NEURAL FRIENDLY EXTENSIONS: TRAINING NEURAL NETWORKS WITH SET FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Integrating discrete computational steps into deep learning architectures is an important consideration when learning to reason over discrete objects. However, many tasks that involve discrete choices are defined via (combinatorial) set functions, and therefore pose challenges for end-to-end training. In this work, we explore a general framework for continuously and tractably extending such discrete functions that enables training via gradient-based optimization methods. Our neural friendly extension framework includes well-known constructions as special cases (such as the Lovász and multilinear extensions). Moreover, it facilitates the design of novel continuous extensions based on problem-specific considerations, including constraints. We demonstrate the versatility of our framework on tasks ranging from combinatorial optimization to image classification.

1 INTRODUCTION

End-to-end differentiability is a cornerstone of modern deep learning. Consequently, neural networks are still faced with significant obstacles when it comes to reliably performing elementary combinatorial tasks such as finding shortest paths on graphs and sorting numbers. Such tasks typically involve non-differentiable operations on sets of discrete objects, a feature that does not align well with the gradient-based optimization paradigm predominant in deep learning.

Various mechanisms for dealing with non-differentiable nodes in the computational graph of a neural network have been proposed. Previous work has intensively studied the problem of backpropagation through stochastic operations, as well as through discrete deterministic operations of specific forms. The former include methods for differentiable sampling (Jang et al., 2017; Maddison et al., 2017), while the latter include differentiable versions of sorting (Blondel et al., 2020), QP and SDP solvers (Amos & Kolter, 2017; Wang et al., 2019), and shortest path algorithms (Pogančić et al., 2019; Berthet et al., 2020).

Differently, this work focuses on back-propagating through black-box set functions—general discrete deterministic functions that map sets of discrete entities onto real numbers. The class of set functions is rather general: it includes functions that are ubiquitous in machine learning applications like the argmax, entropy, and rank, as well as functions that are routinely used to describe structural properties like coloring numbers, covers, and cuts in a graph (Boros & Hammer, 2002). Submodular set functions have found many further applications in machine learning (Bach, 2013; Bilmes, 2013; Krause & Jegelka, 2013). Set functions can also be viewed as atomic blocks that may be composed to perform complex algorithmic computations (Veličković & Blundell, 2021), and could be used at intermediate stages of a model or be incorporated directly in the objective function. Perhaps the most well known solution for discrete function optimization is REINFORCE (Williams, 1992), also known as the score function estimator, which allows for an unbiased gradient estimate. In practice, the score function estimator is hard to train as it suffers from high variance. This has been the impetus for works that aim to reduce variance through control variates, also known in reinforcement learning as baselines (Gu et al., 2017; Liu et al., 2018; Grathwohl et al., 2018; Wu et al., 2018; Cheng et al., 2020).

We propose *neural friendly extensions* (NFEs), a general framework for designing differentiable extensions of black-box set functions. NFEs are based on an axiomatic approach meant to ensure their appropriateness for neural network

training: 1) they can be computed efficiently in closed-form, 2) their minima correspond to the minima of the discrete function, and 3) any continuous point can be discretized without increasing the cost function.

Defining and using NFEs is simple: the main idea is to interpret the continuous output of a neural network as the expectation of a discrete distribution supported on a small number of well chosen sets. The NFE at that point is then equal to the expected value of the discrete function under the same distribution. The aforementioned procedure provides a principled recipe for deriving differentiable extensions based on problem-specific considerations (such as simple constraints) by selecting an appropriate distribution. Intriguingly, it also includes as special cases well-known extensions of discrete functions, such as the Lovász (Lovász, 1983) and multilinear extensions (Calinescu et al., 2011).

We demonstrate that NFEs can be used to train models on a variety of tasks that involve discrete computational steps. We find that NFEs provide a competitive alternative when it comes to extending and optimizing set functions arising in settings including combinatorial optimization and even standard image classification problems. Furthermore, we show how the choice of a suitable support for the distribution of our relaxation can yield improved performance by better aligning the extension with the structure of the problem.

2 Related work

The subject of differentiation through discrete modules in neural architectures has witnessed remarkable development in recent years. While our work falls under the same broad category, an important distinction from prior work is our focus on enabling the extension and differentiability of *arbitrary set functions*, instead of focusing on a specific operation like sampling or sorting.

Differentiating through black-box functions. The score function estimator is an established approach when optimizing black-box functions. There are two key differences between the score estimator and our framework: 1) The score estimator enables differentiation by calculating a (high variance) gradient estimate from samples of a learned distribution using the log derivative trick; the only necessary condition is that the probabilities of the distribution are differentiable with respect to the model parameters. In our framework, the extension is entirely deterministic and avoids the introduction of stochastic nodes in the computation graph. 2) Our extension provides evaluations of the function at *continuous* points, which could be then combined in a differentiable manner with other operations in the forward pass, while the score estimator is restricted to the original domain of the function.

In the context of neural combinatorial optimization, an unsupervised framework for differentiable relaxations of combinatorial objective functions has been recently proposed (Karalias & Loukas, 2020). The distributional assumptions of the proposed instantiation of that framework correspond to the multilinear extension (Calinescu et al., 2011), and the proposed relaxations can be employed in the case where a closed form expectation can be derived and efficiently computed. In contrast to that, our extensions can, in principle, be designed *without imposing strong conditions* on the set function and therefore can be used to train directly with a discrete objective.

Differentiating through discrete non-differentiable operations. Research on this subject has focused on developing gradient estimation techniques that enable backpropagation through specific non-differentiable operations. Those may appear at several points in a learning pipeline, either in the loss function or as intermediate steps in the forward pass. Sorting and ranking have been at the forefront of those developments and efficient differentiable versions have already been designed (Blondel et al., 2020; Grover et al., 2018; Xie et al., 2020). Sampling is another operation that has attracted a lot of attention, as it introduces stochastic nodes in the computation graph which are non differentiable. A straightforward way to circumvent the problem is the Straight-Through Estimator (Bengio et al., 2013), which treats the sampling operation as an identity map in the backward pass to yield a biased estimate of the gradient. Sampling has also been the motivation behind the use of the Gumbel-Softmax trick (Maddison et al., 2017; Jang et al., 2017), which allows differentiable sampling from discrete (categorical) distributions. The Gumbel trick has since been extended to enable sampling subsets instead of one-hot vectors (Xie & Ermon, 2019) and sequences without replacement (Kool et al., 2019). Gumbel-Softmax can be seen as a special case of the more general framework, Stochastic Softmax Tricks

(SST) (Paulus et al., 2020), which enables unbiased gradient estimation with differentiable samples that are obtained as solutions to a regularized convex program. A crucial difference between gradient estimation with SST and our approach is that the former address the non-differentiability of the samples that are used as input to (loss) functions, while our framework addresses *the non-differentiability and the lack of continuity of the functions themselves*. Indeed, unbiased gradient estimation in the case of SST is achieved under the assumption that the function operating on the samples is continuous and differentiable.

SST in general can be understood from the perspective of perturbation models (Papandreou & Yuille, 2011; Hazan & Jaakkola, 2012), where samples are drawn from a distribution by optimizing a random objective. Building on the concept of perturbation models, recent work (Berthet et al., 2020) has proposed a way to backpropagate through black-box solvers. This is one of several important papers in that direction, which aims to integrate solvers as differentiable layers in neural architectures (Amos & Kolter, 2017; Agrawal et al., 2019; Pogančić et al., 2019; Wang et al., 2019; Paulus et al., 2021). While the above works on solver integration can allow for differentiable optimization of discrete functions, they do not provide a recipe on how to compute their values at continuous points, and they generally rely on specific assumptions about the function (e.g., linear costs) or the problem formulation (e.g., SDP or QP). Instead, our framework has minimal requirements for the set functions that are being extended.

While works on gradient estimation through differentiable sampling and solver integration are different in both methodology and goals from ours, we emphasize that our continuous extension framework is *compatible* with such approaches. For instance, one could obtain soft samples with a stochastic softmax trick which could then be evaluated with an extension of a set function. We believe there is ample opportunity for creative combinations of those ideas.

3 DIFFERENTIATING THROUGH SET FUNCTIONS

The common practice of training via backpropagation makes it challenging to incorporate functions defined on discrete spaces into deep networks. This section proposes a framework for differentiating through one of the most fundamental classes of discrete functions capable of modelling collections of objects: functions whose input domain is a set. Importantly our framework applies to functions that are *only* defined at discrete values by providing a method for extending set functions onto continuous domains.

