RUN: A RELAXATION-FREE BRIDGE FROM MIN–MAX/MAX–MIN OPTIMIZATION TO NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Min–max problems sit at the centre of worst-case machine learning, and yet they have remained almost closed to neural networks. A saddle point is not the minimizer of any single objective, so a network asked to find one must either play the two-player game explicitly, and inherit the instability of gradient descent–ascent, or replace the stated objective by a relaxed proxy (meaning a loss of tightness). Algorithm unrolling, the established way of turning an optimization algorithm into a network, applies only to problems in a single vector variable. In this paper we remove that restriction, and, to the best of our knowledge, the passage we construct has not been available before. Building on the majorization–minimization principle, we replace the min–max problem at each iteration by a surrogate saddle point that is convex in the primal variable and concave in the dual one, and show that it reduces to a single-variable problem in two ways. Eliminating the dual variable in closed form leaves a plain minimization, while eliminating the primal variable and exchanging the two operators — an exchange that Sion's minimax theorem licenses precisely here — leaves a plain maximization. The saddle point is therefore gone before any network is built, the problem that remains is the worst-case one that was posed, and what solves it is an ordinary solver whose iterations become layers and whose step sizes are the only weights. We derive one architecture per reduction and prove that they decrease the original objective at every layer whatever their weights, with a convergent objective and bounded iterates whose limit points are stationary when the inner problem is solved exactly. Which reduction to open is decided by the conditioning of the problem and the cost of the inner solve. On min–max quadratics, fair principal component analysis and worst-case-group risk, the network built on the chosen reduction outperforms, at a matched budget, the saddle-point methods it replaces.

1 INTRODUCTION

A neural network is trained by descending a single objective, and this is also the reason networks have had so little to contribute to min–max optimization. Many of the quantities modern machine learning cares about are not losses but the largest member of a family of losses. Adversarial training asks for a predictor that stays accurate under the most damaging admissible perturbation of its input [37], distributionally robust optimization and its group-structured variants ask for one that stays accurate on the subpopulation where it performs worst [42, 50, 34, 19], fair dimensionality reduction asks for a subspace serving the worst-off group [52, 59, 45], and a generative adversarial network is a two-player game by construction [22]. In each case the training problem is a saddle-point problem

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{z}\in\mathcal{Z}} g(\mathbf{x},\mathbf{z}), \tag{1}$$

in which one variable is minimized and the other maximized, and the object being minimized, $f(\mathbf{x}) = \max_{\mathbf{z}\in\mathcal{Z}} g(\mathbf{x},\mathbf{z})$, is itself available only through an inner optimization. No single objective has a saddle point for its minimizer, and that one fact is what separates such a problem from every problem a network is comfortable with. The standard response is to descend in $\mathbf{x}$ and ascend in $\mathbf{z}$ [43, 13, 60, 35], at a well-understood price. Two step sizes must be set whose *ratio* governs stability, the iterates may rotate about the saddle instead of approaching it [38, 17, 20], and nothing

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

anywhere along the trajectory guarantees that f itself has decreased. Nor are these implementation difficulties, since local optimality is delicate even to define once the problem is neither convex nor concave [28] and approximately solving a constrained min–max problem is hard in general [18].

What trained networks have contributed here is limited by the structure of the problem. Algorithm unrolling takes a fixed number of iterations of a solver, treats them as layers, promotes the step sizes to weights and trains the result over a distribution of instances — LISTA [23] and its successors [15, 36, 62, 10, 63, 41, 55] — and what is unrolled in every one of these is a solver for a problem in a *single* variable, since that is the only kind whose iteration is a plain map from one iterate to the next. The one prominent exception, learned primal–dual [2], unrolls a saddle-point method but replaces its proximal operators by generic convolutional blocks, so the two-player structure survives into the network along with its instability, and the layers are no longer tied to the objective they came from. Neither neighbouring family comes closer, since optimization layers must solve an embedded problem or fixed point to high accuracy and differentiate through it implicitly [4, 3, 6], and learned optimizers are trained as black boxes and made reliable only from outside, by constraining each step against a fixed energy or by reverting to a trusted method when they misbehave [5, 14, 40, 26]. The neural solvers built on graph representations [53, 21] make the point explicit. Faced with a hard discrete or worst-case criterion they do not attempt the stated objective at all, but substitute a smooth — typically convex, or probabilistic — proxy a graph network can descend over aggregated neighbourhood features, recovering a feasible answer by rounding or search [30, 9, 29, 54, 12]. The proxy is what makes the network applicable, and its price is that whatever guarantee follows attaches to the proxy and not the criterion — the same objection that applies to semidefinite relaxations of worst-case criteria, which lift the problem into a much larger space and are not tight in general [46]. What is missing is therefore not a network but a *bridge*, a way of taking a min–max problem *in the form in which it is posed* and producing an iteration a network can be assembled out of, with no relaxation (meaning a loss of tightness) and no proxy in between.

It is worth being precise about what the literature already offers. Every unrolled architecture we are aware of that touches a saddle point keeps both of its players. Learned primal–dual retains the recursion and replaces its proximal operators by convolutional blocks, its stochastic counterpart unrolls the same recursion with stochastic operators [58], and the constrained-unrolling work trains two coupled networks that jointly approach a Lagrangian saddle point, one per variable, with descent and ascent constraints imposed layer by layer [24, 25], so stability is imposed by constraints and does not follow from the iteration itself. Where a network is placed inside a worst-case problem, what is learned is again one of the two players, an adversarial inner maximizer trained by learning-to-learn with the game left intact [61]. The worst-case criteria that are solved outright, finally, are solved after a relaxation (meaning a loss of tightness), fair principal component analysis through a semidefinite program being the standard example. To the best of our knowledge no existing method removes the saddle point *exactly* — before any network is built, and without relaxing the problem it came from — and then unrolls what is left.

Our contribution is based on that passage. Building on the majorization–minimization (MM) principle [33, 27, 32, 49, 57, 51], we majorize f only after writing it as a maximum over an auxiliary variable, so that each iteration replaces the min–max problem by a surrogate saddle point that is convex–concave *by construction*. Such a surrogate is usually passed to a saddle-point solver, which postpones the two-player problem without removing it; we take a different route. Because the surrogate is convex–concave, its inner problem is solvable in closed form in one variable or the other, and substituting that solution back removes the variable altogether. Eliminating the dual variable leaves a plain minimization, and eliminating the primal variable and exchanging the two operators — an exchange Sion's minimax theorem [56] licenses exactly because the surrogate is convex–concave — leaves a plain maximization. We call this pair of reductions a *window*. The saddle point disappears before any network is built, one variable returned exactly and for free by an explicit formula, the other left as a genuine single-variable problem of precisely the kind unrolling was built for.

MAIN CONTRIBUTIONS

The main contributions of this work are summarized as follows.

(i) **A reduction that removes the saddle point, and to our knowledge the first of its kind.** We show that either variable of the convex–concave surrogate can be eliminated in closed form,

leaving an ordinary minimization or an ordinary maximization in one variable instead of two. Existing unrolled saddle-point architectures keep both players and existing worst-case solvers relax the problem at a loss of tightness; we are aware of no method that removes the saddle point exactly and then unrolls what remains. The minimax theorem licensing the second route is stated where it is used, and we prove that the routes agree on the optimal value and, under strict convexity–concavity, on the iterate. Since the passage is an identity and involves no approximation, neither rounding nor lifting is required.

(ii) **A bridge to unrolled networks, and two architectures on it.** The reduction turns a saddle-point problem into an iterative single-variable one, precisely the kind of problem to which unrolling applies. In the resulting *reduced unrolled networks* RUN-P and RUN-D the elimination is a fixed backbone and only the step sizes of the retained solver are learned; setting those to a single constant turns the layer back into the fixed-step scheme it was built from, so that scheme is the initialization and training never begins below it.

(iii) **Guarantees that do not depend on the weights.** Danskin's theorem makes a layer a composition of explicit differentiable expressions, so its weight gradients follow by ordinary backpropagation without an implicit function theorem or an inner fixed point. We prove that a layer's output is kept only when it decreases the original objective, that the objective sequence is monotone, bounded and hence convergent within an explicit bound of the unsmoothed worst case, and that the iterates stay bounded with limit points that are stationary when the inner problem is solved exactly.

(iv) **Consistent gains on worst-case learning problems.** On min–max quadratics, fair principal component analysis and worst-case-group risk, the network suited to the problem outperforms the saddle-point methods it replaces at matched budget, and, where the algorithm it unfolds is available, improves on it too; the choice of window proves worth many orders of magnitude at equal depth, and the descent test lets a trained network run many times deeper than it was trained without diverging.

Notation. Boldface lowercase letters denote vectors (e.g. $\mathbf{x}$) and boldface uppercase letters matrices (e.g. $\mathbf{X}$). The transpose of $\mathbf{A}$ is $\mathbf{A}^\top$, the identity is $\mathbf{I}$, and $\mathbf{1}$, $\mathbf{0}$ are the all-ones and all-zeros vectors, their dimensions inferred from context. $\mathbb{R}^n$ is the set of n-dimensional real vectors, $\|\cdot\|$ and $\|\cdot\|_*$ the Euclidean and nuclear norms, $\Delta_m = \{\mathbf{z} \in \mathbb{R}^m : \mathbf{z} \geq \mathbf{0},\ \mathbf{1}^\top \mathbf{z} = 1\}$ the probability simplex, $\mathrm{tr}(\cdot)$ the trace and $\mathrm{polar}(\mathbf{A}) = \mathbf{A}(\mathbf{A}^\top \mathbf{A})^{-1/2}$ the orthogonal polar factor of a matrix of full column rank. The gradient of ϕ in $\mathbf{x}$ is $\nabla_{\mathbf{x}} \phi$, and $\Pi_{\mathcal{X}}(\cdot)$ is the projection, or retraction, onto $\mathcal{X}$. The superscript t indexes the outer iteration, which is also the network layer, and the subscript s the inner solver step.

2 PROBLEM FORMULATION

We are given a convex set $\mathcal{X} \subseteq \mathbb{R}^n$ and an objective $f : \mathcal{X} \to \mathbb{R}$, possibly both non-convex and non-smooth, and wish to solve $\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$. Building on the majorization–minimization (MM) principle [33, 27, 32, 49, 57, 51], we replace f at every iteration by an easier surrogate, built only after writing f as a maximum over an auxiliary variable. Concretely, we assume f admits the *max formulation*

$$f(\mathbf{x}) \;=\; \max_{\mathbf{z} \in \mathcal{Z}} \; g(\mathbf{x}, \mathbf{z}), \qquad g(\mathbf{x}, \cdot) \text{ concave on the convex set } \mathcal{Z} \subseteq \mathbb{R}^m, \tag{2}$$

and majorize only the $\mathbf{x}$-dependence of g at $\mathbf{x}^t$, obtaining a surrogate $g_s(\cdot, \cdot \mid \mathbf{x}^t)$ with

$$g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t) \;\geq\; g(\mathbf{x}, \mathbf{z}) \quad \forall\, \mathbf{x} \in \mathcal{X},\ \mathbf{z} \in \mathcal{Z}, \qquad g_s(\mathbf{x}^t, \mathbf{z} \mid \mathbf{x}^t) \;=\; g(\mathbf{x}^t, \mathbf{z}) \quad \forall\, \mathbf{z} \in \mathcal{Z}, \tag{3}$$

convex in $\mathbf{x}$ and concave in $\mathbf{z}$, so that each iteration replaces the original problem by the surrogate saddle point

$$\min_{\mathbf{x} \in \mathcal{X}} \; \max_{\mathbf{z} \in \mathcal{Z}} \; g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t). \tag{4}$$

Taking the maximum over $\mathbf{z}$ on both sides of equation 3 gives

$$d(\mathbf{x} \mid \mathbf{x}^t) \;:=\; \max_{\mathbf{z} \in \mathcal{Z}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t) \;\geq\; f(\mathbf{x}), \qquad d(\mathbf{x}^t \mid \mathbf{x}^t) \;=\; f(\mathbf{x}^t), \tag{5}$$

so the reduced primal objective $d(\cdot \mid \mathbf{x}^t)$ majorizes f and is tight at $\mathbf{x}^t$. This buys more than tractability, since it places the surrogate in exactly the form the minimax theorem requires, and the

whole of Section 3 turns on that (for an overview of MM, see Supplementary A). The class admitting equation 2 is large, containing every pointwise maximum of finitely many functions, every support function, and every constrained problem entered through its Lagrangian.

The instance we work with throughout is the worst-case objective over m branches, $f(\mathbf{x}) = \max_{1\le k\le m} f_k(\mathbf{x})$, with f_k the loss on the k-th group, the risk under the k-th distribution or the criterion of the k-th class pair. Writing the maximum in its variational form over the simplex gives the max formulation,

$$\max_{1\le k\le m} v_k \;=\; \max_{\mathbf{z}\in\Delta_m} \sum_{k=1}^{m} z_k v_k, \qquad \text{so that} \qquad f(\mathbf{x}) \;=\; \max_{\mathbf{z}\in\Delta_m} \sum_{k=1}^{m} z_k f_k(\mathbf{x}), \tag{6}$$

which is linear, hence concave, in $\mathbf{z}$, as equation 2 requires. That maximum is attained at a vertex of Δ_m, so its maximizer is not unique when two branches are tied and the objective is non-smooth exactly there. We therefore add τ times the negative entropy of $\mathbf{z}$,

$$f_\tau(\mathbf{x}) \;=\; \max_{\mathbf{z}\in\Delta_m} \Big\{ \sum_{k=1}^{m} z_k f_k(\mathbf{x}) \;-\; \tau \sum_{k=1}^{m} z_k \log z_k \Big\} \;=\; \tau \log \sum_{k=1}^{m} e^{f_k(\mathbf{x})/\tau}, \tag{7}$$

which is Nesterov smoothing [44, 8]. The added term is strictly concave on Δ_m, so the inner maximizer becomes unique and is a softmax, which is what makes both elimination maps of Section 3 single-valued. The effect is a bias that is uniform and known exactly,

$$f(\mathbf{x}) \;\le\; f_\tau(\mathbf{x}) \;\le\; f(\mathbf{x}) + \tau \log m \qquad \text{for every } \mathbf{x} \in \mathcal{X}, \tag{8}$$

so τ is a genuine accuracy knob and not a hyper-parameter whose effect must be discovered empirically. It is worth mentioning that the smoothing does not relax the problem. The feasible set remains unchanged, no lifting or rounding is required, and $f_\tau \to f$ uniformly as $\tau \to 0$, so that the original objective is recovered exactly in the limit [8, 7]. Moreover, since the bound in equation 8 is one-sided, any minimizer $\mathbf{x}_\tau$ of f_τ satisfies $f(\mathbf{x}_\tau) \le \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) + \tau \log m$, so that the resulting guarantee applies to the original worst-case criterion itself. We work with f_τ throughout.

