Supplementary Material - MDL-NAS: A Joint Multi-domain Learning Framework for Vision Transformer

A. Gradient estimation for S_i

In mask sharing policy, we introduce scoring parameters $S_i = [S_i^j]$ for channels in the i-th linear layer, where $j = 1, 2, ..., C_{out}^{max}$, and define the indicator function as follows:

$$\mathbb{I}(S_i^j) = \begin{cases} 1, & if\ S_i^j \geq TH_r \\ 0, & otherwise. \end{cases} \tag{1}$$

Then, we can obtain the parameter sharing mask $M_i \in \mathbb{R}^{C_{out}^{max}}$ as follows

$$M_i = [\mathbb{I}(S_i^1), \mathbb{I}(S_i^2), ..., \mathbb{I}(S_i^{C_{out}^{max}})]. \tag{2}$$

where M_i is used to derive the task-share and task-specific weight:

$$w_i^{share} = W_i[: c_{out}, : c_{in}] \otimes M_i[: c_{out}], \tag{3}$$

$$w_{k,i}^{spe} = W_i^{spe}[: c_{out}, : c_{in}] \otimes (1 - M_i)[: c_{out}], \tag{4}$$

where $\otimes$ denotes element-wise multiplication; k denotes the task index, $k = 1, 2, ..., K$.

Finally, we can obtain the current weight as $w_{k,i}$ by

$$w_{k,i} = w_i^{share} + w_{k,i}^{spe}, \tag{5}$$

In the backward pass of the network, the gradient of S_i^j is formulated as:

$$\nabla_{S_i^j} w_{k,i} = \frac{\partial w_{k,i}}{\partial \mathbb{I}(S_i^j)} \frac{\partial \mathbb{I}(S_i^j)}{\partial S_i^j}. \tag{6}$$

However, such optimization problem cannot be directly optimized with stochastic gradient descent since the gradient of the indicator function $\mathbb{I}(\cdot)$ is zero at almost all points and undefined at TH_r. To make the problem learnable, we use a straight-through gradient estimator. That is, modifying the backward pass and using:

$$\nabla_{S_i^j} w_{k,i} = \frac{\partial w_{k,i}}{\partial \mathbb{I}(S_i^j)}. \tag{7}$$

B. Searching space

For non-hierarchical vision transformer (ViT), following AutoFormer [2], we set the embedding dimension, MLP ratio, heads num, and share ratio are set to {528, 572, 624}, {3, 3.5, 4}, {8, 9, 10}, {0.4, 0.5, 0.6} respectively for all stages. In all experiments, the scaling factor d_h and network depth are set to 64 and 16, respectively Note that the heads number and MLP ratios are varied across layers.

For hierarchical vision transformer (Swin Transformer), the number of blocks, embedding dimension, heads number, and MLP ratio are set to **{**2, 2, {7, 8}, {1, 2}**}**, **{**{96, 128}, {160, 192}, {416, 448}, {704, 746}**}**, **{**{2, 3}, {5, 6}, {11, 12, 13}, {19, 20, 21}**}** and **{**{2, 2.5, 3}, {2.5, 3, 3.5}, {3.5, 4, 4.5}, {3.5, 4, 4.5}**}**respectively for four stages. We set window size to 14 for the second and third stage. The scaling factor d_h is set to 32 in all experiments. For fine search space, we search shared ratio for FFN layer and MHSA layer in each stage, with shared ratio setting to **{**{0.5, 0.7, 0.9}, {0.5, 0.7, 0.9}, {0.3, 0.5, 0.7}, {0.1, 0.3, 0.5}**}** respectively for four stages. Note that FFN layer and MHSA layer have two identical yet independent search space.

C. Implement details in supernet pretraining

Implement details of learning features from multiple domains. Since each task has its own domain, we use multiple GPUs to optimize the tasks, where the task-shared and task-specific parameters are optimized in the global group and task-specific group respectively, which is implemented by defining multiple communication groups in DDP of PyTorch. In each training iteration, we sample a subnet for each task from the search space and update its corresponding weights in MDL-NAS while freezing the rest. For task-shared parameters, we average their gradients across all GPUs. Thus, the gradient vectors of shared parameters on all GPUs are equivalent. If we ensure that the initialization of parameters, the learning rate, and weight decay for each task are the same, we can finally get task-shared parameters for all tasks. For task-specific parameters, we average their gradients across GPUs within the task-specific group.