Let $\Omega \subseteq 2^{[d]}$ denote a family of subsets of a ground set $[d] = \{1, \dots, d\}$, where each $S \in \Omega$ is associated with the binary vector $\mathbf{1}_S \in \{0,1\}^d$ whose ith entry is one if $i \in S$, and zero otherwise. We consider the problem of minimizing functions

$$f : \Omega \to \mathbb{R} \quad \text{with} \quad \Omega \subseteq 2^{[d]},$$

where $f(S)$ is ill-defined whenever $S \notin \Omega$. Without loss of generality we will always assume that f is centered with $f(\varnothing) = 0$. Such set functions appear in a wide variety of problems, including the important case of selecting nodes or edges of a graph, e.g., to find a maximum cover or a shortest path.

Due to the discrete topology of the domain Ω, the definition of the derivative of $f : \Omega \to \mathbb{R}$ cannot even be formulated. To resolve this, we propose a method for constructing a new function $\mathfrak{F} : [0,1]^d \to \mathbb{R}$ that admits continuous optimization.

3.1 WHAT PROPERTIES SHOULD A NEURAL FRIENDLY EXTENSION $\mathfrak{F}$ OF A SET FUNCTION f POSSESS?

In order for optimizing $\mathfrak{F}$ to be a useful proxy for optimizing f as part of a deep network, it is not sufficient for $\mathfrak{F}$ to merely be an *extension* of f (i.e., to agree with f on all $\mathbf{x} \in \Omega$). Without additional constraints on $\mathfrak{F}$, an arbitrary extension may introduce spurious minima which do not correspond to good solutions to f. Equally fundamentally, an arbitrary extension may not be continuous or differentiable, which are needed to allow gradient-based optimization.

We argue that the following properties are essential when training $\mathfrak{F}$ as a proxy for f:

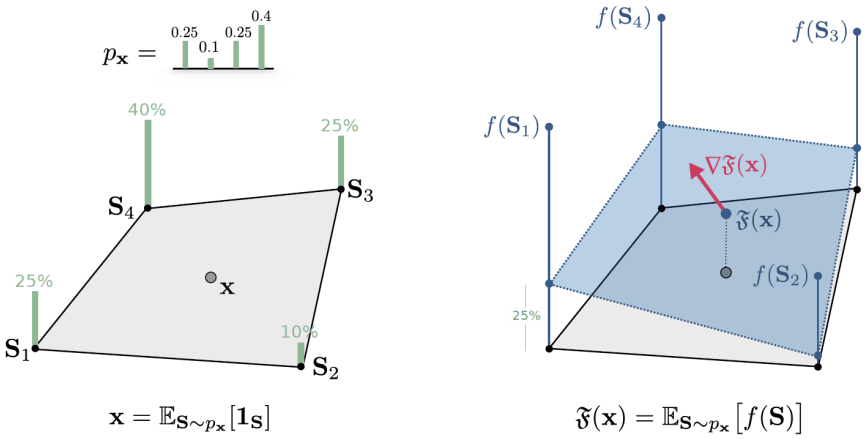

Figure 1: **Neural friendly extensions:** $f(\mathbf{x})$ is not defined for non-integral points $\mathbf{x}$. To assign a function value to non-integral $\mathbf{x}$ we first reinterpret $\mathbf{x}$ as an expectation over integral points $\mathbf{1}_S$ in the domain Ω of f. Then, in place of the ill-defined quantity $f(\mathbf{x}) = f(\mathbb{E}_{S\sim p_{\mathbf{x}}}[\mathbf{1}_S])$ we obtain a well-defined value $\mathfrak{F}(\mathbf{x}) \coloneqq \mathbb{E}_{S\sim p_{\mathbf{x}}}[f(S)]$ by exchanging the order of operation of f and the expectation so that f is only evaluated at points S in its domain Ω.

P1. (Extension) $\mathfrak{F}(\mathbf{1}_S) = f(S)$ for all $S \in \Omega$.

P2. (Minimality) The minima of $\mathfrak{F}$ belong to the convex hull of the minima of f.

P3. (Continuity) $\mathfrak{F}$ is Lipschitz continuous (and hence almost everywhere differentiable).

P4. (Rounding) Given an $\mathbf{x} \in [0,1]^d$, we can efficiently recover an $S \in \Omega$ for which $f(S) \leq \mathfrak{F}(\mathbf{x})$.

P1 requires that $\mathfrak{F}$ is a functional extension of f. P2 suggests that by minimizing $\mathfrak{F}$ we should expect to recover good solutions to f. P3 enables gradient-based optimization. Finally, P4 asks that, starting from a fractional point in $[0,1]^d$, we can easily recover a discrete solution that is at least as good. This last point is important since downstream tasks often require *exact* discrete values (e.g. computing the size of a clique can only be done for a discrete set of nodes).

3.2 Neural Friendly Extensions

We propose a general framework for defining extensions via convex combinations of f evaluated at discrete points. Phrased probabilistically, we define $\mathfrak{F}$ to be the expected value of f over a distribution of points in Ω.

Definition (Proper NFE)**.** We call $\mathfrak{F}$ a proper NFE of $f : \Omega \to \mathbb{R}$ if and only if

$$\mathfrak{F}(\mathbf{x}) \coloneqq \mathbb{E}_{S\sim p_{\mathbf{x}}}\big[f(S)\big] = \sum_{S\in\Omega} p_{\mathbf{x}}(S) f(S) \quad \text{and} \quad \mathbf{x} = \mathbb{E}_{S\sim p_{\mathbf{x}}}[\mathbf{1}_S] \quad \text{for all} \quad \mathbf{x} \in \text{Hull}(\Omega), \tag{1}$$

where $p_{\mathbf{x}}$ is any distribution supported on Ω and $\text{Hull}(\Omega)$ denotes the convex hull of points $\mathbf{1}_S$ for $S \in \Omega$. If there is an $\mathbf{x}$ for which $\mathbf{x} \neq \mathbb{E}_{S\sim p_{\mathbf{x}}}\big[\mathbf{1}_S\big]$, we call $\mathfrak{F}$ an *improper* extension.

The requirement that $\mathbf{x} = \mathbb{E}_{S\sim p_{\mathbf{x}}}\big[\mathbf{1}_S\big]$ ensures that proper extensions $\mathfrak{F}$ do not introduce bad new minima (see P2 on minimality). Thus, to extend a set function f at some continuous point $\mathbf{x}$ one needs to identify a distribution $p_{\mathbf{x}}$ over Ω whose expectation is $\mathbf{x}$. Figure 1 provides a toy example of a proper extension in which $p_{\mathbf{x}}$ is a categorical distribution supported over k sets such that $\mathbf{x} = \sum_{i=1}^{k} p_{\mathbf{x}}(S_i)\mathbf{1}_{S_i}$, which exists whenever $\mathbf{x}$ lies in the convex hull of $\{\mathbf{1}_{S_i}\}_{i=1}^{k}$. The function $\mathfrak{F}$ and its gradient, if it exists, are, respectively, given by

$$\mathfrak{F}(\mathbf{x}) = \sum\nolimits_{i=1}^{k} p_{\mathbf{x}}(S_i) f(S_i) \quad \text{and} \quad \nabla_{\mathbf{x}}\mathfrak{F}(\mathbf{x}) = \sum\nolimits_{i=1}^{k} f(S_i)\, \nabla_{\mathbf{x}} p_{\mathbf{x}}(S_i).$$

Notice that, although f does not have gradients, $\mathfrak{F}$ does whenever $\nabla_{\mathbf{x}} p_{\mathbf{x}}$ exists. Crucially, the computation of both quantities can be completed using k calls to the black-box discrete function f, which ensures that the extension remains tractable whenever the support of $p_{\mathbf{x}}$ is small.

In the following, we provide sufficient conditions on $p_{\mathbf{x}}$ under which $\mathfrak{F}$ satisfies properties P1-4.

P1. (*Extension*) The property that $\mathfrak{F}$ is a mathematical extension of f, can easily be enforced by requiring that for all $S \in \Omega$ we have $p_{\mathbf{1}_S} = \delta_S$ (the point mass at S).

P2. (*Minimality*) The minimality property follows from the definition of a proper NFE without additional assumptions on $p_{\mathbf{x}}$:

Proposition 1. Consider $f : \Omega \to \mathbb{R}$ such that $\min_{S \in \Omega} f(S) < 0$. The following statements hold:

1. $\min_{\mathbf{x} \in \text{Hull}(\Omega)} \mathfrak{F}(\mathbf{x}) = \min_{S \in \Omega} f(S)$
2. $\arg\min_{\mathbf{x} \in \text{Hull}(\Omega)} \mathfrak{F}(\mathbf{x}) \subseteq \text{Hull}(\arg\min_{\mathbf{1}_S : S \in \Omega} f(S))$

The assumption on f is mild. Recalling that we always assume that $f(\varnothing) = 0$ the assumption merely asserts that the minimization problem does not have the trivial solution $S = \varnothing$.

