Joint-DetNAS: Upgrade Your Detector with NAS, Pruning and Dynamic Distillation

Lewei Yao[1*] Renjie Pi[1*] Hang Xu[2†] Wei Zhang[2] Zhenguo Li[2] Tong Zhang[1]
[1]Hong Kong University of Science and Technology [2]Huawei Noah's Ark Lab

Abstract

We propose Joint-DetNAS, a unified NAS framework for object detection, which integrates 3 key components: Neural Architecture Search, pruning, and Knowledge Distillation. Instead of naively pipelining these techniques, our Joint-DetNAS optimizes them jointly. The algorithm consists of two core processes: ***student morphism*** *optimizes the student's architecture and removes the redundant parameters, while* ***dynamic distillation*** *aims to find the optimal matching teacher. For* ***student morphism****, weight inheritance strategy is adopted, allowing the student to flexibly update its architecture while fully utilize the predecessor's weights, which considerably accelerates the search; To facilitate* ***dynamic distillation****, an elastic teacher pool is trained via integrated progressive shrinking strategy, from which teacher detectors can be sampled without additional cost in subsequent searches. Given a base detector as the input, our algorithm directly outputs the derived student detector with high performance without additional training. Experiments demonstrate that our Joint-DetNAS outperforms the naive pipelining approach by a great margin. Given a classic R101-FPN as the base detector, Joint-DetNAS is able to boost its mAP from 41.4 to 43.9 on MS COCO and reduce the latency by 47%, which is on par with the SOTA EfficientDet while requiring less search cost. We hope our proposed method can provide the community with a new way of jointly optimizing NAS, KD and pruning.*

1. Introduction

Finding the optimal tradeoff between model performance and complexity has always been a core problem for the community. The mainstream approaches aiming at addressing this issue are: Neural Architecture Search (NAS) [13, 8, 40, 15] is proposed to automatically search for promising model architectures; pruning [16, 18, 26] removes redundant parameters from a model while maintaining its performance; and Knowledge Distillation (KD) [14, 5, 17, 38, 9] aims to transfer the learnt knowledge from a cumbersome teacher model to a more compact student model. These methods share the same ultimate goal: boosting the model's performance while making it more compact. However, jointly optimizing them is a challenging task, especially for detection, which is much more complex than classification. In this paper, we propose Joint-DetNAS, a unified framework for detection which jointly optimizes NAS, pruning and KD.

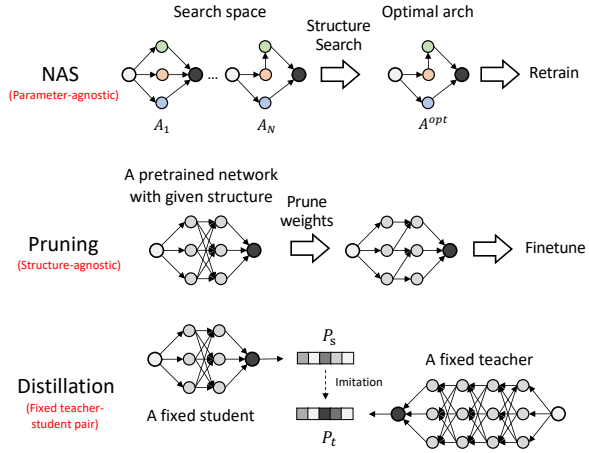


Figure 1: Limitation of NAS, pruning and KD. NAS is parameter-agnostic: The model search and training processes are decoupled, the searched architecture is retrained from scratch; Pruning is structure-agnostic: the pre-trained model has a fixed architecture; KD transfers knowledge between a fixed student-teacher pair while neglecting the structural dependence between the student and the teacher. Our work aims to jointly optimize all three methods.

The aforementioned methods each have some limitations, as illustrated in Figure 1. NAS and pruning only focus on one aspect while neglecting the other: The current de facto paradigm of NAS considers the architecture to be the sole factor that impacts the model's performance, while pruning only takes parameters into account and is structure-agnostic. A recent work [11] has observed an interesting phenomenon: the pruned model's final performance highly depends on its retraining initialization. This observation indicates that the architecture and parameters are closely cou-

*Equal contribution
†Corresponding author: xbjxh@live.com

pled with each other, both of them play important roles in the model's final performance, which motivates us to optimize them jointly.

On the other hand, the architecture of student-teacher pair is arbitrary and fixed during training in conventional KD. However, recent works [9, 23] have pointed out the existence of structural knowledge in KD, which implies that the teacher's architecture has to match with the student to facilitate knowledge transfer. Therefore, we are inspired to incorporate dynamic KD into our framework, where the teacher is dynamically sampled to find the optimal matching for the student.

We propose Joint-DetNAS, a unified framework consisting of two integrated processes: **student morphism** and **dynamic distillation**. **Student morphism** aims to optimize the student's architecture while remove the redundant parameters. To this end, an action space along with a weight inheritance training strategy are carefully designed, which eliminates the prerequisite of backbone's ImageNet pre-training and allows the student to flexibly adjust its architecture while fully utilize the predecessor's weights. **Dynamic distillation** targets at finding the optimal matching teacher and transferring its knowledge to the student. To facilitate teacher search without repeated training, an elastic teacher pool is built to provide sufficient powerful detectors, which trains a super-network only once and obtains all the sub-networks with competitive performances. During the search, we adopt a neat hill climbing strategy to evolve the student-teacher pair. Thanks to weight inheritance and the elastic teacher pool, each student-teacher pair can be evaluated at the cost of fewer epochs and the final obtained student detector requires no additional training

Our framework enables further exploration on the relationship between the architectures of student-teacher pair. We observe two interesting phenomena: (1) a more powerful detector does not necessarily make a better teacher; (2) the capacities of the student and teacher are highly correlated. These facts indicate the existence of structural knowledge and architecture matching in KD for detection.

