Journal of Machine Learning Research 25 (2024) 1-46 Submitted 9/23; Revised 3/24; Published 5/24

Learning to Warm-Start Fixed-Point Optimization Algorithms

Rajiv Sambharya RAJIVS@PRINCETON.EDU
Operations Research and Financial Engineering, Princeton University, Princeton, NJ, USA

Georgina Hall GEORGINA.HALL@INSEAD.EDU
Decision Sciences, INSEAD, Fontainebleau, France

Brandon Amos BDA@META.COM
Meta AI, New York City, NY, USA

Bartolomeo Stellato BSTELLATO@PRINCETON.EDU
Operations Research and Financial Engineering, Princeton University, Princeton, NJ, USA

Editor: Quentin Berthet

Abstract

We introduce a machine-learning framework to warm-start fixed-point optimization algorithms. Our architecture consists of a neural network mapping problem parameters to warm starts, followed by a predefined number of fixed-point iterations. We propose two loss functions designed to either minimize the fixed-point residual or the distance to a ground truth solution. In this way, the neural network predicts warm starts with the end-to-end goal of minimizing the downstream loss. An important feature of our architecture is its flexibility, in that it can predict a warm start for fixed-point algorithms run for any number of steps, without being limited to the number of steps it has been trained on. We provide PAC-Bayes generalization bounds on unseen data for common classes of fixed-point operators: contractive, linearly convergent, and averaged. Applying this framework to well-known applications in control, statistics, and signal processing, we observe a significant reduction in the number of iterations and solution time required to solve these problems, through learned warm starts.

Keywords: learning to optimize, fixed-point problems, warm start, generalization bounds, parametric convex optimization.

1. Introduction

We consider *parametric fixed-point problems* of the form

$$\text{find } z \quad \text{such that} \quad z = T_\theta(z), \tag{1}$$

where $z \in \mathbf{R}^p$ is the decision variable and $\theta \in \Theta \subseteq \mathbf{R}^d$ is the *problem parameter* defining each instance of (1) via the *fixed-point operator* T_θ. We assume that θ is drawn from an unknown distribution Q, accessible only via samples, and that for every $\theta \in \Theta$, problem (1) is solvable (*i.e.*, T_θ admits a fixed-point). Our focus is on the setting where the fixed-point problem (1) represents the optimality conditions of a parametric convex optimization problem. In this setting, solving (1) amounts to solving a convex optimization problem. As it turns out, almost all convex optimization problems can be represented as finding the fixed point of an operator (Ryu and Yin, 2022; Garstka et al., 2019; O'Donoghue et al., 2019; Stellato et al., 2020), which makes the techniques we discuss in this paper broadly applicable. To solve problem (1), we repeatedly apply the operator T_θ, obtaining the iterations

$$z^{i+1} = T_\theta(z^i). \tag{2}$$

We assume that the iterations (2) converge to a fixed-point, that is, $\lim_{i\to\infty} \|z^i - z^\star(\theta)\| = 0$, where $z^\star(\theta)$ is a fixed point of T_θ. In practice, it is common to return an ϵ-approximate solution, corresponding to a vector z^i for which the *fixed-point residual*, $\|T_\theta(z^i) - z^i\|_2$ is below ϵ. Many optimization algorithms correspond to fixed-point iterations of the form (2); see Table 1 for some examples.

Table 1: Many optimization algorithms can be written as fixed-point iterations.

Algorithm	Problem	Iterates $z^{i+1} = T_\theta(z^i)$
Gradient descent	min $f_\theta(z)$	$z^{i+1} = z^i - \alpha\nabla f_\theta(z^i)$
Proximal gradient descent	min $f_\theta(z) + g_\theta(z)$	$z^{i+1} = \mathbf{prox}_{\alpha g_\theta}(z^i - \alpha\nabla f_\theta(z^i))$
ADMM (Douglas and Rachford, 1956) (Gabay and Mercier, 1976)	min $f_\theta(u) + g_\theta(u)$	$\tilde{u}^{i+1} = \mathbf{prox}_{g_\theta}(z^i)$ $u^{i+1} = \mathbf{prox}_{f_\theta}(2\tilde{u}^{i+1} - z^i)$ $z^{i+1} = z^i + u^i - \tilde{u}^i$
OSQP (Stellato et al., 2020)	min $(1/2)x^TPx + c^Tx$ s.t. $l \le Ax \le u$ dual var. (y) with $\theta = (\mathbf{vec}(P), \mathbf{vec}(A), c, l, u)$	$(x^i, v^i) = z^i$ $w^{i+1} = \Pi_{[l,u]}(v^i)$ solve $Qx^{i+1} = \sigma x^i + \rho A^T(2w^{i+1} - v^i)$ $v^{i+1} = \rho Ax^{i+1} + (1+\rho)v^i - (2\rho+1)w^{i+1}$ $z^{i+1} = (x^{i+1}, v^{i+1})$ with $Q = P + \sigma I + \rho A^TA$ primal-dual solution $(x, y) = (x, \rho(v - \Pi_{[l,u]}(v)))$
SCS (O'Donoghue, 2021)	min $(1/2)x^TPx + c^Tx$ s.t. $Ax + s = b$ dual var. (y) $s \in \mathcal{K}$ with $\theta = (\mathbf{vec}(P), \mathbf{vec}(A), c, b)$	solve $Q\tilde{u}^{i+1} = z^i$ $u^{i+1} = \Pi_{\mathcal{C}}(2\tilde{u}^{i+1} - z^i)$ $z^{i+1} = z^i + u^{i+1} - \tilde{u}^{i+1}$ with $Q = \begin{bmatrix} P + I & A^T \\ -A & I \end{bmatrix}$ $\mathcal{C} = \mathbf{R}^n \times \mathcal{K}$ primal-dual solution $(x, y) = u$

We denote **prox** as the proximal operator (Parikh and Boyd, 2014) and **vec** as the vectorization operator stacking the columns of a matrix (See the notation paragraph in 1 for formal definitions). See Appendix A for more information on the algorithms in this table.

Applications. *Parametric fixed-point problems* arise in several applications in machine learning, operations research, and engineering, where we repeatedly solve a problem of the form (1) with varying parameter θ. For example, in optimal control, we update the inputs (*e.g.*, propeller thrusts) as sensor signals (*e.g.*, system state) and goals (*e.g.*, desired trajectory) vary (Borrelli et al., 2017, Section 7.1). Other examples include backtesting financial models (Boyd et al., 2017), power flow optimization (Hentenryck, 2021; Zamzam and Baker, 2020), and image restoration (Elad and Aharon, 2006). In non-convex optimization, finding a stationary point can also be cast as a fixed-point problem (Wang et al., 2019; Hong et al., 2016). In game theory, finding the Nash equilibrium of a multi-player game can be formulated as a fixed-point problem under some mild assumptions on the utility functions of each player (Briceño-Arias and Combettes, 2013; Ryu and Boyd, 2016). Finding fixed-points are also important in other areas, such as finding the optimal policy of Markov decision processes (Bellman, 1957) and solving variational inequality problems (Rockafellar and Wets, 1998; Bauschke and Combettes, 2011).

Acceleration. In spite of the widespread use of fixed-point iterative algorithms, they are known to suffer from slow convergence to high-accuracy solutions (Zhang et al., 2020). Acceleration schemes (Zhang et al., 2020; Walker and Ni, 2011; d'Aspremont et al., 2021; Sopasakis et al., 2019) are an active area of research designed to speed up the solving of fixed-point problems. These methods, such as Anderson acceleration (Walker and Ni, 2011; Zhang et al., 2020), combine past iterates to generate the next one in order to improve the convergence behavior. Although acceleration methods are known to work well in certain cases, such as Nesterov acceleration to solve smooth, convex optimization problems, it is still an open research question to design schemes that are robust and versatile.

Learning for optimization. Instead of designing acceleration methods for single problems, recent approaches take advantage of the parametric structure of fixed-point problems encountered in practice to learn efficient solution methods. In particular, they *learn algorithm steps* using data from past solutions (Amos, 2023; Chen et al., 2022b). Despite recent successes in a variety of fields, *e.g.*, in sparse coding (Gregor and LeCun, 2010; Liu et al., 2019), convex optimization (Ichnowski et al., 2021; Venkataraman and Amos, 2021), and meta-learning (Li and Malik, 2016; Finn et al., 2017), most of these approaches lack *convergence guarantees* because they directly alter the algorithm iterations with learned variants (Chen et al., 2022b; Amos, 2023). Although some efforts have been made to safeguard the learned iterations (Prémont-Schwarz et al., 2022; Heaton et al., 2023; Banert et al., 2021), guaranteeing convergence for general learned optimizers is still a challenge. In addition, most of these approaches do not provide *generalization guarantees* on unseen data (Chen et al., 2022b; Amos, 2023).

Another data-driven approach to reduce the number of iterations is to *learn warm starts* rather than the steps of the algorithm (Chen et al., 2022a; Baker, 2019). An advantage to learning warm starts as opposed to algorithm steps is that this approach can be integrated with existing algorithms that provably converge from any starting point. However, existing methods to learn warm starts still lack generalization guarantees. They also decouple the learning procedure from the algorithm behavior after warm-starting, which can lead to suboptimality and infeasibility issues on unseen problem instances.

Our contributions. We present a learning framework that predicts warm starts for iterative algorithms of the form (2), which solve parametric fixed-point problems of the type given in (1). The framework consists of two modules. The first module maps the parameter to a warm start via a neural network, and the second runs a predefined number of steps of the fixed-point algorithm. We propose two loss functions. The first one is the *fixed-point residual* loss which directly penalizes the fixed-point residual of the output of the architecture. The second one is the *regression* loss which penalizes the distance between the output of the architecture and a given ground truth fixed-point (among possibly many).

Compared to existing literature on learning warm starts, we train our architecture by differentiating through the fixed-point iterations. In this way, we construct warm-start predictions that perform well after a specific number of fixed-point iteration in an *end-to-end* fashion. Furthermore, after training, our architecture allows the flexibility of selecting an arbitrary number of fixed-point iterations to perform and is not limited to the number it was originally trained on.

By combining operator theory with the PAC-Bayes framework (McAllester, 1998; Shawe-Taylor and Williamson, 1997), we provide two types of guarantees on the performance of our framework. First, we give bounds on the fixed-point residual when we apply our framework to an arbitrary number of steps, larger than the number used during training. Second, we provide generalization bounds to unseen problems for common classes of operators: contractive, linearly convergent, and averaged.

Finally, we apply our framework to a variety of algorithms including gradient descent, proximal gradient descent, and the alternating direction method of multipliers (ADMM). In our benchmarks, we show that our learned warm starts lead to a significant reduction in the required number of iterations used to solve the fixed-point problems. We also demonstrate compatibility with state-of-the-art solvers by learning architectures specifically tailored to the Splitting Conic Solver (SCS) (O'Donoghue et al., 2019) and the Operator Splitting Quadratic Program solver (OSQP) (Stellato et al., 2020), and inputting warm starts into the corresponding C implementations.

Notation. We denote the set of non-negative vectors of length n as $\mathbf{R}^n_+$, and the set of vectors with positive entries of length n as $\mathbf{R}^n_{++}$. We let the set of $n \times n$ positive semidefinite and positive definite matrices be $\mathbf{S}^n_+$ and $\mathbf{S}^n_{++}$ respectively. We define the set of fixed-points of the operator T, assumed to be non-empty, as $\mathbf{fix}\, T$. For any closed and convex set S, we denote $\mathbf{dist}_S : \mathbf{R}^n \to \mathbf{R}$ to be the distance function, where $\mathbf{dist}_S(x) = \min_{s \in S} \|s - x\|_2$. For any set $S \subset \mathbf{R}^n$, we define the indicator function $\mathcal{I}_S : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ where $\mathcal{I}_S(x) = 0$ if $x \in S$ and $\mathcal{I}_S(x) = +\infty$ otherwise. We take k applications of any single-valued operator T to be $T^k : \mathbf{R}^n \to \mathbf{R}^n$. For any matrix A, we denote its spectral norm and Frobenius norm with $\|A\|_2$ and $\|A\|_F$ respectively. For a matrix $Z \in \mathbf{R}^{m \times n}$, $\mathbf{vec}(Z)$ is the vector obtained by stacking the columns of Z. For a symmetric matrix $Y \in \mathbf{S}^n$, $\mathbf{vec}(Y)$ is the vector obtained by taking the upper triangular entries of matrix Y. We let the all-ones vectors of length n be $\mathbf{1} \in \mathbf{R}^n$. The proximal operator, $\mathbf{prox}_h : \mathbf{R}^n \to \mathbf{R}^n$, of h is defined as (Parikh and Boyd, 2014)

$$\mathbf{prox}_h(v) = \operatorname{argmin}_x \Big(h(x) + (1/2)\|x - v\|_2^2 \Big).$$

Outline. We structure the rest of the paper as follows. In Section 2, we review some related work on learned solvers. In Section 3, we present our learning to warm-start framework. In Section 4, we provide generalization guarantees to unseen data for our method. In Section 5, we discuss choosing the right architecture, namely the choice of loss function and the number steps to train on. Section 6 presents various numerical benchmarks.

2. Related work

Learning warm starts. A common approach to reduce the number of iterations of iterative algorithms is to learn a mapping from problem parameters to high-quality initializations. This paper extends our previous efforts (Sambharya et al., 2023) to learn warm starts for Douglas-Rachford splitting in the context of convex quadratic programs (QPs) in three main directions. First, we consider fixed-point problems arising in parametric convex optimization, which generalizes the QP case. Second, we present new generalization guarantees for averaged, linearly convergent, and contractive operators, as opposed to just contractive operators (Sambharya et al., 2023). We also construct generalization guarantees when, at evaluation time, we evaluate the fixed point operator for more steps than the ones used during training. Finally, we consider two losses to judge warm starts: the fixed-point residual loss (Sambharya et al., 2023), and a new regression loss based on the distance between the predicted solution and the optimal one. In contrast to our approach, most of the techniques to learn warm starts don't consider the downstream algorithm in the warm start prediction. Baker (2019) and Mak et al. (2023) use machine learning to warm-start the optimal power flow problem. In the model predictive control (MPC) (Borrelli et al., 2017) paradigm, Chen et al. (2022a) use a neural network to accelerate the optimal control law computation by warm-starting an active set method. Other works in MPC use machine learning to predict an approximate optimal solution and, instead of using it to warm-start an algorithm, directly ensure feasibility and optimality. Chen et al. (2018) and Karg and Lucia (2020) use a constrained neural network architecture that guarantees feasibility by projecting its output onto the QP feasible region. Zhang et al. (2019) uses a neural network to predict the solution while also certifying suboptimality of the output. Our paper differs from these works in that the training of the neural network we propose is designed to minimize the loss after many fixed-point steps, allowing us to improve solution quality. Our work is also more general in scope since we consider general parametric fixed-point problems. Finally, we provide generalization guarantees to unseen data which other works lack.

Learning algorithm steps for convex optimization. In the area of learning to optimize (Chen et al., 2022b) or amortized optimization (Amos, 2023), a parallel approach to learning warm starts consists in learning the algorithm steps themselves to solve convex optimization problems. Ichnowski et al. (2021) and Jung et al. (2022) use reinforcement learning to solve quadratic programs quickly by learning high-quality hyperparameters of algorithms. Venkataraman and Amos (2021) learns to accelerate fixed-point problems that correspond to convex problems quickly. One risk of some of these approaches is that convergence may not be guaranteed (Amos, 2023). To solve this problem, some works safeguard learned optimizers to guarantee convergence by reverting to a fallback update if the learned update starts to diverge (Heaton et al., 2023; Prémont-Schwarz et al., 2022). Other strategies guarantee convergence by making sure that the learned algorithm does not deviate too much from a known convergent algorithm (Banert et al., 2021) or by providing convergence rate bounds (Tan et al., 2023). In addition to convergence challenges, approaches that learn algorithm steps generally do not have generalization guarantees to unseen data (Amos, 2023; Chen et al., 2022b). Lastly, these methods generally cannot interface with existing algorithms that are written in C.

Learning algorithm steps beyond convex optimization. Many works have learned algorithm steps for problems outside of convex optimization. For example, in non-convex optimization, Sjölund and Bånkestad (2022) use graph neural networks (Wu et al., 2022) to accelerate algorithms to solve matrix factorization problems, and Bai et al. (2022) learn the acceleratation scheme to solve fixed-point problems quickly. The research area of *data-driven algorithm design* focuses on learning hyperparameters for algorithms used in combinatorial optimization problems (Balcan, 2020) like partitioning problems (Balcan et al., 2017). In this work, we instead focus on convex optimization problems.

The idea of learning algorithm steps has ventured beyond optimization. There has been a surge in recent years to learn algorithm steps to solve *inverse problems*, that is, problems where one wishes to recover a true signal, rather than minimizing an objective (Chen et al., 2022b). This is typically done by embedding algorithm steps or reasoning layers (Chen et al., 2020) into a deep neural network and has been applied to various fields such as sparse coding (Gregor and LeCun, 2010; Liu et al., 2019; Wu et al., 2020), image restoration (Diamond et al., 2017; Zhang et al., 2017; Chang et al., 2017), and wireless communication (He et al., 2020; Balatsoukas-Stimming and Studer, 2019). A widely used technique involves *unrolling* algorithmic steps (Monga et al., 2021), meaning differentiating through these steps to minimize a performance loss. While we also unroll algorithm steps, our work is different in scope since we aim to solve optimization problems rather than inverse problems, and in method since we learn warm starts rather than algorithm steps. Additionally, generalization and convergence remain issues in the context of learning to solve inverse problems (Chen et al., 2022b; Amos, 2023).

Learning surrogate optimization problems. Instead of solving the original parametric problem, several works aim to learn a surrogate model of large optimization problems. Then, an approximate solution can be obtained by solving the simpler or smaller optimization problem. For instance, Wang et al. (2020) learn a mapping to reduce the dimensionality of the decision variables in the surrogate problem. Li et al. (2023) use a neural approximator with reformulation and relaxation steps to solve linearly constrained optimization problems. Other works predict which constraints are active (Misra et al., 2022) and the value of the optimal integer solutions (Bertsimas and Stellato, 2021, 2022). In contrast, our approach refrains from approximating any problem; instead, we warm-start the fixed-point iterations. This allows us to clearly quantify the suboptimality achieved within a set number of fixed-point iterations.

