Despite its promising results, ViT adopts the same architecture as Transformer for NLP [42]. This naturally leads to the question of how the performance of ViT can be further advanced with design techniques of CNN. Therefore, we propose to incorporate two design techniques of CNN, which are spatial reduction and neural architecture search, and present ViT-ResNAS, an efficient multi-stage **ViT** architecture with **res**idual spatial reduction and designed with **NAS**.

First, we propose *residual spatial reduction* to decrease sequence lengths and increase embedding sizes for deeper transformer blocks. As illustrated in Fig. 1, we transform the original single-stage architecture into a multi-stage one, with each stage having the same sequence length and embedding size. Additionally, we add *skip connections* when reducing sequence lengths, which can further improve performance and stabilize training deeper networks. ViT with residual spatial reduction is named ViT-Res. Second, we propose weight-sharing neural architecture search with *multi-architectural sampling* to improve the architecture of ViT-Res as shown in Fig. 2. We enlarge ViT-Res network by increasing its depth and width. Its sub-networks are utilized to define a search space. Then, a super-network cover-

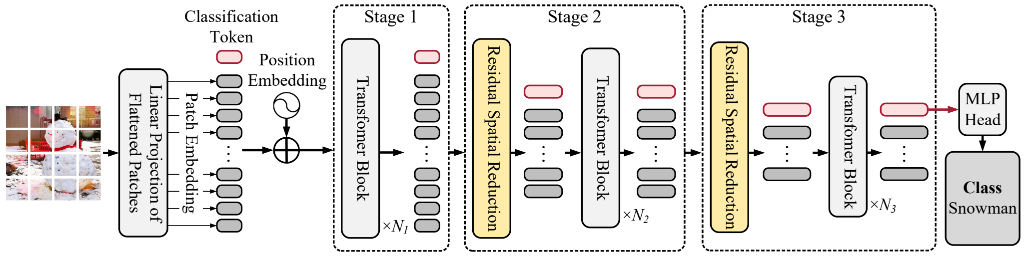

Figure 1: **Architecture of ViT-Res.** We propose residual spatial reduction (light orange) to reduce sequence length and increase embedding size for deeper blocks, which divides the network into several stages. Each stage has the same sequence length and embedding size and consists of several transformer blocks.

ing all sub-networks is trained to directly evaluate their performance. For each training iteration and given a batch of examples, we sample and train *multiple* sub-networks with *one* forward-backward pass to efficiently train the super-network. Once the super-network is trained, evolutionary search [30, 31] is applied to discover high-performance ViT-ResNAS networks. Experiments on ImageNet [32] verify the effectiveness of our proposed ViT-ResNAS as demonstrated in Appendix A.

2. Method

2.1. Residual Spatial Reduction

ViT maintains the same sequence length throughout the network. In contrast, CNNs decrease the resolution of feature maps and increase the channel size for deeper layers. The design technique has been widely adopted in high-performance CNNs [15, 28], which motivates whether it can be introduced to improve the efficiency of ViT. To this end, we propose residual spatial reduction. As illustrated in Fig. 3, since there is spatial relationship in patch embeddings, we reshape the 1D sequence into a 2D feature map and then apply layer normalization [1] and strided convolution ("Norm" and "Conv" in Fig. 3) to reduce the sequence length of patch embeddings and increase channel size (embedding dimension). Since the spatial dimension (sequence length) is changed, we update the relative position information by adding new *position embeddings*. To maintain the same channel size of all embeddings in the sequence, we apply layer normalization and a linear layer ("Norm" and "Linear" in Fig. 3) to the classification token. These constitute a *residual branch*.

Although introducing only the residual branch to ViT can significantly improve the accuracy-MAC trade-offs, training deeper networks with only the residual branch can be unstable. Specifically, under the training setting of DeiT-distill [40], using the residual branch significantly improves the accuracy of DeiT-Tiny from 74.5% to 79.6% with little increase in MACs. However, training loss can become infinity when training deeper networks such as our ViT-Res super-networks. To remedy this, we introduce an extra skip connection [15] without any learnable operations as motivated by the residual structure of transformer blocks [42, 52]. We use *2D average pooling* to reduce sequence length ("Average pooling" in Fig. 3) and concatenate embeddings with zero tensors ("Zero pad" in Fig. 3) to increase channel size. The structure results in the *main branch* and helps stabilize training and improve performance.

The residual and main branches form *residual spatial reduction*. We insert 2 residual spatial reduction blocks into ViT and divide the architecture evenly into 3 stages. Following the design rule of ResNet [15], we double embedding dimension when halving spatial resolution. The resulting ViT architecture is called ViT-Res as illustrated in Fig. 1. Please refer to Appendix C for further details.

2.2. Weight-Sharing NAS with Multi-Architectural Sampling

Another design technique used in CNNs is neural architecture search (NAS). Due to its efficiency, we adopt weight-sharing NAS to improve the archtiecture of ViT-Res in terms of designing better *architectural parameters* such as numbers of attention heads and transformer blocks.

Algorithm Overview. We enlarge ViT-Res network by increasing its depth and width. Sub-networks with smaller depths and widths are utilized to define a search space. A super-network covering all sub-networks is trained to directly evaluate their performance. For each training iteration, we propose to sample and train *multiple* sub-networks with *one* forward-backward pass to efficiently train the super-network. After that, evolutionary search is applied to discover architectures of high-performance sub-networks satisfying some resource constraints like MACs. Finally, the best sub-network evaluated becomes the searched network and is trained from scratch to convergence.