We conduct extensive experiments to verify the effectiveness of each component (i.e., KD, pruning and the proposed elastic teacher pool) on detection task. Our Joint-DetNAS presents clear performance enhancement over 1) the input FPN baseline, 2) pipelining NAS->pruning->KD. Given a classic R101-FPN as the base detector, our framework is able to boost its AP from 41.4 to 43.9 on MS COCO and reduce its latency by 47%, which is on par with the SOTA EfficientDet [35] while requiring less search cost.

Our contributions are as follows: **1)** We investigate KD and pruning for detection and carefully analyze their effectiveness. **2)** We propose an elastic teacher pool containing sufficient powerful detectors which can be directly sampled without training. **3)** We develop a unified framework which jointly optimizes NAS, pruning and dynamic KD. **4)** Extensive experiments are conducted to investigate the matching pattern between the student-teacher pair and verify the performance of our proposed framework.

2. Related Work

Object Detection. State-of-the-art detection networks can be classified as one-stage, two-stage and anchor-free detectors. One-stage detectors such as [27, 22, 28] directly makes prediction on the feature maps. Two-stage detectors such as [29, 19] uses a region proposal network (RPN) to identify the foreground boxes and passes the corresponding features to an RCNN head for final prediction. Recently, works such as [37, 10, 39] propose to eliminate anchor priors and makes prediction directly.

Neural Architecture Search. NAS aims at finding an efficient network architecture for a task automatically. There are numerous works proposing different NAS methods for classification tasks [1, 44, 2, 42, 32] and detection tasks [7, 21, 8]. One recent paper [23] proposed to combine NAS with knowledge distillation by searching for the best student model given a fixed teacher model, which also proves the existence of structural knowledge in KD.

Knowledge Distillation. KD was first introduced in [14] and its effectiveness for classification task has been validated by extensive works [41, 12, 36, 30]. However, few works have proposed KD methods for object detection [5, 38, 33], which introduce only limited performance gain.

Pruning. Pruning methods have been well studied for classification tasks [26, 16, 25]. which focus on reducing the model complexity without much performance degradation. However, few works have verified its effectiveness on detection tasks.

3. Proposed Method

3.1. The Joint-DetNAS framework

3.1.1 Overview

As illustrated in Fig. 2, our Joint-DetNAS framework comprises two core processes: **student morphism** and **dynamic distillation**:

Student morphism aims to optimize the student's architecture while reduce the redundant parameters. However, integrating the two objectives is non-trivial: pruning requires pre-trained weights, which is incompatible with current NAS paradigm, since it is practically infeasible to obtain pre-trained weights that satisfy pruning requirements for all sampled architectures. To address this issue, we propose a carefully designed action space and a weight inheritance strategy, which enable the student to flexibly adjust its architecture while fully utilize the predecessor's weights.

Dynamic distillation targets at finding the optimal matching teacher to adapt to the student's structural

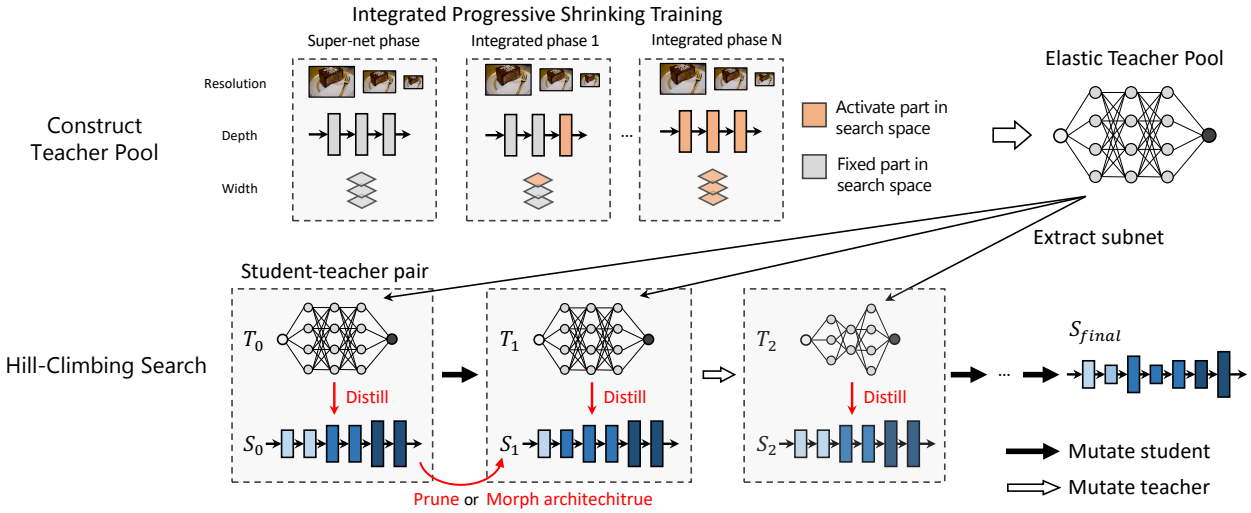


Figure 2: Illustration of our Joint-DetNAS. The algorithm consists of student morphism and dynamic KD, which interleave with each other. While student morphism optimizes the student's architecture, dynamic distillation aims to find the optimal matching teacher. An elastic teacher pool is trained via integrated progressive shrinking strategy, from which teacher detectors can be sampled without additional cost in subsequent searches; For student morphism, weight inheritance strategy is adopted, allowing the student to flexibly update its architecture while fully utilize the predecessor's weights.