Meta-learning. Meta-learning (Hospedales et al., 2021; Vilalta and Drissi, 2001; Ruder, 2017) or learning to learn overlaps with the learning for optimization literature when the tasks are general machine learning tasks (Chen et al., 2022b). A wide array of works learn the update function to gradient-based methods to speed up machine learning tasks with a variety of techniques including reinforcement learning (Li and Malik, 2016), unrolled gradient steps (Andrychowicz et al., 2016), and evolutionary strategies (Metz et al., 2022). More in the spirit of our work, Finn et al. (2017) learn the initial model weights so that a new task can be learned after only a few gradient updates. While the initialization of the model weights for their method is shared across the tasks, in our method we, instead, predict the warm start from the problem parameter. This tailors our initialization to the specific parametric problem under consideration.

Algorithms with predictions. Another area that uses machine learning to improve algorithm performance is algorithms with predictions (Mitzenmacher and Vassilvitskii, 2020; Kraska et al., 2018; Khodak et al., 2022). Here, algorithms take advantage of a possibly imperfect prediction of some aspect of the problem to improve upon worst-case analysis. This idea has been applied to many problems such as ski-rental (Purohit et al., 2018), caching (Rohatgi, 2020), and bipartite matching (Dinitz et al., 2021). Even though the prediction can be used to improve the warm start for algorithms (Dinitz et al., 2021; Sakaue and Oki, 2022), the task we consider is fundamentally different since we aim to solve parametric problems as quickly as possible rather than to take advantage of a prediction.

Generalization guarantees. The generalization guarantees we provide use a PAC-Bayes framework, which has been used in prior work in the amortized optimization setting (Gupta and Roughgarden, 2017; Bartlett et al., 2022). Chen et al. (2020) provide generalization guarantees for architectures with reasoning layers, using a local Rademacher complexity analysis. Balcan et al. (2021) use ideas from statistical learning theory to translate pseudo-dimension bounds to generalization guarantees for combinatorial problems. However, to the best of our knowledge, generalization guarantees have not been obtained with methods that aim to solve fixed-point problems quickly. Additionally, the bounds from these works mentioned above are obtained in methods where the algorithm steps are learned rather than the warm start. Unlike Sambharya et al. (2023) which focused on solving QPs, we obtain guarantees in the non-contractive case by using the PAC-Bayes framework rather than Rademacher complexity theory. We later expand our work in this paper and develop a framework to train learned optimizers to directly minimize an out-of-sample loss using PAC-Bayes theory (Sambharya and Stellato, 2024).

Classical and computer-assisted convergence analysis. Worst-case convergence analysis of first-order methods offers strong performance guarantees for many fixed-point operators, including contractive, linearly convergent, and averaged (Ryu and Yin, 2022, Section 2) (Bauschke and Combettes, 2011, Section 4). Recently, computed-assisted approaches like the performance estimation problem (PEP) (Drori and Teboulle, 2014; Taylor et al., 2017a,b; Ryu et al., 2020) have improved classical convergence results by finding a worst-case convergence certificate via semidefinite programming (SDP). While often tight, such guarantees do not exploit the specific structure of parametric optimization problems and fail to account for warm-starting, which can lead to conservative results. Addressing this, Ranjan and Stellato (2024) introduced a technique inspired by neural network verification (Fazlyab et al., 2022) to compute worst-case guarantees of warm-started fixed-point algorithms for parametric QPs. In this work, rather than computing worst-case guarantees, we take a probabilistic approach by focusing on generalization guarantees on unseen problem instances. Specifically, we combine classical convergence analysis with the PAC-Bayes framework to analyze the performance of fixed-point algorithms with learned warm-starts.

3. Learning to warm-start framework

We now present our learning framework to learn warm starts to solve the parametric fixed-point problem (1). A key feature of our framework is the inclusion of a predefined number of fixed-point steps within the architecture. In this way, the warm-start predictions are tailored for the downstream algorithm, and we conduct end-to-end learning. The section is organized as follows. In Section 3.1, we provide intuition as to why learning end-to-end can be beneficial through a small illustrative example. In Section 3.2, we describe our architecture, and in Section 3.3 we introduce the two different loss functions we consider. A concise summary of these aspects is depicted in Figure 2.

3.1 An illustrative example

To build intuition, we provide a two-dimensional example that illustrates the importance of tailoring the warm-start prediction to the downstream algorithm. Consider the problem

$$\begin{array}{ll} \text{minimize} & (1/2)z^TQz + c^Tz \\ \text{subject to} & z \geq 0, \end{array} \tag{3}$$

where $Q = \mathbf{diag}(10, 1)$ and $c = (0, -1)$. We solve problem (3) using proximal gradient descent (see Table 1) with the iterates

$$z^{i+1} = \Pi(z^i - \alpha \nabla f(z^i)),$$

where $\nabla f(z) = Qz + c$, and Π is the projection onto the non-negative orthant. Here, the step size $\alpha \in \mathbf{R}_{++}$ is picked as $2/(\mu + L)$ where $\mu = 1$ and $L = 10$ are the strong convexity and smoothness parameters of the objective. This step size provides the strongest bound on the worst-case convergence rate (Ryu and Boyd, 2016). The optimal solution for problem (3) is at the point $z^\star = (0, 1)$, and we consider three different warm starts shown in Figure 1. All three are equidistant to the optimal solution, but lead to different convergence behavior. The purple warm start has the fastest convergence since the projection step clips non-negative values to zero. The orange warm start converges more quickly than the green warm start due to the difference in scaling of the objective function along each axis. This results in faster convergence for the orange warm start compared with the green one since the orange warm start is closer to the z_1 axis. This example shows the necessity of considering the downstream algorithm when choosing a warm start. All three warm starts in this case appear of equal quality as they are equidistant from $z^\star$, but when considering the downstream algorithm, there is a clear hierarchy in terms of convergence speed: purple takes the lead, followed by orange, then green.

3.2 Learning to warm-start architecture

Our learning architecture consists of two modules, a neural network with L layers and k iterations of operator T_θ; see Figure 2. The neural network uses ReLU activation functions defined as $\phi(z) = \max(0, z)$ element-wise. We let $w = \{W_i\}_{i=1}^L$ be the neural network weights for each layer where $W_i \in \mathbf{R}^{m_i \times n_i}$. Our warm-start prediction is computed as

$$h_w(\theta) = W_L \phi(W_{L-1}\phi(\dots \phi(W_1 \theta))). \tag{4}$$

While we do not explicitly represent bias terms, we can include them by appending a new column to matrices W_i for $i = 1, \dots, L$, and a 1 to the input vector. The warm-start prediction $h_w(\theta) \in \mathbf{R}^p$ feeds into the fixed-point algorithm parametrized by θ. The second

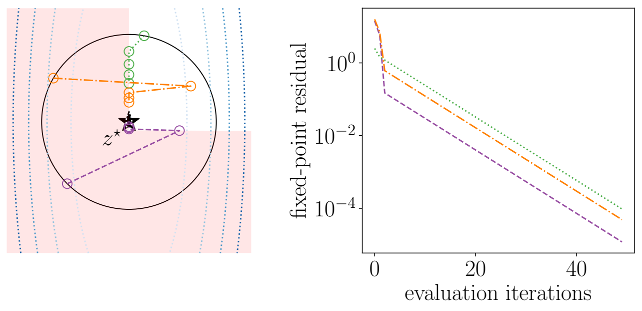


Figure 1: The iterates of proximal gradient descent to solve problem (3) with different warm starts. For three different warm starts equidistant to the optimal solution $z^\star$, we plot the first 5 iterates on the left. The contour lines of the objective function are in blue and the infeasible region is shaded in pink. We plot the fixed-point residuals for the different warm starts on the right. Depending on the warm start, the convergence to the optimal solution, can vary greatly.

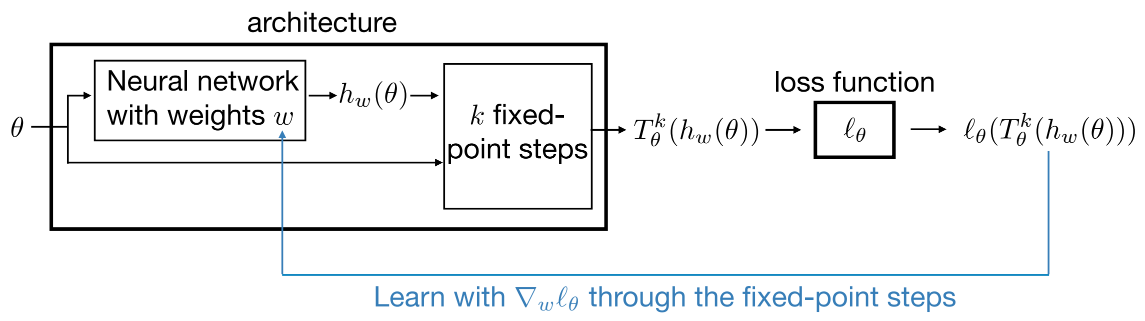

architecture
θ
Neural network with weights w
$h_w(\theta)$
k fixed-point steps
$T_\theta^k(h_w(\theta))$
loss function
ℓ_θ
$\ell_\theta(T_\theta^k(h_w(\theta)))$
Learn with $\nabla_w \ell_\theta$ through the fixed-point steps

Figure 2: Illustration of the learning framework. The architecture consists of two modules: a neural network mapping the parameter θ to a warm start $h_w(\theta)$, and a second module executing k fixed-point iterations starting from $h_w(\theta)$ to obtain the candidate solution $T_\theta^k(h_w(\theta))$. The fixed-point steps in the architecture depend on the parameter θ, and have no learnable weights. There are two options for the loss function ℓ_θ: the fixed-point residual loss ℓ_θ^{fp}, or the regression loss ℓ_θ^{reg}. We backpropagate from the loss through the fixed-point iterates to learn the neural network weights w.

part of our architecture consists of k applications of the operator T_θ to the warm start $h_w(\theta)$. The final output is the candidate solution $T_\theta^k(h_w(\theta))$.

We only consider feedforward neural networks as the first module in this work, but any architecture that maps the problem parameters to warm starts could be used. Also note that our approach is a generalization of the Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) framework. Indeed, MAML learns a constant initialization shared over a family of problems, whereas our method learns *instance-dependent initializations* where the warm start $h_w(\theta)$ changes as the parameter θ varies.

3.3 Loss functions

Training for k steps. We propose two loss functions to analyze the output of our learning to warm-start architecture, $T_\theta^k(h_w(\theta))$. The first one is the *fixed-point residual loss*

$$\ell_\theta^{\text{fp}}(z) = \|T_\theta(z) - z\|_2, \tag{5}$$

which measures of the distance to convergence of the fixed-point algorithm (2) (Ryu and Yin, 2022, Section 2.4). The second one is the *regression loss*

$$\ell_\theta^{\text{reg}}(z) = \|z - z^\star(\theta)\|_2, \tag{6}$$

where $z^\star(\theta)$ is a known (possibly non-unique) fixed-point of T_θ. The learning problem is

$$\text{minimize} \quad \mathbf{E}_{\theta\sim Q}\ell_\theta(T_\theta^k(h_w(\theta))), \tag{7}$$

where ℓ_θ is either ℓ_θ^{fp} or ℓ_θ^{reg}, and k is the number of fixed-point iterations in our architecture. We use the 2-norm in Equations (5) and (6) as opposed to its square, as this is more coherent with the standard optimality metrics used in optimization solvers (Busseti et al., 2018; Ryu and Yin, 2022, Section 2.4). Note that choosing $k = 0$ decouples the learning procedure from the downstream algorithm, thereby making our architecture no longer end-to-end.

It is generally infeasible to evaluate the objective in problem (7) because the distribution Q is unknown. Instead, we minimize its empirical estimate over training data hoping to attain generalization to unseen data. We leverage stochastic gradient descent (SGD) methods to efficiently train the neural network weights, by constructing stochastic approximations of the gradient of the empirical risk (Sra et al., 2011). To compute such gradient estimates, we use automatic differentiation (Baydin et al., 2017) techniques to differentiate through the k fixed-point iterations. We note that due to the inclusion of ReLU layers and projection steps in the fixed-point algorithms (*e.g.*, the projection step in OSQP), there are non-differentiable mappings in the architecture. At non-differentiable points, SGD uses subgradients (Rockafellar and Wets, 1998) to estimate directional derivatives of the loss. By tailoring the warm-start prediction to the downstream fixed-point algorithm, our framework constitutes an end-to-end learning scheme.

Testing for t steps. We now evaluate the learned model with t fixed-point iterations (t possibly different from k used during training) on an unseen parameter θ. While we consider two different loss functions for training, we always measure the test performance on unseen problems by the fixed-point residual since it is a standard measure of progress (Ryu and Yin, 2022, Section 2.4). To analyze the generalization of our architecture, we define the *risk* as the following function of t:

$$R^t(w) = \mathbf{E}_{\theta\sim Q}\ell_\theta^{\text{fp}}(T_\theta^t(h_w(\theta))). \tag{8}$$

Since we only access the distribution Q via N samples $\theta_1, \dots, \theta_N$, we define the *empirical risk* as

$$\hat{R}^t(w) = \frac{1}{N}\sum_{i=1}^{N} \ell_{\theta_i}^{\text{fp}}(T_{\theta_i}^t(h_w(\theta_i))). \tag{9}$$

4. PAC-Bayes generalization bounds

In this section, we provide generalization bounds for our approach using the PAC-Bayes framework (Shawe-Taylor and Williamson, 1997; McAllester, 1998). More specifically, we provide a generalization guarantee on the risk in Equation (8) after any number of evaluation steps t (t need not be equal to the number of fixed-point steps k taken during training).

First, we introduce preliminary results and definitions needed for our proofs in Section 4.1. In particular, we define the *marginal fixed-point residual*, a key ingredient of our proof technique, which measures the maximum fixed-point residual incurred by a warm start when subjected to a bounded perturbation. Then, we derive our main generalization bound result, Theorem 2, in Section 4.2. Finally, in Section 4.3, we specialize Theorem 2 to three different cases of operators: contractive, linearly convergent, and averaged.

4.1 Preliminaries

In this subsection, we introduce our marginal fixed-point residual in Equation (10) and McAllester's bound in inequality (11).

Marginal fixed-point residual. We define the marginal fixed-point residual to be the worst-case fixed-point residual for a warm start subjected to a bounded perturbation:

$$g^t_{\gamma,\theta}(z) = \max_{\|\Delta\|_2 \le \gamma} \ell^{\text{fp}}_\theta(T^t_\theta(z+\Delta)). \tag{10}$$

Similarly, we define the marginal risk and marginal empirical risk in the same way as for the non-marginal case from Section 3 with

$$R^t_\gamma(w) = \mathbf{E}_{\theta\sim Q} g^t_{\gamma,\theta}(h_w(\theta)) \quad \text{and} \quad \hat{R}^t_\gamma(w) = \frac{1}{N}\sum_{i=1}^N g^t_{\gamma,\theta_i}(h_w(\theta_i)).$$

Setting $\gamma = 0$ corresponds to the original fixed-point residual and risk functions, *i.e.*, $g^t_{0,\theta}(z) = \ell^{\text{fp}}_\theta(T^t_\theta(z))$, $\hat{R}^t_0(w) = \hat{R}^t(w)$, and $R^t_0(w) = R^t(w)$ from Equations (8) and (9).

McAllester's bound. The PAC-Bayesian framework provides generalization bounds for randomized predictors, as opposed to a learned single predictor. Randomized predictors are obtained by sampling in a set of basic predictors based on a specific probability distribution (Alquier, 2021). This is especially useful in our setting because we can manipulate the bounds on the randomized predictors into bounds on our learned predictors.

In our case, h_w from Equation (4) corresponds to the fixed warm-start prediction parameterized by the weights of the neural network $w \in \mathcal{W}$ where $\mathcal{W}$ is a set of possible weights. We aim to bound $R^t(w)$, the risk after t fixed-point steps from Equation (8), in terms of empirical quantities. To do so, we consider perturbations of the neural network weights given by the random variable u whose distribution may also depend on the training data. Now, we have a distribution of predictors h_{w+u}, where w is fixed and u is random. Given a prior distribution π over the set of predictors that is independent of the training data, the expected marginal risk of the randomized predictor $\mathbf{E}_u[R^t_\gamma(w+u)]$ can be bounded as (McAllester, 2003)

$$\mathbf{E}_u[R^t_\gamma(w+u)] \le \mathbf{E}_u[\hat{R}^t_\gamma(w+u)] + 2C_\gamma(t)\sqrt{\frac{2(\text{KL}(w+u\|\pi) + \log(2N/\delta))}{N-1}}, \tag{11}$$

with probability at least $1-\delta$. Here $\mathrm{KL}(p||\pi)$ is the KL-divergence between the distributions $p = w + u$ and π,

$$\mathrm{KL}(p||\pi) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{\pi(x)}\right) dx.$$

We define the maximum possible fixed-point residual after t steps as

$$C_\gamma(t) = \max_{\theta \in \Theta, w \in \mathcal{W}} g^t_{\gamma,\theta}(h_w(\theta)).$$

Note that the marginal risk and empirical marginal risk lie in the range $[0, C_\gamma(t)]$. In order to bound $C_\gamma(t)$, we will consider predictors where the distance from warm start to the set of fixed-points is upper bounded by D:

$$\mathbf{dist}_{\mathbf{fix}\, T_\theta}(h_w(\theta)) \leq D \quad \forall \theta \in \Theta,\ w \in \mathcal{W}. \tag{12}$$

In Section 4.3, we bound $C_\gamma(t)$ in terms of t, γ, D, and properties of the operator T_θ.

4.2 Generalization bounds

In this subsection, we use the marginal fixed-point residual and the McAllester bound from Section 4.1, *i.e.*, Equation (11), to bound the generalization gap. We first transform the McAllester bound in (11), which provides a generalization bound on the expected marginal risk of the randomized predictor, to a bound on the risk with the following lemma.

Lemma 1 *Let $h_w : \Theta \to \mathbf{R}^p$ be any predictor to a warm start learned from the training data such that $g^t_{\gamma/2,\theta}(h_w(\theta)) \leq C_{\gamma/2}(t)$, $\forall \theta \in \Theta$. Let h_w be any learned predictor parametrized by w and π be any distribution that is independent from the training data. Then, for any $\delta, \gamma > 0$, with probability at least $1-\delta$ over a training set of size N and for any random perturbation u such that $\mathbf{P}(\max_{\theta \in \Theta} \|h_{w+u}(\theta) - h_w(\theta)\|_2 \leq \gamma/2) \geq 1/2$ we have*

$$R^t(w) \leq \hat{R}^t_\gamma(w) + 4C_{\gamma/2}(t)\sqrt{\frac{\mathrm{KL}(w+u||\pi) + \log(6N/\delta)}{N-1}}.$$

See Appendix B.1 for the proof. In the above expression, w is fixed and u is a random variable. This lemma bears resemblance to Neyshabur et al. (2018, Lemma 1), and the proof is nearly identical. Next, we use Lemma 1 to obtain generalization bounds with our main theorem.