Search Space. We construct a large network by *uniformly* increasing depth and width of all stages of ViT-Res, and sub-networks contained in the large network define a search space. For *each stage*, we search *embedding dimension* and the *number of transformer blocks*. For *different blocks*, we search *different numbers of attention heads* h in multi-head self-attention (MHSA) and *different hidden sizes* d_{hidden} in feed-forward network (FFN). The range of each *searchable*

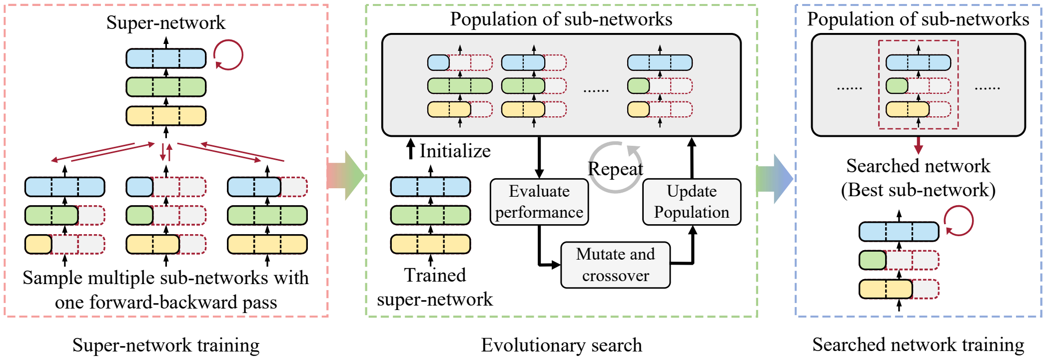

Figure 2: **Algorithm flow of NAS.** First, we train a ViT-Res super-network with multi-architectural sampling. The performance of sub-networks can be directly evaluated using the super-network's trained weights without further training. Then, we perform evolutionary search to find high-performance sub-networks. Finally, the best sub-network becomes our searched network and is trained from scratch to convergence.

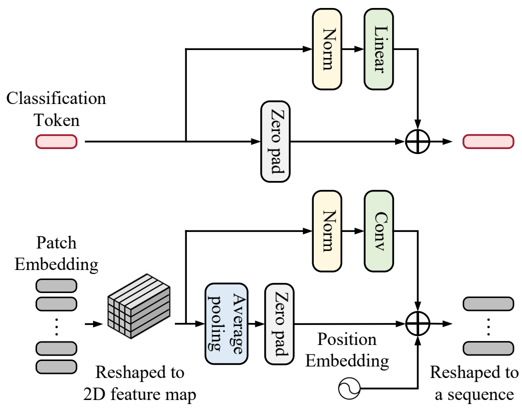

Figure 3: **Structure of residual spatial reduction.** We use strided convolution to reduce sequence length of patch embeddings. A skip connection as shown in the lower branch is added to stabilize training and improve performance.

dimension is pre-defined. During super-network training and evolutionary search, sub-networks of different configurations are sampled for training and evaluation. Details of search space are presented in Appendix D.

Multi-Architectural Sampling for Super-Network Training. To evaluate the performance of sub-networks, their weights have to be optimized to reflect their relative quality. Therefore, we train a super-network whose architecture is the same as the large network defining the search space and whose weights are shared across sub-networks. During super-network training, we sample and train different sub-networks for different training iterations. Generally, the more sub-networks we sample, the more accurate the relative quality of sub-networks indicated by the trained super-network can be. Previous works on NAS for CNNs [4, 13, 56] sample and train a *single* sub-network with *one* forward-backward pass for each training iteration. Given a fixed amount of iterations, this results in room for improvement when training ViT-Res super-networks as we can sample *multiple* sub-networks with *one* forward-backward pass for each training iteration.

Unlike batch normalization [21] in CNNs, layer normalization (LN) [1] in ViT avoids normalizing along batch dimension, which enables sampling *different* sub-networks with *one* forward-backward pass without mixing their statistics. Specifically, for each training iteration, we sample N_a sub-networks and divide a batch into N_a sub-batches. Each sub-network is trained with its corresponding sub-batch. This can be achieved efficiently with *one* forward-backward pass and *channel masking* (*ordered dropout*) [43, 54, 56]. As shown in Fig. 4 (a), different masks are generated for different feature maps to *zero out* different channels and simulate sampling different sub-networks. The shapes of feature maps remain the same, which maintains regular batch computation and therefore enables a single forward-backward pass.

We *re-calibrate* the statistics in LN when sampling multiple sub-networks to prevent discrepancy between super-network training and standard training. As shown in Fig. 4 (b) and (c), LN can incorrectly normalize over a larger channel dimension when sampling networks with smaller channel sizes. This is because channel masking only changes the values of feature maps not shapes. To avoid the problem, we propose *masked layer normalization* (MLN). Instead of only looking at the shape of input tensors, MLN calculates the ratio of the number of masked channels (zeroed out) to the total number of channels and uses that ratio to re-scale the channel dimension and re-calibrate statistics as in Fig. 4 (d). With MLN, the statistics become the same as we train sub-networks separately as in Fig. 4 (b). Eventually, with channel masking and MLN, we can sample *multiple* sub-networks with *one* forward-backward pass, which improves sample efficiency when training ViT-Res super-networks and therefore the performance of searched networks. Other details of super-network training can be found in Appendix E.

