Regularized Frank-Wolfe for Dense CRFs: Generalizing Mean Field and Beyond

Đ.Khuê Lê-Huu **Karteek Alahari**
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK
38000 Grenoble, France
{khue.le,karteek.alahari}@inria.fr

Abstract

We introduce *regularized Frank-Wolfe*, a general and effective algorithm for inference and learning of dense conditional random fields (CRFs). The algorithm optimizes a nonconvex continuous relaxation of the CRF inference problem using vanilla Frank-Wolfe with approximate updates, which are equivalent to minimizing a regularized energy function. Our proposed method is a generalization of existing algorithms such as mean field or concave-convex procedure. This perspective not only offers a unified analysis of these algorithms, but also allows an easy way of exploring different variants that potentially yield better performance. We illustrate this in our empirical results on standard semantic segmentation datasets, where several instantiations of our regularized Frank-Wolfe outperform mean field inference, both as a standalone component and as an end-to-end trainable layer in a neural network. We also show that dense CRFs, coupled with our new algorithms, produce significant improvements over strong CNN baselines.

1 Introduction

Fully-connected or *dense* conditional random fields (CRFs) [34]—combined with strong pixel-level classifiers such as a convolutional neural network (CNN) [25, 42]—have been a highly-successful paradigm for semantic segmentation. Top-performing systems on the PASCAL VOC benchmark [22] used to include a CRF as either a post-processing step [13, 14, 15, 20, 43, 44, 45] or a trainable component [4, 49, 64, 68, 73, 76]. However, as CNNs got stronger, the improvements that CRFs brought decreased over time, and as a result they fell out of favor since 2017 [45].

In this paper, we revisit dense CRFs with two contributions. First, on the theoretical side, we propose *regularized Frank-Wolfe*, a new class of algorithms for inference and learning of CRFs that perform better than the popular *mean field* [34, 35, 58]—the method of choice in the aforementioned works. Regularized Frank-Wolfe optimizes a nonconvex continuous relaxation of the CRF inference problem (§2) by performing approximate conditional-gradient updates (§3.1), which is equivalent to minimizing a regularized energy using the generalized Frank-Wolfe method [53] (§3.2). Several of its instantiations lead to new algorithms that have not been studied before in the MAP inference literature (§3.3). Moreover, we show that it also includes several existing methods, including mean field and the concave-convex procedure [75], as special cases (§3.4). This generalized perspective allows a unified analysis of all these old and new algorithms in a single framework (§4). In particular, we show that they achieve a sublinear rate of convergence $\mathcal{O}(1/\sqrt{k})$ for suitable stepsize schemes, and in certain cases (such as strongly-convex regularizer or concave energy) this can be improved to $\mathcal{O}(1/k)$ (§4.1). Furthermore, we provide a tightness analysis for the resulting nonconvex relaxation of the regularized energy, recovering some existing tightness results [10, 41, 61] as special cases (§4.2). The proposed algorithms are easy to implement, converge quickly in practice, and have (sub)differentiable iterates. Such properties are important for successful *gradient-based learning* via backpropagation [47, 62].

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

Our second contribution lies on the practical side. In addition to mean field and regularized Frank-Wolfe variants, we re-implement several existing first-order inference methods [39, 41, 48]—those that are amenable to gradient-based learning—for comparison. Remarkably, we find that dense CRFs can still achieve important improvements over the strong DeepLabv3+ [17] CNN model for all these solvers (§5). In particular, our best variant of regularized Frank-Wolfe achieves a mean intersection-over-union (mIoU) score of 88.0 on the PASCAL VOC test set (§5.3), improving over DeepLabv3+. We hope that these encouraging results could attract interest from the community in considering dense CRFs (again) for tasks such as semantic segmentation. Our source code is made publicly available under the GNU general public license for this purpose.[1]

2 Background

2.1 Inference in CRFs

Let $\mathbf{s} \in \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ denote an assignment to n discrete random variables $S_1, \dots, S_n$, where each variable S_i takes values in a finite set of states (or *labels*) $\mathcal{S}_i$. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph of n nodes ($\mathcal{V} = \{1, 2, \dots, n\}$). A Markov random field (MRF) defined by $\mathcal{G}$ encodes a family of joint distributions that can be factorized as follows, where $\phi_i : \mathcal{S}_i \to \mathbb{R}_+$ and $\phi_{ij} : \mathcal{S}_i \times \mathcal{S}_j \to \mathbb{R}_+$ are the so-called *unary* and *pairwise* (respectively) potential functions, and Z is a normalization factor:

$$p(\mathbf{s}) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \phi_i(s_i) \prod_{ij \in \mathcal{E}} \phi_{ij}(s_i, s_j). \tag{1}$$

Note that (1) can also include conditional distributions, i.e., $p(\mathbf{s} \mid \mathbf{o})$ with observed variables $\mathbf{o}$. In this case the potentials may also depend on $\mathbf{o}$, e.g., $\phi_i(s_i; \mathbf{o})$, and this model is referred to as conditional random field (CRF) [37]. We will present later (§5.1) such a model for image segmentation. In the following, we use MRF and CRF interchangeably. We assume further that all nodes have the same set of labels: $\mathcal{S}_i = \mathcal{S}\ \forall i$, with cardinality $d = |\mathcal{S}|$. It is convenient to express $p(\mathbf{s})$ as $\frac{1}{Z}\exp(-e(\mathbf{s}))$, where the so-called *energy* $e(\mathbf{s})$ is defined as

$$e(\mathbf{s}) = \sum_{i \in \mathcal{V}} \theta_i(s_i) + \sum_{ij \in \mathcal{E}} \theta_{ij}(s_i, s_j), \quad \text{with } \theta_i(s_i) = -\log \phi_i(s_i) \text{ (idem for } \theta_{ij}). \tag{2}$$

The task of maximum a posteriori (MAP) inference consists in finding the most probable joint assignment, also known as *energy minimization* (which is NP-Hard in general [65]):

$$\mathbf{s}^* = \operatorname*{argmax}_{\mathbf{s} \in \mathcal{S}^n} p(\mathbf{s}) = \operatorname*{argmin}_{\mathbf{s} \in \mathcal{S}^n} e(\mathbf{s}). \tag{3}$$

2.2 Continuous relaxation of MAP inference

Let x_{is} be a binary variable such that $x_{is} = 1$ iff label s is assigned to node i. Then, $\mathbf{x}_i = (x_{is})_{s \in \mathcal{S}} \in \{0, 1\}^d$ denotes the one-hot vector for node i. Let $\mathbf{x} \in \{0, 1\}^{nd}$ be the concatenation of all $\mathbf{x}_i$, and let $\boldsymbol{\theta}_i = (\theta_i(s))_{s \in \mathcal{S}} \in \mathbb{R}^d$, $\boldsymbol{\Theta}_{ij} = (\theta_{ij}(s,t))_{t \in \mathcal{S}}^{s \in \mathcal{S}} \in \mathbb{R}^{d \times d}$. The energy (2) is then

$$E(\mathbf{x}; \boldsymbol{\theta}) = \sum_{i \in \mathcal{V}} \boldsymbol{\theta}_i^\top \mathbf{x}_i + \sum_{ij \in \mathcal{E}} \mathbf{x}_i^\top \boldsymbol{\Theta}_{ij} \mathbf{x}_j, \tag{4}$$

where $\boldsymbol{\theta}$ is a parameter vector composed of all $\boldsymbol{\theta}_i$ and $\boldsymbol{\Theta}_{ij}$. The MAP inference problem (3) transforms to minimizing $E(\mathbf{x}; \boldsymbol{\theta})$ over the new variables $\mathbf{x}$. A natural approach is to relax the binary constraint and solve the continuous relaxation $\min_{\mathbf{x} \in \mathcal{X}} E(\mathbf{x}; \boldsymbol{\theta})$ with

$$\mathcal{X} = \left\{ \mathbf{x} \in \mathbb{R}^{nd} : \mathbf{x} \geq \mathbf{0}, \mathbf{1}^\top \mathbf{x}_i = 1\ \forall i \in \mathcal{V} \right\}. \tag{5}$$

This continuous relaxation is known to be *tight* [10, 41, 61]. For convenience, we represent the unary potentials as a vector $\mathbf{u}(\boldsymbol{\theta}) = (\boldsymbol{\theta}_i)_{i \in \mathcal{V}} \in \mathbb{R}^{nd}$, and the pairwise potentials as a symmetric $n \times n$ block matrix $\mathbf{P}(\boldsymbol{\theta}) \in \mathbb{R}^{nd \times nd}$, where the (i, j) block is $\boldsymbol{\Theta}_{ij}$. The problem is reduced to

$$\min_{\mathbf{x} \in \mathcal{X}} E(\mathbf{x}; \boldsymbol{\theta}) \triangleq \frac{1}{2} \mathbf{x}^\top \mathbf{P}(\boldsymbol{\theta}) \mathbf{x} + \mathbf{u}(\boldsymbol{\theta})^\top \mathbf{x}. \tag{6}$$

[1] https://github.com/netw0rkf10w/CRF

The reason we have made $\boldsymbol{\theta}$ explicit in (4) and (6) is to provide more clarity when discussing the differentiability of their solutions for the *learning* task. Note that an optimal solution $\mathbf{x}^*$ to (4) and (6) is a function of $\boldsymbol{\theta}$, and thus should be written as $\mathbf{x}^*(\boldsymbol{\theta})$. Therefore, when we say a solution is differentiable, it is understood that it is so with respect to $\boldsymbol{\theta}$. When there is no ambiguity, we omit $\boldsymbol{\theta}$ and write simply $E(\mathbf{x})$, $\mathbf{P}$, and $\mathbf{u}$. We will be interested in problem (6) in the rest of the paper, though it should be noted that our method also applies to the so-called linear programming (LP) relaxation, which takes the form $\min_{\mathbf{x}\in\mathcal{X}_{\text{LP}}} E_{\text{LP}}(\mathbf{x};\boldsymbol{\theta})$, where $E_{\text{LP}}(\mathbf{x};\boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{x}$ and $\mathcal{X}_{\text{LP}}$ is the so-called *local polytope* [72]. We refer to Appendix A for the details.

2.3 Vanilla Frank-Wolfe algorithm for MAP inference

Since the continuous energy is differentiable, it is natural to apply first-order methods such as Frank-Wolfe [23] to solving (6) [41]. Starting from a feasible $\mathbf{x}^0 \in \mathcal{X}$, Frank-Wolfe approximately solves (6) by iterating the following steps, where $\alpha_k \in [0, 1]$ follows some stepsize scheme:

$$\mathbf{p}^k \in \operatorname*{argmin}_{\mathbf{p}\in\mathcal{X}} \left\langle \nabla E(\mathbf{x}^k), \mathbf{p} \right\rangle, \qquad \mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k(\mathbf{p}^k - \mathbf{x}^k). \tag{7}$$

The same idea has also been successfully applied to other types of continuous relaxations of (6), such as linear programming (LP) [52] or convex quadratic programming (QP) [21, 61].

3 Regularized Frank-Wolfe for inference

In this section, we introduce the proposed regularized Frank-Wolfe as a general class of algorithms for MAP inference. The motivation and general framework are presented in §3.1 and §3.2. We show that several new inference algorithms can be obtained using different regularizers (§3.3). Moreover, regularized Frank-Wolfe also includes a number of existing algorithms as special cases (§3.4).

3.1 A smoothing perspective

We have seen in §2.3 the (vanilla) Frank-Wolfe method for solving MAP inference. Unfortunately, from a *learning* perspective, this algorithm is problematic. Indeed, CRF learning with stochastic gradient descent (SGD) is typically done by backpropagating through the optimization steps [35, 63, 76], but the iterate $\mathbf{p}^k$ in (7) is piecewise constant and thus its gradient (w.r.t. $\boldsymbol{\theta}$) is zero almost everywhere (§C.1), which makes learning not possible.[2] This issue will be illustrated later in the experiments. A potential solution is to add a (typically strongly-)convex regularization term:

$$\mathbf{p}^k \in \operatorname*{argmin}_{\mathbf{p}\in\mathcal{X}} \left\{ \left\langle \nabla E(\mathbf{x}^k), \mathbf{p} \right\rangle + r(\mathbf{p}) \right\}, \qquad \mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k(\mathbf{p}^k - \mathbf{x}^k). \tag{8}$$

With appropriately chosen regularizers, (8) becomes suitable for gradient-based learning.

The technique of approximating an update step by a regularized one is quite standard in the optimization literature. For example, the classical proximal gradient method [48] can be interpreted the same way. Our update (8) is inspired by Nesterov's smoothing [56], and thus is similar in spirit to its many applications [31, 57, 66]. In the next section, we present a different, more general perspective to view (8), which offers more flexibility in designing new algorithms, in making connections to existing ones, and in analyzing their theoretical properties in a unified manner.

3.2 A regularized energy perspective

Instead of approximating the local updates of a *given algorithm* (in this case: vanilla Frank-Wolfe) to minimize the *same* objective function, one may choose to keep the algorithm, and approximate instead the objective. At first glance, however, this idea does not seem to be a good one. Indeed, if we replace E by some function E' and apply vanilla Frank-Wolfe, the updates (7) remains piecewise constant, thus we have the same zero-gradient issue. It turns out that, if we choose a more appropriate "given algorithm", this idea can work. Such a choice is the *generalized Frank-Wolfe* algorithm [5, 12, 53].

[2] Note that backpropagation typically requires the operations to be differentiable (at least) almost everywhere, which $\mathbf{p}^k$ satisfies. Therefore, the issue here does not lie in *differentiability*, but in the resulting *zero* gradients. Blackbox differentiation [60] can deal with this scenario, but it is limited to LP relaxations.

Consider the following problem, where $f : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is differentiable but possibly nonconvex, and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is proper, closed, and convex but possibly non-differentiable:

$$\min_{\mathbf{x}} F(\mathbf{x}) \triangleq f(\mathbf{x}) + g(\mathbf{x}). \tag{9}$$

Generalized Frank-Wolfe solves (9) by iterating

$$\mathbf{p}^k \in \operatorname*{argmin}_{\mathbf{p}} \left\{ \left\langle \nabla f(\mathbf{x}^k), \mathbf{p} \right\rangle + g(\mathbf{p}) \right\}, \qquad \mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k(\mathbf{p}^k - \mathbf{x}^k). \tag{10}$$

If g is the indicator function $\delta_{\mathcal{X}}$ of $\mathcal{X}$ (i.e., $\delta_{\mathcal{X}}(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathcal{X}$ and $\delta_{\mathcal{X}}(\mathbf{x}) = +\infty$ otherwise), then the algorithm clearly reduces to vanilla Frank-Wolfe (7) for $\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$. Therefore, generalized Frank-Wolfe applied to (6) with $f = E$ and $g = \delta_{\mathcal{X}}$ will yield exactly the same updates (7). Now let us apply this algorithm to an approximate objective $E_r(\mathbf{x}) = E(\mathbf{x}) + r(\mathbf{x})$ for some function r. Choosing $f = E$ and $g = r + \delta_{\mathcal{X}}$, it is straightforward that (10) reduces to (8). Therefore, we have recovered the same algorithm as in §3.1, but this time through different machinery.

This framework offers a great flexibility as one can choose f and g in many different ways to obtain new algorithms. The only conditions are f being differentiable and g being convex, so that the subproblem in (10) is well-defined and globally solvable.[3] For example, instead of choosing $f = E$ and $g = r + \delta_{\mathcal{X}}$ as above, one can choose $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{P}\mathbf{x}$ and $g(\mathbf{x}) = \mathbf{u}^\top \mathbf{x} + r(\mathbf{x}) + \delta_{\mathcal{X}}(\mathbf{x})$. We will recover later in §3.4 some existing algorithms (as special cases) through this kind of decomposition. Finally, we present Algorithm 1 for (approximately) solving MAP inference (6).

Algorithm 1 Generic regularized Frank-Wolfe for (approximately) solving MAP inference (6).

1: Choose a regularizer r such that there exist f (differentiable) and g (convex) satisfying $f + g = E + r + \delta_{\mathcal{X}}$. Typically (but not necessarily) r is convex on $\mathcal{X}$ and is constant on $\mathcal{X} \cap \{0, 1\}^{nd}$.
2: Initialization: $k \leftarrow 0$, $\mathbf{x}^0 \in \mathcal{X}$, number of iterations N.
3: Compute $\mathbf{p}^k \in \operatorname{argmin}_{\mathbf{p}} \left\{ \left\langle \nabla f(\mathbf{x}^k), \mathbf{p} \right\rangle + g(\mathbf{p}) \right\}$ and compute the stepsize α_k.
4: Update $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k(\mathbf{p}^k - \mathbf{x}^k)$. Let $k \leftarrow k + 1$ and go to Step 3 until $k = N$.
5: Rounding: convert $\mathbf{x}$ to a discrete solution and return.

While the choice of (r, f, g) can be highly flexible, it would make little sense to optimize a function that has nothing to do with the original objective (i.e., the discrete energy). Let $\overline{\mathcal{X}} = \mathcal{X} \cap \{0, 1\}^{nd}$ denote the discrete domain of our problem. If we choose r such that it is constant on $\overline{\mathcal{X}}$ (as suggested in Step 1 above), then minimizing E on $\overline{\mathcal{X}}$ is equivalent to minimizing $E + r$ on $\overline{\mathcal{X}}$, and thus Algorithm 1 actually solves the continuous relaxation of a (different) discrete problem that is equivalent to MAP inference. Further discussion on this matter, as well as on the rounding Step 5, are deferred until §4.2.

Finally, we should note that adding a strongly-convex regularizer is not new in the MAP inference literature [29, 31, 52, 67, 69]. In particular, some previous work even applied (vanilla) Frank-Wolfe to optimizing such regularized energy [52, 67, 69]. All these algorithms, however, suffer from the aforementioned zero-gradient issue, as already explained in the beginning of this section.

3.3 Particular instantiations

The previous section presents regularized Frank-Wolfe as a general algorithm for inference. We now discuss concrete examples of its instantiations. To the best of our knowledge, *all the algorithms presented in this section are new and have not been studied previously in the MAP inference literature*. In particular, despite some similarities with proximal gradient [48] and mirror descent [8, 55], our following euclidean and entropic variants are actually different from these methods.[4]

Euclidean Frank-Wolfe Perhaps the most natural choice is ℓ_2 regularization. In Algorithm 1, let us choose $f(\mathbf{x}) = E(\mathbf{x})$ and $r(\mathbf{x}) = \frac{\lambda}{2}\|\mathbf{x}\|_2^2$, where $\lambda > 0$ is a regularization weight. Let $\Pi_{\mathcal{X}}(\mathbf{v})$ be the projection of a vector $\mathbf{v}$ onto $\mathcal{X}$. It can be shown (§D.1) that Step 3 in Algorithm 1 becomes

$$\mathbf{p}^k = \operatorname*{argmin}_{\mathbf{p}\in\mathcal{X}} \left\{ \left\langle \mathbf{P}\mathbf{x}^k + \mathbf{u}, \mathbf{p} \right\rangle + \frac{\lambda}{2}\|\mathbf{p}\|_2^2 \right\} = \Pi_{\mathcal{X}}\left(-\frac{1}{\lambda}(\mathbf{P}\mathbf{x}^k + \mathbf{u}) \right) \quad \forall k \geq 0. \tag{11}$$

[3] Further mild conditions are required for convergence (§4.1).