Theorem 2 *Assume that $\|\theta\|_2 \leq B$ for all $\theta \in \Theta$. Let $h_w : \Theta \to \mathbf{R}^p$ be an L-layer neural network with ReLU activations where $g^t_{\gamma/2,\theta}(h_w(\theta)) \leq C_{\gamma/2}(t)$, $\forall \theta \in \Theta$. Let $c = B^2 L^2 \bar{h} \log(L\bar{h}) \Pi_{j=1}^L \|W_j\|_2^2 \sum_{i=1}^L \|W_i\|_F^2 / \|W_i\|_2^2$ and let $\bar{h} = \max_i n_i$ be the largest number of output units in any layer. Then for any $\delta, \gamma > 0$ with probability at least $1-\delta$ over a training set of size N,*

$$R^t(w) \leq \hat{R}^t_\gamma(w) + \begin{cases} \mathcal{O}\left(C_{\gamma/2}(t)\sqrt{\frac{c + \log(\frac{LN}{\delta})}{\gamma^2 N}}\right) & \text{if } \Pi_{j=1}^L \|W_j\|_2 \geq \frac{\gamma}{2B} \\ C_{\gamma/2}(t)\sqrt{\frac{\log(1/\delta)}{2N}} & \text{else.} \end{cases} \tag{13}$$

See Appendix B.2 for the proof. With Theorem 2, we bound the risk in terms of the empirical marginal risk and a penalty term. The main case is when the weights are sufficiently large: $\Pi_{j=1}^{L}\|W_j\|_2 \geq \gamma/(2B)$. In this case, we use the PAC-Bayesian framework to provide the generalization bound. We directly use the perturbation bound from Neyshabur et al. (2018, Lemma 2) which bounds the change in the warm start $h_w(\theta)$ with respect to the change in the neural network weights w. In the other case, if $\Pi_{j=1}^{L}\|W_j\|_2 \leq \gamma/(2B)$, then the warm start $h_w(\theta)$ is close to the zero vector. Here, we leverage Hoeffding's inequality to get the generalization bound.

As $t \to \infty$, the generalization gap in Theorem 2 approaches zero since $C_{\gamma/2}(t)$ goes to zero. Intuitively, this happens because the algorithm is run until convergence. On the other hand, as $N \to \infty$, the second term in each of the cases disappears and the generalization gap becomes the difference between the marginal empirical risk and the risk for a fixed γ. Our bounds also generalizes the setting where the warm start is not learned. Setting all of the weights to zero corresponds to warm-starting every problem from the zero vector. In this case, with high probability, $R^t(0) \leq \hat{R}^t(0) + C_{\gamma/2}(t)\sqrt{\log(1/\delta)/(2N)}$.

4.3 Bounding the empirical marginal risk

Theorem 2 bounds the risk $R^t(w)$ in terms of the empirical marginal risk $\hat{R}^t_\gamma(w)$ plus a penalty term. In this subsection, we use operator theory to bound two things: i) $\hat{R}^t_\gamma(w)$, thus removing the dependency on the marginal component, and ii) $C_{\gamma/2}(t)$ in terms of D given by Equation (12). We first assume that the operator T_θ is non-expansive, which is a common characteristic of solving convex problems (Ryu and Boyd, 2016).

Definition 3 (Non-expansive operator) *An operator T is non-expansive if*

$$\|Tx - Ty\|_2 \leq \|x - y\|_2, \quad \forall x, y \in \mathbf{dom}\, T.$$

Since non-expansiveness is not enough to guarantee convergence, we break our analysis into three different cases of fixed-point operators which converge: contractive in Section 4.3.1, linearly convergent in Section 4.3.2, and averaged in Section 4.3.3. By using the different properties of each, we can bound the marginal fixed-point residual after t steps, $g^t_{\gamma,\theta}(z)$ defined in (10). Since the empirical marginal risk is the average of these marginal fixed-point residuals, we can remove the dependence on the empirical marginal risk in Theorem 2. The sets of the three different types of operators are not mutually exclusive as seen in the set relationships depicted in Figure 3. The contractive case provides the strongest bounds, followed by the linearly convergent case, and then the averaged case.

To help in the subsequent analysis, we define the following functions which give the distance to optimality and marginal distance to optimality:

$$r_\theta(z) = \mathbf{dist}_{\mathbf{fix}\, T_\theta}(z), \quad f^t_{\gamma,\theta}(z) = \max_{\|\Delta\|_2 \leq \gamma} r_\theta(T^t_\theta(z + \Delta)). \tag{14}$$

We give the following lemma to relate the fixed-point residual to the distance to optimality.

Lemma 4 *For any non-expansive operator T_θ,*

$$\ell^{\text{fp}}_\theta(z) \leq 2 r_\theta(z).$$

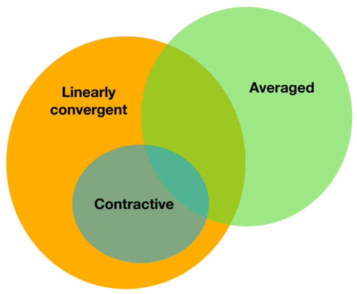


Figure 3: The set relationship between the different types of operators we consider in this section.

See Appendix B.3 for the proof. Note that the more general statement where the closest point in $\mathbf{fix}\, T_\theta$ is replaced by *any* point in $\mathbf{fix}\, T_\theta$ holds, i.e., it holds that for any $y \in \mathbf{fix} T_\theta$, we have that $\ell^{\rm fp}_\theta(z) \leq 2\|z - y\|_2$ for any z.

4.3.1 Contractive operators

We first consider contractive operators which give the strongest perturbation bounds.

Definition 5 (β-contractive operator) *An operator T is β-contractive for $\beta \in (0,1)$ if*

$$\|Tx - Ty\|_2 \leq \beta \|x - y\|_2 \quad \forall x, y \in \mathbf{dom}\, T.$$

If T_θ is β-contractive, then

$$g^t_{\gamma,\theta}(z) \leq \ell^{\rm fp}_\theta(T^t_\theta(z)) + 2\beta^t \gamma, \tag{15}$$

which follows from $\ell^{\rm fp}_\theta(T^t_\theta(\cdot))$ being $2\beta^t$-Lipschitz (Sambharya et al., 2023, Appendix A.1). In the contractive case, we remove the marginal risk dependency with the following corollary.

Corollary 6 *We define B and $\bar{h}$ as in Theorem 2. Let T_θ be β-contractive for any $\theta \in \Theta$. Let h_w be an L-layer neural network with ReLU activations such that* (12) *holds with bound D. Let $c = B^2 L^2 \bar{h} \log(L\bar{h}) \Pi_{j=1}^L \|W_j\|_2^2 \sum_{i=1}^L \|W_i\|_F^2 / \|W_i\|_2^2$. Then for any $\delta, \gamma > 0$ with probability $\geq 1 - \delta$ over a training set of size N,*

$$R^t(w) \leq \hat{R}^t(w) + 2\beta^t\gamma + \begin{cases} \mathcal{O}\left(\beta^t(D + \frac{\gamma}{2})\sqrt{\frac{c + \log(\frac{LN}{\delta})}{\gamma^2 N}}\right) & \textit{if } \Pi_{j=1}^L \|W_j\|_2 \geq \frac{\gamma}{2B} \\ 2\beta^t(D + \frac{\gamma}{2})\sqrt{\frac{\log(1/\delta)}{2N}} & \textit{else.} \end{cases}$$

Proof We remove the marginal dependence by applying inequality (15) to get

$$\hat{R}^t_\gamma(w) = \frac{1}{N}\sum_{i=1}^N g^t_{\gamma,\theta}(h_w(\theta_i)) \leq 2\beta^t\gamma + \frac{1}{N}\sum_{i=1}^N \ell^{\rm fp}_\theta(T^t_\theta(h_w(\theta_i))) = \hat{R}^t(w) + 2\beta^t\gamma.$$

We bound the worst-case fixed-point residual as $C_{\gamma/2}(t) \leq 2\beta^t(D + \gamma/2)$ which comes from $C_0(t) \leq 2\beta^t D$ (Sambharya et al., 2023, Appendix A.1) and the inequality

$$\mathbf{dist}_{\mathbf{fix}\, T_\theta}(h_w(\theta) + \Delta) \leq \|\Pi_{\mathbf{fix}\, T_\theta}(h_w(\theta)) - (h_w(\theta) + \Delta)\|_2 \leq \mathbf{dist}_{\mathbf{fix}\, T_\theta}(h_w(\theta)) + \|\Delta\|_2. \tag{16}$$

Here, $\Pi_{\mathbf{fix}\, T_\theta}$ is the projection on the set $\mathbf{fix}\, T_\theta$. The first inequality in (16) uses the definition of the distance function $\mathbf{dist}_{\mathbf{fix}\, T_\theta}$, and the second uses the triangle inequality. ■

4.3.2 Linearly convergent operators

Now, we consider a broader category of operators, linearly convergent operators.

Definition 7 (β-linearly convergent operator) *An operator T is β-linearly convergent for $\beta \in [0, 1)$ if*

$$\mathbf{dist}_{\mathbf{fix}\, T}(Tx) \leq \beta \mathbf{dist}_{\mathbf{fix}\, T}(x) \quad \forall x \in \mathbf{dom}\, T.$$

If the operator T_θ is not contractive, then we get the weaker property that $\ell_\theta^{\text{fp}}(T_\theta^t(\cdot))$ is 2-Lipschitz. To provide tighter perturbation bounds, we first establish the following lemma.

Lemma 8 *For any non-expansive operator T_θ and for any $t \geq 0$,*

$$|r_\theta(T_\theta^t(z)) - r_\theta(T_\theta^t(w))| \leq 2\|z - w\|_2.$$

See Appendix B.4 for the proof. We now use Lemma 8 and the linear convergence guarantee, Definition 7, to bound $g_{\gamma,\theta}^t$ in terms of empirical quantities.

Lemma 9 *Assume that T_θ is β-linearly convergent where $\beta \in (0, 1)$. Then the following bounds hold for all $t \geq 0$:*

$$f_{\gamma,\theta}^t(z) \leq r_\theta(T_\theta^t(z)) + 2\gamma, \quad f_{\gamma,\theta}^{t+1}(z) \leq \beta f_{\gamma,\theta}^t(z)$$

$$g_{\gamma,\theta}^t(z) \leq 2 f_{\gamma,\theta}^t(z).$$

Proof The inequality $f_{\gamma,\theta}^{t+1}(z) \leq \beta f_{\gamma,\theta}^t(z)$ comes from

$$f_{\gamma,\theta}^{t+1}(z) = r_\theta(T_\theta^{t+1}(z + \Delta^\star)) \leq \beta r_\theta(T_\theta^t(z + \Delta^\star)) \leq \beta f_{\gamma,\theta}^t(z),$$

where $\|\Delta^\star\|_2 \leq \gamma$ is the maximizer to $f_{\gamma,\theta}^{t+1}(z)$. The first inequality comes from Definition 7 and the second from (14). The inequality $f_{\gamma,\theta}^{t+1}(z) \leq r_\theta(T_\theta^{t+1}(z)) + 2\|\Delta\|_2$ in Lemma 9 follows from Lemma 8. The final inequality in Lemma 9 is derived as follows:

$$g_{\gamma,\theta}^t(z) = \ell_\theta^{\text{fp}}(T_\theta^t(z + \Delta^\star)) \leq 2 r_\theta(T_\theta^t(z + \Delta^\star)) \leq 2 f_{\gamma,\theta}^t(z).$$

Here, $\|\Delta^\star\|_2 \leq \gamma$ is the maximizer for $g_{\gamma,\theta}^t(z)$, and Lemma 4 gives the first inequality. ■

Using Lemma 9, we can bound the marginal empirical risk for the linearly convergent case. For the β-linearly convergent case, $C_{\gamma/2}(t)$ is bounded by $2\beta^t(D + \gamma/2)$ which uses (16) and $C_0(t) \leq 2r_\theta(T_\theta^t(z)) \leq 2\beta^t D$. The first inequality comes from Lemma 4 and the second inequality follows from Definition 7.

4.3.3 Averaged operators

Lastly, we consider the averaged operator case which in general gives sublinear convergence.

Definition 10 (α-averaged operator) *An operator T is α-averaged for $\alpha \in (0,1)$ if there exists a non-expansive operator R such that $T = (1-\alpha)I + \alpha R$.*

Lemma 11 *Let T_θ be an α-averaged operator. Then the following bound holds:*

$$g^t_{\gamma,\theta}(z) \leq \min_{j=0,\dots,t} \sqrt{\frac{\alpha}{(1-\alpha)(t-j+1)}}(r_\theta(T^j_\theta(z)) + 2\gamma) \quad \textit{for } t \geq 0.$$

Proof Let $\bar{\alpha}_{t,j} = \sqrt{\frac{\alpha}{(1-\alpha)(t-j+1)}}$. There exists $\|\Delta^\star\|_2 \leq \gamma$ such that the equality below holds by definition of the marginal fixed-point residual:

$$g^t_{\gamma,\theta}(z) = \ell^{\text{fp}}_\theta(T^t_\theta(z+\Delta^\star)) \leq \bar{\alpha}_{t,j}(r_\theta(T^j_\theta(z+\Delta^\star))) \leq \bar{\alpha}_{t,j} f^j_{\gamma,\theta}(z) \leq \bar{\alpha}_{t,j}(r_\theta(T^j_\theta(z)) + 2\gamma). \quad (17)$$

The three inequalities comes from Ryu and Yin (2022, Theorem 1), the definition of $f^j_{\gamma,\theta}(z)$ in (14), and Lemma 9 respectively. Equation (17) holds for all $0 \leq j \leq t$. ■

Using Lemma 11, we can bound the marginal empirical risk for the averaged case. We bound the worst-case marginal fixed-point residual with $C_{\gamma/2}(t) \leq \sqrt{\alpha/((1-\alpha)(t+1))}(D+\gamma)$ which follows from Lemma 11 by letting $z = h_w(\theta)$ and $j = 0$. Then the inequality holds for every $\theta \in \Theta$.

5. Choosing the right computational architecture

In this section, we discuss how the number of fixed-point steps the model is trained on, k, and the loss function affect performance.

5.1 Bounds on the fixed-point residual for t evaluation steps

In this subsection, we derive bounds on the fixed-point residual after t steps, $\ell^{\text{fp}}_\theta(T^t_\theta(z))$, in terms of the loss after k steps, $\ell_\theta(T^k_\theta(z))$ where $k < t$. A summary of these results is given in Table 2 where we provide the bound for each of the two loss functions. The bounds in Table 2 using the fixed-point residual loss in the denominator are given by either applying the definition of contractiveness in the contractive case or non-expansiveness in the other two cases. To get the bounds in Table 2 when using the regression loss in the denominator, we first establish the inequality

$$\ell^{\text{fp}}_\theta(z) \leq 2\ell^{\text{reg}}_\theta(z), \quad (18)$$

for any non-expansive operator T_θ. This result follows from Lemma 4 since $r_\theta(z) \leq \ell^{\text{reg}}_\theta(z)$. The results in the contractive and linearly convergent cases follow from applying the definition of each and inequality (18). In the averaged case, we directly apply Ryu and Yin (2022, Theorem 1). Unless the operator is contractive, the results from Table 2 indicate that stronger bounds can be obtained from using the regression loss.

These bounds are particularly useful in the context of learning algorithm steps rather than initializations (Chen et al., 2022b; Amos, 2023). When replacing algorithm steps with learned variants, the implications of running more algorithm steps are uncertain. This is in contrast to our learned warm starts approach, where running more algorithm steps always leads to an improved performance. This is reflected in the guarantees on the fixed-point residual we can obtain after t steps, given the loss after k steps, where $t > k$. Moreover, if it is known that T_θ satisfies certain properties for all possible $\theta \in \Theta$ (*e.g.*, β-contractiveness) then the bounds in Table 2 will hold not only for a given θ, but for *every* $\theta \in \Theta$.

Table 2: Bounds for the ratios of testing at t steps and training at k steps. Here, we bound the ratio of the fixed-point residual after t steps and the loss after k steps where $t > k$. The value in the table provides the bound, *e.g.*, for a β-contractive operator, $\ell_\theta^{\rm fp}(T_\theta^t(z))/\ell_\theta^{\rm reg}(T_\theta^k(z)) \leq 2\beta^{t-k}$.

Operator	$\frac{\ell_\theta^{\rm fp}(T_\theta^t(z))}{\ell_\theta^{\rm fp}(T_\theta^k(z))}$	$\frac{\ell_\theta^{\rm fp}(T_\theta^t(z))}{\ell_\theta^{\rm reg}(T_\theta^k(z))}$
β-contractive	β^{t-k}	$2\beta^{t-k}$
β-linearly convergent	1	$2\beta^{t-k}$
α-averaged	1	$\sqrt{\frac{\alpha}{(1-\alpha)(t-k+1)}}$

5.2 Training for the fixed-point residual vs regression loss

The fixed-point residual (5) and regression (6) losses, align with the main distinction of learning methods mentioned in Amos (2023, Section 2.2) which splits between learning strategies that penalize suboptimality directly and those that penalize the distance to known ground truth solutions. The primary advantages of using our fixed-point residual loss are twofold: i) there is no need to compute a ground truth solution $z^\star(\theta)$ for each problem instance before training, and ii) the loss exactly corresponds to the evaluation metric, the fixed-point residual. On the other hand, there are two main advantages to using the regression loss: i) the regression loss uses the global information of the ground truth solution $z^\star(\theta)$, while the fixed-point residual loss exploits only local information, and ii) as mentioned in Section 5.1, stronger bounds on future iterations can be obtained when using the regression loss.

6. Numerical experiments

We now illustrate our method on examples of fixed-point algorithms from Table 1. We implemented our architecture in the JAX library (Bradbury et al., 2018) using the Adam (Kingma and Ba, 2015) optimizer to train. We use 10000 training problems and evaluate on 1000 test problems for the examples except the first one in Section 6.1. In our examples, we conduct a hyperparameter sweep over learning rates of either 10^{-3} or 10^{-4}, and architectures with $0, 1$, or 2 layers with 500 neurons each. We decay the learning rate by a factor of 5 when the training loss fails to decrease over a window of 10 epochs. All computations were run on the Princeton HPC Della Cluster and each example could be trained under 5 hours. The code to reproduce our results is available at

`https://github.com/stellatogrp/l2ws`.

