Reviewers, thanks for your feedbank. We want to reiterate that our main contribution is *practical* poisoning, demonstrated by being first to poison DNNs trained from scratch & first to break an industrial API (Google) via poisoning.

R1 & R2: **Justification for two-step approximation** Our decision to unroll the inner objective for a few steps (K=2) vs. the whole training (K$\approx 10^5$) is a key distinction between our approach and that of Domke, MacLaurin, & Munoz-Gonzalez. Our ablation study vs. K in Supplementary Material Fig 7 confirms that gains diminish beyond K=2, corroborating with prior work on higher-order backprop, e.g. Finn et al MAML. There's two reasons for this. First, Shaban et al "Truncated Back-propagation for Bilevel Optimization" §3.1 theoretically show the approximation error of few-step gradient evaluations to decrease exponentially with K, so the gradient is already well approximated with a few steps. Meanwhile, unrolling many steps leads to numerical instability due to the ill-posedness of the gradient operator, as observed in MacLaurin et al even for convex problems. Early stopping of the approximation prevents this instability. Second, Franceschi et al "Bilevel Programming for Hyperparameter Optimization and Meta-Learning" §5.1 shows that computing the exact bilevel solution (our Eqs. 1-2) can lead to overfitting the outer objective. They show instead that the approximate gradients from small K act as an implicit regularizer for *generalization*, which is more desirable here than the exact solution b/c the initialization, SGD order, architecture, and other nuisance variables differ when the bilevel objective is evaluated by the victim. Lastly, small K is cheap, while K$\approx 10^5$ is intractable. In summary, the best solution to *poisoning problem* is one that generalizes and computes fast; both point to a small K approximation. **Justification for reinitialization and ensembling** As R1 correctly points out, reinitialization and ensembling are regularization techniques to help poisons further build invariances to nuisance variables above by seeing more surrogate models, yielding better generalization. *We've performed an ablation study (which will be added to the revision) showing that neither reinitialization w/o ensembling, nor emsembling w/ fixed initialization, suffice to yield poisoning success*. Algorithmic details will be more clearly hashed out in §2.2. **Related bilevel work** We will relay the discussions above, highlight relation to other bilevel methods, and include the valuable citations from R2.

R1: **Why more unrolls hurts** Actually, Supplementary Material §G says K$\geq$2 "seems to not affect the result much." **Why 24-model ensemble** Our ablation on ensemble size in Supplementary Material §E sees gains diminsh beyond 24. **1% budget needed for decent success** We need only 0.04% (20 poison) budget to achieve $\sim$90% success for fine-tuning (Fig 3) and $\sim$20% success for training-from-scratch (Fig 4)—alarming levels for industrial security. **Compare to convex polytope (CP)** *Using Zhu et al's setup for CP, MetaPoison gets 60% success on ResNet20 transfer learning*, compared to Zhu's 52%. **Defense evaluation** We obtained code for Peri et al's "Deep *k*NN Defense Against Clean-label Poisoning Attacks" (which reports 100% detection of CP attacks w/ minimal false positives), and evaluated on MetaPoison. *Deep-kNN fails to detect any MetaPoisons at any k*. This makes sense as Fig 3 shows MetaPoison's features don't lie in the target's neighborhood, whereas FC & CP's do. These and the CP results above will be added. **Line 316**'s substantiated by Fig 5. Our method is also **reverse-mode autodiff** based like the literature. We're unaware of **better complexity non-autodiff** methods applicable to DNNs and are open to suggestions.

R2: **Indiscriminate and multi-target attacks** Per your request, *we ran new attacks along the indiscriminate/multi-target/single-target spectrum, including 1. fully indiscriminate (error-generic), 2. indiscriminate for specific class (error-specific), 3. multiple (5) distinct target objects, 4. multiple (>10) augmentations of same target object*. Briefly, MetaPoison did okay on the more indiscriminate attacks (error-increase of 8% on attack 1 and 15% on attack 2 using 5% budget) and quite well on the more targeted attacks (success of 34% on attack 3 and 52% on attack 4 using 1% budget). Practically, targeted attacks are far more concerning, as indiscriminate attacks can be easily detected by evaluating on a holdout set. Detailed plots will be added to the revision. **Compare to Munoz-Gonzalez** Munoz-Gonzalez focuses on proof-of-concept using a toy setup (fixed initialization, small dataset of 1000, unbounded perturbation, label flips) rather than practicality, whereas MetaPoison focuses on practicality by considering real-world constraints. Munoz-Gonzalez's method differs from ours in that they unroll the whole training procedure, don't do reinitialization or ensembling, and use nonstochastic GD. *Our replication of their method and setup yielded comparable performance (6% indiscriminate error above label-flipping with 5% poisons) on an (unrealistic) victim with the same initialization but null performance (0%) on victims with different initializations*. Complexity-wise, Munoz-Gonzalez reports O(K) (K = num training steps) per outer step per poison while ours is O(1) since we fix K=2 and ensemble=24. **Computation cost vs. Shafahi** Per poison, MetaPoison takes 2 (unrolling steps) x 2 (backprop thru unrolled steps) x 60 (outer steps) x 24 (ensemble size) = 5760 forward+backward propagations. In contrast Shafahi reports 12000 forward+backward props. Thus MetaPoison has similar cost if we discount training of the surrogate models. This is reasonable given that pretrained surrogate models along with intermediate checkpoints can be precomputed and reused. Our latest code samples weights from a database of checkpoints w/ no degradation.

R4: **Ensembling defense** Recall that success is averaged over multiple runs *and* target images. For a *specific* target image, success tends to be quite stable; e.g., success for bird 0 and 6 is 100% over 4 runs on Google AutoML (each w/ different architectures). This suggests that ensembling will not be effective. **Unstable across architectures** It's true poisons don't transfer to all models equally, though transferabilty to VGG is quite good–87% from ConvNetBN. **Non-image tasks & dirty-label attacks** While outside this paper's scope, these are important and will be a focus of future work. **Fig 7 (right) unclear** We'll clarify this figure more in the revision. **Theoretical poisoning** We'll expand our related works with this and other theoretical works.