Both parts of this result can be understood intuitively. First, $\mathfrak{F}$ is defined as a convex combination of points $f(S)$ with $S \in \Omega$, and so cannot attain a smaller value than the smallest $f(S)$. Second, if there are multiple distinct minima $\{S_1, \ldots, S_m\}$ of f, then a linear combination of these points also attains the minimum. One can then show that since $\mathbf{x} = \mathbb{E}_{S \sim p_{\mathbf{x}}}[\mathbf{1}_S]$, any minima of $\mathfrak{F}$ must be a convex combination of minima of f (see Appendix A for proof).

P3. (*Continuity*) If $\mathfrak{F}$ is Lipschitz continuous, then $\mathfrak{F}$ is differentiable almost everywhere (Rademacher's theorem), enabling the optimization of $\mathfrak{F}$ using first-order gradient methods. It is straightforward to deduce that $\mathfrak{F}$ is Lipschitz continuous (and hence P3 holds) whenever the mapping $\mathbf{x} \mapsto p_{\mathbf{x}}(S)$ is itself Lipschitz continuous for all $S \in \Omega$ and $\sum_S f(S)$ is finite. For completeness, we include a proof in Appendix A (Lemma 1).

P4. (*Rounding*) Our framework provides a simple way to recover exact discrete solutions whose function value are at least as good as the originally obtained fractional solution $\mathfrak{F}(\mathbf{x})$. Specifically, given solution $\mathbf{x} \in [0, 1]^d$, simply return an S that attains the minimum of f over the support of $p_{\mathbf{x}}$. That is, any S solving $\min_{S : p_{\mathbf{x}}(S) > 0} f(S)$. Such an S is guaranteed to yield a solution at least as good as $\mathbf{x}$ since $\mathfrak{F}(\mathbf{x})$ is an expectation over $p_{\mathbf{x}}$. This minimization can be done efficiently so long as the cardinality of the support of $p_{\mathbf{x}}$ is not too large, as is typically the case for the examples given in Section 3.3. If the support is large, drawing samples from $p_{\mathbf{x}}$ yields a discrete solution at least as good as $\mathbf{x}$ with high probability.

3.3 Examples of neural friendly extensions

Next, we demonstrate the generality of our framework by illustrating how it captures several interesting extensions.

Lovász extension. The Lovász extension or Choquet integral (Choquet, 1954) is a widely known continuous extension. While it is defined for *any* set function, it has been particularly impactful in the context of submodular set functions: a central result by Lovász (1983) states that this extension is convex if and only if f is submodular. This insight allows to reduce submodular minimization to convex optimization, forming the basis for the first polynomial-time submodular minimization algorithm (Grötschel et al., 1981).

The Lovász extension is a special case of our framework. For a point $\mathbf{x} \in [0, 1]^d$, suppose w.l.o.g. that $\mathbf{x}$ is sorted so that $x_1 \geq x_2 \geq \ldots \geq x_d$. Then the Lovász extension is defined as

$$\mathfrak{F}(\mathbf{x}) = \sum_{i=1}^{d} (x_i - x_{i+1}) f(S_i), \qquad \text{(Lovász extension)}$$

where $S_i = \{1, \ldots, i\}$ is the set of the indices of the i largest coordinates of $\mathbf{x}$.

To formulate the Lovász extension as a NFE we set $p_{\mathbf{x}}(S_i) = x_i - x_{i+1}$ for each $i \in [d]$ (we take $x_{d+1} = 0$) and assign all remaining probability mass to the empty set $p_{\mathbf{x}}(\varnothing) = 1 - x_1$. So $p_{\mathbf{x}}(S)$ is supported on $\{\varnothing\} \cup \{S_i\}_{i=1}^{d}$, i.e., the level sets of $\mathbf{x}$, and satisfies $\mathbf{x} = \mathbb{E}_{S \sim p_{\mathbf{x}}}[\mathbf{1}_S]$. Properties P2 and P4 follow automatically from the fact that it is a proper NFE. Similarly, it is easy to see that $p_{\mathbf{1}_S} = \delta_S$ for all $S \in \Omega$ and so P1 also holds. Using Lemma 1, we can also prove that P3 holds. We refer the reader to Appendix A.1.1 for the details.

Singleton extension. Consider an arbitrary set function $f : \Omega \to \mathbb{R}$, where $\Omega = \{E_1, \ldots, E_d\}$ contains only sets $E_i = \{i\}$ with cardinality one. Functions operating on single items may arise, e.g., when taking an $\arg\max$ over several values. To define $p_{\mathbf{x}}(S)$, we again assume that $x_1 \geq x_2 \geq \ldots \geq x_d$ and then select $S_i = \{i\}$ for all $i \in [d]$. We then define our distribution as $p_{\mathbf{x}}(S_i) = x_i - x_{i+1}$ with $x_{d+1} = 0$ and assign all remaining probability mass to the empty set $p_{\mathbf{x}}(\varnothing) = 1 - x_1$. It is easy to see that P1 holds since $p_{\mathbf{1}_S} = \delta_S$ for all $S \in \Omega$, and P4 holds since $p_{\mathbf{x}}$ is supported on d sets. P3 follows as a consequence of P3 holding for the Lovász extension. Finally, we show in Appendix A.1.2 that although $\mathfrak{F}$ is improper, P2 still holds.

Fixed cardinality extension. It is often necessary to search over subsets of a particular size (e.g., when deciding if a graph has a k-clique). In this case it is natural to define a $p_{\mathbf{x}}$ that is supported on sets of a fixed size k. Consider the following construction: for $1 \leq i \leq n - k$, let $S_i = \{i, i + 1, \ldots, k + i\}$ and take the corresponding coefficient to be $p_{\mathbf{x}}(S_i) = x_{i+k-1} - x_{i+k}$ where as before we assume that $\mathbf{x}$ is sorted high to low and that $x_{d+1} = 0$. P1 and P4 both hold. However, although this construction yields an efficiently computable expression for $\mathfrak{F}$, it fails to satisfy $\mathbf{x} = \mathbb{E}_{S \sim p_{\mathbf{x}}}[\mathbf{1}_S]$ and is therefore an improper NFE. In Section 3.4 we will propose a general procedure for remedying this by carefully modifying $p_{\mathbf{x}}$ (and f) in order to create a proper NFE for which P2 holds. Finally, P3 fails to hold in this case, although it satisfies the weaker property of piecewise linearity.

Multilinear extension. The multilinear extension is another well known tool for set function optimization, including submodular functions (Calinescu et al., 2011). The extension takes the form

$$\mathfrak{F}(\mathbf{x}) \coloneqq \sum_{S \subseteq [d]} f(S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i). \qquad \text{(multilinear extension)}$$

This fits our framework by fixing $p_{\mathbf{x}}(S) = \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i)$, i.e., a product distribution over items. It is easy to see that $\mathbf{x} = \mathbb{E}_{S \sim p_{\mathbf{x}}}[\mathbf{1}_S]$ and so P2 holds. It is also apparent that P1 holds, and that $p_{\mathbf{x}}$ is Lipschitz continuous, implying that the corresponding $\mathfrak{F}$ is also Lipschitz continuous (P3). The challenge of the multilinear extension is the complexity of computing $\mathfrak{F}$ that naively entails summing over all subsets $S \subseteq 2^{[d]}$. Still, several functions of interest admit an efficient computation. For instance, the graph cut, set cover, and facility location functions can be extended in $\widetilde{\mathcal{O}}(d^2)$ time (Iyer et al., 2014). More generally, any f that is a $\mathcal{O}(1)$ degree polynomial can be extended in polynomial time. For linear f, both the multilinear and Lovász extension become linear functions.

3.4 Constructing proper NFEs from improper ones

An NFE is described as improper if there is an $\mathbf{x}$ for which $\mathbf{x} \neq \mathbb{E}_{S \sim p_{\mathbf{x}}}[\mathbf{1}_S]$ and called a proper NFE if equality holds always. Proper NFEs are preferred since Proposition 1 ensures that P2 holds: that $\mathfrak{F}$ does not introduce any new bad minima. However, there may still be instances where one wishes to define $\mathfrak{F}$ using an improper NFEs that have other desired properties. One example is the construction given in Section 3.3 of an extension whose distribution is supported on sets of fixed cardinality. Although this is improper, it has the distinction of searching over sets of a fixed size, and so may still be of interest for problems for which we are searching for sets of a particular size.