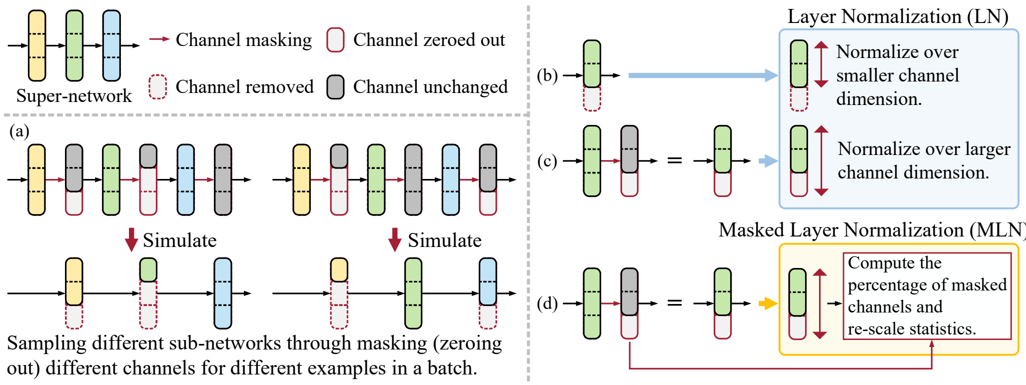

Figure 4: **Channel masking and masked layer normalization for multi-architectural sampling.** **(a)** We simulate sampling different sub-networks by masking different channels. **(b)** The channel dimension LN normalizes over when training a sub-network alone. **(c)** The channel dimension LN incorrectly normalizes over when sampling a sub-network during super-network training. **(d)** MLN re-calibrates statistics by considering the number of masked channels.

Evolutionary Search. We sample sub-networks satisfying pre-defined resource constraints (e.g. MACs) and evaluate their performance (e.g. accuracy) with trained super-network weights. Evolutionary search maintains a population of networks and refines top-performing ones for many iterations. We start with an initial population of P_0 randomly sampled sub-networks. At every iteration, N_{parent} sub-networks with the highest performance in the population serve as parent networks that generate N_{child} new sub-networks through *mutation* and *crossover*. For mutation, we randomly select one sub-network from parent networks and modify every architectural parameter of this sub-network with a pre-defined probability p_{mutate}. For crossover, we choose two random sub-networks from parent networks and randomly fuse their architectural parameters. Mutation and crossover generates the same amount of new sub-networks, and they are added to the population. The process is repeated for T_{search} times, and the best sub-network in the population becomes the searched network.

2.3. Extra Techniques

Token Labeling with CutMix and Mixup. We incorporate *token labeling* [11, 22] and propose to improve it with Mixup [59]. Token labeling provides labels for all patches in an input image. This enables training ViT to predict class labels of all *patch tokens* (patch embeddings) in addition to predicting class label of an input image with classification token and can improve training Transformers [8, 11, 22]. Token labeling generates an input image, its class label, and patch-wise class labels through patch-wise CutMix [58]. Please refer to these works [11, 22] for details.

We find that applying Mixup [59] along with token labeling can improve performance, which is contrary to previous results [11, 22]. We surmise that they first apply Mixup and then perform patch-wise CutMix could lead to noisy training data. In contrast, we improve token labeling with Mixup through *switching* between patch-wise CutMix and Mixup. Specifically, we choose either patch-wise CutMix or Mixup to generate an image, its label and patch-wise labels. When the latter is chosen, the image and its label are generated in the same way as Mixup, and this label is assigned to all patches to produce the patch-wise labels.

Convolution before Tokenization. Following previous works [10, 12, 22], we add some convolutional layers before tokenization. Specifically, we add three 3×3 convolutional layers, each with C output channels. A residual skip connection is added between outputs of the first and the third layers. Further details can be found in Appendix C and D.

3. Experiments

With the proposed methods, we design our ViT-ResNAS networks of three different complexity. As shown in Table 1, ViT-ResNAS networks achieve better accuracy-MACs trade-offs than previous works. Please refer to Appendix A for complete empirical results.

Method	MACs (G)	Top-1 Accuracy (%)
DeiT-Ti [40]	1.3	72.2
T2T-ViT-12 [57]	2.2	76.5
PiT-XS [16]	1.4	78.1
PVT-Tiny [47]	1.9	75.1
ViT-ResNAS-Tiny	1.8	80.8
DeiT-Small [40]	4.6	79.9
T2T-ViT-14 [57]	5.2	81.5
PiT-S [16]	2.9	80.9
PVT-Small [47]	3.8	79.8
Swin-T [25]	4.5	81.3
ViT-ResNAS-Small	2.8	81.7
DeiT-Base [40]	17.6	81.8
T2T-ViT-19 [57]	8.9	81.9
PiT-B [16]	12.5	82.0
PVT-Large [47]	9.8	81.7
ViT-ResNAS-Medium	4.5	82.4

Table 1: **Comparison with related works.**

References

[1] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2, 3, 10

[2] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le. Attention augmented convolutional networks. In *International Conference on Computer Vision (ICCV)*, 2019. 1

[3] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V. Le. Can weight sharing outperform random architecture search? an investigation with tunas. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 10, 11