Baselines. We compare our learned warm start, for both the fixed-point residual loss and the regression loss functions, against the following initialization approaches:

Cold start. We initialize the fixed-point algorithm for a test problem with parameter θ with the prediction $h_{w_{\text{cs}}}(\theta)$ where w_{cs} has been randomly initialized.

Nearest-neighbor warm start. The nearest-neighbor warm start initializes the test problem with an optimal solution of the nearest of the training problems measured by distance in terms of its parameter $\theta \in \mathbf{R}^d$. In most of our examples, the parametrized problems are sufficiently far apart so that the nearest-neighbor initializations do not significantly improve upon the cold start.

MAML initialization. The Model-Agnostic Meta-Learning (MAML) framework (Finn et al., 2017) learns a shared initialization for all the problems. We apply MAML with various values of k, namely $k = 0$, 1, 5, 15, and 60 while using the fixed-point residual loss. We then report the best MAML results out of these. Our method generalizes MAML by predicting instance-dependent initializations. We see in the experiments that the shared initializations from MAML cannot match the performance of our instance-dependent initializations.

In every experiment, we plot the average of the fixed-point residuals of the test problems for varying t as defined in Section 3.3. Additionally, we plot the average *gain* of each initialization relative to the cold start across the test problems. This gain for a given parameter θ corresponds to the ratio

$$\frac{\ell_\theta^{\text{fp}}(T_\theta^t(h_{w_{\text{cs}}}(\theta)))}{\ell_\theta^{\text{fp}}(T_\theta^t(h_w(\theta)))},$$

where h_w is the initialization technique in question and $h_{w_{cs}}$ is the cold-start predictor described above. Importantly, we code exact replicas of the OSQP and SCS algorithms in JAX. This allows us to input the learned warm starts into the corresponding C implementations; moreover, we report the solve times in milliseconds to reach various tolerances for the experiments we run with OSQP in Section 6.3 and SCS in Section 6.4.

6.1 Gradient descent

6.1.1 Unconstrained QP

We first consider a stylized example to illustrate why unrolling fixed-point steps can significantly improve over a decoupled approach, where $k = 0$. Consider the problem

$$\text{minimize} \quad (1/2)z^T P z + c^T z,$$

where $P \in \mathbf{S}_{++}^n$, and $c \in \mathbf{R}^n$ are the problem data and $z \in \mathbf{R}^n$ is the decision variable. The parameter is $\theta = c$.

Numerical example. We consider a small example where $n = 20$. We have a single hidden layer with 10 neurons, and 100 training problems. Let $P \in \mathbf{S}_{++}^n$ be a diagonal matrix where the first 10 diagonals take the value 100 and the last ten take the value of 1.

Let $\theta = c \in \mathbf{R}^n$. Here, the i-th index of θ is sampled according to the uniform distribution $\psi_i \mathcal{U}[-10, 10]$, where $\psi_i = 10000$ if $i \leq 10$ else 1. The idea is that the first 10 indices of the optimal solution $z^\star(\theta)$ vary much more than the last 10, but the first 10 indices of z will converge much faster. Note that there exists a linear mapping between the parameters and the optimal solution. Only including 10 neurons in the hidden layer prevents the training loss from achieving zero.

Results. Figure 4 and Table 3 show the convergence behavior of our method. The decoupled approaches prioritize minimizing the error to predict the first 10 indices and fail to improve on the cold start. By unrolling these gradient steps, our learning framework with $k > 0$ is able to adapt the warm start to take advantage of the downstream algorithm. These gains remain constant as the number of evaluation steps increases. We remark that both curves trained with $k = 0$ have a lower value for the fixed-point residual at $t = 0$. However, as t increases, their performance worsens compared to the cold start approach. This is attributed to the fact that training with $k = 0$ does not take into account the downstream algorithm. Consequently, despite an initially lower fixed-point residual for $k = 0$ trained curves, their performance declines relative to other methods, such as cold starts, for higher values of t.

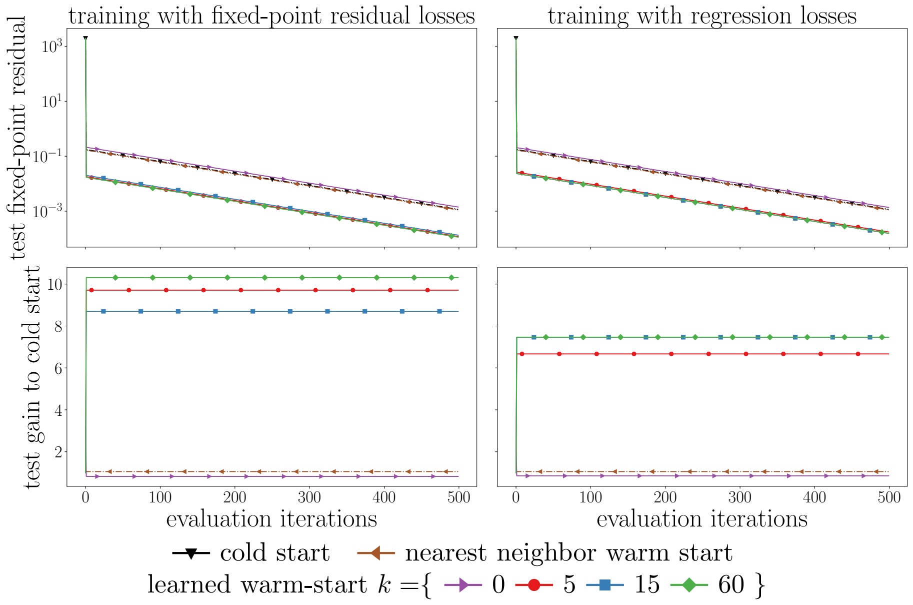


Figure 4: Unconstrained QP. All of the learned warm starts provide large improvements over the cold start and nearest neighbor initializations except for the ones learned with $k = 0$.

6.2 Proximal gradient descent

6.2.1 Lasso

We first consider the lasso problem

$$\text{minimize} \quad (1/2)\|Az - b\|_2^2 + \lambda \|z\|_1,$$

Table 3: Unconstrained QP.

(a) Mean iterations to reach a given fixed point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	57	51	56	76	**1**	**1**	**1**	**1**	80	**1**	**1**	**1**	**1**
0.01	286	280	285	305	59	59	70	53	309	**33**	97	86	86
0.001	515	510	514	534	288	289	299	283	538	**262**	326	315	315
0.0001	744	739	743	763	518	518	529	512	767	**491**	555	544	544

(b) Mean reduction in iterations from a cold start to a given fixed-point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	0	0.11	0.02	-0.33	**0.98**	**0.98**	**0.98**	**0.98**	-0.40	**0.98**	**0.98**	**0.98**	**0.98**
0.01	0	0.02	0.00	-0.07	0.79	0.79	0.76	0.81	-0.08	**0.88**	0.66	0.70	0.70
0.001	0	0.01	0.00	-0.04	0.44	0.44	0.42	0.45	-0.04	**0.49**	0.37	0.39	0.39
0.0001	0	0.01	0.00	-0.03	0.30	0.30	0.29	0.31	-0.03	**0.34**	0.25	0.27	0.27

where $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$, and $\lambda \in \mathbf{R}_{++}$ are problem data and $z \in \mathbf{R}^n$ is the decision variable. The parameter here is $\theta = b$.

Numerical example. We generate $A \in \mathbf{R}^{50 \times 100}$ with i.i.d entries drawn from $\mathcal{N}(0, 0.01)$ and pick $\lambda = 10$. We sample each b vector from the uniform distribution $\mathcal{U}[10, 30]$.

Results. Figure 5 and Table 4 show the convergence behavior of our method. While most of the learned warm starts significantly improve upon the baselines, the warm starts learned with $k = 60$ for both the regression loss and the fixed-point residual loss perform the best.

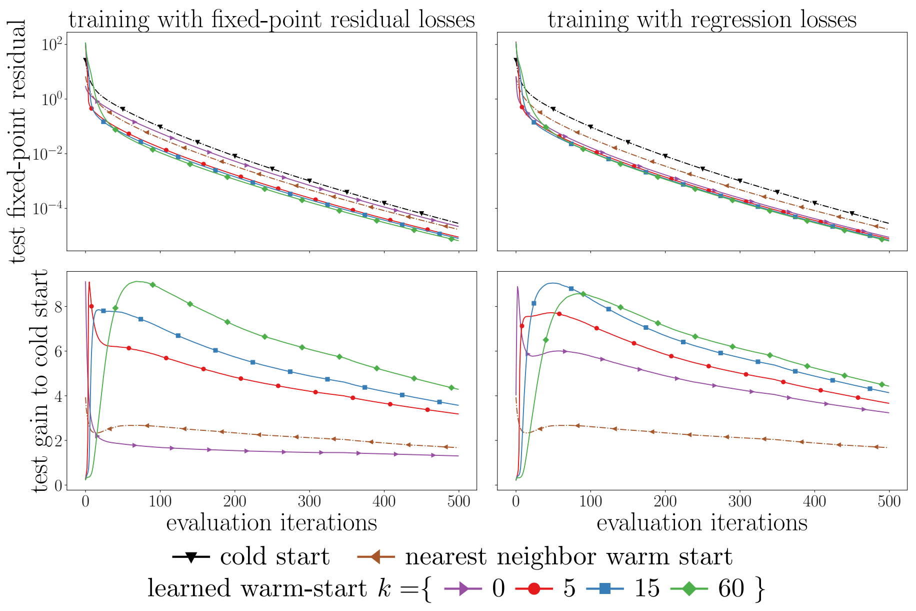


Figure 5: Lasso. All of the learned warm starts except for $k = 0$ with fixed-point residual loss significantly improve upon the baselines for any number of steps up to 500.

Table 4: Lasso.

(a) Mean iterations to reach a given fixed point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	99	65	63	79	51	40	34	36	41	39	34	**31**	39
0.01	192	153	149	172	135	121	113	**104**	121	118	112	106	105
0.001	301	262	257	282	242	226	218	**207**	225	223	217	210	**207**
0.0001	425	389	385	407	366	350	341	**329**	349	346	340	333	**329**

(b) Mean reduction in iterations from a cold start to a given fixed-point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	0	0.34	0.36	0.20	0.48	0.60	0.66	0.64	0.59	0.61	0.66	**0.69**	0.61
0.01	0	0.20	0.22	0.10	0.30	0.37	0.41	**0.46**	0.37	0.39	0.42	0.45	0.45
0.001	0	0.13	0.15	0.06	0.20	0.25	0.28	**0.31**	0.25	0.26	0.28	0.30	**0.31**
0.0001	0	0.08	0.09	0.04	0.14	0.18	0.20	**0.23**	0.18	0.19	0.20	0.22	**0.23**

6.3 OSQP

In this subsection, we apply our learning framework to the OSQP (Stellato et al., 2020) algorithm from Table 1 to solve convex quadratic programs (QPs).We compare solve times using OSQP code written in C for our learned warm starts against the baselines. Table 5 shows the sizes of the problems we run: model predictive control of a quadcopter in Section 6.3.1 and image deblurring in Section 6.3.2.

Table 5: Sizes of QPs from Table 1 that we use OSQP to solve. We give the number of primal constraints (m), size of the primal variable (n), and the parameter size, d.

	Quadcopter	Image deblurring
constraints m	600	2102
variables n	550	802
parameter size d	44	784

6.3.1 Model predictive control of a quadcopter

In our next example, we use model predictive control (Borrelli et al., 2017) to control a quadcopter to follow a reference trajectory. The idea of MPC is to optimize over a finite horizon length, but then to only implement the first control before optimizing again. Since the family of problems is sequential in nature, instead of MAML, we use the previous-solution baseline, where the warm start corresponds to the solution of the previous problem shifted by one time index. This initialization technique is commonly used in practice for control problems (Diehl et al., 2009).

We model the quadcopter as a rigid body controlled by four motors as in Song and Scaramuzza (2022). The state vector is $x = (p, v, q) \in \mathbf{R}^{n_x}$ where the state size is $n_x = 10$. The position vector $p = (p_x, p_y, p_z) \in \mathbf{R}^3$ and the velocity vector $v = (v_x, v_y, v_z) \in \mathbf{R}^3$ indicate the coordinates and velocities of the center of the quadcopter respectively. The vector $q = (q_w, q_x, q_y, q_z) \in \mathbf{R}^4$ is the quaternion vector indicating the orientation of the quadcopter. The inputs are $u = (c, \omega_x, \omega_y, \omega_z) \in \mathbf{R}^{n_u}$ where the input size is $n_u = 4$. The first input is the vertical thrust, and the last three are the angular velocities in the body frame. The dynamics are

$$\dot{p} = v, \qquad \dot{v} = \begin{bmatrix} 2(q_w q_y + q_x q_z)c \\ 2(q_w q_y + q_x q_z)c \\ (q_w^2 - q_x^2 - q_y^2 + q_z^2)c - g \end{bmatrix}, \qquad \dot{q} = \frac{1}{2}\begin{bmatrix} -w_x q_x - w_y q_y - w_z q_z \\ w_x q_w - w_y q_z + w_z q_y \\ w_x q_z + w_y q_w - w_z q_x \\ w_x q_y + w_y q_x + w_z q_w \end{bmatrix},$$

where g is the gravitational constant. At each time step, the goal is to track a reference trajectory given by $x^{\text{ref}} = (x_1^{\text{ref}}, \dots, x_T^{\text{ref}})$, while satisfying constraints on the states and the controls. We discretize the system with Δt and solve the QP

$$\begin{array}{ll} \text{minimize} & (x_T - x_T^{\text{ref}})^T Q_T (x_T - x_T^{\text{ref}}) + \sum_{t=1}^{T-1} (x_t - x_t^{\text{ref}})^T Q (x_t - x_t^{\text{ref}}) + u_t^T R u_t \\ \text{subject to} & x_{t+1} = A x_t + B u_t \quad t = 0, \dots, T-1 \\ & u_{\min} \leq u_t \leq u_{\min} \quad t = 0, \dots, T-1 \\ & x_{\min} \leq x_t \leq x_{\max} \quad t = 1, \dots, T \\ & |u_{t+1} - u_t| \leq \Delta u \quad t = 1, \dots, T-1. \end{array}$$

Here, the decision variables are the states $(x_1, \dots, x_T)$ where $x_t \in \mathbf{R}^{n_x}$ and the controls $(u_1, \dots, u_{T-1})$ where $u_t \in \mathbf{R}^{n_u}$. The dynamics matrices A and B are determined by linearizing the dynamics around the current state x_0, and the previous control input u_0 (Diehl et al., 2009). The matrices $Q, Q_T \in \mathbf{S}_+^{n_x}$ penalize the distance of the states to the reference trajectory, $(x_1^{\text{ref}}, \dots, x_T^{\text{ref}})$. The matrix $R \in \mathbf{S}_{++}^{n_u}$ regularizes the controls. The parameter is $\theta = (x_0, u_0, x_1^{\text{ref}}, \dots, x_T^{\text{ref}}) \in \mathbf{R}^{(T+1)n_x + n_u}$. We generate many different trajectories where the simulation length is larger than the time horizon T.

Numerical example. We discretize our continuous time model with a value of $\Delta t = 0.05$ seconds. The gravitational constant is 9.8. Each trajectory has a length of 100, and the horizon we consider at each timestep for each QP is $T = 10$. We use state bounds of $x_{\max} = -x_{\min} = (1, 1, 1, 10, 10, 10, 1, 1, 1, 1)$. We constrain the controls with $u_{\max} = (20, 6, 6, 6)$ and $u_{\min} = (2, -6, -6, -6)$, and set $\Delta u = (18, 6, 6, 6)$. For each simulation, the quadcopter is initialized at $p = v = 0$ and $q = (1, 0, 0, 0)$. We sample 5 waypoints for each of the (x, y, z) coordinates for each trajectory from the uniform distribution, $\mathcal{U}[-0.5, 0.5]$. Then we use a B-spline (de Boor, 1972) to smoothly interpolate between the waypoints to generate 100 points for the entire trajectory. Since each reference trajectory is made up of (x, y, z) coordinates rather than the full state vector, we shorten the parameter size to $\theta \in \mathbf{R}^{n_x + n_u + 3T}$.

Results. Figure 6 and Table 6 show the convergence behavior of our method. While all of the warm starts learned with the regression losses deliver substantial improvements over the baselines, our method using $k = 60$ with the fixed-point residual loss stands out as the best for a larger number of steps. To simulate a strict latency requirement, we also compare various initialization techniques in a closed-loop system where only 15 OSQP iterations are allowed per QP in Figure 7. The learned warm start approach can more accurately track the reference trajectory compared with the other two methods.

Table 6: Quadcopter.