[4] In proximal gradient and mirror descent, the current iterate is constrained to stay close to the previous one, while this is not the case in our method. See Appendix D for the details.

Entropic Frank-Wolfe In Algorithm 1, let us choose $f(\mathbf{x}) = E(\mathbf{x})$ and $r(\mathbf{x}) = -\lambda H(\mathbf{x})$, where $\lambda > 0$ is a regularization weight and $H(\mathbf{x}) = -\sum_{i\in\mathcal{V}}\sum_{s\in\mathcal{S}} x_{is}\log x_{is}$ is the entropy of $\mathbf{x}$ over $\mathcal{X}$. It can be shown (§D.2) that Step 3 in Algorithm 1 becomes

$$\mathbf{p}^k = \operatorname*{argmin}_{\mathbf{p}\in\mathcal{X}} \left\{\langle \mathbf{P}\mathbf{x}^k + \mathbf{u}, \mathbf{p}\rangle - \lambda H(\mathbf{p})\right\} = \operatorname{softmax}\left(-\frac{1}{\lambda}(\mathbf{P}\mathbf{x}^k + \mathbf{u})\right) \quad \forall k \geq 0, \tag{12}$$

where $\mathbf{v} = \operatorname{softmax}(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^{nd}$ means $\mathbf{v} \in \mathbb{R}^{nd}$ and $v_{is} = \frac{\exp(x_{is})}{\sum_{t\in\mathcal{S}}\exp(x_{it})}$ $\forall i \in \mathcal{V}, \forall s \in \mathcal{S}$. The resulting algorithm has a tight connection with (parallel) mean field [34, 35] (discussed in §3.4).

Other variants One can consider more sophisticated regularizers, e.g., a weighted combination of ℓ_2 norm and entropy. Other options include the many different regularizers that have been used in diverse machine learning applications, such as ℓ_p norm [57], lasso variants [57], or binary entropy [3]. Although these variants also lead to new MAP inference algorithms, their implementations are more sophisticated since their subproblems (10) require numerical solutions as no closed form ones exist.

3.4 Recovering existing algorithms as special cases

In addition to the above new algorithms, regularized Frank-Wolfe also includes several existing ones as special cases. We present some of them below and refer to Appendix A for further details.

Mean field This is a special case of the above Entropic Frank-Wolfe. Indeed, if we choose $\lambda = 1$ in (12) and a constant stepsize $\alpha_k = 1$ $\forall k \geq 0$ in Algorithm 1, then it is straightforward that this algorithm is reduced to the following update step, where $\mathcal{N}_i$ is the set of neighbors of node i:

$$\mathbf{x}^{k+1} = \operatorname{softmax}(-\mathbf{P}\mathbf{x}^k - \mathbf{u}) \iff x_{is}^{k+1} = \tfrac{1}{Z_i}\exp\left(-\theta_i(s) - \textstyle\sum_{j\in\mathcal{N}_i}\sum_{t\in\mathcal{S}}\theta_{ij}(s,t)x_{jt}^k\right) \forall i \in \mathcal{V}, s \in \mathcal{S}.$$

This is precisely a (parallel) mean field update [34, 35]. To conclude, parallel mean field is an instance of Entropic Frank-Wolfe with unit regularization weight and unit stepsize. Interestingly, the update (12) is the well-known softmax function with *temperature* in the deep learning literature [28]. One could have easily come up with such a simple extension of mean field by adding a temperature to softmax (yet surprisingly this has not been tried before), but here we have provided a principled way to achieve that. As shown later in the experiments, with suitable λ, this extension yields much better results than vanilla mean field. Finally, we should note that the tight connection between mean field and first-order methods has been noticed before. Krähenbühl and Koltun [35] proposed several mean-field-type variants based on the concave-convex procedure [75], while closely similar variants can also be obtained through proximal gradient [2, 6], but unlike our generalized algorithm, these algorithms cannot recover *exactly* the original mean field of Krähenbühl and Koltun [34].

Concave-convex procedure CCCP [75] solves (9), assuming f is concave and g is convex, by updating $\mathbf{x}^{k+1}$ as a solution to $-\nabla f(\mathbf{x}^k) \in \partial g(\mathbf{x}^{k+1})$,[5] which is precisely (10) with stepsize $\alpha_k = 1$. We conclude that CCCP is a special case of generalized Frank-Wolfe with f concave and unit stepsize. As a result, many existing CCCP-based inference algorithms [21, 35] can be seen as special cases of regularized Frank-Wolfe. For example, the ones presented by Desmaison et al. [21] are instantiations of the proposed algorithm with either $f(\mathbf{x}) = -\mathbf{x}^\top \operatorname{diag}(\mathbf{c})\mathbf{x}$ and $r(\mathbf{x}) = E(\mathbf{x}) + \mathbf{x}^\top \operatorname{diag}(\mathbf{c})\mathbf{x}$ (where $\mathbf{c} \in \mathbb{R}^{nd}$ is large enough so that $r(\mathbf{x})$ is convex), or $f(\mathbf{x}) = \mathbf{x}^\top(\mathbf{P} - \mathbf{C})\mathbf{x}$ and $r(\mathbf{x}) = \mathbf{u}^\top\mathbf{x} + \mathbf{x}^\top\mathbf{C}\mathbf{x}$ (where $\mathbf{C}$ is some matrix such that f is concave and r is convex). Note that in these instantiations, Step 3 in Algorithm 1 requires an iterative (numerical) solution. Finally, all the algorithms presented by Krähenbühl and Koltun [35] are also instantiations of the proposed method because they are based on CCCP. We refer to Appendix A for further details.

Vanilla Frank-Wolfe This is trivially a special case of regularized Frank-Wolfe and we briefly discuss it for completeness. Choosing $f(\mathbf{x}) = E(\mathbf{x})$ and $r(\mathbf{x}) = 0$ we obtain the algorithm by Lê-Huu and Paragios [41]. Likewise, the one by Desmaison et al. [21] corresponds to $f(\mathbf{x}) = E(\mathbf{x}) - \mathbf{c}^\top\mathbf{x} + \mathbf{x}^\top \operatorname{diag}(\mathbf{c})\mathbf{x}$ and $r(\mathbf{x}) = 0$, where $\mathbf{c} \in \mathbb{R}^{nd}$ is large enough for f to be convex. In addition, we can also recover existing LP-based algorithms by choosing $\mathcal{X} = \mathcal{X}_{\text{LP}}, r(\mathbf{x}) = 0$, and $f(\mathbf{x}) = E_{\text{LP}}(\mathbf{x}) + R(\mathbf{x})$ with suitable $R(\mathbf{x})$. Indeed, the one by Meshi et al. [52] takes $R(\mathbf{x})$ as the squared ℓ_2-norm of linear constraints, while the ones by Sontag and Jaakkola [67] and Tang et al. [69] correspond to $R(\mathbf{x})$ being an entropy approximation and its generalization, respectively (see §A).

[5] In the original CCCP [75], g is differentiable, thus the update becomes $-\nabla f(\mathbf{x}^k) = \nabla g(\mathbf{x}^{k+1})$.

4 Theoretical analysis

4.1 Convergence

We provide a convergence analysis for the generalized Frank-Wolfe algorithm, and the results for CRF inference special cases will then follow as a consequence. Convergence of vanilla Frank-Wolfe has been well studied in the literature [24, 30, 36, 38]. For generalized Frank-Wolfe, different analyses exist for the case where both f and g in (9) are convex [5, 26, 51, 74]. We are particularly interested in the general case where f is **nonconvex**,[6] as the CRF energy is often highly so in practice. Mine and Fukushima [53] (and subsequently Bredies et al. [12]) proved the global convergence of the algorithm under mild conditions, though no rate of convergence was given. Recently, Beck [7] obtained an $\mathcal{O}(1/\sqrt{k})$ rate of convergence for convex g under adaptive or line-search stepsizes. We extend their analysis with several contributions. First, we include the case where g is strongly convex, which is important as our main variants for inference (e.g., mean field or ℓ_2-Frank-Wolfe) use strongly-convex regularizers. Second, to also include CCCP [75] as a special case, we relax their Lipschitz smoothness assumption on f to semi-concavity (which is weaker, as any L-smooth function is also L-concave). Third, we also consider much weaker stepsize schemes such as *constant* or *non-summable* ones. We show that for either concave f or strongly-convex g, a better $\mathcal{O}(1/k)$ rate of convergence can be achieved, even under the (weak) constant stepsize. It should be noted that our results are new.

All the results in this section are stated under the following assumptions, where L_f and σ_g are non-negative constants and $\|\cdot\|$ denotes the ℓ_2 norm. Their proofs are given in Appendix B.

Assumption 1. *f is differentiable and L_f-semi-concave (i.e., $f(\mathbf{x}) - \frac{L_f}{2}\|\mathbf{x}\|^2$ is concave) on $\operatorname{dom} f$, which is assumed to be open and convex. When $L_f = 0$, f is concave.*

Assumption 2. *g is proper, closed, and σ_g-strongly-convex (i.e., $g(\mathbf{x}) - \frac{\sigma_g}{2}\|\mathbf{x}\|^2$ is convex), and $\operatorname{dom} g \subseteq \operatorname{dom} f$ is compact. When $\sigma_g > 0$, g is strongly convex.*

Let $\mathbf{p}_\mathbf{x}$ denote a solution of $\min_\mathbf{p} \{\langle \nabla f(\mathbf{x}), \mathbf{p}\rangle + g(\mathbf{p})\}$ and let $\mathbf{p}^k = \mathbf{p}_{\mathbf{x}^k}$. The following quantity, called the *conditional gradient norm* [7], will serve as an optimality measure:

$$S(\mathbf{x}) = \langle \nabla f(\mathbf{x}), \mathbf{x} - \mathbf{p}_\mathbf{x}\rangle + g(\mathbf{x}) - g(\mathbf{p}_\mathbf{x}). \tag{13}$$

Lemma 1. *$S(\mathbf{x}) \geq \frac{\sigma_g}{2}\|\mathbf{x} - \mathbf{p}_\mathbf{x}\|^2 \ \forall \mathbf{x} \in \operatorname{dom} f$, and $S(\mathbf{x}) = 0$ iff $\mathbf{x}$ is a stationary point of (9).*

The following theorem contains our convergence results for the most common stepsize schemes, including the following *adaptive* and *line-search* stepsizes, respectively:

$$\alpha_k = \min\left\{1, \frac{1}{L_f + \sigma_g}\left(\frac{S(\mathbf{x}^k)}{\|\mathbf{p}^k - \mathbf{x}^k\|^2} + \frac{\sigma_g}{2}\right)\right\}, \qquad \alpha_k = \operatorname*{argmin}_{\alpha \in [0,1]} F(\mathbf{x}^k + \alpha(\mathbf{p}^k - \mathbf{x}^k)). \tag{14}$$

Theorem 1. *Let F^* be the minimum value of F, Ω be the diameter of $\operatorname{dom} g$, $\Delta_k = F(\mathbf{x}^k) - F^*$, $\omega = \frac{\sigma_g}{L_f + \sigma_g}$, $\rho(\alpha) = \alpha \min\left\{1, 2 - \frac{\alpha}{\omega}\right\}$, $\eta(\alpha) = \frac{1}{2}\left[(L_f + \sigma_g)\alpha - \sigma_g\right]$, and $\mu = \sqrt{2L_f\Delta_0}$. For any $k \geq 0$, we have $\min_{0 \leq i \leq k} S(\mathbf{x}^i) \leq B_k$, where the bound B_k is given as follows:*

	constant stepsize $\alpha_k = \alpha > 0 \ \forall k$	constant step length $\alpha_k = \frac{\alpha}{\|\mathbf{p}^k - \mathbf{x}^k\|} \ \forall k$	non-summable $\sum_{k=0}^{+\infty} \alpha_k = \infty$	adaptive or line search (14)
convex g	$\frac{\Delta_0}{\alpha(k+1)} + \frac{L_f\Omega^2\alpha}{2}$	$\frac{\Delta_0\Omega}{\alpha(k+1)} + \frac{L_f\Omega\alpha}{2}$	$\frac{\Delta_0 + \frac{L_f\Omega^2}{2}\sum_{i=0}^k \alpha_i^2}{\sum_{i=0}^k \alpha_i}$	$\max\left(\frac{2\Delta_0}{k+1}, \frac{\mu\Omega}{\sqrt{k+1}}\right)$
strongly convex g	$\frac{\Delta_0}{\alpha(k+1)} + \eta(\alpha)\Omega^2 \ \forall \alpha \geq 2\omega$ $\frac{\Delta_0}{\rho(\alpha)(k+1)} \ \forall \alpha < 2\omega$	$\left(\frac{\Delta_0}{\alpha\sqrt{2\sigma_g}(k+1)} + \frac{(L_f + \sigma_g)\alpha}{2\sqrt{2\sigma_g}}\right)^2$	$\frac{\Delta_{k(\omega)}}{\sum_{i=k(\omega)}^k \alpha_i}$	$\frac{\Delta_0}{\omega(k+1)}$
concave f	$\frac{\Delta_0}{\alpha(k+1)}$	$\frac{\Delta_0\Omega}{\alpha(k+1)}$	$\frac{\Delta_0}{\sum_{i=0}^k \alpha_i}$	$\frac{2\Delta_0}{k+1}$

In the above, $k(\omega) = \min\{k : \alpha_i < 2\omega \ \forall i \geq k\}$, with further assumption that $\lim_{k\to\infty} \alpha_k = 0$ for (jointly) non-concave f and non-summable α_k. For the non-highlighted cases, we have $\lim_{k\to\infty} S(\mathbf{x}^k) = 0$ and any limit point of the sequence $(\mathbf{x}^k)_{k\geq 0}$ is a stationary point of (9).

The table in Theorem 1 also provides rates of convergence for the algorithm. Prior to our work, the $\mathcal{O}(1/\sqrt{k})$ rate for the adaptive or line-search stepsizes (top-right cell of the table, due to Beck [7])

[6] Note that g is still assumed to be convex, so that the subproblem (10) can be solved to global optimality.

was the best for *nonconvex* objectives.[7] We have improved this rate to $\mathcal{O}(1/k)$ when f is concave or g is strongly convex, even under weaker stepsize schemes. In particular, convergence is guaranteed for all considered stepsize schemes when f is concave, for which the best bound is obtained when $\alpha_k = 1\ \forall k$, which explains the default unit stepsize in CCCP [75] (see §3.4). Convergence is also guaranteed for the (diminishing) non-summable scheme (which includes common stepsizes such as $\alpha_k = 2/(k+2)$ or $\alpha_k = 1/\sqrt{k}$), but the rate depends on the rate of divergence of $\sum_{i=0}^{k} \alpha_i$. More detailed results and analyses can be found in Appendix B.

Convergence for MAP inference For all the instantiations of regularized Frank-Wolfe presented in §3.3 and §3.4, it is easy to check that Assumptions 1 and 2 are satisfied. In addition, the regularizers in most of them (euclidean or entropic variants, including mean field) are strongly convex, thus we would expect a rate of convergence of at least $\mathcal{O}(1/k)$ in practice for these algorithms under the adaptive, line search, or (suitable) constant stepsizes. Note that the adaptive scheme requires to know L_f and σ_g, which is possible in our case: a lower bound on σ_g is λ for both euclidean and entropic variants, while an upper bound on L_f is $\|\mathbf{P}\|_2$ for the energy (6). In practice, however, these bounds could be too loose to yield good convergence.

Convergent mean field It is well-known that parallel mean field may diverge [35]. Our Entropic Frank-Wolfe can be viewed as an improved variant of mean field that is globally convergent for different stepsize schemes, without resorting to a concave approximation as done by Krähenbühl and Koltun [35]. Our above analysis also provides an explanation for a known phenomenon [6]: *damped* mean field (corresponding to Entropic Frank-Wolfe with $\lambda = 1$ and $\alpha_k = \alpha < 1\ \forall k$) is more likely (than mean field) to guarantee convergence when the energy is not concave.

4.2 Tightness of the relaxation

We have seen that regularized Frank-Wolfe (Algorithm 1) minimizes a modified continuous energy. It is thus reasonable to ask whether doing so also minimizes the original discrete energy (which is the main objective). In this section, we partially answer this question by providing some tightness guarantee for this regularized continuous relaxation. Our analysis is quite general and also includes several existing tightness results [10, 41, 61] as special cases. All proofs can be found in Appendix C.2.

The last step in Algorithm 1 consists in converting $\mathbf{x}$ to a discrete solution. We consider two such rounding schemes. The simplest one is perhaps **nearest rounding**, which assigns each node i with the label $s_i \in \operatorname{argmax}_{t \in \mathcal{S}} x_{it}$. Intuitively, this sets $\mathbf{x}_i$ to the nearest vertex of the simplex $\left\{\mathbf{x}_i \in \mathbb{R}^d_+ : \mathbf{1}^\top \mathbf{x}_i = 1\right\}$. The second scheme, called **BCD rounding** [41, 61], consists in minimizing $E(\mathbf{x})$ over $\mathbf{x}_i$ while keeping all $\mathbf{x}_j$ ($j \neq i$) fixed (i.e., block coordinate descent), which amounts to iteratively assigning each node i with label $s_i \in \operatorname{argmin}_{s \in \mathcal{S}} \left\{\theta_i(s) + \sum_{j \in \mathcal{N}_i} \sum_{t \in \mathcal{S}} \theta_{ij}(s,t) x_{jt}\right\}$. In practice, we only use nearest rounding because BCD rounding is too expensive for dense graphs. However, an important property of the latter is that it does not increase the energy, which is useful for our theoretical analysis. The following theorem provides an additive bound on the energy.

Theorem 2. *Let $\mathbf{x}_r^*$ be a global minimum of $E_r(\mathbf{x}) = E(\mathbf{x}) + r(\mathbf{x})$ over $\mathcal{X}$, $\bar{\mathbf{x}}_r^*$ be the discrete solution rounded from $\mathbf{x}_r^*$, and E^* be the minimum discrete energy. Assume that $r(\mathbf{x})$ is bounded:*[8] *$m \leq r(\mathbf{x}) \leq M\ \forall \mathbf{x} \in \mathcal{X}$. We have $E^* \leq E(\bar{\mathbf{x}}_r^*) \leq E^* + M - m + C$, where $C = \sqrt{n\left(1 - \frac{1}{d}\right)}\left(\|\mathbf{u}\|_2 + \sqrt{n}\,\|\mathbf{P}\|_2\right)$ for nearest rounding and $C = 0$ for BCD rounding.*

Let us derive the energy BCD bound for some particular cases (see §C.2 for details). Obviously with no regularization ($r = 0$), we have $M = m = 0$ and thus $E(\bar{\mathbf{x}}_r^*) \leq E^* \leq E(\bar{\mathbf{x}}_r^*)$, yielding $E(\bar{\mathbf{x}}_r^*) = E^*$, i.e., the relaxation is tight. We have thus recovered a previously known result [10, 41, 61]. For $r(\mathbf{x}) = -\mathbf{c}^\top \mathbf{x} + \mathbf{x}^\top \operatorname{diag}(\mathbf{c})\mathbf{x}$ with $\mathbf{c} \geq \mathbf{0}$, we have $M = 0$ and $m = -\frac{1}{4}\mathbf{1}^\top \mathbf{c}$, which recovers exactly the additive bound given by Ravikumar and Lafferty [61] for the convex QP relaxation. For the ℓ_2 regularizer $r(\mathbf{x}) = \frac{\lambda}{2}\|\mathbf{x}\|_2^2$, we have $M = \frac{\lambda n}{2}$ and $m = \frac{\lambda n}{2d}$, thus we obtain a bound of $\frac{\lambda n}{2}\left(1 - \frac{1}{d}\right)$. For the entropy regularizer $r(\mathbf{x}) = -\lambda H(\mathbf{x})$, we have $M = 0$ and $m = -\lambda n \log d$, thus the bound is $\lambda n \log d$, which is worse than the ℓ_2 bound for any $d \geq 5$.