changes, which calls for a way of obtaining sufficient powerful teachers with low cost. The mainstream NAS approach [13, 40, 35] using a proxy task (e.g., training with fewer epochs) to train the teacher does not guarantee the quality of teacher's supervision. On the other hand, training every teacher detector from scratch is too costly. Therefore, inspired by the recent work [3], we propose to construct an elastic teacher pool (ETP) containing sub-networks with high performances, which can be directly sampled as teachers to supervise the student. Empowered by the proposed ETP, teachers can be dynamically optimized according to the current status of the students with high efficiency.

A neat hill climbing algorithm is adopted to integrate the two processes, which enables adjusting the student's architecture and finding the matching teacher simultaneously. Due to the use of weight inheritance strategy and ETP, the search cost of our framework is significantly reduced.

3.1.2 Student Morphism

Our goal is to adjust an input detector's backbone and enable better adaptation to the given task. This is accomplished by continuously applying beneficial actions to the backbone while fully utilize the predecessor's parameters.

Action Space. An action space $\mathbb{A}$ containing pruning and network morphism is proposed to allow the student to flexibly adjust its architecture while fully utilize the predecessor's weights. **Pruning** removes the redundant parameters to make the model compact**,** which includes 2 actions: **(1) Layer Pruning** directly removes a whole layer with least importance, while **(2) Channel Pruning** removes the channels of convolutions which are insignificant. **Network morphism** flexibly adjusts the student's architecture, two

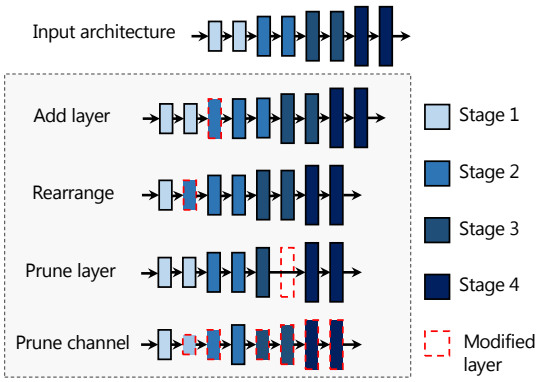


Figure 3: Illustration of the student action space $\mathbb{A}$. All actions are compatible with weight inheritance strategy.

actions are considered in this category: **(3) Add-Layer** inserts a new layer at a given position and introduces more capacity to enhance the model's performance; **(4) Rearrange** moves a layer from one stage to its neighbor stage, which enables flexibly re-allocating the backbone's computational budget for detection task. The proposed action space supports various stage-based backbone families for detector, e.g., ResNet, ResNeXt, and MobileNet series, etc.

Weight Inheritance. The trained weights of the predecessor are inherited to (1) provide the initial pre-trained weights for pruning, and (2) eliminate the expensive ImageNet pre-training prerequisites for faster evaluation. Specifically, we define the inheritance process as a function f_{evolve}: $f_{evolve}(S^{\theta}_{old}, a) \longrightarrow S^{\theta'}_{new}$, which accepts a detector S^{θ}_{old} and an action a as its inputs and outputs a new detector $S^{\theta'}_{new}$ with adjusted architecture and inherited parameters. The aforementioned action space is highly com-

patible with Weight Inheritance and the detail of f_{evolve} for each action is elaborated in the appendix.

Search with Dynamic Resolution. The resolution of input images play an important role in the performance and inference speed of detectors. Instead of directly incorporating input resolutions into the search process, which expands the search space considerably, we propose to train the student by dynamically sampling a resolution in each training iteration. Thus, multiple resolutions can be evaluated after training, which boosts the search efficiency.

3.1.3 Dynamic distillation with Elastic Teacher Pool

We are inspired by the recent work [3], in which a progressive shrinking strategy is proposed to train a super-network only once and obtain all the subnets with competitive performances. This approach fits our requirement for building a pool of teachers containing sufficient powerful detectors. However, the complexity of detection task initiates new challenges for this already complicated pipeline.

Subnet space. For a backbone with multiple stages, each subnet is determined by sampling the width and depth in a given range at each stage, while the combination of all the subnets form the subnet space. Other than the backbone, the FPN (neck) also plays an important role in detection, which fuses the features maps of different scales to obtain richer spatial information. Thus, we incorporate its widths variations in our implementation. To facilitate the search, we design our subnet space to cover architectures ranging from that of ResNet18 (1.0x width) to ResNet101 (1.5x width), which contains roughly 765000 networks (including different image resolutions) with competitive performances. More implementation details of the subnet space can be found in Section 4 and the Appendix.

Training with Integrated Progressive Shrinking (IPS). The training is divided into several phases: In the first phase, only the largest super-net is trained; In the following phases, subnets with shrunk depths and widths are gradually added into the subnet space, while the super-net acts as the teacher to distill all subnets using our KD method proposed in 3.2. In contrast to the progressive shrinking (PS) strategy proposed in [3], where the shrinkage of width and depth are performed sequentially, we propose an integrated progressive shrinking strategy (IPS) to jointly optimize smaller depths and widths, thus significantly reduces the training cost. More details can be found in the Appendix.

