MTNAS: Search Multi-Task Networks for Autonomous Driving

Hao Liu[1], Dong Li[2], JinZhang Peng[2], Qingjie Zhao[1], Lu Tian[2], and Yi Shan[2]

[1] Beijing Institute of Technology, Beijing,CHN {3120181007,zhaoqj}@bit.edu.cn
[2] Xilinx Inc. Beijing,CHN {dongl,jinzhang,lutian,yishan}@xilinx.com

Abstract. Multi-task learning (MTL) aims to learn shared representations from multiple tasks simultaneously, which has yielded outstanding performance in widespread applications of computer vision. However, existing multi-task approaches often demand manual design on network architectures, including shared backbone and individual branches. In this work, we propose MTNAS, a practical and principled neural architecture search algorithm for multi-task learning. We focus on searching for the overall optimized network architecture with task-specific branches and task-shared backbone. Specifically, the MTNAS pipeline consists of two searching stages: branch search and backbone search. For branch search, we separately optimize each branch structure for each target task. For backbone search, we first design a pre-searching procedure t1o pre-optimize the backbone structure on ImageNet. We observe that searching on such auxiliary large-scale data can not only help learn low-/mid-level features but also offer good initialization of backbone structure. After backbone pre-searching, we further optimize the backbone structure for learning task-shared knowledge under the overall multi-task guidance. We apply MTNAS to joint learning of object detection and semantic segmentation for autonomous driving. Extensive experimental results demonstrate that our searched multi-task model achieves superior performance for each task and consumes less computation complexity compared to prior hand-crafted MTL baselines. Code and searched models will be released at https://github.com/RalphLiu/MTNAS.

Keywords: Multi-task learning, neural architecture search, autonomous driving

1 Introduction

Multi-task learning (MTL) [1] is one of the popular research topics among a broad family of machine learning algorithms, which aims at learning multiple tasks simultaneously. By learning shared representations across different tasks, MTL can further improve the performance for each task and reduce model complexity for inference. With the remarkable success of deep Convolutional Neural Networks (CNNs), Multi-task networks have shown outstanding performance in widespread applications of computer vision such as joint learning of face detection and landmark prediction [2], object detection and instance segmentation [3],

pose estimation and action recognition [4]. MTL also has great significance in practical systems (e.g. autonomous driving) where both high performance and fast inference speed are required, especially on the resource-constrained devices.

However, most of the existing MTL methods directly use hand-crafted network architectures for joint learning of multiple tasks. This might be sub-optimal because: (1) The backbone is typically designed for large-scale image classification tasks (e.g., ImageNet). Thus the backbone architecture is not adaptive for different downstream tasks;(2) The branch often consists of simple architectures (e.g., a few stacked convolution or deconvolution layers), which may not sufficiently transfer the shared representations to each specific task.

Recently, neural architecture search (NAS) has achieved great progress by automatically seeking the optimal network architectures, instead of relying on expert knowledge and tedious trials. Much effort has been made on employing NAS for different tasks such as image classification [6,7,8,9], object detection [10,11,12] and semantic segmentation [13,14]. However, there have been little considerations on NAS for MTL in a unified framework. It is infeasible to simply apply existing NAS methods for jointly learning multi-task networks. First, proxy datasets/tasks are often required for searching architectures [6,7,8,9], which are often based on relatively small input and aim at classifying images only. The resulting structure is likely to be sub-optimal for other tasks, especially for those requiring high-resolution inputs. Although recent work [15] has been explored by directly optimizing the target task, it mainly focuses on the single basic classification task. Second, for more challenging tasks beyond classification, existing NAS algorithms only search part of networks, e.g., backbone [11,13], FPN structure [10] and ASPP architecture [14], which would not suffice the requirement of overall optimization for multiple tasks. Third, conventional ImageNet pre-training procedure is usually adopted to learn initialized network weights but can not optimize the network architectures for target downstream tasks.

To alleviate these problems, we propose a multi-task neural architecture search (MTNAS) algorithm in this work. We aim to optimize an *overall* network architecture for MTL with two stages: branch search and backbone search. For branch search, we optimize each branch architecture for each task separately. We observe that searching for *task-specific* branch architectures helps better adapt the feature representations to each target task. For backbone search, our goal is to optimize *task-shared* backbone architecture for better learning general shared representations with different tasks. To this end, we first design a pre-searching procedure to learn the initialized backbone architecture and further optimize it under the overall multi-task guidance. Unlike ImageNet pre-training, our pre-searching procedure can optimize architecture parameters and network weights simultaneously, offering good initialization for subsequent optimization. Our MTNAS pipeline is built on the recent differentiable NAS algorithms [9,16,17] by searching for the optimal computation cells in both of branch and backbone search stages.

We apply the proposed MTNAS approach for joint learning of object detection and semantic segmentation for autonomous driving using both single

and mixed datasets. We focus on automatically generating light-weight multi-task networks for fast inference speed and high performance in such practical scenarios. Experimental results consistently demonstrate our MTNAS method performs favorably against the existing single-task / multi-task learning baselines. In particular, on the challenging mixed-set benchmark (i.e., combination of CityScapes, Waymo and BDD100K), our searched multi-task model, with 65% FLOPs only, achieves 3.5% higher mAP on detection and 2.0% higher mIoU on segmentation compared to the hand-crafted MTL counterpart. Besides, compared to the conventional ImageNet pre-training which only updates the network weights, our pre-searching procedure achieves superior performance (e.g., 1.2% gain on segmentation) by simultaneously optimizing network weights and architectures.