[7] If both f and g are *convex*, a better rate of $\mathcal{O}(1/k)$ exists [5, 7, 74]. In addition, if g is the indicator function of a convex set $\mathcal{X}$ (i.e., vanilla Frank-Wolfe) and either f or $\mathcal{X}$ is strongly convex, a linear rate can be obtained [32, 50, 59]. Note that we use quite different machinery from all these analyses, due to the nonconvexity.

[8] A sufficient condition is r being continuous, as $\mathcal{X}$ is compact.

Note that the bound provided by Theorem 2 is achieved from a *global* minimum of the regularized relaxation. This can be attained in some cases, e.g., when the energy is submodular or when the (convex) regularizer is large enough to make the objective convex. In the general case, however, the algorithm is only guaranteed to reach a stationary point, and the (theoretical) quality of such point remains unknown. It would be interesting to investigate whether the algorithm can provide an approximation guarantee for some classes of energies (e.g., supermodular ones), similar to some existing algorithms [11]. These open questions are left for future work.

5 Experiments

We compare regularized Frank-Wolfe with existing methods on the semantic segmentation task, in terms of both *inference* and *learning* performance. Two variants, namely Euclidean Frank-Wolfe (ℓ_2**FW**) and Entropic Frank-Wolfe (e**FW**) (§3.3), will be compared to the following methods: Mean field (**MF**) [34, 35] (which is our baseline), nonconvex vanilla Frank-Wolfe (**FW**) [41] (§2.3), projected gradient descent (**PGD**) [39, 41], fast proximal gradient method (**PGM**) [9, 48], and alternating direction method of multipliers (**ADMM**) [40, 41]. Convex vanilla Frank-Wolfe [21] and (entropic) mirror descent [8, 55] were found to perform poorly in our experiments, and thus excluded from the presentation. Other methods based on CCCP [21] or LP relaxation [1, 21] are also excluded due to their sophisticated implementations. For all methods, we set the initial solution to $\mathbf{x}^0 = \mathrm{softmax}(-\mathbf{u})$, following previous work [34]. Further details on implementation, running time, and memory footprint can be found in Appendices D–E.

5.1 Experimental setup

Our segmentation model is a standard combination of a CNN and a CRF [35, 76] (Appendix E.1). For the CNN part, we consider two strong architectures: DeepLabv3 with ResNet101 backbone [16], and DeepLabv3+ with Xception65 backbone [17]. The CRF part is a fully-connected one [34] in which any pair of pixels (i, j) is an edge with potential $\theta_{ij}(s,t) = \mu(s,t)k(\mathbf{f}_i, \mathbf{f}_j)\ \forall s,t \in \mathcal{S}$, where $\mu : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is called label compatibility function, and k is a Gaussian kernel over image features based on pixel coordinates and colors. The setup of our models are similar to Zheng et al. [76]. We use the Potts compatibility function: $\mu(s,t) = w\mathbb{1}_{[s \neq t]}$ with $w = 1$ for the inference experiments in §5.2, and also for CRF initialization in the learning experiments in §5.3. For all experiments, a fully-trained CNN is needed. We follow closely the published recipes [16, 17] for this task. We first pretrain DeepLabv3 and DeepLabv3+ on the COCO dataset [46] and then finetune them on PASCAL VOC (*trainaug*) and Cityscapes (*train*) to obtain similar results to previous work [16, 17] (Table 1, CNN column). Finally, we perform experiments on two popular datasets: (augmented) PASCAL VOC [22] and Cityscapes [19]. Further details are given in Appendix E.

5.2 Inference performance

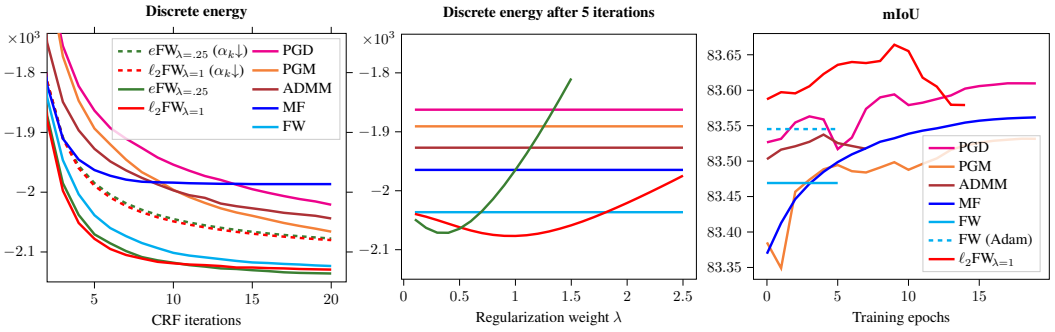
In this section, we compare the performance of regularized Frank-Wolfe against the competing methods in terms of inference. We consider a Potts CRF on top of a CNN, which is the typical setup for using dense CRF in post-processing. Figure 1a shows the *discrete* energy per inference iteration for each method, averaged over the 1449 *val* images of PASCAL VOC, using DeepLabv3+. One can observe that Frank-Wolfe variants completely outperform the other methods. In addition, regularized Frank-Wolfe outperforms all the other methods for a large range of λ, as shown in Figure 1b.

Table 1 shows the performance on the validation sets of PASCAL VOC and Cityscapes, for a Potts CRF with both DeepLabv3 and DeepLabv3+ as backbone. In this experiment, we run all the methods for 10 iterations. One can observe that ℓ_2FW achieved the best performance, followed by eFW.

We should note some inconsistency compared to the energy results previously presented in Figure 1a. For example, eFW achieved much lower energy than MF, yet the mIoU gap is marginal; also, FW accuracy is slightly worse than MF while the energy is much better (lower). This can

		CNN	PGD	PGM	ADMM	MF	FW	eFW$_{.7}$	eFW$_{.3}$	ℓ_2FW
VOC	DL3	81.83	82.23	82.23	82.22	82.21	82.27	82.26	**82.29**	**82.29**
	DL3+	82.89	83.36	83.37	83.38	83.45	83.43	83.45	83.48	**83.50**
CITY	DL3	76.73	76.88	76.86	76.95	76.97	76.86	76.99	76.99	**77.03**
	DL3+	79.55	79.64	79.63	**79.66**	79.63	79.64	79.65	**79.66**	**79.66**

Table 1: Validation mIoU using a Potts CRF on top of the pretrained CNN models. DL means DeepLab.



(a) Discrete energy comparison. **(b)** Effect of regularization weight. **(c)** Learning failure of vanilla FW.

Figure 1: Results on PASCAL VOC validation set using DeepLabv3+ and Potts dense CRF. **(a)** Comparison between CRF solvers ($\alpha_k\!\downarrow$ means $\alpha_k = k/(k+2)\ \forall k$) shows that Frank-Wolfe variants clearly outperform the other methods in terms of energy minimization. **(b)** Performance of regularized Frank-Wolfe can be greatly affected by λ, but it can still achieve lower energies than the other methods for a large range of λ. **(c)** Vanilla Frank-Wolfe completely fails to learn because of the zero-gradient issue.

be explained by the fact that the Potts model is not a perfect representation (i.e., lower energy in this model does not necessarily translate to higher accuracy). In the next section, we will see how the methods perform when the CRF parameters are learned from data.

5.3 Learning performance

In this section, we evaluate the performance of the methods for joint CNN-CRF end-to-end training. The CNN is initialized with its fully-trained weights on the corresponding dataset, and the CRF is initialized with the Potts model with random noise added. We train the model for 20 epochs with 5 CRF iterations,[9] using the same poly schedule as before. As the CNN has been already fully-trained, we set its learning rate to a small value of 0.0001. For the CRF, we tried 4 different values of initial learning rates $\eta_0 \in \{1.0, 0.1, 0.01, 0.001\}$ and found that 1.0 is too high (training diverges quickly) while 0.001 is too low (slow progress) for all methods. For the remaining candidates $\{0.1, 0.01\}$, we perform 4 additional trainings for each method (i.e., a total of 5 runs for each configuration).

Let us summarize our findings. First, we observe that (vanilla) FW fails to learn. This is illustrated in Figure 1c, where we show the validation accuracy per epoch on PASCAL VOC for each method: FW did not make any progress. We tried a different optimizer (Adam [33]) and obtained similar results. This is expected as the gradient in vanilla FW is zero almost everywhere, as previously discussed in §3.1 (see also §C.1). Our second observation is that training is quite unstable for PGD, PGM, ADMM, $e\text{FW}_{.3}$, and $\ell_2\text{FW}$. In particular, $\eta_0 = 0.1$ is still too high for these methods, and even with $\eta_0 = 0.01$, some of the runs produced bad results. By contrast, MF and $e\text{FW}_{.7}$ are stable for both learning rates, with 0.1 being slightly better. A possible explanation is that PGD, PGM, ADMM, and $\ell_2\text{FW}$ all employ a simplex projection step that is not fully differentiable (but only so almost everywhere). For $e\text{FW}_{.3}$ (which is fully differentiable), we hypothesize that the low regularization makes the problem less "smooth", which may also harm gradient-based training. Finally, with the above training scheme, we observe that none of the CRF methods could improve over the CNN (but rather the opposite) on Cityscapes. We have seen that the Potts CRF was able to achieve some marginal improvements (Table 1), thus it is reasonable to expect even better performance with end-to-end training.

In view of the above observations, we present a simple trick to make CRF training more stable. The idea is to replace the CRF output $\mathbf{x}^*$ with $\frac{1}{2}(\mathbf{x}^* + \mathbf{x}^0)$, where we recall that the initialization $\mathbf{x}^0$ is the $\mathrm{softmax}$ of the CNN logits. Intuitively, this adds a skip connection from the CNN to the CRF output in the computation graph, which makes the gradient of the loss propagate directly to the CNN. We found that this trick also slightly improves $e\text{FW}_{.7}$, but has a negative effect on MF. Therefore, it is applied to all methods except MF. Finally, as Cityscapes requires a very high number of epochs, we set this value to 100. Also because training on Cityscapes requires a lot more computing resources, we only perform a single run on DeepLabv3+. The results are presented in Table 2.

[9] While we use the same number of iterations at *test time* to simplify the evaluation protocol, it should be noted that using more iterations could be beneficial. See Appendix F.3 for some results.

Again, ℓ_2FW consistently achieves the best results. Interestingly, while eFW$_{.3}$ achieved similar performance to ℓ_2FW in terms of energy minimization (Figure 1a and Table 1), its performance is worse in joint training. Compared to Table 1, we see that joint training produced much larger improvements over the CNNs, up to 2.25% on PASCAL VOC and 0.4% on Cityscapes.

		CNN	PGD	PGM	ADMM	MF	eFW$_{.7}$	eFW$_{.3}$	ℓ_2FW
VOC	DL3	81.83	83.69 ± 0.20	**83.75** ± 0.23	83.68 ± 0.06	83.69 ± 0.10	83.50 ± 0.10	83.25 ± 0.20	**83.75** ± 0.13
	DL3+	82.89	84.82 ± 0.23	84.79 ± 0.20	84.83 ± 0.06	84.87 ± 0.17	84.64 ± 0.23	84.50 ± 0.16	**85.14** ± 0.09
CITY	DL3+	79.55	79.80	79.62	79.62	79.74	79.70	79.58	**79.95**

Table 2: Validation mIoU under joint training. For PASCAL VOC, we report the mean and standard deviation from 5 runs.

Performance on the test sets We select the best performing method (DeepLabv3+ with ℓ_2FW CRF) for evaluation on the test sets. For PASCAL VOC, we further train our model on the union of the *train* and *val* subsets for 50 epochs. For Cityscapes, we further train 200 epochs on *train* and *train_extra*, using the high-quality annotations provided by Tao et al. [70] (for *train_extra*). At the 150^{th} epoch, we replace *train_extra* with *val*. For this fine-tuning step, learning rates were set to 0.001 for CNN and 0.1 for CRF. For prediction, we apply test time augmentation including left-right flipping and multi-scales. For reference, we train DeepLabv3+ alone using the same recipes. Table 3 shows that we were able to closely match the performance reported by Chen et al. [17]. Adding the ℓ_2FW CRF yields an improvement of 0.4 points on PASCAL VOC. Unfortunately we only observe a marginal improvement on Cityscapes.

Model	VOC	CITY
DeepLabv3+ [17]	87.8	82.1
DeepLabv3+ (this work)	87.6	83.5
DeepLabv3+ with ℓ_2FW CRF	**88.0**	**83.6**

Table 3: Performance on the *test* sets. Submission URLs are given in Appendix F.1.

5.4 Ablation studies

Trainable α_k and λ It is possible to learn α_k and λ from data by simply setting them to be *trainable*. We carried out such an experiment with ℓ_2FW and eFW but did not observe significant improvements, though we should note that a more sophisticated training recipe (e.g., using custom learning rates for these variables) might lead to better results. Details are provided in Appendix F.2.

Fine-grained analysis We observe that CRF improved over CNN on most of the semantic classes. In particular, on `bicycle` (known to be the most challenging class of PASCAL VOC [16]), ℓ_2FW and eFW achieved improvements of over 10% absolute in mIoU. See Appendix F.3 for the details.

6 Discussion & conclusion

Why does it work? Theoretically, all the methods in §5 should reach a stationary point, so how can one be better than another? In fact, Figure 1a only shows that Frank-Wolfe variants work better than the other methods *in the first few iterations*, but not necessarily in a later stage. Indeed, the same conclusion no longer holds after 100 iterations (see §F.4), but this long regime is not practical because it would lead to vanishing/exploding gradients [76] and to potentially prohibitive memory consumption. As to why Frank-Wolfe achieves lower energy in the early stage, we hypothesize that this could be due to the discreteness of its iterates (7). With small λ, the solution by regularized Frank-Wolfe should be close to the vanilla one, and thus also benefits from this property. It is important to note that the benefit of regularized Frank-Wolfe does not lie in the extra (sometimes small) energy improvement over vanilla Frank-Wolfe, but in its ability to seamlessly solve the zero-gradient issue.

How to tune λ? We found that similar curves to Figure 1b can be obtained using a small random subset (e.g., 10 samples) of the data, which suggests a quick way of tuning λ by random subsampling. In practice, this step takes only a few seconds, which is negligible in most training scenarios.

Limitations While one variant of regularized Frank-Wolfe (ℓ_2FW) consistently achieves the best results, the difference compared to the other methods is sometimes small. In addition, the improvement of dense CRFs over CNNs is marginal on the Cityscapes test set. Nevertheless, we hope the encouraging results on PASCAL VOC could attract interest from the community in CRF research, potentially leading to creative ways of overcoming these limitations.

Societal impact Semantic segmentation models can be used in surveillance systems, which might raise potential privacy concerns. Furthermore, the datasets that our models were trained on are known to present strong built-in bias [71], thus they should be used with caution.

Acknowledgments and Disclosure of Funding

This work was supported in part by the ANR grant AVENUE (ANR-18-CE23-0011), and was partly done when the first author was affiliated with Manifold Perception (mption.com). The experiments were performed using HPC resources from GENCI-IDRIS (Grants 2020-AD011011321 and 2020-AD011011881). The authors thank the anonymous reviewers and meta-reviewer for their constructive feedback that helped improve the manuscript.

References

[1] Thalaiyasingam Ajanthan, Alban Desmaison, Rudy Bunel, Mathieu Salzmann, Philip HS Torr, and M Pawan Kumar. Efficient linear programming for dense crfs. In *Computer Vision and Pattern Recognition*, 2017.

[2] Thalaiyasingam Ajanthan, Puneet K Dokania, Richard Hartley, and Philip HS Torr. Proximal mean-field for neural network quantization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4871–4880, 2019.

[3] Brandon Amos, Vladlen Koltun, and J Zico Kolter. The limited multi-label projection layer. *arXiv preprint arXiv:1906.08707*, 2019.

[4] Anurag Arnab, Sadeep Jayasumana, Shuai Zheng, and Philip HS Torr. Higher order conditional random fields in deep neural networks. In *European Conference on Computer Vision*, pages 524–540. Springer, 2016.

[5] Francis Bach. Duality between subgradient and conditional gradient methods. *SIAM Journal on Optimization*, 25(1):115–129, 2015.

[6] Pierre Baqué, Timur Bagautdinov, François Fleuret, and Pascal Fua. Principled parallel mean-field inference for discrete random fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5848–5857, 2016.

[7] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[8] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Oper. Res. Lett.*, 31(3):167–175, 2003.

[9] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009.

[10] Marc Berthod. Definition of a consistent labeling as a global extremum. In *International Conference on Pattern Recognition*, pages 339–341, 1982.

[11] Yatao An Bian, Joachim M. Buhmann, and Andreas Krause. Optimal continuous dr-submodular maximization and applications to provable mean field inference. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 644–653. PMLR, 2019.

[12] Kristian Bredies, Dirk A Lorenz, and Peter Maass. A generalized conditional gradient method and its connection to an iterative shrinkage method. *Computational Optimization and Applications*, Advanced online publication, 2007. doi: 10.1007/s10589-007-9083-3.

[13] Siddhartha Chandra and Iasonas Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. In *European conference on computer vision*, pages 402–418. Springer, 2016.

[14] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3640–3649, 2016.

[15] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[16] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[17] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[18] Laurent Condat. Fast projection onto the simplex and the ℓ_1 ball. *Mathematical Programming*, 158(1-2): 575–585, 2016.

[19] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[20] Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1635–1643, 2015.

[21] Alban Desmaison, Rudy Bunel, Pushmeet Kohli, Philip HS Torr, and M Pawan Kumar. Efficient continuous relaxations for dense crf. In *European Conference on Computer Vision*, pages 818–833. Springer, 2016.

[22] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

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

[24] Robert M Freund and Paul Grigas. New analysis and results for the frank–wolfe method. *Mathematical Programming*, 155(1-2):199–230, 2016.

[25] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[26] Zaid Harchaoui, Anatoli Juditsky, and Arkadi Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152(1-2):75–112, 2015.

[27] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[28] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[29] Xu Hu and Guillaume Obozinski. Sdca-powered inexact dual augmented lagrangian method for fast CRF learning. In *AISTATS*, volume 84 of *Proceedings of Machine Learning Research*, pages 988–997. PMLR, 2018.

[30] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th international conference on machine learning*, pages 427–435, 2013.

[31] Vladimir Jojic, Stephen Gould, and Daphne Koller. Accelerated dual decomposition for map inference. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 503–510, 2010.

