DFWLayer: Differentiable Frank-Wolfe Optimization Layer

Zixuan Liu[1], Liu Liu[2], Xueqian Wang[1], Peilin Zhao[2]
[1]Tsinghua University, [2]Tencent AI Lab
{zx-liu21@mails,wang.xq@sz}.tsinghua.edu.cn,
{leonliuliu,masonzhao}@tencent.com

Abstract

Differentiable optimization has received a significant amount of attention due to its foundational role in the domain of machine learning based on neural networks. This paper proposes a differentiable layer, named Differentiable Frank-Wolfe Layer (DFWLayer), by rolling out the Frank-Wolfe method, a well-known optimization algorithm which can solve constrained optimization problems without projections and Hessian matrix computations, thus leading to an efficient way of dealing with large-scale convex optimization problems with norm constraints. Experimental results demonstrate that the DFWLayer not only attains competitive accuracy in solutions and gradients but also consistently adheres to constraints.

1 Introduction & Related Work

Recent years have witnessed a variety of combining neural networks and conventional optimization as differentiable optimization layers to integrate expert knowledge into machine learning systems. With objectives and constraints deriving from that knowledge, the output of each differentiable optimization layer is the solution to a specific optimization problem whose parameters are outputs from previous layers (Amos & Kolter, 2017; Agrawal et al., 2019; Sun et al., 2022; Landry, 2021). Nevertheless, adapting the implicit functions (mapping parameters to solutions) to the training procedure of deep learning architecture is not easy since the explicit differentiable closed-form solutions are not available for most conventional optimization algorithms. On top of directly obtaining differentiable closed-form solutions, one alternative is to recover gradients respect to some parameters after solving for the optimal solution (Landry, 2021).

In this study, we introduce a novel differentiable unrolling optimization layer designed to enhance the speed of both optimization and backpropagation processes in the presence of norm constraints. Drawing inspiration from the Frank-Wolfe algorithms (Frank et al., 1956), also known as conditional gradient algorithms, we have developed the Differentiable Frank-Wolfe Layer (DFWLayer). This layer is specifically tailored to efficiently handle norm constraints where projection onto the corresponding feasible region is computationally expensive.

In terms of related work, two main categories of recovering gradients for optimal solutions have emerged: differentiating the optimality conditions (Amos & Kolter, 2017; Agrawal et al., 2019) and rolling out solvers (Donti et al., 2020). Furthermore, Sun et al. (2022) leverage on alternating direction method of multipliers (ADMM) (Boyd et al., 2011) to differentiate the optimality conditions at each iterative step. However, these existing methods often encounter challenges. They can be computationally intensive or suffer from prolonged convergence times, leading to suboptimal solutions that potentially diminish the overall system performance (Bambade et al., 2023).

2 Methodology

In this section, we consider such an convex optimization problem with norm constraints, $\min_x f(x; \theta)$ s.t. $\|w \circ x\| \leq t$, where $f : \mathbb{R}^n \to \mathbb{R}$ is convex and L-smooth, and $\mathcal{C}(\theta)$ is the convex feasible region; $x \in \mathbb{R}^n$ is the variable, and $\theta \in \mathbb{R}^m$ are the parameters of the optimization problem; $w \in \mathbb{R}^n$ is a weight vector, $t \in \mathbb{R}^+$ is a non-negative constant, and $\circ$ denotes the Hadamard production.

Given the widespread use of ℓ_p-norm constraints[1] in various applications, we considering ℓ_p-norm constraints, $\|w \circ x\|_p \le t$, so that the linear approximation of the original objective problem, $\arg\min_{s \in \mathcal{C}} \langle \nabla_x f(x_k), s \rangle$, can solved by $s_{k,i} = -\frac{\alpha t}{w} \cdot \text{sign}(g_{tw,i}(x_k)) \cdot |g_{tw,i}(x_k)|^{\frac{p}{q}}$, where $g_{tw}(\cdot) = \frac{t}{w} \circ \nabla_x f(\cdot)$, $i = 1, ..., n$ stands for the ith entry of the vector, q is a positive constant such that $\frac{1}{p} + \frac{1}{q} = 1$, and α is a constant such that $\|s_k\|_q = 1$. It is noted that, when $p > 1$, $s_{k,i}$ is differentiable and can directly integrated into automatic differentiation.