The main contributions of this paper are summarized as follows:

- We propose a practical and principled neural architecture search algorithm beyond single-task to multi-task learning. We search for the complete architectures instead of sub-optimal parts of network structure by optimizing task-specific branches and task-shared backbone.
- We develop a simple but effective pre-searching procedure by simultaneously optimizing the network weights and architectures for backbone. Such scheme serves as a better alternative of conventional ImageNet pre-training and provides good initialization of backbone structure for subsequent optimization.
- We apply the proposed MTNAS method to the joint learning of object detection and semantic segmentation in the autonomous driving scenarios. Extensive experimental results demonstrate the superiority of our searched multi-task model over the existing hand-crafted MTL baselines in terms of recognition performance and computation complexity.

2 Related Work

2.1 Multi-Task Learning

Multi-task learning is a learning framework where multiple tasks are learned simultaneously [1]. By sharing commonalities across different tasks, MTL has widely shown promising performance for various applications [2,3,4]. Owing to the shared architectures and weights, multi-task networks can also improve the inference efficiency compared to separate models of each task.

One line of recent research on MTL is balancing different tasks [18,19] or seeking solutions with trade-offs between different tasks [20]. These works either learn fixed optimal weights or learn a set of optimal solutions. Another line of literature on MTL is investigating the optimal strategies of sharing network activations or parameters across different tasks [21,22,23]. Our approach is orthogonal to these strategies as we focus on searching optimal network architectures by jointly learning multiple tasks in this work.

Recently, some work has attempted to use one single network architecture for MTL by exploiting network redundancies to pack multiple tasks [24], developing

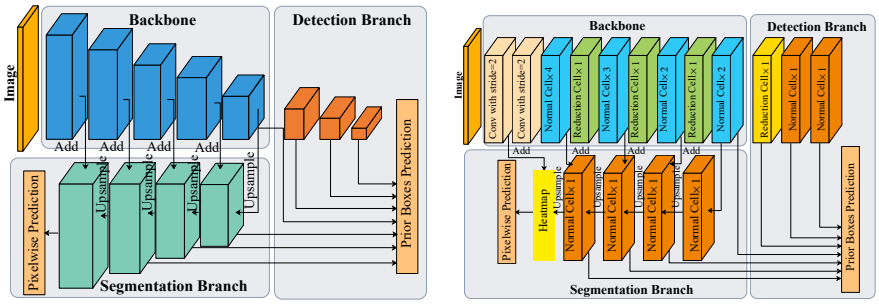
multiple internal virtual models with different configurations for different tasks [25] or dynamically selecting the optimal model from a pool of candidates [26]. However, these approaches are often built for different classification tasks (e.g., ImageNet classification, scene classification, and fine-grained classification). Differently, our method can address more challenging tasks beyond classification.

Previous work that is mostly related to our method includes [27,28]. The work of [27] builds a self-organized neural network by dynamically composing different function blocks for each input. The work of [28] explores different evolution ways of routing and shared modules in MTL. Our method differs from these related approaches in three aspects. First, they perform architecture search by reinforcement learning or evolutionary algorithms, while we employ the efficient differentiable NAS to search for the optimal cell structure. Second, only final fully-connected layers are routed in the network by [27], whereas our method aims to optimize the overall multi-task network architecture instead of parts of the network. Third, these methods are only applied to classification tasks. Differently, we bring the best practices to realize a multi-task NAS on more challenging tasks (e.g., object detection and semantic segmentation) simultaneously.

2.2 Neural Architecture Search

In the past years, plentiful efforts have been made into manual design of high-performance networks [5,29,30,31,32,33]. Neural architecture search, as an emerging alternative approach, aims at designing networks automatically, which has attracted recent research interests. Three key components of NAS lie in search space, search algorithm and performance evaluator. (1) For search space, existing work constructs the final network architectures by directly searching for the entire network [6,7,34], searching for the repeatable cell structure [9,35,36] or exploiting hierarchical structure [13,37,38]. (2) For search algorithms, some works exploit reinforcement learning (RL) [7,15,34,39] to train recurrent neural network controllers for generating architectural hyperparameters of networks. Evolutionary algorithms (EA) provide an alternative to searching for neural architectures by evolving them with crossover and mutation [8,11]. However, these RL and EA methods tend to require intensive computation even on small input images. Recent work attempts to reduce the computational cost by weight sharing [6] or gradient-based optimization [9,35,40]. (3) For performance evaluator, existing work has explored mono-objective (i.e., accuracy) or multi-objective (e.g., accuracy, latency, and FLOPs) schemes. In this work, we follow the differentiable NAS [9,16,17] to search for the optimal cell structure for both branches and backbone, and combine multiple rewards from different tasks to guide the search procedures in an alternating way.

Most of existing NAS approaches target on image classification tasks, either requiring proxy datasets / tasks [7,9,12,35] or directly optimizing on target tasks [15]. The most recent NAS papers [41,42,43] mainly focus on improving the search algorithm for a single task while we aim to realize a multi-task NAS pipeline. Recent methods have investigated NAS for more tasks, e.g., object detection [11,12,44] and semantic segmentation [13]. However, these methods



(a) MTL Baseline (b) MTNAS

Fig. 1. Illustration of our hand-crafted multi-task network baseline and MTNAS architecture which includes backbone, detection branch, and segmentation branch.

only search parts of networks such as the backbone or other functional units. In contrast, our method extends NAS from single-task to multi-task learning and aims to search for the overall network architecture.

