We thank all reviewers for your valuable comments and positive recognition of sound motivation, interesting solution, strong results, and clear writing. We respond to the concerns point-by-point as below.

——— To Reviewer #1 ———

Q1.1: Why distilling prioritized paths improves architecture rating? A1.1: Similar to the discussion in DNA [6], distilling prioritized paths can boost the training of subnets and alleviate the insufficient training issue in weight-sharing regime. The more sufficient/full training of subnets leads to a more accurate architecture rating [6](Sec.4.3).

Q1.2: The set used to train the matching network? A1.2: The data to train the meta matching network $\mathcal{M}$ is randomly sampled from the training set, containing 20k images. For more details, please refer to the provided code (file: *supernet_function.py*, line: 232). We will revise the manuscript to make this point clearer. Thanks for helpful comments.

Q1.3: AutoAugment (AA). A1.3: For a fair comparison, we retrain DNA-d with AA and it obtains up to 0.3% improvements w.r.t. top-1 acc. on ImageNet. In the revision, we will clarify that DNA does not utilize AA for retraining.

——— To Reviewer #2 ———

Q2.1: Discussion with FasterSeg. A2.1: Thanks for reminding and we will add the following discussion into Sec. 5. There are two fundamental differences. 1) FasterSeg searches the teacher model by architecture parameters, while our method through meta network, allowing each subnet to select its matched teacher model. 2) FasterSeg is a gradient-based differentiable method, while ours is a sampling-based one-shot method.

Q2.2: Memory and random search. A2.2: 1) Similar to the previous works [5,6,7], utilizing KD for NAS will increase GPU memory cost. However, KD is capable of speeding up supernet training while improving architecture ranking. 2) The 3-time random searches of ~450 Flops model gets 75.1±0.7% top-1 accuracy, being 4.1% inferior to our method.

Q2.3: About meta network. A2.3: 1) We tested three types of inputs to the meta network, i.e., the difference, addition, and concatenation of feature logits. The corresponding top-1 acc. are 79.2/78.9/79.0, respectively. Considering complexity-accuracy tradeoff, we choose the first option. 2) To keep simple and efficient, the meta network only contains one fully-connected layer with 1,000 hidden nodes. We will revise the manuscript accordingly. Thanks!

——— To Reviewer #3 ———

Q3.1: Search space. A3.1: The search spaces of most current one-shot methods are inconsistent, including the publications from Google. For example, EfficientNet sets se_ratio=0.25, expansion_rate=6, max_depth={5-22}, while MobileNetV3 are {0,0.25}, {3,4,6} and 4, respectively. Our search space is closely similar to the ones in DNA [6] and OFA [5], the only difference is expansion_rate: ours is {4,6}, while DNA is {3,6} and OFA is {3,4,6}. Our method surpasses DNA-c and DNA-d by 1.4 and 1.6 points w.r.t. top-1 acc. on ImageNet. Our method performs comparably to the finetuned OFA (sometimes slightly better, e.g., 200-300M Flops), but 4.4× faster than OFA, i.e., 12 v.s. 53 GPU days. The ablation studies in Tab. 1 (the same search space) verify the efficacy of our method. #Params are provided as below.

#	1	2	3	4	5	6
Params	7.0	6.5	6.9	6.8	7.1	7.2

	@30	@20	@10
τ	0.370	0.381	0.389

Method	mAP	FLops
EfficientDet-D0	34.6	390
Hit-Detector	36.9	1,839
Ours-S	33.2	285
Ours-M	36.8	470

Q3.2: Iterations for updating $\mathcal{M}$ and Kendall rank τ. A3.2: 1) Updating $\mathcal{M}$ need to calculate second-order gradient, which is 7.8× slower than first-order gradient (*i.e.*, 0.5 v.s. 3.9 s/iter). The total iterations are 1.2M/1K/200*120≃720. 2) We provide the additional results of τ@10 and τ@20 in the right table. We can observe that τ@10 is slightly superior.

Q3.3: Architecture selection. A3.3: The architectures in Tab. 5 are searched by multiple runs. In each search, users are required to input the desired *min* and *max* Flops (see Alg. 1). The prioritized paths in $\mathbb{B}$ are forced to satisfy the Flops constraint, i.e., $Flops \in [min, max]$. Evolution algorithms, such as NSGA-III in CARS, are not adopted in our method, thus there is no small model trap issue. Eq. (3) follows Occam's razor principle during model selection.

Q3.4: EfficientNet-B0/Det and Hit-Detector. A3.4: 1) EfficientNet adopted AutoAugment and EMA in the original paper (Sec. 5.2) and released code (Line 30). For a fair comparison, we retrain EfficientNet-B0 using our setting and obtain 76.2 top-1 acc., which is comparable to 76.3 reported in ICML19 paper [9] while inferior to 77.3 reported in the latest ArXiv version. 2) The comparisons with EfficientDet-D0 and Hi-Detector are reported in the upper-right table and will be updated into revision. 3) The reason why KD improves ranking is discussed in *Q1.1*. Your guess provides a logical and insightful understanding of this work. We will revise the manuscript in light of these helpful comments.

——— To Reviewer #4 ———

Q4.1: Novelty. A4.1: This work is a new upgrade of the single-path one-shot method, pushing architecture performance into new frontiers. The key novelty lies in two aspects: 1) We introduce prioritized paths to remove the the third-party teacher models from supernet training. The matching between prioritized paths and subnets is meta-learned; 2) Different from previous methods using evolution search [5,8] to find promising architectures, our method can directly select the best performing ones from the prioritized paths, which is much more efficient (30× faster than SPOS [8] and OFA [5]).

Q4.2: Why the weights trained to optimize one path can be reused for others? A4.2: Weight sharing across paths are originally proposed to amortize training cost [13,14]. Feature reuse is the predominant reason for efficient learning in one-shot NAS. Though now feature reuse lacks clear theoretical guarantee [15,38], it allows ranking architectures, which would be sufficient if the estimated performance correlates strongly with the actual performance.

Q4.3: About broader impact. A4.3: Our work does not have immediate societal impact, since the algorithm is only designed for image classification, but it can indirectly impact society. As an example, our work may inspire the creation of new algorithms and applications with direct societal implications. We will add more discussions. Thanks!