(a) Mean iterations to reach a given fixed point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	Prev. sol.	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	103	26	28	86	115	**9**	10	34	14	27	14	15	39
0.01	682	127	115	696	689	480	79	50	**25**	268	26	26	54
0.001	2262	1210	1416	2582	2172	4268	5728	**277**	596	1588	673	604	636
0.0001	9958	6573	6906	9980	9494	12938	14000	**5681**	6041	7082	5957	6015	6053

(b) Mean reduction in iterations from a cold start to a given fixed-point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	Prev. sol.	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	0	0.75	0.73	0.17	-0.12	**0.91**	0.90	0.67	0.86	0.74	0.86	0.85	0.62
0.01	0	0.81	0.83	-0.02	-0.01	0.30	0.88	0.93	**0.96**	0.61	0.96	0.96	0.92
0.001	0	0.47	0.37	-0.14	0.04	-0.89	-1.53	**0.88**	0.74	0.30	0.70	0.73	0.72
0.0001	0	0.34	0.31	-0.00	0.05	-0.30	-0.41	**0.43**	0.39	0.29	0.40	0.40	0.39

(c) Mean solve times (in milliseconds) in OSQP with absolute and relative tolerances set to tol.

tol.	Cold Start	Near. Neigh.	Prev. sol.	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	0.19	0.17	0.17	0.32	0.24	0.17	0.18	0.19	**0.16**	0.17	**0.16**	**0.16**	0.18
0.01	2.44	0.23	0.21	2.32	2.16	0.68	0.25	0.21	**0.16**	0.25	0.17	**0.16**	0.28
0.001	12.0	4.69	6.33	11.77	11.4	17.65	35.89	**0.77**	1.18	6.06	1.23	1.23	1.45
0.0001	54.59	24.63	28.66	50.15	49.92	68.56	104.1	**14.34**	18.26	31.05	18.86	18.51	18.6
1e-05	114.6	76.03	78.21	103.4	109.3	128.5	174.6	**63.35**	65.23	80.27	67.0	65.88	65.43

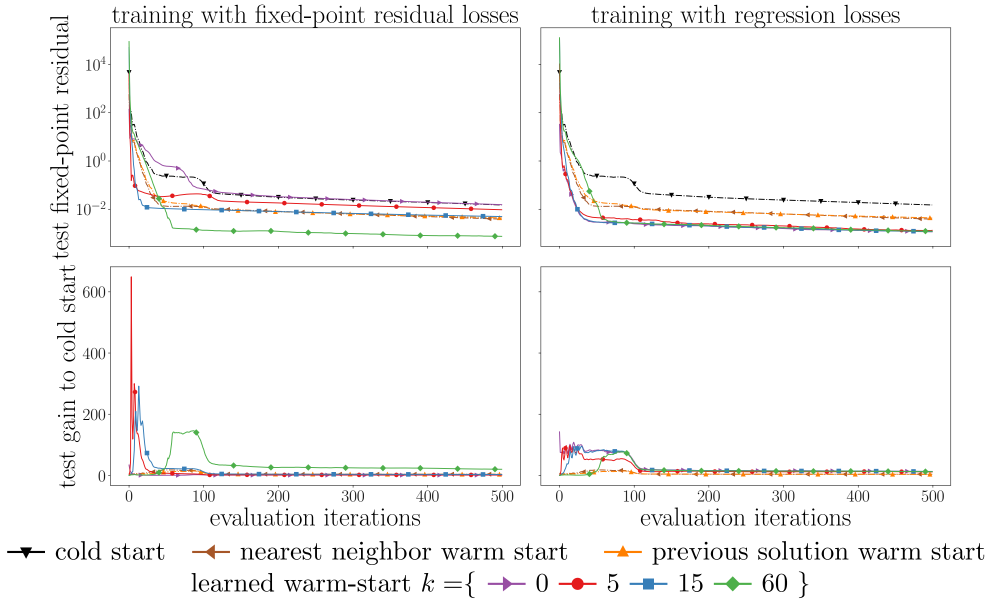
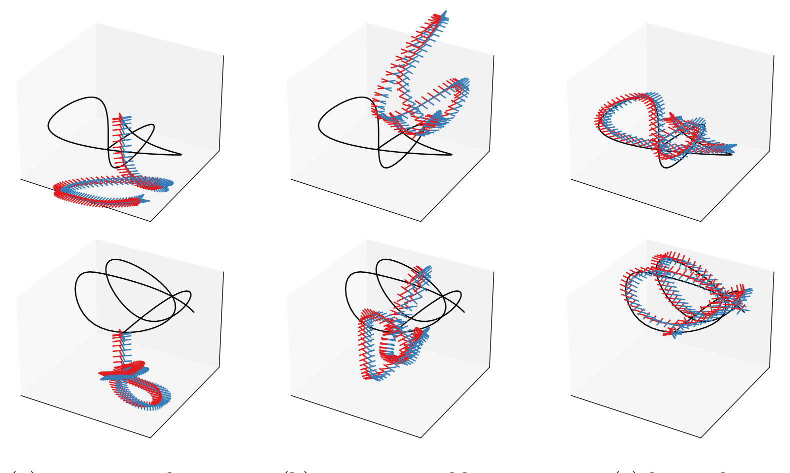


Figure 6: Quadcopter. Learned warm starts offer substantial improvements over the baselines. In particular, warm starts learned with $k = 60$ and the fixed-point residual loss have the largest gain for evaluation steps over about 50.

6.3.2 Image deblurring

We turn our attention to the task of image deblurring. Given a blurry image $b \in \mathbf{R}^n$, the goal is to recover the original image $x \in \mathbf{R}^n$. Both the noisy vector b and the target



(a) previous solution (b) nearest neighbor (c) learned

Figure 7: Visualizing closed-loop MPC of flying a quadcopter to track a reference trajectory. Each row corresponds to a different unseen reference trajectory. Each column uses a different initialization scheme to track the same unseen black reference trajectory in a closed-loop. Each technique is given a budget of 15 OSQP iterations to solve each QP. The learned approach which is trained on $k = 5$ with the regression loss tracks the trajectory well compared against the other two.

vector x are formed by stacking the columns of their respective images. We formulate this well-studied problem (Beck and Teboulle, 2009; Benvenuto et al., 2010) as

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2^2 + \lambda \|x\|_1 \\ \text{subject to} & 0 \leq x \leq 1. \end{array}$$

Here, the matrix $A \in \mathbf{R}^{n \times n}$ is the blur operator which represents a two-dimensional convolutional operator. The regularization hyperparameter $\lambda \in \mathbf{R}_{++}$, weights the fidelity term $\|Ax - b\|_2^2$, relative to the ℓ_1 penalty. The ℓ_1 penalty is used as it less sensitive to outliers and encourages sparsity (Beck and Teboulle, 2009). The constraints ensure that the deblurred image has pixel values within its domain.

Numerical example. We consider handwritten letters from the EMNIST dataset (Cohen et al., 2017). We apply a Gaussian blue of size 8 to each letter and then add i.i.d. Gaussian noise with standard deviation 0.001. The hyperparameter weighting term is $\lambda = 1e - 4$.

Results. Figure 8 and Table 7 show the convergence behavior of our method. Learned warm starts with the regression loss tend to outperform the learned warm starts with the fixed-point residual loss. We show visualizations of our method in Figure 9. For images that are particularly challenging, the image quality after 50 OSQP steps is significantly better for the learned warm start than the baseline initializations.

6.4 SCS

In this subsection, we apply our learning framework to the SCS (O'Donoghue et al., 2019) algorithm from Table 1 to solve convex conic optimization problems. We compare solve times using SCS code written in C for our learned warm starts against the baselines. We run our experiments on two second-order cone programs (SOCPs) in robust Kalman filtering

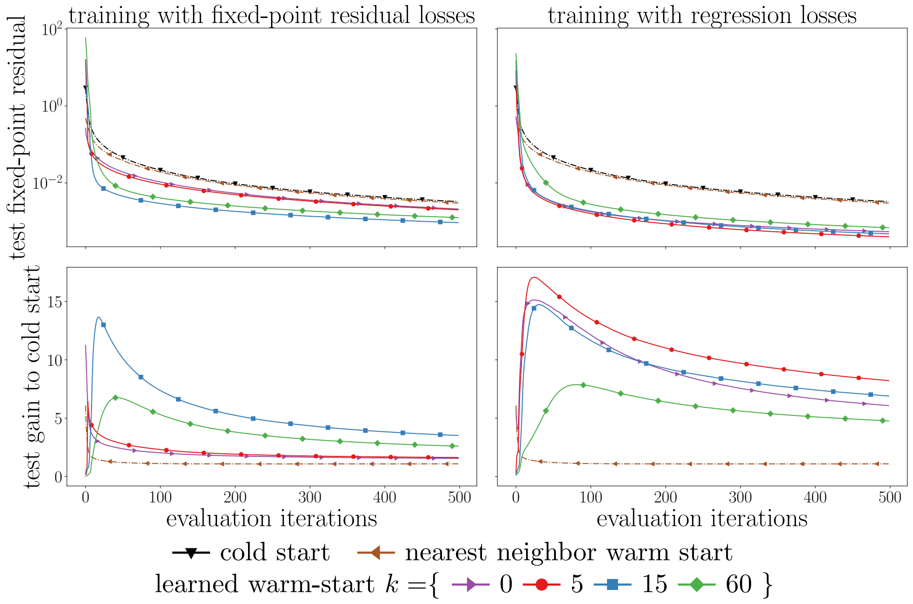


Figure 8: Image deblurring. Warm starts learned with the regression loss provide bigger gains compared with those learned with the fixed-point residual loss.

Table 7: Image deblurring.

(a) Mean iterations to reach a given fixed point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	24	16	23	6	7	**5**	8	12	**5**	**5**	6	8	11
0.01	194	181	188	107	102	93	17	36	16	16	**15**	18	42
0.001	1348	1253	1290	982	938	953	454	658	215	196	**171**	208	321
0.0001	7767	7607	7252	6613	6500	6689	5668	6812	4238	4038	**3292**	3779	4744

(b) Mean reduction in iterations from a cold start to a given fixed-point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	0	0.33	0.04	0.75	0.71	**0.79**	0.67	0.50	**0.79**	**0.79**	0.75	0.67	0.54
0.01	0	0.07	0.03	0.45	0.47	0.52	0.91	0.81	0.92	0.92	**0.92**	0.91	0.78
0.001	0	0.07	0.04	0.27	0.30	0.29	0.66	0.51	0.84	0.85	**0.87**	0.85	0.76
0.0001	0	0.02	0.07	0.15	0.16	0.14	0.27	0.12	0.45	0.48	**0.58**	0.51	0.39

(c) Mean solve times (in milliseconds) in OSQP with absolute and relative tolerances set to tol.

tol.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	6.34	6.37	6.46	6.32	6.35	6.31	**6.30**	6.32	6.33	6.36	6.34	6.32	6.33
0.01	9.13	9.62	8.42	**6.32**	6.34	6.33	**6.32**	6.33	6.33	6.40	6.33	6.35	6.33
0.001	62.37	68.25	60.14	48.00	46.52	38.51	9.90	18.38	9.51	9.14	**7.55**	8.21	15.54
0.0001	463.9	472.7	439.9	398.7	386.8	363.2	230.0	333.9	115.9	108.9	**80.76**	106.0	168.5
1e-05	3048	2996	2847	2629	2529	2777	2500	2934	1957	1891	**1691**	1831	2090

in Section 6.4.1 and robust non-negative least squares in Section 6.4.2 and two semidefinite programs (SDPs) in phase retrieval in Section 6.4.3 and sparse PCA in Section 6.4.4.

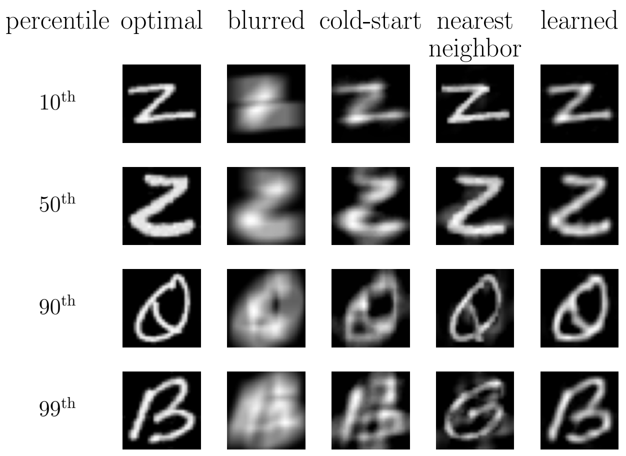


Figure 9: EMNIST image deblurring. Each row corresponds to an unseen sample from the EMNIST dataset. The last three columns depict several different initialization techniques after 50 OSQP steps. In the learned column, we use the regression loss with $k = 5$. To adjust the difficulty of the images displayed, we select images corresponding to different percentiles of distance from the nearest neighbor to the optimal solution.

Table 8: Sizes of conic problems from Table 1 that we use SCS to solve. We give the number of primal constraints (m), size of the primal variable (n), and the parameter size, d. Then, we provide the sizes of the cones for each conic program. For the second-order and the positive semidefinite cones, we supply arrays specifying the lengths of each respective cone. The notation $100 \times [3]$ means that there are 100 second-order cones each of size 3.

	Kalman filter	robust least squares	phase retrieval	sparse PCA
constraints m	600	2102	3480	4022
variables n	550	802	1600	2420
parameter size d	100	500	120	55
zero	600	0	240	1
non-negative	550	800	0	3201
second-order	$100 \times [3]$	[801,501]	0	0
positive semidefinite	0	0	[80]	[40]

6.4.1 Robust Kalman filtering

Kalman filtering (Kalman, 1960) is a widely used technique for predicting system states in the presence of noise in dynamic systems. In our first SOCP example, we use robust Kalman filtering (Xie and Soh, 1994) which mitigates the impact of outliers on the filtering process and model misspecifications to track a moving vehicle from noisy data location as in Venkataraman and Amos (2021). The dynamical system is modeled by

$$x_{t+1} = Ax_t + Bw_t, \quad y_t = Cx_t + v_t, \quad \text{for} \quad t = 0, 1, \dots, \tag{19}$$

where $x_t \in \mathbf{R}^{n_x}$ is the state, $y_t \in \mathbf{R}^{n_o}$ is the observation, $w_t \in \mathbf{R}^{n_u}$ is the input, and $v_t \in \mathbf{R}^{n_o}$ is a perturbation to the observation. The matrices $A \in \mathbf{R}^{n_x \times n_x}$, $B \in \mathbf{R}^{n_x \times n_u}$, and $C \in \mathbf{R}^{n_o \times n_x}$ give the dynamics of the system. Our goal is to recover the state x_t from the noisy measurements y_t. To do so, we solve the problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=1}^{T-1} \|w_t\|_2^2 + \mu\psi_\rho(v_t) \\ \text{subject to} & x_{t+1} = Ax_t + Bw_t \quad t = 0,\dots,T-1 \\ & y_t = Cx_t + v_t \quad t = 0,\dots,T-1. \end{array}$$

Here, the Huber penalty function (Huber, 1964) parametrized by $\rho \in \mathbf{R}_{++}$ that robustifies against outliers is

$$\psi_\rho(a) = \begin{cases} \|a\|_2 & \|a\|_2 \leq \rho \\ 2\rho\|a\|_2 - \rho^2 & \|a\|_2 \geq \rho \end{cases},$$

and $\mu \in \mathbf{R}_{++}$ weights this penalty term. The decision variables are the x_t's, w_t's, and v_t's. The parameters are the observed y_t's, *i.e.*, $\theta = (y_0, \dots, y_{T-1})$. In this example, we take advantage of rotational invariance of the problem. We rotate the noisy trajectory so that y_T is on the x-axis for every problem. After solving the transformed problem (for any initialization) we reverse the rotation to obtain the solution of the original problem.

Numerical example. As in Venkataraman and Amos (2021), we set $n_x = 4$, $n_o = 2$, $n_u = 2$, $\mu = 2$, $\rho = 2$, and $T = 50$. The dynamics matrices are

$$A = \begin{bmatrix} 1 & 0 & (1-(\gamma/2)\Delta t)\Delta t & 0 \\ 0 & 1 & 0 & (1-(\gamma/2)\Delta t)\Delta t \\ 0 & 0 & 1-\gamma\Delta t & 0 \\ 0 & 0 & 0 & 1-\gamma\Delta t \end{bmatrix}, B = \begin{bmatrix} 1/2\Delta t^2 & 0 \\ 0 & 1/2\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

where $\Delta t = 0.5$ and $\gamma = 0.05$ are fixed to be respectively the sampling time and the velocity dampening parameter. We generate the problem instances in the following way. We generate true trajectories $\{x_0^*, \dots, x_{T-1}^*\}$ of the vehicle by first letting $x_0^* = 0$. Then we sample the inputs as $w_t \sim \mathcal{N}(0, 0.01)$ and $v_t \sim \mathcal{N}(0, 0.01)$. The trajectories are then fully defined via the dynamics equations in Equation (19) with the sampled w_t's and v_t's.

Results. Since this is a control example, we use the shifted previous solution as a warm start from Section 6.3.1. Figure 10 and Table 9 show the convergence behavior of our method. In this example, the learned warm starts do well with the fixed-point residual loss for $k = 5$ and $k = 15$ and the regression loss for $k = 5$, but hardly improve in the other cases. In all cases, the gains relative to the cold start remain nearly constant throughout the evaluation iterations. Figure 11 illustrates how our learned solutions after 5 iterations outperforms the solution returned after 5 iterations from the baselines.

6.4.2 Robust non-negative least squares

We consider the problem of non-negative least squares with matrix uncertainty

$$\min_{x \geq 0} \quad \max_{\|\Delta A\| \leq \rho} \|(\hat{A} + \Delta A)x - b\|_2,$$

where the right-hand side vector $b \in \mathbf{R}^m$, nominal matrix $\hat{A} \in \mathbf{R}^{m \times n}$, and maximum perturbation $\rho \in \mathbf{R}_{++}$ are the problem data. The decision variable of the minimizer and maximizer are $x \in \mathbf{R}^n$ and $\Delta A \in \mathbf{R}^{m \times n}$ respectively. Here, $\|\Delta A\|$ denotes the largest

Table 9: Robust Kalman filtering.