At $\mathbf{x}^t$ we majorize the $\mathbf{x}$-dependence of equation 7, obtaining a surrogate satisfying equation 3 that is convex in $\mathbf{x}$ and concave in $\mathbf{z}$. When every branch is already convex, as for the min–max quadratics of Section 6, the weighted sum $\sum_k z_k f_k(\mathbf{x})$ is convex for every $\mathbf{z} \in \Delta_m$, no majorization is needed and $g_s = g$. When some branch is concave, as for fair principal component analysis, it suffices to majorize the concave branches by their tangents at $\mathbf{x}^t$ and to leave the rest untouched, setting $\bar{f}_k = f_k$ wherever f_k is already convex,

$$\bar{f}_k(\mathbf{x} \mid \mathbf{x}^t) \;=\; f_k(\mathbf{x}^t) + \nabla f_k(\mathbf{x}^t)^\top (\mathbf{x} - \mathbf{x}^t) \;\ge\; f_k(\mathbf{x}), \qquad \bar{f}_k(\mathbf{x}^t \mid \mathbf{x}^t) = f_k(\mathbf{x}^t), \tag{9}$$

which is affine, hence convex, and tight at $\mathbf{x}^t$; since the z_k are non-negative the weighted sum of the $\bar{f}_k$ majorizes that of the f_k for every $\mathbf{z} \in \Delta_m$ at once, so $g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t) = \sum_k z_k \bar{f}_k(\mathbf{x} \mid \mathbf{x}^t) - \tau \sum_k z_k \log z_k$ satisfies equation 3 and is convex–concave, as required.

Two worst-case learning problems instantiate this. Fair PCA [52, 59] seeks the rank-r projection serving the worst-off of m groups with second-moment matrices $\mathbf{C}_k$,

$$\max_{\mathbf{X}\in\mathbb{R}^{d\times r},\; \mathbf{X}^\top\mathbf{X}=\mathbf{I}} \; \min_{1\le k\le m} \; \operatorname{tr}(\mathbf{X}^\top \mathbf{C}_k \mathbf{X})/\alpha_k, \tag{10}$$

each group's explained variance normalized by α_k, the best rank-r variance that group could obtain alone, so the m ratios are comparable; its branches are concave, so equation 9 applies. Worst-case-group risk [50, 19] is the running example, with f_k the empirical risk on group k.

3 Removing the saddle point

We now derive the reduction. The surrogate saddle-point problem equation 4 reduces to a problem in a *single* vector variable in two different ways, neither involving any approximation, so that the min–max structure is not postponed, smoothed away or relaxed but eliminated. We first record the result that makes the second reduction legitimate, stated immediately after the surrogate has been formed because that is where it is used — majorization is what put g_s into the form the theorem requires [56]. Figure 1 is the whole construction in one picture, with the surrogate saddle point at the left, the two single-variable problems it reduces to, and the two networks obtained by unfolding the solver each of them leaves behind.

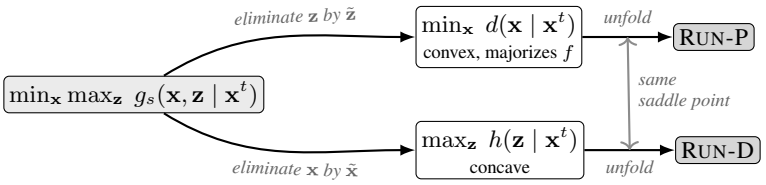

Figure 1: The window, and the two networks obtained by unfolding it.

Theorem 1 (Minimax swap for the surrogate)**.** *Let $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Z} \subseteq \mathbb{R}^m$ be convex and closed, and suppose that for every $\mathbf{z} \in \mathcal{Z}$ the map $\mathbf{x} \mapsto g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)$ is quasi-convex and lower semicontinuous, and for every $\mathbf{x} \in \mathcal{X}$ the map $\mathbf{z} \mapsto g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)$ is quasi-concave and upper semicontinuous. If, in addition, either* (i) *$\mathcal{X}$ is compact, or* (ii) *$\mathcal{Z}$ is compact and $g_s(\cdot, \mathbf{z} \mid \mathbf{x}^t)$ has bounded sublevel sets for some $\mathbf{z} \in \mathcal{Z}$, then the saddle value is attained and the two operators commute (for the proof, see Supplementary C.2):*

$$\min_{\mathbf{x} \in \mathcal{X}} \max_{\mathbf{z} \in \mathcal{Z}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t) \;=\; \max_{\mathbf{z} \in \mathcal{Z}} \min_{\mathbf{x} \in \mathcal{X}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t). \tag{11}$$

Theorem 1 is what makes the window two-way. Without equation 11 only the primal route would be legitimate, since $\max_{\mathbf{z}} \min_{\mathbf{x}} g_s$ would merely lower-bound the quantity of interest and the max–min problem would be a *relaxation* of the min–max one, with the loss of tightness that implies; with it, that problem is an exactly equivalent reformulation, free to be selected on computational grounds. This is the difference between a reduction and a relaxation. Both hypotheses hold in every instance we treat (Remark 9).

3.1 Two eliminations, two single-variable problems

Because $g_s(\cdot, \cdot \mid \mathbf{x}^t)$ is convex in $\mathbf{x}$ and concave in $\mathbf{z}$, its inner problem is a well-posed convex program in whichever variable is held second, and in the cases of interest it is solvable in closed form; substituting that solution back removes the variable entirely. Writing the two eliminations side by side,

$$\textit{primal window}: \quad \tilde{\mathbf{z}}(\mathbf{x} \mid \mathbf{x}^t) = \arg\max_{\mathbf{z} \in \mathcal{Z}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t), \quad d(\mathbf{x} \mid \mathbf{x}^t) = g_s\big(\mathbf{x}, \tilde{\mathbf{z}}(\mathbf{x} \mid \mathbf{x}^t) \mid \mathbf{x}^t\big), \tag{12}$$

$$\textit{dual window}: \quad \tilde{\mathbf{x}}(\mathbf{z} \mid \mathbf{x}^t) = \arg\min_{\mathbf{x} \in \mathcal{X}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t), \quad h(\mathbf{z} \mid \mathbf{x}^t) = g_s\big(\tilde{\mathbf{x}}(\mathbf{z} \mid \mathbf{x}^t), \mathbf{z} \mid \mathbf{x}^t\big), \tag{13}$$

the problem in equation 4 becomes, through the primal window, the plain minimization

$$\mathbf{x}^{t+1} \;=\; \arg\min_{\mathbf{x} \in \mathcal{X}} d(\mathbf{x} \mid \mathbf{x}^t), \qquad d(\cdot \mid \mathbf{x}^t) \text{ convex, majorizing } f \text{ and tight at } \mathbf{x}^t, \tag{14}$$

and, through the dual window, the plain maximization

$$\mathbf{z}^{t+1} \;=\; \arg\max_{\mathbf{z} \in \mathcal{Z}} h(\mathbf{z} \mid \mathbf{x}^t), \qquad \mathbf{x}^{t+1} \;=\; \tilde{\mathbf{x}}(\mathbf{z}^{t+1} \mid \mathbf{x}^t), \qquad h(\cdot \mid \mathbf{x}^t) \text{ concave}. \tag{15}$$

Both reduced problems involve one variable and an ordinary optimality condition, neither contains an adversary, and they are not two approximations of one problem but two exact descriptions of it.

Proposition 2 (The two windows agree)**.** *Under the hypotheses of Theorem 1 the two reduced problems share an optimal value, $\min_{\mathbf{x} \in \mathcal{X}} d(\mathbf{x} \mid \mathbf{x}^t) = \max_{\mathbf{z} \in \mathcal{Z}} h(\mathbf{z} \mid \mathbf{x}^t)$. If moreover $g_s(\cdot, \cdot \mid \mathbf{x}^t)$ is* strictly *convex–concave, that value is attained at the unique saddle point $(\mathbf{x}^\star, \mathbf{z}^\star)$ of g_s with $\mathbf{x}^\star = \tilde{\mathbf{x}}(\mathbf{z}^\star \mid \mathbf{x}^t)$, so the two windows, walked to convergence within a layer, return the same iterate (for the proof, see Supplementary C.7).*

Strictness upgrades equality of *values* to equality of the *iterates*, so the windows can differ only in how they approach a point they agree on. It holds for the min–max quadratics and the regularized group risks. For fair PCA, whose surrogate is affine in $\mathbf{X}$, the values agree as stated and the iterates agree whenever $\mathbf{A}(\mathbf{z})$ has full column rank, which is when the polar factor is unique (Remark 12).

3.2 THE DESIGN SPACE THE TWO ELIMINATIONS OPEN

For the worst-case objective with entropic smoothing the primal elimination is a softmax,

$$\tilde{\mathbf{z}}(\mathbf{x} \mid \mathbf{x}^t) \;=\; \operatorname{softmax}\big(\bar{f}(\mathbf{x} \mid \mathbf{x}^t)/\tau\big), \qquad d(\mathbf{x} \mid \mathbf{x}^t) \;=\; \tau \log \sum_{k=1}^{m} e^{\bar{f}_k(\mathbf{x}|\mathbf{x}^t)/\tau}, \tag{16}$$

with $\bar{f}_k$ the majorized branch of equation 9, leaving equation 14 an n-dimensional convex problem. The dual elimination needs the minimizer of the weighted sum and is closed form whenever that minimizer is, through a linear solve $\tilde{\mathbf{x}}(\mathbf{z}) = (\sum_k z_k \mathbf{H}_k)^{-1}(\sum_k z_k \mathbf{c}_k)$ for the min–max quadratics, and the orthogonal polar factor $\tilde{\mathbf{x}}(\mathbf{z}) = \operatorname{polar}(\mathbf{A}(\mathbf{z}))$ for fair PCA, by von Neumann's trace inequality [39]. Problem equation 15 is then m-dimensional and concave on the simplex, and the inner solver it needs is what we unfold in Section 4, being the only part of the iteration that is neither closed form nor free of tuning. Supplementary D writes out every map and gradient.

Since both windows describe the same problem, choosing between them is purely computational — and it is not a small choice. Per layer RUN-P pays $O(m)$ for a softmax and retains an n-dimensional unconstrained problem inheriting the conditioning of the f_k, while RUN-D pays for a linear solve or an SVD and retains an m-dimensional problem on the simplex. The trade is between paying once per layer for an exact solve and paying repeatedly for first-order steps on a possibly ill-conditioned one. Eliminating $\mathbf{x}$ removes the primal conditioning *entirely*, because $\tilde{\mathbf{x}}(\mathbf{z} \mid \mathbf{x}^t)$ solves the $\mathbf{x}$-subproblem exactly in one step, while the dual window's advantage decays as m grows. Where $\arg\min_{\mathbf{x}} \sum_k z_k f_k(\mathbf{x})$ has no closed form, the dual window is *forced* open with an inner solver, and every layer stays monotone whatever the accuracy of that solve (Proposition 7).

4 FROM THE REDUCED PROBLEM TO A NETWORK

Section 3 left us with an ordinary optimization problem in one variable — equation 14 through the primal window, equation 15 through the dual one — to be solved once per outer iteration, and which in general has no closed form of its own. This is the nested inner loop of the underlying algorithm, where all of its computation, free constants and inexactness live, and for that reason the only place in the construction where a network belongs. We therefore unfold that inner solver and leave everything around it fixed. A layer is one outer iteration, which fixes $\mathbf{x}^t$, forms the surrogate, applies the closed-form elimination as a fixed, parameter-free backbone, runs S steps of a first-order solver on the retained problem, and applies a descent test. The two inner recursions are

$$\text{RUN-P}: \quad \mathbf{x} \leftarrow \Pi_{\mathcal{X}}\big(\mathbf{x} - \eta_{t,s}\, \mathbf{P}_t\, \nabla_{\mathbf{x}} d(\mathbf{x} \mid \mathbf{x}^t)\big), \qquad s = 1{:}S, \quad \text{then test,} \tag{17}$$

$$\text{RUN-D}: \quad \mathbf{z} \leftarrow \operatorname{softmax}\big(\log \mathbf{z} + \eta_{t,s} \nabla_{\mathbf{z}} h(\mathbf{z} \mid \mathbf{x}^t)\big), \qquad s = 1{:}S, \quad \text{then } \mathbf{x} \leftarrow \tilde{\mathbf{x}}, \text{ test,} \tag{18}$$

where RUN-P projects, or retracts when $\mathcal{X}$ is a manifold, and RUN-D uses mirror ascent because the dual feasible set is a simplex, on which the entropy mirror map gives the exponentiated-gradient step in closed form. Both gradients come from Danskin's theorem (Proposition 6) at the cost of one elimination each, and Algorithms S1–S2 in Supplementary B state the networks step by step.

The learnable weights are the step sizes $\{\eta_{t,s}\}$ and, optionally, a diagonal preconditioner $\mathbf{P}_t$; the eliminations, the mirror map, the retraction and the descent test are all fixed by the derivation. Since the step sizes are exactly what the reduction leaves free, setting every $\eta_{t,s}$ to a single tuned constant turns the layer back into the fixed-step scheme it was built from. That scheme therefore lies inside the hypothesis class, and it is where we initialize, so a learned network can improve on it but never starts below it. The dual iterate is warm-started across layers and the parameter count is $O(TS + n)$, independent of the number of instances. The descent test keeps the candidate $\mathbf{x}^+$ if $d(\mathbf{x}^+ \mid \mathbf{x}^t) \le d(\mathbf{x}^t \mid \mathbf{x}^t)$ and otherwise backtracks along $-\nabla_{\mathbf{x}} d(\mathbf{x}^t \mid \mathbf{x}^t)$, halving until it does; its cost is one evaluation of d, and $d(\mathbf{x}^t \mid \mathbf{x}^t) = f_\tau(\mathbf{x}^t)$ is already known from the previous layer. It is what allows the layer between two eliminations to be arbitrary, and therefore what makes a learned layer admissible at all.

5 CONVERGENCE ANALYSIS

We establish what the two networks guarantee, in the order the matter arises. A single layer decreases the objective whatever its weights, the *objective* sequence is monotone, bounded and there-

fore convergent, and the *iterate* sequence is bounded with limit points attaining the limiting value; two structural results then make the bridge usable. The gradient formulas and the monotonicity of every layer are checked numerically in the released test suite.

