We would like to thank the reviewers for their insightful comments. As has been noted by all four reviewers, our paper presents a novel algorithm for batch reinforcement learning that is not only simpler and faster than the state-of-the-art algorithms, but also performs significantly better. Because the BCQ and BEAR code are publicly available, we were able to make a careful and comprehensive comparison of the performance of BAIL, BCQ, BEAR, MARWIL and naive behavioral cloning using the MuJoCo benchmark. For our experiments, we created non-expert training batches in a manner identical to what was done in the BCQ paper and included additional partially-trained training batches for the environments Ant and Humanoid using SAC. Our experiments showed that BAIL wins for 20 of the 22 batches, with overall performance 42% or more higher than the other algorithms. Moreover, BAIL is computationally 30-50 times faster than BCQ and BEAR. We provide robust anonymized code for reproducibility. We will also make our datasets publicly available for future benchmarking.

BAIL learns a value function by training a neural network to obtain the "upper envelope of the data". To the best of our knowledge, the notion of the upper envelope of a dataset is novel, and can possibly be applied to other RL problems in the future.

Response to Reviewer 2: Thank you for your positive feedback with an accept recommendation. In terms of testing on alternative domains, we are currently focused on MuJoCo, where Ant and Humanoid are the most challenging environments. But we will consider testing on alternative benchmarks in future work.

Response to Reviewer 4: Thank you for your positive feedback with an accept recommendation. We will add to the paper more intuition on why BAIL performs better than the other methods. Intuitively, BAIL performs better than BCQ and BEAR because BCQ and BEAR rely on carefully tuned policy constraints to prevent the use of out-of-distribution actions. A loose constraint can cause extrapolation error to accumulate quickly, and a tight constraint will prevent the policy from choosing some of the good actions. BAIL, however, identifies and imitates the highest-performing actions in the dataset, thus avoiding the need to carefully tune such a constraint. In terms of the computation, for BCQ and BEAR, we used the authors' implementations. These Q-learning based algorithms typically have more, larger networks, and sample multiple candidate actions for each update. Thus it is not surprising that BAIL can be 30-50 times faster.

Response to Reviewer 3: Thank you for your mostly positive review, and for pointing out the papers (Oh et al, 2018) and (Vinyals et al, 2019), which we will add to the related work section. In our view, all DRL algorithms are heuristics, and performance guarantees for schemes using neural-network function-approximators are rare. We remark the experimental results presented in the main body indeed use "sub-optimal training data," since those batches were obtained from training data well before optimal performance is achieved (and in many cases using sub-optimal algorithms for training). We will make this more clear in the revision. The results for the execution batches are summarized in the main body and presented in detail in the appendix.

We decided to use L_2 regularization in the definition of the upper-envelope since it leads to a clean definition and theory. In our computational experiments, we found L_2 regularization and early-stopping regularization to give similar performance, with early stopping being faster. So we chose to use early stopping in our experiments. We note that there is a well-known theory showing, under some conditions, the equivalence of early stopping to L_2 regularization (see the Deep Learning book by Goodfellow et al., Section 7.8).

Response to Reviewer 1: Thank you for the feedback. It seems you are very concerned about our table results. We would like to point out that they are definitely not "misleading". In fact, the table is presented in this manner to make fair comparisons, as clearly explained in the paper. On page 8, line 277, we explain: "For each batch, all the algorithms that are within 10% of the highest average value are considered winners and are indicated in bold." So it is possible for multiple algorithms to be in bold in a table row. Indeed, when two algorithms have very similar scores, this should be considered a tie, since their rankings can change easily when using a different random seed. To complement the tables, on page 8, line 283, we also explain how we computed the average improvement ratio, which shows that BAIL performs 42% better than BCQ, and 101% better than BC.

You ask "how close to expert does the training data have to be for BAIL to perform competitively?" This question is answered starting at page 2, line 39, where we explained that we generated a large variety of batches, including far-from-expert datasets with different random noise levels and with different learning agents. Our batch generation procedure is also consistent with the procedures in prior work. Thus Table 1 answers your question by showing that BAIL performs well on all the training datasets with various noise levels.

For the execution batches, we used the same hyper-parameters for the training set, and the results show that BAIL is better than BEAR, similar to BCQ, and slightly weaker than naive behavioral cloning. If we change the hyper-parameters we will get at least the same performance as behavioral cloning (which is the strongest in the execution batches). Thus, in summary, BAIL is the clear winner for the training batches, and BAIL, BCQ, and naive behavioral cloning are equally good for the execution batches. For the other minor points you suggested, thank you for pointing them out, and we will modify them in our revision.