[4] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 3, 10, 11

[5] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 10

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, 2020. 1

[7] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *arXiv preprint arXiv:2104.13840*, 2021. 9, 10

[8] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations (ICLR)*, 2020. 4

[9] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. *arXiv preprint arXiv:2103.10697*, 2021. 10

[10] 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 (ICLR)*, 2021. 1, 4, 10

[11] Chengyue Gong, Dilin Wang, Meng Li, Vikas Chandra, and Qiang Liu. Improve vision transformers training by suppressing over-smoothing. *arXiv preprint arXiv:2104.12753*, 2021. 4, 8, 10

[12] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. *arXiv preprint arXiv:22104.01136*, 2021. 4

[13] 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 (ECCV)*, 2020. 1, 3, 10

[14] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 9, 10

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[16] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. *arXiv preprint arXiv:2103.16302*, 2021. 4, 9, 10

[17] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 8

[18] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 10

[19] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(8):2011–2023, 2020. 1

[20] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision (ECCV)*, 2016. 8

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 3

[22] Zihang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Xiaojie Jin, Anran Wang, and Jiashi Feng. Token labeling: Training a 85.5% top-1 accuracy vision transformer with 56m parameters on imagenet. *arXiv preprint arXiv:2104.10858*, 2021. 4, 8, 10

[23] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. *arXiv preprint arXiv:2103.12424*, 2021. 10

[24] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019. 10

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 4, 9, 10

[26] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable visual transformers with hierarchical pooling. *arXiv preprint arXiv:2103.10619*, 2021. 10

[27] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning (ICML)*, 2018. 10

[28] Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár. Designing network design

spaces. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[29] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019. 1

[30] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019. 2, 10

[31] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning (ICML)*, 2017. 2, 10

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1, 2, 8, 10

[33] David So, Quoc Le, and Chen Liang. The evolved transformer. In *International Conference on Machine Learning (ICML)*, 2019. 10

[34] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[35] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path mobile automl: Efficient convnet design and nas hyperparameter optimization. In *arXiv preprint arXiv:1907.00959*, 2019. 1, 10

[36] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. In *arXiv preprint arXiv:1904.02877*, 2019. 1, 10

[37] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *International Conference on Computer Vision (ICCV)*, 2017. 1, 10

[38] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 10

[39] Haotian* Tang, Zhijian* Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European Conference on Computer Vision (ECCV)*, 2020. 10, 11

[40] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 1, 2, 4, 8, 9, 10

[41] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 9, 10

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Conference on Neural Information Processing (NeurIPS)*, 2017. 1, 2

[43] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, Peter Vajda, and Joseph E. Gonzalez. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[44] Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. Hat: Hardware-aware transformers for efficient natural language processing. In *Annual Conference of the Association for Computational Linguistics (ACL)*, 2020. 1, 10

[45] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. *arXiv preprint arXiv:2012.00759*, 2020. 1

[46] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision (ECCV)*, 2020. 1

[47] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *International Conference on Computer Vision (ICCV)*, 2021. 4, 9, 10

[48] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[49] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 8

[50] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 10

[51] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021. 9, 10

[52] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning (ICML)*, 2020. 2

[53] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *European Conference on Computer Vision (ECCV)*, 2018. 1, 10

[54] Tien-Ju Yang, Yi-Lun Liao, and Vivienne Sze. Netadaptv2: Efficient neural architecture search with fast super-network

training and architecture optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3, 10, 11

[55] Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers. *ArXiv*, abs/1903.11728, 2019. 1, 10

[56] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 3, 10

[57] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021. 4, 9, 10

[58] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision (ICCV)*, 2019. 4

[59] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018. 4

[60] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021. 9, 10

[61] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[62] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations (ICLR)*, 2021. 1

[63] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 10

[64] Barret Zoph, V. Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 10

Methods	Top-1 Acc.
Spatial reduction	78.5
Residual spatial reduction	78.8 (+0.3)
+ Token labeling with CutMix	79.6 (+0.8)
+ Token labeling with CutMix & Mixup	80.1 (+0.5)

Table 2: **Additive study of improving multi-stage network with residual connections and token labeling.** We start with DeiT-Tiny with spatial reduction and progressively introduce residual connections and token labeling. Adding residual connections and combining token labeling with Mixup in the proposed manner improve accuracy without increasing MACs.

Appendix

A. Experiments

In this section, we first describe the experiment setup. Then, we conduct experiments to study the effectiveness of our proposed methods. Finally, the comparison with related works is presented.

A.1. Experiment Setup

The dataset used is ImageNet [32]. Our implementation is based on timm library [49] and that of DeiT [40]. Most of the training settings follow those in DeiT [40] except that we do not use repeated augmentation [17]. We train models with 8 GPUs for 300 epochs, and batch size is 1024 and input resolution is 224×224. As for token labeling, we adopt the same loss function as previous works [11, 22], and the associated loss is directly added to the original classification loss without any scaling. Unless otherwise stated, we incorporate token labeling with patch-wise CutMix and Mixup to train our networks and include several convolutional layers before tokenization. For results presented in Section A.2, when token labeling is used, drop path [20] rate is increased from 0.1 to 0.2.