In this section our goal is, given an improper $\mathfrak{F}$ defined by $p_{\mathbf{x}}$, to provide a principled way of constructing a similar *proper* NFE $\mathfrak{F}'$ that can be optimized in place of $\mathfrak{F}$, with the important benefit of knowing that $\mathfrak{F}'$ does not behave badly by introducing bad minima (i.e., $\mathfrak{F}'$ satisfies P2). The forthcoming proposition states the result, which for each point $\mathbf{x}$ where $\mathbf{x} \neq \mathbb{E}_{S \sim p_{\mathbf{x}}}[\mathbf{1}_S]$, constructs a proper $\mathfrak{F}'$ by adding a carefully designed penalty to the improper $\mathfrak{F}$.

	Max-Cut			Max-Clique		
	ENZYMES	PROTEINS	IMDB-Binary	ENZYMES	PROTEINS	IMDB-Binary
Straight-through	$0.877_{\pm 0.084}$	$0.904_{\pm 0.174}$	$0.531_{\pm 0.253}$	$0.556_{\pm 0.128}$	$0.309_{\pm 0.443}$	$0.795_{\pm 0.354}$
Erdős penalty loss	$0.946_{\pm 0.032}$	$0.930_{\pm 0.063}$	$0.992_{\pm 0.019}$	$0.837_{\pm 0.172}$	$0.900_{\pm 0.139}$	$0.893_{\pm 0.149}$
Greedy alg.	$0.969_{\pm 0.025}$	$0.977_{\pm 0.034}$	$1.000_{\pm 0.000}$	$0.973_{\pm 0.076}$	$0.978_{\pm 0.072}$	$0.950_{\pm 0.071}$
Lovasz NFE	$0.968_{\pm 0.200}$	$0.971_{\pm 0.021}$	$0.959_{\pm 0.048}$	$0.720_{\pm 0.247}$	$0.875_{\pm 0.125}$	$0.903_{\pm 0.214}$

Table 1: **Unsupervised combinatorial optimization**: Approximation ratios for combinatorial problems. Values closer to 1 are better.

Proposition 2. Let $p_{\mathbf{x}}$ be a distribution for which $\mathbf{x}' \coloneqq \mathbb{E}_{S\sim p_{\mathbf{x}}}[\mathbf{1}_S] \neq \mathbf{x}$ and define $w \coloneqq \min\left(\{x_i/x'_i\}_{i=1}^d \cup \{1\}\right)$. There exists a perturbed distribution $p'_{\mathbf{x}}(S,z)$ with $p'_{\mathbf{x}}(S,1) = p_{\mathbf{x}}(S)$ such that

$$\mathfrak{F}'(\mathbf{x}) \coloneqq \mathbb{E}_{(S,z)\sim p'_{\mathbf{x}}}\big[f(S,z)\big], \quad \text{with} \quad f(S,z) \coloneqq \begin{cases} f(S) & \text{if } z=1 \\ \beta & \text{if } z=0, \end{cases} \quad \text{and} \quad z \sim \text{Bernoulli}(w)$$

is a proper NFE.

Note that w measures the closeness of $p'_{\mathbf{x}}$ to $p_{\mathbf{x}}$ and as $\mathbf{x}' \to \mathbf{x}$ we have $w \to 1$ and so $p'_{\mathbf{x}} \to p_{\mathbf{x}}$ in distribution. In order words as $\mathbf{x}' \to \mathbf{x}$ we recover the original extension $\mathfrak{F}(\mathbf{x})$. Although the construction introduces a "perturbed" distribution $p'_{\mathbf{x}}$, that approximates $p_{\mathbf{x}}$, the distribution $p'_{\mathbf{x}}$ does not feature at all in the computations required to compute $\mathfrak{F}'$ in practice. This is because $\mathfrak{F}'$ can be efficiently computed using $\mathfrak{F}$ by the law of total expectation: $\mathfrak{F}'(\mathbf{x}) = w\mathfrak{F}(\mathbf{x}) + (1-w)\beta$. So computationally evaluating $\mathfrak{F}'$ amounts to adding a penalization term to $\mathfrak{F}$ that punishes more when $(1-w)$ is large. This occurs precisely when $\mathbf{x}'$ and $\mathbf{x}$ are far apart, and so the proper extension $\mathfrak{F}'$ can also be viewed as encouraging the improper extension $\mathfrak{F}$ to search over $\mathbf{x}$ for which $\mathbf{x} \approx \mathbb{E}_{S\sim p_{\mathbf{x}}}[\mathbf{1}_S]$.

We experimentally validate the practical usefulness of this principled and computational efficient technique in Section 4.2 where we apply it to convert the improper k-subset extension into a proper extension. We show that this proper extension performs well at the task of finding k-cliques.

4 Experiments

4.1 Unsupervised combinatorial optimization

Set functions commonly appear in combinatorial objectives, as they are essential in describing properties of discrete structures like graphs and sets. Here, we leverage continuous extensions to formulate differentiable *unsupervised* loss functions for combinatorial problems on graphs. We test on two problems, the maximum clique (constrained) and the maximum cut (unconstrained). Our models are trained in a *minimal setting* in order to emphasize the contribution of the loss function and are not meant to establish state of the art results. We compare with two other neural approaches. The unsupervised relaxations from Karalias & Loukas (2020) and the Straight-Through estimator (Bengio et al., 2013). We use the Lovász extension as an instantiation of our framework. As a sanity check, we also list the performance of a non-neural greedy heuristic.

In all cases except for the greedy heuristic, given an input graph $G = (V, E)$, a neural network is trained to produce a vector $\mathbf{x} \in [0,1]^{|V|}$. For the maximum cut problem, we use the Lovaśz extension to train the network to minimize the negative cut function $f(S;G) = -\text{Cut}(S;G)$, where S are subsets of the graph G. For the Erdős framework, we derive a continuous relaxation by calculating the expected value of the discrete objective. For the Straight-Through

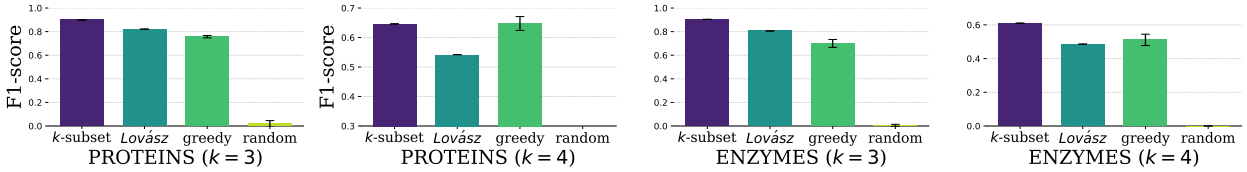

Figure 2: **Learning to find k-cliques:** higher F1-score is better. The k-subset extension, which defines $\mathfrak{F}$ as a convex combination of sets of size k, is better aligned with the task and significantly improves over the Lovász extension.

estimator, we sample from the distribution of level sets of $\mathbf{x}$ using random thresholds and average the evaluations of the objective on those. Then we backpropagate through the sampling operation by treating it as the identity mapping.

For the maximum clique problem, the Lovaśz extension is trained directly with a function that incorporates the constraint as a multiplicative term. For the Erdős framework, we use the loss that the authors provided in the paper. For the Straight-Through, we use the same discrete function as in the Lovaśz extension.

We train with a 60-20-20 split and display the test performance of the models that achieved the best validation accuracy. We report the mean approximation ratio across the test set along with the standard deviation. We train modern graph neural network architectures on real world datasets with graph sizes of up to a few hundred nodes. For details of the setup, we refer the reader to Appendix B.1.

Results. The results reported on Table 1 suggest that the Lovaśz extension indeed provides a way to directly minimize a set function. It is able to outperform the Erdős probabilistic loss (Karalias & Loukas, 2020) on two datasets on max-cut and is competitive on max-clique. On the other hand, the Straight-Through estimator delivers inconsistent results, especially on max-clique where an objective with constraints is optimized.

4.2 Constraint satisfaction problems

Constraint satisfaction problems are an important class of widely studied combinatorial problems (Kumar, 1992; Cappart et al., 2021). In this section we study the applicability of our framework to one such problem: the detection of k-cliques. Given a graph $G = (V, E)$, the goal is to determine if it contains a clique of size k or more.

We demonstrate the adaptability of our framework by building the cardinality into $\mathfrak{F}$. Specifically, we take Ω to be the family of subsets of the node set V of size exactly k, and use the fixed set size extension discussed in Section 3.3. The goal is to learn to output `true` if we successfully find a k-clique, and `false` otherwise. We compare this approach to the Lovász extension, which searches over sets of different cardinalities, and two non-neural baselines: 1) Greedily constructing a set S, starting with $S = \varnothing$, iterating over all nodes i and updating $S \leftarrow S \cup \{i\}$ for the the first i for which $S \cup \{i\}$ is a clique (termination when no such i exists). We return `true` if the terminal set S is a k-clique and `false` otherwise. 2) Randomly generating 1000 sets of size k and returning `true` if any of these sets is a k-clique, and `false` otherwise.