For MDL-NAS equipped with non-hierarchical vision

		L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15	L16
MDAL-NAS†+mask classification	Embed Dim	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528
	Heads Num	9	9	9	9	10	10	9	9	9	10	9	10	10	9	9	9
	MLP Ratio	3	3	3	3	3	3	3	3	3	3	3.5	4	3	3	3	3
	Share Ratio	0.4	0.4	0.4	0.5	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.6	0.4	0.4	0.4
MDAL-NAS†+mask detection	Embed Dim	576	576	576	576	576	576	576	576	576	576	576	576	576	576	576	576
	Heads Num	9	10	9	9	10	10	9	10	10	10	9	9	9	9	9	9
	MLP Ratio	3	3	3	3.5	4	3.5	3	3	4	3	3.5	3.5	4	3.5	3.5	4
	Share Ratio	0.5	0.6	0.5	0.5	0.4	0.5	0.5	0.4	0.4	0.5	0.4	0.6	0.4	0.4	0.6	0.4
MDAL-NAS†+mask segmentation	Embed Dim	624	624	624	624	624	624	624	624	624	624	624	624	624	624	624	624
	Heads Num	9	9	10	9	9	9	10	10	10	9	10	9	10	10	9	10
	MLP Ratio	4	4	3.5	4	3.5	3.5	3.5	3.5	4	3	4	3.5	3.5	4	4	3.5
	Share Ratio	0.5	0.5	0.4	0.4	0.4	0.6	0.5	0.4	0.6	0.5	0.5	0.4	0.4	0.6	0.6	0.6
MDAL-NAS†+seq classification	Embed Dim	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528
	Heads Num	9	10	9	10	10	10	10	10	9	10	9	9	9	9	9	9
	MLP Ratio	3.5	3	3	3	3	3.5	4	4	4	4	4	3.5	3	4	4	3.5
	Share Ratio	0.6	0.4	0.5	0.6	0.6	0.4	0.5	0.4	0.5	0.6	0.4	0.4	0.4	0.5	0.6	0.6
MDAL-NAS†+seq detection	Embed Dim	624	624	624	624	624	624	624	624	624	624	624	624	624	624	624	624
	Heads Num	9	9	10	9	10	9	10	9	10	9	10	10	9	9	9	10
	MLP Ratio	3	4	3.5	3.5	3.5	3	3.5	3	4	3	4	3.5	3	3.5	3.5	3.5
	Share Ratio	0.4	0.5	0.5	0.4	0.4	0.6	0.5	0.6	0.5	0.4	0.5	0.5	0.6	0.5	0.6	0.6
MDAL-NAS†+seq segmentation	Embed Dim	624	624	624	624	624	624	624	624	624	624	624	624	624	624	624	624
	Heads Num	9	9	9	10	10	9	10	9	10	10	9	10	10	10	10	10
	MLP Ratio	3.5	3	4	4	4	4	4	3.5	3.5	4	4	3.5	4	3	3	4
	Share Ratio	0.4	0.5	0.5	0.4	0.4	0.6	0.6	0.6	0.6	0.6	0.4	0.6	0.6	0.6	0.4	0.6

Table 1. Architectures of MDAL-NAS-B† with mask or sequential sharing policy for different vision tasks.