For neural architecture search, we increase the depth and width of our ViT-Res network to build search spaces and search the architectures of ViT-ResNAS networks of three different sizes, which we name Tiny, Small, and Medium. For ViT-ResNAS-Tiny, we enlarge ViT-Res-Tiny to build a super-network and train both the super-network and the searched network with drop path rate 0.2. For ViT-ResNAS-Small and Medium, we further enlarge ViT-Res-Tiny super-network and share the same search space. The super-network is trained with drop path rate 0.3. The drop path rates for Small and Medium are 0.3 and 0.4, respectively. The details of the search space are presented in Table 6. A super-network is trained for 120 epochs, with other settings being the same as mentioned above. We sample N_a sub-networks with one forward-backward pass during super-network training, experiment with different values of N_a and empirically set N_a to 16. For evolutionary search, the resource constraint is MAC. We set search iteration $T_{search} = 20$, the number of parent networks $N_{parent} = 75$, the initial population size $P_0 = 500$, the number of new sub-networks for each iteration $N_{child} = 150$ and mutation probability $p_{mutate} = 0.3$. We train our searched ViT-ResNAS networks with EMA.

In addition, following DeiT [40], we fine-tune networks at larger resolutions to obtain higher capacity. A network is fine-tuned for 30 epochs, with batch size 512, learning rate 5×10^{-6}, weight decay 10^{-8} and drop path rate 0.75.

A.2. Analysis on Proposed Methods

Multi-Stage Network with Residual Connection and Token Labeling. We study how the performance of vanilla multi-stage networks can be enhanced with the proposed residual connections and improved token labeling training. We build such a network by starting with DeiT-Tiny network, adding three convolutional layers before tokenization and inserting two spatial reduction blocks (i.e. only the *residual branch* of residual spatial reduction). Please note that transforming DeiT-Tiny into this vanilla multi-stage architecture significantly improves the top-1 accuracy from 72.2% to 78.5%. Further results are shown in Table 2. Without token labeling, introducing only two residual connections (i.e. *main branch* of residual spatial reduction) can improve accuracy from 78.5% to 78.8% with negligible overhead. When token labeling is used, incorporating Mixup in the proposed manner can further improve the accuracy by 0.5%. With residual spatial reduction and token labeling, we can improve the accuracy by 1.6% without increasing MACs, and our ViT-Res-Tiny achieves 80.1% top-1 accuracy with 1.8G MACs.

We also study the effectiveness of residual connections (i.e. main branch) in the proposed residual spatial reduction under another training setting. When using the training setting of DeiT-distill [40], which includes repeated augmentation and distilling knowledge of CNN, residual connections can improve the accuracy from 79.6% to 80.1%. Moreover, when we train deeper networks such as our ViT-Res-Tiny super-network under the training setting of DeiT-distill, without residual connections, the training can be unstable with training loss "nan".

Weight-Sharing NAS with Multi-Architectural Sampling. With the proposed residual spatial reduction and token labeling, we study how weight-sharing NAS with multi-architectural sampling can further improve the performance of ViT-Res. During super-network training, we sample and train N_a sub-networks with one forward-backward pass. Given the same amount of training iterations and training

Number of sampled sub-networks per forward-backward pass (N_a)	Single-arch.	Multi-arch.		
	1	8	16	32
Top-1 accuracy (%)	80.5	80.6 (+0.1)	80.8 (+0.3)	80.6 (+0.1)

Table 3: **Effect of numbers of sampled sub-networks with one forward-backward pass (N_a) on the performance of searched networks.** The type of resource constraint is MAC, and all searched networks have MAC around 1.8G. Empirically, $N_a = 16$ results in the best searched network.

Method	Model Size (M)	MACs (G)	Top-1 Accuracy (%)	Throughput (images/second)
DeiT-Ti [40]	5	1.3	72.2	1968
T2T-ViT-12 [57]	7	2.2	76.5	1192
PiT-XS [16]	11	1.4	78.1	1647
PVT-Tiny [47]	13	1.9	75.1	1133
ViL-Tiny [60]	7	1.3	76.3	754
ViT-Res-Tiny	43	1.8	80.1	1807
ViT-ResNAS-Tiny	41	1.8	80.8	1579
DeiT-Small [40]	22	4.6	79.9	846
T2T-ViT-14 [57]	22	5.2	81.5	682
PiT-S [16]	24	2.9	80.9	986
PVT-Small [47]	25	3.8	79.8	628
TNT-S [14]	24	5.2	81.5	353
Swin-T [25]	29	4.5	81.3	600
Twins-PCPVT-S [7]	24	3.7	81.2	622
ViT-ResNAS-Small	65	2.8	81.7	1084
DeiT-Base [40]	86	17.6	81.8	290
T2T-ViT-19 [57]	39	8.9	81.9	428
PiT-B [16]	74	12.5	82.0	316
PVT-Large [47]	61	9.8	81.7	284
ViL-Small [60]	25	4.9	82.0	310
CvT-13 [51]	20	4.5	81.6	587
ViT-ResNAS-Medium	97	4.5	82.4	751
Swin-S [25]	50	8.7	83.0	351
Twins-PCPVT-B [7]	44	6.4	82.7	403
CvT-21 [51]	32	7.1	82.5	379
ViT-ResNAS-Medium $\uparrow 280$	97	7.1	83.1	467
ViL-Medium-Wide [60]	40	11.3	82.9	177
Twins-PCPVT-L [7]	61	9.5	83.1	282
CaiT-S36 [41]	68	13.9	83.3	191
ViT-ResNAS-Medium $\uparrow 336$	97	10.6	83.5	292
Swin-B [25]	88	15.4	83.3	243
CaiT-XS24 $\uparrow 384$ [41]	27	19.3	83.8	57
ViT-ResNAS-Medium $\uparrow 392$	97	15.2	83.8	194