3.1.4 Search Algorithm

Different from mainstream NAS methods, our framework aims to upgrade a base detector S_{base} rather than exploring the whole search space. Comparing with sample-based search algorithms (e.g., RL [43, 34, 13], BO [31], etc.), the Hill Climbing (HL) approach efficiently evolves the student-teacher pairs and is highly compatible with weight inheritance strategy.

Algorithm 1 Hill Climbing Search of Joint-DetNAS

Input: base detector S^{θ}_{base}; student action space $\mathbb{A}$; resolution choices $R = \{r_i\}_{i=1,..,k}$; an elastic teacher pool P with P_{super} as the largest super-net.
top-k-list $\leftarrow \emptyset$; $\left\{S^{\theta}_{old}, T_{old}\right\} \leftarrow \left\{S^{\theta}_{base}, P_{super}\right\}$;
Start hill climbing
repeat
  $\left\{S^{\theta}_{old}, T_{old}\right\} \leftarrow$ sample from top-k list
  choice $\leftarrow$ select to evolve teacher or student
  if choice is student **then**
    $a \leftarrow$ sample from $\mathbb{A}$
    $S^{\theta^{'}}_{new} \leftarrow f_{evolve}\left(S^{\theta}_{old}, a\right); T_{new} \leftarrow T_{old}$
  else
    $T_{new} \leftarrow$ mutate T_{old}
    Extract T_{new} from P; $S^{\theta^{'}}_{new} \leftarrow S^{\theta}_{old}$
  end if
  Fast evaluate $\left\{S^{\theta^{'}}_{new}, T_{new}\right\}$ with all r_i
  $s \leftarrow \max_i \left(H\left(S^{\theta^{'}}_{new}, r_i\right)\right)$
  Update top-k list with $\left\{S^{\theta^{'}}_{new}, T_{new}, s\right\}$
until Convergence

During the search, we optimize the student and the teacher alternatively. Specifically, the algorithm starts with an initial student-teacher pair. During each iteration, either the student is updated by applying an action (as described in Section 3.1.2) or the teacher is mutated by modifying the depth or width in each backbone stage. Benefiting from the weight inheritance strategy, each student-teacher pair can be evaluated with only a few epochs of training. We use the following scoring metric to evaluate a student-teacher pair:

$$H\left(S, R\right) = mAP\left(S\right) \times \left[\left(\frac{C\left(S\right)}{C_{base}}\right) \times \left(\frac{R}{R_{base}}\right)^{\beta}\right]^{-\alpha}$$

where S is the student detector; C is the complexity metric, which we adopt $FLOPS$ since we do not target any particular device; R is the resolution of input image; C_{base} and R_{base} are the base complexity and base resolution; α is a coefficient that balances the performance and complexity trade-off; β balances the complexity introduced by the architecture and the input resolution.

The search procedure is illustrated in Algorithm 1. Our framework can be parallelized on multiple machines to boost the search efficiency.

3.2. Knowledge Distillation for Detection

Detection KD requires delicate design to distill spatial and localization information. Our detection KD method includes two components: a) **Feature-level distillation** maximizes the agreement between teacher and student's backbone features in interested areas; b) **Prediction-level distil-**

lation uses predictions outputs from teacher's heads as soft labels to train the students.

3.2.1 Feature-level Distillation

Feature maps encode important semantic information. However, imitating the whole feature maps is hindered by severe imbalance between the foreground instances and background regions. To this end, we only distill the features of object proposals, the objective can be formulated as:

$$L_{feat} = \frac{1}{N_p} \sum_{l=1}^{L} \sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{c=1}^{C} \left(f_{adap} \left(F_S^l \right)_{ijc} - \left(F_T^l \right)_{ijc} \right)^2$$

where F_S and F_T are features after ROI align; $f_{adap}(\cdot)$ is an adaptation function mapping F_S and F_T to the same dimension; N_p is the number of mask's positive points; L is number of FPN layers; W, H, C are feature dimensions.

3.2.2 Prediction-level Distillation

The prediction level KD loss can be expressed in terms of classification and regression KD loss: $L_{pred} = L_{cls} + L_{loc}$.

Uncertainty from Classification. Similar to classification, the student is optimized by soft cross entropy loss using teacher's logits as targets, which can be written as: $L_{cls} = -\frac{1}{N} \sum_i^N \mathbf{P}_t^i \log \mathbf{P}_s^i$, where N is the number of training data; $\mathbf{P}_t$ and $\mathbf{P}_s$ are predicted score vectors of the teacher and the student, respectively.

Uncertainty from Localization. Simply imitating the four coordinates from teacher's outputs provides limited information about how teacher localize objects, which motivates us to incorporate the class "uncertainty" knowledge into this process, i.e., utilizing prediction for all classes generated by the regression decoder. The class-aware localization outputs encode the teacher's ability of localizing proposals (can be viewed as a parts of objects) given different class hypotheses. Specifically, we calculate the sum of regression values weighted by classification scores: $L_{reg} = \frac{1}{N} \sum^N \mid \sum_{i=0}^{C} p_t^i \times \left(reg_t^i - reg_s^i \right) \mid$, where C is the number of classes; p_i and reg_i are the classification score and regression outputs of foreground class i; superscripts s and t stand for student and teacher.