3 Proposed Method

3.1 Hand-Crafted Multi-Task Network Baseline

We first introduce the hand-crafted multi-task network baseline in this section. As shown in Fig. 1 (a), we build a backbone architecture to learn task-shared representations, followed by task-specific decoders that learn adapted representations for each task.

Backbone acts as a feature extractor to extract different abstract levels of shared representations in a multi-task network, which has a great impact on the performance of consecutive tasks. We use ImageNet pre-trained networks (e.g., ResNet) as our backbone but remove the last fully-connected classification layer associated with the pre-training task. We consider joint learning of object detection and semantic segmentation in this work. For the detection branch, we employ the SSD detection head [45] and aggregate prior box predictions from multiple feature maps of different resolutions. For the segmentation branch, similar to FCN [32], we stack several upsampling layers (each followed by ReLU activation and Batch Normalization) for pixel-wise prediction. We also integrate FPN structure [46] to further improve the quality of representations for both tasks.

3.2 Multi-Task Neural Architecture Search

In this section, we first describe our search space design and search algorithm and then introduce the proposed MTNAS pipeline.

Search Space. We follow [9] to search for two types of computation cells (i.e., normal cell and reduction cell) as the building block of network architecture. Each cell is a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_i\}|_{i=0}^{N-1}$ and $\mathcal{E} = \{e(i,j)\}|_{0 \le i < j \le N-1}$. Each node v_i represents a certain feature map in CNNs and each directed edge $e(i,j)$ represents a certain operation $o(i,j) \in \mathcal{O}$ transforming v_i to v_j. In this work, each cell has $|\mathcal{V}| = 7$ nodes (i.e., two input nodes, four intermediate nodes and one output node) and we set $|\mathcal{O}| = 8$ candidate operations [3] as our base search space. The base search space is shared across the entire search process including searching for architectures of backbone and branches.

Search Algorithm. Following [9], we solve the bi-level optimization problem to attain the optimal architecture parameters ϕ:

$$\min_{\phi} \mathcal{L}_{\text{val}}(\omega^*(\phi), \phi) \quad s.t.\ \omega^*(\phi) = \arg\min_{\omega} \mathcal{L}_{\text{train}}(\omega, \phi)\ . \tag{1}$$

where $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ mean the training loss and validation loss, respectively. After obtaining the optimized architecture parameters, we can derive the final network architecture by preserving operations with the two largest probabilities. We also apply strategies of partial connection [17] and early stopping [16,47] to reduce the computation overhead during the multi-task search process. Besides, we apply SyncBN [48] to allow calculation of batch statistics across multiple GPU cards for the challenging tasks (e.g., detection and segmentation) which often require high-resolution input.

Multi-Task Search Pipeline. We denote a multi-task network as $\mathcal{N}(\phi, \omega)$ where $\phi = \{\phi_0, \phi_1, \ldots, \phi_n\}$ and $\omega = \{\omega_0, \omega_1, \ldots, \omega_n\}$ represent the architecture parameters and network weights, respectively. Specifically, the index 0 indicates backbone and $\{i\}|_1^n$ indicate n branches in the multi-task network. Our goal is to search for the overall multi-task model with optimal architectures and weights:

$$\mathcal{N}(\phi^*, \omega^*)|_{\phi^{(0)}, \omega^{(0)}}\ . \tag{2}$$

where $\phi^{(0)}$ and $\omega^{(0)}$ mean the initialization of architecture parameters and network weights, respectively. We note that it is hard to directly optimize the overall multi-task architecture ϕ in practice because (1) Different optimization targets of each task may incur conflicts during searching for the task-specific branch architectures; (2) It requires substantial GPU memory consumption since large search space is caused by optimizing different normal and reduction cells for multiple tasks at the same time. Thus, to reduce the search space and ease the optimization of joint search, we propose a two-stage multi-task search process: branch search and backbone search, as illustrated in Fig. 2. Specifically, we first search for the optimal branch architectures for each task separately and then

[3] *zero, skip-connect, max-pool-3x3, avg-pool3x3, sep-conv-3x3, sep-conv-5x5, dil-conv-3x3, dil-conv5x5*

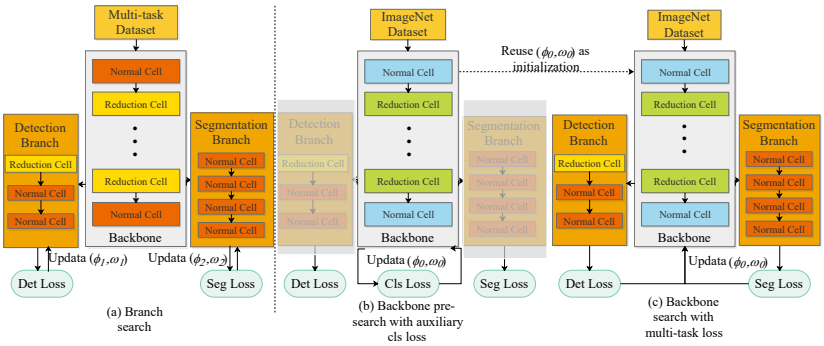


Fig. 2. Proposed multi-task neural architecture search (MTNAS) algorithm pipeline.

search for the optimal backbone architecture under the overall guidance. For the reason of search order, we explain that our goal is to search for task-specific branches and task-shared backbone. When optimizing the backbone, we need to compute the loss from all the tasks. After obtaining the optimal branch architectures, backbone can benefit from each branch architecture and then learn shared knowledge across all tasks, leading to overall optimization for MTL.