When $p = 1$, a differentiable expression of vertex is $\hat{s}_k = -\frac{t}{w} \circ \text{sign}(g_{tw}(\hat{x}_k)) \circ \text{softmax}(r(\hat{x}_k)/\tau)$ with probability approximation, where $r(\cdot) = |g_{tw}(\cdot)|$, and $\hat{p}_k(\tau) = \hat{p}(\cdot \mid r(\hat{x}_k); \tau)$ for simplicity, and τ is the temperature. According to Theorem 2.1, an annealing temperature is designed as $\tau_k = 2^{-k//T}$, which decreases per T steps, to obtain reliable solutions and gradients at the same time.

Theorem 2.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a L-smooth convex function on a convex region $\mathcal{C}$ with diameter M and $x^* = \arg\min_{x \in \mathcal{C}} f(x)$. Under Assumption A.1, the suboptimality gap of DFWLayer for ℓ_1 norm constraints is bounded by*

$$h(\hat{x}_k) = f(\hat{x}_k) - f(x^*) \le \delta(\tau_k) + \frac{5LM^2}{2k+4}. \tag{1}$$

The proof of Theorem 2.1 is left to Appendix A.2. Therefore, through the differentiable step size $\gamma_k = \min\{\frac{\langle \nabla_x f(\hat{x}_k), \hat{x}_k - \hat{s}_k \rangle}{L\|\hat{x}_k - \hat{s}_k\|^2}, 1\}$ and the Frank-Wolfe update $\hat{x}_{k+1} = (1 - \gamma_k)\hat{x}_k + \gamma_k \hat{s}_k$[2], the automatic differentiation can be involved to obtain the derivatives $\frac{\partial \hat{x}^*}{\partial \theta}$ for general ℓ_p-norm constraints.

3 EXPERIMENTAL RESULTS

In this section, we evaluate DFWLayer with several experiments to validate its performance to solve different-scale quadratic programs regarding the efficiency and accuracy, compared with state-of-the-art methods. Besides, the results for robotics tasks are left to Appendix A.3.2. We choose CvxpyLayer (Agrawal et al., 2019) and Alt-Diff (Sun et al., 2022) as the baselines. The code is available in `https://github.com/Panda-Shawn/DFWLayer`.

As is shown in Table 1, DFWLayer performs much more efficiently than other two baselines and the speed gap becomes larger as the problem scale, namely the dimension of variable, increases. Meanwhile, DFWLayer also has a competitive accuracy and results can be found in Appendix A.3.1.

Table 1: Comparison of running time (s) between different-scale optimization problems. The average and standard deviation are obtained over 5 trials. Lower values are better for the running time.

	Small	Medium	Large
Dimension of variable	500	1000	2000
CvxpyLayer	7.68 ± 0.07	59.02 ± 0.16	481.78 ± 2.54
Alt-Diff	1.05 ± 0.14	3.84 ± 0.12	22.51 ± 0.13
DFWLayer	$\mathbf{0.18 \pm 0.01}$	$\mathbf{0.21 \pm 0.01}$	$\mathbf{0.31 \pm 0.02}$

4 CONCLUSIONS

In this paper, we have proposed DFWLayer for solving convex optimization problems with L-smooth functions and norm constraints in an efficient way. Naturally derived from the Frank-Wolfe, DFWLayer accelerates to obtain solutions and gradients based on first-order optimization methods which avoid projections and Hessian matrix computations. Especially for ℓ_1-norm constraints, DFWLayer modifies non-differentiable operators with probabilistic approximation so that gradients can be efficiently computed through the unrolling sequence with automatic differentiation. Also, an annealing temperature is designed to guarantee the quality of both solutions and gradients. In future work, we plan to address a significant limitation of the DFWLayer: its current restriction to handling only norm constraints. We aim to explore and adapt the DFWLayer for a broader range of assumptions, thereby expanding its applicability in more diverse computational scenarios.

---

[1]As for others, for instance, trace norm can be considered with its dual norm: operator norm.

[2]$\hat{x}_k = x_k$ and $\hat{s}_k = s_k$ for $p > 1$ without probability approximation.

REFERENCES

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pp. 136–145. PMLR, 2017.

Antoine Bambade, Fabian Schramm, Adrien Taylor, and Justin Carpentier. Qplayer: efficient differentiation of convex quadratic optimization. *Optimization*, 2023.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Gábor Braun, Alejandro Carderera, Cyrille W Combettes, Hamed Hassani, Amin Karbasi, Aryan Mokhtari, and Sebastian Pokutta. Conditional gradient methods. *arXiv preprint arXiv:2211.14103*, 2022.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Guillaume Dalle, Léo Baty, Louis Bouvier, and Axel Parmentier. Learning with combinatorial optimization layers: a probabilistic approach. *arXiv preprint arXiv:2207.13513*, 2022.