[32] Thomas Kerdreux, Lewis Liu, Simon Lacoste-Julien, and Damien Scieur. Affine invariant analysis of frank-wolfe on strongly convex sets. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 5398–5408. PMLR, 2021.

[33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[34] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

[35] Philipp Krähenbühl and Vladlen Koltun. Parameter learning and convergent inference for dense random fields. In *International Conference on Machine Learning*, pages 513–521, 2013.

[36] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in neural information processing systems*, pages 496–504, 2015.

[37] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, 2001. ISBN 1-55860-778-1.

[38] Guanghui Lan. The complexity of large-scale convex programming under a linear optimization oracle. *arXiv preprint arXiv:1309.5550*, 2013.

[39] Måns Larsson, Anurag Arnab, Fredrik Kahl, Shuai Zheng, and Philip Torr. A projected gradient descent method for crf inference allowing end-to-end training of arbitrary pairwise potentials. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 564–579. Springer, 2017.

[40] D. Khuê Lê-Huu and Nikos Paragios. Alternating direction graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4914–4922, 2017.

[41] D. Khuê Lê-Huu and Nikos Paragios. Continuous relaxation of map inference: A nonconvex perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5533–5541, 2018.

[42] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1 (4):541–551, 1989.

[43] Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3193–3202, 2017.

[44] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3194–3203, 2016.

[45] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017.

[46] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[47] Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), University of Helsinki*, pages 6–7, 1970.

[48] Pierre-Louis Lions and Bertrand Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

[49] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Semantic image segmentation via deep parsing network. In *Proceedings of the IEEE international conference on computer vision*, pages 1377–1385, 2015.

[50] Francesco Locatello, Rajiv Khanna, Michael Tschannen, and Martin Jaggi. A unified optimization view on generalized matching pursuit and frank-wolfe. In *AISTATS*, volume 54 of *Proceedings of Machine Learning Research*, pages 860–868. PMLR, 2017.

[51] Julien Mairal. Optimization with first-order surrogate functions. In *International Conference on Machine Learning*, pages 783–791, 2013.

[52] Ofer Meshi, Mehrdad Mahdavi, and Alex Schwing. Smooth and strong: Map inference with linear convergence. In *Advances in Neural Information Processing Systems*, pages 298–306, 2015.

[53] Hisashi Mine and Masao Fukushima. A minimization method for the sum of a convex function and a continuously differentiable function. *Journal of Optimization Theory and Applications*, 33(1):9–23, 1981.

[54] Miguel Monteiro, Mário AT Figueiredo, and Arlindo L Oliveira. Conditional random fields as recurrent neural networks for 3d medical imaging segmentation. *arXiv preprint arXiv:1807.07464*, 2018.

[55] Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

[56] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1): 127–152, 2005. ISSN 0025-5610.

[57] Vlad Niculae and Mathieu Blondel. A regularized framework for sparse and structured neural attention. In *Advances in neural information processing systems*, pages 3338–3348, 2017.

[58] Giorgio Parisi. *Statistical field theory*. Addison-Wesley, 1988.

[59] Fabian Pedregosa, Geoffrey Négiar, Armin Askari, and Martin Jaggi. Linearly convergent frank-wolfe without line-search. In *AISTATS*, volume 108 of *Proceedings of Machine Learning Research*, pages 1–10. PMLR, 2020.

[60] Marin Vlastelica Pogancic, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *ICLR*. OpenReview.net, 2020.

[61] Pradeep Ravikumar and John Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *Proceedings of the 23rd international conference on Machine learning*, pages 737–744. ACM, 2006.

[62] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[63] Alexander G Schwing and Raquel Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.

[64] Falong Shen, Rui Gan, Shuicheng Yan, and Gang Zeng. Semantic segmentation via structured patch prediction, context crf and guidance crf. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1953–1961, 2017.

[65] Solomon Eyal Shimony. Finding maps for belief networks is np-hard. *Artificial Intelligence*, 68(2): 399–410, 1994.

[66] Hyun Oh Song, Ross Girshick, Stefanie Jegelka, Julien Mairal, Zaid Harchaoui, and Trevor Darrell. On learning to localize objects with minimal supervision. In *ICML-31st International Conference on Machine Learning*, volume 32, pages 1611–1619. JMLR, 2014.

[67] David A. Sontag and Tommi S. Jaakkola. New outer bounds on the marginal polytope. In *NIPS*, pages 1393–1400. Curran Associates, Inc., 2007.

[68] Haiming Sun, Di Xie, and Shiliang Pu. Mixed context networks for semantic segmentation. *arXiv preprint arXiv:1610.05854*, 2016.

[69] Kui Tang, Nicholas Ruozzi, David Belanger, and Tony Jebara. Bethe learning of graphical models via MAP decoding. In *AISTATS*, volume 51 of *JMLR Workshop and Conference Proceedings*, pages 1096–1104. JMLR.org, 2016.

[70] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv preprint arXiv:2005.10821*, 2020.

[71] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *CVPR*, pages 1521–1528. IEEE Computer Society, 2011.

[72] Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE transactions on information theory*, 51(11):3697–3717, 2005.

[73] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.07122.

[74] Yaoliang Yu, Xinhua Zhang, and Dale Schuurmans. Generalized conditional gradient for sparse estimation. *The Journal of Machine Learning Research*, 18(1):5279–5324, 2017.

[75] Alan L Yuille and Anand Rangarajan. The concave-convex procedure (cccp). In *Advances in neural information processing systems*, pages 1033–1040, 2002.

[76] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] In the appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Our code is available at `https://github.com/netw0rkf10w/CRF`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Some details in the main content and full details in the appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In the appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] We use standard datasets freely available for academic research. They are well-known so we did not mention their licenses.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code and pre-trained models are available at `https://github.com/netw0rkf10w/CRF`.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendices

A Details on special cases of regularized Frank-Wolfe inference

We have seen in §3 multiple instantiations of regularized Frank-Wolfe, leading to new algorithms for MAP inference, as well as recovering many existing ones. In this section we provide further details on this matter.

Recall the notation $n = |\mathcal{V}|, m = |\mathcal{E}|, d = |\mathcal{S}|$, where $\mathcal{V}, \mathcal{E}$ and $\mathcal{S}$ are the sets of nodes, edges, and labels, respectively.

A.1 Algorithms based on QP relaxation with vanilla Frank-Wolfe

Nonconvex vanilla Frank-Wolfe This algorithm, previously studied by Lê-Huu and Paragios [41], was already presented in §2.3. It consists in applying vanilla Frank-Wolfe directly to the energy (6).

Convex vanilla Frank-Wolfe This involves the *convex* QP relaxation of MAP inference introduced by Ravikumar and Lafferty [61]. The idea is to add a sufficiently large vector $\mathbf{c}$ to the diagonal of $\mathbf{P}$ to make it positive semidefinite. If $\mathbf{x} \in \{0,1\}^{nd}$ then it is easy to check that $\mathbf{x}^\top \operatorname{diag}(\mathbf{c})\mathbf{x} = \mathbf{c}^\top \mathbf{x}$ for any $\mathbf{c} \in \mathbb{R}^{nd}$. Therefore, the (discrete) energy can be written as

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top (2\operatorname{diag}(\mathbf{c}) + \mathbf{P})\mathbf{x} + (\mathbf{u} - \mathbf{c})^\top \mathbf{x}. \tag{15}$$

It can be shown that the above function is convex if $\mathbf{c}$ is chosen as follows:

$$c_{is} = \frac{1}{2}\sum_{j \in \mathcal{N}_i}\sum_{t \in \mathcal{S}} \theta_{ij}(s,t) \quad \forall i \in \mathcal{V}, s \in \mathcal{S}, \tag{16}$$

where $\mathcal{N}_i$ denotes the set of neighbors of node i. Applying vanilla Frank-Wolfe to minimizing the above convex energy over $\mathcal{X}$ yields the algorithm presented in section 4 of Desmaison et al. [21].

A.2 Algorithms based on LP relaxation with vanilla Frank-Wolfe

Let us first present the LP relaxation of MAP inference. We use the same notation leading to the energy formulation (4), namely the indicator variables $x_{is} \in \{0,1\}$, the indicator vectors $\mathbf{x}_i \in \{0,1\}^d$, and the potential vectors $\boldsymbol{\theta}_i \in \mathbb{R}^d$ for all nodes $i \in \mathcal{V}$ and labels $s \in \mathcal{S}$. In addition, define for all edges $ij \in \mathcal{E}$ and pairs of labels $(s,t) \in \mathcal{S}^2$:

- New pairwise indicator variables $x_{ijst} = x_{is}x_{jt} \in \{0,1\}$.
- New pairwise indicator vectors $\mathbf{x}_{ij} = (x_{ijst})_{s\in\mathcal{S},t\in\mathcal{S}} \in \{0,1\}^{d^2}$.
- New pairwise potential vectors $\boldsymbol{\theta}_{ij} = (\theta_{ij}(s,t))_{s\in\mathcal{S},t\in\mathcal{S}} \in \mathbb{R}^{d^2}$, which can be viewed as the flatten version of the potential matrices $\boldsymbol{\Theta}_{ij}$ in (4).

Then, the energy (4) can be rewritten as a linear function:

$$E_{\mathrm{LP}}(\mathbf{x};\boldsymbol{\theta}) = \sum_{i\in\mathcal{V}} \boldsymbol{\theta}_i^\top \mathbf{x}_i + \sum_{ij\in\mathcal{E}} \boldsymbol{\theta}_{ij}^\top \mathbf{x}_{ij}, \tag{17}$$

where by slight abuse of notation, we let $\mathbf{x}$ and $\boldsymbol{\theta}$ again denote the vectors of all variables and parameters, respectively. Note that $\mathbf{x}$ and $\boldsymbol{\theta}$ are now $(nd + md^2)$-dimensional vectors and not nd-dimensional as in (4). The LP relaxation consists in minimizing E_{LP} over the following *local polytope* [72]:

$$\mathcal{X}_{\mathrm{LP}} = \left\{ \mathbf{x} \in \mathbb{R}^{nd+md^2} \;\middle|\; \begin{array}{c} \mathbf{x} \geq \mathbf{0}, \\ \mathbf{1}^\top \mathbf{x}_i = 1 \; \forall i \in \mathcal{V}, \\ \sum_{t\in\mathcal{S}} x_{ijst} = x_{is} \; \forall ij \in \mathcal{E}, \forall s \in \mathcal{S}, \\ \sum_{s\in\mathcal{S}} x_{ijst} = x_{jt} \; \forall ij \in \mathcal{E}, \forall t \in \mathcal{S}. \end{array} \right\}. \tag{18}$$

The last two constraints in the above (called *local consistency*) can be written as $\mathbf{A}\mathbf{x} = \mathbf{0}$ for some $(2md) \times (nd + md^2)$ matrix $\mathbf{A}$. We can thus rewrite the LP relaxation compactly as:

$$\min E_{\mathrm{LP}}(\mathbf{x};\boldsymbol{\theta}) \triangleq \boldsymbol{\theta}^\top \mathbf{x}, \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{X}_{\mathrm{LP}} \triangleq \left\{ \mathbf{x} \in \mathbb{R}_+^{nd+md^2} : \mathbf{1}^\top \mathbf{x}_i = 1 \; \forall i \in \mathcal{V}, \mathbf{A}\mathbf{x} = \mathbf{0} \right\}. \tag{19}$$

As presented in §3.4, Sontag and Jaakkola, Meshi et al., Tang et al. [67, 52, 69] apply vanilla Frank-Wolfe to minimize a regularized LP energy:

$$\min_{\mathbf{x}\in\mathcal{X}_{\mathrm{LP}}} E_{\mathrm{LP}}(\mathbf{x};\boldsymbol{\theta}) + r(\mathbf{x}) \tag{20}$$

for some regularizer r. These works differ in the choice of r.

Local-consistency regularization Choosing $r(\mathbf{x}) = \frac{\lambda}{2}\|\mathbf{A}\mathbf{x}\|_2^2$ we obtain the algorithm presented by Meshi et al. [52] (which corresponds to the primal algorithm in the top-right cell of their Table 1).

Bethe and TRW entropic regularization Sontag and Jaakkola [67] also apply vanilla Frank-Wolfe to a regularized LP energy (corresponding to Step 3 in their Algorithm 1; note that we consider only the first outer iteration of their algorithm). They consider regularizers of the form

$$r(\mathbf{x}) = -\tilde{H}(\mathbf{x}), \tag{21}$$

where $\tilde{H}(\mathbf{x})$ is some approximation to the entropy $H(\mathbf{x})$ of the distribution over $\mathbf{x}$.

Define the singleton entropy

$$H(\mathbf{x}_i) = -\sum_{s\in\mathcal{S}} x_{is} \log x_{is} \quad \forall i \in \mathcal{V}, \tag{22}$$

and the pairwise mutual information

$$I(\mathbf{x}_{ij}) = \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{S}} x_{ijst} \log \frac{x_{ijst}}{x_{is} x_{jt}} = -H(\mathbf{x}_{ij}) + H(\mathbf{x}_i) + H(\mathbf{x}_j) \quad \forall ij \in \mathcal{E}. \tag{23}$$

The so-called Bethe approximation is defined as:

$$\tilde{H}_{\text{Bethe}}(\mathbf{x}) = \sum_{i \in \mathcal{V}} H(\mathbf{x}_i) - \sum_{ij \in \mathcal{E}} I(\mathbf{x}_{ij}). \tag{24}$$

The second approximation considered by [67] is called tree-reweighted (TRW) approximation. To achieve this, we decompose the the graph into a convex combination of spanning trees according to some distribution (over the trees), and let ρ_{ij} be the so-called *edge appearance probability*, which is computed as the number of spanning trees containing the edge ij in the current decomposition, divided by the total number of all possible spanning trees containing ij (in the entire distribution). The TRW approximation is then given by

$$\tilde{H}_{\text{TRW}}(\mathbf{x}) = \sum_{i \in \mathcal{V}} H(\mathbf{x}_i) - \sum_{ij \in \mathcal{E}} \rho_{ij} I(\mathbf{x}_{ij}). \tag{25}$$

$\boldsymbol{\rho}$-reweighted entropic regularization Tang et al. [69] consider a more general term than the previous ones, based on the following approximation to $z \log z$ for $z \in [0, 1]$, parameterized by $\eta \in [0, 1]$:

$$H_\eta(z) = \begin{cases} -z \log z & \text{if } z \in [\eta, 1], \\ -\eta \log \eta - (1 + \log \eta)(z - \eta) - \frac{(z-\eta)^2}{2\eta} & \text{if } z \in [0, \eta]. \end{cases} \tag{26}$$

Define a similar version for vectors:

$$H_\eta(\mathbf{z}) = \sum_{i=1}^{p} H_\eta(z_i) \quad \forall \mathbf{z} \in \mathbb{R}^p. \tag{27}$$

Their $\boldsymbol{\rho}$-reweighted approximation to the entropy $H(\mathbf{x})$ is given by:

$$\tilde{H}^\rho_\eta(\mathbf{x}) = \sum_{i \in \mathcal{V}} H_\eta(\mathbf{x}_i) - \sum_{ij \in \mathcal{E}} \rho_{ij} \left[-H_\eta(\mathbf{x}_{ij}) + H_\eta(\mathbf{x}_i) + H_\eta(\mathbf{x}_j) \right] \tag{28}$$

Tang et al. [69] apply vanilla Frank-Wolfe to $E_{\text{LP}} + r$ where $r = -\tilde{H}^\rho_\eta$. Note that their work consists in learning parameters of graphical models through maximum likelihood estimation. Here we only consider the inference part presented in their Section 3.2, which is used as a subroutine for learning.

A.3 Algorithms based on the concave-convex procedure

In the dense CRF model proposed by Krähenbühl and Koltun [34], the pairwise potentials consist of weighted sums of Gaussian kernels:

$$\theta_{ij}(s, t) = \sum_{c=1}^{C} \mu^{(c)}(s, t) k^{(c)}(\mathbf{f}_i, \mathbf{f}_j) \quad \forall i, j \in \mathcal{V}, \forall s, t \in \mathcal{S}, \tag{29}$$

where C is the number of components, $\mu^{(c)} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ are the so-called label compatibility functions, and $k^{(c)}$ are Gaussian kernels over some image features $(\mathbf{f}_i, \mathbf{f}_j)$ (§E.1 presents a concrete example implemented for our experiments).

Define kernel matrices $\mathbf{K}^{(c)} \in \mathbb{R}^{n \times n}$ with elements $K^{(c)}_{ij} = k(\mathbf{f}_i, \mathbf{f}_j)$ and compatibility matrices $\mathbf{M}^{(c)} \in \mathbb{R}^{d \times d}$ with elements $M^{(c)}_{st} = \mu^{(c)}(s, t)$. Let

$$\mathbf{M} = \sum_{c=1}^{C} \mathbf{M}^{(c)}. \tag{30}$$

If we assume that $\mathbf{K}^{(c)} \in \mathbb{R}^{n \times n}$ has unit diagonal: $K^{(c)}_{ii} = 1 \; \forall i, \forall c$, then our pairwise potential matrix $\mathbf{P}$ can be written as

$$\mathbf{P} = \sum_{c=1}^{C} \left(\mathbf{K}^{(c)} - \mathbf{I}_n \right) \otimes \mathbf{M}^{(c)} = -\mathbf{I}_n \otimes \mathbf{M} + \sum_{c=1}^{C} \mathbf{K}^{(c)} \otimes \mathbf{M}^{(c)}, \tag{31}$$

where $\otimes$ denotes the Kronecker product, and $\mathbf{I}_n$ is the $n \times n$ identity matrix.

In the following, the concave-convex procedure (CCCP) [75] is applied to minimizing $f(\mathbf{x}) + g(\mathbf{x})$ where f is concave and g is convex. (We integrate the constraint set $\mathcal{X}$ into g using its indicator function $\delta_{\mathcal{X}}$, for consistency with our presentation of regularized Frank-Wolfe.)

Convergent mean field Krähenbühl and Koltun [35] proposed (in their section 3.1) to minimizing a regularized energy $E(\mathbf{x}) + \mathbf{x}^\top \log \mathbf{x}$ (entropic regularizer) by applying CCCP to:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top(\mathbf{P} + \mathbf{I}_n \otimes \mathbf{M})\mathbf{x} + \mathbf{u}^\top \mathbf{x}, \tag{32}$$

$$g(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^\top(\mathbf{I}_n \otimes \mathbf{M})\mathbf{x} + \mathbf{x}^\top \log \mathbf{x} + \delta_{\mathcal{X}}(\mathbf{x}). \tag{33}$$

Convergent mean field using concave approximation Krähenbühl and Koltun [35] proposed (in their section 3.2) a more efficient algorithm using:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top(\mathbf{P} + \mathbf{I}_n \otimes \mathbf{M})\mathbf{x} + \mathbf{u}^\top \mathbf{x}, \tag{34}$$

$$g(\mathbf{x}) = \mathbf{x}^\top \log \mathbf{x} + \delta_{\mathcal{X}}(\mathbf{x}) \tag{35}$$

CCCP for QP relaxation 1 Desmaison et al. [21] proposed (in their section 5.1) the following application of CCCP:

$$f(\mathbf{x}) = -\mathbf{x}^\top \operatorname{diag}(c)\mathbf{x}, \tag{36}$$

$$g(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top(2\operatorname{diag}(\mathbf{c}) + \mathbf{P})\mathbf{x} + \mathbf{u}^\top \mathbf{x} + \delta_{\mathcal{X}}(\mathbf{x}), \tag{37}$$

where $\mathbf{c}$ is defined by (16).