S1: Branch Search. The goal of this search stage is to optimize the task-specific branch architectures for each task. Thus, we address each target task separately so that these tasks will not affect each other. In other words, when optimizing i-th branch, only the loss $\mathcal{L}_i$ associated with this task is backpropagated through the network. The architectures and weights of other branches $(\phi_j, \omega_j)|_{j=1,j\neq i}^{n}$ are frozen. The backbone will be initialized with the same cells as the current branch and will be re-initialized when searching for another branch. After the branch search stage, we can obtain the task-specific branch models:

$$\mathcal{N}(\phi_i^*, \omega_i^*)|_{\phi_i^{(0)}, \omega_i^{(0)}}, \quad i \in \{1, 2, \ldots, n\} \ . \tag{3}$$

S2: Backbone Search. The goal of this search stage is to optimize task-shared backbone architectures for all tasks of interest. ImageNet pre-training has been widely used for various vision tasks as it can learn from large-scale data and provide a good weight initialization for different downstream tasks. We observe that large-scale data is also important to help optimize the backbone architecture. Thus, we propose a simple but effective pre-searching procedure to search for an initialized backbone architecture under the guidance of auxiliary ImageNet classification task. To this end, we freeze all branches (including architectures and weights) and append a fully-connected classification layer to the backbone. After pre-searching, a well-initialized backbone model is obtained:

$$\mathcal{N}(\phi_0^{'}, \omega_0^{'})|_{\phi_0^{(0)}, \omega_0^{(0)}} \ . \tag{4}$$

The proposed pre-searching procedure can be viewed as a better alternative of ImageNet pre-training in the context of NAS. The auxiliary large-scale

data not only helps learn the low-/mid-level features but also helps learn general cell architectures. Unlike ImageNet pre-training which only updates the network weights, our pre-searching can update both of architecture parameters and network weights, and provide better initialization of backbone structure for subsequent optimization.

With the pre-searched architecture parameters and network weights of the backbone as initialization, we further optimize the backbone structure under the overall multi-task guidance. Specifically, we design an alternating optimization strategy by incorporating iterative supervision from each task. For each iteration, only one loss from a single task will be backpropagated. The benefits of such scheme are two-fold. First, it helps improve training stability for the collaborative optimization of multiple tasks empirically. Second, it is flexible for MTL since it enables the utilization of datasets where only annotations of a single task are available. That is, we do not require complete labeled data of all tasks on a single dataset. In the stage of backbone search, branch architectures remain unchanged. Backbone can benefit from the optimal branches to learn shared knowledge from all tasks. The task-shared backbone will be generated after this backbone search stage:

$$\mathcal{N}(\phi_0^*, \omega_0^*)|_{\phi_0^{(0)}=\phi_0', \omega_0^{(0)}=\omega_0'} \ . \tag{5}$$

The entire MTNAS algorithm pipeline is also described in Algorithm 1. We show in Fig. 1 (b) an example of the final overall architecture of the searched multi-task network.

Multi-Task Finetuning. After obtaining the optimized multi-task network architecture, we further finetune the overall network weights $\hat{\omega^*} \leftarrow \omega^*$ and the overall network architectures $\phi_i^*|_{i=0}^n$ remain unchanged during this process.

4 Experiments

4.1 Datasets and Evaluation Metrics

In this work, we apply the proposed MTNAS method to joint learning of object detection and segmentation for autonomous driving. Our experiments are conducted on four public datasets, including KITTI [49], CityScapes [50], BDD100K [51] and Waymo [52]. The KITTI dataset contains 7,481 training images and 7,518 test images with three categories of *car*, *pedestrian* and *cyclist* for object detection. The Cityscapes dataset contains 2,975 training, 500 validation and 1525 test images of 19 categories for semantic segmentation. Waymo and BDD100K are recent large-scale datasets with diverse autonomous driving scenes and are challenging for both detection and segmentation tasks. BDD100K includes annotations for both detection and segmentation while Waymo includes detection annotations only. Note that our method can be flexibly applied to the case that annotations are available for a certain task but not for other tasks. Based on these datasets, we create two sets of benchmarks for evaluating multi-task networks in our experiments.

Algorithm 1 MTNAS - Multi-Task Neural Architecture Search

Input:
Base search space $\mathcal{O}$.
ImageNet dataset $\mathcal{D}_0$.
Task-specific datasets $\mathcal{D}_i|_{i=1}^n$ associated with n tasks.
Hyperparameters of early stopping policy: K, T.
Branch Search:
for $i = 1 \rightarrow n$ **do**
Initialize ϕ_i with task-specific search space and initialize ω_i randomly.
repeat
Update (ϕ_i, ω_i) according to Eq. 1
until $\mathcal{N}(\phi_i, \omega_i)$ does not change for K iterations.
end for
Output the resulting branches $(\phi_i^*, \omega_i^*)|_{i=1}^n$.
Backbone Search:
Freeze branches and add an auxiliary ImageNet classification layer.
Initialize ϕ_0 and ω_0 randomly.
while number of *skip-connect* operation $\leq T$ **do**
Update (ϕ_0, ω_0) according to Eq. 1
end while
Output the pre-searched backbone $(\phi_0^{'}, \omega_0^{'})$.
Activate branches and remove the ImageNet classification layer.
Initialize (ϕ_0, ω_0) with $(\phi_0^{'}, \omega_0^{'})$ and fix branch architectures $\phi_i|_{i=1}^n$.
repeat
Update (ϕ_0, ω_0) according to Eq. 1
until $\mathcal{N}(\phi_0, \omega_0)$ does not change for K iterations.
Output the resulting backbone $\mathcal{N}(\phi_0^*, \omega_0^*)$.
Output:
Derive the final optimal multi-task network architecture $\mathcal{N}(\phi_i^*, \omega_i^*)|_{i=0}^n$.

