Implicit Regularization in Overparameterized Bilevel Optimization

Paul Vicol [1 2] **Jonathan Lorraine** [1 2] **David Duvenaud** [1 2] **Roger Grosse** [1 2]

Abstract

Bilevel problems involve inner and outer parameters, each optimized for its own objective. Most prior work makes the simplifying assumption that the inner and outer objectives have unique solutions, but often in practice, at least one of them is overparameterized. In this case, there are many ways to choose among equivalent optima, which lead to different results. Building on recent studies of implicit regularization in single-level optimization, we investigate the inductive biases of different gradient-based algorithms for jointly optimizing the inner and outer parameters. We distinguish between two different solution concepts—cold-start and warm-start equilibria—and show that the converged solution or long-run behavior depends to a surprising degree on algorithmic choices such as the hypergradient approximation, which depends on higher order Hessian-vector products. We also show that with warm-start equilibria, the inner parameters can encode a surprising amount of information about the outer objective, even when the outer parameters are low-dimensional.

1. Introduction

A bilevel optimization problem consists of two nested subproblems, called the *outer* and *inner* problems, where the outer problem must be solved subject to optimality of the inner problem. Let $F, f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ denote the outer and inner objectives, respectively, and let $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$ denote the outer and inner parameters. The bilevel problem is defined as follows:

$$\mathbf{x}^* \in \arg\min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}^*) \tag{1}$$

$$\mathbf{y}^* \in \mathcal{S}(\mathbf{x}) = \arg\min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \tag{2}$$

Important examples of bilevel optimization in machine learning include hyperparameter optimization [1–5], dataset distillation [6, 7], influence function estimation [8], meta-learning [9–11], example reweighting schemes [12, 13], neural architecture search [14, 15], multi-agent RL [16], adversarial learning [17, 18], and implicit layers [19]. Gradient-based approaches to bilevel optimization require computing the gradient of the outer objective with respect to the outer parameters, called the *hypergradient*. This is defined as follows for a given solution $\mathbf{y}^* \in \mathcal{S}(\mathbf{x})$, which is called a *best-response* to $\mathbf{x}$: $\frac{dF(\mathbf{x},\mathbf{y}^*)}{d\mathbf{x}} = \frac{\partial F}{\partial \mathbf{x}} + \frac{\partial F}{\partial \mathbf{y}^*}\frac{\partial \mathbf{y}^*}{\partial \mathbf{x}}$. For many problems of interest, such as hyperparameter optimization, the direct gradient $\frac{\partial F}{\partial \mathbf{x}}$ is 0 (e.g., regularizers are not typically applied at validation-time); $\mathbf{x}$ affects F indirectly through its influence on the inner parameters $\mathbf{y}^*$, and thus the response Jacobian $\frac{\partial \mathbf{y}^*}{\partial \mathbf{x}}$ is crucial. Bilevel problems are challenging to optimize because exactly computing this Jacobian is typically intractable; approximating the Jacobian crucially depends on efficient higher-order Hessian-vector products. We investigate the effect of different approximations on the solutions obtained in overparameterized problems.

There are many optimization algorithms that can be used for bilevel problems. One approach is to run the inner optimization to convergence and compute the gradient for the outer parameters by differentiating through the unrolled optimization [1, 2, 5] or using implicit differentiation [20, 21, 4]; we refer to this as *cold-starting*. This is impractical due to the expense of full inner optimization. Instead, a common approach is to jointly optimize the inner and outer parameters in an *online* fashion [22, 3, 23, 24], e.g., alternating gradient steps with their respective objectives. In this case, the inner parameters are *warm-started* from the approximate solution to the inner optimization, given the outer parameters in the previous iteration. The optimization dynamics lead to an implicit regularization effect, which we investigate.

Contributions.

- We formalize warm-started bilevel optimization and consider *warm-start equilibrium* as the solution concept for the resulting game. We also consider *cold-start equilibrium* as the solution concept for cold-started bilevel optimization.
- We investigate how different approximations to the hypergradient $\frac{dF}{d\mathbf{x}}$—including iterative and implicit differentiation—lead to different outer solutions $\mathbf{x}^* \in \arg\min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}^*)$.
- We present synthetic tasks illustrating the effects of hypergradient approximations and overparameterization in the inner and outer problems.

[1]University of Toronto [2]Vector Institute. Correspondence to: Paul Vicol <pvicol@cs.toronto.edu>. *ICML 2021 Beyond First-Order Methods in ML (BFOM) Workshop.*

	Cold-Start	Warm-Start
Update	$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \frac{\partial F}{\partial \mathbf{y}^*} \frac{\partial \mathbf{y}^*}{\partial \mathbf{x}}$ $\mathbf{y}^*_{t+1} \in \arg\min_{\mathbf{y} \in \mathcal{S}(\mathbf{x}_{t+1})} \|\mathbf{y} - \mathbf{y}_0\|^2$	$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \frac{\partial F}{\partial \mathbf{y}^*_t} \frac{\partial \mathbf{y}^*_t}{\partial \mathbf{x}}$ $\mathbf{y}^*_{t+1} \in \arg\min_{\mathbf{y}} \{f(\mathbf{x}_{t+1}, \mathbf{y}) + \frac{\epsilon}{2}\|\mathbf{y} - \mathbf{y}_t\|^2\}$
Response Jacobian	$\left(\frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}^\top}\right)^{-1} \frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{x}}$	$\left(\frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}^\top} + \epsilon I\right)^{-1} \frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{x}}$
Neumann Approx.	$H^{-1} \approx \sum_{k=0}^{K} (I - H)^k$	$(H + \epsilon I)^{-1} \approx \sum_{k=0}^{K} ((1-\epsilon)I - H)^k$

Table 1: Comparison of warm-start and cold-start bilevel optimization.

2. Background

We provide an overview of related work in App. A, and a summary of our notation in App. B.

Approximating the Best-Response and its Jacobian. In practice, exactly computing the best-response or its Jacobian is expensive, and one typically approximates $\mathbf{y}^*$ or $\frac{\partial \mathbf{y}^*}{\partial \mathbf{x}}$ or both; this leads to two sources of approximation error for the hypergradient. A common approach to approximate the best-response is to use truncated unrolls of the inner problem rather than full unrolls [1, 2, 26, 5], so that $\mathbf{y}^*(\mathbf{x}) \approx \Phi_k(\mathbf{y}_0, \mathbf{x})$, representing k steps of unrolling from initialization $\mathbf{y}_0$ with outer parameters $\mathbf{x}$. The two main ways to compute the best-response Jacobian are: 1) differentiation through unrolling $\frac{d\mathbf{y}^*}{d\mathbf{x}} \approx \frac{d\Phi_k(\mathbf{y}_0, \mathbf{x})}{d\mathbf{x}}$; or 2) implicit differentiation [21, 27, 20, 28, 29, 4, 30, 31], which is applicable when we are at the converged solution to the inner problem, $\frac{d\mathbf{y}^*}{d\mathbf{x}} = -\left(\frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}^\top}\right)^{-1} \frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{x}}$. The exact inverse Hessian is intractable to compute for large networks; two common approximations involve: 1) using truncated conjugate gradient (CG) [29], and 2) using the truncated Neumann series [32, 4]: $\left(\frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}^\top}\right)^{-1} \approx \sum_{j=0}^{K} \left(I - \frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}^\top}\right)^j$. Lorraine et al. [4] showed that unrolling differentiation for i steps starting from optimal inner parameters $\mathbf{w}^*$ is equivalent to approximating the inverse Hessian with the first i terms in the Neumann series. Both CG and Neumann can be implemented with efficient Hessian-vector products using modern autodiff libraries [33–36]. Mathematically, the IFT is only applicable when the inner optimization has converged; however, in practice it is often applied in an online fashion, when the inner parameters are far from convergence [29, 4, 30, 31].