Results. Figure 6 shows that by specifically searching over sets of size k using the fixed set size extension we are able to significantly improve the performance compared to the Lovász extension. While the Lovász extension performs broadly comparably to the greedy baseline, the k-subset extension shows a clear improvement over both. Meanwhile, the random sampling baseline fails in all settings, confirming the non-triviality of the tasks.

4.3 Directly minimizing training error

The problem of non-differentiability also occurs in settings far beyond combinatorial optimization. Consider for example the standard supervised learning setup. Inference using a K-way classifier $h : \mathcal{X} \to \mathbb{R}^K$ is typically per-

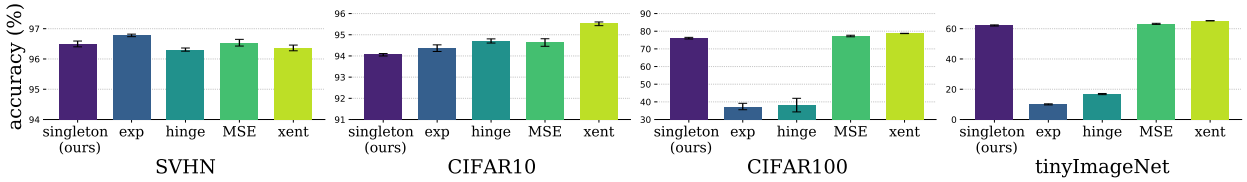

Figure 3: **Extending supervised training error:** we treat the training error as a non-differentiable loss function, and directly minimize the via the singleton set extension defined in Section 3.3.

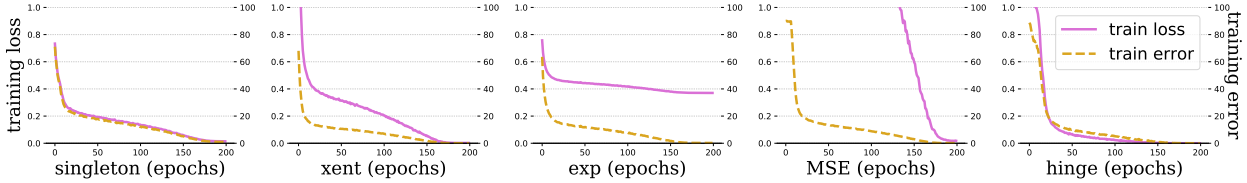

Figure 4: **CIFAR10:** Unlike common losses that indirectly optimize training error (e.g., cross-entropy), the singleton extension loss *closely approximates* the exact (non-differentiable) training error at the same numerical scale.

formed by returning the class with the largest score $\hat{y}(x) = \arg\max_{k=1,\dots,K} h(x)_k$, and an overall *training error* $\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i \neq \hat{y}(x_i)\}$ is calculated using a labeled training dataset $\{(x_i, y_i)\}_{i=1}^{n}$. The training error is non-differentiable since the error on a single sample is calculated using the set function $i \mapsto \mathbf{1}\{y \neq i\}$. Since training error cannot be directly minimized, it is standard practice to optimize a differentiable surrogate objective such as the cross-entropy loss.

Existing methods for indirectly optimizing training accuracy come with powerful motivations – e.g., via the maximum likelihood principle for cross-entropy. Our goal is therefore not to show an improvement over these widely used methods, or to argue that our framework is more principled. Instead, we aim to show that it is possible to derive a valid approach to supervised classification from the perspective of set function extension. To this end, we use the singleton extension example given in Section 3.3 as our extension of $i \mapsto \mathbf{1}\{y \neq i\}$.

We train ResNet-18 classifiers on several vision datasets, and compare the singleton extension loss to a number of standard supervised losses: cross-entropy, mean squared error, hinge, and exponential loss. We tune the learning rate for each loss using a simple exhaustive grid search over the interval $lr \in \{0.01, 0.05, 0.1, 0.2\}$ on a held out validation set. We report average test set performance over three random seeds.

Results. Figure 3 shows that the singleton extension loss performs comparably to the baselines, and is robust enough to be effective on $\{100, 200\}$-way classification tasks where the exponential and hinge losses fail. This demonstrates the practical viability of our extension approach for classification problems. The unique feature of the singleton loss is that since it is a direct extension of the empirical training error, its training loss closely tracks the training error (see Figure 4). While all training losses are, as expected, highly correlated with training error (all have Pearson correlation of more than 0.99) the singleton loss is the only loss able to directly track training error at the same numerical scale.

5 Conclusion

Although there has been significant focus on gradient estimation techniques in the literature, extending functions to continuous domains in the context of neural network training has remained relatively unexplored. We have presented "neural friendly extensions", a principled method for designing continuous and differentiable extensions of set functions. As we have verified experimentally, our framework offers viable alternatives when training neural networks with set functions in both classical settings such as image recognition, as well as combinatorial problems like k-clique detection and max-cut.

REFERENCES

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in Neural Information Processing Systems*, 32:9562–9574, 2019.

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

F. Bach. *Learning with Submodular Functions: A Convex Optimization Perspective*. Foundations and Trends in Machine Learning, 2013.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis R Bach. Learning with differentiable pertubed optimizers. In *NeurIPS*, 2020.

J. Bilmes. Deep mathematical properties of submodularity with applications to machine learning. Tutorial at the Conference on Neural Information Processing Systems (NIPS), 2013.

Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, pp. 950–959. PMLR, 2020.

Ravi Boppana and Magnús M Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, 32(2):180–196, 1992.

Endre Boros and Peter L Hammer. Pseudo-boolean optimization. *Discrete applied mathematics*, 123(1-3):155–225, 2002.

G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. *SIAM J. Computing*, 40(6), 2011.

Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 4348–4355. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/595. URL `https://doi.org/10.24963/ijcai.2021/595`. Survey Track.

Ching-An Cheng, Xinyan Yan, and Byron Boots. Trajectory-wise control variates for variance reduction in policy gradient methods. In *Conference on Robot Learning*, pp. 1379–1394. PMLR, 2020.

G. Choquet. Theory of capacities. *Annales de l'Institut Fourier*, 5:131–295, 1954.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid algorithm and its consequences in combinatorial optimization. *Combinatorica*, 1:499–513, 1981.

Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. In *International Conference on Learning Representations*, 2018.

S Gu, T Lillicrap, Z Ghahramani, RE Turner, and S Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings*, 2017.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL `https://www.gurobi.com`.

Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

Tamir Hazan and Tommi Jaakkola. On the partition function and random maximum a-posteriori perturbations. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 1667–1674, 2012.

Rishabh Iyer, Stefanie Jegelka, and Jeff Bilmes. Monotone closure of relaxed constraints in submodular optimization: Connections between minimization and maximization: Extended version. In *UAI*, 2014.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Int. Conf. on Learning Representations (ICLR)*, 2017.

Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In *NeurIPS*, 2020.

Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, pp. 3499–3508. PMLR, 2019.

A. Krause and S. Jegelka. Submodularity in Machine Learning: New directions. Tutorial at the International Conference on Machine Learning (ICML), 2013.

Vipin Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI magazine*, 13(1):32–32, 1992.

Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. Action-dependent control variates for policy optimization via stein identity. In *International Conference on Learning Representations*, 2018.

László Lovász. Submodular functions and convexity. In *Mathematical programming the state of the art*, pp. 235–257. Springer, 1983.

C Maddison, A Mnih, and Y Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Int. Conf. on Learning Representations (ICLR)*, 2017.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL `www.graphlearning.io`.

George Papandreou and Alan L Yuille. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *2011 International Conference on Computer Vision*, pp. 193–200. IEEE, 2011.

Anselm Paulus, Michal Rolinek, Vit Musil, Brandon Amos, and Georg Martius. Comboptnet: Fit the right np-hard problem by learning integer programming constraints. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8443–8453. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/paulus21a.html`.

Max Benedikt Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. Gradient estimation with stochastic softmax tricks. In *NeurIPS 2020*, 2020.

Marin Vlastelica Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2019.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Petar Veličković and Charles Blundell. Neural algorithmic reasoning. *Patterns*, 2(7):100273, 2021. ISSN 2666-3899.

Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pp. 6545–6554. PMLR, 2019.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *International Conference on Learning Representations*, 2018.

Sang Michael Xie and Stefano Ermon. Reparameterizable subset sampling via continuous relaxations. In *IJCAI*, 2019.

Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. *Advances in Neural Information Processing Systems*, 33, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

Li Yujia, Tarlow Daniel, Brockschmidt Marc, Zemel Richard, et al. Gated graph sequence neural networks. In *International Conference on Learning Representations*, 2016.

