To Reviewer 1 (R1): Thanks for the comments and we will reorganize the paper according to your suggestions. R1 may have some misunderstandings and we will clarify it first. R1 may think NAT as a NAS method. Actually, the problem for NAT (*i.e.*, architecture optimization defined in Sec. 3.1) is different from what is addressed in NAS methods (*e.g.*, ENAS and NAO) and can be regarded as "a post-processing step" (recognized by R2).

Q1. Limitation of large space in NAO: Although NAO can do a similar job as NAT, NAO and NAT focus on different problems. The search space of NAO is designed for NAS and might be unnecessarily large for architecture optimization, leading to unstable results (*e.g.*, lower accuracy on ENAS and much higher cost on VGG in Tables 1 and 2). Instead, the search space of NAT is relatively smaller and suitable for this problem, making the model easier to train for this task.

Q2. How to get skip connections in VGG? To build a general representation for both hand-crafted and NAS based architectures, we add null connections into VGG to ensure that each node has two input nodes (defined in Line 120). Then, NAT can add skip connections into VGG by replacing the null connections (see more discussions in Section 4.5).

Q3. Why the generated networks have two inputs "-2" and "-1": For fair comparison, we follow the same and also a common setting in [21, 22, 24, 26] to build the cells with two input nodes, denoted by "-2" and "-1". Here, "-2" and "-1" represent the outputs of the second nearest and the most nearest cell in front of the current one, respectively.

Q4. Reasonable to find best expected operation setting in Eqn. (2): Thanks for the eagle eye to spot the typo in Eqn. (2). The order of the maximum and expectation operators should be reversed, *i.e.*, $\mathbb{E}_{\varpi\sim q(\varpi)} \max_{\alpha,w} R\left(\alpha, w|\varpi\right)$, which means that NAT learns a general optimizer by optimizing the expected reward of the optimal architecture for any ϖ. Actually, this objective is used in our code and works well in practice. We will release the code upon publishment.

Q5. More details about solving three challenges: The first two challenges indicate the difficulties in solving the constraint $c(\alpha) \leq \kappa$ in Eq.(2). To address these, we convert the constrained problem into an optimization problem that improves accuracy without introducing extra cost. To achieve this goal, we restrict the replacement of operations under the rule $c(O) > c(S) > c(N)$ (*i.e.*, the transition problem in Fig. 2(c)). For the third challenge regarding the expectation of reward, we estimate the expectation value by sampling architectures from $q(\varpi)$ (see Algorithm 1).

Q6. Training cost of NAT: The training of NAT takes 0.17 GPU days (also reported in Fig. 8 of supplementary).

Q7. Effect of small space of NAT: Due to the problem differences between NAT and NAS, it is unfair to directly compare the effect of different search spaces. For NAO, the large search space designed for NAS is unnecessary for architecture optimization (see Q1). With a smaller space, NAT consistently outperforms NAO in Tables 1, 2, and A.

Q8. Details about GCN: We have provided the details about GCN in Sec. B of supplementary and will make it clearer.

To Reviewer 2 (R2):

Q9. Comparison with random search & Results on resource-efficient cells: From Table A, NAT consistently outperforms the random search baseline and obtains promising results on MobileNetV2 and MnasNet.

Q10. Differences from (Cao *et al.*, ICLR, 2019): The differences lie in 1) the tasks: NAT focuses on architecture optimization for better accuracy while the ICLR paper focuses on architecture compression for higher compactness; and 2) model generalization ability: NAT learns a general optimizer for any arbitrary architecture while the ICLR paper has to learn a compression model for each given pre-trained model. We will cite and discuss this work in the paper.

Table A: More results on CIFAR-10. #P and #F denotes the number of parameters and FLOPs.

Method	DARTS + Cutout		MobileNetV2		MnasNet + Cutout		NAONet-WS + Cutout	
	Acc. (%)	#P / #F (M)	Acc. (%)	#P / #F (M)	Acc. (%)	#P / #F (M)	Acc. (%)	#P / #F (M)
Baseline	97.06	3.3 / 533	94.47	2.3 / 91	95.67	3.1 / 169	96.47	2.5 / 352
Random Search	95.17 $\pm$ 0.24	2.8 / 447	93.24 $\pm$ 0.23	2.0 / 70	93.71 $\pm$ 0.18	2.7 / 134	95.01 $\pm$ 0.21	2.1 / 303
NAO [24]	97.10 $\pm$ 0.06	3.6 / 583	95.05 $\pm$ 0.09	2.5 / 134	95.49 $\pm$ 0.08	3.3 / 195	96.55 $\pm$ 0.09	2.8 / 393
NAT	**97.30 $\pm$ 0.07**	3.0 / 477	**95.37 $\pm$ 0.08**	2.2 / 85	**96.13 $\pm$ 0.06**	3.0 / 152	**96.95 $\pm$ 0.07**	2.2 / 311

To Reviewer 3 (R3):

Q11. Specific form of $\mathcal{L}(\varpi, w)$: We use cross-entropy (CE) to compute $\mathcal{L}(\varpi, w) = \frac{1}{N}\sum_{i=1}^{N} \mathrm{CE}(f_{\varpi}(x_i; w), y_i)$, where $f_{\varpi}(x_i; w)$ is the prediction of x_i based the architecture ϖ parameterized by w, y_i denotes the ground-truth label.

Q12. How to apply "$s.t.\ c(\alpha) \leq \kappa$" in Eqn. (2) in optimization: Essentially, the constraint moves from the optimization process into the design of search space. Please refer to **Q5** of **R1** for more details.

Q13. Will each computation architecture in the network be pruned in the same way? For all the considered models, we use the same pruned/optimized cell to replace all the original cells to build the network.

Q14. Search space & improvement compared to NAO: Since NAT and NAO focus on different problems, *i.e.*, architecture optimization vs. architecture search, NAO exploits a NAS search space but NAT considers simple replacements to optimize architectures. In practice, NAT yields better results than NAO in both accuracy and complexity (see Table A). Instead, NAO may yield lower accuracy (*e.g.*, ENAS) and higher cost (*e.g.*, VGG) in Tables 1 and 2.

Q15. Performance on strong baselines: Due to the problem differences mentioned in Q14, NAT and NAS cannot be replaced by each other. From Tables A, 1 and 2, NAT consistently improve the performance of different architectures. Although the improvement on ENAS is not that significant (but significant accuracy improvement on DARTS on ImageNet (1.3% in Table 2)), NAT has its own novelty and plays an important role in architecture optimization.

Q16. Can NAT further boost NAONet? Based on NAONet, NAT is still able to obtain better architectures with higher accuracy and lower cost (see Table A). We will combine Table A and Table 3 in the paper.

Q17. Usage of Cutout: For all the considered architectures, we follow the same settings of the original papers. In the experiments, we only apply Cutout to NAS based architectures on CIFAR-10. We will include these details in the paper.

Q18. Missed related work. We will cite and discuss this work (So *et al.*, ICML, 2019) in the final paper.