Theorem 3 (A single scalar test makes any layer monotone)**.** *Let* $d(\cdot \mid \mathbf{x}^t)$ *satisfy equation 5. Then, for* any $\mathbf{x}^+ \in \mathcal{X}$ *produced by any means,*

$$d(\mathbf{x}^+ \mid \mathbf{x}^t) \;\le\; d(\mathbf{x}^t \mid \mathbf{x}^t) \qquad \Longrightarrow \qquad f_\tau(\mathbf{x}^+) \;\le\; f_\tau(\mathbf{x}^t), \tag{19}$$

and the test costs one evaluation of d*, that is, one application of the closed-form primal elimination. If it fails, backtracking along* $-\nabla_{\mathbf{x}} d(\mathbf{x}^t \mid \mathbf{x}^t)$ *satisfies it after finitely many halvings whenever* $\nabla_{\mathbf{x}} d(\mathbf{x}^t \mid \mathbf{x}^t) \neq \mathbf{0}$*, while if* $\nabla_{\mathbf{x}} d(\mathbf{x}^t \mid \mathbf{x}^t) = \mathbf{0}$ *then* $\mathbf{x}^t$ *minimizes the convex* $d(\cdot \mid \mathbf{x}^t)$ *over* $\mathcal{X}$ *and is a stationary point of* f_τ*. Every layer of Algorithms S1 and S2 is therefore monotone unconditionally (for the proof, see Supplementary C.3).*

Theorem 3 is the reason the reduction is a good place to put a network. Monotonicity is normally a property of an *algorithm*, lost the moment its update rule is replaced by learned weights; here it is a property of a *test*, and the update it tests may be arbitrary. This decouples the guarantee from the architecture, which is what unrolled networks ordinarily surrender.

Theorem 4 (Monotonicity, boundedness and convergence of the objective)**.** *Let* $\{\mathbf{x}^t\}_{t\ge 0}$ *be produced by Algorithm S1 or S2 with* arbitrary *weights, and let* f_τ *be bounded below on* $\mathcal{X}$*,* $f_\tau^{\inf} := \inf_{\mathbf{x}\in\mathcal{X}} f_\tau(\mathbf{x}) > -\infty$*. Then* $f_\tau(\mathbf{x}^{t+1}) \le f_\tau(\mathbf{x}^t)$ *for every* $t \ge 0$*, so* $\{f_\tau(\mathbf{x}^t)\}$ *is non-increasing and confined to* $[\, f_\tau^{\inf}, f_\tau(\mathbf{x}^0)\,]$ *and therefore converges to a limit* f^∞ *in that interval. Furthermore every limit point of the sequence* $\{f(\mathbf{x}^t)\}$ *of* unsmoothed *worst-case values lies in* $[\, f^\infty - \tau \log m,\; f^\infty\,]$ *(for the proof, see Supplementary C.4).*

The statement concerns f_τ itself, not a surrogate of it, and nothing is assumed about the weights, so it holds before, during and after training; its last clause carries the guarantee over to the unsmoothed worst case.

Theorem 5 (Boundedness, limit points and stationarity)**.** *Assume the hypotheses of Theorem 4 and that* $\mathcal{X}$ *is compact or* f_τ *has bounded sublevel sets. Then* (a) *the sequence* $\{\mathbf{x}^t\}$ *stays in* $\{\mathbf{x} \in \mathcal{X} : f_\tau(\mathbf{x}) \le f_\tau(\mathbf{x}^0)\}$ *and is bounded, so its set of limit points is non-empty and compact and every limit point* $\bar{\mathbf{x}}$ *satisfies* $f_\tau(\bar{\mathbf{x}}) = f^\infty$*;* (b) *if in addition each layer returns an exact minimizer of equation 14, or an exact maximizer of equation 15 followed by equation 13, and* $d(\cdot \mid \mathbf{x}^t)$ *is* μ*-strongly convex uniformly in* t*, then*

$$\sum_{t\ge 0} \left\|\mathbf{x}^{t+1} - \mathbf{x}^t\right\|^2 \;\le\; \tfrac{2}{\mu}\big(f_\tau(\mathbf{x}^0) - f^\infty\big) \;<\; \infty, \tag{20}$$

so $\|\mathbf{x}^{t+1} - \mathbf{x}^t\| \to 0$ *and the set of limit points is also connected; and* (c) *under the hypotheses of* (b)*, with* f_τ *and* d *continuously differentiable, every limit point is a first-order stationary point of* f_τ *over* $\mathcal{X}$ *(for the proof, see Supplementary C.5).*

Part (a) holds for the learned network whatever its weights, parts (b) and (c) hold whenever the inner solver is run to optimality, and Theorem 3 keeps every layer monotone in between.

Proposition 6 (Reduced gradients by Danskin)**.** *If* $\tilde{\mathbf{z}}(\mathbf{x} \mid \mathbf{x}^t)$ *and* $\tilde{\mathbf{x}}(\mathbf{z} \mid \mathbf{x}^t)$ *are unique, then* $\nabla_{\mathbf{x}} d(\mathbf{x} \mid \mathbf{x}^t) = \nabla_{\mathbf{x}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)\big|_{\mathbf{z}=\tilde{\mathbf{z}}(\mathbf{x}\mid\mathbf{x}^t)}$ *and* $\nabla_{\mathbf{z}} h(\mathbf{z} \mid \mathbf{x}^t) = \nabla_{\mathbf{z}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)\big|_{\mathbf{x}=\tilde{\mathbf{x}}(\mathbf{z}\mid\mathbf{x}^t)}$ *(for the proof, see Supplementary C.6).*

Proposition 7 (An inexactly opened window is still safe)**.** *Suppose the dual elimination is available only through a map* $\hat{\mathbf{x}}(\mathbf{z})$ *with* $g_s(\hat{\mathbf{x}}(\mathbf{z}), \mathbf{z} \mid \mathbf{x}^t) \le \min_{\mathbf{x}\in\mathcal{X}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t) + \epsilon$ *for every* $\mathbf{z} \in \mathcal{Z}$*. Then* $h \le \hat{h} \le h + \epsilon$ *pointwise, so the reduced dual objective is optimized to within* ϵ*; the conclusions of Theorems 3, 4 and 5 (a) are unaffected, because the test evaluates* d*, computed from the* primal *elimination, which treats the candidate as a black box (for the proof, see Supplementary C.8).*

Neither expression in Proposition 6 contains a derivative of the elimination map, so a layer is a composition of explicit differentiable expressions whose weight gradients follow by ordinary backpropagation — with no implicit function theorem, adjoint system or inner fixed point, unlike optimization layers and deep equilibrium models [4, 3, 6]. Proposition 7 extends the bridge to every problem whose inner minimizer can be computed accurately, with or without a closed form.

Table 1: *Correct digits* in the objective on min–max quadratics after T layers; higher is better, and >16 marks the accuracy limit of double-precision arithmetic.

varying the condition number κ

method	T=2	T=4	T=8	T=16
$\kappa = 10$				
Optimistic GDA (2018) [17]	1.8	2.3	3.1	4.5
GDA (2020) [35]	2.2	2.7	3.4	5.2
MM4MM-P (2024) [51]	3.8	5.5	8.6	13.6
MM4MM-D (2024) [51]	4.4	5.8	8.2	12.9
Unrolled GDA (control)	2.6	3.7	5.7	9.5
RUN-P (ours)	5.0	8.8	14.1	14.8
RUN-D (ours)	**7.4**	**11.4**	**15.4**	**>16**
$\kappa = 10000$				
Optimistic GDA (2018) [17]	1.7	1.8	2.1	2.5
GDA (2020) [35]	2.1	2.4	2.8	3.3
MM4MM-P (2024) [51]	2.8	3.3	4.1	5.3
MM4MM-D (2024) [51]	8.3	11.9	**>16**	**>16**
Unrolled GDA (control)	2.3	2.9	3.6	3.9
RUN-P (ours)	3.0	3.6	4.1	5.3
RUN-D (ours)	**12.4**	**>16**	**>16**	**>16**

varying the number of branches m

method	T=2	T=4	T=8	T=16
$m = 2$				
Optimistic GDA (2018) [17]	1.5	1.8	2.0	2.3
GDA (2020) [35]	1.8	2.0	2.2	2.4
MM4MM-P (2024) [51]	2.2	2.7	3.2	3.9
MM4MM-D (2024) [51]	9.5	12.7	**>16**	**>16**
Unrolled GDA (control)	2.0	2.3	2.5	2.8
RUN-P (ours)	2.5	2.7	3.3	3.9
RUN-D (ours)	**15.7**	**>16**	**>16**	**>16**
$m = 50$				
Optimistic GDA (2018) [17]	2.8	3.2	3.4	4.4
GDA (2020) [35]	2.9	3.5	4.9	7.4
MM4MM-P (2024) [51]	5.7	8.5	13.7	**>16**
MM4MM-D (2024) [51]	6.4	9.2	14.4	**>16**
Unrolled GDA (control)	3.4	4.4	6.4	10.5
RUN-P (ours)	7.6	**13.2**	14.8	**>16**
RUN-D (ours)	**7.8**	12.5	**>16**	**>16**

6 NUMERICAL RESULTS

Three things decide whether the reduction earns its place — the choice of window, the value of learning inside one, and the margin over the saddle-point methods it replaces — and three problems settle them. *Min–max quadratics* make both eliminations exact and closed form, so the design decision of Section 3.2 is isolated from every other effect; *fair PCA* equation 10 opens both windows and makes the polar factorization the matched budget; and *worst-case-group risk* shuts the dual window, which must then be forced. Between them they cover the standard settings of this literature, with real data in both applications alongside synthetic sets drawn from seeds, so every instance is reproducible.

Instances are drawn independently, and the networks are trained on one set of instances and evaluated on a *disjoint* one [23, 41]. Every method receives the same initialization and the same budget, measured in the operation dominating the problem at hand. Every free step size is grid-searched on the *training* set and never on the evaluation set, and our networks start at the grid-searched constant of the algorithm they unfold, so every improvement is over a tuned baseline. Means are over three to five seeds, on CPU in double precision. What each compared method does is described in Supplementary E and the protocol in Supplementary F.

No existing network solves these problems without keeping the saddle point (Section 1), so we compare against the state of the art in *pure optimization* for each problem — gradient descent–ascent [35] and its optimistic variant [17], with their two step sizes searched jointly, the fixed-step schemes the networks are unfolded from, each at its tuned constant step, and the specialist solvers of the applications. To these we add the strictest opponent available, a trained network that keeps the saddle point. UNROLLED GDA unrolls the two-player recursion and learns its two step sizes per layer, following the design of learned primal–dual [2]. It receives exactly the training our own networks receive, so whatever separates it from them is the reduction, not the learning.

The quadratics are scored in *correct digits*, $-\log_{10}$ of the suboptimality $f_\tau(\mathbf{x}_T) - f_\tau^\star$ against a damped-Newton reference, averaged geometrically over instances, with κ the condition number of the branch Hessians and m the number of branches. Every method starts between 1.2 and 1.3 correct digits, and sixteen is the most double-precision arithmetic can resolve, so an entry says how many digits a depth buys; throughout, **bold** marks the best value in a column and underline the second.

Three things are visible in Table 1. The first is that *the choice of window decides the speed*. Both windows describe the same saddle point and, on these problems, agree on the iterate as well (Proposition 2), so the table measures how fast each reaches a common answer. As κ grows the dual window reaches the double-precision limit within four layers while the primal window advances a fraction as fast, and at $\kappa = 10^4$ and $T = 4$ the primal window has under four correct digits and the dual window all sixteen (Supplementary G.1 gives every setting). This is a property of ill-conditioning, not a blanket superiority of one route, exactly as Section 3.2 predicts — eliminating $\mathbf{x}$ solves the primal subproblem exactly at every layer, so the conditioning of the f_k never enters —

Table 2: Worst-group objective for a logistic predictor at depth T; lower is better.

SPURIOUS					DIGITS				
method	T=4	T=8	T=16	T=32	method	T=4	T=8	T=16	T=32
ERM-GD	0.5343	0.5238	0.5222	0.5226	ERM-GD	0.6411	0.5858	0.5283	0.3033
Subgradient	0.6071	0.5660	0.5313	0.5016	Subgradient	0.6969	0.6852	0.4876	0.2752
Optimistic GDA (2018) [17]	0.5615	0.5247	0.4924	0.4718	Optimistic GDA (2018) [17]	0.6304	0.5775	0.4962	0.3962
Online group-DRO (2020) [50]	0.5084	0.4909	0.4726	0.4563	Online group-DRO (2020) [50]	0.6239	0.5615	0.4711	0.2371
GDA (2020) [35]	0.5130	0.4917	0.4735	0.4581	GDA (2020) [35]	0.5793	0.5298	0.3771	0.2529
MM4MM-P (2024) [51]	0.5288	0.4982	0.4770	0.4629	MM4MM-P (2024) [51]	0.4755	0.4571	0.4309	0.3897
Unrolled GDA (control)	0.5068	0.4872	0.5181	0.4574	Unrolled GDA (control)	0.3995	0.4341	0.3204	0.2529
RUN-P (ours)	0.5279	0.4975	0.4757	0.4629	RUN-P (ours)	0.4756	0.4441	0.4283	0.3812
RUN-D (Newton-3, ours)	**0.4749**	**0.4562**	**0.4473**	**0.4469**	RUN-D (Newton-3, ours)	**0.1135**	**0.1076**	**0.1076**	**0.1076**

and it shrinks with the conditioning until the two fixed-step algorithms trade places at $\kappa = 10$. The right-hand panel shows the other boundary, the advantage decaying as the retained dual problem grows with m, from a lead of thirteen digits at $m = 2$ and $T = 2$ to two networks that agree within a factor of 1.4 at $m = 50$. The second observation is that *learning helps within a window*. The networks improve on the algorithms they are initialized at by three correct digits at $\kappa = 10$ and $T = 4$ and by more than five at $T = 8$, and on a heterogeneous set of instances the gain reaches an order of magnitude (Supplementary G.2). The third is the margin over the saddle-point methods. RUN-D and the algorithm it starts from both lead every saddle-point method in every cell of the study. Training one helps — UNROLLED GDA gains four digits on tuned GDA at $\kappa = 10$ — but it still ends six digits short of RUN-D at $\kappa = 10$ and twelve at $\kappa = 10^4$.

Fair PCA confirms the ordering on a real problem where both windows are open, the dual window leading every competing method at the tighter budgets and on a par with the best of them at the most generous one (Supplementary G.4).