CCCP for QP relaxation 2 Inspired by Krähenbühl and Koltun [35], Desmaison et al. [21] also proposed (in their section 5.2) another more efficient variant:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top(\mathbf{P} + \mathbf{I}_n \otimes \mathbf{M})\mathbf{x}, \tag{38}$$

$$g(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^\top(\mathbf{I}_n \otimes \mathbf{M})\mathbf{x} + \mathbf{u}^\top \mathbf{x} + \delta_{\mathcal{X}}(\mathbf{x}). \tag{39}$$

A.4 Summary of special cases

We provide in Table 4 a summary of special cases discussed in this section as well as in §3.3 and §3.4. There we show how these algorithms can be obtained from regularized Frank-Wolfe by suitably choosing f, g and r in Algorithm 1. Recall that $f + g = E + r + \delta_{\mathcal{X}}$.

B Detailed convergence analysis

In this section, let $\|\cdot\|$ denote the ℓ_2 norm. The following lemma is useful for the proofs.

Lemma 2. *If f and g satisfy Assumption 1 and 2, then*

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L_f}{2}\|\mathbf{y} - \mathbf{x}\|^2 \quad \forall \mathbf{x}, \mathbf{y} \in \operatorname{dom} f, \tag{40}$$

$$g(\mathbf{y}) \geq g(\mathbf{x}) + \langle \mathbf{d}, \mathbf{y} - \mathbf{x} \rangle + \frac{\sigma_g}{2}\|\mathbf{y} - \mathbf{x}\|^2 \quad \forall \mathbf{x}, \mathbf{y} \in \operatorname{dom} g, \forall \mathbf{d} \in \partial g(\mathbf{x}). \tag{41}$$

Proof. For a convex function h, we have

$$h(\mathbf{y}) \geq h(\mathbf{x}) + \langle \mathbf{d}, \mathbf{y} - \mathbf{x} \rangle \quad \forall \mathbf{x}, \mathbf{y}, \forall \mathbf{d} \in \partial h(\mathbf{x}). \tag{42}$$

Applying the above inequality with, respectively, $h(\mathbf{x}) = -f(\mathbf{x}) + \frac{L_f}{2}\|\mathbf{x}\|_2^2$ and $h(\mathbf{x}) = g(\mathbf{x}) - \frac{\sigma_g}{2}\|\mathbf{x}\|_2^2$, we obtain (40) and (41). (Note that for the second case, h is convex and thus $\partial g(\mathbf{x}) = \partial h(\mathbf{x}) + \sigma_g \mathbf{x}$.) □

Algorithm	$f(\mathbf{x})$	$g(\mathbf{x}) - \delta_{\mathcal{X}}(\mathbf{x})$
Parallel mean field Krähenbühl and Koltun [34]	$E(\mathbf{x})$	$\mathbf{x}^\top \log \mathbf{x}$
Convergent mean field 1 §3.1 in Krähenbühl and Koltun [35]	$E(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top (\mathbf{I}_n \otimes \mathbf{M})\, \mathbf{x}$	$-\frac{1}{2}\mathbf{x}^\top (\mathbf{I}_n \otimes \mathbf{M})\, \mathbf{x} + \mathbf{x}^\top \log \mathbf{x}$
Convergent mean field 2 §3.2 in Krähenbühl and Koltun [35]	$E(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top (\mathbf{I}_n \otimes \mathbf{M})\, \mathbf{x}$	$\mathbf{x}^\top \log \mathbf{x}$
Nonconvex Frank-Wolfe Lê-Huu and Paragios [41]	$E(\mathbf{x})$	0
Convex Frank-Wolfe §4 in Desmaison et al. [21]	$E(\mathbf{x}) - \mathbf{c}^\top \mathbf{x} + \mathbf{x}^\top \operatorname{diag}(\mathbf{c}) \mathbf{x}$	0
CCCP for QP relaxation 1 §5.1 in Desmaison et al. [21]	$-\mathbf{x}^\top \operatorname{diag}(\mathbf{c}) \mathbf{x}$	$E(\mathbf{x}) + \mathbf{x}^\top \operatorname{diag}(\mathbf{c}) \mathbf{x}$
CCCP for QP relaxation 2 §5.2 in Desmaison et al. [21]	$E(\mathbf{x}) + \frac{1}{2}\mathbf{x}^\top (\mathbf{I}_n \otimes \mathbf{M})\, \mathbf{x} - \mathbf{u}^\top \mathbf{x}$	$-\frac{1}{2}\mathbf{x}^\top (\mathbf{I}_n \otimes \mathbf{M}) \mathbf{x} + \mathbf{u}^\top \mathbf{x}$
LP local-consistency Frank-Wolfe Meshi et al. [52]	$E_{\mathrm{LP}}(\mathbf{x}) + \frac{\lambda}{2} \left\|\mathbf{A}\mathbf{x}\right\|_2^2$	0
LP Bethe Frank-Wolfe Sontag and Jaakkola [67]	$E_{\mathrm{LP}}(\mathbf{x}) + \tilde{H}_{\mathrm{Bethe}}(\mathbf{x})$	0
LP TRW Frank-Wolfe Sontag and Jaakkola [67]	$E_{\mathrm{LP}}(\mathbf{x}) + \tilde{H}_{\mathrm{TRW}}(\mathbf{x})$	0
LP $\boldsymbol{\rho}$-reweighted Frank-Wolfe Tang et al. [69]	$E_{\mathrm{LP}}(\mathbf{x}) + \tilde{H}^{\rho}_{\eta}(\mathbf{x})$	0
Euclidean Frank-Wolfe (This work)	$E(\mathbf{x})$	$\frac{\lambda}{2} \left\|\mathbf{x}\right\|_2^2$
Entropic Frank-Wolfe (This work)	$E(\mathbf{x})$	$\lambda \mathbf{x}^\top \log \mathbf{x}$
Lasso Frank-Wolfe (This work, not implemented)	$E(\mathbf{x})$	$\lambda \left\|\mathbf{x}\right\|_1$

Table 4: Summary of special cases of regularized Frank-Wolfe.

B.1 Proof of Lemma 1

First we show that

$$S(\mathbf{x}) \geq \frac{\sigma_g}{2} \left\|\mathbf{x} - \mathbf{p}_{\mathbf{x}}\right\|^2 \ \forall \mathbf{x} \in \operatorname{dom} f. \tag{43}$$

Notice that

$$\mathbf{p}_{\mathbf{x}} \in \operatorname*{argmin}_{\mathbf{p}} \left\{\langle \nabla f(\mathbf{x}), \mathbf{p} \rangle + g(\mathbf{p})\right\} \iff -\nabla f(\mathbf{x}) \in \partial g(\mathbf{p}_{\mathbf{x}}). \tag{44}$$

Hence, applying (41) we obtain

$$g(\mathbf{x}) \geq g(\mathbf{p}_{\mathbf{x}}) + \langle -\nabla f(\mathbf{x}), \mathbf{x} - \mathbf{p}_{\mathbf{x}} \rangle + \frac{\sigma_g}{2} \left\|\mathbf{x} - \mathbf{p}_{\mathbf{x}}\right\|^2, \tag{45}$$

which is precisely (43).

To complete the proof, we need to show that $S(\mathbf{x}^*) = 0$ if and only if $\mathbf{x}^*$ is a stationary point of (9), i.e., $-\nabla f(\mathbf{x}^*) \in \partial g(\mathbf{x}^*)$. The following is due to Beck [7]. Notice that

$$S(\mathbf{x}) = \max_{\mathbf{p}} \left\{\langle \nabla f(\mathbf{x}), \mathbf{x} - \mathbf{p} \rangle + g(\mathbf{x}) - g(\mathbf{p})\right\}, \tag{46}$$

we have

$$\begin{aligned} S(\mathbf{x}^*) = 0 \iff S(\mathbf{x}^*) \leq 0 &\iff \langle \nabla f(\mathbf{x}^*), \mathbf{x}^* - \mathbf{p} \rangle + g(\mathbf{x}^*) - g(\mathbf{p}) \leq 0 \ \forall \mathbf{p} && (47)\\ &\iff g(\mathbf{p}) \geq g(\mathbf{x}^*) + \langle -\nabla f(\mathbf{x}^*), \mathbf{p} - \mathbf{x}^* \rangle \ \forall \mathbf{p} && (48)\\ &\iff -\nabla f(\mathbf{x}^*) \in \partial g(\mathbf{x}^*). && (49) \end{aligned}$$

The proof is completed.

B.2 Proof of Theorem 1

We need an additional lemma.

Lemma 3. *For any* $\mathbf{x} \in \operatorname{dom} f$ *and any* $\alpha \in [0,1]$ *we have*

$$F(\mathbf{x}+\alpha(\mathbf{p}_{\mathbf{x}}-\mathbf{x}))-F(\mathbf{x}) \leq -\alpha S(\mathbf{x}) + K(\alpha)\left\|\mathbf{p}_{\mathbf{x}}-\mathbf{x}\right\|^2, \tag{50}$$

where $K(\alpha)=\frac{1}{2}\left[(L_f+\sigma_g)\alpha^2-\sigma_g\alpha\right]$.

Proof. On one hand, from (40) we have

$$f(\mathbf{x}+\alpha(\mathbf{p}_{\mathbf{x}}-\mathbf{x})) \leq f(\mathbf{x}) + \alpha\left\langle \nabla f(\mathbf{x}), \mathbf{p}_{\mathbf{x}}-\mathbf{x}\right\rangle + \frac{L_f\alpha^2}{2}\left\|\mathbf{p}_{\mathbf{x}}-\mathbf{x}\right\|^2. \tag{51}$$

On the other hand, from the σ_g-strong-convexity of g:

$$g(\mathbf{x}+\alpha(\mathbf{p}_{\mathbf{x}}-\mathbf{x})) \leq (1-\alpha)g(\mathbf{x}) + \alpha g(\mathbf{p}_{\mathbf{x}}) - \frac{\sigma_g\alpha(1-\alpha)}{2}\left\|\mathbf{p}_{\mathbf{x}}-\mathbf{x}\right\|^2. \tag{52}$$

Summing up the above two inequalities, we obtain (50). □

Let $S_k = S(\mathbf{x}^k)$, $r_k = \left\|\mathbf{p}^k-\mathbf{x}^k\right\|^2$, and $F_k = F(\mathbf{x}^k)$. Applying (50) we have

$$\boxed{F_k - F_{k+1} \geq \alpha_k S_k - K(\alpha_k) r_k.} \tag{53}$$

Therefore,

$$\Delta_0 = F_0 - F^* \geq F_0 - F_{k+1} = \sum_{i=0}^{k}(F_i - F_{i+1}) \geq S\sum_{i=0}^{k}\alpha_i - \sum_{i=0}^{k} r_i K(\alpha_i), \tag{54}$$

which implies

$$S \leq \frac{\Delta_0 + \sum_{i=0}^{k} r_i K(\alpha_i)}{\sum_{i=0}^{k}\alpha_i}. \tag{55}$$

This is an important inequality that will help us to obtain the convergence results for the weak stepsize schemes such as constant and non-summable ones.

For the adaptive (and line-search) stepsizes, the following observations will be useful. Notice that the RHS of (50) can be written as $\frac{1}{2}at^2 - bt$ where

$$a = (L_f+\sigma_g)\left\|\mathbf{p}_{\mathbf{x}}-\mathbf{x}\right\|^2, \quad b = S(\mathbf{x}) + \frac{\sigma_g}{2}\left\|\mathbf{p}_{\mathbf{x}}-\mathbf{x}\right\|^2.$$

Therefore:

- If $L_f = \sigma_g = 0$ then the RHS is just $-tS(\mathbf{x})$.
- If $L_f = 0$ then the RHS is $-tS(\mathbf{x}) - \frac{\sigma_g}{2}t(1-t)\left\|\mathbf{p}_{\mathbf{x}}-\mathbf{x}\right\|^2 \leq -tS(\mathbf{x})$.
- If $L_f + \sigma_g > 0$ then the minimum of the RHS is

$$-\frac{b^2}{2a} = -\frac{\left\|\mathbf{p}_{\mathbf{x}}-\mathbf{x}\right\|^2}{2(L_f+\sigma_g)}\left(\frac{S(\mathbf{x})}{\left\|\mathbf{p}_{\mathbf{x}}-\mathbf{x}\right\|^2} + \frac{\sigma_g}{2}\right)^2,$$

  achieved at

$$t^* = \frac{b}{a} = \frac{1}{L_f+\sigma_g}\left(\frac{S(\mathbf{x})}{\left\|\mathbf{p}_{\mathbf{x}}-\mathbf{x}\right\|^2} + \frac{\sigma_g}{2}\right).$$

The adaptive stepsize (14) are defined for the case $L_f + \sigma_g > 0$ is thus:

$$\alpha_k = \min\left\{1, \alpha_k^*\right\}, \quad \text{where } \alpha_k^* = \frac{1}{L_f+\sigma_g}\left(\frac{S_k}{r_k} + \frac{\sigma_g}{2}\right). \tag{56}$$

Notice that the RHS of (53) is a quadratic function of α_k with critical point α_k^*. If $\alpha_k^* \leq 1$, or equivalently $\frac{\sigma_g}{2} + L_f \geq \frac{S_k}{r_k}$, then $\alpha_k = \alpha_k^*$ and thus (53) becomes

$$F_k - F_{k+1} \geq \alpha_k^* S_k - K(\alpha_k^*) r_k = \frac{r_k}{2(L_f + \sigma_g)} \left(\frac{S_k}{r_k} + \frac{\sigma_g}{2} \right)^2. \tag{57}$$

If $\alpha_k^* > 1$, or equivalently $\frac{\sigma_g}{2} + L_f < \frac{S_k}{r_k}$, then $\alpha_k = 1$ and thus (53) becomes

$$F_k - F_{k+1} \geq S_k - K(1) r_k = S_k - \frac{L_f}{2} r_k. \tag{58}$$

For simplicity and clarity, we will consider separately the two cases: g is strongly convex ($\sigma_g > 0$) or simply convex ($\sigma_g = 0$). Recall that Ω is the (finite) diameter of $\operatorname{dom} g$, and thus we have $r_k \leq \Omega^2 \; \forall k$, a fact that we will be using repeatedly in the sequel. Let $S = \min_{0 \leq i \leq k} S_i$.

B.2.1 Convex g

In this section we consider the case where g is convex but not strongly convex, i.e., $\sigma_g = 0$. Inequality (55) becomes

$$S \leq \frac{\Delta_0 + \frac{L_f}{2} \sum_{i=0}^k r_i \alpha_i^2}{\sum_{i=0}^k \alpha_i}. \tag{59}$$

Constant stepsize Consider $\alpha_k = \alpha > 0 \; \forall k$. From $r_k \leq \Omega^2$ and (59) we obtain

$$S \leq \frac{\Delta_0}{(k+1)\alpha} + \frac{L_f \Omega^2 \alpha}{2}. \tag{60}$$

The right-hand side converges to $\frac{1}{2} L_f \Omega^2 \alpha$ as $k \to \infty$, i.e., the lower-bound S on the conditional gradient norm converges to within $\frac{1}{2} L_f \Omega^2 \alpha$. It is easy to deduce from the last inequality that $S \leq L_f \Omega^2 \alpha$ within $k \leq \frac{2\Delta_0}{L_f \Omega^2 \alpha^2}$ steps. We conclude that the algorithm converges to an approximate stationary point for the constant stepsize.

Constant step length For a constant step length: $\left\| \mathbf{x}^{k+1} - \mathbf{x}^k \right\| = \alpha$. Recall that $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k (\mathbf{p}^k - \mathbf{x}^k)$, the corresponding stepsize is thus $\alpha_k = \frac{\alpha}{\|\mathbf{p}^k - \mathbf{x}^k\|} = \frac{\alpha}{\sqrt{r_k}}$. Inequality (59) becomes

$$S \leq \frac{\Delta_0 + \frac{L_f}{2}(k+1)\alpha^2}{\alpha \sum_{i=0}^k \frac{1}{\sqrt{r_k}}} \leq \frac{\Delta_0 + \frac{L_f}{2}(k+1)\alpha^2}{\alpha \frac{k+1}{\Omega}} = \frac{\Delta_0 \Omega}{(k+1)\alpha} + \frac{L_f \Omega \alpha}{2}. \tag{61}$$

Therefore, S converges to within $\frac{L_f \Omega \alpha}{2}$, and $S \leq L_f \Omega \alpha$ within $k \leq \frac{2\Delta_0}{L_f \alpha^2}$ steps. We conclude that the algorithm converges to an approximate stationary point for the stepsizes with constant step length.

Non-summable but square-summable stepsizes Assume that the stepsizes α_k satisfy

$$\sum_{k=0}^{+\infty} \alpha_k = \infty, \qquad \sum_{k=0}^{+\infty} \alpha_k^2 < \infty. \tag{62}$$

A typical example is $\alpha_k = \frac{\alpha}{k+\beta}$, where $\alpha > 0$ and $\beta \geq 0$. This includes the common Frank-Wolfe stepsize $\alpha_k = \frac{2}{k+2}$. From $r_k \leq \Omega^2$ and (59) we obtain

$$S \leq \frac{\Delta_0 + \frac{L_f \Omega^2}{2} \sum_{i=0}^k \alpha_i^2}{\sum_{i=0}^k \alpha_i}, \tag{63}$$

which clearly converges to 0 as $k \to \infty$. Therefore, the algorithm is guaranteed to converge to a stationary point in this case.

Diminishing (and non-summable) stepsizes Assume that the stepsizes α_k satisfy

$$\sum_{k=0}^{+\infty} \alpha_k = \infty, \qquad \lim_{k\to\infty} \alpha_k = 0. \tag{64}$$

A typical example is $\alpha_k = \frac{\alpha}{\sqrt{k}}$, where $\alpha > 0$. Notice that for any $\epsilon > 0$, we have $\alpha_i^2 < \epsilon\alpha_i$ with i large enough, it is straightforward to show that $\frac{\sum_{i=0}^k \alpha_i^2}{\sum_{i=0}^k \alpha_i} \to 0$ as $k \to \infty$, and thus (63) implies that $S \to 0$ as well. We conclude that the algorithm is guaranteed to converge to a stationary point.