3. Equilibrium Concepts

When the inner problem is overparameterized, there are many equally good solutions to the inner optimization, so the best response $\mathcal{S}(\boldsymbol{\lambda})$ is a set and not unique [37, 38]. Different choices of $\mathbf{y}^* \in \mathcal{S}(\mathbf{x})$ yield different best-response Jacobians $\frac{\partial \mathbf{y}^*}{\partial \mathbf{x}}$, which lead to different hypergradients, and the solution we converge to depends on the initialization of the inner parameters and the optimization algorithm used.

In the bilevel optimization literature, two specific choices of solutions have been studied, termed the *optimistic* and *pessimistic* solutions [25]. Fig. 1 illustrates these solutions for a simple 2D classification problem. Importantly, neither of these solutions is tractable to compute for large problems. We focus on two alternatives that correspond to practical applications of bilevel optimization: *cold-start* and *warm-start* optimization and their solutions (also illustrated in Fig. 1, and described in Table 1). Cold- and warm-start algorithms can be seen as different bilevel optimizers, which are necessarily higher-order and depend on efficient Jacobian-vector products to handle the response Jacobian. Overparameterization can also occur in the outer problem, such that $\arg\min_{\mathbf{x}} F(\mathbf{x}, \mathbf{y}^*)$ is not unique; in this case, we may converge to different solutions $\mathbf{x}$ depending on algorithmic choices such as the number of gradient steps we unroll, or the number of Neumann series iterations we perform to approximate the inverse Hessian.

Warm-Starting as Proximal Optimization. Given outer parameters $\mathbf{x}$ and current inner parameters $\mathbf{y}_k$, we formalize warm-started joint optimization by considering the following proximally-regularized inner objective:

$$\mathbf{y}^* \in \arg\min_{\mathbf{y}} \{f(\mathbf{x}, \mathbf{y}) + \frac{\epsilon}{2}\|\mathbf{y} - \mathbf{y}_k\|^2\} \tag{3}$$

Here, ϵ controls the amount of inner optimization we perform. In the following, we denote by $\hat{f}$ the proximally-regularized objective. One iteration of gradient descent (GD) with a small step size is approximately equivalent to minimizing the proximal objective with large ϵ [39]. Assume that after one step of GD, we are at the minimum of $\hat{f}$. Because the inner parameters $\mathbf{y}$ are at a stationary point of the gradient field $\nabla_{\mathbf{y}} \hat{f}$, the conditions needed to apply IFT are satisfied, allowing us to compute the local best-response Jacobian as: [1]

$$\frac{\partial \hat{\mathbf{y}}^*}{\partial \mathbf{x}} = -\left(\frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}^\top} + \epsilon I\right)^{-1} \frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{y}} \tag{4}$$

Farnia et al. [40] introduced a similar proximal objective in the context of GAN zero-sum sequential games, and introduced a corresponding solution concept called the *proximal*

[1] We derive the local response Jacobian in App. E.

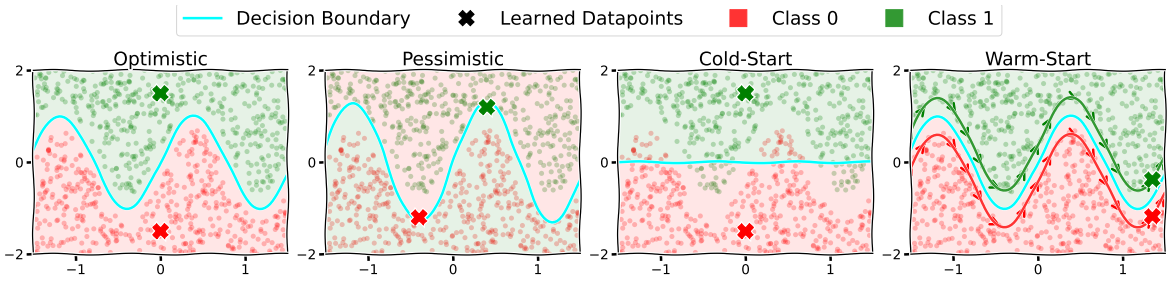


Figure 1: Handcrafted figures for a 2D data distillation task, to illustrate the differences between types of bilevel optimization (BLO). In *optimistic* BLO [25], we choose the inner parameters that achieve the *best* outer-objective value, $\arg\min_{\mathbf{y}\in\mathcal{S}(\mathbf{x})} F(\mathbf{x},\mathbf{y})$. In *pessimistic* BLO [25], we choose $\mathbf{y}\in\mathcal{S}(\mathbf{x})$ that achieves the *worst* outer-objective value, $\arg\max_{\mathbf{y}\in\mathcal{S}(\mathbf{x})} F(\mathbf{x},\mathbf{y})$. In practice, due to the implicit bias of gradient descent, the $\mathbf{y}\in\mathcal{S}(\mathbf{x})$ we end up at depends on the inner initialization $\mathbf{y}_0$: in cold-start BLO, we obtain $\mathbf{y}$ that minimize the distance from $\mathbf{y}_0$: $\arg\min_{\mathbf{y}\in\mathcal{S}(\mathbf{x})} ||\mathbf{y}-\mathbf{y}_0||_2^2$. With warm-start BLO, the trajectory of outer parameters $\mathbf{x}$ during joint optimization (shown by the arrows) influences the inner parameters $\mathbf{y}$; we investigate this behavior in Section 4.1.

equilibrium. We make use of a similar solution concept for bilevel optimization, which we call the *warm-start equilibrium*. Next, we define solution concepts of cold-start and warm-start equilibria.[2]

Definition 3.1 (Cold-Start Equilibrium)**.** $(\mathbf{x}^*,\mathbf{y}^*)$ *is a cold-start equilibrium for an initialization* $\mathbf{y}_0$ *if:*

$$\mathbf{x}^* \in \arg\min_{\mathbf{x}} F(\mathbf{x},\mathbf{y}) \quad , \quad \mathbf{y}^* \in \arg\min_{\mathbf{y}\in\mathcal{S}(\mathbf{x}^*)} ||\mathbf{y}-\mathbf{y}_0||^2$$

where $\mathcal{S}(\mathbf{x}^*) = \arg\min_{\mathbf{y}} f(\mathbf{x}^*,\mathbf{y})$.