For logistic losses the dual elimination has no closed form, so the dual algorithm is unavailable. We compare the primal window against tuned baselines and *force* the dual window open with an inner Newton solver, charged $1+N$ layers for its N steps to keep the budget matched. The forced dual window (Table 2) is then strongest at every budget on both datasets, and by a wide margin on DIGITS, where four layers reach a worst-group objective less than half what any baseline reaches at thirty-two. Those four layers cost in total what the first-order baselines spend over their whole budget, so the accuracy comes at a matched price. What decides the design is therefore whether the dual elimination is *accurate and cheap*, a weaker requirement than a closed form. A stress test closes the section, in which the descent test keeps weights trained at $T = 8$ stable when the networks are run to $T = 128$, while the primal network without it diverges, as Theorem 3 requires (Supplementary G.5).

7 LIMITATIONS

Our problems are non-convex, where no algorithm can guarantee a global minimizer [11]. We establish monotone descent whatever the weights, a bounded and convergent objective, and bounded iterates whose limit points form a non-empty compact set, each of them first-order stationary when the inner problem is solved exactly.

8 CONCLUSION

Neural networks have had almost nothing to say about min–max optimization for want of a passage, since no single objective has a saddle point for its minimizer while the architectures machine learning builds descend exactly such an objective. This paper supplies it, replacing the problem at each iteration by a convex–concave surrogate and eliminating either variable in closed form — Sion's theorem licensing the second route — to leave a relaxation-free single-variable iteration a network can be built from. Of the two architectures it yields, the one suited to the problem outperforms the saddle-point methods it replaces at a matched budget.

AI USE STATEMENT

AI-assisted editing was used for language editing and proofreading of the main paper and supplementary materials. The authors take full responsibility for the final content of this paper.

REPRODUCIBILITY STATEMENT

We have made the following efforts to ensure that this work can be reproduced. The complete source code of our method, together with the datasets used in the experiments and the scripts that regenerate every table and plot of this paper, is submitted as a supplementary `.zip` archive alongside the paper. The assumptions behind each theoretical claim are stated with the claim itself in Sections 3 and 5, and complete proofs are given in Supplementary C; the elimination maps and their gradients are written out in Supplementary D, and the experimental protocol, the data-generating procedures and the hyper-parameters are described in Supplementary F, with the search grids recorded in the released scripts. The archive also contains a test suite that checks the elimination maps, the gradient formulas, the minimax identity and the monotonicity of every layer numerically, so that the correspondence between the statements in the paper and the released implementation can be checked directly.

REFERENCES

[1] P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

[2] Jonas Adler and Ozan Öktem. Learned primal-dual reconstruction. *IEEE Transactions on Medical Imaging*, 37(6):1322–1332, 2018.

[3] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[4] Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

[5] Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

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

[7] Amir Beck. *First-Order Methods in Optimization*. SIAM, 2017.

[8] Amir Beck and Marc Teboulle. Smoothing and first order methods: A unified framework. *SIAM Journal on Optimization*, 22(2):557–580, 2012.

[9] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[10] Mark Borgerding, Philip Schniter, and Sundeep Rangan. AMP-inspired deep networks for sparse linear inverse problems. *IEEE Transactions on Signal Processing*, 65(16):4293–4308, 2017.

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[12] Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.

[13] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

[14] 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, 2022.

[15] Xiaohan Chen, Jialin Liu, Zhangyang Wang, and Wotao Yin. Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[16] John M. Danskin. *The Theory of Max-Min and Its Application to Weapons Allocation Problems*. Springer, 1967.

[17] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training GANs with optimism. In *International Conference on Learning Representations*, 2018.

[18] Constantinos Daskalakis, Stratis Skoulakis, and Manolis Zampetakis. The complexity of constrained min-max optimization. In *ACM SIGACT Symposium on Theory of Computing (STOC)*, 2021.

[19] John C. Duchi and Hongseok Namkoong. Learning models with uniform performance via distributionally robust optimization. *The Annals of Statistics*, 49(3):1378–1406, 2021.

[20] Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial networks. In *International Conference on Learning Representations*, 2019.

[21] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 2017.

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

[23] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *International Conference on Machine Learning (ICML)*, 2010.

[24] Samar Hadou and Alejandro Ribeiro. Unrolled graph neural networks for constrained optimization. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2386–2390, 2026.

[25] Samar Hadou, Navid NaderiAlizadeh, and Alejandro Ribeiro. Robust stochastically-descending unrolled networks. *IEEE Transactions on Signal Processing*, 72:5484–5499, 2024.

[26] Howard Heaton, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Safeguarded learned convex optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):7848–7855, 2023.

[27] David R. Hunter and Kenneth Lange. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004.

[28] Chi Jin, Praneeth Netrapalli, and Michael I. Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *International Conference on Machine Learning*, 2020.

[29] Nikolaos Karalias and Andreas Loukas. Erdős goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In *Advances in Neural Information Processing Systems*, 2020.

[30] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, 2017.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[32] Kenneth Lange. *MM Optimization Algorithms*. SIAM, Philadelphia, 2016.

[33] Kenneth Lange, David R. Hunter, and Ilsoon Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1–20, 2000.

[34] Daniel Levy, Yair Carmon, John C. Duchi, and Aaron Sidford. Large-scale methods for distributionally robust optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[35] Tianyi Lin, Chi Jin, and Michael I. Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *International Conference on Machine Learning (ICML)*, 2020.

[36] 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 (ICLR)*, 2019.

[37] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[38] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *International Conference on Machine Learning*, 2018.

[39] Leon Mirsky. A trace inequality of John von Neumann. *Monatshefte für Mathematik*, 79(4): 303–306, 1975.

[40] Michael Moeller, Thomas Möllenhoff, and Daniel Cremers. Controlling neural networks via energy dissipation. In *International Conference on Computer Vision (ICCV)*, 2019.

[41] 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.

[42] Hongseok Namkoong and John C. Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[43] Arkadi Nemirovski. Prox-method with rate of convergence O(1/t) for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

[44] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[45] Matt Olfat and Anil Aswani. Convex formulations for fair principal component analysis. In *AAAI Conference on Artificial Intelligence*, 2019.

[46] Mohammad Mahdi Omati, Prabhu Babu, Petre Stoica, and Arash Amini. A max-min approach to the worst-case class separation problem. *Transactions on Machine Learning Research*, 2025.

[47] Alexander M. Ostrowski. *Solution of Equations and Systems of Equations*. Academic Press, New York, 2nd edition, 1966.

[48] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[49] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

[50] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations (ICLR)*, 2020.

[51] Astha Saini, Petre Stoica, Prabhu Babu, and Aakash Arora. Min-max framework for majorization-minimization algorithms in signal processing applications: An overview. *Foundations and Trends in Signal Processing*, 18(4):310–389, 2024.

[52] Samira Samadi, Uthaipon Tantipongpipat, Jamie Morgenstern, Mohit Singh, and Santosh Vempala. The price of fair PCA: One extra dimension. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[53] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[54] Martin J. A. Schuetz, J. Kyle Brubaker, and Helmut G. Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4:367–377, 2022.

[55] Nir Shlezinger, Jay Whang, Yonina C. Eldar, and Alexandros G. Dimakis. Model-based deep learning. *Proceedings of the IEEE*, 111(5):465–499, 2023.

[56] Maurice Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8(1):171–176, 1958.

[57] Ying Sun, Prabhu Babu, and Daniel P. Palomar. Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, 2017.

[58] Junqi Tang, Subhadip Mukherjee, and Carola-Bibiane Schönlieb. Stochastic primal-dual deep unrolling. *arXiv preprint arXiv:2110.10093*, 2021.

[59] Uthaipon Tantipongpipat, Samira Samadi, Mohit Singh, Jamie Morgenstern, and Santosh Vempala. Multi-criteria dimensionality reduction with applications to fairness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[60] Kiran K. Thekumparampil, Prateek Jain, Praneeth Netrapalli, and Sewoong Oh. Efficient algorithms for smooth minimax optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[61] Yuanhao Xiong and Cho-Jui Hsieh. Improved adversarial training via learned optimizer. In *European Conference on Computer Vision (ECCV)*, pages 85–100, 2020.

[62] Yan Yang, Jian Sun, Huibin Li, and Zongben Xu. Deep ADMM-net for compressive sensing MRI. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[63] Jian Zhang and Bernard Ghanem. ISTA-net: Interpretable optimization-inspired deep network for image compressive sensing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Supplementary Material

A THE MAJORIZATION–MINIMIZATION PRINCIPLE

Everything below is built on the majorization–minimization (MM) principle, so we recall it in the form we shall use, and then in the min–max form our reduction needs. MM is a *principle* and not a particular algorithm, prescribing a way of replacing a hard problem by a sequence of easy ones and leaving the design of the replacement to the user. Consider $\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$, with $\mathbf{x}$ the decision variable, f the objective and $\mathcal{X}$ the feasible set. An MM-based method introduces at each iteration t a surrogate $u(\mathbf{x} \mid \mathbf{x}^t)$ overestimating f everywhere on $\mathcal{X}$ but agreeing with it at the current iterate, and takes as next iterate

$$\mathbf{x}^{t+1} \in \arg\min_{\mathbf{x}\in\mathcal{X}} u\big(\mathbf{x} \mid \mathbf{x}^t\big), \qquad \text{where} \qquad u\big(\mathbf{x} \mid \mathbf{x}^t\big) \geq f(\mathbf{x}) \ \forall \mathbf{x} \in \mathcal{X}, \quad u\big(\mathbf{x}^t \mid \mathbf{x}^t\big) = f\big(\mathbf{x}^t\big). \tag{S1}$$

These two conditions are what force the objective downhill without a line search and without a step size to tune,

$$f\big(\mathbf{x}^{t+1}\big) \leq u\big(\mathbf{x}^{t+1} \mid \mathbf{x}^t\big) \leq u\big(\mathbf{x}^t \mid \mathbf{x}^t\big) = f\big(\mathbf{x}^t\big), \tag{S2}$$

the first inequality being majorization, the second holding because $\mathbf{x}^{t+1}$ minimizes $u(\cdot \mid \mathbf{x}^t)$ over a set containing $\mathbf{x}^t$, and the last being tightness. The argument nowhere requires f to be smooth or $\mathcal{X}$ convex, and its middle inequality asks only for a *decrease* of the surrogate, so any rule at all that decreases it inherits the monotonicity — an observation that, pushed as far as it will go, later admits a learned layer into the iteration without disturbing the guarantee.

Our problem, however, is the saddle-point problem equation 1, and a convenient majorizer of $f(\mathbf{x}) = \max_{\mathbf{z}} g(\mathbf{x}, \mathbf{z})$ is rarely available, f being defined implicitly and in general non-smooth. The extension of the principle to this setting majorizes *before* the maximum is taken [51]. One assumes the max formulation equation 2, available through conjugate duality or the Lagrangian of a constrained problem, and majorizes only the $\mathbf{x}$-dependence of g at $\mathbf{x}^t$ to obtain $g_s(\cdot, \cdot \mid \mathbf{x}^t)$ satisfying equation 3, convex in $\mathbf{x}$ and concave in $\mathbf{z}$, so that each iteration replaces equation 1 by the surrogate saddle-point problem equation 4. Taking the maximum over $\mathbf{z}$ on both sides of equation 3 gives equation 5, so $d(\cdot \mid \mathbf{x}^t)$ is a majorizer of f, tight at $\mathbf{x}^t$, and convex as a pointwise maximum of convex functions; symmetrically $h(\mathbf{z} \mid \mathbf{x}^t) := \min_{\mathbf{x}\in\mathcal{X}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)$ is concave on $\mathcal{Z}$. What majorization buys here is more than tractability of the $\mathbf{x}$-problem, and the whole of Section 3 turns on it, because by forcing g_s to be convex–concave it places the surrogate in exactly the form the minimax theorem requires, and thereby licenses an exchange of the two operators for which equation 1 itself has no licence. The min–max structure is not merely postponed by majorization — it is made *reducible* by it.

B THE TWO NETWORKS

Algorithms S1 and S2 write out layer by layer the two networks that Figure 1 draws. They are the same object seen through the two windows, and each layer re-majorizes at the current iterate, applies the closed-form elimination, takes S learned steps on whichever single-variable problem was retained, and then applies the descent test. Only the step sizes $\{\eta_{t,s}\}$ and the optional preconditioner $\{\mathbf{P}_t\}$ are learned; every other line is fixed by the derivation of Section 3. The routine DESCENTTEST returns its first argument when $d(\cdot \mid \mathbf{x}^t)$ did not increase, and otherwise backtracks along $-\nabla_{\mathbf{x}} d(\mathbf{x}^t \mid \mathbf{x}^t)$ until it does (Theorem 3). Setting every $\eta_{t,s}$ to a single tuned constant turns each layer back into the fixed-step scheme it was built from, which is the sense in which that scheme lies inside the hypothesis class.

C PROOFS

Throughout, $g_s(\cdot, \cdot \mid \mathbf{x}^t) : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ is convex in $\mathbf{x}$, concave in $\mathbf{z}$, and satisfies the majorization and tightness conditions equation 3; $d(\mathbf{x} \mid \mathbf{x}^t) = \max_{\mathbf{z}\in\mathcal{Z}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)$ and $h(\mathbf{z} \mid \mathbf{x}^t) = \min_{\mathbf{x}\in\mathcal{X}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)$. We write f for the objective that the max formulation equation 2 defines; in the smoothed instances of Section 2 this objective is f_τ, and every statement below applies to it verbatim.