A SECTION 3 PROOFS

Proposition 3. Consider $f : \Omega \to \mathbb{R}$ such that $f(\varnothing) = 0$ and $\min_{S\in\Omega} f(S) < 0$. The following statements hold:

1. $\min_{\mathbf{x}\in\text{Hull}(\Omega)} \mathfrak{F}(\mathbf{x}) = \min_{S\in\Omega} f(S)$
2. $\arg\min_{\mathbf{x}\in\text{Hull}(\Omega)} \mathfrak{F}(\mathbf{x}) \subseteq \text{Hull}(\arg\min_{\mathbf{1}_S : S\in\Omega} f(S))$

The assumptions on f are mild. Every set function can be shifted (without changing its minima) to ensure $f(\varnothing) = 0$, in which case the non-negativity condition $\min_{S\in\Omega} f(S) < 0$ merely asserts that the minimization problem does not have the trivial solution $\varnothing$.

Proof. The inequality $\min_{\mathbf{x}\in\text{Hull}(\Omega)} \mathfrak{F}(\mathbf{x}) \leq \min_{S\in\Omega} f(S)$ automatically holds since $\Omega \subseteq \text{Hull}(\Omega)$, so it remains to show the reverse. To this end let $\mathbf{x} \in \text{Hull}(\Omega)$ be an arbitrary point. Then,

$$\begin{aligned}
\mathfrak{F}(\mathbf{x}) &= \mathbb{E}_{S\sim p_{\mathbf{x}}}[f(S)] \\
&= \sum_{S\in\Omega} p_{\mathbf{x}}(S) \cdot f(S) \\
&\geq \sum_{S\in\Omega} p_{\mathbf{x}}(S) \cdot \min_{S\in\Omega} f(S) \\
&= \min_{S\in\Omega} f(S)
\end{aligned}$$

where the inequality holds since we know that $\min_{S\in\Omega} f(S) \leq 0$, and the last equality simply uses the fact that $\sum_{S\in\Omega} p_{\mathbf{x}}(S)$. This proves the first claim.

To prove the second claim, assume that $\mathbf{x}$ minimizes $\mathfrak{F}(\mathbf{x})$ over $\mathbf{x} \in \text{Hull}(\Omega)$. This implies that the inequality in the above derivation must be tight, which is true if and only if

$$p_{\mathbf{x}}(S) \cdot f(S) = p_{\mathbf{x}}(S) \cdot \min_{\mathbf{x}\in\Omega} f(\mathbf{x}) \quad \text{for all } S \in \Omega.$$

This implies that either $p_{\mathbf{x}}(S) = 0$ or $f(S) = \min_{\mathbf{x}\in\Omega} f(\mathbf{x})$. Since $\mathbf{x} = \mathbb{E}_{p_{\mathbf{x}}}[\mathbf{1}_S] = \sum_{S\in\Omega} p_{\mathbf{x}}(S) \cdot \mathbf{1}_S \geq \sum_{S:p_{\mathbf{x}}(S)>0} p_{\mathbf{x}}(S) \cdot \mathbf{1}_S$. This is precisely a convex combination of points $\mathbf{1}_S$ for which $f(S) = \min_{S\in\Omega} f(S)$, proving the claim.

□

Lemma 1. If the mapping $\mathbf{x} \mapsto p_{\mathbf{x}}(S)$ is Lipschitz continuous and $f(S)$ is finite for all S, then $\mathfrak{F}$ is also Lipschitz continuous.

Proof. The Lipschitz continuity of $\mathfrak{F}(\mathbf{x})$ follows directly from definition:

$$\begin{aligned}
\big|\mathfrak{F}(\mathbf{x}) - \mathfrak{F}(\mathbf{x}')\big| &= \left| \sum_{S\in\Omega} p_{\mathbf{x}}(S) \cdot f(S) - \sum_{S\in\Omega} p_{\mathbf{x}'}(S) \cdot f(S) \right| \\
&= \left| \sum_{S\in\Omega} \big(p_{\mathbf{x}}(S) - p_{\mathbf{x}'}(S)\big) f(S) \right| \leq \left(2kL \max_{S\in\Omega} f(S) \right) \|\mathbf{x} - \mathbf{x}'\|,
\end{aligned}$$

where L is the maximum Lipschitz constant of $\mathbf{x} \mapsto p_{\mathbf{x}}(S)$ over any S, whereas k is the maximal size of the support of any $p_{\mathbf{x}}$. □

A.1 EXTENSION EXAMPLES

A.1.1 LOVÁSZ EXTENSION

Recall the definition: $\mathbf{x}$ is sorted so that $x_1 \geq x_2 \geq \ldots \geq x_d$. Then the Lovász extension corresponds to taking $S_i = \{1, \ldots, i\}$, and letting $p_{\mathbf{x}}(S_i) = x_i - x_{i+1}$, the non-negative increments of $\mathbf{x}$ (where recall we take $x_{d+1} = 0$). We show that the Lovász extension satisfies conditions P1-4. For convenience, we introduce the shorthand notation $a_i = p_{\mathbf{x}}(S_i) = x_i - x_{i+1}$

P1: since we assume $\mathbf{x}$ is sorted, it has the form $\mathbf{x} = (\underbrace{1, 1, \ldots, 1}_{i \text{ times}}, 0, 0, \ldots 0)^\top$. Therefore, for each $j < i$ we have $a_j = x_j - x_{j+1} = 1 - 1 = 0$ and for each $j > i$ we have $a_j = x_j - x_{j+1} = 0 - 0 = 0$. The only non-zero probability is $a_i = x_i - x_{i+1} = 1 - 0 = 1$. Finally, by definition S_i corresponds exactly to the vector $(\underbrace{1, 1, \ldots, 1}_{i \text{ times}}, 0, 0, \ldots 0)^\top = \mathbf{x}$, so we see that $p_{\mathbf{x}} = \delta_{\mathbf{x}}$ for $\mathbf{x} \in \Omega$.

P2: this follows directly from Proposition 1 once we confirm that the Lovász extension is a *proper* probabilistic extension, i.e., $\mathbf{x} = \mathbb{E}_{S \sim p_{\mathbf{x}}}[\mathbf{1}_S]$. Consider an x_i. Then every S_j with $j \geq i$ contains element i, and no S_j with $j < i$ contain i. So the ith coordianite of the expectation $\mathbb{E}_{S \sim p_{\mathbf{x}}}[\mathbf{1}_S]$ is precisely

$$\sum_{i=j}^{d} a_i = \sum_{i=j}^{d} (x_i - x_{i+1}) = x_i - x_{d+1} = x_i$$

which verifies that the Lovász extension is a proper probabilistic extension.

P3: By Proposition 1, our goal is to show that $\mathbf{x} \mapsto p_{\mathbf{x}}(S)$ is Lipschitz for all $S \in \Omega$.

Lemma 2. Let $p_{\mathbf{x}}$ be as defined for the Lovász extension. Then $\mathbf{x} \mapsto p_{\mathbf{x}}(S)$ is Lipschitz for all $S \in \Omega$.

Proof. First note that $p_{\mathbf{x}}$ is piecewise linear, with one piece per possible ordering $x_1 \geq x_2 \geq \ldots \geq x_d$ (so $d!$ pieces in total). Within the interior of each piece $p_{\mathbf{x}}$ is linear, and therefore Lipschitz. So in order to prove global Lipschitzness, it suffices to show that $p_{\mathbf{x}}$ is continuous at the boundaries between pieces (the Lipschitz constant is then the maximum of the Lipschitz constants for each linear piece).

Now consider a point $\mathbf{x}$ with $x_1 \geq \ldots \geq x_i = x_{i+1} \geq \ldots \geq x_d$. Consider the perturbed point $\mathbf{x}_\delta = \mathbf{x} - \delta \mathbf{e}_i$ with $\delta > 0$, and $\mathbf{e}_i$ denoting the ith standard basis vector. To prove continuity of $p_{\mathbf{x}}$ it suffices to show that for any $S \in \Omega$ we have $p_{\mathbf{x}_\delta}(S) \to p_{\mathbf{x}}(S)$ as $\delta \to 0^+$.

There are two sets in the support of $p_{\mathbf{x}}$ whose probabilities are different under $p_{\mathbf{x}_\delta}$, namely: $S_i = \{1, \ldots, i\}$ and $S_{i+1} = \{1, \ldots, i, i+1\}$. Similarly, there are two sets in the support of $p_{\mathbf{x}_\delta}$ whose probabilities are different under $p_{\mathbf{x}}$, namely: $S'_i = \{1, \ldots, i-1, i+1\}$ and $S'_{i+1} = \{1, \ldots, i, i+1\} = S_{i+1}$.