Definition 3.2 (Warm-Start Equilibrium)**.** $(\mathbf{x}^*,\mathbf{y}^*)$ *is a* warm-start equilibrium *if:*

$$\mathbf{x}^* \in \arg\min_{\mathbf{x}} F(\mathbf{x},\mathbf{y}^*)$$
$$\mathbf{y}^* \in \arg\min_{\mathbf{y}} \{f(\mathbf{x}^*,\mathbf{y}) + \frac{\epsilon}{2}||\mathbf{y}-\mathbf{y}^*||^2\}$$

Similarly to classic results on implicit regularization of gradient descent in single-level optimization, cold-starting will lead to inner parameters with minimum distance from their initialization. In contrast, we conjecture that warm-starting yields inner parameters that aim to fit well for all the outer parameters encountered during joint optimization simultaneously (see App. D.1).

4. Implicit Regularization in Bilevel Opt.

In this section, we introduce two toy problems based on online dataset distillation, which involve jointly optimizing a model and the data it is trained on. First, we show that when the inner problem is overparameterized, the inner parameters can retain information associated with different settings of the outer parameters seen over the course of joint optimization. Then, we show that when the outer problem is overparameterized, the choice of hypergradient approximation can affect which outer solution is found. Experimental details and extended results are provided in App. D.

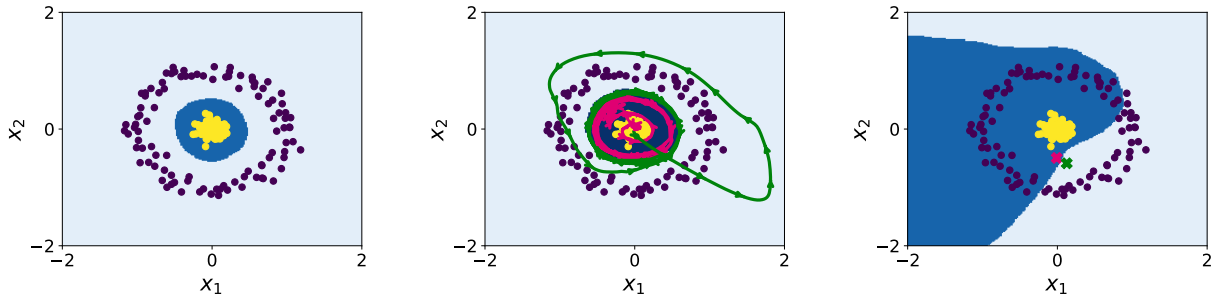
[2]We discuss warm-start further in App. C.

4.1. Inner Overparameterization: Dataset Distillation

Because the outer objective is only used directly to update the outer variables, it would seem intuitive that all of the information about the outer objective is compressed into the outer variables. This is the intuition behind dataset distillation [6], which aims to learn a *synthetic dataset*, such that a model trained from scratch on the learned data generalizes well to the original data (which is the "validation set" in the bilevel problem). Since the original dataset is only used in the outer objective, one would expect it to be summarized by the outer variables, e.g., the lower-dimensional distilled dataset. While this intuition is correct for the cold-start equilibrium, it does not hold for the warm-start equilibrium: a surprisingly large amount of information can leak from the original dataset to the inner variables (network weights). Consider a binary classification task with inputs $\mathbf{x}\in\mathbb{R}^2$, where the classes form concentric rings in 2D space (Fig. 2a). We aim to learn *two distilled datapoints, one per class*, to train an MLP. With alternating updates to the MLP parameters and the learned datapoints, the datapoints follow a nontrivial *trajectory*, tracing out the decision boundary between classes over time (Fig. 2b). **Result**: Warm-started bilevel optimization gives a model that achieves nearly the same validation loss as one trained directly on the validation data, despite only using a single datapoint from each class. In contrast, a cold-started model trained from scratch on the final distilled datapoints fails on the original dataset (Fig. 2c).

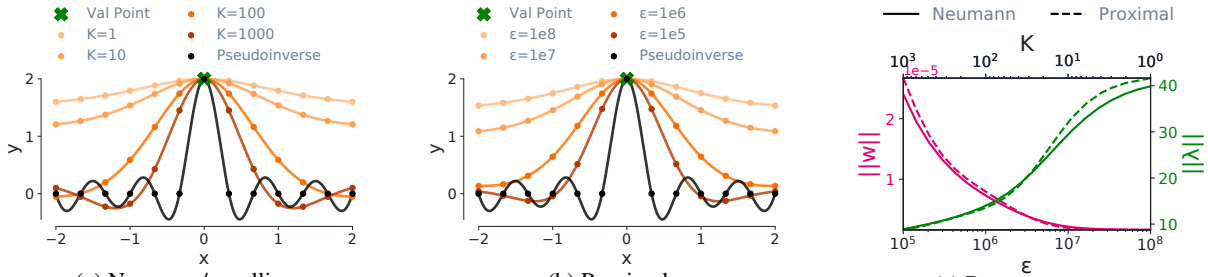
4.2. Outer Overparameterization: Anti-Distillation

Next, we consider a scenario where the *outer problem* is overparameterized. We propose a new task related to dataset distillation, but where we have *more* learned datapoints than original dataset examples, which we term *anti-distillation*. Here, there are many valid ways to set the learned datapoints such that a model trained on those points achieves good performance on the original data. Concretely, consider a 1D linear regression problem, where $\mathbf{\Phi}$ is a de-



(a) Training on original data. (b) Warm-start: joint optimization. (c) Re-train on final points

Figure 2: Dataset distillation for binary classification, with two learned datapoints (outer parameters) adapted jointly with the model weights (inner parameters). **(a)** Reference performance when training a model on the original data (e.g., the "validation data"); **(b)** Joint optimization (e.g., warm-starting) yields a *trajectory* of the learned datapoints that traces out the boundary between the classes, and the final model obtains low validation loss; **(c)** Training from scratch on the final learned datapoints yields a decision boundary that is correct for the distilled points but has high validation loss.



(a) Neumann/unrolling (b) Proximal (c) Parameter norms

Figure 3: Fourier-basis 1D linear regression, where the outer objective is overparameterized. We learn the y-component of 13 synthetic datapoints such that a regressor trained on those points will fit a single "validation" datapoint, shown by the green $\mathbf{x}$ at $(0, 2)$. Figs. **(a,b)** show the learned datapoints (outer parameters) obtained via different hypergradient approximations: truncated Neumann and differentiation through truncated unrolls have the same fixed points because this problem is convex; we also perform proximal optimization analytically, using the closed-form proximal pseudoinverse. Fig **(c)** shows the norms of the inner and outer parameters, $||\mathbf{w}||$ and $||\boldsymbol{\lambda}||$ respectively, as a function of K (for Neumann/unrolling) or ϵ (for proximal). **Result:** The amount of inner optimization we perform affects the trade-off between the norms of the inner and outer parameters (fewer unrolling steps or a larger ϵ yield smaller-norm inner parameters and larger-norm outer parameters).