Single Set. We apply a single small dataset for each task in this version. Specifically, KITTI is used for detection and CityScapes is used for segmentation. This single-set benchmark is used for performance comparisons with existing work on each task.

Mixed Set. We also employ large-scale data to further improve the performance for the practical application. In detail, we combine Waymo and BDD100K by merging their common categories for detection and combine CityScapes and BDD100K for segmentation similarly. This leads to 4 classes for detection and 16 classes for segmentation. We also randomly divide the mixed data into *train*, *validation* and *test* sets. For object detection, we have 120k, 2k, 10k images for *train*, *validation* and *test* sets, respectively. For semantic segmentation, we have 10k, 500, 1500 images for *train*, *validation* and *test* sets, respectively. This mixed-set benchmark is used for our main multi-task results.

For evaluation metrics, we use the standard mean Average Precision (mAP) for detection and mean Intersection over Union (mIoU) for segmentation.

4.2 Implementation Details

MTL Baseline. We take Resnet-18 as the backbone of our hand-crafted multi-task network baseline. For all tasks, we resize images to the same 320×512 resolution as input. We train the multi-task network with a batch size of 32 for 150k iterations on 2 NVIDIA V100 GPUs. We use SGD for optimization with an initial learning rate of 0.01 (decreased with a linear cosine policy), momentum of 0.9 and weight decay of 3×10^{-5}. As for balancing different branches' losses, we simply set the loss weights of detection and segmentation as 1:1.

MTNAS. We set $K = 10$ and $T = 2$ in Algorithm 1. We use the same batch size of 32 with MTL baseline but different initial learning rate of 0.1 for searching. The maximum iterations of searching for each branch and backbone are set as 5k and 10k, respectively. On the single-set benchmark, the branch search, backbone search and multi-task finetuning processes cost 2, 12, 4 GPU days, respectively. All of our experiments are conducted on PyTorch.

4.3 MTL Results on Mixed Set

After obtaining the searched cells for both backbone and branches on the mixed set, we stack a light-weight multi-task network and further finetune its weights. We show the main multi-task results on the mixed-set benchmark in Table 1. We compare the single-task baseline (i.e., SSD for detection and FCN for segmentation) and the multi-task baseline described in Section 3.1. We also implement recent MTL methods on our mix-set benchmark including Pareto MTL [20] and TripleNet [53]. Since there are little considerations of NAS methods on MTL, we also implement several related differentiable NAS baselines and extend them to our multi-task setting by searching on a proxy CIFAR10 dataset. Table 1 shows that the proposed MTNAS method achieves the best performance on both tasks and simultaneously consumes the least computation cost compared to the single-task, multi-task and NAS baselines. (1) Our method obtains significant improvement over each single task baseline (e.g., 3.5% mIoU gain for segmentation). By sharing representations in the multi-task network, the computation complexity of the model is largely reduced. We only require around 35% fewer FLOPs than a summation of two separate models (i.e., 9.0 vs. 13.5+12.1=25.6) but obtain higher performance (i.e., 43.7% vs. 41.5% for detection, 46.2% vs. 42.7% segmentation). (2) Our method outperforms the hand-crafted MTL baseline with 3.5% higher mAP for detection, 2.0% higher mIoU for segmentation and only consumes 65% fewer FLOPs. We also compare MTNAS with other state-of-the-art MTL methods [20,53] using the same network and experimental setting as MTL baseline. The results show that MTNAS outperforms Pareto MTL by 1.9% for detection and 1.3% for segmentation, and outperforms TripleNet by 1.5% for detection and 1.0% for segmentation. (3) We compare our method with those demanding proxy tasks (CIFAR10 classification). The results show the consistent superiority of MTNAS, e.g., 43.7% (Ours) vs. 38.6% (DARTS) for detection.

Table 1. Performance comparisons of multi-task learning on mixed set in terms of detection accuracy (mAP), segmentation accuracy (mIoU) and computation complexity (FLOPs). † denotes conventional Imagenet pre-training and * denotes the proposed ImageNet pre-searching procedure.