(a) Mean iterations to reach a given fixed point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	Prev. sol.	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	40	36	37	34	29	**4**	8	35	24	9	9	33	35
0.01	90	85	86	87	82	33	**25**	81	77	41	27	81	81
0.001	142	139	140	140	135	84	**73**	133	131	93	76	134	133
0.0001	195	193	194	195	189	137	**126**	187	185	146	129	187	186

(b) Mean reduction in iterations from a cold start to a given fixed-point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	Prev. sol.	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	0	0.10	0.07	0.15	0.28	**0.9**	0.80	0.12	0.4	0.78	0.78	0.18	0.12
0.01	0	0.06	0.04	0.03	0.09	0.63	**0.72**	0.10	0.14	0.54	0.70	0.10	0.10
0.001	0	0.02	0.01	0.01	0.05	0.41	**0.49**	0.06	0.08	0.35	0.46	0.06	0.06
0.0001	0	0.01	0.01	0.00	0.03	0.30	**0.35**	0.04	0.05	0.25	0.34	0.04	0.05

(c) Mean solve times (in milliseconds) in SCS with absolute and relative tolerances set to tol.

tol.	Cold Start	Near. Neigh.	Prev. sol.	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	0.49	0.49	0.49	0.45	0.45	0.49	0.48	0.49	**0.35**	0.48	0.49	0.49	0.49
0.01	0.67	0.49	0.49	0.56	**0.45**	0.49	0.48	0.50	0.49	0.49	0.48	0.49	0.52
0.001	1.36	1.31	1.33	1.47	1.24	0.50	**0.48**	1.13	1.23	0.50	0.49	1.13	1.12
0.0001	2.23	2.19	2.22	2.33	2.08	1.18	**0.94**	1.97	2.12	1.36	1.02	1.99	1.96
1e-05	3.11	3.13	3.15	3.31	2.97	2.05	**1.76**	2.83	3.03	2.31	1.87	2.86	2.81

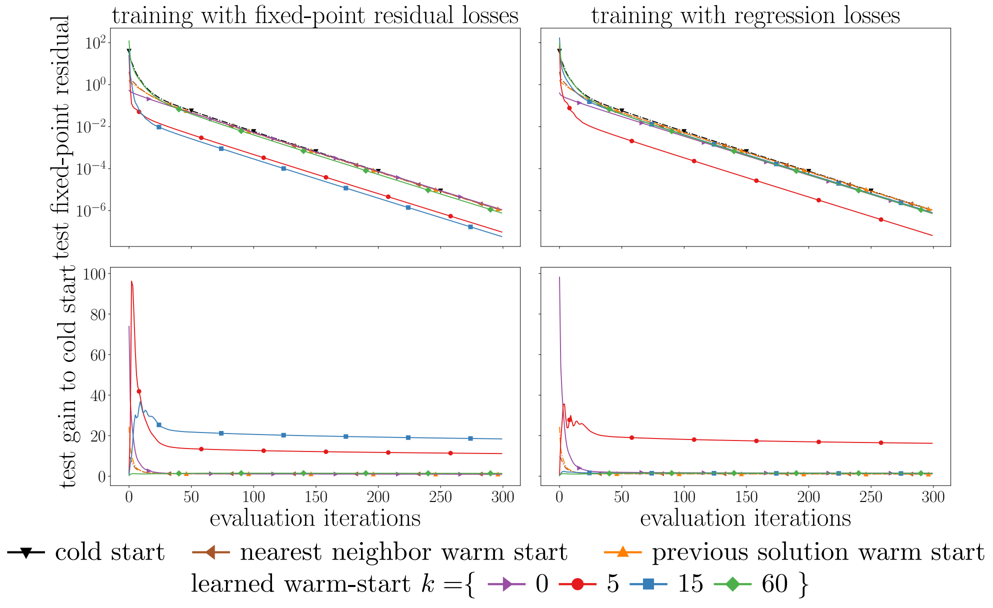


Figure 10: Robust Kalman filtering. The learned warm starts that train with $k = 5$ for both losses and with $k = 15$ for the fixed-point residual loss have significant gains over the baselines.

singular value of the perturbation matrix ΔA. El Ghaoui and Lebret (1997) provide an

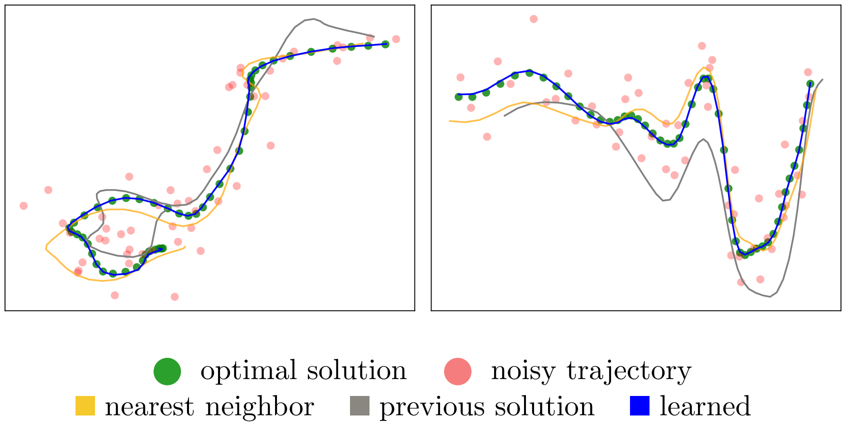


Figure 11: Visualizing test problems for robust Kalman filtering. Each plot is a separate test problem. The noisy, observed trajectory are the red points which serve as problem data for the SOCP. The robust Kalman filtering recovery, the optimal solution of the SOCP, is shown as green dots. After 5 iterations, SCS with our learned warm start using the regression loss with $k = 5$ is very close to the optimal solution while SCS initialized with both the shifted previous solution and the nearest neighbor still is noticeably far away from optimality.

SOCP formulation for this problem

$$\begin{array}{ll} \text{minimize} & u + \rho v \\ \text{subject to} & \|\hat{A}x - b\|_2 \leq u \\ & \|x\|_2 \leq v \\ & x \geq 0, \end{array}$$

where $x \in \mathbf{R}^n$, $u \in \mathbf{R}$, and $v \in \mathbf{R}$ are the decision variables. The parameter is $\theta = b$.

Numerical example. We pick $\rho = 4$ and $\hat{A} \in \mathbf{R}^{500 \times 800}$ where the entries of $\hat{A}$ are sampled the uniform distribution $\mathcal{U}[-1, 1]$. We sample b in an i.i.d. fashion from $\mathcal{U}[1, 2]$.

Results. Figure 12 and table 10 show the convergence behavior of our method. The learned warm starts with positive k substantially improve upon the baselines for both losses. Figure 12 show linear convergence of our method; this results in the gains from the learned warm starts staying roughly constant as the number of evaluation steps increases.

6.4.3 Phase retrieval

Our first SDP example is the problem of phase retrieval (Fienup, 1982) where the goal is to recover an unknown signal $x \in \mathbf{C}^n$ from observations. This problem has applications in X-ray crystallogpraphy (Millane, 1990) and coherent diffractive imaging (Shechtman et al., 2015). Specifically, for known vectors $a_i \in \mathbf{C}^n$, we have m scalar measurements: $b_i = |\langle a_i, x\rangle|^2, \quad i = 1, \dots, m$. Since the values are complex, we denote the conjugate transpose of a matrix A by A^*. Noting that $|\langle a_i, x\rangle|^2 = (a_i^* x)(x^* a_i)$, we introduce a matrix variable $X \in \mathbf{S}^n_+$ and matrices $A_i = a_i a_i^*$. The exact phase retrieval problem becomes a

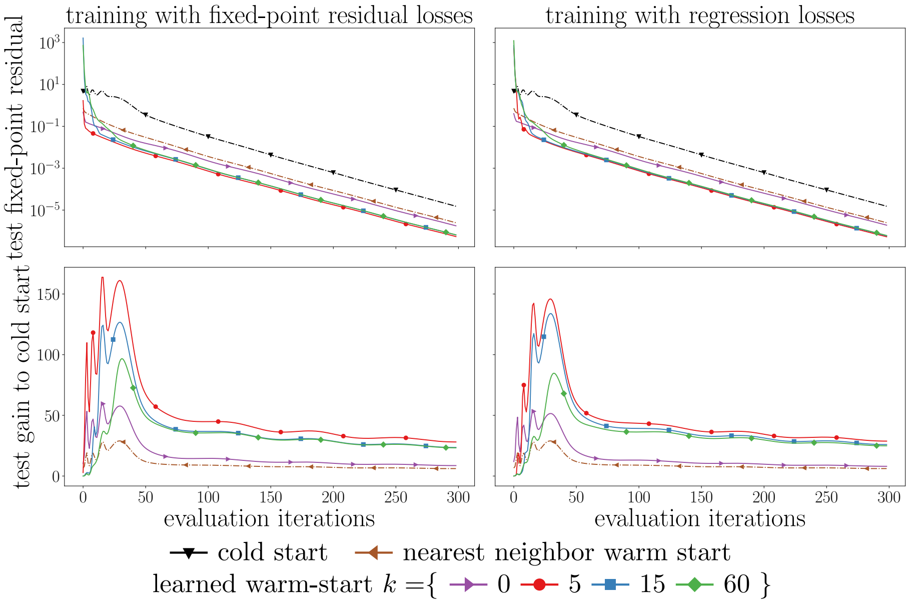


Figure 12: Robust non-negative least squares. All of the learned warm starts apart from the ones with $k = 0$ substantially improve the gain over the cold start.

Table 10: Robust non-negative least squares.

(a) Mean iterations to reach a given fixed point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	76	26	28	12	78	**3**	12	19	15	5	8	14	20
0.01	131	78	63	65	133	**37**	42	45	68	41	39	41	46
0.001	189	136	120	124	192	**94**	99	99	127	97	95	97	99
0.0001	249	197	182	186	252	**156**	160	161	189	158	**156**	158	160

(b) Mean reduction in iterations from a cold start to a given fixed-point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	0	0.66	0.63	0.84	-0.03	**0.96**	0.84	0.75	0.8	0.93	0.89	0.82	0.74
0.01	0	0.40	0.52	0.50	-0.02	**0.72**	0.68	0.66	0.48	0.69	0.70	0.69	0.65
0.001	0	0.28	0.37	0.34	-0.02	**0.50**	0.48	0.48	0.33	0.49	0.50	0.49	0.48
0.0001	0	0.21	0.27	0.25	-0.01	**0.37**	0.36	0.35	0.24	0.37	**0.37**	0.37	0.36

(c) Mean solve times (in milliseconds) in SCS with absolute and relative tolerances set to tol.

tol.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	61.79	4.16	22.05	**2.23**	74.1	22.13	26.95	26.73	2.32	26.44	22.44	22.39	22.14
0.01	101.73	46.01	42.14	42.08	121.7	**24.13**	31.61	36.37	42.59	38.59	26.17	25.47	37.31
0.001	141.5	101.7	84.50	89.95	169.9	**61.95**	79.9	76.75	95.76	77.25	63.65	64.69	67.32
0.0001	185.4	145.5	137.8	148.4	225.2	**107.7**	142.3	149.7	143.7	138.2	112.3	115.1	119.7
1e-05	241.1	199.6	184.2	187.8	287.5	169.5	197.3	198.4	195.2	194.2	168.4	167.4	**164.6**

feasibility problem over the matrix variable with a rank constraint

$$\begin{array}{ll} \text{find} & X \\ \text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \dots, m \\ & \mathbf{rank}(X) = 1, \quad X \succeq 0. \end{array}$$

We arrive at the following SDP relaxation by dropping the rank constraint:

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(X) \\ \text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \dots, m \\ & X \succeq 0. \end{array}$$

To parameterize each problem, we let $\theta = b \in \mathbf{R}^m$.

Numerical example. For the signal, we sample x from a complex normal distribution, *i.e.*, we sample the real and imaginary parts of each component independently from $\mathcal{N}(\mu, \sigma^2)$. To construct the constraint matrices, we use the coded diffraction pattern model (Candès et al., 2015). The specific modulating waveforms follow the setup from Yurtsever et al. (2021, Section F.1). For a signal of size n, we generate $d = 3n$ measurements. Specifically, we draw 3 independent modulating waveforms $\psi_j \in \mathbf{R}^n$, $j = 1, \dots, 3$. Each component of ψ_j is the product of two random variables, with one drawn uniformly from $\{1, i, -1, -i\}$ and the other drawn from $\{\sqrt{2}/2, \sqrt{3}\}$ with probabilities 0.8 and 0.2, respectively. Then, each a_i corresponds to computing a single entry of the Fourier transform of x after being modulated by the waveforms. Letting W be the $n \times n$ discrete Fourier transform matrix, the a_i's can be written explicitly as $a_{(j-1)n+l} = W_l^T(\mathbf{diag}(\psi_j))^\star$ where W_l^T is the l-th row of W. We take $n = 40$, $\mu = 5$, and $\sigma = 1$.

Results. Figure 13 and Table 11 show the convergence behavior of our method. In this case, while the decoupled approach with $k = 0$ offers the largest gains over the first few iterations, the gain degrades as t increases to the point where it's performance becomes worse than that of the nearest-neighbor initialization. The learned warm starts with the regression loss for positive k tend to sustain their gains for a larger value of t compared with the learned warm starts that use the fixed-point residual loss.

6.4.4 Sparse PCA

Next, we examine the problem of sparse PCA (Zou et al., 2006). Unlike standard PCA (Jolliffe, 2005), which typically finds principal components that depend on all observed variables, sparse PCA identifies principal components that rely on only a small subset of the variables. The Sparse PCA problem is

$$\begin{array}{ll} \text{maximize} & x^T A x \\ \text{subject to} & \|x\|_2 \le 1 \\ & \mathbf{card}(x) \le c, \end{array} \tag{20}$$

where $x \in \mathbf{R}^n$ is the decision variable and $\mathbf{card}(x)$ is the number of nonzero terms of vector x. The covariance matrix $A \in \mathbf{S}_+^n$ and desired cardinality $c \in \mathbf{R}_+$ are problem data. We consider an SDP relaxation of the non-convex problem (20) which takes the form

$$\begin{array}{ll} \text{maximize} & \mathbf{tr}(AX) \\ \text{subject to} & \mathbf{tr}(X) = 1 \\ & \mathbf{1}^T |X| \mathbf{1} \le c \\ & X \succeq 0, \end{array} \tag{21}$$

Table 11: Phase retrieval.

(a) Mean iterations to reach a given fixed point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	531	254	324	320	327	280	209	**76**	322	140	122	115	129
0.01	2558	1629	2368	2259	2186	2710	2890	1982	2249	817	**666**	669	686
0.001	6478	5547	6305	6133	5965	6803	7113	6127	6098	3762	**3609**	3611	3667
0.0001	11512	10837	11482	11281	11009	12021	12379	11303	11223	8617	8322	8268	**8217**

(b) Mean reduction in iterations from a cold start to a given fixed-point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	0	0.52	0.39	0.40	0.38	0.47	0.61	**0.86**	0.39	0.74	0.77	0.78	0.76
0.01	0	0.36	0.07	0.12	0.15	-0.06	-0.13	0.23	0.12	0.68	**0.74**	0.74	0.73
0.001	0	0.14	0.03	0.05	0.08	-0.05	-0.10	0.05	0.06	0.42	**0.44**	0.44	0.43
0.0001	0	0.06	0.00	0.02	0.04	-0.04	-0.08	0.02	0.03	0.25	0.28	0.28	**0.29**

(c) Mean solve times (in milliseconds) in SCS with absolute and relative tolerances set to tol.

tol.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	72.59	48.5	48.68	35.34	35.52	**34.92**	35.33	35.19	35.27	35.15	35.14	35.23	35.39
0.01	490.6	257.6	439.2	495.5	480.6	512.8	364.7	**118.4**	500.3	153.7	140.4	127.8	146.9
0.001	3229	1978	2759	3152	3037	3732	3925	2813	3111	1175	**1001**	1043	1046
0.0001	8358	6365	6901	7965	8018	8829	9190	7882	7924	5151	**4887**	5003	5071
1e-05	12673	11113	10413	12236	12667	12514	12698	12490	12231	11104	10954	**10844**	10864

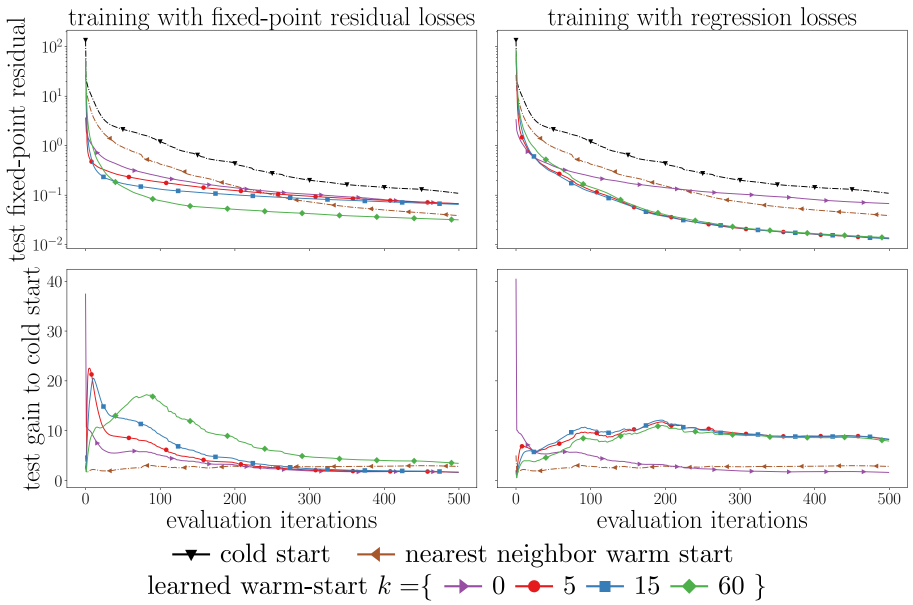


Figure 13: Phase retrieval. Other than the $k = 0$ case, the learned warm starts with regression loss improvements are maintained for many evaluation steps.

where the decision variable is $X \in \mathbf{S}^n_+$. We use an r-factor model (Boyd et al., 2017) and set $A = F\Sigma F^T$ where $F \in \mathbf{R}^{n\times r}$ is the factor loading matrix and $\Sigma \in \mathbf{S}^n_+$ is a matrix that holds the factor scores. The parameter is $\theta = \mathbf{vec}(\Sigma)$.

Numerical example. We run our experiments with matrix size of $n = 40$, a factor size of $r = 10$, and a cardinality size of $c = 10$. To generate the covariance matrices, we first generate a random nominal matrix A_0, whose entries are sampled as an i.i.d. standard Gaussian. We then take the singular value decomposition of $A_0 = U\Sigma_0 U^T$, and let the shared factor loading matrix $F \in \mathbf{R}^{n \times r}$ be the first r singular vectors of U. Let $B_0 \in \mathbf{S}^r_+$ be the diagonal matrix found by taking the square root of the first r singular values of A_0. Then, for each problem, we take $\Sigma = BB^T$ where $B = \Delta + B_0$. Here, the elements of $\Delta \in \mathbf{R}^{r \times r}$ are sampled i.i.d. from the uniform distribution $\mathcal{U}[-0.1, 0.1]$.

Table 12: Sparse PCA.

(a) Mean iterations to reach a given fixed point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	26	1	11	**0**	1	2	8	10	**0**	3	5	8	17
0.01	122	62	63	262	130	92	70	38	262	40	**33**	37	55
0.001	338	269	262	491	370	313	289	160	490	171	**145**	151	172
0.0001	982	822	785	1002	1000	881	935	738	1004	712	**681**	698	709

(b) Mean reduction in iterations from a cold start to a given fixed-point residual (Fp res.)