3.3. Model Pruning

Pruning is incorporated into the framework as a part of student morphism to reduce student detector's complexity. We utilize both layer-wise and channel-wise pruning, which reduce the student's depth and width respectively. **Layer-wise Pruning** removes backbone's entire layer with the least L1-Norm. **Channel-wise Pruning** reduces the width of detector's backbone, for which we apply network slimming approach [25]. The method determines the channel importance according to the magnitude of BN's weights. Then the channels with least importance are removed. To encourage channel sparsity, we add a regularization loss to BN's weight parameters γ: $L_{BN} = \sum_{\gamma \in \Gamma} \mid \gamma \mid$. A small pruning percentage is set during each student morphism to progressively shrink the student without causing much performance deterioration.

Base model	Method	Input size	AP
R18-FPN	standard	$800 \times 600/1333 \times 800$	34.3/36.0
	our ETP	$800 \times 600/1333 \times 800$	$35.1^{+0.8}$/$36.1^{+0.1}$
R50-FPN	standard	$800 \times 600/1333 \times 800$	38.4/39.5
	our ETP	$800 \times 600/1333 \times 800$	$41.8^{+3.4}$/$42.4^{+2.9}$
R101-FPN	standard	$800 \times 600/1333 \times 800$	39.7/41.4
	our ETP	$800 \times 600/1333 \times 800$	$43.1^{+3.4}$/$44.1^{+2.7}$

Table 1: The subnets sampled from elastic teacher pool (ETP) consistently outperform their equivalent baselines trained under standard training strategy (2x+ms).

Model	Pruning percentage	Backbone FLOPS(G)	AP
R50-FPN	0% (baseline)	84.1	37.1
	10%/20%/30%	74.4/70.0/65.8	37.3/37.0/36.4
R101-FPN	0% (baseline)	160.2	39.0
	10%/20%/30%	140.9/125.7/112.5	39.2/38.8/38.3

Table 2: Pruning results for R50-FPN and R101-FPN given different channel pruning percentages. The detector's FLOPS can be effectively reduced without much performance degradation.

KD Method	Student	Teacher	AP
FGFI [38]	R50-half	R50	34.8
TAR [33]	R50	R152+R101	40.1
our KD	R18 (36.0)	R50 (39.5)	**38.1**$^{+2.1}$
	R50 (39.5)	R101 (41.4)	**41.6**$^{+2.1}$

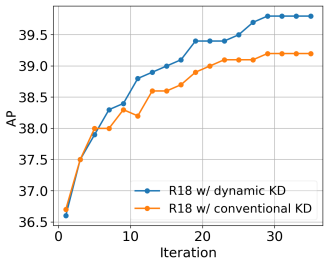
Table 3: Comparison between our detection KD method with baselines and previous KD works. The values in parentheses are the baseline AP and teacher's AP, respectively. Our KD method outperforms others by a large margin.

Overall Loss for Training Students The total loss for training student detectors can be represented as: $L = L_{det} + L_{feat} + L_{pred} + \lambda L_{BN}$, where L_{det} denotes the normal detection training loss; λ is the coefficient of the regularization loss for pruning, which is set to 0.00001. L is enforced on the student throughout the search process.

4. Experiments

Datasets and evaluation metrics. We use MS COCO [20] to conduct experiments. The mAP for IoU thresholds from 0.5 to 0.95 is used as the performance metric.

Implementation details. We use ResNet-based detectors to construct our elastic teacher pool, the subnet space for the backbone contains depth ranging from [2,2,2,2] to [3,4,23,3] for four stages, and the width for each backbone stage and the neck can be sampled from [W, $1.25 \times W$,



Group	Teacher	AP
Conventional KD	fixed	39.2
Dynamic KD	searchable	**39.8**

Figure 4: The comparison between dynamic KD and convention KD (with the super-net of ETP as the teacher). Dynamic KD can boost the student faster and help it reach a higher final performance.

$1.5W$], where W is the width of standard ResNet. During search, each student-teacher pair is trained with 3 epochs for fast evaluation. For each teacher subnet sampled from the teacher pool, we reset its BN statistics by forwarding a batch of images, which is essential for performance recovery. More details are in the Appendix.

4.1. Ablation Study

4.1.1 Decoupling the Framework

Each component plays an important role in the overall Joint-DetNAS framework. Thus, it is essential to decouple them from the framework and separately analyze their effectiveness in detail.

Quality of Elastic Teacher Pool. Our framework requires the teacher detectors sampled from the ETP to have competitive performances. To demonstrate the quality of our ETP, we compare its sampled subnets with their equivalent classic FPN detectors trained under standard 2x schedule and multi-scale training (for easier notation, we denote this as 2x+ms in later sections) strategy in Table 1. The former consistently outperforms the latter.

Pruning. We conduct experiments to prune the backbone of R50-FPN and R101-FPN detectors given different channel pruning percentages in Table 2. The detectors are pre-trained for 12 epochs before pruning and fine-tuned for extra 3 epochs afterwards. The detector's parameter can be effectively reduced without much performance degradation. e.g., For both detectors, the performance after pruning 30% channels is still comparable to the original.

Distillation. Our detection KD framework is simple yet effective. We compare our detection KD method with baselines and previous KD works in Table 3. The R18-FPN and R50-FPN detectors are adopted as the students, with R50-FPN and R101-FPN as the teachers, respectively. To demonstrate effectiveness of our KD method, stronger baselines (2x+ms) are used. The results show that our method outperform the others by a large margin.