Methods		mAP (%)	mIoU (%)	FLOPs (G)
Manual design†	SSD [45]	41.5	-	13.5
	FCN [32]	-	42.7	12.1
	MTL baseline	40.2	44.2	13.7
	Pareto MTL [20]	41.8	44.9	13.7
	TripleNet [53]	42.2	45.2	21.2
Search on proxy tasks	DARTS [9]	36.2	40.1	9.7
	DARTS† [9]	38.6	42.6	9.7
	PC-DARTS [17]	34.0	43.0	11.4
	PC-DARTS† [17]	36.0	44.5	11.4
	DARTS+ [16]	34.8	44.2	11.4
	DARTS+† [16]	37.7	45.0	11.4
Search on target tasks	Random search [54]	38.6	41.5	10.4
	Co-search	41.1	43.0	10.5
	Ours†, *w/o* branch search	41.4	45.5	10.9
	Ours†, *w/o* backbone search	40.5	44.5	11.8
	Ours†	42.1	45.4	**9.0**
	Ours*	**43.7**	**46.2**	**9.0**

We show our contributions from the algorithmic components in the last group of Table 1. (1) By discarding either branch or backbone search stage, the performance on both tasks will drop. This validates the necessity of searching for overall architecture in MTL. (2) Compared with the results of the last two rows in Table 1, we demonstrate the effectiveness of our pre-searching procedure. Unlike ImageNet pre-training that updates network weights only, our pre-searching procedure can optimize both of architecture parameters and network weights and provide good initialization for the subsequent backbone search process. (3) To demonstrate the effectiveness of our search method, we follow [54] to implement the random search baseline and the result is worse than our baseline, and we tried the co-search scheme (searching backbone and branches at same time) but get worse results than MTNAS, because we need to reduce the batch size to fit the memory limit in the co-search manner (only 2 images can be loaded for training with 32G memory) and the performance is harmed. We further test the latency on P100: MTL baseline (8.4ms), SSD (8.2ms), FCN (8.1ms), MTNAS (6.2ms). More results on CIFAR10 [55], VOC12 [56] and cross dataset are in our supplementary material.

We show more detailed ablation studies on backbone search as well as the searching time in Table 2. In these experiments, we use the searched architecture for two branches. (1) Our method (PS+BS) outperforms that of using the pre-trained hand-crafted ResNet as backbone (PT) by a large margin, which validates the superiority of optimizing the backbone structure. (2) Our method outperforms pre-searching only or searching with the multi-task loss only, which demonstrates that both auxiliary large-scale data and task-specific data are ben-

Table 2. Ablation studies with respect to backbone search on the mixed-set benchmark. PS: Pre-search backbone on ImageNet. BS: Search backbone with the multi-task loss. PT: Pre-train backbone on ImageNet.

PS	BS	PT	Time (GPU days)	mAP (%)	mIoU (%)
		√	14	40.5	44.5
√			8	40.5	44.3
	√		12	41.2	44.7
	√	√	26	42.1	45.4
√	√		20	**43.7**	**46.2**

Table 3. Performance comparisons on single set in terms of per-class AP and mAP on KITTI, mIoU accuracy on CityScapes and FLOPs.

Methods	*Car*	*Pedestrian*	*Cyclist*	mAP (%)	mIoU (%)	FLOPs (G)
PSPNet [58]	-	-	-	-	81.2	412.2
DeepLabv2 [59]	-	-	-	-	63.1	457.8
DeepLabv3+ [60]	-	-	-	-	82.1	496
SegNet [61]	-	-	-	-	57.0	286
SQ [62]	-	-	-	-	59.8	270
DPN [63]	-	-	-	-	59.1	270
SSD-VGG16 [64]	75.9	50.6	50.2	58.9	-	157.4
Faster-RCNN [65]	81.6	65.6	63.4	70.2	-	181
RTSeg-MobileNet [57]	-	-	-	-	61.5	13.8
FCN-ResNet18 [32]	-	-	-	-	51.1	12.1
SSD-ResNet18 [45]	77.4	56.1	54.6	62.7	-	13.5
SqueezeDet [66]	66.1	-	-	-	-	9.7
YOLOv2 [67]	62.8	-	-	-	-	35
MTNAS (Ours)	79.2	65.4	61.8	68.8	63.4	12.6

eficial for optimizing the backbone architecture. (3) Compared to ImageNet pre-training, our pre-searching procedure obtains higher performance on both tasks (43.7% vs. 42.1% for detection, 46.2% vs. 45.4% for segmentation).

4.4 Comparisons to State-of-the-Arts on Single Set

We search for another light-weight multi-task network on the single set. Table 3 compares our MTNAS with the state-of-the-art methods on each task. We group these existing methods by their FLOPs and our searched network achieves competitive performance with other light-weight models on both tasks. For example, compared to RTSeg-MobileNet [57] for segmentation, we achieve 1.9% higher mIoU and consumes less FLOPs (12.6G vs. 13.8G). Compared to SSD-ResNet18 [45] for detection, we achieve 6.1% higher mAP and also consumes less FLOPs (12.6G vs. 13.5G).

4.5 MTNAS Architecture and Discussions

Fig. 3 shows our searched backbone architectures on the target tasks and those searched by DARTS on the proxy task. Our searched normal cell has a deeper

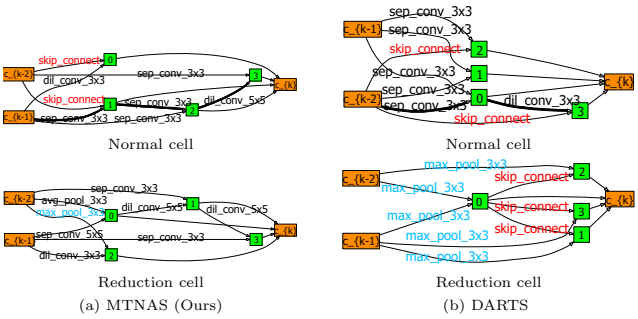


Fig. 3. Searched cell architectures of backbone by our MTNAS algorithm and the original DARTS method.