	downsp. rate		MDL-NAS + mask classification	MDL-NAS + mask detection	MDL-NAS + mask segmentation	MDL-NAS + seq classification	MDL-NAS + seq detection	MDL-NAS + seq segmentation
stage1	4x	Embed dim	128	128	128	128	128	96
		Heads Num	3	3	2	3	3	3
		MLP Ratio	2	3	3	2	2.5	3
		Depth	2	2	2	2	2	2
		MHSA Share Ratio	0.9	0.9	0.7	0.9	0.5	0.9
		FFN Share Ratio	0.9	0.9	0.9	0.9	0.5	0.9
stage2	8x	Embed dim	160	192	192	160	192	192
		Heads Num	5	5	6	5	6	6
		MLP Ratio	3.5	3.5	3.5	3	3	3.5
		Depth	2	2	2	2	2	2
		MHSA Share Ratio	0.7	0.7	0.9	0.5	0.5	0.9
		FFN Share Ratio	0.7	0.9	0.5	0.5	0.5	0.9
stage3	16x	Embed dim	448	448	448	448	448	448
		Heads Num	12	13	13	13	13	13
		MLP Ratio	4	4.5	4.5	4	4	4
		Depth	8	8	8	8	8	8
		MHSA Share Ratio	0.9	0.7	0.7	0.7	0.5	0.7
		FFN Share Ratio	0.9	0.7	0.7	0.7	0.5	0.7
stage3	16x	Embed dim	704	736	736	704	736	704
		Heads Num	21	21	20	20	19	19
		MLP Ratio	3.5	4.5	4.5	4.5	3.5	4.5
		Depth	1	2	2	1	2	1
		MHSA Share Ratio	0.5	0.5	0.5	0.3	0.5	0.1
		FFN Share Ratio	0.5	0.3	0.5	0.5	0.5	0.5

Table 2. Architectures of MDAL-NAS-T‡ with mask or sequential sharing policy for different vision tasks.

transformer, we pre-train the supernet with the following settings: AdamW optimizer with weight decay 0.05 and total 500 epochs, initial learning rate 2e-3 and minimal learning rate 1e-5 with cosine scheduler, 20 epochs warmup, batch size of 2048, 0.1 label smoothing, and stochastic depth with drop rate 0.1. We leverage 32 Nvidia Tesla V100 GPUs to train the supernet. Data augmentation techniques, including RandAugment, Cutmix, Mixup and random erasing, are adopted with the same hyperparameters as in DeiT [9] except the repeated augmentation.

For MDL-NAS equipped with hierarchical vision transformer, we train the supernet with the following settings: AdamW optimizer for 300 epochs using a cosine decay learning rate scheduler and 20 epochs of linear warm-up;

A batch size of 1024, an initial learning rate of 0.001, and a weight decay of 0.05 are used. We adopt sandwich training [12] to optimize the weights of supernet, i.e., sampling the largest, the smallest, and two middle models for each iteration and fusing their gradients to update the supernet. Data augmentation techniques, including RandAugment, Cutmix, Mixup and random erasing are also adopted with the same hyperparameters as in Swin [8].

D. Implement details in supernet finetuning

For MDL-NAS equipped with non-hierarchical/hierarchical vision transformer, we employ 48/24 Nvidia Tesla V100 GPUs to finetune the supernet for all tasks, that is, classification, segmentation and detection takes up 16/8 GPUs respectively. For main results in the paper, we define the total number of training iterations as scheduler 3x based on object detection. In all experiments corresponding to supernet finetuning, we also leverage the sandwich training [12] rule. Note that for all tasks, the training recipe is exactly the same so as to derive task-shared parameters. The training recipe for non-hierarchical/hierarchical vision transformer is set as: AdamW optimizer with weight decay 0.05/0.05, initial learning rate 8e-5/1e-4 with cosine scheduler, and stochastic depth with drop rate 0.1/0.1. We set the batch size 512/1024 for image classification, 16/16 for object detection and 16/16 for semantic segmentation. Since the length of each dataset is different, we define the total number of training iterations based on object detection, schedule $1\times$ and $3\times$. Notably, the naive ViT employs constant widths throughout all of its blocks, which is not conducive to object detection. Thus, for MDL-NAS equipped with non-hierarchical vision transformer, we adhere to ViTDeT [7], employing the simple feature pyramid from a single-scale feature map (without the typical FPN design) and using window attention (without shifting) supplemented by a small number of cross-window propagation blocks.

E. Implement details in joint-subnet search algorithm

Our proposed search algorithm inherits most of the principles of the naive evolution search algorithm in SPOS [6]. We set the population size to 50 and number of generations to 20. Each generation we pick the top 10 architectures as the parents to generate child networks by mutation and crossover. The mutation probalilty P_m is set to 0.1. During search process, we employ Top1 Acc. of classification, AP^b of detection, and mIoU of segmentation as proxy score. Thus, the total score **S-Score** is given by a linear sum of above three proxy scores.