sign matrix, $\mathbf{w}$ is a vector of weights, and $\mathbf{y}$ is a vector of targets (or labels): $L(\mathbf{w}, \mathbf{y}) = \frac{1}{2}||\boldsymbol{\Phi}\mathbf{w} - \mathbf{y}||_2^2$. Here, $\mathbf{w}$ are the inner parameters and $\mathbf{y}$ are the outer parameters. To keep the exposition simple, we only learn the targets $\mathbf{y}$, not the inputs. The gradient of L is $\frac{\partial L(\mathbf{w},\mathbf{y})}{\partial \mathbf{w}} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi}\mathbf{w} - \boldsymbol{\Phi}^\top\mathbf{y}$, and the Hessian and second-order mixed partials are $\frac{\partial^2 L(\mathbf{w},\mathbf{y})}{\partial \mathbf{y}\partial \mathbf{w}} = \boldsymbol{\Phi}^\top$ and $\frac{\partial^2 L(\mathbf{w},\mathbf{y})}{\partial \mathbf{w}\partial \mathbf{w}^\top} = \mathbf{H} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi}$, respectively. The minimum-norm best-response Jacobian, is $\frac{\partial \mathbf{w}^*(\mathbf{y})}{\partial \mathbf{y}} = \arg\min_{\mathbf{HM}=\boldsymbol{\Phi}} ||\mathbf{M}||_F^2 = \boldsymbol{\Phi}^+$, which is the Moore-Penrose pseudoinverse. Now, we design a specific instance of this problem, and illustrate empirically the effect of different hypergradient approximations on the converged solution (Fig. 3). Here, we have one validation data point, so any solution that places a learned datapoint on top of the validation point perfectly fits the outer objective. To demonstrate how different hypergradient approximations can lead to different anti-distillations, we use Fourier-basis regression, where the low frequency terms have larger amplitude than high frequency terms (details in App. D.2). In Fig. 3, we show the solutions obtained via different numbers of Neumann series terms, proximal optimization with various ϵ values, and unrolling for different number of iterations.

We found that the quality of hypergradient approximations directly induces a trade-off between the inner and outer parameter norms—e.g., we can achieve the good performance for the outer objective by either making larger updates to the inner or the outer parameters.

5. Conclusion

We investigated overparameterized bilevel optimization, where either the inner or outer problems may admit non-unique solutions. We discussed different equilibrium notions, including cold-start and warm-start, corresponding to different common optimization methods. We showed that these choices can dramatically affect results when combined with overparameterization. We presented tasks illustrating that these choices can significantly affect our solutions in practice. More generally, we hope that we have highlighted the importance of and laid the groundwork for analyzing the effects of overparameterization in nested optimization problems.

Acknowledgements

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

References

[1] Justin Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pages 318–326, 2012.

[2] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning (ICML)*, pages 2113–2122, 2015.

[3] Matthew MacKay, Paul Vicol, Jon Lorraine, David Duvenaud, and Roger Grosse. Self-Tuning Networks: Bilevel optimization of hyperparameters using structured best-response functions. In *International Conference on Learning Representations (ICLR)*, 2019.

[4] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1540–1552, 2020.

[5] Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1723–1732, 2019.

[6] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

[7] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021.

[8] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, pages 1885–1894, 2017.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pages 1126–1135, 2017.

[10] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimilano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning (ICML)*, 2018.

[11] Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[12] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, pages 41–48, 2009.

[13] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning (ICML)*, pages 4334–4343, 2018.

[14] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[15] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.

[16] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 157–163. 1994.

[17] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[18] David Pfau and Oriol Vinyals. Connecting generative adversarial networks and actor-critic methods. In *NeurIPS Workshop on Adversarial Training*, 2016.

[19] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[20] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural Computation*, 12(8):1889–1900, 2000.

[21] Jan Larsen, Lars Kai Hansen, Claus Svarer, and M Ohlsson. Design and regularization of neural networks: The optimal use of a validation set. In *IEEE Signal Processing Society Workshop*, pages 62–71, 1996.

[22] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International Conference on Machine Learning (ICML)*, pages 2952–2960, 2016.

[23] Juhan Bae and Roger Grosse. Delta-STN: Efficient bilevel optimization for neural networks using struc-

tured response Jacobians. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[24] Zhiqiang Tang, Yunhe Gao, Leonid Karlinsky, Prasanna Sattigeri, Rogerio Feris, and Dimitris Metaxas. OnlineAugment: Online data augmentation with less domain knowledge. In *ECCV*, 2020.

[25] Stephan Dempe. *Bilevel Optimization: Theory, Algorithms and Applications*. TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik, 2018.

[26] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning (ICML)*, 2017.

[27] Dingding Chen and Martin T Hagan. Optimal use of regularization and cross-validation in neural network modeling. In *International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 1275–1280, 1999.

[28] Chuan-sheng Foo, Chuong B Do, and Andrew Y Ng. Efficient multiple hyperparameter learning for log-linear models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 377–384, 2008.

[29] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning (ICML)*, 2016.

[30] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Meta approach to data augmentation optimization. *arXiv preprint arXiv:2006.07965*, 2020.

[31] Aniruddh Raghu, Maithra Raghu, Simon Kornblith, David Duvenaud, and Geoffrey Hinton. Teaching with commentaries. In *International Conference on Learning Representations (ICLR)*, 2020.

[32] Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, KiJung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard Zemel. Reviving and improving recurrent back-propagation. In *International Conference on Machine Learning (ICML)*, 2018.

[33] Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

[34] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[35] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[36] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[37] Risheng Liu, Pan Mu, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton. In *International Conference on Machine Learning (ICML)*, pages 6305–6315, 2020.

[38] Risheng Liu, Jiaxin Gao, Jin Zhang, Deyu Meng, and Zhouchen Lin. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *arXiv preprint arXiv:2101.11517*, 2021.

[39] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[40] Farzan Farnia and Asuman Ozdaglar. Do GANs always have Nash equilibria? In *International Conference on Machine Learning (ICML)*, 2020.

[41] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[42] Jascha Sohl-Dickstein, Roman Novak, Samuel S Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *arXiv preprint arXiv:2001.07301*, 2020.

[43] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[44] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt.

In *International Conference on Learning Representations (ICLR)*, 2020.

[45] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32): 15849–15854, 2019.

[46] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2016.

[47] Yogesh Balaji, Mohammadmahdi Sajedi, Neha Mukund Kalibhat, Mucong Ding, Dominik Stöger, Mahdi Soltanolkotabi, and Soheil Feizi. Understanding overparameterization in generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[48] Jean-Yves Franceschi, Emmanuel de Bézenac, Ibrahim Ayed, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. A neural tangent kernel perspective of gans. *arXiv preprint arXiv:2106.05566*, 2021.

[49] Otto Neall Strand. Theory and methods related to the singular-function expansion and Landweber's iteration for integral equations of the first kind. *SIAM Journal on Numerical Analysis*, 11(4):798–825, 1974.

[50] Nelson Morgan and Hervé Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. *Advances in Neural Information Processing Systems (NeurIPS)*, 2:630–637, 1989.

[51] Jerome Friedman and Bogdan E Popescu. Gradient directed regularization for linear regression and classification. Technical report, Technical Report, Statistics Department, Stanford University, 2003.

[52] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[53] Mor Shpigel Nacson, Nathan Srebro, and Daniel Soudry. Stochastic gradient descent on separable data: Exact convergence with a fixed learning rate. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3051–3059, 2019.