Fp res.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	0	0.96	0.58	**1.0**	0.96	0.92	0.69	0.62	**1.0**	0.88	0.81	0.69	0.35
0.01	0	0.49	0.48	-1.15	-0.07	0.25	0.43	0.69	-1.15	0.67	**0.73**	0.7	0.55
0.001	0	0.20	0.22	-0.45	-0.09	0.07	0.14	0.53	-0.45	0.49	**0.57**	0.55	0.49
0.0001	0	0.16	0.20	-0.02	-0.02	0.10	0.05	0.25	-0.02	0.27	**0.31**	0.29	0.28

(c) Mean solve times (in milliseconds) in SCS with absolute and relative tolerances set to tol.

tol.	Cold Start	Near. Neigh.	MAML	Fp $k=0$	Fp $k=1$	Fp $k=5$	Fp $k=15$	Fp $k=60$	Reg $k=0$	Reg $k=1$	Reg $k=5$	Reg $k=15$	Reg $k=60$
0.1	9.66	**0.59**	7.38	0.65	10.56	10.62	8.35	8.22	0.73	10.55	10.06	8.27	8.4
0.01	26.34	9.63	**7.81**	97.97	30.58	14.31	8.81	8.22	88.35	9.91	10.09	8.25	9.94
0.001	67.88	45.91	39.84	121.9	83.92	71.17	43.23	17.23	110.4	20.18	20.09	**16.92**	25.31
0.0001	136.2	107.6	90.61	187.4	155.7	140.6	100.7	63.95	170.0	76.57	73.65	**61.11**	70.46
1e-05	255.5	204.3	166.0	309.5	272.7	264.9	201.3	151.2	279.2	189.1	174.6	**144.5**	153.8

Results. Figure 14 and table 12 show the convergence behavior of our method. In this example, both the fixed-point residual loss and the regression loss perform with $k = 0$. All of the other learned warm starts with the regression loss and some with the fixed-point residual loss show good performance.

7. Conclusion

We present a machine-learning framework to warm-start fixed-point algorithms for parametric convex optimization. The architecture first maps the problem parameters to initial warm start iterates, and then runs k fixed-point algorithm steps. We propose two different loss functions, a regression loss and a fixed-point residual loss. Training the warm start predictor with gradient-based methods amounts to backpropagating through the fixed-point steps. In this way, the learned initializations are tailored for the downstream fixed-point algorithm. To provide guarantees to unseen data, we combine the PAC-Bayes framework with operator theory and obtain guarantees when the fixed-point operator is contractive, linearly convergent, and averaged. We showcase the effectiveness of this method on a wide

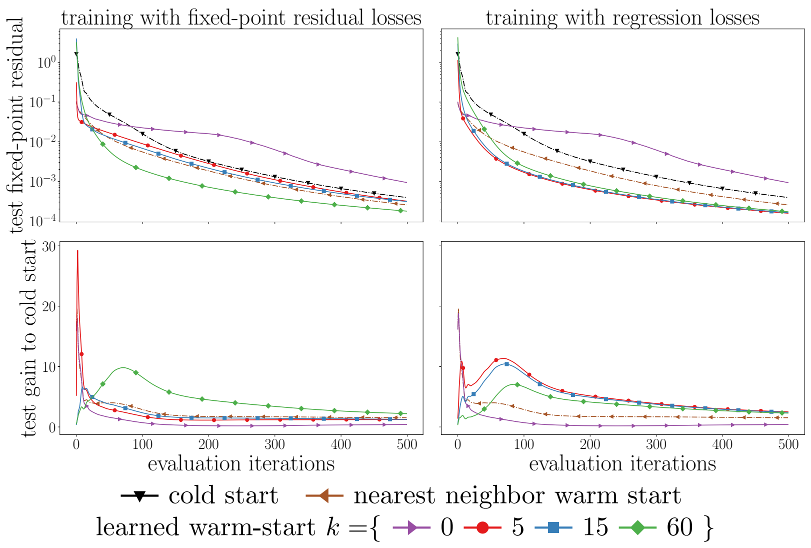


Figure 14: Sparse PCA. The learned warm starts with positive k that use the regression loss provide large gains.

variety of fixed-point algorithms including gradient descent, proximal gradient descent, and ADMM.

Future research directions of interest include exploring different predictors for the initial iterates, developing warm start strategies for non-convex optimization, and investigating methods to scale our computational architecture to large-scale problems.

Appendix A. Examples of fixed-point algorithms

Gradient descent. Here, $z \in \mathbf{R}^n$ is the decision variable and f_θ is a convex and L-smooth function. Recall that $f : \mathbf{R}^n \to \mathbf{R}$ is L-smooth if $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2 \quad \forall x, y \in \mathbf{R}^n$. If $\alpha \in (0, 2/L)$, then the iterates of gradient descent are guaranteed to converge to an optimal solution (Ryu and Boyd, 2016). If f_θ is strongly convex, then the fixed-point operator is a contraction (Ryu and Boyd, 2016).

Proximal gradient descent. Here, $z \in \mathbf{R}^n$ is the decision variable, f_θ is a convex and L-smooth function, and g_θ is a convex but possibly non-smooth function. The iterations of proximal gradient descent converge to a solution if $\alpha \in (0, 2/L)$ (Parikh and Boyd, 2014).

Alternating direction method of multipliers (ADMM). Here, $u \in \mathbf{R}^n$ is the decision variable and f_θ and g_θ are closed, convex, proper, and possibly non-smooth functions. The iterations of ADMM generate a sequence of iterates, resulting in the convergence of both $\tilde{u}^i$ and u^i to each other and to a solution of the problem. The $z \in \mathbf{R}^n$ variable serves as the associated dual variable. We use the equivalence of ADMM to Douglas-Rachford splitting (Gabay, 1983) and write the Douglas-Rachford splitting iterations in Table 1. While the associated fixed-point operator to ADMM is averaged (Ryu and Boyd, 2016), ADMM is known to converge linearly under certain conditions (Eckstein, 1989; Giselsson and Boyd, 2017).

OSQP. The operator splitting quadratic program (OSQP) (Stellato et al., 2020) solver is based on ADMM. Here, $P \in \mathbf{S}^n_+$, $A \in \mathbf{R}^{m \times n}$, $c \in \mathbf{R}^n$, $l \in \mathbf{R}^m$, and $u \in \mathbf{R}^m$ are problem data, and $\Pi_{[l,u]}$ is the projection onto the box $[l,u]$. The decision variable is $x \in \mathbf{R}^n$. The algorithm steps in Table 1 are given by Banjac et al. (2019).

SCS. The splitting conic solver (SCS) (O'Donoghue, 2021) is also based on ADMM. Here, $P \in \mathbf{S}^n_+$, $A \in \mathbf{R}^{m \times n}$, $c \in \mathbf{R}^n$, and $b \in \mathbf{R}^m$ are problem data, and $\Pi_{\mathcal{C}}$ is the projection onto the cone $\mathcal{C}$. The decision variables are $x \in \mathbf{R}^n$ and $s \in \mathbf{R}^m$. For simplicity, Table 1 includes the simplified version of the SCS algorithm without the homogeneous self-dual embedding. The SCS algorithm, the one we use in the numerical experiments in Section 6.4, is based on the homogeneous self-dual embedding; see O'Donoghue (2021) for the details. As in Venkataraman and Amos (2021), our implementation normalizes the fixed-point residual by the τ scaling factor to ensure that the fixed-point residual is not artificially small.

Appendix B. Proofs

B.1 Proof of Lemma 1

Let $w' = w + u$ and let S_w be the set of perturbations w' such that

$$S_w \subset \{w' \mid \max_{\theta \in \Theta} \|h_{w'}(\theta) - h_w(\theta)\|_2 \leq \gamma/2\}.$$

Let q be the probability density function over w'. We construct a new distribution $\tilde{Q}$ over predictors $h_{\tilde{w}}$ where $\tilde{w}$ is restricted to S_w with the probability density function $\tilde{q}(\tilde{w}) = (1/Z)q(\tilde{w})$ if $\tilde{w} \in S_w$ and otherwise 0, where Z is a normalizing constant. By the assumption of the lemma, $Z = \mathbf{P}(w' \in S_w) \geq 1/2$. By the definition of $\tilde{Q}$, we have

$$\max_{\theta \in \Theta} \|h_{\tilde{w}}(\theta) - h_w(\theta)\|_2 \leq \gamma/2.$$

Therefore, $\ell^{\rm fp}_\theta(T^t_\theta(h_w(\theta))) \leq g^t_{\gamma/2,\theta}(h_{\tilde{w}}(\theta)) \leq g^t_{\gamma,\theta}(h_w(\theta))$ almost surely for every $\theta \in \Theta$. Hence, for every $\tilde{w}$ drawn from the probability density function $\tilde{Q}$, almost surely,

$$R^t(w) \leq R^t_{\gamma/2}(\tilde{w}), \quad \hat{R}^t_{\gamma/2}(\tilde{w}) \leq \hat{R}^t_\gamma(w). \tag{22}$$

Now using these two inequalities above and the PAC-Bayes theorem, we get

$$\begin{aligned}
R^t(w) &\leq \mathbf{E}_{\tilde{w}}[R^t_{\gamma/2}(\tilde{w})] \\
&\leq \mathbf{E}_{\tilde{w}}[\hat{R}^t_{\gamma/2}(\tilde{w})] + 2C_{\gamma/2}(t)\sqrt{(2KL(\tilde{w}||\pi) + \log(2N/\delta))/(N-1)} \\
&\leq \hat{R}^t_\gamma(w) + 2C_{\gamma/2}(t)\sqrt{(2KL(\tilde{w}||\pi) + \log(2N/\delta))/(N-1)} \\
&\leq \hat{R}^t_\gamma(w) + 4C_{\gamma/2}(t)\sqrt{(2KL(w'||\pi) + \log(6N/\delta))/(N-1)}.
\end{aligned}$$

The first and third inequalities come from (22), and the second inequality follows from (11). The last inequality comes from the following calculation which we repeat from Neyshabur et al. (2018, Section 4). Let S^c_w denote the complement of S_w and $\tilde{q}^c$ denote the density function q restricted to S^c_w and normalized. Then we get

$$\mathrm{KL}(q||p) = Z\mathrm{KL}(\tilde{q}||p) + (1-Z)\mathrm{KL}(\tilde{q}^c||p) - H(Z),$$

where $H(Z) = -Z \log Z - (1-Z)\log(1-Z)$ is the binary entropy function. Since the KL-divergence is always positive,

$$\mathrm{KL}(\tilde{q}||p) = [\mathrm{KL}(q||p) + H(Z) - (1-Z)KL(\tilde{q}^c||p)]/Z \leq 2(\mathrm{KL}(q||p) + 1).$$

Using the additive properties of logarithms, $1 + \log(2N/\delta) \leq \log(6N/\delta)$.

B.2 Proof of Theorem 2

Our proof follows a similar structure as the proof of Neyshabur et al. (2018, Theorem 1). Let $\zeta = (\Pi_{i=1}^L \|W_i\|_2)^{1/L}$ and consider a neural network with weights $\tilde{W}_i = \zeta W_i/\|W_i\|_2$. Due to the homogeneity of the ReLU, we have $h_w(\theta) = h_{\tilde{w}}(\theta)$ for all $\theta \in \Theta$ (Neyshabur et al., 2018). Since $(\Pi_{i=1}^L \|W_i\|_2)^{1/L} = (\Pi_{i=1}^L \|\tilde{W}_i\|_2)^{1/L}$ and $\|W_i\|_F/\|W_i\|_2 = \|\tilde{W}_i\|_F/\|\tilde{W}_i\|_2$, inequality (13) is the same for w and $\tilde{w}$. Therefore, it is sufficient to prove the theorem only for the normalized weights $\tilde{w}$ and we can assume that the spectral norm of the weight matrix is equal across all layers, *i.e.*, $\|W_i\|_2 = \zeta$. Now, we break our proof into two cases depending on the product of the spectral norm of the weight matrices. The main difference between our proof and the proof for Neyshabur et al. (2018, Theorem 1) is that we introduce a secondary case. The main case analysis is similar.

Main case. In the main case, $\zeta^L \geq \gamma/(2B)$. We choose the prior distribution π to be $\mathcal{N}(0, \sigma^2)$ and consider the perturbation $u \sim \mathcal{N}(0, \sigma^2)$. As in Neyshabur et al. (2018), since the prior distribution π cannot depend on ζ, we consider predetermined values of $\tilde{\zeta}$ on a grid and then do a union bound. For now, we consider $\tilde{\zeta}$ fixed and consider all ζ such that $|\zeta - \tilde{\zeta}| \leq \zeta/L$. This ensures that each relevant value of ζ is covered by some $\tilde{\zeta}$ on the grid. Since $|\zeta - \tilde{\zeta}| \leq \zeta/L$ we get the inequalities

$$\zeta^{L-1}/e \leq \tilde{\zeta}^{L-1} \leq e\zeta^{L-1}. \tag{23}$$

This follows from the inequalities $(1+1/x)^{x-1} \leq e$ and $1/e \leq (1-1/x)^{x-1}$ which themselves are consequences the inequality $1 + y \leq e^y$ for all y. Since the entries of each U_i are drawn from $\mathcal{N}(0, \sigma^2)$, we have the bound on the spectral norm of each U_i (Tropp, 2011)

$$\mathbf{P}_{U_i \sim \mathcal{N}(0,\sigma^2)}(\|U_i\|_2 > t) \leq 2\bar{h}e^{-t^2/(2\bar{h}\sigma^2)}.$$

We can take a union bound to get

$$\mathbf{P}_{U_1,\dots,U_L \sim \mathcal{N}(0,\sigma^2)}(\|U_1\|_2 \leq t, \dots, \|U_L\|_2 \leq t) \geq 1 - 2L\bar{h}e^{-t^2/(2\bar{h}\sigma^2)}. \tag{24}$$

By setting the right hand side of (24) to 1/2, we establish that with probability at least 1/2, the spectral norm of every perturbation U_i is bounded by $\sigma\sqrt{2\bar{h}\log(4L\bar{h})}$ simultaneously. We choose $\sigma = \gamma/(21LB\tilde{\zeta}^{L-1}\sqrt{\bar{h}\log(4\bar{h}L)})$ and now verify that with probability at least 1/2, $\|U_i\|_2 \leq \|W_i\|_2/L = \zeta/L$ holds, a condition of Neyshabur et al. (2018, Lemma 2):

$$\begin{aligned}\|U_i\|_2 &\leq \sigma\sqrt{2\bar{h}\log(4L\bar{h})} = \gamma\sqrt{2}/(21LB\tilde{\zeta}^{L-1}) \\ &\leq e2\sqrt{2}\gamma/(42LB\zeta^{L-1}) \leq 2\sqrt{2}e\zeta/(21L) \leq \zeta/L.\end{aligned}$$

In the first line, the inequality comes from the perturbation bound on $\|U_i\|_2$, and the equality follows from plugging in for σ. The second line follows from (23), and the assumption from the main case that $\zeta^L > \gamma/(2B)$. Now that the conditions are met, we apply Neyshabur et al. (2018, Lemma 2). The following holds with probability at least 1/2:

$$\begin{aligned}\max_{\theta\in\Theta} \|h_w(\theta) - h_{w+u}(\theta)\|_2 &\leq eB\zeta^{L-1} \textstyle\sum_{i=1}^L \|U_i\|_2 \\ &\leq e^2 LB\tilde{\zeta}^{L-1}\sigma\sqrt{2\bar{h}\log(4L\bar{h})} \leq \gamma/2.\end{aligned}$$

In the second inequality, we use (23). The last inequality follows from the choice of σ. Now we calculate the KL-term with $\pi \sim \mathcal{N}(0,\sigma^2)$ and u chosen with the above value of σ:

$$\begin{aligned}\mathrm{KL}(w+u||\pi) &\leq \tfrac{\|w\|_2^2}{2\sigma^2} = \tfrac{21^2 L^2 B^2 \tilde{\zeta}^{2L-2}\bar{h}\log(4\bar{h}L)}{2\gamma^2} \textstyle\sum_{i=1}^L \|W_i\|_F^2 \\ &\leq \tfrac{21^2\zeta^{2L}}{2\gamma^2} B^2L^2\bar{h}\log(4L\bar{h}) \textstyle\sum_{i=1}^L \tfrac{\|W_i\|_F^2}{\zeta^2}.\end{aligned} \tag{25}$$

What remains is to take a union bound over the different choices of $\tilde{\zeta}$. We only need to consider values of ζ in the range of

$$(\gamma/(2B))^{1/L} \leq \zeta \leq (\gamma\sqrt{N}/(2B))^{1/L}. \tag{26}$$

Since we are in the main case, we do not have to consider $\zeta^L < \gamma/(2B)$. Alternatively, if $\zeta^L > \gamma\sqrt{N}/(2B)$, then the upper bound on the KL term in (25) is greater than N. To see this, first note that the frobenius norm is always at least the operator norm of a given matrix, so $\|W_i\|_F \geq \zeta$ for $i = 1,\dots,L$. Then, the right hand side of (25) becomes at least $21^2L^2\bar{h}\log(4L\bar{h})N/8$ which is greater than N. Theorem 2 is obtained by using the bound in the right hand side of (25) for the KL term in Lemma 1. Therefore Theorem 2 holds trivially since $C_{\gamma/2}(t)$ upper bounds $R^t(w)$ and the entire square root term in Lemma 1 is at least one. Hence, we only need to consider ζ in the range of (26).

The condition $L|\tilde{\zeta} - \zeta| \leq (\gamma/(2B))^{1/L}$ is sufficient to satisfy the required condition that $|\tilde{\zeta} - \zeta| \leq \zeta/L$ since $\zeta^L \geq \gamma/(2B)$. For each $\tilde{\zeta}$ that we pick, we consider ζ within a distance of $(\gamma/(2B))^{1/L}/L$. We need to pick enough $\tilde{\zeta}$'s to cover the whole region in (26). Picking a cover size of $LN^{\frac{1}{2L}}$ satisfies this condition since

$$\frac{(\frac{\gamma\sqrt{N}}{2B})^{1/L} - (\frac{\gamma}{2B})^{1/L}}{\frac{1}{L}(\frac{\gamma}{2B})^{1/L}} = L(N^{1/(2L)} - 1).$$

Therefore, by using Lemma 1, with probability at most $\tilde{\delta}$ and for all $\tilde{w}$ such that $|\zeta - \tilde{\zeta}| \leq \zeta/L$, the following bound is violated:

$$R^t(w) \leq \hat{R}^t_\gamma(\tilde{w}) + \mathcal{O}\left(\sqrt{\frac{B^2L^2\log(L\bar{h})\Pi_{j=1}^L\|\tilde{W}_j\|_2^2\sum_{i=1}^L\frac{\|\tilde{W}_i\|_F^2}{\|\tilde{W}_i\|_2^2} + \log(\frac{N}{\tilde{\delta}})}{\gamma^2 N}}\right).$$

By applying the union bound over the cover size, with probability at most $\tilde{\delta}LN^{1/(2L)}$, the same bound is violated for at least one of the $\tilde{\zeta}$'s out of the cover. Setting $\delta = \tilde{\delta}LN^{1/(2L)}$ and recalling that the proof generalizes from normalized weights $\tilde{w}$ to weights w gives the final result.