Adaptive stepsizes This result was obtained previously by Beck [7]. These stepsizes are computed according to (56), which can be simplified as the following for $\sigma_g = 0$:

$$\alpha_k = \min\left\{1, \alpha_k^*\right\}, \quad \text{where } \alpha_k^* = \frac{S_k}{L_f r_k}. \tag{65}$$

Then, if $\alpha_k^* \leq 1$, (57) yields

$$F_k - F_{k+1} \geq \frac{S_k^2}{2L_f r_k} \geq \frac{S_k^2}{2L_f\Omega^2}. \tag{66}$$

If $\alpha_k^* > 1$ then (58) and $L_f r_k < S_k$ yield

$$F_k - F_{k+1} \geq S_k - \frac{L_f}{2} r_k \geq \frac{S_k}{2}. \tag{67}$$

Combining the two cases, we obtain

$$F_k - F_{k+1} \geq \frac{S_k}{2} \min\left\{1, \frac{S_k}{L_f\Omega^2}\right\} \geq \frac{S}{2} \min\left\{1, \frac{S}{L_f\Omega^2}\right\}. \tag{68}$$

Therefore,

$$\Delta_0 = F_0 - F^* \geq F_0 - F_{k+1} = \sum_{i=0}^{k} (F_i - F_{i+1}) \geq (k+1)\frac{S}{2} \min\left\{1, \frac{S}{L_f\Omega^2}\right\}, \tag{69}$$

which yields

$$S \leq \max\left\{\frac{2\Delta_0}{k+1}, \frac{\sqrt{2L_f\Omega^2\Delta_0}}{\sqrt{k+1}}\right\}. \tag{70}$$

Therefore, the algorithm is guaranteed to converge to a stationary point, and the rate of convergence convergence is at least $\mathcal{O}(1/\sqrt{k})$.

B.2.2 Strongly-convex g

In this section we consider the case where g is strongly convex with parameter $\sigma_g > 0$.

Recall from (53) and Lemma 3 that

$$F_k - F_{k+1} \geq \alpha_k S_k - K(\alpha_k) r_k \;\forall k \geq 0, \quad \text{where } K(\alpha) = \frac{1}{2}\alpha\left[(L_f + \sigma_g)\alpha - \sigma_g\right]. \tag{71}$$

Thus if $\alpha_k \leq \frac{\sigma_g}{L_f + \sigma_g}$ we have $K(\alpha_k) \leq 0$ and (71) yields

$$F_k - F_{k+1} \geq \alpha_k S_k. \tag{72}$$

Consider now the case $\alpha_k > \frac{\sigma_g}{L_f + \sigma_g}$ for which $K(\alpha_k) > 0$. From (43) we have $r_k \leq \frac{2S_k}{\sigma_g}$, and thus (71) yields

$$F_k - F_{k+1} \geq \alpha_k S_k - K(\alpha_k)\frac{2S_k}{\sigma_g} = \left(\alpha_k - \frac{2K(\alpha_k)}{\sigma_g}\right) S_k = \alpha_k \left(2 - \frac{L_f + \sigma_g}{\sigma_g}\alpha_k\right) S_k. \tag{73}$$

Combining the two cases, we obtain

$$F_k - F_{k+1} \geq \alpha_k \min\left(1, 2 - \frac{L_f + \sigma_g}{\sigma_g}\alpha_k\right) S_k. \tag{74}$$

If $\alpha_k \geq \frac{2\sigma_g}{L_f + \sigma_g}$ then the RHS of (74) is non-positive, thus this inequality is not helpful. In this case, we can obtain another inequality from (71), noticing that $r_k \leq \Omega^2 \;\forall k$ and $K(\alpha_k) > 0$:

$$F_k - F_{k+1} \geq \alpha_k S_k - K(\alpha_k)\Omega^2 \tag{75}$$

Constant stepsize Assume that $\alpha_k = \alpha \; \forall k \geq 0$. If $0 < \alpha < \frac{2\sigma_g}{L_f + \sigma_g}$ then (74) yields

$$F_k - F_{k+1} \geq \alpha \min\left(1, 2 - \frac{L_f + \sigma_g}{\sigma_g}\alpha\right) S \; \forall k \geq 0. \tag{76}$$

Hence

$$\Delta_0 \geq F_0 - F_{k+1} = \sum_{i=0}^{k}(F_i - F_{i+1}) \geq (k+1)\alpha \min\left(1, 2 - \frac{L_f + \sigma_g}{\sigma_g}\alpha\right) S. \tag{77}$$

We obtain

$$S \leq \frac{\Delta_0}{\alpha \min\left(1, 2 - \frac{L_f + \sigma_g}{\sigma_g}\alpha\right)(k+1)} \quad \forall \alpha < \frac{2\sigma_g}{L_f + \sigma_g}. \tag{78}$$

We conclude that the algorithm is guaranteed to converge to a stationary point with rate of convergence of (at least) $\mathcal{O}(1/k)$ for any $0 < \alpha < \frac{2\sigma_g}{L_f + \sigma_g}$.

For the remaining case $\alpha \geq \frac{2\sigma_g}{L_f + \sigma_g}$, we will derive an upper bound for S. Applying (75) we obtain

$$\Delta_0 \geq \sum_{i=0}^{k}(F_i - F_{i+1}) \geq \alpha \sum_{i=0}^{k} S_i - (k+1)K(\alpha)\Omega^2 \geq (k+1)\alpha S - (k+1)K(\alpha)\Omega^2, \tag{79}$$

which yields

$$S \leq \frac{\Delta_0}{\alpha(k+1)} + \frac{K(\alpha)\Omega^2}{\alpha} = \frac{\Delta_0}{\alpha(k+1)} + \frac{1}{2}\left[(L_f + \sigma_g)\alpha - \sigma_g\right]\Omega^2. \tag{80}$$

Therefore, for $\alpha \geq \frac{2\sigma_g}{L_f + \sigma_g}$ the algorithm converges to an approximate stationary point at which the conditional gradient norm is bounded above by $\frac{1}{2}\left[(L_f + \sigma_g)\alpha - \sigma_g\right]\Omega^2$.

Constant step length Consider the stepsize $\alpha_k = \frac{\alpha}{\|\mathbf{p}^k - \mathbf{x}^k\|} = \frac{\alpha}{\sqrt{r_k}}$ for which $\left\|\mathbf{x}^{k+1} - \mathbf{x}^k\right\| = \alpha$. Inequality (71) becomes

$$F_k - F_{k+1} \geq \frac{\alpha}{\sqrt{r_k}} S_k - K\left(\frac{\alpha}{\sqrt{r_k}}\right) r_k \tag{81}$$

$$= \frac{\alpha}{\sqrt{r_k}} S_k - \frac{1}{2}\left[(L_f + \sigma_g)\frac{\alpha^2}{r_k} - \sigma_g \frac{\alpha}{\sqrt{r_k}}\right] r_k \tag{82}$$

$$= \frac{\alpha}{\sqrt{r_k}} S_k + \frac{\sigma_g \alpha \sqrt{r_k}}{2} - \frac{1}{2}(L_f + \sigma_g)\alpha^2 \tag{83}$$

$$\geq 2\sqrt{\frac{\alpha}{\sqrt{r_k}} S_k \frac{\sigma_g \alpha \sqrt{r_k}}{2}} - \frac{1}{2}(L_f + \sigma_g)\alpha^2 \tag{84}$$

$$= \alpha\sqrt{2\sigma_g S_k} - \frac{1}{2}(L_f + \sigma_g)\alpha^2. \tag{85}$$

It follows that

$$\Delta_0 \geq (k+1)\alpha\sqrt{2\sigma_g S} - \frac{k+1}{2}(L_f + \sigma_g)\alpha^2 \tag{86}$$

$$\implies \sqrt{S} \leq \frac{\Delta_0}{\alpha\sqrt{2\sigma_g}(k+1)} + \frac{(L_f + \sigma_g)\alpha}{2\sqrt{2\sigma_g}}. \tag{87}$$

For this stepsize scheme, the algorithm converges to an approximate stationary point, within $\frac{(L_f + \sigma_g)^2 \alpha^2}{8\sigma_g}$. One can observe that, even though the strong convexity of g still cannot guarantee convergence to a stationary point, it helps improve the bound as well as the rate of convergence from (61).

Diminishing (and non-summable) stepsizes Assume that the stepsizes α_k satisfy

$$\sum_{k=0}^{+\infty} \alpha_k = \infty, \qquad \lim_{k\to\infty} \alpha_k = 0. \tag{88}$$

This scheme also includes the non-summable but square-summable one. Since $\lim_{k\to\infty} \alpha_k = 0$, there exists an integer $k(\omega)$ such that $\alpha_k \leq \omega = \frac{\sigma_g}{L_f+\sigma_g}$ $\forall k \geq k(\omega)$. Now applying (72)

$$\Delta_{k(\omega)} \geq F_{k(\omega)} - F_{k+1} = \sum_{i=k(\omega)}^{k} (F_i - F_{i+1}) \geq \sum_{i=k(\omega)}^{k} \alpha_i S_i \geq \left(\sum_{i=k(\omega)}^{k} \alpha_i \right) S, \tag{89}$$

which yields

$$S \leq \frac{\Delta_{k(\omega)}}{\sum_{i=k(\omega)}^{k} \alpha_i}. \tag{90}$$

Since (α_k) is non-summable, the algorithm converges to a stationary point. Compared to the non-strongly-convex case, we observe that the assumption that $\operatorname{dom} g$ is compact can be relaxed (its diameter Ω is not used in the proof).

Adaptive stepsizes Recall that the stepsizes in this scheme are given by (56) as

$$\alpha_k = \min\left\{1, \alpha_k^*\right\}, \quad \text{where } \alpha_k^* = \frac{1}{L_f+\sigma_g}\left(\frac{S_k}{r_k} + \frac{\sigma_g}{2}\right). \tag{91}$$

If $\alpha_k^* \leq 1$, from (57) and the inequality $(a+b)^2 \geq 4ab$, we obtain

$$F_k - F_{k+1} \geq \frac{r_k}{2(L_f+\sigma_g)} 4 \frac{S_k}{r_k}\frac{\sigma_g}{2} = \frac{\sigma_g S_k}{L_f+\sigma_g}. \tag{92}$$

If $\alpha_k^* > 1$, which is $r_k < \frac{S_k}{\frac{\sigma_g}{2}+L_f}$, then (58) yields

$$F_k - F_{k+1} \geq S_k - \frac{L_f}{2}\frac{S_k}{\frac{\sigma_g}{2}+L_f} = \frac{\sigma_g+L_f}{\sigma_g+2L_f} S_k \geq \frac{\sigma_g}{\sigma_g+L_f} S_k. \tag{93}$$

Therefore, we always have $F_k - F_{k+1} \geq \omega S_k$ where $\omega = \frac{\sigma_g}{\sigma_g+L_f}$. It then follows that

$$\Delta_0 \geq \sum_{i=0}^{k} (F_i - F_{i+1}) \geq \sum_{i=0}^{k} \omega S_k \geq (k+1)\omega S \implies S \leq \frac{\Delta_0}{\omega(k+1)}. \tag{94}$$

Finally, the line search scheme is guaranteed to achieve the best decrease in the objective, thus the inequality $F_k - F_{k+1} \geq \omega S_k$ also holds and we obtain the same results for this scheme.

B.2.3 Concave f

The results for this case can be obtained in a straightforward manner by setting $L_f = 0$ in the "convex g" case.

B.2.4 Summary of convergence results

We summarize the results in Table 5.

B.2.5 Convergence of $S(\mathbf{x}^k)$

To complete the proof of Theorem 1, we need to show that for the non-highlighted cases of its table (page 6), we have $\lim_{k\to\infty} S(\mathbf{x}^k) = 0$ and any limit point of the sequence $(\mathbf{x}^k)_{k\geq 0}$ is a stationary point of (9). Indeed, for these cases, $(F_k)_{k\geq 0}$ is a decreasing sequence because $F_k - F_{k+1}$ is bounded below by a non-negative quantity δ_k (according to Table 5 presented in the previous section). Therefore, F_k is convergent as it is bounded below by F^*. Consequently $F_k - F_{k+1} \to 0$, which implies $\delta_k \to 0$ and thus $S_k \to 0$ as well for the considered cases (see Table 5). The results follow in a straightforward manner.

	stepsize	decrease lower bound δ_k $(F_k - F_{k+1} \geq \delta_k)$	optimality upper bound B_k $(\min_{0\leq i\leq k} S_i \leq B_k)$
convex g	$\alpha_k = \alpha > 0$	$\alpha S_k - \frac{L_f \Omega^2 \alpha^2}{2}$	$\frac{\Delta_0}{(k+1)\alpha} + \frac{L_f \Omega^2 \alpha}{2}$
	$\alpha_k = \frac{\alpha}{\|\mathbf{p}^k - \mathbf{x}^k\|}$	$\frac{\alpha}{\Omega} S_k - \frac{L_f \alpha^2}{2}$	$\frac{\Delta_0 \Omega}{(k+1)\alpha} + \frac{L_f \Omega \alpha}{2}$
	$\sum_{k=0}^{+\infty} \alpha_k = \infty$ $\lim_{k\to\infty} \alpha_k = 0$	$\alpha_k S_k - \frac{L_f \Omega^2 \alpha_k^2}{2}$	$\frac{\Delta_0}{\sum_{i=0}^k \alpha_i} + \frac{L_f \Omega^2}{2} \frac{\sum_{i=0}^k \alpha_i^2}{\sum_{i=0}^k \alpha_i}$
	adaptive or line search (14)	$\frac{1}{2} \min\left(S_k, \frac{S_k^2}{L_f \Omega^2}\right)$	$\max\left(\frac{2\Delta_0}{k+1}, \frac{\sqrt{2 L_f \Omega^2 \Delta_0}}{\sqrt{k+1}}\right)$
strongly-convex g	$\alpha_k = \alpha < 2\omega$	$\alpha \min\left(1, 2 - \frac{\alpha}{\omega}\right) S_k$	$\frac{\Delta_0}{\alpha \min\left(1, 2-\frac{\alpha}{\omega}\right)(k+1)}$
	$\alpha_k = \alpha \geq 2\omega$	$\alpha S_k - K(\alpha)\Omega^2$	$\frac{\Delta_0}{\alpha(k+1)} + \frac{K(\alpha)}{\alpha}\Omega^2$
	$\alpha_k = \frac{\alpha}{\|\mathbf{p}^k - \mathbf{x}^k\|}$	$\alpha\sqrt{2\sigma_g S_k} - \frac{1}{2}(L_f + \sigma_g)\alpha^2$	$\left(\frac{\Delta_0}{\alpha\sqrt{2\sigma_g}(k+1)} + \frac{(L_f+\sigma_g)\alpha}{2\sqrt{2\sigma_g}}\right)^2$
	$\sum_{k=0}^{+\infty} \alpha_k = \infty$ $\lim_{k\to\infty} \alpha_k = 0$	$\alpha_k \min\left(1, 2 - \frac{\alpha_k}{\omega}\right) S_k$	$\frac{\Delta_{k(\omega)}}{\sum_{i=k(\omega)}^k \alpha_i}$
	adaptive or line search (14)	ωS_k	$\frac{\Delta_0}{\omega(k+1)}$
concave f	$\alpha_k = \alpha > 0$	αS_k	$\frac{\Delta_0}{(k+1)\alpha}$
	$\alpha_k = \frac{\alpha}{\|\mathbf{p}^k - \mathbf{x}^k\|}$	$\frac{\alpha}{\Omega} S_k$	$\frac{\Delta_0 \Omega}{(k+1)\alpha}$
	$\sum_{k=0}^{+\infty} \alpha_k = \infty$	$\alpha_k S_k$	$\frac{\Delta_0}{\sum_{i=0}^k \alpha_i}$
	adaptive or line search (14)	$\frac{1}{2} S_k$	$\frac{2\Delta_0}{k+1}$

Table 5: Summary of convergence analysis of the generalized Frank-Wolfe algorithm. Recall that $\omega = \frac{\sigma_g}{L_f + \sigma_g}$. Whenever a result does not involve Ω, the assumption that $\operatorname{dom} g$ being compact can be relaxed.

C Proofs of other theoretical results

C.1 Vanilla Frank-Wolfe fails to learn: the zero-gradient issue

We claimed in §3.1 that vanilla Frank-Wolfe (7) is problematic for learning with SGD because its iterates are piecewise-constant and thus their gradients are zero almost everywhere (more precisely the gradient is undefined on the boundaries while being zero everywhere else). In this section, we present a theoretical justification for this claim.

It suffices to show that $\mathbf{p}^* = \operatorname{argmin}_{\mathbf{p}\in\mathcal{X}} \langle \mathbf{c}, \mathbf{p} \rangle$ is piecewise-constant with respect to $\mathbf{c}$. Let Δ_d denote the simplex $\left\{\mathbf{z} \in \mathbb{R}^d \mid \mathbf{1}^\top \mathbf{z} = 1, \mathbf{z} \geq \mathbf{0}\right\}$. Clearly, the set $\mathcal{X}$ (defined by (5)) can be written as $\left\{\mathbf{x} \in \mathbb{R}^{nd} \mid \mathbf{x}_i \in \Delta_d \; \forall i \in \mathcal{V}\right\}$, and thus the above minimization problem can be reduced to solving the following problem for each $i \in \mathcal{V}$ independently:

$$\mathbf{p}_i^* \in \operatorname*{argmin}_{\mathbf{p}_i \in \Delta_d} \langle \mathbf{c}_i, \mathbf{p}_i \rangle . \tag{95}$$

For notational convenience, consider the following problem with a constant vector $\mathbf{b} = (b_1, b_2, \ldots, b_d) \in \mathbb{R}^d$:

$$\mathbf{z}^* \in \operatorname*{argmin}_{\mathbf{z}\in\Delta_d} \langle \mathbf{b}, \mathbf{z} \rangle . \tag{96}$$

Let s^* be the index of the minimum element of $\mathbf{b}$, i.e., $s^* = \operatorname{argmin}_s b_s$. Let $\mathbf{e}_s \in \{0,1\}^d$ denote the one-hot vector where the s^{th} element is one. We have:

$$\langle \mathbf{b}, \mathbf{z} \rangle = \sum_{s=1}^d b_s z_s \geq \sum_{s=1}^d b_{s^*} z_s = b_{s^*} \sum_{s=1}^d z_s = b_{s^*} = \langle \mathbf{b}, \mathbf{e}_{s^*} \rangle \quad \forall \mathbf{z} \in \Delta_d. \tag{97}$$

Therefore, $\mathbf{e}_{s^*}$ is an optimal solution to (96). It is straightforward that the index of the minimum element of a vector is piecewise constant, thus $\mathbf{e}_{s^*}$ is also piecewise constant (as a function of $\mathbf{b}$). Therefore, $\mathbf{e}_{s^*}$ is not continuous (thus non-differentiable) on the boundaries, while in the constant regions, its gradient is zero.

Remark. We can deduce that the iteration complexity of vanilla Frank-Wolfe is $\mathcal{O}(nd)$.

C.2 Proof of the relaxation tightness (Theorem 2)

We give a proof of Theorem 2 in §4.2. Recall that we have to prove $E^* \leq E(\bar{\mathbf{x}}_r^*) \leq E^* + M - m + C$, where

$$C = \begin{cases} \sqrt{n\left(1-\frac{1}{d}\right)}\left(\|\mathbf{u}\|_2 + \sqrt{n}\,\|\mathbf{P}\|_2\right) & \text{for nearest rounding} \\ 0 & \text{for BCD rounding.} \end{cases} \tag{98}$$

Let $\mathbf{x}^*$ be such that $E(\mathbf{x}^*) = E^*$ and consider first the BCD rounding scheme. As this scheme is guaranteed to not increase the energy, we have

$$E(\bar{\mathbf{x}}_r^*) \leq E(\mathbf{x}_r^*) = E_r(\mathbf{x}_r^*) - r(\mathbf{x}_r^*) \leq E_r(\mathbf{x}^*) - r(\mathbf{x}_r^*) = E(\mathbf{x}^*) + r(\mathbf{x}^*) - r(\mathbf{x}_r^*) \leq E^* + M - m.$$

It remains to prove the result for the nearest rounding scheme. In this scheme, the discrete energy may increase (or decrease), but it can be shown that the variation is bounded by the given constant: $|E(\bar{\mathbf{x}}_r^*) - E(\mathbf{x}_r^*)| \leq C$. Then, the rest of the proof is similar to the BCD case. This bounding inequality is proved as follows.