Priya L Donti, David Rolnick, and J Zico Kolter. Dc3: A learning method for optimization with hard constraints. In *International Conference on Learning Representations*, 2020.

Benjamin Ellenberger. Pybullet gymperium. `https://github.com/benelot/pybullet-gym`, 2018–2019.

Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Kazumi Kasaura, Shuwa Miura, Tadashi Kozuno, Ryo Yonetani, Kenta Hoshino, and Yohei Hosoe. Benchmarking actor-critic deep reinforcement learning algorithms for robotics control with action constraints. *IEEE Robotics and Automation Letters*, 2023.

Benoit Landry. *Differentiable and Bilevel Optimization for Control in Robotics*. Stanford University, 2021.

Haixiang Sun, Ye Shi, Jingya Wang, Hoang Duong Tuan, H Vincent Poor, and Dacheng Tao. Alternating differentiation for optimization layers. In *The Eleventh International Conference on Learning Representations*, 2022.

A APPENDIX

A.1 ALGORITHM DETAILS

The DFWLayer for ℓ_1-norm constraints is presented in Algorithm 1.

Algorithm 1 Differentiable Frank-Wolfe Layer

Input: Parameters θ from the previous layers.
Parameter: Parameters w and t.
Output: The optimal solutions to the problem: $\min_x f(x; \theta)$ s.t. $\|w \circ x\|_1 \leq t$.

1: **Forward:**
2: Let $x_0 \in \mathcal{C}$ and $\hat{x}_0 = x_0$
3: **for** $k = 0, ..., K$ **do**
4:     Compute $\hat{s}_k$ using Equation (3).
5:     Obtain $\gamma_k = \min\{\frac{\langle \nabla_x f(\hat{x}_k), \hat{x}_k - \hat{s}_k \rangle}{L\|\hat{x}_k - \hat{s}_k\|^2}, 1\}$.
6:     Update $\hat{x}_{k+1}$ by $\hat{x}_{k+1} = (1 - \gamma_k)\hat{x}_k + \gamma_k \hat{s}_k$.
7: **end for**

8: **Backward:**
9: Obtain $\frac{\partial \hat{x}^*}{\partial \theta}$ with automatic differentiation.

A.2 THEORETICAL RESULTS

In this section, we provides convergence analysis for DFWLayer dealing with ℓ_1-norm constraints. The exploration begins on a modest assumption, keeping the analysis approachable and generalizable.

Inspired by (Dalle et al., 2022) turning combinatorial optimization layers into probabilistic layers, searching vertex of the feasible region can be regarded as the expectation from $\mathcal{S} = \{s^1, ..., s^{2n}\}$, which denotes the vertices of $\mathcal{C}$, over a probabilistic distribution $p(\cdot \mid \nabla_x f(x_k))$,

$$s_k = \mathbb{E}_{s \sim p(\cdot \mid \nabla_x f(x_k))}[S] = \sum_{s \in \mathcal{S}} s \cdot p(s \mid \nabla_x f(x_k)). \tag{2}$$

Intuitively, the probabilistic distribution can be chosen as the Dirac mass $p(\cdot \mid \nabla_x f(x_k)) = \delta_{s^*}(s)$ with $s^* = \arg\min_{s \in \mathcal{S}} \langle \nabla_x f(x_k), s \rangle$, though sharing the lacks of differentiability with $\arg\max$. Thus, we aim to find a differentiable $\hat{p}(\cdot \mid \nabla_x f(x_k))$ so that the Jacobian matrix of s_k is tractable, and the Boltzmann distribution is an appropriate candidate, leading to $\hat{p}(e_i \mid \nabla_x f(x_k)) \propto e^{\nabla_x f(x_k)^T e_i} = e^{\nabla_{x,i} f(x_k)}$. Thus, we can derive Equation (2) as

$$\begin{aligned}
\hat{s}_k &= \mathbb{E}_{s \sim \hat{p}_k(\tau)}[s] \\
&= -\frac{t}{w} \circ \text{sign}(g_{tw}(\hat{x}_k)) \circ \sum_i e_i \cdot \frac{e^{r_i(\hat{x}_k)/\tau}}{\sum_i e^{r_i(\hat{x}_k)/\tau}} \\
&= -\frac{t}{w} \circ \text{sign}(g_{tw}(\hat{x}_k)) \circ \text{softmax}(r(\hat{x}_k)/\tau),
\end{aligned} \tag{3}$$

