We thank the reviewers for their helpful comments which we will address in our final submission, as outlined below.

Reviewer 1: We are going to improve the expansion and heuristic sections in the final submission.

A naive implementation of the expansion step can be computationally expensive, since we have to compute the potential expanders for multiple values of p until we find one such that $|g_t(W_t^\epsilon, A_t(p))| > 0$. In our implementation, we keep track of the nodes such that $|g_t(W_t^\epsilon, A_t(p))| = 0$ to make this step efficient. Further improvements are possible by exploiting properties of the kernel (or metric $d(\cdot,\cdot)$), which often encodes that expandability is a local property. Similar locality considerations can greatly improve the efficiency in many parts of the algorithm.

We keep track of $\bar{S}_t^o$ and $\bar{S}_t^p$ using the graph library `networkx` and by adding/removing edges as we acquire new data points. Since at each iteration only few edges are removed/added, this step is not computationally expensive. We are going to add details of an efficient implementation to the appendix and will release our code.

Reviewer 2: We agree that there are similarities between GoOSE and the method by Berkenkamp 2017 (B17), in that we use the same statistical analysis tools to build accurate confidence intervals and guarantee reachability-based safety constraints during exploration. However, there are fundamental differences between the two works.

The most important difference is that of exploration. While B17 provides safety guarantees in the continuous domain, for the exploration analysis they discretize the domain. Thus, in our context, their method can be thought of as an extension of SafeOpt by Sui 2015 and SafeMDP by Turchetta 2016 to the continuous domain. While the three methods focus on safety in different settings and thus use different tools to construct pessimistic estimates of the safe set, they all collect data within their respective safe set estimates by following the same strategy that provably explores the entire safe set. In particular, they all expand the safe set by reducing the maximal uncertainty within the current safe set[1], which is easy to analyze but can be very data inefficient in practice. Thus, B17 addresses the problem of safe identification of dynamical systems with continuous state-action spaces by learning about the transition dynamics uniformly over the domain. This is evident from their exploration guarantees in Theorem 4 iii), which hold exclusively for the exploration strategy introduced in equation (6) and which is the same as in SafeOpt and SafeMDP.

In contrast, GoOSE is a safety add-on layer with strong safety and completeness guarantees that focuses on improving the sample efficiency in exploration. In particular, our analysis allows *any* goal-directed exploration strategy to be used in order to efficiently learn about the part of the domain that is relevant toward the achievement of a given goal. Since the method B17 fundamentally builds on a discrete exploration analysis, it might be possible to use the more efficient exploration scheme of GoOSE in their setting too. While this would require more analysis due to inherent challenges in continuous domains, GoOSE provides a first step in this direction. Thus, we think that our work is complementary to B17 rather than overlapping with it.

There are several other differences between the two approaches: *(i)* Different sources of uncertainty and, therefore, of risk. GoOSE considers safety-critical external environments with known transition dynamics, while B17 addresses uncertainty in the dynamics but does not account for external factors. *(ii)* Different constraints: B17 focuses on stability constraints, a specific type of constraint that is relevant in dynamical systems. GoOSE looks at level sets of generic unknown functions, which can model safety constraint in a variety of IML scenarios that may not involve dynamical systems, including BO and active learning. *(iii)* Different assumptions: GoOSE assumes a known model and, under this assumption, stability constraints have been extensively studied. B17 assumes that a Lyapunov function, whose choice implicitly affects the quality of the estimate of the asymptotically stable region, is given. Moreover, B17 implicitly assumes that state action pairs are safely reachable, which is not required for GoOSE.

Reviewer 3: γ_t denotes the information capacity of the safety constraint, which is an information theoretic complexity measure of the encoded function class. We use it quantify how many data points we require to learn the function up to a certain accuracy. We will clarify this in the final submission.

Without additional assumptions on the environment, it is not possible to provide a bound that does not require complete exploration. As a counter example, consider a graph $\mathcal{G}$ that is a chain of nodes, where each node only provides information about the safety of the next one in the chain. Thus, if the unsafe IML oracle suggests a node at the end of the chain, we must individually learn about each node in the chain to expand the safe set, which is fundamentally what the bound in Theorem 1 encodes. In the general case, this is intuitive if one thinks of SMDP by Turchetta 2016 as the safe equivalent of breadth first search and GoOSE as the equivalent of A^*. Depending on the graph, the location of the target and the heuristic, both breadth first search and A^* may need to visit all the nodes in the graph to find a path to the target. Similarly, in our worst case analysis, it may be necessary for both GoOSE and SMDP to learn about the safety of all nodes in the graph before evaluating the oracle suggestion.

[1] SafeOpt additionally considers maximizers and expanders for efficiency, but their analysis focuses on complete exploration.