We thank all reviewers for their valuable feedback and suggestions. As a submission to NeurIPS, we hope our work will be useful for the large AutoML audience present in this conference. Here we will first address a shared concern on how to make the "result" section more useful, then discuss specific concerns from individual reviewers.

R1, R3, R4: Improving the usefulness of our experimental evaluation.

All three reviewers noted that PyGlove's key innovations were on the framework side rather than on the NAS algorithms side, and expressed concerns that the empirical evaluation of search results was largely orthogonal to the paper's main contributions. We agree with the reviewers' concerns. To address them, we will make the following changes: 1) mention that we have also successfully reproduced a wide variety of popular NAS papers with diverse search spaces and algorithms, as evidence for PyGlove's expressive power: NASBench, MNASNet, NAS-FPN, SpineNet, ProxylessNAS, TuNAS. 2) clarify that the "result" section should be a "case study" section, which demonstrates that PyGlove can easily try out different search spaces and algorithms, and quantifies the reduction in engineering effort.

R1: *"The language for specifying the search space may not be expressive enough..."* **and** *"... include use cases where they replicate prior work in NAS using PyGlove..."*

We agree that expressiveness is important. PyGlove can express very complex search spaces, in particular 1) conditional search spaces, representable by nested "oneof" or "manyof"; 2) interdependent parameters, which can be expressed by high-order symbolic values, discussed in Appendix A.4. The expressiveness is further verified by our reproduced NAS papers listed above, which will be added. For clarity, we will also add more details to the "search space" sub-section.

R3: *"Unlike all other packages that I have seen and reviewed, such as Keras Tuner, NNI, AutoGluon, Optuna (btw reference missing to Optuna, you should consider adding), this paper introduces something innovative and elegant."*

Thanks for going through an extensive list of AutoML toolkits in verifying the fairness of our claim and writing an in-depth analysis of PyGlove's method. We will add a discussion of Optuna in the revision.

R3: *"Define the cost more precisely."*

We will define the training/search cost more precisely using GPU hours.

R4: *"One weakness is the exposition and relation to prior work...For instance, they introduce their objects as symbolic trees, which are simply abstract syntax trees..."*

Thanks for bringing up the relation to PL/PS. To our best knowledge, our work can be related to symbolic programming, non-deterministic programming, functional programming, and program synthesis. We will discuss their connections in more detail with the following references. We would be happy to add more based on further reviewer suggestions.

- GJ Sussman, MIT, 2007, Building robust systems an essay
- Abelson, H. and Sussman, G.J., 1996. Structure and interpretation of computer programs
- Andre, D., and Russell, S. J. 2002. State abstraction in programmable reinforcement learning
- H. Søndergaard, P. Sestoft, 1992, Non-determinism in functional languages
- A Solar-Lezama, The sketching approach to program synthesis, Springer, 2009
- S. Gulwani. Synthesis of loop-free programs. In PLDI, pp. 62–73, 2011

Regarding the terminology, we preferred "symbolic tree" over "AST" for two reasons. First, as R4 suggested, "symbolic tree" was more approachable for people in the ML community. Second, the symbolic tree is declared by the user using decorators and serves to represent high-level program constructs, which is different from the AST that represents all the syntactic structures for the program. For example, the full Python AST contains information about objects' class methods, whereas our symbolic representation does not.

R4: *"Second, most of their tool/language design could be summarized as adding some kind of non deterministic/parametric choice ... It's extension to ML does not introduce anything particularly new ..."*

We agree with R4 that symbolic programming and non-deterministic programming are well-studied topics in the PL community. However, we would like to emphasize that this work is the first to introduce such concepts to AutoML to significantly reduce engineering effort, which is a novel and useful contribution. For example, PyGlove leverages symbolic manipulation to decouple the search algorithm, search space and child program, which enabled us to unify the interface among search methods with and without weight sharing. To enable symbolic programming in Python, PyGlove implements an object model for maintaining the consistency of program state during symbolic manipulation.

R4 *"Provide the grammar in the main text"*

We understand the "grammar" here as a reference to the formal definition of the search space specification. We will revise current Appendix Table 3 into a formal definition, and add it to the "search space" sub-section.