[54] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[55] Arun Suggala, Adarsh Prasad, and Pradeep K Ravikumar. Connecting optimization and regularization paths. *Advances in Neural Information Processing Systems (NeurIPS)*, 31:10608–10619, 2018.

[56] Tomaso Poggio, Andrzej Banburski, and Qianli Liao. Theoretical issues in deep networks: Approximation, optimization and generalization. In *Proceedings of the National Academy of Sciences*, 2019.

[57] Ziwei Ji and Matus Telgarsky. The implicit bias of gradient descent on nonseparable data. In *Conference on Learning Theory*, pages 1772–1798, 2019.

[58] Alnur Ali, J Zico Kolter, and Ryan J Tibshirani. A continuous-time view of early stopping for least squares regression. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1370–1378, 2019.

[59] Alnur Ali, Edgar Dobriban, and Ryan Tibshirani. The implicit regularization of stochastic gradient flow for least squares. In *International Conference on Machine Learning (ICML)*, pages 233–244, 2020.

[60] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. 2017.

[61] Jonathan Lorraine and David Duvenaud. Stochastic hyperparameter optimization through hypernetworks. In *NIPS Meta-Learning Workshop*, 2017.

[62] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster autoaugment: Learning augmentation strategies using backpropagation. In *European Conference on Computer Vision (ECCV)*, pages 1–16, 2020.

[63] D. Ponsa E. Rublee E. Riba, D. Mishkin and G. Bradski. Kornia: An open source differentiable computer vision library for PyTorch. In *Winter Conference on Applications of Computer Vision*, 2020. URL https://arxiv.org/pdf/1910.02190.pdf.

[64] Xi Peng, Zhiqiang Tang, Fei Yang, Rogerio S Feris, and Dimitris Metaxas. Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2226–2234, 2018.

[65] Saypraseuth Mounsaveng, Issam Laradji, Ismail Ben Ayed, David Vazquez, and Marco Pedersoli. Learning data augmentation with online bilevel optimization for image classification. In *Winter Conference on Applications of Computer Vision*, pages 1691–1700, 2021.

[66] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning (ICML)*, pages 2731–2741, 2019.

[67] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi,

Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[68] Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

[69] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[70] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8697–8710, 2018.

[71] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[72] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning (ICML)*, pages 4095–4104, 2018.

[73] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning (ICML)*, pages 550–559, 2018.

[74] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[75] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 4780–4789, 2019.

[76] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations (ICLR)*, 2019.

[77] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.

[78] Ilia Sucholutsky and Matthias Schonlau. 'less than one'-shot learning: Learning n classes from m< n samples. *arXiv preprint arXiv:2009.08449*, 2020.

[79] Lillian Ratliff. An introduction to learning in games. *Lecture Notes*, 2021.

[80] S Karczmarz. Angenaherte auflosung von systemen linearer gleichungen. *Bull. Int. Acad. Pol. Sic. Let., Cl. Sci. Math. Nat.*, pages 355–357, 1937.

[81] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2020.

[82] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E Turner, and Mohammad Emtiyaz Khan. Continual deep learning by functional regularisation of memorable past. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[83] Quang Pham, Doyen Sahoo, Chenghao Liu, and Steven CH Hoi. Bilevel continual learning. *arXiv preprint arXiv:2007.15553*, 2020.

Appendix

This appendix is structured as follows:

- In Section A we provide an overview of related work.
- In Section B we provide an overview of the notation we use throughout the paper.
- In Section C we discuss different warm-start algorithms.
- In Section D we provide experimental details and extended results.
- In Section E we provide derivations of formulas used in the main body.
- In Section F we describe the algorithms for unrolling and proximal hyperparameter optimization.

A. Related Work

Overparameterization. Overparameterization has long been studied in single-level optimization, generating key insights such as neural network behavior in the infinite-width limit [41–43], double descent phenomena [44, 45], the ability to fit random labels [46], and studying inductive biases of optimizers. However, little attention has been paid to the study of overparameterization in the bilevel setting; the only work we are aware of in this area is on overparameterized GANs [47] and on using the NTK to understand GANs [48].

Implicit Regularization. The effect of training dynamics leading to certain minima rather than others is known as implicit regularization. Implicit regularization has a long history in machine learning: many works have observed and studied the connection between early stopping and L_2 regularization [49–52]. Interest in implicit regularization has increased over the past few years [53–57]. In the case of L_2-regularized linear regression, Ali et al. [58, 59] have shown close connections between the solutions (and solution paths) obtained via gradient flow (continuous-time optimization) starting from the origin, and via the closed-form solution to the L_2 regularized problem.

Hyperparameter Optimization (HO). There are three main approaches for gradient-based HO: 1) differentiating through unrolls of the inner problem, sometimes called *iterative differentiation* [1, 2, 5]; 2) using *implicit differentiation* to compute the best-response Jacobian assuming that the inner optimization has converged [21, 20, 28, 29]; and 3) using a *hypernetwork* [60] to approximate the best-response function locally, $\hat{\mathbf{w}}^*_\phi \approx \mathbf{w}^*$, such that the outer gradient can be computed using the chain rule through the hypernetwork, $\frac{\partial \mathcal{L}_V}{\partial \boldsymbol{\lambda}} = \frac{\partial \mathcal{L}_V}{\partial \hat{\mathbf{w}}_\phi(\boldsymbol{\lambda})} \frac{\partial \hat{\mathbf{w}}_\phi(\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}}$. Hypernetworks have been applied to HO in [61, 3, 23]. [3, 23] have observed that STNs learn online hyperparameter schedules (e.g., for dropout rates and augmentations) that can outperform any fixed hyperparameter value. We believe warm-start effects at least partially explain the observed improvements from hyperparameter schedules.

Data Augmentation. A special case of hyperparameter optimization that has received widespread attention is automatic data augmentation [30, 62–65], where the aim is either to learn the strengths with which to apply different augmentations, or an *augmentation network* that takes an input image and potentially a source of random noise, and outputs an augmented example [4, 24]. The former approach typically involves tens to hundreds of outer parameters (the coefficients of pre-specified augmentations), while the latter approach involves millions of outer parameters (the weights of the augmentation network). The Population-Based Augmentation (PBA) algorithm [66] searches for augmentation schedules using Population-Based Training [67]; the authors found that training with the PBA schedule outperformed using the fixed final hyperparameters or training with a shuffled schedule (e.g., using the magnitudes of augmentations from the schedule, but in a random order). Because augmentations directly modify the data input to the model, tuning augmentations is conceptually related to curriculum learning [68, 12]. Another form of curriculum learning is in techniques that automatically reweight examples during training [13].

Neural Architecture Search (NAS). NAS can also be considered a special case of HO, where the search space is generally discrete [14, 69–73]. Most approaches to NAS are based on reinforcement learning [14] or evolutionary algorithms [74, 75], while DARTS [15], ProxylessNAS [76], and Stochastic NAS [77] are gradient-based approaches.

Data Distillation. Maclaurin et al. [2] were among the first to consider learning a training dataset in a bilevel formulation. More recent work includes [6, 7, 78].