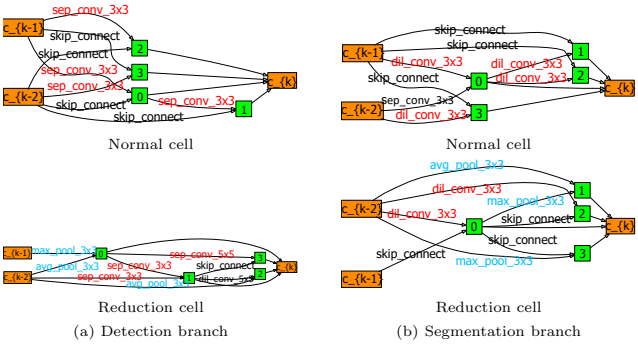


Fig. 4. Searched cell architectures for our detection and segmentation branches by our method.

structure which is likely to help improve the representation ability of networks. In terms of reduction cell, DARTS suffers from the collapse problem as it only generates weight-free operations like max-pooling and skip-connection. We observe that these weight-free operations are easier to converge, which incur selection bias during the optimization process. Besides, many pooling operations will cause location information loss and negatively affect the performance of detection and segmentation tasks. In contrast, our MTNAS method leads to more diverse architecture for learning richer information in the reduction cell.

Fig. 4 shows our searched branch architectures. By optimizing the branch structure for each task separately, we can learn task-specific cell architectures, e.g., separate convolution for detection and dilated convolution for segmentation, which are crucial for good performance.

4.6 Qualitative Evaluations

Fig. 5 visualizes some examples of MTL results on the BDD100K dataset. In general, our MTNAS method can achieve more accurate detection and segmentation results compared to the hand-crafted MTL baseline. For example, MTNAS can better detect small objects (e.g., *traffic sign*) in the first row and crowd people

(a) Input (b) MTL baseline (c) MTNAS

Fig. 5. Example results of multi-task learning on the BDD100K dataset. Our MTNAS method can achieve more accurate detection and segmentation results compared to the hand-crafted multi-task network baseline.

in the second row. Besides, MTNAS can better segment the objects with a well-defined shape (e.g., *car*) and amorphous background regions (e.g., *sidewalk* and *sky*) as shown in the last three rows of Fig. 5.

5 Conclusion

In this paper, we propose a practical and principled neural architecture search algorithm for multi-task learning, named MTNAS. Our method aims to search for the overall optimized network architecture for multi-task learning with two stages. For branch search, we separately optimize the task-specific neural architecture for each branch based on its own optimization objective. For backbone search, we first propose a pre-searching procedure to obtain the initial backbone structure and then refine them under the overall multi-task guidance. We apply the proposed MTNAS pipeline for the challenging autonomous driving scenarios by jointly learning object detection and semantic segmentation. Experimental results demonstrate our searched multi-task model surpasses the hand-crafted single-task and multi-task baselines largely and consumes less computation cost. We believe that our proposed method can provide new insights into neural architecture search on multi-task learning and has broad real-world applications.

References

1. Caruana, R.: Multitask learning. Machine learning **28** (1997) 41–75 1, 3
2. Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multitask cascaded convolutional networks. SPL **23** (2016) 1499–1503 1, 3
3. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV. (2017) 1, 3
4. Luvizon, D.C., Picard, D., Tabia, H.: 2d/3d pose estimation and action recognition using multitask deep learning. In: CVPR. (2018) 2, 3
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016) 4
6. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameters sharing. In: ICML. (2018) 2, 4
7. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: CVPR. (2018) 2, 4
8. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: AAAI. (2019) 2, 4
9. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. ICLR (2018) 2, 4, 6, 11
10. Ghiasi, G., Lin, T.Y., Le, Q.V.: Nas-fpn: Learning scalable feature pyramid architecture for object detection. In: CVPR. (2019) 2
11. Chen, Y., Yang, T., Zhang, X., Meng, G., Xiao, X., Sun, J.: Detnas: Backbone search for object detection. In: NeurIPS. (2019) 2, 4
12. Peng, J., Sun, M., ZHANG, Z.X., Tan, T., Yan, J.: Efficient neural architecture transformation search in channel-level for object detection. In: NeurIPS. (2019) 2, 4
13. Liu, C., Chen, L.C., Schroff, F., Adam, H., Hua, W., Yuille, A.L., Fei-Fei, L.: Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: CVPR. (2019) 2, 4
14. Chen, L.C., Collins, M., Zhu, Y., Papandreou, G., Zoph, B., Schroff, F., Adam, H., Shlens, J.: Searching for efficient multi-scale architectures for dense image prediction. In: NeurIPS. (2018) 2
15. Cai, H., Zhu, L., Han, S.: Proxylessnas: Direct neural architecture search on target task and hardware. In: ICLR. (2019) 2, 4
16. Liang, H., Zhang, S., Sun, J., He, X., Huang, W., Zhuang, K., Li, Z.: Darts+: Improved differentiable architecture search with early stopping. arXiv preprint arXiv:1909.06035 (2019) 2, 4, 6, 11
17. Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.J., Tian, Q., Xiong, H.: Pc-darts: Partial channel connections for memory-efficient architecture search. In: ICLR. (2019) 2, 4, 6, 11
18. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: CVPR. (2018) 3
19. Chen, Z., Badrinarayanan, V., Lee, C.Y., Rabinovich, A.: Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In: ICML. (2018) 3
20. Lin, X., Zhen, H.L., Li, Z., Zhang, Q.F., Kwong, S.: Pareto multi-task learning. In: NeurIPS. (2019) 3, 10, 11
21. Misra, I., Shrivastava, A., Gupta, A., Hebert, M.: Cross-stitch networks for multi-task learning. In: CVPR. (2016) 3
22. He, X., Zhou, Z., Thiele, L.: Multi-task zipping via layer-wise neuron sharing. In: NeurIPS. (2018) 3