Assumption A.1. *The Maximum Mean Discrepancy (MMD) between the approximating distribution $\hat{p}$ and the target distribution p is upper bounded by*

$$\begin{aligned}
&\text{MMD}(\Psi, \hat{p}_k(\tau), p_k) \\
&= \sup_{\psi \in \Psi} \mathbb{E}_{\hat{p}_k(\tau)}[\psi(s)] - \mathbb{E}_{p_k}[\psi(s)] \leq \delta(\tau), \text{ for all } k,
\end{aligned} \tag{4}$$

where $\delta(\tau)$ is a positive constant related to temperature τ.

Considering the temperature τ of the softmax function, Assumption A.1 is reasonably modest because the softmax function can always approach the hardmax function with a small temperature and thus $\delta(\tau) \to 0$ when $\tau \to 0$.

Implication A.2. *Under Assumption A.1, for all p_k, there exists a temperature τ_k for $\hat{p}_k(\tau_k)$ such that $\delta(\tau_k) \le O(\frac{1}{k})$.*

The relationship between the hardmax and the softmax function makes Implication A.2 directly follow Assumption A.1. With the implication, we can further understand Theorem 2.1.

The theorem has shown that the suboptimality is bounded by the approximation gap and the origin gap. According to Implication A.2, if $\delta(\tau_k) \le O(\frac{1}{k})$ through a certain τ_k, we can say DFWLayer converges at a sublinear rate. In section 2, we discuss an annealing temperature $\tau_k = 2^{-k//T}$ which decreases per T steps, and validate its improvement compared to constant temperatures.

As for the proof of Theorem 2.1, we can first split it into two terms,

$$
\begin{aligned}
h(\hat{x}_k) &= f(\hat{x}_k) - f(x^*) \\
&= \underbrace{f(\hat{x}_k) - f(x_k)}_{\text{approximation gap}} + \underbrace{f(x_k) - f(x^*)}_{\text{origin gap}}.
\end{aligned}
\tag{5}
$$

The original gap of the Frank-Wolfe with the short path rule and its proof are provided by Braun et al. (2022), we restate it as the following lemma.

Lemma A.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a L-smooth convex function on a convex region $\mathcal{C}$ with diameter M. The Frank-Wolfe with short path rule converges as follows:*

$$
h(x_k) = f(x_k) - f(x^*) \le \frac{2LM^2}{k+3} \le \frac{2LM^2}{k+2}.
\tag{6}
$$

Then, we need to present a useful remark, which is used for the proof of Lemma A.3 in (Braun et al., 2022) and the following proof of Theorem 2.1.

Remark A.4. In essence, the proof of Theorem 2.1 uses the modified agnostic step size $\gamma_k = \frac{2}{k+3}$, instead of the standard agnostic step size $\gamma_k = \frac{2}{k+2}$, because the convergence rate for the short path rule dominates the modified agnostic step size.

With Lemma A.3 and Remark A.4, we now start to prove Theorem 2.1.

Proof of Theorem 2.1. The original solutions x_k can be recursively expressed as the combination of initial x_0 and vertex s_k at each iteration,

$$
\begin{aligned}
x_k &= (1-\gamma)x_{k-1} + \gamma s_{k-1} \\
&= (1-\gamma)((1-\gamma)x_{k-2} + \gamma s_{k-2}) + \gamma s_{k-1} \\
&= (1-\gamma)^k x_0 + \gamma \sum_{o}^{k-1} (1-\gamma)^{k-1-i} s_i.
\end{aligned}
\tag{7}
$$

Similar with the original solutions, the solutions obtained by DFWLayer can also be derived as follows:

$$
\hat{x}_k = (1-\gamma)^k x_0 + \gamma \sum_{o}^{k-1} (1-\gamma)^{k-1-i} \hat{s}_i.
\tag{8}
$$

And thus we subtract Equation (7) from Equation (8) and obtain

$$
\hat{x}_k - x_k = \gamma \sum_{o}^{k-1} (1-\gamma)^{k-1-i} (\hat{s}_i - s_i).
\tag{9}
$$

