We thank all reviewers for their detailed feedback. We will be sure to address all questions and incorporate all suggestions in the final version of the paper. Please see individual responses below.

Reviewer #1. Thank you for your positive comments!

Reviewer #2. ***"The decoding procedure in Phase 3 is quite elaborate..."***. The regression procedures in phases 1 and 2 only allow us to approximate K_∞, representing the solution of the Riccati equations for (A, B, Q). This does not immediately allow us to approximate the optimal controller, which is a function of the latent state $\mathbf{x}_t$ and is given by $\pi_\star(\mathbf{y}_t) = K_\infty \mathbf{x}_t$. To approximate $\pi_\star(\mathbf{y}_t)$, we need to learn a decoder $\hat{f}_t(\mathbf{y}_t) \approx \mathbf{x}_t$. The decoder learned during the first phase is only guaranteed to be accurate on the state distribution generated by taking random actions from the start state. There is no guarantee that this decoder will be accurate on the state distribution induced by a near-optimal policy, so we need to learn a new decoder at each step with new data generated using $\widehat{\pi}$. We emphasize that this is simply an instance of a common technical issue in statistical learning; In general, given a function $\hat{f}$ such that $\mathbb{E}_P\|\hat{f}(x) - f^\star(x)\|^2 \leq \varepsilon$ for a distribution P, we have no guarantee that $\mathbb{E}_Q\|\hat{f}(x) - f^\star(x)\|^2 \leq \varepsilon$ for a different distribution Q unless we put strong structural assumptions on either P/Q or the function class $\mathcal{F}$. Since we do not make such assumptions, we solve this problem by re-learning on the new distribution.

"***In Algorithm 4 line 28, why is noise added to the optimal policy...?***" This is closely related to the point above: Since we train on a distribution in which random noise is injected, our decoders are only guaranteed to have low error on this distribution. However, since the noise decays with $O(\varepsilon)$, the resulting controller is still ε-suboptimal.

"***Most practical systems are only locally linear... How difficult is it to extend this algorithm to the locally-linear setting?***" Extending our algorithm to the locally linear setting is a very exciting direction for future research, but we are not yet aware of sample complexity guarantees for locally linear control even when the state is fully observed, let alone for the more challenging nonlinear-observation setup we consider.

Reviewer #3. ***"The paper lacks any empirical evaluation...With such an experiment this paper merits a higher score in my opinion"*** We believe that our paper represents a substantial theoretical contribution and stands on its own merits even without experiments. Nonetheless, we have performed some basic validation experiments, which we can include in the appendix, focusing in Phase 1 and 2 of the algorithm for simplicity. We considered a 2-d Newtonian dynamical system with unit process noise, where $A \in \mathbb{R}^{4\times4}$ upper triangular matrix and $B \in \mathbb{R}^{4\times2}$. We slightly dampened the dynamics to ensure that $\rho(A) < 1$. The final system is 2-controllable. Observations come in pairs of images, one for position information and one for velocity information. Each image contains one green pixel representing either a vector of position or velocity. Greyscale noise is added to the rest of the pixels (see Figure 1). We model the function h using a 3-layer convolutional neural network with Leaky ReLu nonlinearity. After executing phases 1 and 2 of our algorithm, we successfully recover the systems' dynamics, i.e. the matrices A and B, up to a similarity transform. In our preliminary experiments, using $n_{\mathrm{id}} = 30000$, we can recover the system matrices up to element-wise absolute error of < 0.07.

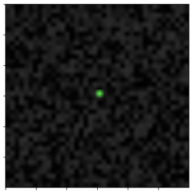

Figure 1: Green dot = position or velocity vector.

Reviewer #4. ***"The paper uses "decoder" in place of what in traditional autoencoding..."*** We will add a note on this to avoid any confusion. ***"...it is certainly not the case that this paper is introducing this as a novel problem..."*** Our main claim is that we introduce the theoretical problem of developing *finite-sample bounds* for this setting, which we believe is true. We will update the abstract to be more precise.

"... if F is a family of neural networks, isn't the search space (capacity) $|F|$ increases exponentially? (much faster than T?)" Our sample complexity depends *logarithmically* in $|\mathcal{F}|$ and polynomially in T, and so even if $|\mathcal{F}|$ were to grow exponentially in T, this would not be an issue—the sample complexity would remain polynomial in T. More broadly, we emphasize that the $\log|\mathcal{F}|$ factor in our theorem arises from a standard generalization bound for the square loss, and can trivially be replaced by more standard learning-theoretic quantities such as the Rademacher complexity or covering numbers of $\mathcal{F}$ (indeed, if you look at the appendix, our intermediate results are already in terms of covering numbers). In particular, this means that we can appeal to modern Rademacher complexity bounds for deep neural networks, such as Bartlett et al. (2017) or Golowich et al. (2018). We will make this clear in the final version.

"During phase 3, is the sequence of decoders f_t has any convergence promise? Can you give more detailed explanation of how parameterized neural network decoder updated for each parameter?" The objective function used to learn the (parameterized neural network) decoder $\hat{f}_t$ is an average square-loss see (20) and (21). Any out-of-the-box neural network architecture / training algorithm (e.g., SGD) can be used to minimize the objective. As long as the objective is approximately minimized, Theorem 4 ensures that the decoders $\hat{f}_t$ will have low decoding error, and thus lead to an approximately optimal controller. The sequence $(\hat{f}_t)$ is not guaranteed to converge to a fixed decoder, but this is not required for our theoretical guarantees to hold.