Games. Many problems in machine learning are instances of *games*, including GANs, adversarial training, actor-critic RL, and hyperparameter optimization. Games come in different varieties: the order of play can be simultaneous or sequential, and the game can be zero-sum (which captures purely competitive or adversarial games) or general-sum (which can capture aspects of both competition and cooperation) [79]. The original formulation of GAN training [17] is a simultaneous zero-sum game (also referred to as a minimax game); HO is a sequential general-sum game. Bilevel problems are hierarchical, involving a specific sequential order of play; such problems are called Stackelberg games. Different solution concepts exist for different types of games: for zero-sum simultaneous games, we seek Nash equilibria; for general-sum Stackelberg games, we seek Stackelberg equilibria. Farnia and Ozdaglar introduced a solution concept for zero-sum sequential games called proximal equilibrium [40], which interpolates between Nash and Stackelberg equilibria.

B. Notation

F	Outer objective
f	Inner objective
$\mathbf{x}$	Outer variables
$\mathbf{y}$	Inner variables
$\mathbf{y}^*$	Best-response
$\frac{\partial \mathbf{y}^*}{\partial \mathbf{x}}$	Best-response Jacobian
$\mathcal{S}(\mathbf{x}) = \arg\min_{\mathbf{y}} f(\mathbf{y}, \mathbf{x})$	Best-response set for $\mathbf{x}$
$\mathbf{w}$	Model weights
$\mathbf{\Phi}$	Design matrix, where each row corresponds to an example
$\|\| \cdot \|\|_F^2$	(Squared) Frobenius norm
ϵ	Weighting of the proximal regularizer
k	Number of unrolling iterations / Neumann steps
HO	Hyperparameter optimization

Table 2: Summary of the notation used in this paper

C. Warm-Start Concepts

Warm-starting refers to initializing the inner optimization from the inner parameters obtained in the previous hypergradient computation (e.g., the previous iteration of joint bilevel optimization). One can consider different warm-start algorithms: (1) using *full inner optimization*, that is, running the inner optimization to convergence starting from $\mathbf{y}_t$ to obtain the next iterate $\mathbf{y}_{t+1}$; or (2) *partial inner optimization*, where we approximate the solution to the inner problem via a small number of gradient steps (e.g., a truncated unroll). Similarly to cold-starting, Approach (1) is computationally expensive, so we focus on Approach (2). We formalized Approach (2) using proximal optimization in Sec. 3 of the main paper. Approach (1) can be formalized as finding the projection of the current iterate $\mathbf{y}_t$ onto the solution set corresponding to an outer parameter $\mathbf{x}$, which we denote $\Pi(\mathbf{y}_t, \mathcal{S}(\mathbf{x})) = \arg\min_{\mathbf{y} \in \mathcal{S}_\mathbf{x}} ||\mathbf{y} - \mathbf{y}_t||^2$. Figure 4 illustrates both types of warm-starts (full and approximate inner optimization), and contrasts them to cold-starting, on a toy linear regression problem where we can visualize the iterates of each algorithm.

D. Experimental Details and Extended Results

Compute Environment. All experiments were implemented using JAX [36], and were run on NVIDIA P100 GPUs. Each instance of the dataset distillation and antidistillation task took approximately 5 minutes of compute on a single GPU.

D.1. Details and Extended Results for Dataset Distillation.

Intuition for the Observed Warm-Start Behavior. Figure 4 illustrates the difference between warm-start and cold-start optimization, on a toy linear regression example where we can visualize the steps of each algorithm in the inner parameter space. We consider a single learned datapoint that is constrained to move along a line in data-space. Because we have a single datapoint parameterized by one coordinate (e.g., the outer parameters have dimension 1) and two weights (e.g., the inner parameters have dimension 2), the inner problem is overparameterized: for each setting of the datapoint, we have a set of feasible solutions. The solution sets for four fixed values of the datapoint are shown by the solid black lines in Figure 4. The validation loss contours are shown in the background.

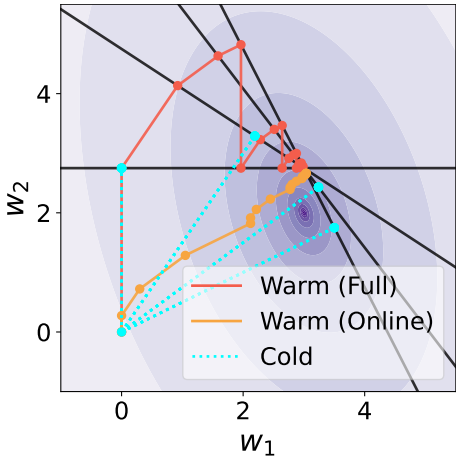


Figure 4: Parameter-space view of warm-start with full inner optimization, warm-start with partial inner optimization (denoted the "online" setting, which most closely resembles what is done in practice), and cold-start optimization.

Here, we can exactly characterize each step of the cold-start and warm-start algorithms (we assume the inner parameters are initialized at the origin): cold-start always projects from the origin onto the solution set for the current datapoint, while warm-start projects from the current weights onto the solution set.

By projecting successively from one solution set to the next, warm-start in this toy setting closely resembles Kaczmarz's algorithm [80]. If we cycle through the solution sets repeatedly, the inner parameters will converge to the intersection point of the solution sets, in effect yielding inner parameters that perform well for multiple outer parameters simultaneously. Note that this procedure does not necessarily converge to the optimal validation loss, but rather to the intersection of the solution sets (or, if the solution sets do not have a single intersection point, we conjecture that in each iteration the inner parameters will decrease their distance to the convex hull containing the intersections of the solution sets).

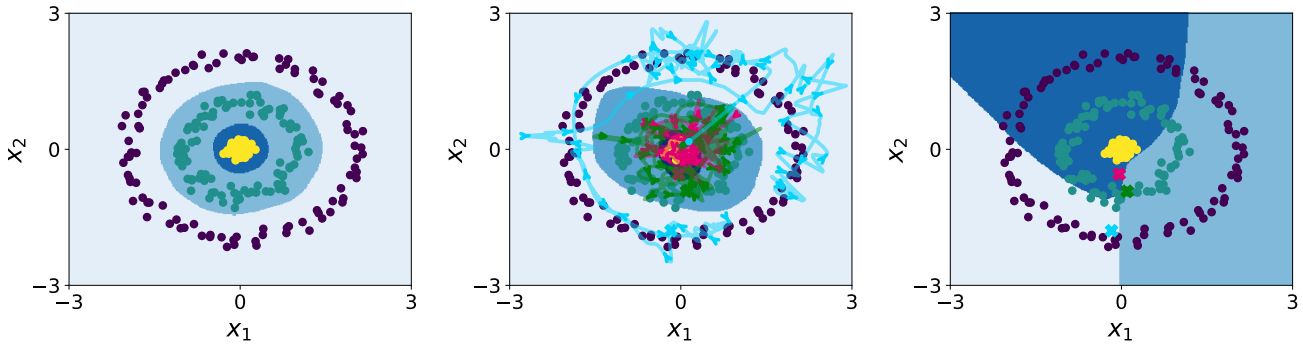
Details. We generated datasets with $N = 100$ datapoints per class, and we trained a 4-layer MLP with 200 hidden units per layer and ReLU activations. For warm-start joint optimization, we computed hypergradients by differentiating through $K = 1$ steps of unrolling, and updated the hyperparameters (learned datapoints) and MLP parameters using alternating gradient descent, with one step on each. We used SGD with learning rate 0.001 for the inner optimization and Adam with learning rate 0.01 for the outer optimization.