So it suffices to show the convergence $p_{\mathbf{x}_\delta}(S) \to p_{\mathbf{x}}(S)$ for these four S. Consider first S_i:

$$\left| p_{\mathbf{x}_\delta}(S_i) - p_{\mathbf{x}}(S_i) \right| = \left| 0 - (x_i - x_{i+1}) \right| = 0$$

where the final equality uses the fact that $x_i = x_{i+1}$. Next consider $S_{i+1} = S'_{i+1}$:

$$\left| p_{\mathbf{x}_\delta}(S_{i+1}) - p_{\mathbf{x}}(S_{i+1}) \right| = \left| (x'_{i+1} - x'_{i+2}) - (x_{i+1} - x_{i+2}) \right| = \left| (x'_{i+1} - x_{i+1}) - (x'_{i+2} - x_{i+2}) \right| = 0$$

Finally, we consider S'_i:

$$\begin{aligned}\left|p_{\mathbf{x}_\delta}(S_i') - p_{\mathbf{x}}(S_i')\right| &= \left|(x_i' - x_{i+1}') - (x_i - x_{i+1})\right| \\ &= \left|(x_{i+1}' - x_{i+1}) - (x_{i+1}' - x_{i+1})\right| \\ &= \left|(x_{i+1} - \delta - x_{i+1}) - (x_{i+1}' - x_{i+1})\right| \\ &= \delta \to 0\end{aligned}$$

completing the proof. □

P4: $p_{\mathbf{x}}$ defines a family of distributions, each of which is supported on d sets. So evaluation of $\mathfrak{F}$, and recovery of integral solutions, requires d evaluations of f.

A.1.2 SINGLETON SET FUNCTION EXTENSION

Recall the setup. We consider $f : \Omega \to \mathbb{R}$ with $\Omega = \{E_1, \ldots, E_d\}$ where $E_i = \{i\}$. To define $p_{\mathbf{x}}(S)$, assume that $\mathbf{x}$ is sorted so that $x_1 \geq x_2 \geq \ldots \geq x_d$. For $1 \leq i \leq d$ let $S_i = E_i = \{i\}$. As with the Lovász extension, let $p_{\mathbf{x}}(S_i) = x_i - x_{i+1}$. Note that in this case, $\mathbf{x}$ will range over the convex hull of Ω, which is the d-dimensional simplex. Again we check P1-4.

P1, P3, and **P4** all follow by the same argument as the Lovász extension.

P2: The ith coordinate of $\mathbb{E}_{S\sim p_{\mathbf{x}}}[\mathbf{1}_S]$ is $p_{\mathbf{x}}(E_i) \cdot \mathbf{1}_{E_i} = x_i - x_{i+1}$ which is not equal to x_i in general. Therefore $p_{\mathbf{x}}$ defines an *improper* probabilistic extension, and the proof of Proposition 1 therefore fails. However, in this case, it turns out that we can give an alternative proof that Proposition 1 is true, and therefore P2 (minimality) still holds in this case despite being an improper probabilistic extension. To see this, consider the same assumptions as Proposition 1. For $\mathbf{x} \in \text{Hull}(\Omega)$,

$$\begin{aligned}\mathfrak{F}(\mathbf{x}) &= \sum_{i=1}^{d} p_{\mathbf{x}}(E_i) f(E_i) \\ &= \sum_{i=1}^{d} (x_i - x_{i+1}) f(E_i) \\ &\geq \sum_{i=1}^{d} (x_i - x_{i+1}) \min_{j\in[d]} f(E_j) \\ &\geq (x_1 - x_{d+1}) \min_{j\in[d]} f(E_j) \\ &\geq x_1 \cdot \min_{j\in[d]} f(E_j) \\ &\geq \min_{j\in[d]} f(E_j)\end{aligned}$$

where the final inequality follows since $\min_{j\in[d]} f(\mathbf{e}_j) < 0$. Taking $\mathbf{x} = (1, 0, 0, \ldots, 0)^\top$ shows that inequalities can be made tight, and the first statement of Proposition 1 holds. For the second statement, suppose that $\mathbf{x} \in \text{Hull}(\Omega)$ minimizes $\mathfrak{F}$. Then all the inequality in the preceding argument must be tight. In particular, tightness of the final inequality implies that x_1, and since $\mathbf{x} \in \text{Hull}(\Omega) = \{\mathbf{x} : x_i \in [0, 1], \sum_i x_i = 1\}$, we conclude that $x_i = 0$ for all $i > 1$. In this case, since the first inequality must also be tight, hence

$$f(E_1) = \sum_{i=1}^{d} (x_i - x_{i+1}) f(E_i) = \sum_{i=1}^{d} (x_i - x_{i+1}) \min_{j\in[d]} f(E_j) = \min_{j\in[d]} f(E_j)$$

and so we conclude that $\mathbf{x} = (1, 0, 0, \ldots, 0)^\top$ belongs to $\arg\min_{j \in [d]} f(E_j)$, finishing the proof.

Constructing proper NFEs from improper ones

Here we prove Proposition 2, which shows how to take an improper NFE and construct a proper NFE that approximates it. We restate the proposition for completeness.

Proposition 4. Let $p_{\mathbf{x}}$ be a distribution for which $\mathbf{x}' \coloneqq \mathbb{E}_{S \sim p_{\mathbf{x}}}[\mathbf{1}_S] \neq \mathbf{x}$ and define $w \coloneqq \min\left(\{x_i / x'_i\}_{i=1}^d \cup \{1\}\right)$. There exists a perturbed distribution $p'_{\mathbf{x}}(S, z)$ with $p'_{\mathbf{x}}(S, 1) = p_{\mathbf{x}}(S)$ such that

$$\mathfrak{F}'(\mathbf{x}) \coloneqq \mathbb{E}_{(S,z) \sim p'_{\mathbf{x}}}\big[f(S, z)\big], \quad \text{with} \quad f(S, z) \coloneqq \begin{cases} f(S) & \text{if } z = 1 \\ \beta & \text{if } z = 0, \end{cases} \quad \text{and} \quad z \sim \text{Bernoulli}(w)$$

is a proper NFE.

Proof. Our goal is to construct a new distribution $p'_{\mathbf{x}}$ that is 1) similar to $p_{\mathbf{x}}$, and 2) yields a proper NFE. We search for a $p'_{\mathbf{x}}$ of the form:

$$p'_{\mathbf{x}}(S, z) = p_{\mathbf{x}}(S) \cdot \mathbf{1}\{z = 1\} + q_{\mathbf{x},w}(S) \cdot \mathbf{1}\{z = 0\}$$

where $z \sim \text{Bernoulli}(w)$ with parameter $w \in [0, 1]$, and $q_{\mathbf{x},w}$ is a distribution of our choice. Our aim is to choose w to be as large as possible so that $p'_{\mathbf{x}}$ approximates $p_{\mathbf{x}}$ well. Recall that once we have chosen $p'_{\mathbf{x}}$, we replace the improper extension $\mathfrak{F}$ with the following proper extension

$$\mathfrak{F}'(\mathbf{x}) \coloneqq \mathbb{E}_{(S,z) \sim p'_{\mathbf{x}}}\big[f(S, z)\big] \quad \text{with} \quad f(S, z) \coloneqq \begin{cases} f(S) & \text{if } b = 1, \\ \beta & \text{if } b = 0. \end{cases}$$

where β is a penalty applied when S is sampled from the proper distribution $q_{\mathbf{x},w}$.

First we choose $q_{\mathbf{x},w}$. We want $\mathbf{x} = \mathbb{E}_{S \sim p'_{\mathbf{x}}}[\mathbf{1}_S] = w\mathbb{E}_{S \sim p_{\mathbf{x}}}[\mathbf{1}_S] + (1-w)\mathbb{E}_{S \sim q_{\mathbf{x},w}}[\mathbf{1}_S] = w\mathbf{x}' + (1-w)\mathbb{E}_{S \sim q_{\mathbf{x},w}}[\mathbf{1}_S]$ which is satisfied if we select $q_{\mathbf{x},w}$ such that $\mathbb{E}_{S \sim q_{\mathbf{x},w}}[\mathbf{1}_S] = (\mathbf{x} - w\mathbf{x}')/(1 - w)$. Any $q_{\mathbf{x},w}$ with this property will do, so take $q_{\mathbf{x},w}$ to be the distribution implied by the Lovász extension of f at $(\mathbf{x} - w\mathbf{x}')/(1 - w)$. Finally, we wish to choose w to be as large as possible. Noting that the only constraints we must respect are that $w \in [0, 1]$ and $[(\mathbf{x} - w\mathbf{x}')/(1 - w)]_i \geq 0$ for each coordinate i, we may fix $w = \max_{t \in [0,1]} t$ subject to $[\mathbf{x} - w\mathbf{x}']_i \geq 0$ for all i. The latter optimization problem admits the closed-form solution the minimum between $\min_i\{x_i / x'_i\}$ and 1. □

B Experimental configurations

B.1 Unsupervised combinatorial optimization

Discrete Objectives. For the maximum cut objective, since we are training with gradient descent, we minimized

$$f_{cut}(S; G) = -\text{cut}(S; G), \tag{2}$$

where S is an input set. It should be noted that the Lovaśz extension of the cut function on a graph, corresponds to the graph total variation. For the maximum clique problem, we select the discrete objective that yielded the most stable results across datasets. It is defined as

$$f_{clique}(S; G) = -w(S)q^2(S), \tag{3}$$

where $q(S) = 2w(S)/(|S|^2 - |S|))$ and $w(S) = \sum_{(i,j) \in E} w_{ij} p_i p_j$.