Considering the smoothness of f and Equation (9), the approximation can be derived as

$$f(\hat{x}_k) - f(x_k) \leq \langle \nabla_x f(x_k), \hat{x}_k - x_k \rangle + \frac{L}{2} \|\hat{x}_k - x_k\|^2$$

$$= \gamma \sum_0^{k-1} (1-\gamma)^{k-1-i} \nabla_x f(x_k)^T (\hat{s}_i - s_i) + \frac{L}{2} \|\gamma \sum_0^{k-1} (1-\gamma)^{k-1-i} (\hat{s}_i - s_i)\|^2$$

$$\leq \gamma \sum_0^{k-1} (1-\gamma)^{k-1-i} (\mathbb{E}_{s \sim \hat{p}_i}[\nabla_x f(x_k)^T s] - \mathbb{E}_{s \sim p_i}[\nabla_x f(x_k)^T s])$$

$$+ \frac{L}{2}\gamma^2 \sum_0^{k-1} (1-\gamma)^{2(k-1-i)} \|\hat{s}_k - s_k\|^2$$

$$\leq \gamma \sum_0^{k-1} (1-\gamma)^{k-1-i} \delta + \frac{L}{2}\gamma^2 \sum_0^{k-1} (1-\gamma)^{2(k-1-i)} M^2$$

$$= (1 - (1-\gamma)^k)\delta + \frac{LM^2}{2} \frac{\gamma}{2-\gamma}(1 - (1-\gamma)^{2k})$$

$$\leq \delta + \frac{LM^2}{2k+4}. \tag{10}$$

Here, we use the triangle inequality in the first inequality. Then, we bound the difference of expectations by δ under Assumption A.1 and the distance between vertices by region diameter M in the second inequality. The last equality is obtained by choosing $\gamma_k = \frac{2}{k+3}$ via heuristic in Remark A.4.

Therefore, substituting the approximation gap from Lemma A.3 and the origin gap in Equation (5) yields Equation (1) as claimed. $\square$

A.3 EXPERIMENTAL DETAILS

All the experiments were implemented on an Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50GHz with 40 GB of memory and a Tesla T4 GPU.

A.3.1 DIFFERENT-SCALE OPTIMIZATION PROBLEMS

This section details the experiments in implementing our method to tackle quadratic programs of varying scales. We clarify the setup and discuss how our method performs in terms of both efficiency and accuracy, as well as the softmax temperature.

Efficiency. With objective $f(x; q) = \frac{1}{2}x^T P x + q^T x$ which chooses q as the parameter requiring gradients and constraints $\|w \circ x\| \leq t$, all the parameters P, q, w, t are generated randomly with symmetric matrix $P \succeq 0$ and $t \geq 0$. The tolerance $\epsilon = \frac{|f(x_{k+1}) - f(x_k)|}{|f(x_k)|}$ is used to terminate DFWLayer instead of a maximum iteration number. For all the methods, the tolerance is set as $\epsilon = 1e - 4$. We executed all the experiments 5 times and reported the average running time with standard deviation in Table 1. It is shown that DFWLayer performs much more efficiently than other two baselines and the speed gap becomes larger as the problem scale increases.

Table 2: Comparison of accuracy w.r.t. solutions and gradients for medium-scale problems. The average and standard deviation are obtained over 5 trials. Higher values are better for the gradients similarity, while lower values are better for others. Constraints violation is shown in red. Gradients similarity and solutions distance are computed with those obtained by CvxpyLayer as the reference.

	Gradients Sim.	Solutions Dist.	Mean Violation	Max Violation
CvxpyLayer	1.000 ± 0.000	0.000 ± 0.000	0.000	0.000
Alt-Diff	$\mathbf{0.980 \pm 0.040}$	$\mathbf{0.001 \pm 0.002}$	0.058	1.000
DFWLayer	$\mathbf{0.980 \pm 0.021}$	0.002 ± 0.001	0.000	0.000

Table 3: Comparison of accuracy w.r.t. solutions and gradients for small-scale problems. The average and standard deviation are obtained over 5 trials. Higher values are better for the gradients similarity, while lower values are better for others. Constraints violation is shown in red. Gradients similarity and solutions distance are computed with those obtained by CvxpyLayer as the reference.