Algorithm S1 Unfolding the primal window as RUN-P

Require: $\mathbf{x}^0$, depth T, inner steps S, weights $\{\eta_{t,s}\}$, $\{\mathbf{P}_t\}$
for $t = 0, \dots, T-1$ **do**
  form $g_s(\cdot, \cdot \mid \mathbf{x}^t)$; $\mathbf{x} \leftarrow \mathbf{x}^t$
  for $s = 1, \dots, S$ **do**
    $\tilde{\mathbf{z}} \leftarrow \arg\max_{\mathbf{z} \in \mathcal{Z}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)$
    $\mathbf{x} \leftarrow \Pi_{\mathcal{X}}\big(\mathbf{x} - \eta_{t,s} \mathbf{P}_t \nabla_{\mathbf{x}} g_s(\mathbf{x}, \tilde{\mathbf{z}} \mid \mathbf{x}^t)\big)$
  end for
  $\mathbf{x}^{t+1} \leftarrow$ DESCENTTEST$(\mathbf{x}, \mathbf{x}^t)$
end for
Output: $\mathbf{x}^T$

Algorithm S2 Unfolding the dual window as RUN-D

Require: $\mathbf{x}^0, \mathbf{z}^0 \in \Delta_m$, T, S, weights $\{\eta_{t,s}\}$
for $t = 0, \dots, T-1$ **do**
  form $g_s(\cdot, \cdot \mid \mathbf{x}^t)$; $\mathbf{z} \leftarrow \mathbf{z}^t$
  for $s = 1, \dots, S$ **do**
    $\tilde{\mathbf{x}} \leftarrow \arg\min_{\mathbf{x} \in \mathcal{X}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)$
    $\mathbf{z} \leftarrow \operatorname{softmax}\big(\log \mathbf{z} + \eta_{t,s} \nabla_{\mathbf{z}} g_s(\tilde{\mathbf{x}}, \mathbf{z} \mid \mathbf{x}^t)\big)$
  end for
  $\mathbf{z}^{t+1} \leftarrow \mathbf{z}$; $\mathbf{x}^{t+1} \leftarrow$ DESCENTTEST$\big(\tilde{\mathbf{x}}(\mathbf{z} \mid \mathbf{x}^t), \mathbf{x}^t\big)$
end for
Output: $\mathbf{x}^T$

C.1 THE REDUCED PROBLEMS

Lemma 8. *$d(\cdot \mid \mathbf{x}^t)$ is convex on $\mathcal{X}$ and majorizes f with equality at $\mathbf{x}^t$; $h(\cdot \mid \mathbf{x}^t)$ is concave on $\mathcal{Z}$.*

Proof. d is a pointwise maximum of the family $\{g_s(\cdot, \mathbf{z} \mid \mathbf{x}^t)\}_{\mathbf{z} \in \mathcal{Z}}$ of convex functions, hence convex. Taking $\max_{\mathbf{z} \in \mathcal{Z}}$ in $g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t) \geq g(\mathbf{x}, \mathbf{z})$ gives $d(\mathbf{x} \mid \mathbf{x}^t) \geq \max_{\mathbf{z}} g(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})$; taking $\max_{\mathbf{z}}$ in the tightness condition $g_s(\mathbf{x}^t, \mathbf{z} \mid \mathbf{x}^t) = g(\mathbf{x}^t, \mathbf{z})$ gives $d(\mathbf{x}^t \mid \mathbf{x}^t) = f(\mathbf{x}^t)$. Symmetrically h is a pointwise minimum of functions concave (indeed affine, in our instances) in $\mathbf{z}$, hence concave. ■

C.2 THEOREM 1 (THE MINIMAX SWAP)

Proof. Sion's theorem [56] states that if M is a compact convex subset of a linear topological space, N a convex subset of a linear topological space, and $\phi : M \times N \to \mathbb{R}$ is quasi-convex and lower semicontinuous in its first argument and quasi-concave and upper semicontinuous in its second, then

$$\min_{u \in M} \sup_{v \in N} \phi(u, v) \;=\; \sup_{v \in N} \min_{u \in M} \phi(u, v). \tag{S3}$$

Under hypothesis (i) apply equation S3 with $M = \mathcal{X}$, $N = \mathcal{Z}$, $\phi = g_s(\cdot, \cdot \mid \mathbf{x}^t)$; the supremum over the compact $\mathcal{Z}$ is attained by upper semicontinuity, giving equation 11.

Under hypothesis (ii), $\mathcal{X}$ need not be compact. Fix $\mathbf{z}_0 \in \mathcal{Z}$ for which $S = \{\mathbf{x} \in \mathcal{X} : g_s(\mathbf{x}, \mathbf{z}_0 \mid \mathbf{x}^t) \leq g_s(\mathbf{x}^t, \mathbf{z}_0 \mid \mathbf{x}^t)\}$ is bounded; S is closed by lower semicontinuity and convex by quasi-convexity, hence compact. Any $\mathbf{x} \notin S$ has $\max_{\mathbf{z}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t) \geq g_s(\mathbf{x}, \mathbf{z}_0 \mid \mathbf{x}^t) > g_s(\mathbf{x}^t, \mathbf{z}_0 \mid \mathbf{x}^t) \geq \min_{\mathbf{x}'} \max_{\mathbf{z}} g_s(\mathbf{x}', \mathbf{z} \mid \mathbf{x}^t)$, so no minimizer of $d(\cdot \mid \mathbf{x}^t)$ lies outside S and the outer minimization may be restricted to S without changing its value. Likewise $\min_{\mathbf{x} \in \mathcal{X}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t) \leq \min_{\mathbf{x} \in S} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)$ with equality for every $\mathbf{z}$, because for $\mathbf{x} \notin S$ the value is dominated along the segment towards $\mathbf{x}^t$ by quasi-convexity. Applying equation S3 with $M = S$ and $N = \mathcal{Z}$ then gives equation 11. ■

Remark 9 (The hypotheses in our instances)**.** $\mathcal{Z}$ is always the probability simplex Δ_m, which is convex and compact, and g_s is affine in $\mathbf{z}$ up to the entropic term $\tau H(\mathbf{z})$, which is continuous and concave on Δ_m; so the $\mathbf{z}$-side hypotheses hold with room to spare. On the $\mathbf{x}$-side, for *min–max quadratics*, $g_s(\mathbf{x}, \mathbf{z}) = \sum_k z_k f_k(\mathbf{x}) + \tau H(\mathbf{z})$ with each $\mathbf{H}_k \succ 0$, so $g_s(\cdot, \mathbf{z})$ is strongly convex for every $\mathbf{z} \in \Delta_m$ and has bounded sublevel sets — hypothesis (ii), and also the strictness needed by Proposition 2. *Fair PCA*, $\mathcal{X} = \{\mathbf{X} : \mathbf{X}^\top \mathbf{X} \preceq \mathbf{I}\}$ is convex and compact (it is the unit ball of the spectral norm) and $g_s(\cdot, \mathbf{z} \mid \mathbf{X}^t)$ is affine, hence convex and continuous — hypothesis (i); Remark 12 treats the iterate statement in this case. *Worst-case-group logistic regression*, $g_s(\cdot, \mathbf{z})$ is strongly convex because of the ridge term $\frac{\lambda}{2}\|\mathbf{x}\|^2$ with $\lambda > 0$, so hypothesis (ii) holds, and $\tilde{\mathbf{x}}$ is computed numerically, the case Proposition 7 covers.

C.3 THEOREM 3

Proof. By Lemma 8, $f \le d(\cdot \mid \mathbf{x}^t)$ everywhere and $d(\mathbf{x}^t \mid \mathbf{x}^t) = f(\mathbf{x}^t)$. Hence for any $\mathbf{x}^+$ satisfying the test,

$$f(\mathbf{x}^+) \le d(\mathbf{x}^+ \mid \mathbf{x}^t) \le d(\mathbf{x}^t \mid \mathbf{x}^t) = f(\mathbf{x}^t).$$

No property of $\mathbf{x}^+$ beyond the inequality is used, which is the point, since the map producing $\mathbf{x}^+$ is unconstrained.

For the backtracking step, let $\mathbf{g} = \nabla d(\mathbf{x}^t \mid \mathbf{x}^t) \neq 0$, which exists by Proposition 6 when the inner maximizer is unique. Since d is convex and differentiable at $\mathbf{x}^t$, $d(\mathbf{x}^t - s\mathbf{g} \mid \mathbf{x}^t) = d(\mathbf{x}^t \mid \mathbf{x}^t) - s\|\mathbf{g}\|^2 + o(s)$ as $s \downarrow 0$, so there is $\bar{s} > 0$ with $d(\mathbf{x}^t - s\mathbf{g} \mid \mathbf{x}^t) < d(\mathbf{x}^t \mid \mathbf{x}^t)$ for all $s \in (0, \bar{s}]$. A backtracking sequence $s_j = 2^{-j} s_0$ therefore satisfies the test after at most $\lceil \log_2(s_0/\bar{s}) \rceil$ halvings. If instead $\mathbf{g} = 0$ then $\mathbf{x}^t$ minimizes the convex $d(\cdot \mid \mathbf{x}^t)$, and by Lemma 8 it is a stationary point of f; the layer returns $\mathbf{x}^t$ and monotonicity holds trivially. ■

Remark 10. The cost of the test is one evaluation of $d(\mathbf{x}^+ \mid \mathbf{x}^t)$, i.e. one application of the closed-form elimination $\tilde{\mathbf{z}}(\cdot \mid \mathbf{x}^t)$ followed by an evaluation of g_s. In the worst-case-group instances this is a softmax over m numbers; in fair PCA it is a softmax over the m group scores. In both cases it is negligible relative to the layer's own arithmetic, and $d(\mathbf{x}^t \mid \mathbf{x}^t) = f(\mathbf{x}^t)$ is already known from the previous layer.

C.4 THEOREM 4

Proof. Every layer of Algorithm S1 and of Algorithm S2 returns a point $\mathbf{x}^{t+1}$ that satisfies the test $d(\mathbf{x}^{t+1} \mid \mathbf{x}^t) \le d(\mathbf{x}^t \mid \mathbf{x}^t)$, since either the candidate produced by the inner solver satisfies it and is kept, or the backtracking step of Theorem 3 produces a point that does, or, when $\nabla d(\mathbf{x}^t \mid \mathbf{x}^t) = \mathbf{0}$, the layer returns $\mathbf{x}^{t+1} = \mathbf{x}^t$ and the test holds with equality. In every case Theorem 3 gives $f(\mathbf{x}^{t+1}) \le f(\mathbf{x}^t)$, which is the monotonicity claim of Theorem 4. No property of the weights was used, so the conclusion is independent of $\{\eta_{t,s}\}$ and $\{\mathbf{P}_t\}$.

The sequence $\{f(\mathbf{x}^t)\}$ is therefore non-increasing, and it is bounded below by $f^{\inf} = \inf_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) > -\infty$ by hypothesis; it is bounded above by its first term $f(\mathbf{x}^0)$ because it never increases. A monotone bounded sequence of reals converges, so $f(\mathbf{x}^t) \to f^\infty$ for some $f^\infty \in [f^{\inf}, f(\mathbf{x}^0)]$.

For the last statement, recall the uniform sandwich equation 8, $f \le f_\tau \le f + \tau \log m$, and apply it at each iterate, which gives $f_\tau(\mathbf{x}^t) - \tau \log m \le f(\mathbf{x}^t) \le f_\tau(\mathbf{x}^t)$. Taking limits inferior and superior and using $f_\tau(\mathbf{x}^t) \to f^\infty$ yields $f^\infty - \tau \log m \le \liminf_t f(\mathbf{x}^t) \le \limsup_t f(\mathbf{x}^t) \le f^\infty$, so every limit point of $\{f(\mathbf{x}^t)\}$ lies in $[f^\infty - \tau \log m,\ f^\infty]$. ■

C.5 THEOREM 5

Proof. (a) By Theorem 4 the sequence $\{f_\tau(\mathbf{x}^t)\}$ never increases, so every iterate lies in the sublevel set $\mathcal{S}_0 = \{\mathbf{x} \in \mathcal{X} : f_\tau(\mathbf{x}) \le f_\tau(\mathbf{x}^0)\}$. Under either hypothesis — $\mathcal{X}$ compact, or f_τ with bounded sublevel sets — the set $\mathcal{S}_0$ is bounded, and it is closed by continuity of f_τ, hence compact. A bounded sequence in $\mathbb{R}^n$ has at least one convergent subsequence, so the set of limit points is non-empty; it is an intersection of nested compact sets and therefore compact. If $\mathbf{x}^{t_j} \to \bar{\mathbf{x}}$ along a subsequence, then $f_\tau(\mathbf{x}^{t_j}) \to f_\tau(\bar{\mathbf{x}})$ by continuity, while $f_\tau(\mathbf{x}^{t_j}) \to f^\infty$ because the whole sequence converges to f^∞; hence $f_\tau(\bar{\mathbf{x}}) = f^\infty$.

(b) Suppose $\mathbf{x}^{t+1} \in \arg\min_{\mathbf{x} \in \mathcal{X}} d(\mathbf{x} \mid \mathbf{x}^t)$ and that $d(\cdot \mid \mathbf{x}^t)$ is μ-strongly convex on $\mathcal{X}$ with $\mu > 0$ independent of t. Strong convexity together with optimality of $\mathbf{x}^{t+1}$ gives the quadratic growth inequality

$$d\big(\mathbf{x} \mid \mathbf{x}^t\big) \ge d\big(\mathbf{x}^{t+1} \mid \mathbf{x}^t\big) + \tfrac{\mu}{2}\big\|\mathbf{x} - \mathbf{x}^{t+1}\big\|^2 \qquad \text{for every } \mathbf{x} \in \mathcal{X},$$

and evaluating it at $\mathbf{x} = \mathbf{x}^t$ yields $d(\mathbf{x}^t \mid \mathbf{x}^t) - d(\mathbf{x}^{t+1} \mid \mathbf{x}^t) \ge \frac{\mu}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2$. Combining this with majorization and tightness,

$$f_\tau\big(\mathbf{x}^t\big) - f_\tau\big(\mathbf{x}^{t+1}\big) \ge d\big(\mathbf{x}^t \mid \mathbf{x}^t\big) - d\big(\mathbf{x}^{t+1} \mid \mathbf{x}^t\big) \ge \tfrac{\mu}{2}\big\|\mathbf{x}^{t+1} - \mathbf{x}^t\big\|^2,$$

where the first inequality uses $f_\tau(\mathbf{x}^t) = d(\mathbf{x}^t \mid \mathbf{x}^t)$ and $f_\tau(\mathbf{x}^{t+1}) \le d(\mathbf{x}^{t+1} \mid \mathbf{x}^t)$. Summing over $t = 0, \ldots, T-1$ telescopes the left-hand side to $f_\tau(\mathbf{x}^0) - f_\tau(\mathbf{x}^T) \le f_\tau(\mathbf{x}^0) - f^\infty$, which gives

equation 20 on letting $T \to \infty$. The terms of a convergent series tend to zero, so $\|\mathbf{x}^{t+1} - \mathbf{x}^t\| \to 0$; a bounded sequence whose successive differences tend to zero has a connected set of limit points by a classical argument [47].