F. Downstream vision tasks experiment settings

In this part, we provide some detailed experimental settings of MDL-NAS in downstream vision tasks, including object detection, semantic segmentation and pose estimation.

Object detection. We conduct the object detection experiments on COCO dataset. We replace the backbone of Mask R-CNN with our discovered MDL-NAS transformer architecture and compare its performance with other prevalent backbones, including CNNs and handcrafted transformers. All the MDL-NAS models are conducted on MMDetection [1]. In all experiments, we use multi-scale training strategy during supernet finetuning (resizing the input such that the shorter side is between 400 and 1400 while the longer side is at most 1600).

Semantic segmentation. We choose ADE20K dataset to test the representation power of MDL-NAS on semantic segmentation task. ADE20K is a widely used scene parsing dataset which contains more than 20K scene-centric images and covers 150 semantic categories. The dataset is split into 20K images for training and 2K images for validation. All the models are trained under MMSegmentation [3] framework. For augmentations, we adopt the default setting in MMSegmentation of random horizontal flipping, random re-scaling within ratio range [0.5, 2.0] and random photometric distortion. Stochastic depth with ratio of 0.1 is applied for all MDL-NAS models. All MDL-NAS models are trained on the standard setting as the previous approaches with an input of 512×512.

CUB-200-2011 classification. The CUB-200-2011 dataset [10] has a total of 200 bird categories, including 5,994 training images and 5,794 testing data. Each category contains about 30 training data. The input image is a 384×384 color image. We add the CUB-200-2011 dataset to MDL-NAS after it has been finetuned on classification, detection, and segmentation tasks. We freeze the task-shared parameters and only leverage the task-specific parameters to fit this new dataset. All the models are trained under MMClassification [5] framework. In training stage, data augmentation is performed through Random Crop and Random HorizontalFlip while in testing phrase Center Crop is used. The training recipe is set as: SGD optimizer, the base learning rate 0.01 with cosine scheduler, weight decay 0.0005, the batch size of 8 and the total epochs 100.

We use the trained model on ImageNet by MDL-NAS-B with mask sharing policy to finetune on CUB dataset without freezing the task-shared parameters to get the new model, whose top1 accuracy on ImageNet is 0.1%. Instead, MDL-NAS can **evade catastrophic forgetting**.

Pose estimation. We choose image-based human body 2D pose estimation task to validate that our proposed MDL-

	Classification	Detection	Segmentation	S-Score
AutoFormer	15.0ms	110.4ms	54.8ms	176.4
MDL-NAS-B†+mask	14.1ms	113.3ms	55.7ms	180.9
MDL-NAS-B†+seq	15.8ms	112.9ms	57.6ms	181.7
Swin-T	14.0ms	37.6ms	55.5ms	171.8
MDL-NAS-T‡+mask	13.1ms	39.8ms	58.6ms	173.9
MDL-NAS-T‡+seq	14.0ms	38.9ms	60.8ms	174.6

Table 3. Latency of MDL-NAS compared with the baselines.

	Top-1 Acc.	AP^b	$mIoU_{ss}$	#Params (s)	S-Score
MDL-NAS-B-AB4†	83.0	46.3	49.6	117M	178.9
MDL-NAS-B† + seq	82.9	46.6	49.9	117M	179.4
MDL-NAS-T-AB4‡	81.6	43.0	45.2	57M	169.8
MDL-NAS-T‡ + seq	81.9	43.3	46.7	53M	171.9

Table 4. The efficacy of inside layer parameter sharing. #Params (s) denotes the parameters of the shared backbone.

NAS is designed to allow incremental learning and evades catastrophic forgetting. We evaluate the performance on the COCO keypoint detection task. The train2017 set includes 57K images and 150K person instances annotated with 17 keypoints, the val2017 set contains 5K images. We add the pose estimation task to MDL-NAS after it has been fine-tuned on classification, detection, and segmentation tasks. We freeze the task-shared parameters and only leverage the task-specific parameters to fit the pose estimation task. All the models are trained under MMPose [4] framework. We use the top-down methods [11] and set the training recipe as: Adam optimizer with the base learning rate $5e-4$ that is dropped to $5e-5$ and $5e-6$ at the 170th and 200th epochs, respectively. The input image size is cropped to 256×192. We report average precision with different thresholds and different object sizes: AP, AP^{50}, AP^{75}, AP^{M} and AP^{L}.