Base model	Group	Input size	FLOPS (G)	FPS	AP
R18-FPN	baseline	1333×800	160.5	28.2	36.0
	ours	1080×720	**117.3**$^{-27\%}$	**33.0**$^{+17\%}$	**38.5**$^{+2.5}$
R50-FPN	baseline	1333×800	215.8	20.5	39.5
	ours	1080×720	**145.7**$^{-32\%}$	**25.4**$^{+24\%}$	**42.3**$^{+2.8}$
R101-FPN	baseline	1333×800	295.7	15.9	41.4
	ours	1080×720	**153.9**$^{-48\%}$	**23.3**$^{+47\%}$	**43.9**$^{+2.5}$
X101-FPN	baseline	1333×800	286.9	13.2	42.9
	ours	1333×800	**266.3**$^{-7\%}$	**14.0**$^{+6\%}$	**45.7**$^{+2.8}$

Table 4: Our Joint-DetNAS can upgrade detectors with various backbone designs. Joint-DetNAS consistently boosts the input baseline detectors' performances as well as substantially reduces their complexities.

4.1.2 Dynamic KD Benefits the Student

We aim to verify the superiority of dynamic KD: whether dynamic teacher is better than a fixed powerful teacher for transferring knowledge. Specifically, we fix the ResNet18-FPN detector as the student and follow the 3-epoch iterative training schedule, then conduct two experiments (1) dynamic KD (DKD): the teacher is dynamically sampled in every iteration and (2) Conventional KD (CKD): the largest super-net in ETP is used as the teacher. The results in Figure 4 shows that DKD can boost the student faster and help it reach a higher final performance. This also implies the underlying structural knowledge in KD, for which we provide further analysis in later Section 4.3.1.

4.2. Main Results

4.2.1 Comparison with Baselines

Joint-DetNAS can upgrade detectors with various backbone designs. We conduct experiments on FPN detectors with R18, R50, R101 and X101 as backbones to verify the effectiveness of our framework. We use 1333×800 resolution with 2x+ms training for baseline and compare with our result using searched resolution. As shown in Table 4, our method consistently boosts the detectors' performances while substantially reduces their complexities. Notably, for R101-FPN, the upgraded detector achieves $+2.5$ gain in AP and 47% reduction in latency.

4.2.2 Joint Optimization Beats Naive Pipelining

Intuitively, NAS, pruning and KD is can be combined by pipelining: first search a detector with NAS, then prune it and train it with KD. We compare our joint optimization approach with pipelining methods: (1) Start with regular R101-FPN detector or a NAS-searched detector with lower complexity; (2) pre-train them with the pruning regularization loss; (3) prune the detector to comparable complexity with the result of Joint-DetNAS (R101-based); (4) train the pruned detector with the proposed KD under standard train-

Method	Intermediate (pre-training)		Final (w/ prune+KD)	
	backbone FLOPS (G)	AP	backbone FLOPS (G)	AP
R101-prune-KD	122.6	39.0	$60.8^{-50\%}$	$42.1^{+3.1}$
NAS-prune-KD	105.2	39.9	$59.1^{-44\%}$	$41.8^{+1.9}$
Joint-DetNAS (R101)	-	-	**58.9**	**43.9**

Table 5: Comparison between Joint-DetNAS and the naive pipelining approach. The results show that the pipelining methods leads to suboptimal, while Joint-DetNAS is capable of better integrating NAS, pruning and KD.

ing strategy (2x+ms) and the same resolution (1080×720). In Table 4, we compare the result of NAS-prune-KD and R101-prune-KD and find that the performance gain brought by NAS diminishes after pruning and KD are applied, indicating that the naive pipelining strategy leads to suboptimal. In contrast, our joint optimization methods outperforms both pipelining methods by a large margin.

4.2.3 Comparison with State-of-the-art

We compare our method with the SOTA manually designed detectors (e.g., FCOS[37], RepPoints[39] and CB-Net[24], etc.) and NAS-based (e.g., NAS-FPN[13], SP-NAS[15], etc.) approaches. The results of the COCO's `test-dev` split are reported in Table 6. Our Joint-DetNAS outperforms SOTA manually designed detectors in terms of both FPS and AP, e.g. our searched detector based on R101 reaches 23.3 FPS and 43.9 AP, outperforming RepPoint-R101's 13.7 FPS and 41.0 AP by a large margin. Furthermore, our method (R101-based) surpasses most mainstream detection NAS methods (e.g., SM-NAS [40] and NAS-FPN [13]) and reaches comparable performance with the SOTA EfficientDet (D2) [35], while requiring much less search cost and no extra post-search training epochs.

4.2.4 Search Efficiency

Search efficiency is a key issue in NAS. We compare Joint-DetNAS with other SOTA detection NAS methods (e.g., [8, 40]) in Table 7. Our framework finds better performance-complexity tradeoff for the detector with less search cost,

4.3. Looking into the Search Results: More analysis

4.3.1 Teacher-student Relationship

As observed in earlier Section 4.1.2, larger detectors may not be better teachers, which naturally prompts us to further explore the matching pattern of promising teachers for different students. To this end, we apply dynamic KD to search optimal matching teachers for students with various complexities (i.e., FPN with R18, R50, R101 as the backbones). As shown in Figure 5, starting with the same teacher, each student can converge to different teachers. The results present a clear pattern: smaller students tend to match teachers with lower capacities, and vice versa. This phenomenon implies the underlying interdependence of complexity between the student-teacher pairs, which can provide useful insights for designing detection KD system.