Table 4: **Comparison with related works on ViT.** "$\uparrow R$" denotes that the model is first trained at resolution 224 and then fine-tuned at resolution R. Other models are trained at resolution 224. Throughput is measured on one Titan RTX GPU with batch size 128.

examples, the value of N_a controls the trade-offs between sample efficiency (i.e. how many sub-networks are sampled) and the quality of training each sub-network (i.e. how many examples are used to train it). Therefore, instead of arbitrarily choosing a large value, we experiment with different N_a. We use our ViT-Res-Tiny super-network to study the effect of different N_a on the performance of searched networks and search for networks with MACs around 1.8G. The results are presented in Table 3. Compared to sampling one sub-network for each training iteration ("Single-arch." in Table 3), sampling multiple sub-networks leads to better searched networks. Among different values, $N_a = 16$ empirically results in the best searched network. Please note that with NAS, the top-1 accuracy is increased from 80.1% to 80.5% and that with the proposed multi-architectural sampling, the accuracy is further increased from 80.5% to 80.8%. Based on the results, we choose $N_a = 16$ to design larger ViT-ResNAS networks.

A.3. Comparison with Related Works

We design our ViT-ResNAS-Tiny, Small and Medium networks with NAS. Following DeiT [40], we fine-tune our ViT-ResNAS-Medium at larger resolutions to obtain models with higher capacity. Since the patch size before the first stage is 14 and there are 2 residual spatial reduction in the network, the spatial resolution is reduced by 56 ($= 14 \times 2 \times 2$) times in the last stage, and therefore we can only increase the input resolution by multiples of 56. We report the performance of fine-tuning ViT-ResNAS-Medium at resolutions 280, 336 and 392. Following the implementation of Swin Transformer [25], we measure the inference throughput of models on a single Titan RTX GPU with batch size 128 as well.

Table 4 summarizes the comparison with previous works and Fig. 5 visualizes accuracy-MACs trade-offs. Please note that why our models have more parameters is that we further reduce the sequence length and increase the channel size of DeiT models. Even though we have more parameters, our ViT-ResNAS networks consistently achieves better accuracy-MACs trade-offs as well as accuracy-throughput trade-offs. Compared to the original single-stage DeiT [40], ViT-ResNAS-Tiny achieves 8.6% higher accuracy than DeiT-Ti while having only 0.5G higher MACs and achieves 0.9% higher accuracy than DeiT-Small with 60% less MACs. Compared to other works on multi-stage architectures like PVT [47] and PiT [16], ViT-ResNAS consistently has better accuracy-MACs trade-offs as well. The computation saving becomes more apparent as the accuracy becomes higher. This suggests the effectiveness of NAS to scale up models.

Additionally, for lower MACs around 2.0G, ViT-ResNAS achieves better accuracy-MACs trade-offs than ViL-Tiny. However, when MACs are around 5.0G, ViT-ResNAS is on par with ViL-Small in terms of accuracy-MACs trade-offs. This probably suggests that utilizing efficient attention mechanisms [7, 25, 60] with *convolution-like locality* to process high-resolution features is necessary for models in higher MAC regimes to generalize better. Nevertheless, those methods are orthogonal to our approaches, and the proposed weight-sharing NAS with multi-architectural sampling could be used to further improve their accuracy-MACs trade-offs as well.

B. Related Works

Vision Transformers. Vision Transformer (ViT) [10] demonstrates that a pure transformer without convolution can perform well on image classification when trained on large datasets like JFT-300M [37]. To make it data-efficient, DeiT [40] uses strong regularization and adds a distillation token to its architecture for knowledge distillation, and demonstrates comparable performance when trained on ImageNet [32] only. Subsequent works improve the performance of ViT on ImageNet through either better training [11, 22] or architectures [9, 14, 16, 25, 26, 41, 47, 51, 57, 60]. For example, they bring locality into network architectures by using convolutions [47, 51] or efficient local attention [7, 14, 25, 60] or similarly adopt multi-stage architectures [16, 26, 47, 51, 60]. Our proposed methods are complementary to these works. Residual spatial reduction can derive a more efficient multi-stage architectures, and weight-sharing NAS with multi-architectural sampling can be applied to further improve performance since they use layer normalization [1] as well.

Neural Architecture Search. There have been increasingly interest in designing efficient architectures with neural architecture search (NAS) [3, 4, 5, 13, 18, 23, 30, 31, 33, 35, 36, 38, 39, 44, 50, 53, 54, 55, 56, 63, 64]. Among different methods, weight-sharing NAS [3, 4, 5, 13, 23, 24, 27, 35, 36, 44, 50, 55, 56] has become popular due to efficiency. They train one over-parametrized super-network whose weights are shared across all networks in a search space to conduct architecture search, which saves computation cost significantly. However, many of these works focus on CNN, which has been researched for years, and have well-designed search space. In contrast, our proposed NAS with multi-architectural sampling focuses on multi-stage ViT, which is much less studied, and we utilize its batch-independent property to further improve the performance of searched networks.

