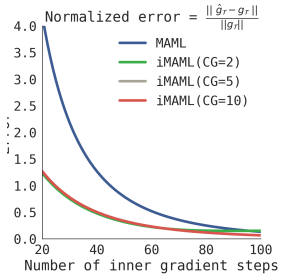

Figure 1: Grad Error

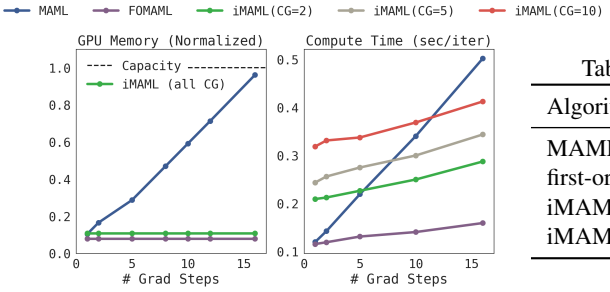

Figure 2: Compute and memory time

Table 1 : MiniImagenet results.

Algorithm	5-way 1-shot
MAML	48.70 $\pm$ 1.84 %
first-order MAML	48.07 $\pm$ 1.75 %
iMAML GD (ours)	48.96 $\pm$ 1.84 %
iMAML HF (ours)	49.30 $\pm$ 1.88 %

We thank the reviewers for the thoughtful feedback! The primary concern of R1 and R2 was a comparison on a more complex dataset. **We have added a comparison on MiniImagenet in Table 1,** and will include this and 5-shot results to the revised version. Again, we find that both the gradient descent (GD) and Hessian-free (HF) versions of the iMAML algorithm perform better than MAML.

Reviewer #1: Thank you for the thoughtful questions!

1. Lemma 1 only requires convexity of G which is easily realizable since the regularization strength (λ) is under our control. Second order smoothness is needed only for finite-time analysis (Theorem 1), and this assumption is made in a number of optimization works works (e.g. Nesterov & Polyak '06, Jin et al. '17, '19, and references therein). We **do not** require convexity of $\mathcal{L}$ anywhere. We also emphasize that, to our knowledge, this is the first finite-time analysis of bilevel optimization problems, which we feel is an important fundamental contribution.
   Furthermore, regularity conditions are often needed for analysis but not to run the algorithm. Successful algorithms such as momentum and AdaGrad are well understood only for convex problems, but are a staple in modern deep learning. Similarly, iMAML shows promising empirical results.
2. We added results on the MiniImagenet domain above.
3. MAML uses GD in the inner level and Adam in the outer level. iMAML also uses Adam in the outer level.
4. Assumptions vs practice was addressed along with (1) above. Please see R2.1 and R2.2 for reasons as to why iMAML might perform better than MAML. On top of these, finite precision considerations might impact practical performance. While understanding the impact of finite precision is an interesting question, it is outside the scope of this work, and orthogonal to our main contributions. We will re-word line 274 appropriately in the revised version.
5. Line 172 refers to MAML, which backpropagates through the optimization path. There is no notion of derivatives through discontinuities or non-differentiable operations like linesearch. We will elaborate in the revised version.

Reviewer #2: Thank you for the valuable comments. We will strive to incorporate all of them for the revised version. Due to space constraints, we address a subset here.

1. Superior performance of iMAML is likely due to more GD steps or due to more powerful optimizer (HF). MAML cannot handle both of these due to memory constraints, requirements of higher-order derivatives, and inability to differentiate through linesearch. We will make this clear in the revision.
2. Even in cases where MAML can be implemented without the above limitations (e.g. small problems), iMAML can better approximate the exact bi-level gradient with finite iterations when the Hessian is smooth (Theorem 1 analyzes this explicitly). To illustrate this, Figure 1 presents a synthetic regression example using Fourier-features of inputs (thus predictions are non-linear in inputs, but linear in parameters). This allows us to analytically compute the exact meta-gradient and compare different algorithms with it. The condition number (κ) is large, thereby necessitating many GD steps. We find that both iMAML and MAML asymptotically match the exact meta-gradient, but iMAML computes a better approximation in finite iterations.
3. Figure 2 presents compute and memory trade-off for MAML and iMAML (on 20-way-5-shot Omniglot). Memory for iMAML is based on hessian-vector product, and is same for GD and HF, and indipendent of number of CG iterations. Memory for MAML grows linearly in grad steps, quickly hitting GPU capacity. Computational cost for iMAML is similar to FOMAML with a constant overhead for CG that depends on the number of CG steps. Compared to FOMAML, the compute cost of MAML grows at a faster rate. FOMAML requires only gradient computations, while backpropagating through GD (as done in MAML) requires a hessian-vector products at each iteration, which are more expensive than gradients.
4. On sinusoid, we verified that iMAML performs better across the number of GD steps. We will update the plot.

Reviewer #3: Thanks for the thoughtful comments. Memory and compute trade-offs are presented in Fig. 2 (please also see R2.3). We will include more details on implicit differentiation and trade-offs of different algorithms in the revised version. iMAML can support a wider set of optimizers, requires less memory than MAML, and has comparable compute requirements. Thus, iMAML is beneficial in cases requiring extended or complex optimization loops.