Relation to Continual Learning. An overparameterized model is able to learn the correct function behavior for part of the true function illustrated by the learned datapoints, and the proximal regularizer prevents catastrophic forgetting of the function fit to previous values of the learned datapoints. This bears conceptual similarities to methods for continual learning [81–83].

Extended Results. Here, we show additional dataset distillation results, using a similar setup to Section 4.1. Figure 5 shows the results where we fit a three-class problem (e.g., three concentric rings) using three learned datapoints. Figure 6 shows the results for fitting three classes using *only two datapoints*, where both the coordinates and *soft labels* are learned. For each training datapoint, in addition to learning its x- an y-coordinates, we learn a C-dimensional vector (where C is the number of classes, in this example $C = 3$) representing the unnormalized class label: this vector is normalized with a softmax when we perform cross-entropy training of the inner model (e.g., we do not train on one-hot labels). Joint adaptation of the model parameters and learned data is able to fit three classes by changing the learned class label for one of the datapoints during training.

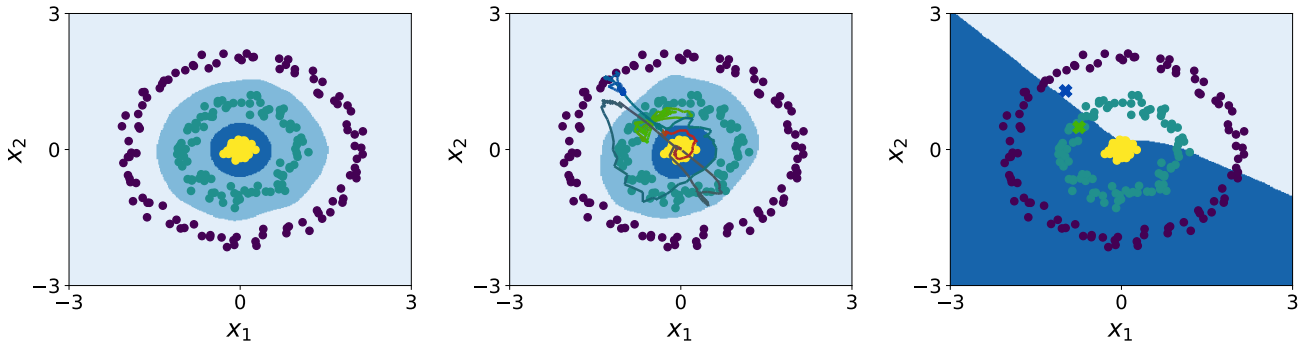
D.2. Details and Extended Results for Anti-Distillation.

Details. For the anti-distillation results in Section 4.2, the synthetic datapoints are initialized at linearly-spaced x-coordinates, with y-coordinate 0, and we only learn the targets $\mathbf{y}$. In the Fourier basis we use, lower frequency components have larger amplitudes. In Section 4.2, we used the following feature function: $\phi(x) = a_0 + \sum_{n=1}^{N} \left(a_n 2^{N-n} \cos(nx) + b_n 2^{N-n} \sin(nx)\right)$. Our Fourier basis functions consisted of 10 sin terms, 10 cos terms, and a bias, yielding 21 total inner parameters. For the exponential-amplitude Fourier basis used in 4.2, we used SGD with learning rates 1e-8 and 1e-2 for the inner and outer parameters, respectively; for the $1/n$ amplitude Fourier basis (discussed below, and used for Figure 7), we used SGD with learning rates 1e-3 and 1e-2 for the inner and outer parameters, respectively.



(a) Train on original data. (b) Warm-start joint optimization. (c) Re-training on final points.

Figure 5: Dataset distillation results fitting three classes with three learned datapoints. Similarly to the results in Sec. 4.1, when using warm-start joint optimization, the three learned datapoints are adapted during training to trace out the data in their respective classes, guiding the network to learn a decision boundary that performs well on the original data. Cold-start re-training yields a model that correctly classifies the three learned datapoints, but has poor validation performance.



(a) Training on original data. (b) Warm-start joint optimization. (c) Re-training on final points.

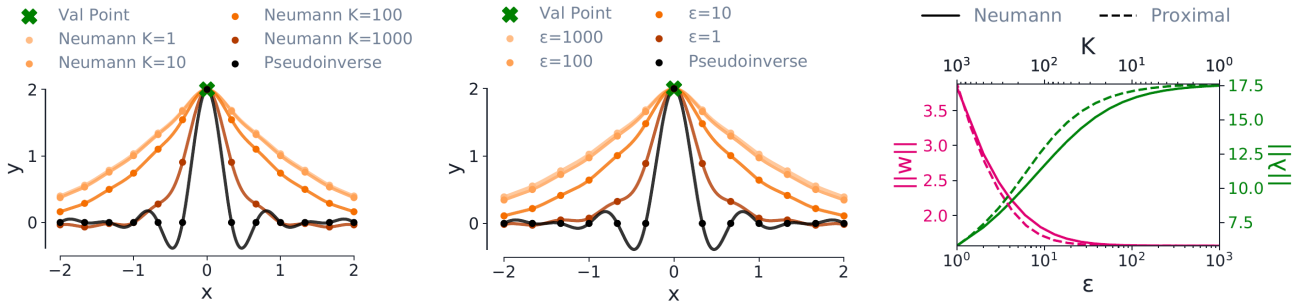
Figure 6: Dataset distillation results fitting three classes with *only two* learned datapoints. In the warm-start plot, the color of the trajectory of each learned datapoint indicates its soft class membership, with magenta, green, and blue corresponding to the inner, middle, and outer rings, respectively. Darker/gray colors indicate soft labels that place approximately equal probability on each class. We see that although we only have two learned datapoints, the class labels change over the course of training such that all three classes are covered. Cold-start re-training yields a model that correctly classifies the three learned datapoints, but has poor validation performance.

An Alternate Set of Fourier Basis Functions. In Section 4.2, we used a Fourier basis in which the lower-frequency terms had exponentially larger amplitudes than the high-frequency terms. Figure 7 presents results using an alternative feature function: $\phi(x) = a_0 + \sum_{n=1}^{N} \left(a_n \left(\frac{1}{n}\right) \cos(nx) + b_n \left(\frac{1}{n}\right) \sin(nx)\right)$.

Using an MLP. Finally, we show that similar conclusions hold when training a multi-layer perceptron (MLP) on the anti-distillation task. We used a 2-layer MLP with 10 hidden units per layer and ReLU activations. We used SGD with learning rate 0.01 for both the inner and outer parameters. Figure 8 shows the learned datapoints and model fits resulting from running several different steps of Neumann iterations or unrolling, as well as the norms of the inner and outer parameters as a function of K. For the Neumann experiments, we first optimize the MLP for 5000 steps to reach approximate convergence of the inner problem, before running K Neumann iterations—the MLP is re-trained from scratch for 5000 steps for each outer parameter update (e.g., cold-started).