23. Meyerson, E., Miikkulainen, R.: Beyond shared hierarchies: Deep multitask learning through soft layer ordering. In: ICLR. (2018) 3
24. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. In: CVPR. (2018) 3
25. Kim, E., Ahn, C., Torr, P.H., Oh, S.: Deep virtual networks for memory efficient inference of multiple tasks. In: CVPR. (2019) 4
26. Ahn, C., Kim, E., Oh, S.: Deep elastic networks with model selection for multi-task learning. In: ICCV. (2019) 4
27. Rosenbaum, C., Klinger, T., Riemer, M.: Routing networks: Adaptive selection of non-linear functions for multi-task learning. In: ICLR. (2018) 4
28. Liang, J., Meyerson, E., Miikkulainen, R.: Evolutionary architecture search for deep multitask networks. In: GECCO. (2018) 4
29. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: CVPR. (2018) 4
30. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: ICLR. (2015) 4
31. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: ECCV. (2018) 4
32. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. (2015) 4, 5, 11, 12
33. Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., Sun, J.: Detnet: Design backbone for object detection. In: ECCV. (2018) 4
34. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: ICLR. (2016) 4
35. Xie, S., Zheng, H., Liu, C., Lin, L.: SNAS: stochastic neural architecture search. In: ICLR. (2019) 4
36. Dong, J.D., Cheng, A.C., Juan, D.C., Wei, W., Sun, M.: Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In: ECCV. (2018) 4
37. Liu, H., Simonyan, K., Vinyals, O., Fernando, C., Kavukcuoglu, K.: Hierarchical representations for efficient architecture search. In: ICLR. (2018) 4
38. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: CVPR. (2019) 4
39. He, Y., Lin, J., Liu, Z., Wang, H., Li, L.J., Han, S.: Amc: Automl for model compression and acceleration on mobile devices. In: ECCV. (2018) 4
40. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: CVPR. (2019) 4
41. Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P.J., Tan, M., Huang, T., Song, X., Pang, R., Le, Q.: Bignas: Scaling up neural architecture search with big single-stage models. In: ECCV. (2020) 4
42. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once-for-all: Train one network and specialize it for efficient deployment. In: ICLR. (2019) 4
43. He, C., Ye, H., Shen, L., Zhang, T.: Milenas: Efficient neural architecture search via mixed-level reformulation. In: CVPR. (2020) 11993–12002 4
44. Guo, J., Han, K., Wang, Y., Zhang, C., Yang, Z., Wu, H., Chen, X., Xu, C.: Hit-detector: Hierarchical trinity architecture search for object detection. In: CVPR. (2020) 11405–11414 4
45. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV. (2016) 5, 11, 12

46. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR. (2017) 5
47. Chen, X., Xie, L., Wu, J., Tian, Q.: Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In: ICCV. (2019) 6
48. Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., Sun, J.: Megdet: A large mini-batch object detector. In: CVPR. (2018) 6
49. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR. (2012) 8
50. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR. (2016) 8
51. Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., Darrell, T.: Bdd100k: A diverse driving video database with scalable annotation tooling. arXiv preprint arXiv:1805.04687 (2018) 8
52. : Waymo open dataset: An autonomous driving dataset (2019) 8
53. Cao, J., Pang, Y., Li, X.: Triply supervised decoder networks for joint detection and segmentation. In: CVPR. (2019) 10, 11
54. Li, L., Talwalkar, A.: Random search and reproducibility for neural architecture search. In: Uncertainty in Artificial Intelligence, PMLR (2020) 367–377 11
55. Krizhevsky, A., et al.: Learning multiple layers of features from tiny images. (2009) 11
56. Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: International Conference on Computer Vision (ICCV). (2011) 11
57. Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., Jagersand, M.: Rtseg: Real-time semantic segmentation comparative study. In: ICIP. (2018) 12
58. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR. (2017) 12
59. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. TPAMI **40** (2017) 834–848 12
60. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV. (2018) 12
61. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. TPAMI **39** (2017) 2481–2495 12
62. Treml, M., Arjona-Medina, J., Unterthiner, T., Durgesh, R., Friedmann, F., Schuberth, P., Mayr, A., Heusel, M., Hofmarcher, M., Widrich, M., et al.: Speeding up semantic segmentation for autonomous driving. In: MLITS, NeurIPS Workshop. (2016) 12
63. Liu, Z., Li, X., Luo, P., Loy, C.C., Tang, X.: Semantic image segmentation via deep parsing network. In: ICCV. (2015) 12
64. Kim, H., Lee, Y., Yim, B., Park, E., Kim, H.: On-road object detection using deep neural network. In: ICCE-Asia. (2016) 12
65. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NeurIPS. (2015) 12
66. Wu, B., Iandola, F., Jin, P.H., Keutzer, K.: Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In: CVPR. (2017) 12
67. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: CVPR. (2017) 12