Reviewer 1: *1. Computation analysis*. Thanks for this useful suggestion! The proposed HPM computes the hypergradient with hypernetworks following STN [8], which adds a linear transformation between hyperparameters and model parameters layer-wisely. Thus, the additional computing cost is comparable to the original one and the whole model could be also efficiently trained by feed-forward and backpropagation operations. On another hand, we train the proposed teacher network (*i.e.*, a small attention network) by freezing the student model, leading to a computational cost less than one hypergradient descent step. A more detailed computational analysis will be added in the final version.

2. Performance variance & Reference. We ran HPM on CIFAR-10 three times and obtained validation/test loss as 1) 0.5598/0.5664; 2) 0.5606/0.5640; and 3) 0.5647/0.5704, showing our result is relatively stable. This is consistent with the observations on synthetic functions. We also thank the reviewer for pointing out the valuable reference. Due to the limited time and space here, we will report performance variance and discuss the provided reference in the final version.

Reviewer 2: *1. Experimental setting*. Thanks for the useful suggestion! In this work, we mainly follow the experimental setting in [8] for a fair comparison to other HPO methods. The validation/test accuracy (%) of PBT [7], STN [8] and HPM on CIFAR-10 are 78.5/78.1, 80.3/80.1, and 81.7/81.1, respectively. We will include them in the final version.

2. More baseline results. In Table 1, we implement 1) GB-HPO + RS by running STN [8] with Random Search given 20 trials and 2) HPM w/o hypertraining by only updating hyperparameters with learnable mutation. Compared with HPM, GB-HPO + RS may not fully explore the hyperparameter space due to the lack of mutation-driven search. While the HPM w/o hypertraining adopts learnable mutation, the hypergradient will decrease slowly and the hyperparameters cannot be seamlessly updated along with model parameters. More results will be included in the final version.

3. Further questions. 1) S in Eq (3). S denotes a agent model in the population-based training, which maintains its parameters (θ, h) and performs one training step (with SGD) once being called. 2) Subscript T in Eqs (4-5). h_T is obtained in a chained update sequence, $(\theta_t^k(h_t^k), h_t^k) \leftarrow (\theta_{t-1}^k(h_{t-1}^k), h_{t-1}^k)$, where h_t is updated by hypergradient and mutation in each step t. Thus, minimizing h_T is equivalent to minimize this schedule: $h_T \leftarrow \cdots h_t \cdots \leftarrow h_0$. We will clarify these in the final version.

Table 1: More baseline results on CIFAR-10.

Methods	Val Loss	Test Loss
GB-HPO + RS	0.5817	0.5832
HPM w/o hypertraining	0.5944	0.6031
HPM (proposed method)	**0.5636**	**0.5649**

Reviewer 3: *1. Additional cost by attention networks*. Thanks for the valuable feedback! The proposed teacher network (*i.e.*, attention networks) is retrained for adapting to the model training process and mutating the hyperparameters on the fly. We agree that it will bring additional cost. Fortunately, the teacher network is trained on the validation set by freezing the student model, which needs a much less computing cost than training students.

2. Activation functions. Previous works like PBT [7] mainly use discrete mutation weights sampled from $\{0.8, 1.2\}$. To empower the flexibility of mutation, we leverage the tanh function to describe the mutation degree in $[-1, 1]$, leading to continuous mutation weights in $[0, 2]$ with Eq. (9). The softmax function is used to compute attention scores. Table 2 compares using LeakyRelu and Softmax in teacher model. We will provide more comparison results in the final version.

3. Hypergradient directed mutation.Thanks for the useful suggestion! We train the teacher model by minimizing $\mathcal{L}_{val}$ w.r.t the mutated hyperparameters. Thus, the hypergradient could be backpropagated to mutation weights and update the teacher network. Due to the limited space, please refer to Appendix A in the supplementary material for more details.

4. Large-scale experiments. The hypernetworks inside HPM are scalable and memory-efficient to compute hypergradients. By using the population-based training, HPM could be further parallelized to handle large-scale datasets.

Reviewer 4: *1. Learnable mutation*. Thanks for this useful suggestion! The teacher model is trained along with hypergradient descent to mutate hyperparameters adaptively, which could provide aggressive mutations in early training steps (when h_t^k exhibits a high variance) and tend to mild mutations when $\mathcal{L}_{val}$ gets converged (see Fig. 4 in the paper). We implement HPM w/o learnable mutation by performing one more hypergradient update step over the cloned hyperparameters. As shown in Table 2, this baseline method degrades the performance due to over-optimizing hyperparameters (the cloned model parameters remained unchanged) and the lack of mutation-driven search.

2. Implementation of teacher model. We thank the reviewer for this great suggestion! In our paper, we implement the teacher model as attention networks, *i.e.*, $g_\phi(h) = 1 + \tanh(W\sigma(V^{\mathrm{T}}h))$ where σ denotes the Softmax function. We expect to use attention mechanism to make V memorize different hyperparameter queries and W focus on learning mutation degree. However, the main contribution of HPM is to learn the mutations with a teacher model for combining the local hypergradient and global population-based search. Hence, some other common network choices in the learning-to-learn regime, like MLP, can also be used as the teacher model of HPM. Particularly, we could implement teacher-MLP by setting the activation function σ in g_ϕ other than Softmax, *e.g.*, setting σ as LeakyRelu. Table 2 shows the comparison result between these two teacher forms.

Table 2: More ablation studies on CIFAR-10.

Methods	Val Loss	Test Loss
HPM w/o learnable mutation	0.6139	0.6267
HPM (the proposed method)	**0.5636**	**0.5649**
HPM (T-MLP-LeakyRelu)	0.5696	0.5745