Suppose that we obtain a discrete solution $\mathbf{y} \in \mathcal{X} \cap \{0,1\}^{nd}$ from some $\mathbf{x} \in \mathcal{X}$ using nearest rounding. We will prove that

$$|E(\mathbf{x}) - E(\mathbf{y})| \leq C, \tag{99}$$

where

$$C = \sqrt{n\left(1-\frac{1}{d}\right)}\left(\|\mathbf{u}\|_2 + \sqrt{n}\,\|\mathbf{P}\|_2\right). \tag{100}$$

Lemma 4. *For any $\mathbf{z} \in \Delta_d$ (see §C.1 for notation) and its rounded vector $\mathbf{v} \in \Delta_d \cap \{0,1\}^d$, i.e., $v_i = 1$ if $i = \operatorname{argmax}_{1\leq j\leq d} v_j$ and $v_j = 0\ \forall j \neq i$, we have*

$$\|\mathbf{z} - \mathbf{v}\|_2^2 \leq 1 - \frac{1}{d}. \tag{101}$$

Proof. Without loss of generality, assume that z_1 is the maximum element of $\mathbf{z}$. Then, we have $v_1 = 1$ and $v_j = 0\ \forall j > 1$.

$$\|\mathbf{z} - \mathbf{v}\|_2^2 = \sum_{i=1}^{d}(z_i - v_i)^2 = (z_1 - 1)^2 + \sum_{i=1}^{d} z_i^2 = S^2 + z_2^2 + \cdots + z_d^2, \tag{102}$$

where $S = z_2 + \cdots + z_d$. We will make use of the following trivial inequality:

$$z_i + S \leq 1 \quad \forall i \geq 2. \tag{103}$$

On one hand, summing the $d-1$ inequalities (103) (for $i = 2, \ldots, d$) we obtain

$$S + (d-1)S \leq d - 1 \implies S \leq 1 - \frac{1}{d}. \tag{104}$$

On the other hand, multiplying (103) with z_i and summing up the obtained $d-1$ inequalities we get

$$\sum_{i=2}^{d} z_i^2 + S^2 \leq S \tag{105}$$

Finally, from (102), (104), and (105) we get (101). □

Back to (99). Applying (101) we have

$$\|\mathbf{x} - \mathbf{y}\|_2^2 = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{y}_i\|_2^2 \leq n\left(1 - \frac{1}{d}\right). \tag{106}$$

On the other hand

$$\|\mathbf{x} + \mathbf{y}\|_2^2 = \sum_{i=1}^{n} \|\mathbf{x}_i + \mathbf{y}_i\|_2^2 \leq \sum_{i=1}^{n} \left[\mathbf{1}^\top(\mathbf{x}_i + \mathbf{y}_i)\right]^2 = 4n. \tag{107}$$

Applying the two above inequalities, together with the triangle and Cauchy-Schwarz inequalities we have:

$$\begin{aligned}
|E(\mathbf{x}) - E(\mathbf{y})| &= \left|\mathbf{u}^\top(\mathbf{x}-\mathbf{y}) + \frac{1}{2}\mathbf{x}^\top\mathbf{P}\mathbf{x} - \frac{1}{2}\mathbf{y}^\top\mathbf{P}\mathbf{y}\right| \\
&\leq \left|\mathbf{u}^\top(\mathbf{x}-\mathbf{y})\right| + \frac{1}{2}\left|\mathbf{x}^\top\mathbf{P}\mathbf{x} - \mathbf{y}^\top\mathbf{P}\mathbf{y}\right| \\
&= \left|\mathbf{u}^\top(\mathbf{x}-\mathbf{y})\right| + \frac{1}{2}\left|(\mathbf{x}-\mathbf{y})^\top\mathbf{P}(\mathbf{x}+\mathbf{y})\right| \\
&\leq \|\mathbf{u}\|_2 \|\mathbf{x}-\mathbf{y}\|_2 + \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2 \|\mathbf{P}\|_2 \|\mathbf{x}+\mathbf{y}\|_2 \\
&\leq \|\mathbf{u}\|_2 \sqrt{n\left(1-\frac{1}{d}\right)} + \frac{1}{2}\sqrt{n\left(1-\frac{1}{d}\right)}\|\mathbf{P}\|_2 2\sqrt{n} \\
&= C.
\end{aligned}$$

This completes the proof.

D Implementation details of all methods

We present the implementation details for all the methods presented in the experiments (§5). Recall that our problem of interest is

$$\min_{\mathbf{x}\in\mathcal{X}} E(\mathbf{x}) \triangleq \frac{1}{2}\mathbf{x}^\top\mathbf{P}\mathbf{x} + \mathbf{u}^\top\mathbf{x},$$

and that the same initialization $\mathbf{x}^0 = \mathrm{softmax}(-\mathbf{u})$ is used for all methods.

High-dimensional filtering for gradient computation For all methods, we need to compute the energy gradient $\nabla E(\mathbf{x}) = \mathbf{P}\mathbf{x} + \mathbf{u}$ at each iteration, where the evaluation of $\mathbf{P}\mathbf{x}$ is an expensive operation because the graph is fully-connected (i.e., $\mathbf{P}$ is dense). Fortunately, since the pairwise potentials are Gaussian, this multiplication can be performed efficiently (and approximately) in $\mathcal{O}(nd)$ time using high-dimensional filtering, which is the key idea behind the original dense CRFs paper [34]. We refer to this reference for more details. Our code is based on the efficient GPU implementation of Monteiro et al. [54].[10]

D.1 Euclidean Frank-Wolfe (ℓ_2FW)

We give the details for the main update (11) of Euclidean Frank-Wolfe as presented in §3.3. This step follows from

$$\begin{aligned}
\forall k \geq 0: \quad \mathbf{p}^k &= \operatorname*{argmin}_{\mathbf{p}\in\mathcal{X}} \left\{ \langle \mathbf{P}\mathbf{x}^k + \mathbf{u}, \mathbf{p}\rangle + \frac{\lambda}{2}\|\mathbf{p}\|_2^2 \right\} && (108) \\
&= \operatorname*{argmin}_{\mathbf{p}\in\mathcal{X}} \left\{ \frac{\lambda}{2}\left\|\mathbf{p} + \frac{1}{\lambda}(\mathbf{P}\mathbf{x}^k + \mathbf{u})\right\|^2 \right\} && (109) \\
&= \Pi_{\mathcal{X}}\left(-\frac{1}{\lambda}(\mathbf{P}\mathbf{x}^k + \mathbf{u})\right). && (110)
\end{aligned}$$

Recall that $\Pi_{\mathcal{X}}(\mathbf{v})$ denotes the projection of a vector $\mathbf{v}$ onto the set $\mathcal{X}$. Recall also from (5) that $\mathcal{X} = \left\{\mathbf{x}\in\mathbb{R}^{nd} : \mathbf{x}\geq\mathbf{0}, \mathbf{1}^\top\mathbf{x}_i = 1\ \forall i\in\mathcal{V}\right\}$, thus the projection on $\mathcal{X}$ clearly reduces to n *independent* projections onto the probability simplex $\Delta_d = \left\{\mathbf{z}\in\mathbb{R}^d : \mathbf{1}^\top\mathbf{z} = 1, \mathbf{z}\geq\mathbf{0}\right\}$ for each $i\in\mathcal{V}$. Projection onto the simplex is a rather well studied problem in the literature [18], and we present below the solution (that also shows how we implemented this operation).

Lemma 5. *For a given vector $\mathbf{c}\in\mathbb{R}^d$, the optimal solution $\mathbf{z}^*$ to*

$$\min_{\mathbf{1}^\top\mathbf{z}=1, \mathbf{z}\geq\mathbf{0}} \|\mathbf{z}-\mathbf{c}\|^2 \qquad (111)$$

[10] https://github.com/MiguelMonteiro/permutohedral_lattice

is given as follows. Sort $\mathbf{c}$ *in decreasing order to obtain a vector* $\mathbf{a} = (a_1, a_2, \ldots, a_d)$ *(i.e.,* $a_1 \geq a_2 \geq \cdots \geq a_d$*) and let*

$$\gamma_k = \frac{1}{k}(a_1 + a_2 + \cdots + a_k - 1), \quad k = 1, 2, \ldots, d. \tag{112}$$

Let k^* *be the largest* k *such that* $a_k > \gamma_k$*, then the optimal solution is given by*

$$\mathbf{z}^* = \max(\mathbf{c} - \gamma_{k^*}, 0). \tag{113}$$

In the above, the "$\max$" and "$-$" operations are understood to be element-wise.

Remark. If we use an $\mathcal{O}(d \log d)$ sorting algorithm, then we see that the per-iteration complexity of Euclidean Frank-Wolfe is $\mathcal{O}(nd \log d)$. It should be noted, however, that highly-efficient simplex-projection algorithms exist and have $\mathcal{O}(d)$ complexity in practice [18, Table 1], yielding $\mathcal{O}(nd)$ complexity, which is the same as in vanilla Frank-Wolfe (see §C.1).

D.2 Entropic Frank-Wolfe (eFW)

We give the details for the main update (12) of Entropic Frank-Wolfe as presented in §3.3. We need to show that

$$\mathbf{p}^k = \operatorname*{argmin}_{\mathbf{p} \in \mathcal{X}} \left\{ \left\langle \mathbf{P}\mathbf{x}^k + \mathbf{u}, \mathbf{p} \right\rangle - \lambda H(\mathbf{p}) \right\} = \operatorname{softmax}\left(-\frac{1}{\lambda}(\mathbf{P}\mathbf{x}^k + \mathbf{u}) \right) \quad \forall k \geq 0, \tag{114}$$

where $H(\mathbf{x}) = -\sum_{i \in \mathcal{V}} \sum_{s \in \mathcal{S}} x_{is} \log x_{is}$. Again, the above reduces to n independent subproblems over each $i \in \mathcal{V}$ to which the solutions are given by the following lemma.

Lemma 6. *For a given vector* $\mathbf{c} \in \mathbb{R}^d$*, the optimal solution* $\mathbf{z}^*$ *to*

$$\min_{\mathbf{1}^\top \mathbf{z} = 1, \mathbf{z} \geq \mathbf{0}} \left\{ \langle \mathbf{c}, \mathbf{z} \rangle + \sum_{s=1}^{d} z_s \log z_s \right\} \tag{115}$$

is $\mathbf{z}^* = \operatorname{softmax}(-\mathbf{c})$.

Proof. The Lagrangian of the above problem is given by

$$L(\mathbf{z}, \boldsymbol{\mu}, \nu) = \langle \mathbf{c}, \mathbf{z} \rangle + \sum_{s=1}^{d} z_s \log z_s + \boldsymbol{\mu}^\top (-\mathbf{z}) + \nu(\mathbf{1}^\top \mathbf{z} - 1) \tag{116}$$

$$= -\nu + \sum_{s=1}^{d} \left(c_s z_s + z_s \log z_s - \mu_s z_s + \nu z_s \right), \tag{117}$$

where $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_d) \geq \mathbf{0}$ and $\nu \in \mathbb{R}$ are the Lagrange multipliers.

Observe that the given problem is convex and the corresponding Slater's constraint qualification holds (i.e., there exists $\mathbf{z} \in \mathbb{R}^d$ such that $\mathbf{1}^\top \mathbf{z} = 1$ and $\mathbf{z} > \mathbf{0}$), it suffices to solve the following Karush–Kuhn–Tucker (KKT) system to obtain the optimal solution:

$$\frac{\partial L(\mathbf{z}, \boldsymbol{\mu}, \nu)}{\partial z_s} = c_s + \log z_s + 1 - \mu_s + \nu = 0 \quad \forall 1 \leq s \leq d, \tag{118}$$

$$\mathbf{1}^\top \mathbf{z} = 1, \tag{119}$$

$$\mathbf{z} \geq \mathbf{0}, \tag{120}$$

$$\boldsymbol{\mu} \geq \mathbf{0}, \tag{121}$$

$$\mu_s z_s = 0 \quad \forall 1 \leq s \leq d. \tag{122}$$

The first equation implies $z_s > 0 \; \forall s$, and thus in combination with the last, we obtain $\mu_s = 0 \; \forall s$. Therefore, the first equation becomes

$$z_s = \exp(-1 - \nu) \exp(-c_s) \; \forall s. \tag{123}$$

Summing up this result for all s, and taking into account the second equation, we obtain

$$\exp(-1 - \nu) = \frac{1}{\sum_{s=1}^{d} \exp(-c_s)}. \tag{124}$$

Combining (123) and (124) we obtain

$$z_s = \frac{\exp(-c_s)}{\sum_{t=1}^{d} \exp(-c_t)} \; \forall 1 \leq s \leq d. \tag{125}$$

In other words, $\mathbf{z} = \mathrm{softmax}(-\mathbf{c})$. □

Remark. It is clear that the per-iteration complexity of Entropic Frank-Wolfe is $\mathcal{O}(nd)$, which is the same as in vanilla Frank-Wolfe.

D.3 Projected gradient descent (PGD)

This algorithm consists in the following updates:

$$\mathbf{p}^k = \Pi_{\mathcal{X}}(\mathbf{x}^k - \nabla E(\mathbf{x})), \qquad \mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k(\mathbf{p}^k - \mathbf{x}^k), \tag{126}$$

where the stepsize α_k follows one of the schemes presented in §4.1. It is worth noting that this variant of PGD is eligible to exact line search (14). We observe in our experiments that using the line search scheme produces the same results as setting $\alpha_k = 1$. Thus we used this constant scheme for both training and prediction. The same applies to the Frank-Wolfe variants.

D.4 Fast proximal gradient method (PGM)

The original PGM [48] consists in updating

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\in\mathcal{X}}{\mathrm{argmin}} \left\{ \left\langle \nabla E(\mathbf{x}^k), \mathbf{x} \right\rangle + \frac{1}{2\alpha_k} \left\| \mathbf{x} - \mathbf{x}^k \right\|_2^2 \right\}, \tag{127}$$

which can be re-written as

$$\mathbf{x}^{k+1} = \Pi_{\mathcal{X}}(\mathbf{x}^k - \alpha_k \nabla E(\mathbf{x}^k)). \tag{128}$$

The above is precisely another variant of PGD (which is not eligible to exact line search (14)). While this algorithm is also supported by our implementation, the results presented in §5 are obtained using another variant called the *fast* PGM, also known as FISTA [9]. This algorithm consists in the following updates, where $\mathbf{y}^0 = \mathbf{x}^0 = \mathrm{softmax}(-\mathbf{u})$ and $t_0 = 1$:

$$\mathbf{x}^{k+1} = \Pi_{\mathcal{X}}(\mathbf{y}^k - \alpha_k \nabla E(\mathbf{y}^k)), \tag{129}$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}, \tag{130}$$

$$\mathbf{y}^{k+1} = \mathbf{x}^{k+1} + \frac{t_k - 1}{t_{k+1}}(\mathbf{x}^{k+1} - \mathbf{x}^k). \tag{131}$$

While the optimal value of α_k can be determined using *backtracking* [7], this process is very expensive as it requires evaluating the energy many times. Therefore, in practice, we use the constant scheme $\alpha_k = \alpha \in [0, 1]$. Doing a grid search on a random subset of 10 validation images, we found (again) that $\alpha_k = 1$ is the best, and thus it is used for all the experiments.

D.5 Entropic mirror descent (EMD)

Mirror descent (MD) [8, 55] is a generalization of PGM to a more general distance function. Each iteration of MD takes the following form:

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\in\mathcal{X}}{\mathrm{argmin}} \left\{ \left\langle \nabla E(\mathbf{x}^k), \mathbf{x} \right\rangle + \frac{1}{\alpha_k} B_\phi(\mathbf{x}, \mathbf{x}^k) \right\}, \tag{132}$$

where $\phi : \mathcal{X} \to \mathbb{R}$ is a convex and continuously differentiable function on the interior of $\mathcal{X}$, and $B_\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is its associated Bregman divergence, defined by

$$B_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - \left\langle \nabla\phi(\mathbf{y}), \mathbf{x} - \mathbf{y} \right\rangle. \tag{133}$$

Clearly, for $\phi(\mathbf{x}) = \frac{1}{2}\left\|\mathbf{x}\right\|_2^2$ we recover the PGM update (127). We provide an implementation for the so-called *entropic* variant of mirror descent [8], corresponding to choosing ϕ to be the negative entropy:

$$\phi(\mathbf{x}) = \sum_{i=1}^{n} \sum_{s=1}^{d} x_{is} \log x_{is}. \tag{134}$$

With this choice of ϕ, it is easy to check that the Bregman divergence (133) becomes the following so-called Kullback-Leibler divergence:

$$B_{\mathrm{KL}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \sum_{s=1}^{d} x_{is} \log \frac{x_{is}}{y_{is}}. \tag{135}$$

The MD update (132) thus becomes

$$\mathbf{x}^{k+1} = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{X}} \left\{ \langle \alpha_k \nabla E(\mathbf{x}^k) - \log \mathbf{x}^k, \mathbf{x} \rangle + \sum_{i=1}^{n} \sum_{s=1}^{d} x_{is} \log x_{is} \right\}, \tag{136}$$

where the $\log$ operation is taken element-wise. According to Lemma 6 (§D.2),we obtain

$$\mathbf{x}^{k+1} = \operatorname{softmax}\left(\log \mathbf{x}^k - \alpha_k \nabla E(\mathbf{x}^k)\right). \tag{137}$$

Let $\mathbf{g}^k$ denote the gradient $\nabla E(\mathbf{x}^k)$, the above reads

$$x_{is}^{k+1} = \frac{x_{is}^k \exp(-\alpha_k g_{is}^k)}{\sum_{t=1}^{d} x_{it}^k \exp(-\alpha_k g_{it}^k)} \quad \forall i \in \mathcal{V}, \forall s \in \mathcal{S}. \tag{138}$$

Numerically stable EMD In practice, the above expression of $\mathbf{x}^{k+1}$ may lead to numerical underflow or overflow. We overcome this by using the following modified iterate:

$$x_{is}^{k+1} = \frac{(x_{is}^k + \epsilon) \exp(-\alpha_k g_{is}^k + m_i^k)}{\sum_{t=1}^{d} (x_{it}^k + \epsilon) \exp(-\alpha_k g_{it}^k + m_i^k)} \quad \forall i \in \mathcal{V}, \forall s \in \mathcal{S}, \tag{139}$$

where $\epsilon = 10^{-10}$ and $m_i^k = \alpha_k \min_{1 \le s \le d} g_{is}^k \; \forall i \in \mathcal{V}$.