	Gradients Sim.	Solutions Dist.	Mean Violation	Max Violation
CvxpyLayer	1.000 ± 0.000	0.000 ± 0.000	0.000	0.000
Alt-Diff	0.975 ± 0.049	$\mathbf{0.002 \pm 0.003}$	0.063	1.000
DFWLayer	$\mathbf{0.977 \pm 0.025}$	0.002 ± 0.001	0.000	0.000

Table 4: Comparison of accuracy w.r.t. solutions and gradients for large-scale problems. The average and standard deviation are obtained over 5 trials. Higher values are better for the gradients similarity, while lower values are better for others. Constraints violation is shown in red. Gradients similarity and solutions distance are computed with those obtained by CvxpyLayer as the reference.

	Gradients Sim.	Solutions Dist.	Mean Violation	Max Violation
CvxpyLayer	1.000 ± 0.000	0.000 ± 0.000	0.000	0.000
Alt-Diff	$\mathbf{0.978 \pm 0.044}$	$\mathbf{0.001 \pm 0.001}$	0.061	1.000
DFWLayer	0.978 ± 0.023	$\mathbf{0.001 \pm 0.001}$	0.000	0.000

Accuracy. As for accuracy, we report the cosine similarity of gradients and the Euclidean distance of solutions with CvxpyLayer as the reference for its stable performance. Also, constraints violation is a significant index for constrained optimization expecially when these constraints stand for safety. Thus, we show the results for medium problem scale in Table 2. It should be noted that we state the maximum violation instead of standard deviation for its widely use in practice. It can be seen from Table 2 that the solutions provided by Alt-Diff violate the constraints, although it has the higher accuracy of solutions than DFWLayer. The constraints violation of Alt-Diff results from insufficient iterations and a lack of projection to ensure the feasibility for its solutions, while our method enforces the feasibility without projection and thus solves the optimization problems in much less time with an acceptable accuracy margin.

Softmax Temperature. The softmax temperature is a crucial hyperparameter for DFWLayer, because it can assist to approximate the target distribution, which is required by Assumption A.1. The annealing schedule $\tau_k = 2^{-k//30}$ is used for the proposed method, so that the distance between the target and the approximating distributions can get closer during the iteration. In order to give a clear explanation of this choice, we plot the curve of cosine similarity and Euclidean distance compared with different constant temperatures.

(a) Solutions distance.
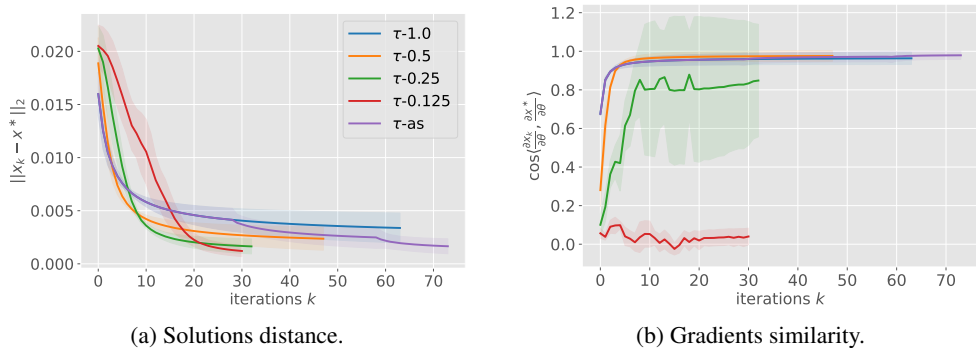
(b) Gradients similarity.

Figure 1: Gradients and solutions distance between CvxpyLayer and DFWLayer with different temperatures for medium-scale problems. All the curves terminate for tolerance $\epsilon = 1e - 4$. The shaded area for all the curves stands for standard deviation over 5 trials and x^* stands for solutions obtained by CvxpyLayer.

It can be seen from Figure 1a that with the decrease of temperature the accuracy of solutions obtained by DFWLayer becomes higher, as shown in Theorem 2.1. However, the automatic differentiation

(a) MSE loss.   (b) Mean violation.   (c) Violation rate.