(c) Let $\bar{\mathbf{x}}$ be a limit point and $\mathbf{x}^{t_j} \to \bar{\mathbf{x}}$. By *(b)* we also have $\mathbf{x}^{t_j+1} \to \bar{\mathbf{x}}$. Optimality of $\mathbf{x}^{t_j+1}$ for the convex problem $\min_{\mathbf{x}\in\mathcal{X}} d(\mathbf{x} \mid \mathbf{x}^{t_j})$ is the variational inequality $\nabla_{\mathbf{x}} d(\mathbf{x}^{t_j+1} \mid \mathbf{x}^{t_j})^\top (\mathbf{x} - \mathbf{x}^{t_j+1}) \geq 0$ for every $\mathbf{x} \in \mathcal{X}$. Letting $j \to \infty$ and using joint continuity of $(\mathbf{x}, \boldsymbol{y}) \mapsto \nabla_{\mathbf{x}} d(\mathbf{x} \mid \boldsymbol{y})$ gives $\nabla_{\mathbf{x}} d(\bar{\mathbf{x}} \mid \bar{\mathbf{x}})^\top (\mathbf{x} - \bar{\mathbf{x}}) \geq 0$ for every $\mathbf{x} \in \mathcal{X}$. Finally, $d(\cdot \mid \bar{\mathbf{x}}) - f_\tau$ is non-negative by equation 5 and zero at $\bar{\mathbf{x}}$, so $\bar{\mathbf{x}}$ is a minimizer of that difference over $\mathcal{X}$ and, both functions being differentiable, the first-order tangency $\nabla_{\mathbf{x}} d(\bar{\mathbf{x}} \mid \bar{\mathbf{x}}) = \nabla f_\tau(\bar{\mathbf{x}})$ holds. Substituting gives $\nabla f_\tau(\bar{\mathbf{x}})^\top (\mathbf{x} - \bar{\mathbf{x}}) \geq 0$ for every $\mathbf{x} \in \mathcal{X}$, which is first-order stationarity of f_τ over $\mathcal{X}$. ■

Remark 11 (Why (b) and (c) are stated for the exact layer)**.** Parts (b) and (c) use a quantitative decrease of $d(\cdot \mid \mathbf{x}^t)$ at every layer, which an exact solve of the retained problem supplies and which a layer approaches as its number of inner steps grows. For arbitrary learned weights the test of Theorem 3 enforces the decrease $d(\mathbf{x}^{t+1} \mid \mathbf{x}^t) \leq d(\mathbf{x}^t \mid \mathbf{x}^t)$ itself, and part (a) holds with no condition on the weights.

C.6 Proposition 6

Proof. The proposition follows from Danskin's theorem [16, 7]. Let $\tilde{\mathbf{z}}(\mathbf{x})$ be the unique maximizer of the concave problem $\max_{\mathbf{z}\in\mathcal{Z}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)$ and suppose g_s is differentiable in $\mathbf{x}$ with $\nabla_{\mathbf{x}} g_s$ continuous. Then d is differentiable at $\mathbf{x}$ with $\nabla d(\mathbf{x} \mid \mathbf{x}^t) = \nabla_{\mathbf{x}} g_s(\mathbf{x}, \tilde{\mathbf{z}}(\mathbf{x}) \mid \mathbf{x}^t)$. Formally, by the chain rule applied to $\mathbf{x} \mapsto g_s(\mathbf{x}, \tilde{\mathbf{z}}(\mathbf{x}) \mid \mathbf{x}^t)$,

$$\nabla d(\mathbf{x} \mid \mathbf{x}^t) = \nabla_{\mathbf{x}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)\big|_{\mathbf{z}=\tilde{\mathbf{z}}(\mathbf{x})} + \underbrace{\left(\frac{\partial \tilde{\mathbf{z}}}{\partial \mathbf{x}}\right)^\top \nabla_{\mathbf{z}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t)\big|_{\mathbf{z}=\tilde{\mathbf{z}}(\mathbf{x})}}_{=0},$$

the second term being zero because $\tilde{\mathbf{z}}(\mathbf{x})$ maximizes $g_s(\mathbf{x}, \cdot \mid \mathbf{x}^t)$, so $\nabla_{\mathbf{z}} g_s = 0$ there when the maximizer is interior, and lies in the normal cone of $\mathcal{Z}$ with $\partial\tilde{\mathbf{z}}/\partial\mathbf{x}$ mapping into the tangent cone otherwise. The dual statement is identical with the roles of the variables exchanged, the minimizer $\tilde{\mathbf{x}}(\mathbf{z} \mid \mathbf{x}^t)$ of a convex problem replacing the maximizer.

The consequence stated in the proposition is that computing ∇d requires only the *value* $\tilde{\mathbf{z}}(\mathbf{x})$, not its Jacobian. Since $\tilde{\mathbf{z}}$ and $\tilde{\mathbf{x}}$ are given by explicit formulas, a layer of equation 17–equation 18 is a composition of explicit differentiable maps, and the gradient of the network output with respect to $\{\eta_{t,s}\}$ is obtained by ordinary backpropagation. In particular no adjoint linear system or fixed-point iteration is needed, in contrast with implicit layers [4, 3, 6]. ■

C.7 Proposition 2

Proof. Theorem 1 already gives the equality of values,

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{z}\in\mathcal{Z}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t) = \max_{\mathbf{z}\in\mathcal{Z}} \min_{\mathbf{x}\in\mathcal{X}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t), \quad \text{i.e.} \quad \min_{\mathbf{x}} d(\mathbf{x} \mid \mathbf{x}^t) = \max_{\mathbf{z}} h(\mathbf{z} \mid \mathbf{x}^t),$$

so it remains to upgrade equality of values to equality of iterates. Strict convexity–concavity makes d strictly convex and h strictly concave, so both problems have unique solutions $\mathbf{x}^\star$ and $\mathbf{z}^\star$, and the pair $(\mathbf{x}^\star, \mathbf{z}^\star)$ is then the unique saddle point of $g_s(\cdot, \cdot \mid \mathbf{x}^t)$, which means that

$$g_s(\mathbf{x}^\star, \mathbf{z} \mid \mathbf{x}^t) \;\leq\; g_s(\mathbf{x}^\star, \mathbf{z}^\star \mid \mathbf{x}^t) \;\leq\; g_s(\mathbf{x}, \mathbf{z}^\star \mid \mathbf{x}^t) \qquad \forall \mathbf{x} \in \mathcal{X},\ \mathbf{z} \in \mathcal{Z}.$$

The right-hand inequality says exactly that $\mathbf{x}^\star$ minimizes $g_s(\cdot, \mathbf{z}^\star \mid \mathbf{x}^t)$, i.e. $\mathbf{x}^\star = \tilde{\mathbf{x}}(\mathbf{z}^\star \mid \mathbf{x}^t)$. Hence the primal window's answer $\arg\min_x d$ and the dual window's answer $\tilde{\mathbf{x}}(\arg\max_z h)$ coincide. ■

Remark 12 (Fair PCA, where the surrogate is affine)**.** The hypothesis holds for the min–max quadratics, where $g_s(\mathbf{x}, \mathbf{z}) = \sum_k z_k f_k(\mathbf{x}) + \tau H(\mathbf{z})$ is strictly convex in $\mathbf{x}$ (each $\mathbf{H}_k \succ 0$) and strictly concave in $\mathbf{z}$ (the entropy is strictly concave on the simplex), and we verify the identity $\min_x d = \max_z h$ numerically to 5.6×10^{-17}. For fair PCA the tangent majorizer makes g_s *affine* in $\mathbf{X}$. The minimax equality still holds under Sion's theorem [56], since affine functions are quasi-convex and the feasible set $\{\mathbf{X} : \mathbf{X}^\top\mathbf{X} \preceq \mathbf{I}\}$ is convex and compact, and $\arg\min_X g_s(\cdot, \mathbf{z})$ is a singleton whenever $\mathbf{A}(\mathbf{z})$ has full column rank r. When $\mathbf{A}(\mathbf{z})$ is rank deficient any orthonormal

completion of its polar factor is a minimizer, and the two windows may return different (equally optimal) iterates. Our implementation resolves the degeneracy by zeroing the directions with numerically negligible singular values; this affects neither Theorem 3, which never uses uniqueness, nor Proposition 6, which needs it only where the maximizer of the *dual* problem is concerned—and that one is unique for every $\tau > 0$.

C.8 Proposition 7

Proof. Write $\hat{h}(\mathbf{z}) = g_s(\hat{\mathbf{x}}(\mathbf{z}), \mathbf{z} \mid \mathbf{x}^t)$. By assumption $g_s(\hat{\mathbf{x}}(\mathbf{z}), \mathbf{z} \mid \mathbf{x}^t) \le \min_{\mathbf{x}} g_s(\mathbf{x}, \mathbf{z} \mid \mathbf{x}^t) + \epsilon = h(\mathbf{z} \mid \mathbf{x}^t) + \epsilon$, and $\hat{h}(\mathbf{z}) \ge h(\mathbf{z} \mid \mathbf{x}^t)$ trivially since h is the minimum; so $h \le \hat{h} \le h + \epsilon$ pointwise, and maximizing $\hat{h}$ instead of h costs at most ϵ in the reduced dual objective.

For the second claim, observe that the descent test of Theorem 3 evaluates $d(\mathbf{x}^+ \mid \mathbf{x}^t) = \max_{\mathbf{z}} g_s(\mathbf{x}^+, \mathbf{z} \mid \mathbf{x}^t)$, which is computed from the *primal* window's elimination $\tilde{\mathbf{z}}(\cdot \mid \mathbf{x}^t)$ and takes $\mathbf{x}^+$ as a black-box input. Whether $\mathbf{x}^+$ came from an exact or an approximate dual elimination is immaterial. Hence Theorem 3 applies verbatim, and a RUN-D layer built on an inexact $\hat{\mathbf{x}}$ remains monotone. ■

D The elimination maps used in the experiments

Min–max quadratics. $f_k(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}_k^\star)^\top \mathbf{H}_k (\mathbf{x} - \mathbf{x}_k^\star)$ with $\mathbf{H}_k \succ 0$. With entropic smoothing, $g_s(\mathbf{x}, \mathbf{z}) = \sum_k z_k f_k(\mathbf{x}) + \tau H(\mathbf{z})$ is already convex–concave, so no majorization is needed and $g_s = g$. In the primal window, $\tilde{\mathbf{z}}(\mathbf{x}) = \mathrm{softmax}(f(\mathbf{x})/\tau)$ and $d(\mathbf{x}) = \tau \log \sum_k e^{f_k(\mathbf{x})/\tau}$, with $\nabla d(\mathbf{x}) = \sum_k \tilde{\mathbf{z}}_k(\mathbf{x}) \mathbf{H}_k (\mathbf{x} - \mathbf{x}_k^\star)$. In the dual window, $\tilde{\mathbf{x}}(\mathbf{z}) = (\sum_k z_k \mathbf{H}_k)^{-1} (\sum_k z_k \mathbf{c}_k)$ with $\mathbf{c}_k = \mathbf{H}_k \mathbf{x}_k^\star$, and $\nabla_z h(\mathbf{z})_k = f_k(\tilde{\mathbf{x}}(\mathbf{z})) - \tau(\log \mathbf{z}_k + 1)$. The reference value $f_\tau^\star$ is computed by damped Newton on the convex d, using the exact Hessian $\sum_k z_k \mathbf{H}_k + \tau^{-1}(\mathbb{E}_{\mathbf{z}}[\mathbf{g}\mathbf{g}^\top] - \mathbb{E}_{\mathbf{z}}[\mathbf{g}]\mathbb{E}_{\mathbf{z}}[\mathbf{g}]^\top)$.

Fair PCA. With $u_k(\mathbf{X}) = -\operatorname{tr}(\mathbf{X}^\top \mathbf{C}_k \mathbf{X})/\alpha_k$ concave, the tangent majorizer $\bar{u}_k(\mathbf{X} \mid \mathbf{X}^t) = -\big(2\operatorname{tr}((\mathbf{X}^t)^\top \mathbf{C}_k \mathbf{X}) - \operatorname{tr}((\mathbf{X}^t)^\top \mathbf{C}_k \mathbf{X}^t)\big)/\alpha_k$ is affine in $\mathbf{X}$, so $g_s(\mathbf{X}, \mathbf{z} \mid \mathbf{X}^t) = \sum_k z_k \bar{u}_k(\mathbf{X} \mid \mathbf{X}^t) + \tau H(\mathbf{z})$. In the primal window, $\tilde{\mathbf{z}}(\mathbf{X}) = \mathrm{softmax}(\bar{u}(\mathbf{X} \mid \mathbf{X}^t)/\tau)$ and $d(\mathbf{X} \mid \mathbf{X}^t) = \tau \log \sum_k e^{\bar{u}_k(\mathbf{X} \mid \mathbf{X}^t)/\tau}$, whose gradient is $-2\sum_k \tilde{\mathbf{z}}_k \mathbf{C}_k \mathbf{X}^t/\alpha_k$; steps are retracted onto the Stiefel manifold by the polar factor [1]. In the dual window, the problem $\min_{\mathbf{X}^\top \mathbf{X} \preceq \mathbf{I}} -2\operatorname{tr}(\mathbf{A}(\mathbf{z})^\top \mathbf{X})$, with $\mathbf{A}(\mathbf{z}) = \sum_k z_k \mathbf{C}_k \mathbf{X}^t/\alpha_k$, is solved by $\tilde{\mathbf{x}}(\mathbf{z}) = \mathrm{polar}(\mathbf{A}(\mathbf{z}))$ by von Neumann's trace inequality [39], giving $h(\mathbf{z} \mid \mathbf{X}^t) = -2\|\mathbf{A}(\mathbf{z})\|_* + \langle \mathbf{z}, c \rangle + \tau H(\mathbf{z})$ with $\mathbf{c}_k = \operatorname{tr}((\mathbf{X}^t)^\top \mathbf{C}_k \mathbf{X}^t)/\alpha_k$; the subgradient of the nuclear norm at $\mathbf{A}$ is its polar factor, so $\nabla_z h(\mathbf{z})_k = -2\operatorname{tr}\big(\mathrm{polar}(\mathbf{A}(\mathbf{z}))^\top \mathbf{C}_k \mathbf{X}^t\big)/\alpha_k + \mathbf{c}_k - \tau(\log \mathbf{z}_k + 1)$.

Worst-case-group logistic regression. f_k is the masked-mean logistic loss on group k plus $\frac{\lambda}{2}\|\mathbf{x}\|^2$. The primal window is as for the quadratics. The dual window is *not* closed form; we approximate it by N Newton steps on $\sum_k z_k f_k$, whose Hessian is $\sum_k z_k \big(\mathbf{X}_k^\top \operatorname{diag}(\sigma_k(1 - \sigma_k)) \mathbf{X}_k / n_k + \lambda \mathbf{I}\big)$. Proposition 7 covers this case.