[1]The FPS of EfficientDet's Pytorch implementation `https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch` is reported for fair comparison.

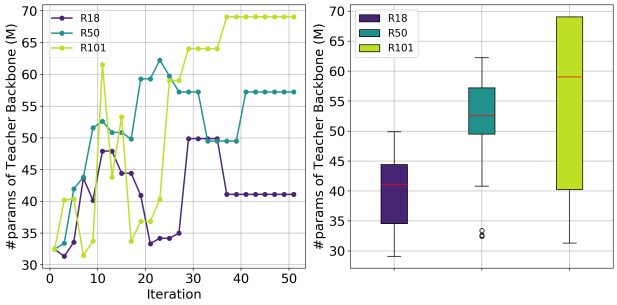


Figure 5: The teacher-student capacity-matching pattern during search. **Left**: the teacher's backbone parameters of the best student in the current iteration. **Right**: distribution of teacher's backbone parameters throughout the search. The matching teacher's complexity is highly correlated with that of the student.

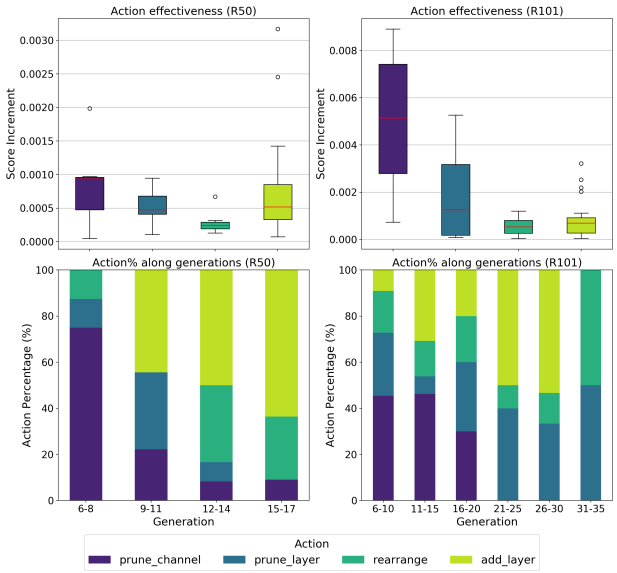


Figure 6: **Top**: the overall score increment brought by each action throughout the search; **Bottom**: percentage of beneficial actions throughout generations. Channel pruning is mostly adopted during early phases, and contributes most score increment. Other actions brings more fine-grained adjustments and occur mainly in later phases.

Method	Backbone	Input size	Post-search training epochs	FPS	AP	$AP_{@.5}$	$AP_{@.7}$	AP_S	AP_M	AP_L
Manually Designed										
Cascade RCNN [4]	R101	1333 × 800	-	13.5 (V100)†	43.6	62.1	47.4	26.1	47.0	53.6
FCOS [37]	R101	1333 × 800	-	17.3 (V100)†	41.5	60.7	45.0	24.4	44.8	51.6
RepPoints [39]	R101	1333 × 800	-	13.7 (V100)†	41.0	62.9	44.3	23.6	44.1	51.7
CB-Net w/ Cascade [24]	R101-TB	1333 × 800	-	5.5 (V100)‡	44.9	63.9	48.9	-	-	-
NAS-Based										
Det-NAS [8]	DetNASNet	1333 × 800	24	20.4 (V100)‡	40.2	61.5	43.6	23.3	42.5	53.8
SM-NAS (E5) [40]	SMNet (searched)	1333 × 800	24	9.3 (V100)‡	45.9	64.6	49.6	27.1	49.0	58.0
SP-NAS [15]	SPNet-XB	1333 × 800	24	5.6 (V100)‡	47.4	65.7	51.9	29.6	51.0	60.4
NAS-FPN (7@384) [13]	AmoebaNet	1280 × 1280	150	3.6 (P100)‡	48.0	-	-	-	-	-
EfficientDet (D2)* [35]	EfficientNet (B2)	768 × 768	300	26.8 (V100)† [1]	43.9	62.7	47.6	-	-	-
Joint-DetNAS (R50)	R50-searched	1080 × 720	-	**25.4** (V100)†	**42.3**	**62.6**	**46.2**	**26.2**	**45.1**	**50.6**
Joint-DetNAS (R101)	R101-searched	1080 × 720	-	**23.3** (V100)†	**43.9**	**63.8**	**47.9**	**27.0**	**46.8**	**52.8**
Joint-DetNAS (X101-Cascade)	X101-searched-DCN	1333 × 800	16	**10.1** (V100)†	**50.7**	**69.6**	**55.4**	**31.3**	**53.8**	**64.0**

Table 6: Comparison with SOTA manually designed and NAS-based methods. We obtain the X101-Cascade model by upgrading the searched X101-based detector with DCN and Cascade head, and further fine-tune it for 16 epochs with HTC [6] teacher. FPS is reported with batch size 1; † and ‡ represent the results obtained on our own V100 device and from the original paper, respectively. * means soft-NMS is adopted.

Search Method	FLOPS	AP	#Searched architectures	Search cost (GPU days)
random	-	-	50	~1200
Det-NAS [8]	289.4	40.0	1000	70
NAS-FPN (R50-7@256) [13]	281.3	39.9	10000	>>500
SP-NAS [15]	349.3	41.7	200	200
Joint-DetNAS (R101-based)	**145.7**	**43.9**	100	200

Table 7: Comparison with other search methods. The search cost consists of 3 parts: (1) pre-training cost (including ImageNet pre-training or ETP training), NAS cost and post-training cost. We only estimate the cost for random search as it is prohibitively expensive (ImageNet pre-training for each sampled detector).