Baselines. We compare with recent work on unsupervised combinatorial optimization Karalias & Loukas (2020). We use the official implementation of the code and the corresponding loss function for the maximum clique problem.

For the maximum cut problem, we use the probabilistic methodology described in the paper to derive a loss function. The resulting loss for an input graph G and learned probabilities $\mathbf{p}$ is

$$L_{\text{maxcut}}(\mathbf{p}; G) = a - \mathbb{E}[\text{cut(S)}] = a - \sum_{i \in V} p_i d_i + \sum_{(i,j) \in E} w_{ij} p_i p_j, \tag{4}$$

where a is an upper bound on the size of the optimal solution, which we omit in practice as it does not affect the optimization. In the original implementation, the code makes use of randomization in the input of the neural network. Specifically, each set of probabilities $\mathbf{p} \in [0,1]^{|V|}$ is computed as $f(\delta_i; G) = \mathbf{p}$, where $\delta_i \in [0,1]^{|V|}$ is a one hot encoding of a randomly chosen node i. This allows for different sets of probabilities conditioned on different inputs to be produced for each graph. Then each set is decoded with the method of conditional expectation into discrete outputs. In order to ensure accurate comparisons, we only apply the method of conditional expectation on just one set of probabilities produced from one randomly generated input. Our implementation of the Lovaśz extension just picks the best level set in the support of $p_{\mathbf{x}}$ so this establishes a level playing field for the comparisons.

We also compared with the Straight-Through gradient estimator (Bengio et al., 2013). This estimator can be used to pass gradients through samples from a distribution, by assuming that the sampling/thresholding operation is the identity. Initially, we observed that by sampling sets from a product distribution on the outputs of a neural network was not able to minimize the loss function. In order to obtain a working baseline with the straight-through estimator, we restricted our attention to the distribution of level sets induced by the output of a neural network. Specifically, given $\mathbf{x} \in [0,1]^{|V|}$ outputs from a neural network, we sample indicator vectors $\mathbf{1}_{S_k}$, where $S_k = \{x_j | x_j \geq t_k\}$ where $t_k \in [0,1]$ is a randomly chosen threshold. Then our loss function was computed as

$$L(\mathbf{x}; G) = \frac{1}{K} \sum_{k=1}^{K} f(\mathbf{1}_{S_k}), \tag{5}$$

where f was the clique or the cut functions used in the Lovaśz setting, and K the total number of samples, which we set to 100. This can be viewed as a differentiable stochastic estimate of the Lovasz extension. It should be noted that this use of the Straight-Through estimator is bound to fail for a general non-differentiable black box f, as no gradients will be able to flow in the backward pass. Furthermore, we have also experimented with different variants of the score function estimator. However, it demonstrated extremely unstable behavior during training and was unable to converge. As it can be seen in figures 6 and 7, our extension minimizes the loss easily across epochs, but the rewards remain unstable during training for the estimator.

We also reported results for greedy heuristics on both combinatorial problems. These are both heuristics provided with the NetworkX package (Hagberg et al., 2008). For the max-cut, the heuristic begins with an empty set and adds at each step the node that maximally increases the cut. For the max-clique heuristic, the implementation relies on an approximate indepedendent set heuristic that utilizes subgraph exclusion (Boppana & Halldórsson, 1992).

Ground truths. We obtain ground truths for all our datasets and for each combinatorial problem by expressing it as a mixed integer program and solving it in Gurobi (Gurobi Optimization, LLC, 2021).

Implementation details and datasets. We conduct our experiments using the Pytorch Geometric API (PyG) (Fey & Lenssen, 2019). The datasets PROTEINS, ENZYMES, and IMDB-Binary are part of the TUDatasets collection of graph benchmarks (Morris et al., 2020) available through the PyG API. For our models, we use as input features the node degrees, followed by a Graph Isomorphism Network (GIN) layer (Xu et al., 2018) and multiple iterations of the Gated Graph Convolution layer (Yujia et al., 2016). We tune the width (64-256), the iterations (3-12), the learning rate (0.00001,0.001) and the batch size (4,128) of our models with a grid search over fixed values in the reported intervals. We train for 500 epochs and report the test performance of the model with the best validation score.

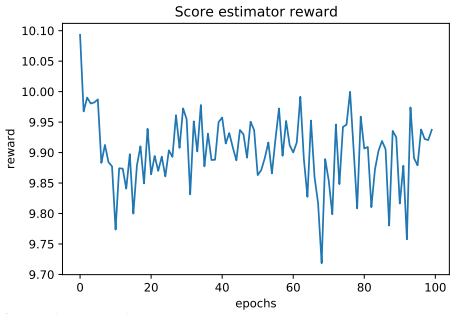

Figure 5: Score function estimator reward over 100 epochs on maximum clique.

Figure 6: Lovaśz extension loss over 100 epochs.

B.2 Constraint satisfaction problems

Ground truths. As before, we obtain ground truths for all our datasets and problems by expressing it as a mixed integer program and using the Gurobi solver (Gurobi Optimization, LLC, 2021).

Implementation details and datasets. Again we use the Pytorch Geometric API (PyG) (Fey & Lenssen, 2019) and access the datasets through TUDatasets collection of graph benchmarks (Morris et al., 2020) available through the PyG API. For the model architecture we use a single GAT layer (Veličković et al., 2018) followed by multiple gated graph convolution layers to extract features $\mathbf{x} = g(G) \in [0,1]^{|V|}$, where $\mathbf{x}$ contains a single value in $[0,1]$ for each node. We use the following same set function as we did for max-clique,

$$f_{clique}(S;G) = -w(S)q^2(S).$$

For both the Lovász extension and k-subset extension, we perform an exhaustive grid search over the following hyper parameters: learning rate $lr = \{0.00001, 0.00005, 0.0001\}$, GNN widths $w \in \{64, 128\}$, batch size $bs \in \{32, 64, 128\}$, and depths $d \in \{8, 10\}$. Finally, we use the correction technique for improper NFEs described in Section 3.4, in which we optimize a penalization of the improper extension:

$$w \cdot \mathfrak{F}(\mathbf{x}) + (1-w) \cdot \beta$$

where recall that $\mathbf{x}' \coloneqq \mathbb{E}_{S\sim p_{\mathbf{x}}}[\mathbf{1}_S] \neq \mathbf{x}$ and we define $w \coloneqq \min\left(\{x_i/x'_i\}_{i=1}^d \cup \{1\}\right)$. We also tune the penalty β during HPO, searching over the two values $\beta \in \{10, 15\}$.

After completing the exhaustive grid search, we select the single model that achieves the highest F1-score on a validation set, and report as the final result the F1-score of that same model on a separate test set.

B.3 DIRECTLY MINIMIZING TRAINING ERROR

Recall that for a K-way classifier $h : \mathcal{X} \to \mathbb{R}^K$ with $\hat{y}(x) = \arg\max_{k=1,\dots,K} h(x)_k$, we consider the training error

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i \neq \hat{y}(x_i)\}$$

calculated over a labeled training dataset $\{(x_i, y_i)\}_{i=1}^n$ to be a discrete non-differentiable loss. The set function in question is $y \mapsto \mathbf{1}\{y_i \neq y\}$, which we relax using the singleton method described in Section 3.3.

Training details. For all datasests we use a standard ResNet-18 backbone, with a final layer to output a vector of the correct dimension. All models are trained for 200 epochs, except for on SVHN, which we uses only 100. We use SGD with momentum $mom = 0.9$ and weight decay $wd = 5 \times 10^{-4}$ and a cosine learning rate schedule. We tune the learning rate for each loss via a simple grid search of the values $lr \in \{0.01, 0.05, 0.1, 0.2\}$. For each loss we select the learning rate with highest accuracy on a validation set, then report the average accuracy on a test set over three separate runs.