E The compared methods

Each method below is implemented from its standard description, and those introduced by a specific paper carry its year and citation in the tables. This section says what each one does; the protocol under which all of them are run is Supplementary F.

ERM-GD. Gradient descent on the average loss over the groups. It optimizes the wrong quantity, since nothing in the average constrains the worst group, so whatever worst-group value it reaches is incidental.

Subgradient. Subgradient descent applied to the pointwise maximum itself. The maximum is non-differentiable exactly where two branches are tied, which is where the worst-case solution lives,

so a step carries information from one active branch alone; convergence then requires a decaying step size, and at the depths these tables report the iterate is still far from the optimum.

Optimistic GDA [17]. Gradient descent–ascent with an extrapolation term that anticipates the opponent's next move, which damps the rotation about the saddle. The two players remain, so two step sizes must be set whose ratio governs stability, and no step is guaranteed to decrease the worst-case objective.

GDA [35]. Two-timescale gradient descent–ascent, the primal variable moving on the slower scale. The separation of timescales is what buys the analysis and what limits the method, since the primal step must be kept small enough for the scale separation to hold, and the guarantee again concerns an approximate stationary point of a two-player problem and says nothing about the objective that was posed.

Online group-DRO [50]. Exponentiated-gradient updates on the group weights interleaved with gradient steps on the predictor. It is a saddle-point method in a different notation, and it inherits the same limitation, since its guarantee attaches to an average of the iterates while the network that is deployed is the last one.

Group-blind PCA. Principal component analysis on all groups at once, included in the fair-PCA tables as the value the worst-case formulation has to beat. It has no mechanism for serving the worst-off group, and the tables show the distance that costs.

MM4MM-P and MM4MM-D [51]. The two fixed-step schemes our networks unfold, one per window, with a single constant step tuned per problem. One constant must serve every instance of a set, so the scheme cannot adapt to an instance whose conditioning differs from the average, which is the gap the unrolled network is built to close.

UNROLLED GDA. Gradient descent–ascent unrolled into a network whose weights are its two step sizes per layer, trained exactly as our own networks are. It keeps the saddle point, and with it the two-player instability, inside every layer; Supplementary F describes its construction.

F EXPERIMENTAL DETAILS

Compute. All experiments run in double precision on the six cores of an AMD Ryzen 5 5600H without a GPU, and the full suite takes a few hours. Wall-clock numbers are best-of-three per configuration, measured with the same thread count.

Training. Adam [31], learning rate 5×10^{-2} with cosine decay, gradient-norm clipping at 10, full batch, 150 steps (200 for the worst-case-group problem and the ablations). Step sizes are stored in log space and initialized at the tuned constant of the corresponding fixed-step scheme, so training begins from a working solver. For the quadratics the training loss is the mean log suboptimality, which is also the reported metric (its exponential, the geometric mean); suboptimality spans many orders of magnitude, so the arithmetic mean would be dominated by the single worst instance.

Min–max quadratics. $\mathbf{H}_k = \mathbf{Q}_k \operatorname{diag}(\lambda)\, \mathbf{Q}_k^\top$ with $\mathbf{Q}_k$ the orthogonal factor of a Gaussian matrix and λ log-spaced in $[1/\kappa, 1]$; branch minimizers $\mathbf{x}_k^\star$ drawn $\mathcal{N}(0, \mathbf{I}/d)$ scaled by the spread, so each branch has minimum value 0. Default $d = 50$, $m = 5$, $\tau = 0.05$; 24 instances per set; 3 seeds. Inner steps $S = 4$ for RUN-P and $S = 2$ for RUN-D.

Fair PCA. For DIGITS the ten classes of scikit-learn `digits` [48] are the groups, $d = 64$, $r = 4$, 35 points per group per instance, disjoint $60/40$ train/test splits; the cap of 35 is well below the smallest class count in the held-out set, so instances genuinely vary. SYNTHETIC has $m = 6$ groups in $d = 30$ with independently drawn dominant rank-3 subspaces. $\tau = 0.02$; five seeds, three at ranks 2 and 8. Budgets are counted in polar factorizations, of which RUN-D spends $T(S{+}1)$ and RUN-P spends TS.

Worst-case-group risk. SPURIOUS has 8 core features with class means $\pm\mu$ and instance noise $\sigma_c \sim \mathcal{U}[1.2, 2.2]$, 2 spurious features with means $\pm 2\nu$ and noise 0.3, majority fraction $p \sim \mathcal{U}[0.80, 0.98]$, four (y, a) groups of up to 64 points. DIGITS takes the label "digit is odd" and the attribute "digit ≥ 5", four groups, $p \sim \mathcal{U}[0.80, 0.95]$; worst-group accuracy is measured on group-balanced instances from the held-out set. $\lambda = 10^{-3}$, $\tau = 0.05$; five seeds.

Tuning the competing methods. Every competing method is implemented from its published description and run with the constants that description prescribes; we change none of them. What those descriptions leave free for a new problem is the step size, which we grid-search for every method on the *training* set only, selecting by mean objective on that set. Our own networks receive no search of their own and start at the constant selected for the algorithm they unfold. The grids, and the number of inner steps used by each nested scheme, are recorded in the released `run_*.py` scripts.

The saddle-point methods need one further remark, because they have *two* free step sizes whose ratio governs whether the iterates approach the saddle or circle it. We therefore search the two jointly over the product of the two grids, so that neither is held at a value tuned for the other. Their dual variable lives on the simplex, so their ascent step is taken in the entropy geometry, the strongest standard choice there, and their primal step is retracted onto the feasible set whenever it is not the whole space; both partial gradients are obtained from a single evaluation of the branch functions, so an iteration of a saddle-point method costs one gradient and the budget is matched in the same currency as everywhere else.

UNROLLED GDA follows the design of learned primal–dual [2], whose convolutional proximal blocks belong to imaging and have no counterpart in these problems, so the two-player recursion is the architecture and its two step sizes per layer are the weights. It is the strictest opponent in these tables, because it keeps the saddle point and receives exactly the training our own networks receive, and it is therefore the row that separates what the *reduction* contributes from what the *training* contributes. It is initialized at the constants selected for tuned GDA and reverts to them if training fails to improve the training-set objective, the same model-selection rule our own networks obey, so its row can never fall below the algorithm it starts from.

G ADDITIONAL RESULTS

G.1 THE COMPLETE WINDOW-CHOICE STUDY

Table 1 shows two condition numbers and two branch counts, which are the settings the discussion in Section 6 turns on. Table S1 gives every setting, adding $\kappa \in \{10^2, 10^3\}$ and $m \in \{5, 20\}$. Nothing in it changes the reading, since across the intermediate settings too the dual window's advantage grows with κ and decays with m until the accuracy limit of double precision caps it. Both tables are generated from the same measurements by the same script, so they cannot disagree.

G.2 THE VALUE OF UNFOLDING A WINDOW

A single grid-searched step must serve a whole set of instances at once. To measure how the value of learning depends on their diversity, we repeat the $\kappa = 10^3$ study of Table S1, where all instances share one condition number, with each drawing its own from $\kappa \sim \log\mathcal{U}[10, 10^4]$ and everything else fixed. Table S2 reports the tuned algorithm's suboptimality divided by the learned network's, so higher is better; "–" marks no gain, and † marks a ratio whose denominator sits on the 10^{-16} measurement floor and is therefore a lower bound.

In the primal window heterogeneity is what brings the gain out, doubling it at $T = 2$ and raising it from under $1.5\times$ to nearly $6\times$ at $T = 4$, the depths at which one constant step fits the spread of conditioning least well. In the dual window the learned network gains on both sets, by two orders of magnitude at $T = 2$ without heterogeneity and by nearly three at $T = 4$, where it reaches the 10^{-16} floor first and the ratio is a lower bound, and by up to $13\times$ with it. Learning therefore pays in both windows on heterogeneous instances, the regime real distributions resemble, and in the dual window on homogeneous ones as well.

Table S1: The complete window-choice study on min–max quadratics, over every condition number and every branch count at each depth T, of which Table 1 shows two of each. Entries are *correct digits* in the objective; higher is better.

method	T=2	T=4	T=8	T=16	method	T=2	T=4	T=8	T=16
varying the condition number κ					*varying the number of branches* m				
$\kappa = 10$					$m = 2$				
Optimistic GDA (2018) [17]	1.8	2.3	3.1	4.5	Optimistic GDA (2018) [17]	1.5	1.8	2.0	2.3
GDA (2020) [35]	2.2	2.7	3.4	5.2	GDA (2020) [35]	1.8	2.0	2.2	2.4
MM4MM-P (2024) [51]	3.8	5.5	8.6	13.6	MM4MM-P (2024) [51]	2.2	2.7	3.2	3.9
MM4MM-D (2024) [51]	4.4	5.8	8.2	12.9	MM4MM-D (2024) [51]	9.5	12.7	**>16**	**>16**
Unrolled GDA (control)	2.6	3.7	5.7	9.5	Unrolled GDA (control)	2.0	2.3	2.5	2.8
RUN-P (ours)	5.0	8.8	14.1	14.8	RUN-P (ours)	2.5	2.7	3.3	3.9
RUN-D (ours)	**7.4**	**11.4**	**15.4**	**>16**	RUN-D (ours)	**15.7**	**>16**	**>16**	**>16**
$\kappa = 100$					$m = 5$				
Optimistic GDA (2018) [17]	1.7	2.0	2.5	3.3	Optimistic GDA (2018) [17]	1.7	1.9	2.2	2.7
GDA (2020) [35]	2.5	3.0	4.0	5.8	GDA (2020) [35]	2.2	2.6	3.2	4.0
MM4MM-P (2024) [51]	2.7	3.5	4.9	7.0	MM4MM-P (2024) [51]	3.1	4.0	5.4	7.8
MM4MM-D (2024) [51]	8.7	13.3	**>16**	**>16**	MM4MM-D (2024) [51]	8.7	12.9	**>16**	**>16**
Unrolled GDA (control)	2.6	3.6	5.5	9.2	Unrolled GDA (control)	2.4	3.1	4.5	4.4
RUN-P (ours)	3.9	6.7	6.2	7.5	RUN-P (ours)	3.4	4.1	5.5	7.9
RUN-D (ours)	**9.2**	**15.0**	**>16**	**>16**	RUN-D (ours)	**10.7**	**>16**	**>16**	**>16**
$\kappa = 1000$					$m = 20$				
Optimistic GDA (2018) [17]	1.7	1.9	2.2	2.7	Optimistic GDA (2018) [17]	2.4	2.7	2.9	3.6
GDA (2020) [35]	2.2	2.6	3.2	4.0	GDA (2020) [35]	2.7	3.2	4.5	6.6
MM4MM-P (2024) [51]	3.1	4.0	5.4	7.8	MM4MM-P (2024) [51]	4.7	6.7	10.3	15.6
MM4MM-D (2024) [51]	8.7	12.9	**>16**	**>16**	MM4MM-D (2024) [51]	6.7	9.6	15.0	**>16**
Unrolled GDA (control)	2.4	3.1	4.5	4.4	Unrolled GDA (control)	3.1	4.2	6.2	10.0
RUN-P (ours)	3.4	4.1	5.5	7.9	RUN-P (ours)	6.1	10.5	15.6	14.8
RUN-D (ours)	**10.7**	**>16**	**>16**	**>16**	RUN-D (ours)	**8.1**	**13.0**	**>16**	**>16**
$\kappa = 10000$					$m = 50$				
Optimistic GDA (2018) [17]	1.7	1.8	2.1	2.5	Optimistic GDA (2018) [17]	2.8	3.2	3.4	4.4
GDA (2020) [35]	2.1	2.4	2.8	3.3	GDA (2020) [35]	2.9	3.5	4.9	7.4
MM4MM-P (2024) [51]	2.8	3.3	4.1	5.3	MM4MM-P (2024) [51]	5.7	8.5	13.7	**>16**
MM4MM-D (2024) [51]	8.3	11.9	**>16**	**>16**	MM4MM-D (2024) [51]	6.4	9.2	14.4	**>16**
Unrolled GDA (control)	2.3	2.9	3.6	3.9	Unrolled GDA (control)	3.4	4.4	6.4	10.5
RUN-P (ours)	3.0	3.6	4.1	5.3	RUN-P (ours)	7.6	**13.2**	14.8	**>16**
RUN-D (ours)	**12.4**	**>16**	**>16**	**>16**	RUN-D (ours)	**7.8**	12.5	**>16**	**>16**

Table S2: Speed-up of the unrolled network over the tuned constant-step algorithm it is initialized at, within the *same* window.

learned / tuned	*homogeneous*, $\kappa = 10^3$				*heterogeneous*, $\kappa \sim \log \mathcal{U}[10, 10^4]$			
	T=2	T=4	T=8	T=16	T=2	T=4	T=8	T=16
primal window	2×	1×	1×	1×	4×	6×	1×	–
dual window	111×	842×†	–	–	2×	13×	3×†	2×†

G.3 WORST-GROUP ACCURACY ON THE GROUP-RISK PROBLEMS

Table 2 reports the worst-group objective, which is the quantity every method in it optimizes. Table S3 reports worst-group accuracy for the same runs, measured on held-out instances, group-balanced ones for DIGITS. The forced dual window leads at every depth on SPURIOUS and on DIGITS up to $T = 16$, and remains competitive at $T = 32$, so its advantage in the objective of Table 2 carries over to the decisions it produces. Both tables are generated from the same result files.

