We thank all the reviewers for their comments.

Reviewer #1. Regarding the relationship between the query lower bound, the tolerance τ and the error ϵ, note first that we must have $\tau \lesssim \epsilon \lesssim \beta$ (where β is a norm lower bound for the concept class), as we discuss in Appendix A. Moreover, because of technical requirements on τ for the DKKZ20 result, we end up picking τ as a function of ϵ in our reduction. So as ϵ decreases, τ decreases as well, and we get a series of incomparable (though still exponential) bounds due to the tradeoffs between query complexity and tolerance. We will include a clearer discussion of these issues in Appendix A. We will also address your additional feedback (including expanding on lines 32-34) when revising the manuscript.

Reviewer #4. (1) A major goal in deep learning is to find provably efficient algorithms for learning classes of neural networks. There are several papers a year on this topic. Here we are showing for the first time that if there is noise in the labels, this goal is impossible even for the simplest networks, ones that consist of a single activation. As we mention in the paper, our results rule out polynomial-time algorithms for *any* nonpolynomial activation (halfspaces and ReLU are just examples). The activation can take Boolean or real-valued outputs. Since algorithms for learning polynomial activations are known, this characterizes the computational complexity of learning single activations.

(2) Regarding the context for Theorems 1, 2, 3, we note that these lower bounds hold broadly for all statistical query algorithms and hold regardless of the structure of the learner's output hypothesis. As for the tolerance, much of its significance lies in capturing what in traditional PAC algorithms would be the sample complexity. Specifically, it takes $\Theta(1/\tau^2)$ samples to simulate an SQ of tolerance τ, and this is sometimes known as the *estimation complexity* of an SQ algorithm.

As for a comparison with prior results such as KKMS08 and DKN10 for halfspaces, our SQ lower bound (like all SQ lower bounds) states that any SQ algorithm must use *either* $\exp(n)$ queries *or* very small ($n^{-1/\epsilon}$) tolerance (which corresponds to sample complexity). KKMS08 (with its sample complexity of n^{1/ϵ^2}) falls into the latter category, and our tolerance bounds show that this is nearly optimal.

(3) Regarding your comments on the difficulty of reading the paper without prior background in the SQ literature, we acknowledge some of your points on where the writing could be improved. It is true that Corollary 2.4 is somewhat confusing. The "accompanying norm lower bound" refers to a β such that $\|g\| \geq \beta$ for all $g \in \mathcal{G}$, as used in Theorem 2.2. Part (a) of the corollary ((b) and (c) are similar) follows from "instantiating" the generic construction $\mathcal{G}$ of Theorem 2.3 with $\phi = \mathrm{ReLU}$ to obtain say $\mathcal{G}_{\mathrm{ReLU}}$, noting that functions in $\mathcal{G}_{\mathrm{ReLU}}$ satisfy a norm lower bound of $\beta = \Omega(1/k^2)$ (with proofs deferred to Appendix D), and then using $\mathcal{C} = \mathcal{G}_{\mathrm{ReLU}}$ in Theorem 2.2 to obtain the final lower bound. As for Theorem 5.1, we do state "suppose that Assumption 3.1 holds for $\mathcal{H}_{\mathrm{ReLU}}$" to try to be as clear as possible. As for Theorem 1.1 (and 1.2 and 1.3), they are proved in Section 5, and we will note this. We will also try to make the organization of the lemmas clearer.

Regarding additional feedback:

- The concept class in Def 2.1 can be either real-valued or Boolean.
- As defined in lines 124-126, D_{f^*} in this case refers to the unique distribution on $\mathbb{R}^n \times \{\pm 1\}$ defined by the (Boolean) p-concept f^*.
- One important reason for using Frank–Wolfe is because it avoids a projection step, as would be required in say standard projected GD. In our L^2 function space, it is not natural to require the base learner to find such a projection of the functional gradient onto $\mathrm{conv}(\mathcal{H})$. Another important reason is that Frank–Wolfe uses a *linear* optimization subproblem, and this is important in order to preserve "SQness" during the reduction (see lines 215-227 and 242-245).

We will address all these points when revising the manuscript.