C. Network Architecture of ViT-Res

We build our ViT-Res-Tiny network by introducing two modifications to DeiT-Tiny [40]. First, residual spatial reduction (RSR) is applied to evenly divide the single-stage architecture into a multi-stage one as described in Sec. 2.1. For "Conv" in Fig. 3, we use 3×3 convolutions to downsample patch embeddings. Second, we add three convolutional layers as mentioned in Sec. 2.3 and use $C = 24$. Table 5 summarizes the archtiecture of ViT-Res-Tiny.

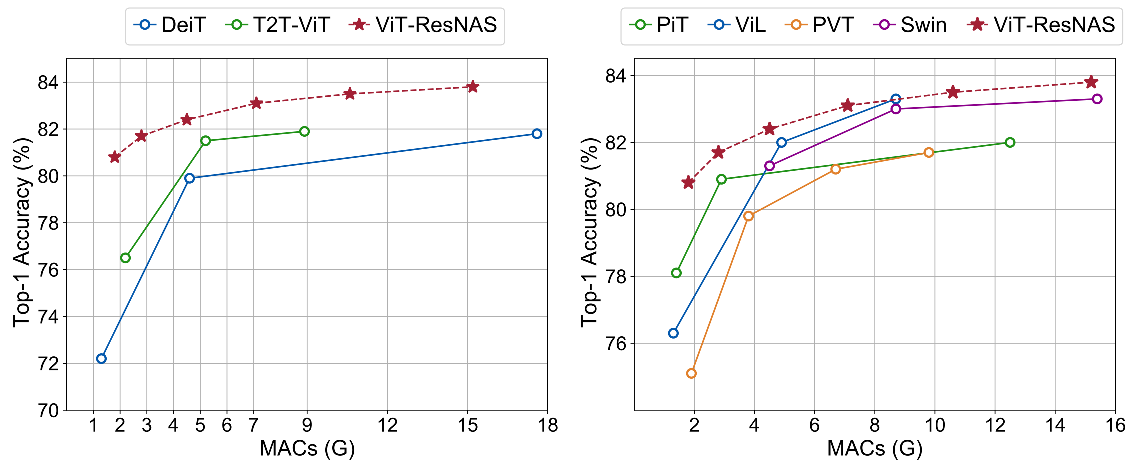

Figure 5: **Accuracy-MACs trade-offs of previous works and ViT-ResNAS.** We separate into two figures for better visualization.

	Output Sequence Length	**ViT-Res-Tiny**
Tokenization	$16 \times 16 + 1$	conv-3_24_2 $\begin{bmatrix} \text{conv-3\_24\_1} \\ \text{conv-3\_24\_1} \end{bmatrix}$ conv-7_192_7
Stage 1	$16 \times 16 + 1$	$\begin{bmatrix} \text{MHSA-64\_3} \\ \text{FFN-768} \end{bmatrix} \times 4$
Stage 2	$8 \times 8 + 1$	RSR-384
	$8 \times 8 + 1$	$\begin{bmatrix} \text{MHSA-64\_6} \\ \text{FFN-1536} \end{bmatrix} \times 4$
Stage 3	$4 \times 4 + 1$	RSR-768
	$4 \times 4 + 1$	$\begin{bmatrix} \text{MHSA-64\_12} \\ \text{FFN-3072} \end{bmatrix} \times 4$

Table 5: **Architecture of ViT-Res-Tiny.** "conv-k_C_s" stands for a $k \times k$ convolutional layer with output channel C and stride s. "MHSA-d_h" is MHSA with head dimension d and h attention heads. "FFN-d_{hidden}" is FFN with hidden size d_{hidden}. "RSR-d_{embed}" is residual spatial reduction with output embedding dimension d_{embed}. Note that the embedding dimension in Stage 1 is determined by the last convolutional layer in tokenization.

D. Details of Search Space

We enlarge ViT-Res-Tiny to build ViT-Res-Tiny super-network. We increase numbers of attention heads h and decrease head dimensions d for the first two stage so that we have more choices of attention heads h. We do not search configurations of convolutional layers in tokenization except that we search the number of output channels in the last layer, which determines the embedding dimension of the first stage. For the search space for ViT-ResNAS-Tiny, each stage has three pairs of transformer blocks. The first block in each pair always remains while the second one is skippable. Therefore, each stage can have three to six transformer blocks. Additionally, we enlarge the ViT-Res-Tiny super-network to construct ViT-Res-Small and Medium super-network and search space by increasing width and adding one block for each stage. Table 6 summarizes the search spaces for ViT-ResNAS networks.

E. Additional Super-Network Training Details

We divide the original training set into *sub-train* and *sub-validation* sets. The sub-validation set contains 25K images, with 25 images for each class. The rest of images form the sub-train set. We train super-networks on the sub-train set, and during evolutionary search, we evaluate the accuracy of sub-networks on the sub-validation set. Besides, during super-network training, we *warm up* training different widths and depths (filter warmup [3, 54] or progressive shrinking [4, 39]). Specifically, at the beginning, we only sample and train the largest sub-network in a search space. As the super-network training proceeds, we *gradually* sample and train sub-networks with smaller widths and depths. After 25% of total training epochs, sub-networks with any width and depth can be sampled and trained.