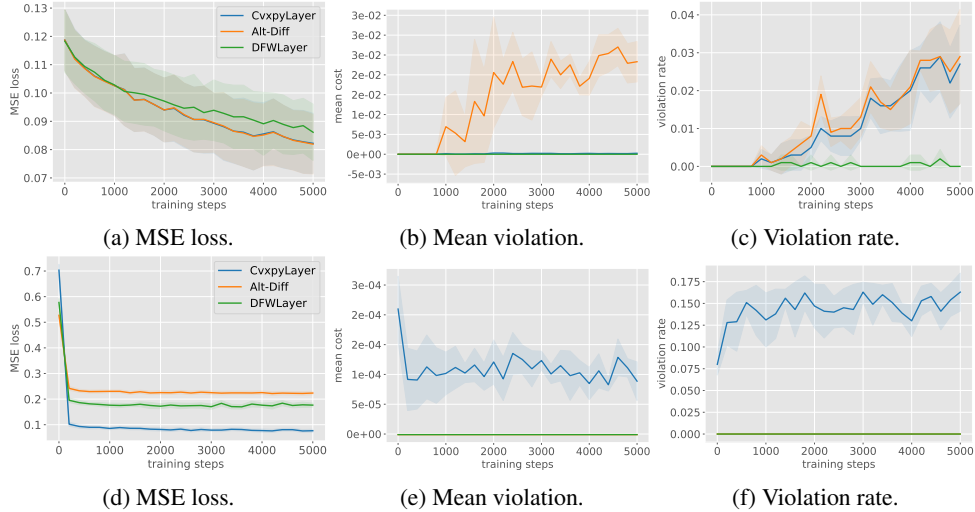(d) MSE loss.   (e) Mean violation.   (f) Violation rate.

Figure 2: MSE loss, mean violation and violation rate for robotics tasks. The first row is for R+O03 and the second row is for HC+O. Mean violation is computed over violated samples and violation rate is the ratio of violated samples to all samples in the testing set. The shaded area for all the curves stands for standard deviation over 5 trials. Lower values are better for all the metrics.

involving a small temperature can be unstable after a large number of iterations. As presented in Figure 1b, DFWLayer fails to compute accurate gradients when the temperature decreases to $\tau = 0.25$ and $\tau = 0.125$. Therefore, we choose to use an annealing temperature so that our method can obtain high-quality solutions and gradients at the same time, which is validated by the curve "τ-as" (the purple lines) in Figure 1.

A.3.2 Robotics Tasks Under Imitation Learning

As for the field of robotics, in this subsection, we discuss the application of our method to specific tasks under an imitation learning framework. HC+O and R+O03 are constrained HalfCheetah and Reacher environments chosen from action-constrained-RL-benchmark [3] in Kasaura et al. (2023). The original enviroments are from OpenAI Gym (Brockman et al., 2016) and PyBullet-Gym (Ellenberger, 2018–2019) and the constraints are presented as follows:

$$\sum_{i}^{d} |w_i a_i| \leq p_{\max}, \tag{11}$$

where w_i and a_i are the angular velocity and the torques corresponding to d joints, respectively, and $p_{\max}$ stands for the power constraint. Specifically, $d = 6$ and $p_{\max} = 20.0$ are for HC+O, while $d = 2$ and $p_{\max} = 0.3$ are for R+O03.

Under an imitation learning framework, optimization layers are added to the neural networks as the last layer, which aims to imitate expert policy and satisfy the power constraints Equation (11). We first collected 1 million and 0.3 million expert demonstrations by running DPro, a variant of TD3 in Kasaura et al. (2023), for HC+O and R+O03 respectively. Then, imitation learning was implemented using 80% data as the training set with the MSE loss function. During the training phase, we tested the loss, mean violation and violation rate every 200 steps using the rest 20% data as the testing set.

The architecture for previous layers is $[400, 300]$ and the activation function is ReLU function. The parameters are updated by Adams optimizer with $1e - 4$ as the learning rate. As for batch size, we have some experiments from $\{8, 16, 32, 64, 128\}$, and choose to use 64 considering the performance and efficiency comprehensively.

---

[3]The benchmark can be accessed through `http://github.com/omron-sinicx/actionconstrained-RL-benchmark`

As is shown in Figure 2, DFWLayer has significantly lower mean violation and violation rate than the other two baselines with comparable MSE loss. For R+O03 whose maximum power $p_{max} = 0.3$, the magnitude of the mean violation (about 3e-2) for Alt-Diff would have negative influence on the robotics tasks. However, DFWLayer and CvxpyLayer (with minor violation) are able to give feasible solutions to accomplish the tasks with accuracy and safety. For HC+O whose maximum power $p_{max} = 20.0$, CvxpyLayer achieves the best MSE loss with minor violation and a relatively high violation rate, while Alt-Diff satisfies the constraints all the time with the worst MSE loss. By contrast, DFWLayer outperforms the two methods comprehesively.