We thank all reviewers for their helpful comments, and provide our response below (reviewers' comments are italicized).

Reviewer 1. 1) [*Polynomial approximation to sigmoid*] We in fact experimented with both $r = 1$ and $r = 3$ for the degree of the polynomial approximation to sigmoid, but observed that $r = 1$ achieved comparable test accuracy to conventional logistic regression, hence only reported $r = 1$. CodedPrivateML can be applied to any r satisfying $N \geq (2r + 1)(K + T - 1) + 1$, where N, K, T represent the number of clients, parallelization parameter, privacy parameter, respectively. The same approximation was also used for the baselines, hence using a larger r would have the same impact on all protocols in terms of accuracy. In terms of total training time, the performance gain remains the same because the training time of both CodedPrivateML and baseline protocols increases linearly as r increases.
2) [*Empirical evaluation with only 50 clients*] We would have liked to go beyond 50 clients, in fact our gains would be even better, but we are limited by the budget cost of running our experiments on the Amazon EC2 cloud. Currently our range is between 10-50 clients, which, compared to the state-of-the-art, already corresponds to a $10\times$ increase.
3) [*Communication cost and size*] Our communication cost per client is $O(mdN/K + dNJ)$ where J denotes the number of iterations. When $N = 50$, the communication size per client is 126MB with the CIFAR-10 dataset. We note that increasing N has two major impacts on the training time: 1) reducing the computation time by choosing a larger $K = O(N)$, and 2) increasing the communication time. Hence, when N is small (~ 10), the computation time dominates the total training time. For the baseline protocols, both communication and computation time increases as N increases, and when $N = 50$, the communication size per client is 900MB. We will add this analysis in our revised version. We will also apply the changes suggested by the reviewer to improve the overall presentation.

Reviewer 2. We would like to clarify/correct a few key points in reviewer's comments, with the hope that this will help highlight the key contributions of our paper.
1) [*Novelty*] It is true that much of the privacy-preserving machine learning (PPML) literature is based on well-known homomorphic encryption or secure MPC primitives. Our contribution, on the other hand, is building the first PPML framework that can scale to a significantly larger number of clients than state-of-the-art PPML approaches (i.e., beyond 3-4 clients) with strong (information-theoretic or statistical) privacy guarantees. This is the first PPML approach that reduces the computation load per client as the number of clients increases, which we hope will open up further research.
2) [*Susceptibility to poisoning*] Our focus is on semi-honest (passive) adversaries (as stated in Section 1), which precludes poisoning attacks. PPML in general is susceptible to poisoning attacks, and this is also true for all MPC-based PPML schemes that we compare with (and build upon). Poisoning attacks is certainly an interesting future direction.
3) [*Privacy against unbounded adversaries*] We would like to emphasize that information-theoretic and statistical privacy are synonyms for "privacy against computationally unbounded adversaries". As discussed after Theorem 1, our privacy guarantee follows from the fact that all building blocks of our algorithm guarantees either information-theoretic privacy or statistical privacy of the individual datasets against any collusions between up to T clients.
4) [*Lack of comparison with the most recent MPC protocols*]: Please see our response to Comment 1 from Reviewer 4.
5) [*Figure 3 - training time decreases*] This is in fact the main contribution of our paper. Our encoding procedure decreases the computation load per client (for training) as the number of clients increases, hence decreasing the overall training time as observed in Figure 3.b. Figure 3.a demonstrates a smaller dataset, in which case the encoding time starts to dominate over the benefits gained from encoding beyond 20-30 clients, and the training time starts to increase.
6) [*Extension to DNN*] This paper focuses on simpler models, and is a first step towards realizing scalable PPML for DNNs. Please note that, even in this domain, the problem is already very challenging and an active area of research.

Reviewer 3. 1) [*Packed secret sharing*] Thank you for raising this. We agree with the reviewer that the encoding of Lagrange coded computing is the same as packed secret sharing while [36] proves its optimality in terms of *recovery threshold* to compute an arbitrary multivariate polynomial. Recovery threshold is defined as the minimum number of computation results that the protocol needs to wait to guarantee decodability. We will add these points in our final paper.
2) [*Pseudo-random secret sharing*] Thank you for this suggestion, we agree with utilizing pseudo-random secret sharing to generate the random parameters instead of a crypto-service provider, and will incorporate this in our final paper.

Reviewer 4. 1) [*Comparison to more recent work*] There are in fact more recent MPC-based privacy-preserving machine learning (PPML) protocols, but we are not aware of any work that goes beyond 3-4 parties. Our work is the first PPML solution that goes substantially beyond that. As there was no prior work at our scale, we implemented two baselines based on well-known MPC protocols which are also the first implementations at that scale. Following the reviewer's suggestions, we ran additional experiments to compare the total training time of CodedPrivateML with [26] (on the same Amazon EC2 setting of [26], by measuring both offline and online time, on a synthesized dataset as in [26, Table 2] with 10000 data points and 1000 features). We observed that CodedPrivateML with $N = 50$ clients achieves $20.9\times$ speedup in the total training time against the OT-based approach of [26], while the scalability of [26] is limited to 2 parties. We will include these results with additional experiments in our revised version.
2) [*Memory consumption and communication costs*] We summarize the communication and storage cost per client in Table 1 where m, d, J denote the number of data points, features, and iterations, respectively. We will include this table in our revised version.

Table 1: Complexity Analysis.

	Comm.	Storage
Data	$O(mdN/K)$	$O(md/K)$
Model	$O(dNJ)$	$O(dN)$