G. Latency of MDL-NAS

We compare the latency of MDL-NAS with that of the baselines, as shown in Tab. 3. The latency of these methods are measured on a Tesla V100 GPU. MDL-NAS-B† with mask/sequential sharing policy surpasses AutoFormer 4.5/5.3 units under S-Score with comparable latency while MDL-NAS-T‡ with mask/sequential sharing policy outperforms Swin-T by 2.1/2.8 units , further demonstrating the efficacy of MDL-NAS.

H. Structures of MDL-NAS

Tab. 1 and Tab. 2 gives the transformer architectures searched by our MDAL-NAS† and MDAL-NAS‡ with mask or sequential sharing policy for different vision tasks in the main results, respectively.

I. The effect of inside layer parameter sharing

In this part, we further investigate the efficacy of our parameter sharing policy that allows different tasks to share partial parameter inside each layer and monopolize the rest. A straightforward baseline to our parameter sharing policy is that only allows different tasks either sharing or monopolizing all parameters in one layer. Specifically, we set the fine search space of each layer in MDL-NAS-B/T as $\{0, 1\}$ and jointly finetune the supernet for the three tasks (i.e., classification, detection and segmentation) with the same training recipe (i.e., $1\times$ training schedule) used in the ablation study. Then, we use the proposed search algorithm to search the optimal backbones for such baseline, which is denoted as MDL-NAS-B/T-AB4. As shown in Tab. 4, MDL-NAS-B/T with sequential sharing policy outperforms MDL-NAS-B/T-AB4 by 0.5/2.1 units, respectively, further demonstrating the efficacy of our parameter sharing policy that uses task-shared parameters inside each layer for learning task-reciprocal features while using the rests as task-specific parameters for mitigating conflicts.

	Top-1 Acc.	AP^b	$mIoU_{ss}$	#Params (t)	S-Score
MDL-NAS-B† + mask (s)	83.0	48.4	46.2	293M	177.6
MDL-NAS-B† + mask (j)	82.6	48.2	50.1	235M	180.9
MDL-NAS-B† + seq (s)	83.1	48.3	46.4	311M	177.8
MDL-NAS-B† + seq (j)	82.9	48.0	50.8	256M	181.7

Table 5. Multi-domain learning vs separate learning.

J. Multi-domain learning vs separate learning

We retrain the searched architecture of MDL-NAS for the three tasks separately. The results are reported in Tab. 5. The overall performance of multi-domain training using fewer parameters is much superior to that of separate training. As depicted in Fig. 2 in the main paper, we discuss that some tasks are elevated whereas some have conflicts against each other under multi-domain learning. Combining the results of Fig. 2 and Tab. 5 reveals that MDL-NAS can minimize the conflict and enjoy benefits.

K. Limitation

In this work, we propose MDL-NAS, a unified framework that concurrently learns multiple vision tasks under different dataset domains. MDL-NAS is storage-efficient since multiple models with a majority of shared parameters in the backbone can be deposited into a single one. However, from the experiments, we observe that the parameters of head network for some tasks are relatively large, yet MDL-NAS does not consider the parameter sharing in the head for all tasks. In the future, we would design an encoder-decoder structure that enables all tasks to share parameters throughout the network rather than only backbone network.

References

[1] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu,

Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 3

[2] 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. 1

[3] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 3

[4] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020. 4

[5] MMClassification Contributors. Openmmlab's image classification toolbox and benchmark. https://github.com/open-mmlab/mmclassification, 2020. 3

[6] 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. 3

[7] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. *arXiv preprint arXiv:2203.16527*, 2022. 3

[8] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 3

[9] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 2

[10] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 3

[11] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 466–481, 2018. 4

[12] 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*, pages 702–717. Springer, 2020. 3