G.4 FAIR PCA AT A MATCHED BUDGET

Table S4 accompanies Section 6 and reports the worst-group explained variance at a matched budget P of polar factorizations, on the ten classes of DIGITS and on a synthetic set of six groups. The dual window leads every competing method at every budget up to $P = 32$ on both datasets and is on a par with the best of them at $P = 64$. It is the better of our two windows because the polar factor solves the primal subproblem exactly, where the primal window must walk to the same point along the Stiefel manifold. The margin is widest where the budget is tightest — at $P = 8$ the gain over the best competitor is two and a half seed-to-seed standard deviations on DIGITS and two on SYNTHETIC — and narrows as the budget grows, as it must, since every reasonable method reaches

Table S3: Worst-group *accuracy* in per cent for the runs of Table 2, at depth T, measured on held-out instances; higher is better.

method	T=4	T=8	T=16	T=32
SPURIOUS				
ERM-GD	69.75	70.88	71.53	71.11
Subgradient	64.58	67.84	69.11	70.29
Optimistic GDA (2018) [17]	69.61	70.19	71.19	71.50
Online group-DRO (2020) [50]	70.71	71.14	71.23	71.22
GDA (2020) [35]	70.61	71.25	71.53	71.75
MM4MM-P (2024) [51]	68.95	70.12	71.03	71.74
Unrolled GDA (control)	70.87	70.85	71.15	71.86
RUN-P (ours)	68.46	70.17	71.10	71.74
RUN-D (Newton-3, ours)	**71.97**	**72.78**	**72.60**	**72.59**
DIGITS				
ERM-GD	61.91	67.64	67.05	73.47
Subgradient	36.62	49.38	65.06	71.22
Optimistic GDA (2018) [17]	67.23	68.73	70.58	72.12
Online group-DRO (2020) [50]	67.92	70.05	71.12	72.99
GDA (2020) [35]	62.84	64.09	70.70	**73.93**
MM4MM-P (2024) [51]	67.52	68.67	70.03	71.21
Unrolled GDA (control)	69.61	70.72	72.24	**73.93**
RUN-P (ours)	67.76	70.18	70.33	71.66
RUN-D (Newton-3, ours)	**72.58**	**73.31**	**73.34**	73.34

Table S4: Fair PCA at a matched budget P of polar factorizations, worst-group explained variance as a percentage of what that group reaches alone; higher is better.

method	P=8	P=16	P=32	P=64
DIGITS (d=64, m=10, r=4)				
Group-blind PCA	36.23 ±0.88	36.23 ±0.88	36.23 ±0.88	36.23 ±0.88
Subgradient	47.82 ±0.54	49.28 ±0.50	50.27 ±0.44	50.89 ±0.38
Optimistic GDA (2018) [17]	49.51 ±0.52	50.98 ±0.43	51.20 ±0.38	**51.38 ±0.30**
GDA (2020) [35]	47.37 ±0.67	48.55 ±0.57	50.85 ±0.47	51.33 ±0.31
MM4MM-P (2024) [51]	49.75 ±0.49	50.48 ±0.42	50.85 ±0.41	51.08 ±0.38
MM4MM-D (2024) [51]	49.27 ±0.74	50.29 ±0.20	51.02 ±0.39	51.21 ±0.35
Unrolled GDA (control)	49.28 ±0.21	50.27 ±0.45	50.92 ±0.39	50.89 ±0.41
RUN-P (ours)	50.46 ±0.46	50.74 ±0.46	50.93 ±0.39	51.13 ±0.36
RUN-D (S=1, ours)	50.80 ±0.45	**51.16 ±0.36**	**51.29 ±0.33**	51.37 ±0.32
RUN-D (S=3, ours)	**50.92 ±0.41**	50.84 ±0.33	51.24 ±0.34	51.34 ±0.32
SYNTHETIC (d=30, m=6, r=3)				
Group-blind PCA	28.27 ±0.43	28.27 ±0.43	28.27 ±0.43	28.27 ±0.43
Subgradient	36.50 ±0.50	37.57 ±0.31	38.17 ±0.30	38.58 ±0.30
Optimistic GDA (2018) [17]	37.37 ±0.40	38.58 ±0.26	39.07 ±0.32	**39.13 ±0.33**
GDA (2020) [35]	36.07 ±0.39	36.02 ±0.38	36.83 ±0.47	38.41 ±0.45
MM4MM-P (2024) [51]	38.35 ±0.32	38.73 ±0.30	38.90 ±0.31	38.99 ±0.32
MM4MM-D (2024) [51]	38.00 ±0.34	38.88 ±0.32	38.98 ±0.32	39.11 ±0.31
Unrolled GDA (control)	38.27 ±0.32	38.62 ±0.34	38.96 ±0.29	38.87 ±0.61
RUN-P (ours)	38.68 ±0.28	38.80 ±0.32	38.92 ±0.32	39.00 ±0.32
RUN-D (S=1, ours)	39.00 ±0.33	39.07 ±0.32	**39.12 ±0.31**	39.11 ±0.34
RUN-D (S=3, ours)	**39.03 ±0.31**	**39.09 ±0.32**	39.07 ±0.32	39.10 ±0.31

the same subspace eventually; by $P = 64$ the whole spread across the four leading methods is under a sixth of a standard deviation. The reduction is built for the short budget, and there it is clearly ahead. Both unrolled variants also improve on the algorithm they unfold at every budget on DIGITS, and on SYNTHETIC at every budget but the largest, where the dual window has converged and RUN-D and MM4MM-D agree to within the last reported digit. Group-blind PCA, which ignores the group structure, is included as the floor the worst-case formulation has to beat, and it clears that floor by a wide margin at every budget.

G.5 DEPTH GENERALIZATION

Figure S1 accompanies Section 6. Panel (a) plots the suboptimality reached by networks trained at $T = 8$ and then run for up to $T = 128$ layers, with and without the descent test; panel (b) plots the fraction of layers on which the test rejects the learned step and backtracks instead. With the test the primal network reaches 10^{-16} at $T = 128$, and without it the same network diverges. Panel (b) explains why, since the test is active on more than 80 per cent of primal layers throughout, while on

dual layers the share rises from 4 at $T = 8$ to 40 at $T = 128$; the guarantee is therefore spent where it is needed and is nearly free at the depth the network was trained for.

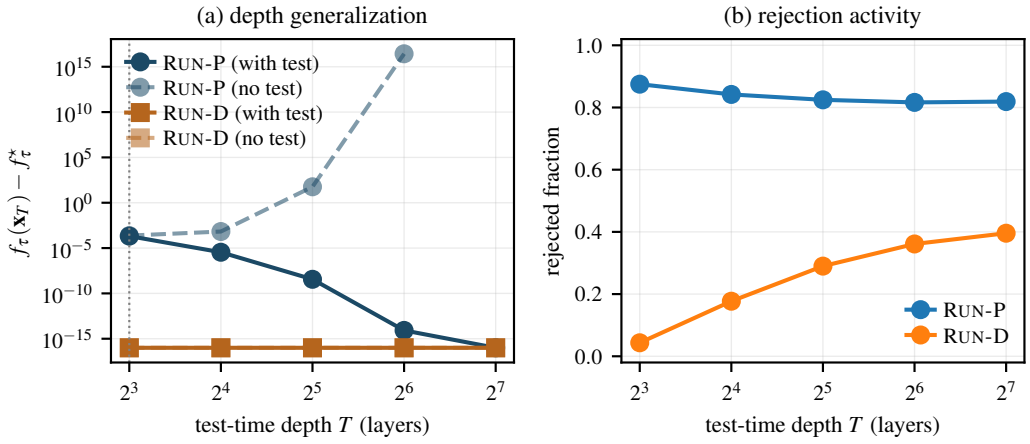

Figure S1: Depth generalization and rejection activity ($\kappa = 10^3$, trained at $T = 8$).

G.6 Fair PCA across target ranks

Table S5 repeats the fair-PCA comparison of Section 6 at target ranks $r \in \{2, 4, 8\}$. The dual window is the best method at nearly every rank and budget, and each unrolled variant improves on the algorithm it unfolds in every cell. The margin is widest at the smaller ranks, where the worst-case tension is greatest. At $r = 8$ even group-blind PCA reaches 58 per cent, so little tension is left to resolve, and at the largest budget the dual window and the trained control are on a par, a fifth of a standard deviation apart, the same convergence at a generous budget that Section 6 describes.

G.7 Scaling the primal dimension

Table S6 varies $d \in \{50, 100, 200\}$ at fixed $\kappa = 10^3$ and $m = 5$. The accuracy ordering is unchanged as d grows by a factor of four, since the dual window still reaches the 10^{-16} floor by $T = 8$ at every d, and the design rule of Section 3.2 therefore turns on *which* elimination is exact, whatever the size of the problem. The dual window reaches that floor, which the primal window does not reach within sixteen layers, at about twice its cost per instance across the whole range, both windows being dominated at these dimensions by the same dense linear algebra.

G.8 Ablations on the budget split and the learned parameterization

The right-hand panel of Table S7 compares four parameterizations of the learned layer on the $\kappa = 10^3$ set at $T = 8$, over three seeds. A *single* step size shared across all layers is the most effective, improving on the tuned constant step by a factor of 45, and also the cheapest, adding one scalar to the model; with 24 training instances it generalizes better than the per-layer and preconditioned variants. The left-hand panel shows that the budget split needs no tuning, since dividing at most 32 polar factorizations between outer layers and inner steps moves the worst-group explained variance by less than two parts in a thousand across $S \in \{1, 2, 4, 8\}$.

No separate ablation on the initialization is needed, because the right-hand panel of Table S7 is one. Every network begins at the constant step size grid-searched for the algorithm it unfolds, and in the main experiments it returns to that value whenever training does not improve the training-set objective. The first row of the panel is that constant step on its own, and every row beneath it is the same network after training, reported without that return so that the panel shows what training does to each parameterization. The distance between them is what training contributes, measured from an algorithm that has already been tuned.

Table S5: Fair PCA across target ranks r on DIGITS, worst-group explained variance in per cent; higher is better.

method	P=8	P=16	P=32
rank $r = 2$			
Group-blind PCA	18.44	18.44	18.44
Subgradient	31.01	34.46	36.96
Optimistic GDA (2018) [17]	31.84	31.49	37.10
GDA (2020) [35]	29.76	29.56	33.45
MM4MM-P (2024) [51]	34.38	37.26	38.61
MM4MM-D (2024) [51]	33.19	37.36	39.02
Unrolled GDA (control)	34.36	37.14	39.07
RUN-P (ours)	36.32	37.63	38.67
RUN-D (S=1, ours)	**37.28**	**39.41**	**40.24**
RUN-D (S=3, ours)	36.80	38.69	39.68
rank $r = 4$			
Group-blind PCA	36.23	36.23	36.23
Subgradient	47.82	49.28	50.27
Optimistic GDA (2018) [17]	49.51	50.98	51.20
GDA (2020) [35]	47.37	48.55	50.85
MM4MM-P (2024) [51]	49.75	50.48	50.85
MM4MM-D (2024) [51]	49.27	50.29	51.02
Unrolled GDA (control)	49.28	50.27	50.92
RUN-P (ours)	50.46	50.74	50.93
RUN-D (S=1, ours)	50.80	**51.16**	**51.29**
RUN-D (S=3, ours)	**50.92**	50.84	51.24
rank $r = 8$			
Group-blind PCA	58.03	58.03	58.03
Subgradient	62.80	63.46	64.02
Optimistic GDA (2018) [17]	63.99	64.39	64.41
GDA (2020) [35]	62.79	63.77	64.29
MM4MM-P (2024) [51]	63.93	64.13	64.30
MM4MM-D (2024) [51]	63.85	64.17	64.34
Unrolled GDA (control)	64.23	64.40	**64.47**
RUN-P (ours)	64.16	64.25	64.34
RUN-D (S=1, ours)	**64.43**	64.39	64.43
RUN-D (S=3, ours)	64.38	**64.43**	64.42

Table S8 reports the inference time per instance on the two application problems, measured at the largest budget each study uses. The dual window is the cheaper of our two networks on both, which is the reverse of what its exact inner solve suggests and follows from the budget it needs, since the layer that solves its subproblem outright needs fewer of them. On the worst-case-group problem the reduction is ahead on total cost as well as on accuracy, since the accuracy of Table 2 is reached in a quarter of the depth while a layer of the budget costs two to three steps of a saddle-point method. The reading to take from the table is therefore that an exact elimination is not an expensive luxury; it is what lets a network be short.

Table S6: Scaling the primal dimension d on min–max quadratics. Entries are *correct digits* in the objective, $-\log_{10}$ of the suboptimality against the damped-Newton reference; higher is better, and >16 marks the accuracy limit of double-precision arithmetic.

method	T=2	T=4	T=8	T=16	ms/inst
$d = 50$					
MM4MM-P (2024) [51]	3.1	4.0	5.4	7.8	4.6
MM4MM-D (2024) [51]	8.7	12.9	**>16**	**>16**	7.0
RUN-P (ours)	3.4	4.1	5.5	7.9	3.0
RUN-D (ours)	**10.7**	**>16**	**>16**	**>16**	5.6
$d = 100$					
MM4MM-P (2024) [51]	3.1	4.0	5.3	7.7	12.3
MM4MM-D (2024) [51]	9.2	13.7	**>16**	**>16**	27.0
RUN-P (ours)	3.5	4.5	5.3	7.7	11.7
RUN-D (ours)	**12.4**	**>16**	**>16**	**>16**	24.1
$d = 200$					
MM4MM-P (2024) [51]	3.1	4.0	5.3	7.7	32.8
MM4MM-D (2024) [51]	9.4	14.2	**>16**	**>16**	86.6
RUN-P (ours)	3.5	3.9	5.3	7.7	33.9
RUN-D (ours)	**13.7**	**>16**	**>16**	**>16**	76.9

Table S7: **Left**, dividing a budget of at most 32 polar factorizations between outer layers $T = \lfloor 32/(S{+}1) \rfloor$ and inner steps S on fair PCA; **right**, which free parameters are worth learning, on min–max quadratics at condition number $\kappa = 10^3$ and depth $T = 8$.

budget split	worst-group variance (%) ↑
$S = 1$ ($T = 16$)	50.97
$S = 2$ ($T = 10$)	51.03
$S = 4$ ($T = 6$)	**51.04**
$S = 8$ ($T = 3$)	50.89

learned quantities	suboptimality ↓
fixed step (tuned MM4MM)	$4.3{\cdot}10^{-6}$
learn step, tied	$\mathbf{9.5{\cdot}10^{-8}}$
learn step per layer	$4.1{\cdot}10^{-6}$
learn step + preconditioner	$3.9{\cdot}10^{-5}$

Table S8: Inference cost at the largest budget. CPU, double precision, best of three, batched over instances.

Fair PCA (DIGITS)		Worst-case group (SPURIOUS)	
method	ms/instance	method	ms/instance
Group-blind PCA	0.743	ERM-GD	0.340
Subgradient	12.523	Subgradient	0.709
Optimistic GDA (2018) [17]	14.184	Optimistic GDA (2018) [17]	1.043
GDA (2020) [35]	14.200	Online group-DRO (2020) [50]	0.767
MM4MM-P (2024) [51]	26.917	GDA (2020) [35]	0.925
MM4MM-D (2024) [51]	15.116	MM4MM-P (2024) [51]	4.570
Unrolled GDA (control)	14.640	Unrolled GDA (control)	0.955
RUN-P (ours)	34.359	RUN-P (ours)	5.319
RUN-D (S=1, ours)	26.312	RUN-D (Newton-3, ours)	2.278
RUN-D (S=3, ours)	13.894		