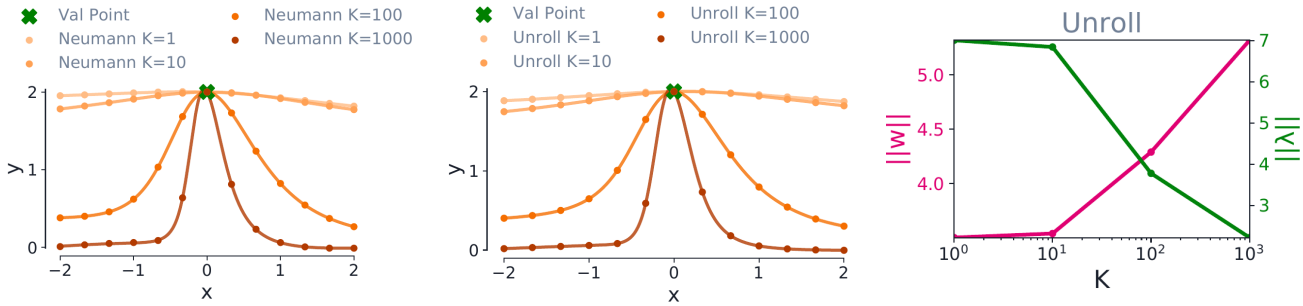
E. Derivations

In this section, we provide a derivation for the local best-response Jacobian used in Section 3, and we review the formula (used in the Kaczmarz algorithm) for closed-form projections onto solution sets given by hyperparameters, used for Figure 4.



(a) Neumann/unroll (b) Proximal (c) Parameter norms.

Figure 7: Fourier-basis 1D linear regression, where the outer objective is overparameterized. We learn 13 synthetic datapoints such that a regressor trained on those points will fit a single "validation" datapoint, shown by the green **x** at $(0, 2)$. The synthetic datapoints are initialized at linearly-spaced x-coordinates, with y-coordinate 0, and we only learn the targets $\mathbf{y}$. In the Fourier basis we use, lower frequency components have larger amplitudes. Here, we use the $1/n$ amplitude scheme.



(a) Neumann (b) Unroll (c) Unroll parameter norms

Figure 8: 1D linear regression with an overparameterized outer objective, where we train an MLP rather than performing Fourier-basis function regression. Note that here we cannot analytically compute the proximal solution as in the linear regression case, so we only include results for full-unrolls with truncated Neumann approximations of the inverse Hessian, and alternating gradient descent with various numbers of unroll steps.

E.1. Proximal Best-Response

Consider the proximal objective $\hat{f}(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}, \mathbf{y}) + \frac{\epsilon}{2}||\mathbf{y} - \mathbf{y}_k||^2$. Here we will treat $\mathbf{y}_k$ as a constant (so we won't consider its dependence on $\mathbf{x}$). Let $\mathbf{y}^*(\mathbf{x}) \in \arg\min_{\mathbf{y}} \hat{f}(\mathbf{x}, \mathbf{y})$ be a fixed point of $\hat{f}$. We want to compute the response Jacobian $\frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}}$. Since $\mathbf{y}^*$ is a fixed point, we have:

$$\frac{\partial \hat{f}(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}} = 0 \tag{5}$$

$$\frac{\partial f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}} + \epsilon(\mathbf{y}^*(\mathbf{x}) - \mathbf{y}_k) = 0 \tag{6}$$

$$\frac{\partial}{\partial \mathbf{x}} \frac{\partial f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}} + \epsilon \frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}} = 0 \tag{7}$$

$$\frac{\partial^2 f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{x} \partial \mathbf{y}} + \frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}^\top} \frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}} + \epsilon \frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}} = 0 \tag{8}$$

$$\left(\frac{\partial^2 f}{\partial \mathbf{y}^2} + \epsilon I\right) \frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}} = -\frac{\partial^2 f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{x} \partial \mathbf{y}} \tag{9}$$

$$\frac{\partial \mathbf{y}^*}{\partial \mathbf{x}} = -\left(\frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}^\top} + \epsilon I\right)^{-1} \frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{y}} \tag{10}$$

This is the similar to the hypergradient given a solution to the full (non-regularized) inner optimization, except with the addition of the ϵI term. With large enough ϵ, this should ensure that the matrix we need to invert, $\left(\frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}^\top} + \epsilon I\right)$ is PSD, and thus would allow us to use conjugate gradient in a principled manner.

E.2. Pseudoinverse as Min-Norm Solution

We need a solution to the linear system $\mathbf{HM} = \mathbf{\Phi}$. Assuming this system is satisfiable, the matrix $\mathbf{Q} = \mathbf{H}^+\mathbf{\Phi}$ is a solution that satisfies $||\mathbf{Q}||_F^2 \leq ||\mathbf{M}||_F^2$ for any matrix $\mathbf{M}$:

$$\mathbf{Q} = \mathbf{H}^+\mathbf{\Phi} = (\mathbf{\Phi}^\top\mathbf{\Phi})^+\mathbf{\Phi} = (\mathbf{\Phi}^\top\mathbf{\Phi})^{-1}\mathbf{\Phi} = \mathbf{\Phi}^+ \tag{11}$$

Thus the min-norm best-response Jacobian is the Moore-Penrose pseudoinverse of the feature matrix, $\mathbf{\Phi}^+$.

F. Algorithms

Algorithm 1 Differentiating through K-step unrolling

Input: K, number of unroll steps
α, outer learning rate
η, inner learning rate
Initialize $\mathbf{y}, \mathbf{x}$
repeat
  $\mathbf{x} \leftarrow \mathbf{x} - \alpha\nabla_{\mathbf{x}} F(\mathbf{x}, \text{Unroll}(\mathbf{y}, \mathbf{x}, K))$
  $\mathbf{y} \leftarrow \mathbf{y} - \eta\nabla_{\mathbf{y}} f(\mathbf{y}, \mathbf{x})$

Algorithm 2 Unroll

Input: $\mathbf{y}$, initial model parameters
$\mathbf{x}$, hyperparameters
η, learning rate
K, number of unroll steps
for $i = 1, \ldots, K$ **do**
  $\mathbf{y} \leftarrow \mathbf{y} - \eta\nabla_{\mathbf{y}} f(\mathbf{y}, \mathbf{x})$
end for
return $\mathbf{y}$ =0

Algorithm 3 Proximal hyperparameter optimization — See Eq. 3

Initialize $\mathbf{y}, \mathbf{x}$
α, outer learning rate
repeat
  $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \alpha\nabla_{\mathbf{x}} F(\mathbf{x}_t, \mathbf{y}_t^*)$
  $\mathbf{y}_{t+1}^* \in \arg\min_{\mathbf{y}}\{f(\mathbf{x}_{t+1}, \mathbf{y}) + \frac{\epsilon}{2}||\mathbf{y} - \mathbf{y}_t||^2\}$