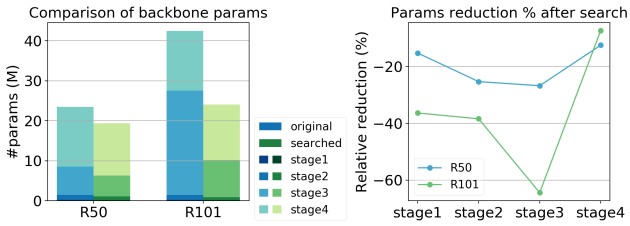


Figure 7: The computation allocation of detector's backbone before and after the search. For detectors with classic ResNet-based backbones, the computation reduction is mostly allocated at stage 3, followed by stage 2 and stage 1.

4.3.2 How Students Evolve: Action Analysis

We study how the student evolves along the search process by analyzing the actions improving the score function H taken throughout the generations (generation increases when student's performances is boosted) for our R50- and R101-based search. Figure 6 shows the shift of focus in balancing the performance-complexity tradeoff. We can see that channel pruning contributes the most score increment. In early phases, channel pruning occurs more often to adjust the network as a whole; while in later phases, Add-layer, Prune-layer and Rearrange follow to adjust the computation allocation at each stage in a fine-grained manner.

4.3.3 Computation Allocation for Detector Backbone

In Figure 7, we show the backbone's computation allocation of our R50- and R101-based detectors before and after the search. The computation at stage 3 is reduced most dramatically, followed by stage 2 and stage 1. This implies the redundancy distribution in manually designed ResNet models, which provides the community with some prior knowledge for detector's backbone design.

5. Conclusion

This paper present a new way of jointly optimizing NAS, pruning and KD to boost the performance and reduce the complexity of object detectors. Extensive experiments are conducted to show the superior performance of our proposed Joint-DetNAS framework. We believe our method has the potential to be extended to tasks other than object detection.

References

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using rein-

forcement learning. *arXiv preprint arXiv:1611.02167*, 2016. 2

[2] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *AAAI*, 2018. 2

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 3, 4

[4] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018. 8

[5] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*, pages 742–751, 2017. 1, 2

[6] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4974–4983, 2019. 8

[7] Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *NIPS*, 2018. 2

[8] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Chunhong Pan, and Jian Sun. Detnas: Neural architecture search on object detection. *arXiv preprint arXiv:1903.10979*, 2019. 1, 2, 7, 8

[9] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4794–4802, 2019. 1, 2

[10] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6569–6578, 2019. 2

[11] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 1

[12] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018. 2

[13] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019. 1, 3, 4, 7, 8

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 2

[15] Chenhan Jiang, Hang Xu, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Sp-nas: Serial-to-parallel backbone search for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11863–11872, 2020. 1, 7, 8

[16] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 1, 2

[17] Quanquan Li, Shengying Jin, and Junjie Yan. Mimicking very efficient network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1

[18] Xin Li, Yiming Zhou, Zheng Pan, and Jiashi Feng. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *CVPR*, pages 9145–9153, 2019. 1

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 5

[21] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. *arXiv preprint arXiv:1901.02985*, 2019. 2

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2

[23] Yu Liu, Xuhui Jia, Mingxing Tan, Raviteja Vemulapalli, Yukun Zhu, Bradley Green, and Xiaogang Wang. Search to distill: Pearls are everywhere but not the eyes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7539–7548, 2020. 2

[24] Yudong Liu, Yongtao Wang, Siwei Wang, TingTing Liang, Qijie Zhao, Zhi Tang, and Haibin Ling. Cbnet: A novel composite backbone network architecture for object detection. 7, 8

[25] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017. 2, 5

[26] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 1, 2

[27] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *CVPR*, 2017. 2

[28] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[30] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 2

[31] Han Shi, Renjie Pi, Hang Xu, Zhenguo Li, James T Kwok, and Tong Zhang. Bridging the gap between sample-based and one-shot neural architecture search with bonas. *arXiv preprint arXiv:1911.09336*, 2019. 4

[32] Han Shi, Renjie Pi, Hang Xu, Zhenguo Li, James T Kwok, and Tong Zhang. Multi-objective neural architecture search via predictive network performance optimization. *arXiv preprint arXiv:1911.09336*, 2019. 2

[33] Ruoyu Sun, Fuhui Tang, Xiaopeng Zhang, Hongkai Xiong, and Qi Tian. Distilling object detectors with task adaptive regularization, 2020. 2, 5

[34] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*, 2018. 4

[35] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 3, 7, 8

[36] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017. 2

[37] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 9627–9636, 2019. 2, 7, 8

[38] Tao Wang, Li Yuan, Xiaopeng Zhang, and Jiashi Feng. Distilling object detectors with fine-grained feature imitation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4933–4942, 2019. 1, 2, 5

[39] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9657–9666, 2019. 2, 7, 8

[40] Lewei Yao, Hang Xu, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Sm-nas: Structural-to-modular neural architecture search for object detection. *arXiv preprint arXiv:1911.09929*, 2019. 1, 3, 7, 8

[41] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016. 2

[42] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *CVPR*, 2018. 2

[43] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 4

[44] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 2