Answers to Reviewer #1.

Q1: Comparison to OpenTuner.

A: We thank the reviewer for pointing out the OpenTuner project. OpenTuner identifies the importance of having domain-specific search techniques in auto-tuning. It was built to adaptively choose among domain-specific search techniques for general program tuning. In contrast, our focus is on NN model compilation rather than generic program tuning. Even though OpenTuner may be regarded as the state-of-the-art for other problems, AutoTVM is more widely accepted as the state-of-the-art in NN model compilation. Our technique is designed and validated specifically for the NN model compilation problem, and we are not claiming we advance the generic problem of program auto-tuning. Furthermore, our technical contribution is beyond the specific search strategies employed during auto-tuning, such as those used in OpenTuner. There can be multiple paths to advance the state-of-the-art of NN model compilation, and our work focuses on one of them. Finally, we find that OpenTuner is not directly applicable to our problem without non-trivial modifications. OpenTuner does not support optimizing PyTorch or TensorFlow models out-of-the-box and requires the user to manually create a configuration manipulator for the target program before optimization. However, it is non-trivial to create such a configuration manipulator for neural network components as their implementation details (e.g., tiling decisions) are often hidden away from end users. Even if we can create such a configuration manipulator, doing so for all tunable operators in a neural network seems time-consuming and tedious because there can be tens or hundreds of them. In contrast, AdaTune is closely coupled with AutoTVM, which automatically generates tunable code templates for neural network components. We will add a citation describing OpenTuner.

Q2: Evaluation on larger NNs on Transformer and ResNet-50.

A: We added additional experiments on Transformer and ResNet-50 in the appendix (also shown results below). AdaTune is 1.4-2.2X faster in optimization time while achieving comparable and sometimes better inference time than the baseline. We also observe that larger models do not necessarily indicate longer optimization time. Take the Transformer as an example. Since each layer of the model has the same model structure, AdaTune only needs to optimize it once and apply the same optimization strategy across all Transformer layers to reduce its latency.

	AutoTVM	AdaTune	Speedup
Resnet-18	22.6h	9.6h	2.4X
Resnet-50	20.0h	14.1h	1.4X
VGG-16	21.9h	16.7h	1.3X
SqueezenetV1	7.6h	5.8h	1.3X
Transformer (Enc.)	3.8h	2.8h	1.4X

Table 1: Optimization time on GPU.

	AutoTVM	AdaTune	Speedup
Resnet-18	2.0h	1.0h	2.0X
Resnet-50	3.6h	1.7h	2.1X
VGG-16	18.9h	6.5h	2.9X
SqueezenetV1	1.2h	0.7h	1.7X
Transformer (Enc.)	8.4h	3.8h	2.2X

Table 2: Optimization time on CPU.

	TVM	AutoTVM	AdaTune
Resnet-18	1.53ms	1.38ms	1.38ms
Resnet-50	4.82ms	4.37ms	4.37ms
VGG-16	3.95ms	3.86ms	3.86ms
SqueezenetV1	2.93ms	0.65ms	0.63ms
Transformer (Enc.)	78.15ms	52.25ms	47.46ms

Table 3: Inference time comparison on GPU.

	TVM	AutoTVM	AdaTune
Resnet-18	79.24ms	52.64ms	52.64ms
Resnet-50	217.12ms	115.76ms	115.68ms
VGG-16	884.94ms	442.01ms	438.68ms
SqueezenetV1	14.41ms	11.36ms	11.25ms
Transformer (Enc.)	2897.27ms	1620.88ms	1607.67ms

Table 4: Inference time comparison on CPU.

Q3: "It is not clear from the text how EI is used to select a promising plan. Is it the entire fitness function or part of it?"

A: We use the same diversity-aware function as the one used in AutoTVM, which is the addition of two terms: one is for the runtime cost estimate and the other considers the diversity when selecting candidates. We replace the run time cost part with EI and keep the second term unchanged to have a fair comparison.

Q4: "Figure 11 and 12, shouldn't the orange line climb up to the peak GFLOP plan faster?"

A: Thanks for pointing out. Since orange and red are similar colors, we accidentally switched the RFEI+CSA+AE and RFEI+CSA+DE label when generating the shared legend as a separate picture. We will use more distinguishable colors.

Answers to Reviewer #2.

Q1: "Whether Fig.8 is on GPU or CPU. I would like to see all these comparisons on both CPU and GPU."

A: Fig.8's result is on GPU. In the appendix (Table 1-4 and Figure 14-17), we included optimization time and inference time on both CPU and GPU. For the hardware measurement vs. space searching cost ratio on GPU (ResNet-18), it is 28% : 72% (22.6h) for the baseline and 44% : 56% (9.6h) for AdaTune. AdaTune reduces the hardware measurement time by 1.5X and reduces the space searching cost by 3X. Together it speedups the optimization by 2.4X.

Answers to Reviewer #3.

We thank the reviewer for the positive feedback and for highlighting the significance of our work.