D.6 Alternating direction method of multipliers (ADMM)

The nonconvex ADMM for MAP inference [41] consists in the following updates, where $\mathbf{z}^0 = \operatorname{softmax}(-\mathbf{u})$ and $\mathbf{y}^0 = \mathbf{0}$:

$$\mathbf{x}^{k+1} = \Pi_{\mathcal{X}} \left(\mathbf{z}^k - \frac{1}{\rho}(\mathbf{y}^k + \frac{1}{2}\mathbf{P}\mathbf{z}^k + \mathbf{u}) \right), \tag{140}$$

$$\mathbf{z}^{k+1} = \Pi_{\mathcal{X}} \left(\mathbf{x}^{k+1} - \frac{1}{\rho}(-\mathbf{y}^k + \frac{1}{2}\mathbf{P}\mathbf{x}^{k+1}) \right), \tag{141}$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \rho(\mathbf{x}^{k+1} - \mathbf{z}^{k+1}). \tag{142}$$

We refer to the original paper [41] for more details. In our experiments, we set $\rho = 1$ for simplicity. Since the expensive computation $\mathbf{P}\mathbf{x}$ are done two times in each ADMM iteration (one in (140), another in (141)), this algorithm is roughly two times slower than the others. For a fair comparison, in our implementation we view (140) and (141) as two separate iterations (note that both $\mathbf{x}^{k+1}$ and $\mathbf{z}^{k+1}$ are feasible points).

Finally, we should note that the adaptive scheme for the penalty parameter ρ proposed by Lê-Huu and Paragios [41] is not applicable to our case, as we use only 5 iterations in our experiments (which is equivalent to only 2.5 regular iterations due to our above iteration separation).

E Detailed experimental setup and environment

E.1 CNN-CRF architectures

CNN-CRF Our segmentation model is a standard combination of a CNN and a CRF [76]. Given an input image $\mathbf{Z} \in \mathbb{R}^{H \times W \times 3}$, the CNN produces an output $\mathbf{Y} \in \mathbb{R}^{H \times W \times K}$ (where K is the number of object classes) called the *logits*, which is then fed into the CRF to produce a final output $\mathbf{X} \in \mathbb{R}^{H \times W \times K}$:

$$\mathbf{Y} = \mathrm{CNN}(\mathbf{Z}; \boldsymbol{\theta}^u), \quad \mathbf{X} = \mathrm{CRF}(\mathbf{Y}; \boldsymbol{\theta}^p), \tag{143}$$

where $\boldsymbol{\theta}^u$ and $\boldsymbol{\theta}^p$ are (typically trainable) parameters. The prediction is then obtained by taking the argmax along the last dimension of $\mathbf{X}$. For the CNN part, we consider two strong architectures: DeepLabv3 with ResNet101 backbone [16], and DeepLabv3+ with Xception65 backbone [17]. The reader is referred to the corresponding references for further details.

Dense CRF The CRF is defined over the input image such that each pixel is a node, and its labels are the object classes. Thus, using the notation defined in §2.1, we have $n = HW$, $d = K$, and $\mathcal{S} = \{1, 2, \dots, K\}$. The CRF produces $\mathbf{X}$ in (143) by minimizing the energy (6) with appropriately constructed potentials, and then simply reshaping the solution $\mathbf{x} \in \mathbb{R}^{HWK}$ into $H \times W \times K$. During training we skip the rounding step in CRF inference, so that the returned $\mathbf{x}$ is real-valued, which is more suitable for learning with the standard cross-entropy loss function. The unary potentials $\mathbf{u}$ is defined by to be the additive inverse of the logits $\mathbf{Y}$, reshaped correctly: $\mathbf{u} = -\operatorname{vec}(\mathbf{Y})$, where vec denotes the flattening operator. We use the fully-connected model introduced by Krähenbühl and Koltun [34] in which any pair of pixels (i, j) is an edge with a pairwise potential of the form $\theta_{ij}(s, t) = \mu(s, t)k(\mathbf{f}_i, \mathbf{f}_j)\ \forall s, t \in \mathcal{S}$, where $\mu : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is the so-called label compatibility function, and k is a Gaussian kernel over some image features $(\mathbf{f}_i, \mathbf{f}_j)$. For a pixel i, we use its position $\mathbf{p}_i \in \mathbb{N}^2$ and its color $\mathbf{c}_i \in [0, 255]^3$ as features, and define the kernel as

$$k(\mathbf{f}_i, \mathbf{f}_j) = w_1 \exp\left(-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|_2^2}{2\alpha^2} - \frac{\|\mathbf{c}_i - \mathbf{c}_j\|_2^2}{2\beta^2}\right) + w_2 \exp\left(-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|_2^2}{2\gamma^2}\right) \quad \forall i, j \in \mathcal{V}, \tag{144}$$

where w_1, w_2 are learnable kernel weights, and α, β, γ are hyperparameters. Following Zheng et al. [76], we use class-dependent kernel weights to increase the number of trainable parameters. Unlike Zheng et al. [76], for simplicity we use the default values $\alpha = 80, \beta = 13, \gamma = 3$ set by Krähenbühl and Koltun [34] in all experiments, instead of doing a cross validation to find the best values. Finally, we use the Potts compatibility function: $\mu(s, t) = w\mathbb{1}_{[s \neq t]}$ with $w = 1$ for the inference experiments in §5.2, and also for CRF initialization in the learning experiments in §5.3.

E.2 Datasets

We provide further details on the datasets. PASCAL VOC [22] contains 4369 images of 21 classes, split into 1464 (*train*), 1449 (*val*), and 1456 (*test*) image subsets. As a standard practice, we augment the dataset with images from Hariharan et al. [27], resulting in 10 582 training images (*trainaug*). Cityscapes [19] contains 5000 images of 19 classes, split into 2975 (*train*), 500 (*val*), and 1525 (*test*) image subsets. In addition, it also provides 19 998 coarsely annotated images (*train_extra*). We report the performance in terms of mIoU across the semantic classes (21 for PASCAL VOC and 19 for Cityscapes).

E.3 CNN training recipes

To fully train DeepLabv3 and DeepLabv3+, we follow closely the published recipes [16, 17] for this task. Below we present the most important information, and refer to the references for further details.

We first pretrain DeepLabv3 and DeepLabv3+ on the COCO [46] dataset (by selecting only the images that contain the classes defined in PASCAL VOC), and then finetune them on PASCAL VOC (*trainaug*) and Cityscapes (*train*). During training, we apply data augmentation by (randomly) left-right flipping, scaling the input images (from 0.5 to 2.0), and cropping (with crop size of 513×513 for PASCAL VOC and 769×769 for Cityscapes). We employ a *poly* learning rate schedule: $\eta_m = \eta_0 \left(1 - \frac{m}{M}\right)^p$, where η_0 is the initial learning rate, m is the step counter, and M is the total number of training steps. For all trainings, we set $p = 0.9$ and $\eta_0 = 0.001$ (except $\eta_0 = 0.01$ for pretraining on COCO), and a batch size of 16 images. The value of M is calculated from the number of training epochs, which we set to be 50 for COCO pretraining, 50 for finetuning on PASCAL VOC, and 500 for finetuning on Cityscapes. We should note some differences compared to the original papers [16, 17]. In particular, they did not specify the learning rate and number of steps for COCO pretraining. Furthermore, they used $\eta_0 = 0.0001$, which did not yield better results than $\eta_0 = 0.001$ in our implementation. Finally, in terms of the number of epochs, we used similar values to theirs. Indeed, they set 30 000 and 90 000 training steps for the 10 582 and 2975 training images of PASCAL VOC and Cityscapes, respectively. With a batch size of 16, these are equivalent to 45 and 484 epochs. Table 6 shows that our obtained results are similar to previous work [16, 17].

E.4 Training time and memory footprint

Our experiments are performed on a Linux server of 4 Nvidia V100 GPUs, using PyTorch 1.7. With a batch size of 16 on 4 GPUs (i.e., 4 images per GPU), both DeepLabv3 and DeepLabv3+ take

Dataset	Model	Published [16, 17]	Reproduced
VOC	DeepLabv3	78.51	81.83
	DeepLabv3+	82.45	82.89
Cityscapes	DeepLabv3	77.82	76.73
	DeepLabv3+	79.14	79.55

Table 6: Performance of our reproduced DeepLab models compared to the original papers [16, 17]. The mIoU scores are obtained on the *val* sets, without test time augmentation.

$\sim$7min/epoch on PASCAL VOC (with 513×513 crops). Thanks to our efficient GPU implementation (which will be made publicly available), plugging in the (5-step) CRF only increases that to $\sim$9 minutes (1.2–1.3$\times$ slower) for all inference methods (here we should note that CRF's running time is dominated by computing $\mathbf{Px}$ at each step, which is why the running is similar across the methods). In terms of memory usage, DeepLabv3+ takes $\sim$27.7GB (per GPU for 4 images) while DeepLabv3 takes $\sim$15.6GB. We found that the additional memory usage of the CRF (which has only 1323 trainable parameters) are negligible for the Frank-Wolfe variants as well as for PGD, while PGM and ADMM require an extra amount of $\sim$300MB (probably due to the additional storage of the variable $\mathbf{y}$ at each iteration, see §D).

F Additional results

F.1 Detailed results on the test sets

The detailed results on the test sets can be found on the corresponding submission websites whose URLs are given in Table 7.

Model	PASCAL VOC	Cityscapes
DeepLabv3+ [17]	87.8	82.1
DeepLabv3+ (this work)	87.6[1]	83.5[3]
DeepLabv3+ with ℓ_2FW CRF	**88.0**[2]	83.6[4]

[1] http://host.robots.ox.ac.uk:8080/anonymous/BUXULK.html

[2] http://host.robots.ox.ac.uk:8080/anonymous/YFJJLW.html

[3] https://www.cityscapes-dataset.com/anonymous-results/?id=845bd062fddae249ec0f4987d30f2f9be6e6716654513e7e6733d3f56e976532

[4] https://www.cityscapes-dataset.com/anonymous-results/?id=84e788da7c55eeeb4840b70407ed665006494c99e5d34e0bd8704d66d9c8b864

Table 7: Performance on the *test* sets.

F.2 Results for trainable α_k and λ

We carried out an experiment with ℓ_2FW and eFW ($\lambda = 0.7$) in which we allow the stepsize α_k at each CRF iteration to be learnable (initialized at 0.5). We observe the stepsizes at all the steps behave very similarly (i.e., increasing or decreasing together). In addition, for eFW they tend to increase during training, while for ℓ_2FW they tend to decrease. In addition, we also tried setting the regularization weight λ to trainable. We initialized it at 1.0 for ℓ_2FW and at 0.7 for eFW. For both solvers, we found that λ increased during training. Regarding accuracy, we did not observe significant differences compared to fixed α_k and fixed λ, although tuning the learning rates specifically for these variables could potentially lead to improved performance. See Table 8 for the details.

F.3 Results for fined-grained analysis

We randomly picked a trained checkpoint among the different runs for DeepLabv3+ with ℓ_2FW ($\lambda = 1.0$) and DeepLabv3+ with eFW ($\lambda = 0.7$), and evaluated them using 5, 10, and 25 CRF iterations on the PASCAL VOC validation set. The results are shown in Table 9.

F.4 Additional inference results

Regularizer	λ	Stepsize	mIoU
ℓ_2	1.0 fixed	1.0 fixed	0.8490489721
	1.0 fixed	0.5 fixed	0.849458456
	1.0 fixed	0.5 learnable	0.849185586
	1.0 learnable	0.5 learnable	0.8492224813
Entropy	0.7 fixed	1.0 fixed	0.8495011926
	0.7 fixed	0.5 fixed	0.8493972421
	0.7 fixed	0.5 learnable	0.849845171
	0.7 learnable	0.5 learnable	0.8491678238

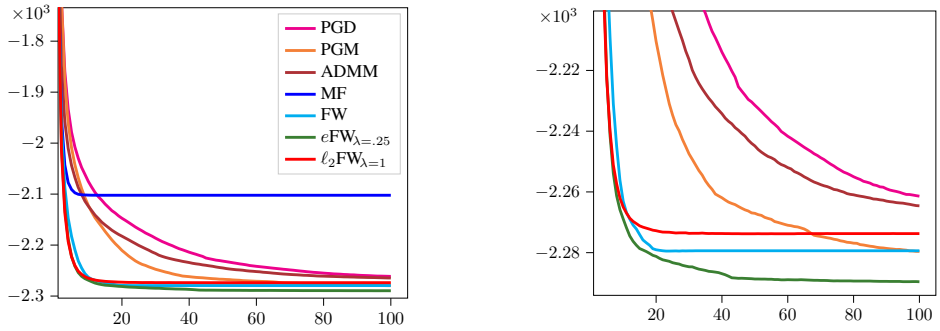
Table 8: Comparison between trainable and fixed λ and α_k

Method	Steps	mIoU	background	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	sofa	train	tvmonitor
CNN		82.89	95.79	91.80	44.89	89.92	71.49	83.54	94.68	91.54	95.42	52.36	95.51	70.25	93.63	93.08	88.27	90.20	68.03	92.62	66.95	92.33	78.45
ℓ_2FW	5	85.51	96.66	93.56	60.56	90.47	80.23	83.51	96.94	91.68	95.38	54.92	95.87	76.11	94.01	93.43	89.45	91.46	69.95	93.59	71.16	95.69	81.00
	10	85.52	96.67	93.56	60.49	90.47	80.23	83.53	96.92	91.68	95.38	55.11	95.87	76.16	94.00	93.41	89.43	91.45	69.98	93.57	71.39	95.67	80.95
	25	85.56	96.67	93.57	60.37	90.47	80.88	83.50	96.90	91.66	95.38	55.32	95.86	76.25	93.98	93.38	89.41	91.44	69.99	93.53	71.55	95.66	80.92
eFW	5	84.55	96.33	94.88	55.66	90.74	75.54	83.63	95.58	89.60	94.71	54.33	95.93	75.78	93.84	93.14	91.21	91.02	69.56	92.96	69.87	90.60	80.65
	10	84.60	96.34	94.88	55.66	90.73	76.53	83.63	95.58	89.58	94.70	54.33	95.97	75.81	93.84	93.14	91.22	91.02	69.54	93.02	69.89	90.60	80.66
	25	84.55	96.33	94.88	55.66	90.73	75.47	83.63	95.58	89.60	94.70	54.33	95.97	75.81	93.84	93.14	91.22	91.02	69.54	93.02	69.89	90.60	80.66

Table 9: Fined-grained results on PASCAL VOC validation set.

F.4.1 Results for longer inference regime

We show in Figure 2 a comparison of the discrete energy across the methods on a subset of 10 *val* images of PASCAL VOC for 100 inference iterations, using DeepLabv3+ and Potts dense CRF.



(a) Energy per CRF iteration.

(b) Zoomed version of the left figure.

Figure 2: CRF energy averaged over a subset of 10 *val* images of **PASCAL VOC** using DeepLabv3+ and Potts dense CRF.

From these results, we observe that:

1. Vanilla FW and ℓ_2FW already converge after around 20 iterations. ℓ_2FW does better than vanilla FW only in the early iterations.
2. PGM surpasses ℓ_2FW at after 70 iterations, and surpasses vanilla FW after 100 iterations.
3. PGD and ADMM are likely to surpass ℓ_2FW and vanilla, too, if given sufficient number of iterations, as these do not show any sign of convergence yet.

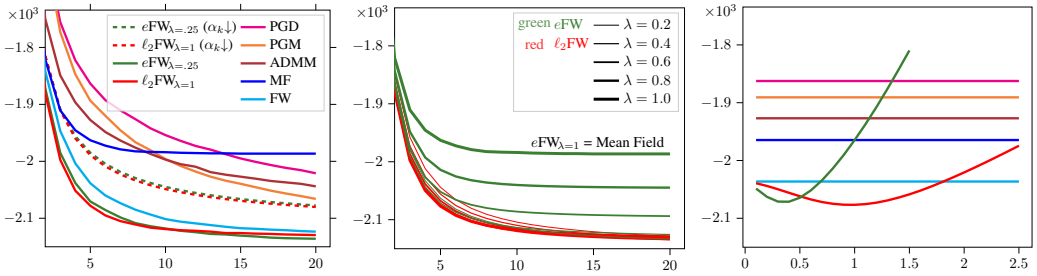
The main observation here is that, the relative performance of the methods are different between the early (typically first 10 iterations) and the later stage. In §6 we gave some hypotheses on why the proposed regularized Frank-Wolfe may work better than the others. Our main argument is that vanilla Frank-Wolfe is already much better than the other methods (in the first few iterations), and what we do is to equip it with the ability of effectively learning with SGD (potential improvements in terms of energy are rather a byproduct and not the main objective, as the improvements are sometimes small). Let us summarize this situation as follows:

1. Vanilla Frank-Wolfe outperforms other first-order methods such as PGD, PGM, and ADMM **during the first few iterations** (and may be surpassed at a later stage, as already shown).

2. For SGD learning, in which only a small number of iterations (due to the vanishing/exploding gradient problems, as already observed in previous work [76]), this behavior (reaching quickly a very low energy) of vanilla Frank-Wolfe is highly desirable.
3. Unfortunately, vanilla Frank-Wolfe iterates are piecewise constant and thus the resulting gradients are zero almost everywhere, which makes learning through backpropagation impossible.
4. Our regularized Frank-Wolfe is designed to precisely solve this zero-gradient issue.

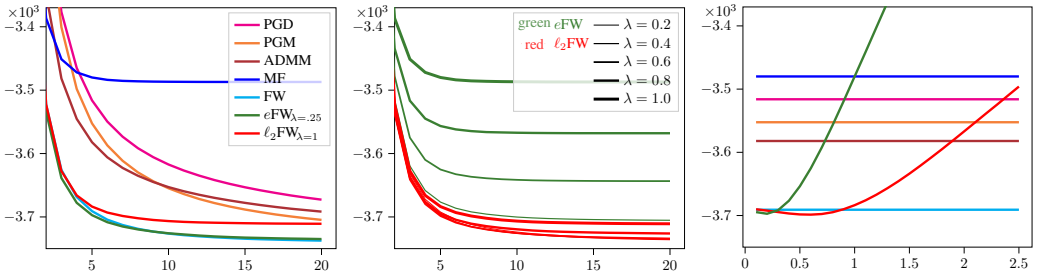
F.4.2 Additional inference results

Figures 3 and 4 show more results for the inference experiments.



(a) Energy per CRF iteration. **(b)** Energy per CRF iteration. **(c)** Per λ, at the 5^{th} CRF iteration.

Figure 3: CRF energy averaged over 1449 *val* images of **PASCAL VOC** using DeepLabv3+ and Potts dense CRF. **(a)** Comparison between regularized Frank-Wolfe and the other methods for some selected values of the regularization weight λ. **(b)** Results of regularized Frank-Wolfe for different values of λ. **(c)** Energy per λ after 5 iterations. Best viewed in color.



(a) Energy per CRF iteration. **(b)** Energy per CRF iteration. **(c)** Per λ, at the 5^{th} CRF iteration.

Figure 4: CRF energy averaged over 500 *val* images of **Cityscapes** using DeepLabv3+ and Potts dense CRF. **(a)** Comparison between regularized Frank-Wolfe and the other methods for some selected values of the regularization weight λ. **(b)** Results of regularized Frank-Wolfe for different values of λ. **(c)** Energy per λ after 5 iterations. Best viewed in color.