We thank the reviewers for their constructive feedback. We will incorporate these comments in the final version, and address the concerns as follows.

General Comments: Regarding Experiment. We put more experiments details in supplementary materials which includes the choice of ϵ mentioned by reviewer#1. We also used normalized gradient and ϵ-ball projection and we'll mention this in our next version. We would also like to thank reviewer#1 and #2 for their helpful advice about writing and typesetting, which will be properly dealt with in our next version.

Reviewer#1

Regarding Stronger Attack. We conduct experiments on stronger attack, the results show our approach can defense stronger attack. The results of PreAct-Res18 on CIFAR10 are shown as follows (average of three experiments)

	Clean	PGD-20	PGD-100	PGD-1000	CW attack
Madry	84.89±0.19	42.32±0.29	42.13±0.27	41.42±0.20	59.30±0.16
YOPO-5-3	83.51±0.22	43.94±0.20	43.17±0.17	42.52±0.36	60.18±0.38

Regarding Clarity. Thanks for pointing this out. The variable p is a "dual" variable. Thus in Theorem 1, we need to construct a p to satisfy the dual certificates. The algorithm uses an iterative scheme to find it. The variable p in the Hamiltonian is the same as the slack variable p. The definition of Hamiltonian is brought from physic and is well known in the control community. It can also be understood as a Fréchet Dual of the original problem.

Regarding Free-m. We would like to point out that the Free-m method is an independent and *concurrent* work (was put on arXiv on April 30 which was just before the NeurIPS' deadline). In our paper, we also show that their method is a special case of ours, namely YOPO-m-1. The epsilon used (1-7) in Free-m paper for imagenet is wired (too small) and the accuracy is far from the state-of-the-art report [1]. Imagenet is still a hard problem mainly due to limited computation resources, and we are still working on it. ([1] Feature Denoising for Improving Adversarial Robustness arXiv:1812.03411)

Regarding using the first k-layers be used for the inner-loop adversary. It is flexible to try k other than 1, but in our experiment, selecting $k = 1$ works the best. We will include an ablation study in the final version.

Regrading the analysis of m and n. Thanks for your suggestion and we will add more ablation study over this. The analysis could be found in Line145-154, we also use YOPO-3-5 and YOPO-5-3 to empirically justify the analysis.

Reviewer # 2

Regarding Twice Continuously Differentiability. The set of non-differentiable part of ReLU is of measure zero. Thus we do not think this will affect the algorithm a lot. The BP algorithm typically requires the activation function to be differentiable but works well empirically. Reviewers can consider there exist a really good smooth function to approximate ReLU. *First order differentiability is enough for the theory in our paper, while twice continuous differentiability may be required for further convergence analysis.*

Regarding Theorem 2. Theorem 2 is used to show the relationship between our algorithm and PMP, and is important for that matter.

Reviewer#3

Regarding comparison with previous work. First of all, as reviewer#1 mentioned, one of the main contributions is discovering the benefits of the control perspective in the *adversarial setting*. We agree the control perspective is not a new idea in deep learning and we have already cited the original Lecun's BP paper and other related papers. At the same time, the long training time is the *main* issue when scaling adversarial training to a larger dataset and networks. That's why most of the adversarial training papers just test CIFAR10. In our work, we showed the power of control perspective in accelerating the heavy training procedure, which we think will help the community to scale up their experiment.

Secondly, there seems to be some misunderstanding that our work is using control to model **feed forward network** but not **RNN**. It's **not time-homogeneous**. It is not clear to use how the BPTT algorithms could be applied in our setting.

Finally, our splitting method provides a new perspective on solving the optimality condition. This new perspective not only provides a description of the algorithm in a more general setting, but also inspires algorithms beyond back-propagation based training.

Regarding the training time. First of all, the computational cost (e.g. FLOP) of YOPO is theoretically smaller than the original adversarial training, typically 1/5-1/4 times smaller. The code is provided in the supplementary for reproducibility. All codes are written in Pytorch. There is also an unofficial TensorFlow code on Github showing that YOPO is quite efficient.