	Output Length	**ViT-ResNAS-Tiny Search Space**
Token.	$d_1 \in \{256, 224, 192, 176, 160\}$	
	$16 \times 16 + 1$	conv-3_24_2 $\begin{bmatrix}\text{conv-3\_24\_1} \\ \text{conv-3\_24\_1}\end{bmatrix}$ conv-7_d_1_1_7
Stage 1	$h_1 \in \{6, 5, 4, 3\}$, $f_1 \in \{768, 704, 640, 576, 512, 448, 384\}$	
	$16 \times 16 + 1$	$\begin{bmatrix}\begin{pmatrix}\text{MHSA-32\_}h_1 \\ \text{FFN-}f_1\end{pmatrix} \\ \begin{pmatrix}\text{(skippable)} \\ \text{MHSA-32\_}h_1 \\ \text{FFN-}f_1\end{pmatrix}\end{bmatrix} \times 3$
Stage 2	$d_2 \in \{512, 448, 384, 352, 320\}$, $h_2 \in \{12, 10, 8, 6\}$, $f_2 \in \{1536, 1408, 1280, 1152, 1024, 896, 768\}$	
	$8 \times 8 + 1$	RSR-d_2
	$8 \times 8 + 1$	$\begin{bmatrix}\begin{pmatrix}\text{MHSA-48\_}h_2 \\ \text{FFN-}f_2\end{pmatrix} \\ \begin{pmatrix}\text{(skippable)} \\ \text{MHSA-48\_}h_2 \\ \text{FFN-}f_2\end{pmatrix}\end{bmatrix} \times 3$
Stage 3	$d_3 \in \{1024, 896, 768, 704, 640\}$, $h_3 \in \{12, 10, 8, 6\}$, $f_3 \in \{3072, 2816, 2560, 2304, 2048, 1792, 1536\}$	
	$4 \times 4 + 1$	RSR-d_3
	$4 \times 4 + 1$	$\begin{bmatrix}\begin{pmatrix}\text{MHSA-64\_}h_3 \\ \text{FFN-}f_3\end{pmatrix} \\ \begin{pmatrix}\text{(skippable)} \\ \text{MHSA-64\_}h_3 \\ \text{FFN-}f_3\end{pmatrix}\end{bmatrix} \times 3$

	Output Length	**ViT-ResNAS-Small and Medium Search Space**
Token.	$d_1 \in \{320, 280, 240, 220, 200\}$	
	$16 \times 16 + 1$	conv-3_24_2 $\begin{bmatrix}\text{conv-3\_24\_1} \\ \text{conv-3\_24\_1}\end{bmatrix}$ conv-7_d_1_1_7
Stage 1	$h_1 \in \{8, 7, 6, 5\}$, $f_1 \in \{960, 880, 800, 720, 640, 560, 480\}$	
	$16 \times 16 + 1$	$\begin{bmatrix}\begin{pmatrix}\text{MHSA-32\_}h_1 \\ \text{FFN-}f_1\end{pmatrix} \\ \begin{pmatrix}\text{(skippable)} \\ \text{MHSA-32\_}h_1 \\ \text{FFN-}f_1\end{pmatrix}\end{bmatrix} \times 3$ $\begin{pmatrix}\text{MHSA-32\_}h_1 \\ \text{FFN-}f_1\end{pmatrix}$
Stage 2	$d_2 \in \{640, 560, 480, 440, 400\}$, $h_2 \in \{16, 14, 12, 10\}$, $f_2 \in \{1920, 1760, 1600, 1440, 1280, 1120, 960\}$	
	$8 \times 8 + 1$	RSR-d_2
	$8 \times 8 + 1$	$\begin{bmatrix}\begin{pmatrix}\text{MHSA-48\_}h_2 \\ \text{FFN-}f_2\end{pmatrix} \\ \begin{pmatrix}\text{(skippable)} \\ \text{MHSA-48\_}h_2 \\ \text{FFN-}f_2\end{pmatrix}\end{bmatrix} \times 3$ $\begin{pmatrix}\text{MHSA-48\_}h_2 \\ \text{FFN-}f_2\end{pmatrix}$
Stage 3	$d_3 \in \{1280, 1120, 960, 880, 800\}$, $h_3 \in \{16, 14, 12, 10\}$, $f_3 \in \{3840, 3520, 3200, 2880, 2560, 2240, 1920\}$	
	$4 \times 4 + 1$	RSR-d_3
	$4 \times 4 + 1$	$\begin{bmatrix}\begin{pmatrix}\text{MHSA-64\_}h_3 \\ \text{FFN-}f_3\end{pmatrix} \\ \begin{pmatrix}\text{(skippable)} \\ \text{MHSA-64\_}h_3 \\ \text{FFN-}f_3\end{pmatrix}\end{bmatrix} \times 3$ $\begin{pmatrix}\text{MHSA-64\_}h_3 \\ \text{FFN-}f_3\end{pmatrix}$

Table 6: **ViT-ResNAS search space. Left:** search space for ViT-ResNAS-Tiny. **Right:** search space for ViT-ResNAS-Small and Medium. For each stage i, we search embedding dimension d_{embed} and numbers of transformer blocks. For each transformer block, we search the number of attention heads h in MHSA and hidden size d_{hidden} in FFN. Different blocks can have different values of h and d_{hidden}. The first rows in tokenization and each stage define the range of each searchable dimension, with d_i, h_i, and f_i corresponding to d_{embed}, h, and d_{hidden} in stage i, respectively. Transformer blocks with "skippable" can be removed during super-network training and evolutionary search, which supports different numbers of blocks in searched networks.