Secondary case. In this case, $\|h_w(\theta)\|_2 \leq B(\Pi_{i=1}^L \|W_i\|_2) \leq \gamma/2$. We get the following:

$$\begin{aligned} R^t(w) &\leq R^t_{\gamma/2}(0) \\ &\leq \hat{R}^t_{\gamma/2}(0) + C_{\gamma/2}(t)\sqrt{\log(1/\delta)/(2N)} \quad \text{w.p. at least } 1-\delta \\ &\leq \hat{R}^t_{\gamma}(w) + C_{\gamma/2}(t)\sqrt{\log(1/\delta)/(2N)} \quad \text{w.p. at least } 1-\delta \end{aligned}$$

The first and third lines come from $\|h_w(\theta)\|_2 \leq \gamma/2$ and the definition of the marginal fixed-point residual. The second lines uses Hoeffding's inequality as in Alquier (2021, Equation 1.3), which is permissible since the prediction is the zero vector and is therefore independent of the data.

B.3 Proof of Lemma 4

First, let $z^\star(\theta)$ be the nearest fixed-point of the operator T_θ to z so that $r_\theta(z) = \|z - z^\star(\theta)\|_2$.

$$\ell^{\text{fp}}_\theta(z) = \|T_\theta(z) - z\|_2 \leq \|T_\theta(z) - z^\star(\theta)\|_2 + \|z - z^\star(\theta)\|_2 \leq 2r_\theta(z)$$

The first inequality uses the triangle inequality, and the second inequality uses the non-expansiveness of T_θ.

B.4 Proof of Lemma 8

$$\begin{aligned} |r_\theta(T^t_\theta(z)) - r_\theta(T^t_\theta(w))| &= |\|T^k_\theta(z) - \Pi_{\mathbf{fix}\, T_\theta}(T^k_\theta(z))\|_2 - \|T^k_\theta(w) - \Pi_{\mathbf{fix}\, T_\theta}(T^k_\theta(w))\|_2| \\ &\leq \|T^k_\theta(z) - \Pi_{\mathbf{fix}\, T_\theta}(T^k_\theta(z)) + \Pi_{\mathbf{fix}\, T_\theta}(T^k_\theta(w)) - T^k_\theta(w)\|_2 \\ &\leq \|T^k_\theta(z) - T^k_\theta(w)\|_2 + \|\Pi_{\mathbf{fix}\, T_\theta}(T^k_\theta(z)) - \Pi_{\mathbf{fix}\, T_\theta}(T^k_\theta(w))\|_2 \\ &\leq 2\|T^k_\theta(z) - T^k_\theta(w)\|_2 \leq 2\|z - w\|_2 \end{aligned}$$

The first two inequalities use the reverse triangle inequality and triangle inequality. Since T_θ is non-expansive, $\mathbf{fix}\, T_\theta$ is a convex set (Ryu and Boyd, 2016, Section 2.4.1). The third inequality follows since the projection onto a convex set is non-expansive (Ryu and Boyd, 2016, Section 3.1). In the last inequality, we use the non-expansiveness of T_θ.

Acknowledgments

We are pleased to acknowledge that the work reported on in this paper was substantially performed using the Princeton Research Computing resources at Princeton University which is consortium of groups led by the Princeton Institute for Computational Science and Engineering (PICSciE) and Office of Information Technology's Research Computing. We thank Vinit Ranjan and three anonymous reviewers for helpful and detailed comments that improved the quality of this work.

References

Pierre Alquier. User-friendly introduction to PAC-Bayes bounds. *arXiv preprint arXiv:2110.11216*, 2021.

Brandon Amos. Tutorial on amortized optimization. *Foundations and Trends in Machine Learning*, 16(5):592–732, 2023.

Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Neural Information Processing Systems*, 2016.

Shaojie Bai, Vladlen Koltun, and Zico Kolter. Neural deep equilibrium solvers. In *International Conference on Learning Representations*, 2022.

Kyri Baker. Learning warm-start points for ac optimal power flow. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019.

Alexios Balatsoukas-Stimming and Christoph Studer. Deep unfolding for communications systems: A survey and some new directions. In *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 266–271, 2019.

Maria-Florina Balcan. Data-driven algorithm design. *arXiv preprint arXiv:2011.07177*, 2020.

Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. In *Conference on Learning Theory*, pages 213–274. PMLR, 2017.

Maria-Florina Balcan, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. How much data is sufficient to learn high-performing algorithms? generalization guarantees for data-driven algorithm design. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 919–932, 2021.

Sebastian Banert, Jevgenija Rudzusika, Ozan Öktem, and Jonas Adler. Accelerated forward-backward optimization using deep learning. *arXiv preprint arXiv:2105.05210*, 2021.

Goran Banjac, Paul Goulart, Bartolomeo Stellato, and Stephen Boyd. Infeasibility detection in the alternating direction method of multipliers for convex optimization. *Journal of Optimization Theory and Applications*, 183, 2019.

Peter L. Bartlett, Piotr Indyk, and Tal Wagner. Generalization bounds for data-driven numerical linear algebra. In *Conference on Learning Theory*, volume 178, pages 2013–2040, 2022.

Heinz. H. Bauschke and Patrick. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 1st edition, 2011.

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18:153:1–153:43, 2017.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm with application to wavelet-based image deblurring. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 693–696, 2009.

Richard Bellman. *Dynamic Programming.* Dover Publications, 1957.

Federico Benvenuto, Riccardo Zanella, Luca Zanni, and Mario Bertero. Nonnegative least-squares image deblurring: improved gradient projection approaches. *Inverse Problems*, 26(2):025004, 2010.

Dimitris Bertsimas and Bartolomeo Stellato. The voice of optimization. *Machine Learning*, 110:249–277, 2021.

Dimitris Bertsimas and Bartolomeo Stellato. Online Mixed-Integer Optimization in Milliseconds. *INFORMS Journal on Computing*, 34(4):2229–2248, 2022.

Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems.* Cambridge University Press, 2017.

Stephen Boyd, Enzo Busseti, Steven Diamond, Ronald N. Kahn, Kwangmoo Koh, Peter Nystrup, and Jan Speth. Multi-period trading via convex optimization, 2017.

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.

Luis M. Briceño-Arias and Patrick L. Combettes. Monotone operator methods for nash equilibria in non-potential games. In *Computational and Analytical Mathematics*, pages 143–159, New York, NY, 2013. Springer New York. ISBN 978-1-4614-7621-4.

Enzo Busseti, Walaa M. Moursi, and Stephen P. Boyd. Solution refinement at regular points of conic problems. *Computational Optimization and Applications*, 74:627 – 643, 2018.

Emmanuel J. Candès, Xiaodong Li, and Mahdi Soltanolkotabi. Phase retrieval from coded diffraction patterns. *Applied and Computational Harmonic Analysis*, 39(2):277–299, 2015.

J.H. Rick Chang, Chun-Liang Li, Barnabás Póczos, B.V.K. Vijaya Kumar, and Aswin C. Sankaranarayanan. One network to solve them all — solving linear inverse problems using deep projection models. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5889–5898, 2017.

Steven Chen, Kelsey Saulnier, Nikolay Atanasov, Daniel D. Lee, Vijay Kumar, George J. Pappas, and Manfred Morari. Approximating explicit model predictive control using constrained neural networks. In *American Control Conference*, pages 1520–1527, 2018.

Steven W. Chen, Tianyu Wang, Nikolay Atanasov, Vijay Kumar, and Manfred Morari. Large scale model predictive control with neural networks and primal active sets. *Automatica*, 135:109947, 2022a.

Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark. *Journal of Machine Learning Research*, 23(189):1–59, 2022b.

Xinshi Chen, Yufei Zhang, Christoph Reisinger, and Le Song. Understanding deep architectures with reasoning layer. In *Neural Information Processing Systems*, 2020.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters, 2017.

Alexandre d'Aspremont, Damien Scieur, and Adrien Taylor. Acceleration methods. *Foundations and Trends in Optimization*, 5(1-2):1–245, 2021.

Carl de Boor. On calculating with b-splines. *Journal of Approximation Theory*, 6(1):50–62, 1972.

Steven Diamond, Vincent Sitzmann, Felix Heide, and Gordon Wetzstein. Unrolled optimization with deep priors. *arXiv preprint arXiv:1705.08041*, 2017.

Moritz Diehl, Hans Joachim Ferreau, and Niels Haverbeke. *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation*. 2009.

Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In *Advances in Neural Information Processing Systems*, 2021.

Jim Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American Mathematical Society*, 82 (2):421–439, 1956.

Yoel Drori and Marc Teboulle. Performance of first-order methods for smooth convex minimization: a novel approach. *Mathematical Programming*, 145(1):451–482, 2014.

Jonathan Eckstein. *Splitting methods for monotone operators with applications to parallel optimization*. PhD thesis, Massachusetts Institute of Technology, 1989.

Laurent El Ghaoui and Hervé Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18:1035–1064, 1997.

Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

Mahyar Fazlyab, Manfred Morari, and George Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 67(1):1–15, 2022.

James Fienup. Phase retrieval algorithms: a comparison. *Applied Optics*, 21(15):2758–2769, 1982.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017.

Daniel Gabay. Applications of the method of multipliers to variational inequalities. 1983.

Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics With Applications*, 2:17–40, 1976.

Michael Garstka, Mark Cannon, and Paul Goulart. COSMO: A conic operator splitting method for large convex problems. In *European Control Conference*, 2019.

Pontus Giselsson and Stephen Boyd. Linear convergence and metric selection for douglas-rachford splitting and ADMM. *IEEE Transactions on Automatic Control*, 62(2):532–544, 2017.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *International Conference on Machine Learning*, Madison, WI, USA, 2010. Omnipress.

Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017.

Hengtao He, Chao-Kai Wen, Shi Jin, and Geoffrey Ye Li. Model-driven deep learning for mimo detection. *IEEE Transactions on Signal Processing*, 68:1702–1715, 2020.

Howard Heaton, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Safeguarded learned convex optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Pascal Van Hentenryck. *Machine Learning for Optimal Power Flows*, chapter 3, pages 62–82. 2021.

Mingyi Hong, Zhi-Quan Luo, and Meisam Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364, 2016.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

Jeffrey Ichnowski, Paras Jain, Bartolomeo Stellato, Goran Banjac, Michael Luo, Francesco Borrelli, Joseph. E Gonzales, Ian Stoica, and Ken Goldberg. Accelerating quadratic optimization with reinforcement learning. In *Advances in Neural Information Processing Systems 35*, 2021.

Ian Jolliffe. *Principal Component Analysis.* John Wiley & Sons, Ltd, 2005. ISBN 9780470013199.

Haewon Jung, Junyoung Park, and Jinkyoo Park. Learning context-aware adaptive solvers to accelerate quadratic programming. *arXiv preprint arXiv:2211.12443*, 2022.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

Benjamin Karg and Sergio Lucia. Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Transactions on Cybernetics*, PP, 2020.

Mikhail Khodak, Nina Balcan, Ameet Talwalkar, and Sergei Vassilvitskii. Learning predictions for algorithms with predictions. In *Advances in Neural Information Processing Systems*, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *International Conference on Management of Data*, 2018.

Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

Meiyi Li, Soheil Kolouri, and Javad Mohammadi. Learning to solve optimization problems with hard linear constraints. *IEEE Access*, 11:59995–60004, 2023.

Jialin Liu, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. ALISTA: Analytic weights are as good as learned weights in LISTA. In *International Conference on Learning Representations*, 2019.

Terrence WK Mak, Minas Chatzos, Mathieu Tanneau, and Pascal Van Hentenryck. Learning regionally decentralized ac optimal power flows with ADMM. *IEEE Transactions on Smart Grid*, 2023.

David A. McAllester. Some PAC-Bayesian theorems. In *Conference on Computational Learning Theory*. Association for Computing Machinery, 1998.

David A. McAllester. Simplified PAC-Bayesian margin bounds. In *Annual Conference Computational Learning Theory*, 2003.

Luke Metz, James Harrison, C Daniel Freeman, Amil Merchant, Lucas Beyer, James Bradbury, Naman Agrawal, Ben Poole, Igor Mordatch, Adam Roberts, et al. Velo: Training versatile learned optimizers by scaling up. *arXiv preprint arXiv:2211.09760*, 2022.

Rick P. Millane. Phase retrieval in crystallography and optics. *J. Opt. Soc. Am. A*, 7(3): 394–411, 1990.

Sidhant Misra, Line Roald, and Yeesian Ng. Learning for constrained optimization: Identifying optimal active constraint sets. *INFORMS Journal on Computing*, 34(1):463–480, 2022.

Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions, 2020.

Vishal Monga, Yuelong Li, and Yonina C. Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018.

B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. SCS: Splitting conic solver, version 2.1.2. `https://github.com/cvxgrp/scs`, 2019.

Brendan O'Donoghue. Operator splitting for a homogeneous embedding of the linear complementarity problem. *SIAM Journal on Optimization*, 31(3):1999–2023, 2021.

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

Isabeau Prémont-Schwarz, Jaroslav Vitku, and Jan Feyereisl. A simple guard for learned optimizers. In *International Conference on Machine Learning*, 2022.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Vinit Ranjan and Bartolomeo Stellato. Verification of first-order methods for parametric quadratic optimization. *arXiv preprint arXiv:2403.03331*, 2024.

Ralph Tyrrell Rockafellar and Roger J.-B. Wets. *Variational Analysis*. Springer Verlag, Heidelberg, Berlin, New York, 1998.

Dhruv Rohatgi. *Near-Optimal Bounds for Online Caching with Machine Learned Advice*, pages 1834–1845. 2020.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Ernest Ryu and Stephen Boyd. Primer on monotone operator methods. *Applied computational math*, 15(1):3–43, 2016.

Ernest Ryu and Wotao Yin. *Large-Scale Convex Optimization: Algorithms amp; Analyses via Monotone Operators*. Cambridge University Press, 2022.

Ernest Ryu, Adrien Taylor, Carolina Bergeling, and Pontus Giselsson. Operator splitting performance estimation: Tight contraction factors and optimal parameter selection. *SIAM Journal on Optimization*, 30(3):2251–2271, 2020.

Shinsaku Sakaue and Taihei Oki. Discrete-convex-analysis-based framework for warm-starting algorithms with predictions. In *Advances in Neural Information Processing Systems*, volume 35, pages 20988–21000, 2022.

Rajiv Sambharya and Bartolomeo Stellato. Data-driven performance guarantees for classical and learned optimizers. *arXiv preprint arXiv:2404.13831*, 2024.

Rajiv Sambharya, Georgina Hall, Brandon Amos, and Bartolomeo Stellato. End-to-End Learning to Warm-Start for Real-Time Quadratic Optimization. In *Proceedings of the 5th Annual Learning for Dynamics and Control Conference*, 2023.

John Shawe-Taylor and Robert C. Williamson. A PAC analysis of a bayesian estimator. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, COLT '97, pages 2–9. Association for Computing Machinery, 1997.

Yoav Shechtman, Yonina C. Eldar, Oren Cohen, Henry Nicholas Chapman, Jianwei Miao, and Mordechai Segev. Phase retrieval with application to optical imaging: A contemporary overview. *IEEE Signal Processing Magazine*, 32(3):87–109, 2015.

Jens Sjölund and Maria Bånkestad. Graph-based neural acceleration for nonnegative matrix factorization. *arXiv preprint arXiv:2202.00264*, 2022.

Yunlong Song and Davide Scaramuzza. Policy search for model predictive control with application to agile drone flight. *IEEE Transactions on Robotics*, 38(4):2114–2130, 2022.

Pantelis Sopasakis, Krina Menounou, and Panagiotis Patrinos. Superscs: fast and accurate large-scale conic optimization. pages 1500–1505, 2019.

Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. *Optimization for Machine Learning*. The MIT Press, 2011. ISBN 026201646X.

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Boyd Stephen. OSQP: An Operator Splitting Solver for Quadratic Programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

Hong Ye Tan, Subhadip Mukherjee, Junqi Tang, and Carola-Bibiane Schönlieb. Data-driven mirror descent with input-convex neural networks. *SIAM Journal on Mathematics of Data Science*, 5(2):558–587, 2023.

Adrien Taylor, Julien Hendrickx, and Francois Glineur. Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Math. Program.*, 161(1-2): 307–345, 2017a.

Adrien Taylor, Julien Hendrickx, and Francois Glineur. Exact worst-case performance of first-order methods for composite convex optimization. *SIAM J. Optim.*, 27(3):1283–1313, 2017b.

Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2011.

Shobha Venkataraman and Brandon Amos. Neural fixed-point acceleration for convex optimization. *arXiv preprint arXiv:2107.10254*, 2021.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18, 2001.

Homer F. Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

Kai Wang, Bryan Wilder, Andrew Perrault, and Milind Tambe. Automatically learning compact quality-aware surrogates for optimization problems. In *Neural Information Processing Systems*, 2020.

Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of ADMM in nonconvex nonsmooth optimization. *J. Sci. Comput.*, 78(1):29–63, 2019.

Kailun Wu, Yiwen Guo, Ziang Li, and Changshui Zhang. Sparse coding with gated learned ista. In *International Conference on Learning Representations*, 2020.

Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Singapore, Singapore, 2022.

Lihua Xie and Yeng Chai Soh. Robust kalman filtering for uncertain systems. *Systems & Control Letters*, 22(2):123–129, 1994.

Alp Yurtsever, Joel A Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 3(1):171–200, 2021.

Ahmed S. Zamzam and Kyri Baker. Learning optimal solutions for extremely fast ac optimal power flow. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2020.

Junzi Zhang, Brendan O'Donoghue, and Stephen Boyd. Globally convergent type-I anderson acceleration for nonsmooth fixed-point iterations. *SIAM Journal on Optimization*, 30(4): 3170–3197, 2020.

Kai Zhang, Wangmeng Zuo, Shugang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2808–2817, Los Alamitos, CA, USA, 2017. IEEE Computer Society.

Xiaojing Zhang, Monimoy Bujarbaruah, and Francesco Borrelli. Safe and near-optimal policy learning for model predictive control using primal-dual neural networks. In *2019 American Control Conference (ACC)*, pages 354–359, 2019.

Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.